

Documented Code For glossaries v4.27

Nicola L.C. Talbot

Dickimaw Books

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

2016-12-16

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 v4.27: L^AT_EX2_ε Package to Assist Generating Glossaries”.

[mfirstuc-manual.pdf](#) The commands provided by the mfirstuc package are briefly described in “mfirstuc.sty: uppercasing first letter”.

[glossaries-code.pdf](#) This document is for advanced users wishing to know more about the inner workings of the glossaries package.

INSTALL Installation instructions.

CHANGES Change log.

README Package summary.

The user level commands described in the user manual ([glossaries-user.pdf](#)) may be considered “future-proof”. Even if they become deprecated, they should still work for old documents (although they may not work in a document that also contains new commands introduced since the old commands were deprecated, and you may need to specify a compatibility mode).

The internal commands in *this* document that aren’t documented in the *user manual* should not be considered future-proof and are liable to change. If you want a new user level command, you can post a feature request at <http://www.dickimaw-books.com/feature-request.html>. If you are a package writer wanting to integrate your package with glossaries, it’s better to request a new user level command than to hack these internals.

Contents

1	Main Package Code	4
1.1	Package Definition	4
1.2	Package Options	5
1.3	Predefined Text	30
1.4	Xindy	40
1.5	Loops and conditionals	49
1.6	Defining new glossaries	55
1.7	Defining new entries	60
1.8	Resetting and unsetting entry flags	85
1.9	Keeping Track of How Many Times an Entry Has Been Unset	88
1.10	Loading files containing glossary entries	93
1.11	Using glossary entries in the text	93
1.12	Adding an entry to the glossary without generating text	153
1.13	Creating associated files	155
1.14	Writing information to associated files	173
1.15	Glossary Entry Cross-References	180
1.16	Displaying the glossary	182
1.17	Acronyms	211
1.18	Predefined acronym styles	215
1.19	Predefined Glossary Styles	247
1.20	Debugging Commands	247
1.21	Compatibility with version 2.07 and below	253
2	Prefix Support (glossaries-prefix Code)	254
3	Glossary Styles	261
3.1	Glossary hyper-navigation definitions (glossary-hypernav package)	261
3.2	In-line Style (glossary-inline.sty)	263
3.3	List Style (glossary-list.sty)	265
3.4	Glossary Styles using longtable (the glossary-long package)	269
3.5	Glossary Styles using longtable and booktabs (the glossary-longbooktabs) package	275
3.6	Glossary Styles using longtable (the glossary-longragged package)	279
3.7	Glossary Styles using multicol (glossary-mcols.sty)	285
3.8	Glossary Styles using supertabular environment (glossary-super package)	291
3.9	Glossary Styles using supertabular environment (glossary-superragged package)	297
3.10	Tree Styles (glossary-tree.sty)	303

4 Backwards Compatibility	313
4.1 glossaries-compatible-207	313
4.2 glossaries-compatible-307	319
5 Accessibility Support (glossaries-accsupp Code)	333
5.1 Defining Replacement Text	334
5.2 Accessing Replacement Text	337
5.3 Displaying the Glossary	353
5.4 Acronyms	354
5.5 Debugging Commands	369
6 Multi-Lingual Support	371
6.1 Polyglossia Captions	371
Glossary	373
Change History	374
Index	397

1 Main Package Code

1.1 Package Definition

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

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries}[2016/12/16 v4.27 (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 {'see' 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 'see' 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       {'see' 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 'see' key}%
62     }%
63   \or
64     \renewcommand*{\@gls@see@noindex}{%
65       \GlossariesWarning{'see' 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 \glsglossarymark

```
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}{%
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 }

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

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

s@setupsort@def  Set up the macros for order of definition sorting.
194 \newcommand*{\@gls@setupsort@def}{%
  Store entry information when it's defined.
195   \def\do@glo@storeentry{\@glo@storeentry}%

```

Defined count register associated with the glossary.

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

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

```
200 \def\@gls@defsort##1##2{%
201   \expandafter\global\expandafter
202   \advance\csname glossary@##1@sortcount\endcsname by 1\relax
203   \expandafter\protected@xdef\csname glo@##2@sort\endcsname{%
204     \expandafter\glssortnumberfmt
205     {\csname glossary@##1@sortcount\endcsname}}%
206 }%
```

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

```
207 \def\@gls@setsort##1{%
208 }
```

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

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

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

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

Defined count register associated with the glossary.

```
211 \def\@gls@defsortcount##1{%
212   \expandafter\global
213   \expandafter\newcount\csname glossary@##1@sortcount\endcsname
214 }%
```

Initialise the sort key to empty.

```
215 \def\@gls@defsort##1##2{%
216   \expandafter\gdef\csname glo@##2@sort\endcsname{%
217 }%
```

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

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

Get the parent, if one exists

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

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

```
220   \ifx\@glo@parent\@empty
221   \else
222     \expandafter\@gls@setsort\expandafter{\@glo@parent}%
223   \fi
```

Set index information for this entry

```
224   \edef\@glo@type{\csname glo@##1@type\endcsname}%
225   \edef\@gls@tmp{\csname glo@##1@sort\endcsname}%
```

```

226 \ifx\@gls@tmp\@empty
227 \expandafter\global\expandafter
228 \advance\csname glossary@\@glo@type @sortcount\endcsname by 1\relax
229 \expandafter\protected\xdef\csname glo@##1@sort\endcsname{%
230 \expandafter\glssortnumberfmt
231 {\csname glossary@\@glo@type @sortcount\endcsname}}}%
232 \@glo@storeentry{##1}%
233 \fi
234 }%
235 }

```

`\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.)

```

236 \newcommand*{\glsdefmain}{%
237 \if@gls@docloaded
238 \newglossary[glg2]{main}{gls2}{glo2}{\glossaryname}%
239 \else
240 \newglossary{main}{gls}{glo}{\glossaryname}%
241 \fi

```

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

```

242 \newcommand*{\gls@tr@set@main@toctitle}{%
243 \translatelet{\glossarytoctitle}{Glossary}%
244 }%
245 }

```

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`

```

246 \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`

```

247 \newcommand*{\acronymtype}{\glsdefaulttype}

```

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

```

248 \@gls@declareoption{nomain}{%
249 \let\glsdefaulttype\relax
250 \renewcommand*{\glsdefmain}{}%
251 }

```

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.

```

252 \define@boolkey{glossaries.sty}{gls}{acronym}[true]{%
253   \ifglsacronym
254     \renewcommand{\@gls@do@acronymsdef}{%
255       \DeclareAcronymList{acronym}%
256       \newglossary[alg]{acronym}{acr}{acn}{\acronymname}%
257       \renewcommand*\@acronymtype{acronym}%

```

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

```

258     \newcommand*\@gls@tr@set@acronym@toctitle{%
259       \translatelet{\glossarytoctitle}{Acronyms}%
260     }%
261   }%
262 \else
263   \let\@gls@do@acronymsdef\relax
264 \fi
265 }

```

\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.

```

266 \AtBeginDocument{%
267   \ifglsacronym
268     \ifbool{glscompatible-3.07}{%
269       {}%
270     }%
271     \providecommand*\printacronyms[1][]{%
272       \printglossary[type=\acronymtype,#1]}%
273   }%
274 \fi
275 }

```

@do@acronymsdef Set default value

```

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

```

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

```

277 \@gls@declareoption{acronyms}{%
278   \glsacronymtrue
279   \renewcommand{\@gls@do@acronymsdef}{%
280     \DeclareAcronymList{acronym}%
281     \newglossary[alg]{acronym}{acr}{acn}{\acronymname}%
282     \renewcommand*\@acronymtype{acronym}%

```

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

```

283     \newcommand*\@gls@tr@set@acronym@toctitle{%
284       \translatelet{\glossarytoctitle}{Acronyms}%
285     }%
286   }%
287 }

```

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

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

`addtoacronymlists`

```
289 \newcommand*{\@addtoacronymlists}[1]{%
290   \ifx\@glsacronymlists\@empty
291     \protected@xdef\@glsacronymlists{#1}%
292   \else
293     \protected@xdef\@glsacronymlists{\@glsacronymlists,#1}%
294   \fi
295 }
```

`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.)

```
296 \newcommand*{\DeclareAcronymList}[1]{%
297   \glsIfListOfAcronyms{#1}{\@addtoacronymlists{#1}}%
298 }
```

`IfListOfAcronyms`

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

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

```
299 \newcommand{\glsIfListOfAcronyms}[1]{%
300   \edef\@do@gls@islistofacronyms{%
301     \noexpand\@gls@islistofacronyms{#1}{\@glsacronymlists}}%
302   \@do@gls@islistofacronyms
303 }
```

Internal command requires label and list to be expanded:

```
304 \newcommand{\@gls@islistofacronyms}[4]{%
305   \def\gls@islistofacronyms##1,#1,##2\end@gls@islistofacronyms{%
306     \def\@before{##1}\def\@after{##2}}%
307   \gls@islistofacronyms,#2,#1,\@nil\end@gls@islistofacronyms
308   \ifx\@after\@nnil
```

Not found

```
309   #4%
310   \else
```

Found

```
311   #3%
312   \fi
313 }
```

`lsglsacronymlist` Convenient boolean.

```
314 \newif\if@lsglsacronymlist
```

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

```
315 \newcommand*{\gls@ckisacronymlist}[1]{%
316   \glsIfListOfAcronyms{#1}%
317   {\@glsisacronymlisttrue}{\@glsisacronymlistfalse}%
318 }
```

`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.)

```
319 \newcommand*{\SetAcronymLists}[1]{%
320   \renewcommand*{\@glsacronymlists}{#1}%
321 }
```

`acronymlists`

```
322 \define@key{glossaries.sty}{acronymlists}{%
323   \DeclareAcronymList{#1}%
324 }
```

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`

```
325 \newcommand{\glscounter}{page}
```

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

```
326 \define@key{glossaries.sty}{counter}{%
327   \renewcommand*{\glscounter}{#1}%
328 }
```

`gls@nohyperlist`

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

`lareNoHyperList`

```
330 \newcommand*{\GlsDeclareNoHyperList}[1]{%
331   \ifdefempty\@gls@nohyperlist
332   {%
333     \renewcommand*{\@gls@nohyperlist}{#1}%
334   }%
335   {%
336     \appto\@gls@nohyperlist{,#1}%
337   }%
338 }
```

`nohypertypes`

```
339 \define@key{glossaries.sty}{nohypertypes}{%
340   \GlsDeclareNoHyperList{#1}%
341 }
```


glossariesWarning Prints a warning message.

```
342 \newcommand*\GlossariesWarning}[1]{%
343   \PackageWarning{glossaries}{#1}%
344 }
```

glossariesWarningNoLine Prints a warning message without the line number.

```
345 \newcommand*\GlossariesWarningNoLine}[1]{%
346   \PackageWarningNoLine{glossaries}{#1}%
347 }
```

nowarn Define package option to suppress warnings

```
348 \@gls@declareoption{nowarn}{%
349   \if@gls@debug
350     \GlossariesWarning{Warnings can't be suppressed in debug mode}%
351   \else
352     \renewcommand*\GlossariesWarning}[1]{}%
353     \renewcommand*\GlossariesWarningNoLine}[1]{}%
354   \fi
355 }
```

warnonglossdefined Issue a warning if overriding \printglossary

```
356 \newcommand*\@gls@warnonglossdefined{%
357   \GlossariesWarning{Overriding \string\printglossary}%
358 }
```

warntheglossdefined Issue a warning if overriding theglossary

```
359 \newcommand*\@gls@warnontheglossdefined{%
360   \GlossariesWarning{Overriding 'theglossary' environment}%
361 }
```

noredefwarn Suppress warning on redefinition of \printglossary

```
362 \@gls@declareoption{noredefwarn}{%
363   \renewcommand*\@gls@warnonglossdefined}{}%
364   \renewcommand*\@gls@warnontheglossdefined}{}%
365 }
```

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

```
366 \newcommand*\@gls@sanitizedesc{%
367 }
```

lssetexpandfield `\glssetexpandfield{<field>}`

Sets field to always expand.

```

368 \newcommand*{\glsetexpandfield}[1]{%
369   \csdef{gls@assign@#1@field}##1##2{%
370     \@@gls@expand@field{##1}{#1}{##2}%
371   }%
372 }

```

setnoexpandfield `\glsetnoexpandfield{<field>}`

Sets field to never expand.

```

373 \newcommand*{\glsetnoexpandfield}[1]{%
374   \csdef{gls@assign@#1@field}##1##2{%
375     \@@gls@noexpand@field{##1}{#1}{##2}%
376   }%
377 }

```

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

```
378 \glsetexpandfield{type}
```

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

```
379 \glsetnoexpandfield{desc}
```

escplural@field

```
380 \glsetnoexpandfield{descplural}
```

ls@sanitizename

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

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

```
382 \glsetnoexpandfield{name}
```

@sanitizesymbol

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

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

```
384 \glsetnoexpandfield{symbol}
```

bolplural@field

```
385 \glsetnoexpandfield{symbolplural}
```

Sanitizing stuff:

ls@sanitizesort

```

386 \newcommand*{\@gls@sanitizesort}{%
387   \ifglssanitizesort
388     \@@gls@sanitizesort
389   \else
390     \@@gls@nosanitizesort
391   \fi
392 }

```

ls@sanitizesort

```
393 \newcommand*\@@gls@sanitizesort{%
394   \@onelevel@sanitize\@glo@sort
395 }
```

@nosanitizesort

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

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

```
397 \newcommand*\@gls@noidx@sanitizesort{%
398   \ifdefvoid\@glo@sort
399   {}%
400   {%
401     \expandafter\@@gls@noidx@sanitizesort\@glo@sort\gls@end@sanitizesort
402   }%
403 }
404 \def\@@gls@noidx@sanitizesort#1#2\gls@end@sanitizesort{%
405   \def\@glo@sort{#1#2}%
406   \@onelevel@sanitize\@glo@sort
407 }
```

@nosanitizesort

```
408 \newcommand*\@@gls@noidx@nosanitizesort{%
409   \ifdefvoid\@glo@sort
410   {}%
411   {%
412     \expandafter\@@gls@noidx@no@sanitizesort\@glo@sort\gls@end@sanitizesort
413   }%
414 }
415 \def\@@gls@noidx@no@sanitizesort#1#2\gls@end@sanitizesort{%
416   \bgroup
417   \glsnoidxstripaccents
418   \protected@xdef\@glo@sort{#1#2}%
419   \egroup
420   \let\@glo@sort\@glo@sort
421 }
```

idxstripaccents

```
422 \newcommand*\glsnoidxstripaccents{%
423   \let\IeC\@firstofone
424   \let\' \@firstofone
425   \let\' \@firstofone
426   \let\~ \@firstofone
427   \let\" \@firstofone
428   \let\u \@firstofone
429   \let\t \@firstofone
430   \let\d \@firstofone
431   \let\r \@firstofone
432   \let\= \@firstofone
```

```

433 \let\.\@firstofone
434 \let\~\@firstofone
435 \let\v\@firstofone
436 \let\H\@firstofone
437 \let\c\@firstofone
438 \let\b\@firstofone
439 \def\AE{AE}%
440 \def\ae{ae}%
441 \def\OE{OE}%
442 \def\oe{oe}%
443 \def\AA{AA}%
444 \def\aa{aa}%
445 \def\L{L}%
446 \def\l{l}%
447 \def\O{O}%
448 \def\o{o}%
449 \def\SS{SS}%
450 \def\ss{ss}%
451 \def\th{th}%
452 }

```

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.

```

453 \define@boolkey[glS]{sanitize}{description}[true]{%
454   \GlossariesWarning{sanitize={description} package option deprecated}%
455   \ifglS@sanitize@description
456     \glSsetnoexpandfield{desc}%
457     \glSsetnoexpandfield{descplural}%
458   \else
459     \glSsetexpandfield{desc}%
460     \glSsetexpandfield{descplural}%
461   \fi
462 }

463 \define@boolkey[glS]{sanitize}{name}[true]{%
464   \GlossariesWarning{sanitize={name} package option deprecated}%
465   \ifglS@sanitize@name
466     \glSsetnoexpandfield{name}%
467   \else
468     \glSsetexpandfield{name}%
469   \fi
470 }

471 \define@boolkey[glS]{sanitize}{symbol}[true]{%
472   \GlossariesWarning{sanitize={symbol} package option deprecated}%
473   \ifglS@sanitize@symbol
474     \glSsetnoexpandfield{symbol}%
475     \glSsetnoexpandfield{symbolplural}%
476   \else
477     \glSsetexpandfield{symbol}%

```

```

478 \glsssetexpandfield{symbolplural}%
479 \fi
480 }

```

sanitizesort

```

481 \define@boolkey{glossaries.sty}[gls]{sanitizesort}[true]{%
482 \ifglsssanitizesort
483 \glsssetnoexpandfield{sortvalue}%
484 \renewcommand*{\@gls@noidx@setsanitizesort}{%
485 \glsssanitizesorttrue
486 \glsssetnoexpandfield{sortvalue}%
487 }%
488 \else
489 \glsssetexpandfield{sortvalue}%
490 \renewcommand*{\@gls@noidx@setsanitizesort}{%
491 \glsssanitizesortfalse
492 \glsssetexpandfield{sortvalue}%
493 }%
494 \fi
495 }

```

Default setting:

```

496 \glsssanitizesorttrue
497 \glsssetnoexpandfield{sortvalue}%

```

setsanitizesort Default behaviour for \makenoidxglossaries is sanitizesort=false.

```

498 \newcommand*{\@gls@noidx@setsanitizesort}{%
499 \glsssanitizesortfalse
500 \glsssetexpandfield{sortvalue}%
501 }

502 \define@choicekey[gls]{sanitize}{sort}{true,false}[true]{%
503 \setbool{glsssanitizesort}{#1}%
504 \ifglsssanitizesort
505 \glsssetnoexpandfield{sortvalue}%
506 \else
507 \glsssetexpandfield{sortvalue}%
508 \fi
509 \GlossariesWarning{sanitize={sort} package option
510 deprecated. Use sanitizesort instead}%
511 }

```

sanitize

```

512 \define@key{glossaries.sty}{sanitize}[description=true,symbol=true,name=true]{%
513 \ifthenelse{\equal{#1}{none}}{%
514 {%
515 \GlossariesWarning{sanitize package option deprecated}%
516 \glsssetexpandfield{name}%
517 \glsssetexpandfield{symbol}%
518 \glsssetexpandfield{symbolplural}%

```

```

519     \glssetexpandfield{desc}%
520     \glssetexpandfield{descplural}%
521 }%
522 {%
523     \setkeys[gls]{sanitize}{#1}%
524 }%
525 }

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

notranslatorhook \@gls@notranslatorhook has been removed.

s@usetranslator
527 \newcommand*\@gls@usetranslator{%
    polyglossia tricks \@ifpackageloaded into thinking that babel has been loaded, so check for
    polyglossia as well.
528     \@ifpackageloaded{polyglossia}%
529     {%
530         \let\glsifusetranslator\@secondoftwo
531     }%
532     {%
533         \@ifpackageloaded{babel}%
534         {%
535             \IfFileExists{translator.sty}%
536             {%
537                 \RequirePackage{translator}%
538                 \let\glsifusetranslator\@firstoftwo
539             }%
540             {}%
541         }%
542     }%
543 }%
544 }

dtranslatordict Checks if given translator dictionary has been loaded.
545 \newcommand{\glsifusedtranslatordict}[3]{%
546     \glsifusetranslator
547     {\ifcsdef{ver@glossaries-dictionary-#1.dict}{#2}{#3}}%
548     {#3}%
549 }

notranslate Provide a synonym for translate=false that can be passed via the document class.
550 \@gls@declareoption{notranslate}{%
551     \glstranslatefalse
552     \let\@gls@usetranslator\relax
553     \let\glsifusetranslator\@secondoftwo
554 }

```

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

```

555 \define@choicekey{glossaries.sty}{translate}[\val\nr]%
556   {true,false,babel}[true]%
557   {%
558     \ifcase\nr\relax
559     \glstranslatetrue
560     \renewcommand*\@gls@usetranslator{%
561       \@ifpackageloaded{polyglossia}%
562       {%
563         \let\glsifusetranslator\@secondoftwo
564       }%
565       {%
566         \@ifpackageloaded{babel}%
567         {%
568           \IfFileExists{translator.sty}%
569           {%
570             \RequirePackage{translator}%
571             \let\glsifusetranslator\@firstoftwo
572           }%
573         }%
574       }%
575     }%
576   }%
577 }%
578 \or
579   \glstranslatefalse
580   \let\@gls@usetranslator\relax
581   \let\glsifusetranslator\@secondoftwo
582 \or
583   \glstranslatetrue
584   \let\@gls@usetranslator\relax
585   \let\glsifusetranslator\@secondoftwo
586 \fi
587 }
```

Set the default value:

```

588 \glstranslatefalse
589 \let\glsifusetranslator\@secondoftwo
590 \@ifpackageloaded{translator}%
591 {%
592   \glstranslatetrue
593   \let\glsifusetranslator\@firstoftwo
594 }%
595 {%
596   \@for\gls@thissty:=tracklang,babel,ngerman,polyglossia\do
597   {
598     \@ifpackageloaded{\gls@thissty}%
599     {%
600       \glstranslatetrue
```

```

601     \@endfortrue
602   }%
603   {}%
604 }
605 }

```

indexonlyfirst Set whether to only index on first use.

```

606 \define@boolkey{glossaries.sty}[gls]{indexonlyfirst}[true]{}
607 \glsindexonlyfirstfalse

```

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

```

608 \define@boolkey{glossaries.sty}[gls]{hyperfirst}[true]{}
609 \glshyperfirsttrue

```

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

```

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

```

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

```

611 \define@boolkey{glossaries.sty}[glsacr]{footnote}[true]{}%
612 \ifbool{glsacrdescription}%
613   {}%
614   {%
615     \renewcommand*{\@gls@sanitizedesc}{}%
616   }%
617 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
618 }

```

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

```

619 \define@boolkey{glossaries.sty}[glsacr]{description}[true]{}%
620 \renewcommand*{\@gls@sanitizesymbol}{}%
621 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
622 }

```

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

```

623 \define@boolkey{glossaries.sty}[glsacr]{smallcaps}[true]{}%
624 \renewcommand*{\@gls@sanitizesymbol}{}%
625 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
626 }

```

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

```

627 \define@boolkey{glossaries.sty}[glsacr]{smaller}[true]{}%
628 \renewcommand*{\@gls@sanitizesymbol}{}%
629 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
630 }

```


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

```

631 \define@boolkey{glossaries.sty}[glsacr]{dua}[true]{%
632   \renewcommand*{\@gls@sanitizesymbol}{}%
633   \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
634 }

```

shortcuts Define acronym shortcuts.

```

635 \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.

```

636 \newcommand*{\glsorder}{word}

```

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

```

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

```

order

```

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

```

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

```

640 \newif\ifglxindy

```

The default is `makeindex`:

```

641 \glxindyfalse

```

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

```

642 \@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.

```

643 \define@boolkey[gl]{xindy}{glsnumbers}[true]{}
644 \gls@xindy@glsnumberstrue

```

y@main@language Define what language to use for each glossary type (if a language is not defined for a particular glossary type the language specified for the main glossary is used.)

```

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

```

Define key to set the language

```

646 \define@key[gl]{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.

```

647 \ifcsundef{inputencodingname}{%
648   \def\gls@codepage{}}{%
649   \def\gls@codepage{\inputencodingname}
650 }

```

Define a key to set the code page.

```

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

```

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

```

652 \define@key{glossaries.sty}{xindy}[]{%
653   \glsxindytrue
654   \setkeys[gls]{xindy}{#1}%
655 }

```

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

```

656 \@gls@declareoption{xindygloss}{%
657   \glsxindytrue
658 }

```

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

```

659 \@gls@declareoption{xindynoglsnumbers}{%
660   \glsxindytrue
661   \gls{xindy=glsnumbersfalse}
662 }

```

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

```

663 \define@boolkey{glossaries.sty}[gls]{automake}[true]{%
664   \ifglsautomake
665     \renewcommand*{\@gls@doautomake}{%
666       \PackageError{glossaries}{You must use
667         \string\makeglossaries\space with automake=true}
668       {%
669         Either remove the automake=true setting or
670         add \string\makeglossaries\space to your document preamble.%
671       }}%
672   }%
673 \else
674   \renewcommand*{\@gls@doautomake}{}%
675 \fi
676 }
677 \glsautomakefalse

```

`@gls@doautomake`

```

678 \newcommand*{\@gls@doautomake}{}
679 \AtEndDocument{\@gls@doautomake}

```

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

```
680 \define@boolkey{glossaries.sty}[gls]{savewrites}[true]{%
681   \ifglssavewrites
682     \renewcommand*{\glswritefiles}{\@glswritefiles}%
683   \else
684     \let\glswritefiles\@empty
685   \fi
686 }
```

Set default:

```
687 \glssavewritesfalse
688 \let\glswritefiles\@empty
```

compatible-3.07

```
689 \define@boolkey{glossaries.sty}[gls]{compatible-3.07}[true]{}
690 \boolfalse{glscpatible-3.07}
```

compatible-2.07

```
691 \define@boolkey{glossaries.sty}[gls]{compatible-2.07}[true]{%
  Also set 3.07 compatibility if this option is set.
692   \ifbool{glscpatible-2.07}{%
693     {%
694       \booltrue{glscpatible-3.07}%
695     }%
696   }%
697 }
698 \boolfalse{glscpatible-2.07}
```

symbols Create a “symbols” glossary type

```
699 \@gls@declareoption{symbols}{%
700   \let\@gls@do@symbolsdef\@gls@symbolsdef
701 }
```

Default is not to define the symbols glossary:

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

@gls@symbolsdef

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

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

```
706   \newcommand*{\gls@tr@set@symbols@toctitle}{%
707     \translatelet{\glossarytoctitle}{Symbols (glossaries)}%
708   }%
709 }
```

numbers Create a “symbols” glossary type

```
710 \@gls@declareoption{numbers}{%  
711   \let\@gls@do@numbersdef\@gls@numbersdef  
712 }
```

Default is not to define the numbers glossary:

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

@gls@numbersdef

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

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

```
717   \newcommand*{\gls@tr@set@numbers@toctitle}{%  
718     \translatelet{\glossarytoctitle}{Numbers (glossaries)}%  
719   }%  
720 }%
```

index Create an “index” glossary type

```
721 \@gls@declareoption{index}{%  
722   \let\@gls@do@indexdef\@gls@indexdef  
723 }
```

Default is not to define index glossary:

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

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

```
725 \newcommand*{\@gls@indexdef}{%  
726   \newglossary[ilg]{index}{ind}{idx}{\indexname}%  
727   \newcommand*{\printindex}[1] [] {\printglossary[type=index,##1]}%  
728   \newcommand*{\newterm}[2] [] {%  
729     \newglossaryentry{##2}%  
730     {type={index},name={##2},description={\nopostdesc},##1}}  
731 }%
```

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

```
732 \@for\CurrentOption :=\@declaredoptions\do{%  
733   \ifx\CurrentOption\@empty  
734   \else  
735     \@expandtwoargs  
736     \in@ {,\CurrentOption ,}{,\@classoptionslist,\@curroptions,}%  
737     \ifin@  
738     \@use@ption  
739     \expandafter \let\csname ds@\CurrentOption\endcsname\@empty  
740   \fi  
741 \fi  
742 }
```

Now process options passed to the package:

```
743 \ProcessOptionsX
```

Load backward compatibility stuff:

```
744 \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:

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

```
746 xindy,xindygloss,xindynoglsnumbers,makeindex,%
```

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

Now define `\setupglossaries`:

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

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

```
750 \ifglsacrshortcuts
```

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

```
752 \else
```

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

```
754 \ifglsacrshortcuts
```

```
755 \DefineAcronymSynonyms
```

```
756 \fi
```

```
757 }%
```

```
758 \fi
```

```
759 \glsacrshortcutsfalse
```

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

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

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

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

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

```
765 \@gls@setacrstyle
```

```
766 \@gls@setupshortcuts
```

```
767 \@gls@do@acronymsdef
```

```
768 \@gls@do@numbersdef
```

```
769 \@gls@do@symbolssdef
```

```
770 \@gls@do@indexdef
```

```
771 }
```

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}`).

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

```
773 {%
```

```
774 \ifcsundef{chapter}{}%
```

```
775 {%
```

```

776 \let\@gls@old@chapter\@chapter
777 \def\@chapter[#1]#2{\@gls@old@chapter[#1]{#2}%
778 \ifcsundef{hyperdef}{\hyperdef{section}{\thesection}{}}}%
779 }%
780 }%
781 {}

```

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

```

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

```

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

```

783 \newcommand*{\@onlypremakeg}[1]{%
784 \ifx\@gls@onlypremakeg\@empty
785 \def\@gls@onlypremakeg{#1}%
786 \else
787 \expandafter\toks@ \expandafter{\@gls@onlypremakeg}%
788 \edef\@gls@onlypremakeg{\the\toks@, \noexpand#1}%
789 \fi
790 }

```

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

```

791 \newcommand*{\@disable@onlypremakeg}{%
792 \@for\@thiscs:=\@gls@onlypremakeg\do{%
793 \expandafter\@disable@premakecs\@thiscs%
794 }}

```

`\sable@premakecs` Disables the given command.

```

795 \newcommand*{\@disable@premakecs}[1]{%
796 \def#1{\PackageError{glossaries}{\string#1\space may only be
797 used before \string\makeglossaries}{You can't use
798 \string#1\space after \string\makeglossaries}}%
799 }

```

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`

```

800 \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`
801 `\providecommand*{\acronymname}{Acronyms}`

`\glsettoctitle` Sets the TOC title for the given glossary.
802 `\newcommand*{\glsettoctitle}[1]{%`
803 `\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`
804 `\providecommand*{\entryname}{Notation}`

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

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

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

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

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

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

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

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

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

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

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

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

eGlossariesLang

```
815 \newcommand*{\RequireGlossariesLang}[1]{%
816   \@ifundefined{ver@glossaries-#1.1df}{\input{glossaries-#1.1df}}{}%
817 }
```

sGlossariesLang

```
818 \newcommand*{\ProvidesGlossariesLang}[1]{%
819   \ProvidesFile{glossaries-#1.1df}%
820 }
```

ssarytocaptions Does nothing if translator hasn't been loaded.

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

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

```
822 \ifglstranslate
```

Load tracklang

```
823 \RequirePackage{tracklang}
```

Load translator if required.

```
824 \@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.)

```
825 \@ifpackageloaded{translator}
```

```
826 {%
```

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.

```
827 \ifboolexpr
```

```
828 {
```

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

```
830   and not
```

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

```
832 }
```

```
833 {%
```

```
834   \let\glsifusetranslator\@secondoftwo
```

```
835 }%
```

```
836 {%
```

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

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

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

```
840     {%
```

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

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



```

843         \renewcommand*{\glossaryname}{\translate{Glossary}}}%
844     }%
845     \expandafter\edef\csname captions#1\endcsname{\the\toks@}%
846 }%
847 }%
848 }%
849 }%
850 {}%

```

Check for tracked languages

```

851 \AnyTrackedLanguages
852 {%
853     \ForEachTrackedDialect{\this@dialect}{%
854         \IfTrackedLanguageFileExists{\this@dialect}%
855         {glossaries-}% prefix
856         {.ldf}%
857         {%
858             \RequireGlossariesLang{\CurrentTrackedTag}%
859         }%
860         {%
861             \PackageWarningNoLine{glossaries}%
862             {No language module detected for ‘\this@dialect’.\MessageBreak
863             Language modules need to be installed separately.\MessageBreak
864             Please check on CTAN for a bundle called\MessageBreak
865             ‘glossaries-\CurrentTrackedLanguage’ or similar}%
866         }%
867     }%
868 }%
869 {}%

```

if using translator use translator interface.

```

870 \glsifusetranslator
871 {%
872     \renewcommand*{\glssettoctitle}[1]{%
873         \ifcsdef{gls@tr@set@#1@toctitle}%
874         {%
875             \csuse{gls@tr@set@#1@toctitle}%
876         }%
877         {%
878             \def\glossarytoctitle{\csname @glotype@#1@title\endcsname}%
879         }%
880     }%
881     \renewcommand*{\glossaryname}{\translate{Glossary}}}%
882     \renewcommand*{\acronymname}{\translate{Acronyms}}}%
883     \renewcommand*{\entryname}{\translate{Notation (glossaries)}}}%
884     \renewcommand*{\descriptionname}{%
885         \translate{Description (glossaries)}}}%
886     \renewcommand*{\symbolname}{\translate{Symbol (glossaries)}}}%
887     \renewcommand*{\pagelistname}{%
888         \translate{Page List (glossaries)}}}%

```

```

889 \renewcommand*{\glssymbolsgroupname}{%
890 \translate{Symbols (glossaries)}}%
891 \renewcommand*{\glsnumbersgroupname}{%
892 \translate{Numbers (glossaries)}}%
893 }{}%
894 \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.

```
895 \DeclareRobustCommand*{\nopostdesc}{}

```

`\@nopostdesc` Suppress next description terminator.

```

896 \newcommand*{\@nopostdesc}{%
897 \let\org@glspostdescription\glspostdescription
898 \def\glspostdescription{%
899 \let\glspostdescription\org@glspostdescription}%
900 }

```

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

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

```

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

```
902 \newcommand{\glspar}{\par}

```

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

```

903 \newcommand{\setStyleFile}[1]{%
904 \renewcommand*{\gls@istfilebase}{#1}%
  Just in case \istfilename has been modified.
905 \ifglsxindy
906 \def\istfilename{\gls@istfilebase.xdy}
907 \else
908 \def\istfilename{\gls@istfilebase.ist}
909 \fi
910 }

```

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

```
911 \@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`

```

912 \ifglsxindy
913 \def\istfilename{\gls@istfilebase.xdy}
914 \else
915 \def\istfilename{\gls@istfilebase.ist}
916 \fi

```

`gls@istfilebase`

```
917 \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 \LaTeX , `\@istfilename` ignores its argument.

`\@istfilename`

```
918 \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`

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

`lsSetCompositor` Sets the compositor.

```
920 \newcommand*{\glsSetCompositor}[1]{%
921   \renewcommand*{\glscompositor}{#1}}
```

Only use before `\makeglossaries`

```
922 \@onlypremakeg\glsSetCompositor
```

(The page compositor is usually defined as a dash when using `makeindex`, but most of the standard counters used by \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 `\langle letter \rangle \langle compositor \rangle \langle number \rangle`. For example, if `\@glsAlphacompositor` is set to `."` then it allows locations such as `A.1` whereas if `\@glsAlphacompositor` is set to `"-`" then it allows locations such as `A-1`.

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

`AlphaCompositor` Sets the alpha compositor.

```
924 \ifglsxindy
925   \newcommand*\glsSetAlphaCompositor[1]{%
926     \renewcommand*\@glsAlphacompositor{#1}}
927 \else
928   \newcommand*\glsSetAlphaCompositor[1]{%
929     \glsnoindywarning\glsSetAlphaCompositor}
930 \fi
```

Can only be used before `\makeglossaries`

```
931 \@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.

```
932 \newcommand*{\gls@suffixF}{}%
```

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

```

933 \newcommand*{\glsSetSuffixF}[1]{%
934   \renewcommand*{\gls@suffixF}{#1}}

```

Only has an effect when used before `\makeglossaries`

```

935 \@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.

```

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

```

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

```

937 \newcommand*{\glsSetSuffixFF}[1]{%
938   \renewcommand*{\gls@suffixFF}{#1}%
939 }

```

`\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".

```

940 \ifcsundef{hyperlink}%
941 {%
942   \newcommand*{\glsnumberformat}[1]{#1}%
943 }%
944 {%
945   \newcommand*{\glsnumberformat}[1]{\glshypernumber{#1}}%
946 }

```

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`

```

947 \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`

```

948 \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`

```
949 \newcommand*{\glossarypreamble}{%
950   \csuse{@glossarypreamble@\currentglossary}%
951 }
```

`\setglossarypreamble`

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

Code provided by Michael Pock.

```
952 \newcommand{\setglossarypreamble}[2][\glsdefaulttype]{%
953   \ifglossaryexists{#1}{%
954     \csgdef{@glossarypreamble@#1}{#2}%
955   }{%
956     \GlossariesWarning{%
957       Glossary ‘#1’ is not defined%
958     }%
959   }%
960 }
```

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`

```
961 \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`.

```
962 \newcommand*{\glossarysection}[2][\@gls@title]{%
963   \def\@gls@title{#2}%
964   \ifcsundef{phantomsection}%
965   {%
966     \@glossarysection{#1}{#2}%
967   }%
968   {%
969     \p@glossarysection{#1}{#2}%
970   }%
```

```

971 \glsglossarymark{\glossarytoctitle}%
972 }

```

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

```

973 \ifcsundef{glossarymark}%
974 {%
975   \newcommand{\glsglossarymark}[1]{\glossarymark{#1}}
976 }%
977 {%
978   \@ifclassloaded{memoir}
979   {%
980     \newcommand{\glsglossarymark}[1]{%
981       \ifglsucmark
982         \markboth{\memUHead{#1}}{\memUHead{#1}}%
983       \else
984         \markboth{#1}{#1}%
985       \fi
986     }
987   }%
988   {%
989     \newcommand{\glsglossarymark}[1]{%
990       \ifglsucmark
991         \@mkboth{\mfirstucMakeUppercase{#1}}{\mfirstucMakeUppercase{#1}}%
992       \else
993         \@mkboth{#1}{#1}%
994       \fi
995     }
996   }
997 }

```

`\glossarymark` Provided for backward compatibility:

```

998 \providecommand{\glossarymark}[1]{%
999   \ifglsucmark
1000     \@mkboth{\mfirstucMakeUppercase{#1}}{\mfirstucMakeUppercase{#1}}%
1001   \else
1002     \@mkboth{#1}{#1}%
1003   \fi
1004 }

```

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`

```

1005 \newcommand*{\setglossarysection}[1]{%
1006 \setkeys{glossaries.sty}{section=#1}}

```

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

glossarysection

```

1007 \newcommand*{\@glossarysection}[2]{%
1008   \ifdefempty\@glossarysecstar
1009   {%
1010     \csname\@glossarysec\endcsname[#1]{#2}%
1011   }%
1012   {%
1013     \csname\@glossarysec\endcsname*{#2}%
1014     \@gls@toc{#1}{\@glossarysec}%
1015   }%

  Do automatic labelling if required
1016   \@glossaryseclabel
1017 }
```

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

```

1018 \newcommand*{\@p@glossarysection}[2]{%
1019   \glsclearpage
1020   \phantomsection
1021   \ifdefempty\@glossarysecstar
1022   {%
1023     \csname\@glossarysec\endcsname{#2}%
1024   }%
1025   {%
1026     \@gls@toc{#1}{\@glossarysec}%
1027     \csname\@glossarysec\endcsname*{#2}%
1028   }%

  Do automatic labelling if required
1029   \@glossaryseclabel
1030 }
```

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.

```

1031 \newcommand*{\gls@doclearpage}{%
1032   \ifthenelse{\equal{\@glossarysec}{chapter}}{%
1033     {%
1034       \ifcsundef{cleardoublepage}%
1035       {%
1036         \clearpage
1037       }%
1038     }%
1039     \ifcsdef{if@openright}%
1040     {%
1041       \if@openright
```

```

1042         \cleardoublepage
1043     \else
1044         \clearpage
1045     \fi
1046 }%
1047 {%
1048     \cleardoublepage
1049 }%
1050 }%
1051 }%
1052 {}%
1053 }

```

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

```

1054 \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`

```

1055 \newcommand*\@gls@toc[2]{%
1056     \ifglstoc
1057         \ifglsnumberline
1058             \addcontentsline{toc}{#2}{\protect\numberline{#1}}%
1059         \else
1060             \addcontentsline{toc}{#2}{#1}%
1061         \fi
1062     \fi
1063 }

```

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

```

1064 \newcommand*\glsnxindywarning[1]{%
1065     \GlossariesWarning{Not in xindy mode --- ignoring \string#1}%
1066 }

```

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

```

1067 \newcommand*\glsnomakeindexwarning[1]{%
1068     \GlossariesWarning{Not in makeindex mode --- ignoring \string#1}%
1069 }

```


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

```
1070 \ifglxsindy
1071   \edef\@xdyattributes{\string"default\string"}%
1072 \fi
```

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

```
1073 \ifglxsindy
1074   \edef\@dyattributelist{}%
1075 \fi
```

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

```
1076 \ifglxsindy
1077   \def\@xdylocref{}
1078 \fi
```

`\@gls@ifinlist`

```
1079 \newcommand*{\@gls@ifinlist}[4]{%
1080   \def\@do@ifinlist##1,#1,##2\end@ifinlist{%
1081     \def\@gls@listsuffix{##2}%
1082     \ifx\@gls@listsuffix\@empty
1083       #4%
1084     \else
1085       #3%
1086     \fi
1087   }%
1088   \@do@ifinlist,#2,#1,\end@ifinlist
1089 }
```

`\@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.

```
1090 \ifglxsindy
1091   \newcommand*{\@xdycounters}{\@glscounter}
1092   \newcommand*\GlsAddXdyCounters[1]{%
1093     \@for\@gls@ctr:=#1\do{%
1094       Check if already in list before adding.
1095       \edef\@do@addcounter{%
1096         \noexpand\@gls@ifinlist{\@gls@ctr}{\@xdycounters}{}%
1097         \noexpand\edef\noexpand\@xdycounters{\@xdycounters,%
1098           \noexpand\@gls@ctr}%
1099       }%
1100     }%
1101     \@do@addcounter
1102   }
1103 }
```

Only has an effect before `\writeist`:

```

1104 \@onlypremakeg\GlsAddXdyCounters
1105 \else
1106   \newcommand*\GlsAddXdyCounters[1]{%
1107     \glsnxindywarning\GlsAddXdyAttribute
1108   }
1109 \fi

```

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

```

1110 \newcommand*\@disabled@glssaddxdycounters{%
1111   \PackageError{glossaries}{\string\GlsAddXdyCounters\space
1112     can't be used after \string\GlsAddXdyAttribute}{Move all
1113     occurrences of \string\GlsAddXdyCounters\space before the first
1114     instance of \string\GlsAddXdyAttribute}%
1115 }

```

`AddXdyAttribute` Adds an attribute.

```

1116 \ifglxsindy

```

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

```

1117 \newcommand*\@glssaddxdyattribute[2]{%

```

Add to xindy attribute list

```

1118   \edef\@xdyattributes{\@xdyattributes ^^J \string"#1\string" ^^J
1119     \string"#2#1\string"}%

```

Add to xindy markup location.

```

1120   \expandafter\toks@\expandafter{\@xdylocref}%
1121   \edef\@xdylocref{\the\toks@ ^^J%
1122     (markup-locref
1123     :open \string"\glstildechar n%
1124       \expandafter\string\csname glsX#2X#1\endcsname
1125       \string" ^^J
1126     :close \string"\string" ^^J
1127     :attr \string"#2#1\string")}%

```

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

```

1128   \expandafter\gdef\csname glsX#2X#1\endcsname##1##2{%
1129     \setentrycounter[##1]{#2}\csname #1\endcsname{##2}%
1130   }%
1131 }

```

High-level command:

```

1132 \newcommand*\GlsAddXdyAttribute[1]{%

```

Add to comma-separated attribute list

```

1133   \ifx\@xdyattributelist\@empty
1134     \edef\@xdyattributelist{#1}%
1135   \else
1136     \edef\@xdyattributelist{\@xdyattributelist,#1}%
1137   \fi

```

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

```

1138 \for\@this@counter:=\@xdycounters\do{%
1139 \protected@edef\gls@do@addxdyattribute{%
1140 \noexpand\@glsaddxdyattribute{#1}{\@this@counter}%
1141 }
1142 \gls@do@addxdyattribute
1143 }%
```

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

```

1144 \let\GlsAddXdyCounters\@disabled@glsaddxdycounters
1145 }
```

Only has an effect before `\writeist`:

```

1146 \@onlypremakeg\GlsAddXdyAttribute
1147 \else
1148 \newcommand*\GlsAddXdyAttribute[1]{%
1149 \glsnoxindywarning\GlsAddXdyAttribute}
1150 \fi
```

`\definedattributes` Add known attributes for all defined counters

```

1151 \ifglxindy
1152 \newcommand*\@gls@addpredefinedattributes{%
1153 \GlsAddXdyAttribute{glsnumberformat}
1154 \GlsAddXdyAttribute{textrm}
1155 \GlsAddXdyAttribute{textsf}
1156 \GlsAddXdyAttribute{texttt}
1157 \GlsAddXdyAttribute{textbf}
1158 \GlsAddXdyAttribute{textmd}
1159 \GlsAddXdyAttribute{textit}
1160 \GlsAddXdyAttribute{textup}
1161 \GlsAddXdyAttribute{textsl}
1162 \GlsAddXdyAttribute{textsc}
1163 \GlsAddXdyAttribute{emph}
1164 \GlsAddXdyAttribute{glshypernumber}
1165 \GlsAddXdyAttribute{hyperrm}
1166 \GlsAddXdyAttribute{hypersf}
1167 \GlsAddXdyAttribute{hypertt}
1168 \GlsAddXdyAttribute{hyperbf}
1169 \GlsAddXdyAttribute{hypermd}
1170 \GlsAddXdyAttribute{hyperit}
1171 \GlsAddXdyAttribute{hyperup}
1172 \GlsAddXdyAttribute{hypersl}
1173 \GlsAddXdyAttribute{hypersc}
1174 \GlsAddXdyAttribute{hyperemph}

1175 \GlsAddXdyAttribute{glsignore}
1176 }
1177 \else
1178 \let\@gls@addpredefinedattributes\relax
1179 \fi
```

dyuseralphabets List of additional alphabets

```
1180 \def\@xdyuseralphabets{}
```

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

```
1181 \ifglxsindy
1182   \newcommand*{\GlsAddXdyAlphabet}[2]{%
1183     \edef\@xdyuseralphabets{%
1184       \@xdyuseralphabets ^^J
1185       (define-alphabet "#1" (#2))}%
1186   \else
1187     \newcommand*{\GlsAddXdyAlphabet}[2]{%
1188       \glsnnoxindywarning\GlsAddXdyAlphabet}
1189 \fi
```

This code is only required for xindy:

```
1190 \ifglxsindy
```

dy@locationlist List of predefined location names.

```
1191   \newcommand*{\@gls@xdy@locationlist}{%
1192     roman-page-numbers,%
1193     Roman-page-numbers,%
1194     arabic-page-numbers,%
1195     alpha-page-numbers,%
1196     Alpha-page-numbers,%
1197     Appendix-page-numbers,%
1198     arabic-section-numbers%
1199   }
```

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.

```
1200   \protected@edef\@gls@roman{\@roman{0}\string"
1201     \string"roman-numbers-lowercase\string" :sep \string"}}%
1202   \@onelevel@sanitize\@gls@roman
1203   \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
1204     :sep \string"}%
1205   \@onelevel@sanitize\@tmp
1206   \ifx\@tmp\@gls@roman
1207     \expandafter
1208       \edef\csname @gls@xdy@Lclass@roman-page-numbers\endcsname{%
1209         \string"roman-numbers-lowercase\string"%
1210       }%
1211   \else
1212     \expandafter
```

```

1213      \edef\csname @gls@xdy@Lclass@roman-page-numbers\endcsname{
1214      :sep \string"\@gls@roman\string"%
1215      }%
1216 \fi

```

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

```

1217 \expandafter\def\csname @gls@xdy@Lclass@Roman-page-numbers\endcsname{%
1218 \string"roman-numbers-uppercase\string"%
1219 }%

```

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

```

1220 \expandafter\def\csname @gls@xdy@Lclass@arabic-page-numbers\endcsname{%
1221 \string"arabic-numbers\string"%
1222 }%

```

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

```

1223 \expandafter\def\csname @gls@xdy@Lclass@alpha-page-numbers\endcsname{%
1224 \string"alpha\string"%
1225 }%

```

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

```

1226 \expandafter\def\csname @gls@xdy@Lclass@Alpha-page-numbers\endcsname{%
1227 \string"ALPHA\string"%
1228 }%

```

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

```

1229 \expandafter\def\csname @gls@xdy@Lclass@Appendix-page-numbers\endcsname{%
1230 \string"ALPHA\string"
1231 :sep \string"\@glsAlphacompositor\string"
1232 \string"arabic-numbers\string"%
1233 }

```

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

```

1234 \expandafter\def\csname @gls@xdy@Lclass@arabic-section-numbers\endcsname{%
1235 \string"arabic-numbers\string"
1236 :sep \string"\glscompositor\string"
1237 \string"arabic-numbers\string"%
1238 }%

```

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

```

1239 \def\@xdyuserlocationdefs{}

```

erlocationnames List of additional user location names

```

1240 \def\@xdyuserlocationnames{}

```

End of xindy-only block:

```

1241 \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.)

```

1242 \ifglsxindy
1243   \newcommand*{\GlsAddXdyLocation}[3][\]{%
1244     \def\@gls@tmp{#1}%
1245     \ifx\@gls@tmp\@empty
1246       \edef\@xdyuserlocationdefs{%
1247         \@xdyuserlocationdefs ^^J%
1248         (define-location-class \string"#2\string"^^J\space\space
1249         \space(:sep \string"{}\glssopenbrace\string" #3
1250         :sep \string"\glsclosebrace\string"))
1251       }%
1252     \else
1253       \edef\@xdyuserlocationdefs{%
1254         \@xdyuserlocationdefs ^^J%
1255         (define-location-class \string"#2\string"^^J\space\space
1256         \space(:sep "\glssopenbrace"
1257         #1
1258         :sep "\glsclosebrace\glssopenbrace" #3
1259         :sep "\glsclosebrace"))
1260       }%
1261     \fi
1262     \edef\@xdyuserlocationnames{%
1263       \@xdyuserlocationnames^^J\space\space\space
1264       \string"#1\string"}%
1265   }

```

Only has an effect before `\writeist`:

```

1266 \@onlypremakeg\GlsAddXdyLocation
1267 \else
1268   \newcommand*{\GlsAddXdyLocation}[2]{%
1269     \glsnnoxindywarning\GlsAddXdyLocation}
1270 \fi

```

`\locationclassorder` Define location class order

```

1271 \ifglsxindy
1272   \edef\@xdylocationclassorder{^^J\space\space\space
1273     \string"roman-page-numbers\string"^^J\space\space\space
1274     \string"arabic-page-numbers\string"^^J\space\space\space
1275     \string"arabic-section-numbers\string"^^J\space\space\space
1276     \string"alpha-page-numbers\string"^^J\space\space\space
1277     \string"Roman-page-numbers\string"^^J\space\space\space
1278     \string"Alpha-page-numbers\string"^^J\space\space\space
1279     \string"Appendix-page-numbers\string"
1280     \@xdyuserlocationnames^^J\space\space\space
1281     \string"see\string"
1282   }
1283 \fi

```

Change the location order.

ationClassOrder

```
1284 \ifglxindy
1285   \newcommand*\GlsSetXdyLocationClassOrder[1]{%
1286     \def\@xdylocationclassorder{#1}}
1287 \else
1288   \newcommand*\GlsSetXdyLocationClassOrder[1]{%
1289     \glsnnoxindywarning\GlsSetXdyLocationClassOrder}
1290 \fi
```

\@xdysortrules Define sort rules

```
1291 \ifglxindy
1292   \def\@xdysortrules{}
1293 \fi
```

\GlsAddSortRule Add a sort rule

```
1294 \ifglxindy
1295   \newcommand*\GlsAddSortRule[2]{%
1296     \expandafter\toks@\expandafter{\@xdysortrules}%
1297     \protected@edef\@xdysortrules{\the\toks@ ^^J
1298       (sort-rule \string"#1\string" \string"#2\string")}%
1299   }
1300 \else
1301   \newcommand*\GlsAddSortRule[2]{%
1302     \glsnnoxindywarning\GlsAddSortRule}
1303 \fi
```

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

```
1304 \ifglxindy
1305   \def\@xdyrequiredstyles{tex}
1306 \fi
```

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

```
1307 \ifglxindy
1308   \newcommand*\GlsAddXdyStyle[1]{%
1309     \edef\@xdyrequiredstyles{\@xdyrequiredstyles,#1}}%
1310 \else
1311   \newcommand*\GlsAddXdyStyle[1]{%
1312     \glsnnoxindywarning\GlsAddXdyStyle}
1313 \fi
```

GlsSetXdyStyles Reset the list of required styles

```
1314 \ifglxindy
1315   \newcommand*\GlsSetXdyStyles[1]{%
1316     \edef\@xdyrequiredstyles{#1}}
1317 \else
1318   \newcommand*\GlsSetXdyStyles[1]{%
1319     \glsnnoxindywarning\GlsSetXdyStyles}
1320 \fi
```

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

```
1321 \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.

```
1322 \def\@xdylanguage#1#2{}
```

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

```
1323 \ifglxindy
1324   \newcommand*\GlsSetXdyLanguage[2][\glsdefaultttype]{%
1325     \ifglossaryexists{#1}{%
1326       \expandafter\def\csname @xdy@#1@language\endcsname{#2}%
1327     }{%
1328       \PackageError{glossaries}{Can't set language type for
1329         glossary type '#1' --- no such glossary}{%
1330         You have specified a glossary type that doesn't exist}}
1331 \else
1332   \newcommand*\GlsSetXdyLanguage[2][]{%
1333     \glsnoxywarning\GlsSetXdyLanguage}
1334 \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.

```
1335 \def\@gls@codepage#1#2{}
```

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

```
1336 \ifglxindy
1337   \newcommand*\GlsSetXdyCodePage[1]{%
1338     \renewcommand*\@gls@codepage{#1}%
1339   }
```

Suggested by egreg:

```
1340   \AtBeginDocument{%
1341     \ifx\@gls@codepage\@empty
1342       \@ifpackageloaded{fontspec}{\def\@gls@codepage{utf8}}{}%
1343     \fi
1344   }
1345 \else
1346   \newcommand*\GlsSetXdyCodePage[1]{%
1347     \glsnoxywarning\GlsSetXdyCodePage}
1348 \fi
```


`\xdylettergroups` Store letter group definitions.

```

1349 \ifglxsindy
1350   \ifglxs@xindy@glnumbers
1351     \def\xdylettergroups{(define-letter-group
1352       \string"glnumbers\string"^^J\space\space\space
1353       :prefixes (\string"0\string" \string"1\string"
1354       \string"2\string" \string"3\string" \string"4\string"
1355       \string"5\string" \string"6\string" \string"7\string"
1356       \string"8\string" \string"9\string")^^J\space\space\space
1357       :before \string"@glfirstletter\string")}
1358   \else
1359     \def\xdylettergroups{}
1360   \fi
1361 \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.

```

1362   \newcommand*\GlsAddLetterGroup[2]{%
1363     \expandafter\toks@\expandafter{\@xdylettergroups}%
1364     \protected@edef\xdylettergroups{\the\toks@^^J%
1365     (define-letter-group \string"#1\string"^^J\space\space\space#2)}%
1366   }%

```

1.5 Loops and conditionals

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

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

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

```

1367 \newcommand*\forallglossaries}[3][\@glo@types]{%
1368   \@for#2:=#1\do{\ifx#2\@empty\else#3\fi}%
1369 }

```

`\forall acronyms`

```

1370 \newcommand*\forallacronyms}[2]{%
1371   \@for#1:=\@glsacronymlists\do{\ifx#1\@empty\else#2\fi}%
1372 }

```

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

```
\forall sentries[<type>]{<cmd>}{<code>}
```

where *<type>* is the glossary label and *<cmd>* is a control sequence which will be set to the entry label in the current iteration.

```

1373 \newcommand*{\forglentries}[3][\glsdefaulttype]{%
1374   \edef\@glo@list{\csname glolist@#1\endcsname}%
1375   \@for#2:=\@glo@list\do
1376   {%
1377     \ifdefempty{#2}{\}{#3}%
1378   }%
1379 }

```

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

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

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

```

1380 \newcommand*{\forallglentries}[3][\@glo@types]{%
1381   \expandafter\forallglossaries\expandafter[#1]{\@thisglo@}%
1382   {%
1383     \forglentries[\@thisglo@]{#2}{#3}%
1384   }%
1385 }

```

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

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

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

```

1386 \newcommand{\ifglossaryexists}[3]{%
1387   \ifcsundef{@glo@type@#1@out}{#3}{#2}%
1388 }

```

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`

```
1389 \newcommand*{\glsdetoklabel}[1]{#1}
```

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

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

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

```
1390 \newcommand{\ifglentryexists}[3]{%
1391   \ifcsundef{glo@\glsdetoklabel{#1}@name}{#3}{#2}%
1392 }
```

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

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

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

```
1393 \newcommand*{\ifglused}[3]{%
1394   \ifbool{glo@\glsdetoklabel{#1}@flag}{#2}{#3}%
1395 }
```

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

\backslash glsdoifexists

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

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

```
1396 \newcommand{\glsdoifexists}[2]{%
1397   \ifglentryexists{#1}{#2}{%
1398     \PackageError{glossaries}{Glossary entry '\glsdetoklabel{#1}'
1399     has not been defined}{You need to define a glossary entry before you
1400     can use it.}}%
1401 }
```

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

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

```
1402 \newcommand{\glsdoifnoexists}[2]{%
1403   \ifglentryexists{#1}{#2}{%
1404     \PackageError{glossaries}{Glossary entry '\glsdetoklabel{#1}' has already
1405     been defined}{}}{#2}%
1406 }
```

\backslash glsdoifexistsorwarn

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

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

```
1407 \newcommand{\glsdoifexistsorwarn}[2]{%
1408   \ifglentryexists{#1}{#2}{%
1409     \GlossariesWarning{Glossary entry '\glsdetoklabel{#1}'
1410     has not been defined}%
1411   }%
1412 }
```

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

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

```

1413 \newcommand{\glsdoifexistsordo}[3]{%
1414   \ifglentryexists{#1}{#2}{%
1415     \PackageError{glossaries}{Glossary entry ‘\glsdetoklabel{#1}’
1416       has not been defined}{You need to define a glossary entry before you
1417       can use it.}%
1418     #3%
1419   }%
1420 }
```

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

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

```

1421 \newcommand{\doifglossarynoexistsordo}[3]{%
1422   \ifglossaryexists{#1}%
1423   {%
1424     \PackageError{glossaries}{Glossary type ‘#1’ already exists}{}%
1425     #3%
1426   }%
1427   {#2}%
1428 }
```

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

```

1429 \newcommand{\ifglshaschildren}[3]{%
1430   \glsdoifexists{#1}%
1431   {%
1432     \def\do@glshaschildren{#3}%
1433     \edef\@gls@thislabel{\glsdetoklabel{#1}}%
1434     \expandafter\forglentries\expandafter
1435     [\csname glo@\@gls@thislabel @type\endcsname]
1436     {\glo@label}%
1437     {%
1438       \letcs\glo@parent{glo@\glo@label @parent}%
1439       \ifdefequal\@gls@thislabel\glo@parent
1440       {%
1441         \def\do@glshaschildren{#2}%
1442         \@endfortrue
1443       }%
1444     }%
1445   }%
1446   \do@glshaschildren
1447 }%
1448 }
```

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

```
1449 \newcommand{\ifglshasparent}[3]{%
1450   \glsdoifexists{#1}%
1451   {%
1452     \ifcseempty{glo@\glsdetoklabel{#1}@parent}{#3}{#2}%
1453   }%
1454 }
```

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

```
1455 \newcommand*{\ifglshasdesc}[3]{%
1456   \ifcseempty{glo@\glsdetoklabel{#1}@desc}{%
1457     {#3}%
1458     {#2}%
1459   }
```

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

```
1460 \newcommand*{\ifglsdescsuppressed}[3]{%
1461   \ifcsequal{glo@\glsdetoklabel{#1}@desc}{@no@post@desc}%
1462   {#2}%
1463   {#3}%
1464 }
```

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

```
1465 \newcommand*{\ifglshassymbol}[3]{%
1466   \letcs{\@glo@symbol}{glo@\glsdetoklabel{#1}@symbol}%
1467   \ifdefempty\@glo@symbol
1468   {#3}%
1469   {%
1470     \ifdefequal\@glo@symbol\@gls@default@value
1471     {#3}%
1472     {#2}%
1473   }%
1474 }
```

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

```
1475 \newcommand*{\ifglshaslong}[3]{%
1476   \letcs{\@glo@long}{glo@\glsdetoklabel{#1}@long}%
1477   \ifdefempty\@glo@long
1478   {#3}%
1479   {%
1480     \ifdefequal\@glo@long\@gls@default@value
1481     {#3}%
1482     {#2}%
1483   }%
1484 }
```

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

```

1485 \newcommand*{\ifglshasshort}[3]{%
1486   \letcs{\@glo@short}{glo@glstdetoklabel{#1}@short}%
1487   \ifdefempty\@glo@short
1488     {#3}%
1489     {%
1490       \ifdefequal\@glo@short\@gls@default@value
1491         {#3}%
1492         {#2}%
1493     }%
1494 }
```

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

```

1495 \newcommand*{\ifglshasfield}[4]{%
1496   \glstdoifexists{#2}%
1497   {%
1498     \letcs{\@glo@thisvalue}{glo@glstdetoklabel{#2}@#1}%

```

First check supplied field label is defined.

```

1499   \ifdef\@glo@thisvalue
1500   {%

```

Is defined, so now check if empty.

```

1501     \ifdefempty\@glo@thisvalue
1502     {%

```

Is empty, so doesn't have field set.

```

1503         #4%
1504     }%
1505   {%

```

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

```

1506     \ifdefequal\@glo@thisvalue\@gls@default@value
1507     {%

```

Value is set to the default value.

```

1508         #4%
1509     }%
1510   {%

```

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

```

1511     \let\glscurrentfieldvalue\@glo@thisvalue
1512     #3%
1513   }%
1514 }%
1515 }%
1516 {%

```

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

```
1517 \gls@fetchfield{\gls@thisfield}{#1}%
```

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

```
1518 \ifdef\gls@thisfield
1519 {%
```

Is defined, so now check if empty.

```
1520 \letcs{\glo@thisvalue}{glo\glsdetoklabel{#2}@\gls@thisfield}%
1521 \ifdefempty\glo@thisvalue
1522 {%
```

Is empty so field hasn't been set.

```
1523 #4%
1524 }%
1525 {%
```

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

```
1526 \ifdequal\glo@thisvalue\gls@default@value
1527 {%
```

Value is set to the default value.

```
1528 #4%
1529 }%
1530 {%
```

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

```
1531 \let\glscurrentfieldvalue\glo@thisvalue
1532 #3%
1533 }%
1534 }%
1535 }%
1536 {%
```

Not defined.

```
1537 \GlossariesWarning{Unknown entry field '#1'}%
1538 #4%
1539 }%
1540 }%
1541 }%
1542 }
```

`\glscurrentfieldvalue`

```
1543 \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
1544 \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.

1545 \newcommand*\@gls@provide@newglossary{%
1546   \protected@write\@auxout{}\string\providecommand\string\@newglossary[4]{}}%

Only need to do this once.

1547 \let\@gls@provide@newglossary\relax
1548 }

\defglsentryfmt Allow different glossaries to have different display styles.

1549 \newcommand*\defglsentryfmt}[2][\glsdefaulttype]{%
1550   \csgdef{gls@#1@entryfmt}{#2}%
1551 }

\gls@doentryfmt
1552 \newcommand*\gls@doentryfmt}[1]{\csuse{gls@#1@entryfmt}}

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

1553 \newcommand*\@gls@forbidtextext}[1]{%
1554   \ifboolexpr{test {\ifdefstring{#1}{tex}}
1555             or test {\ifdefstring{#1}{TEX}}}
1556   {%
1557     \def#1{nottex}%
1558     \PackageError{glossaries}%
1559       {Forbidden '.tex' extension replaced with '.nottex'}%
1560     {I'm sorry, I can't allow you to do something so reckless.\MessageBreak
1561       Don't use '.tex' as an extension for a temporary file.}%
1562   }%
1563   {%
1564   }%
1565 }

\gls@gobbleopt Discard optional argument.

1566 \newcommand*\gls@gobbleopt{\new@ifnextchar[\@gls@gobbleopt]}
1567 \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`

```
1568 \newcommand*{\newglossary}{\@ifstar\s@newglossary\ns@newglossary}
```

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

```
1569 \newcommand*{\s@newglossary}[2]{%
```

```
1570 \ns@newglossary[#1-glg]{#1}{#1-gls}{#1-glo}{#2}%
```

```
1571 }
```

`\ns@newglossary` Define the unstarred version.

```
1572 \newcommand*{\ns@newglossary}[5][glg]{%
```

```
1573 \doifglossarynoexistsordo{#2}%
```

```
1574 {%
```

Check if default has been set

```
1575 \ifundef\glsdefaultttype
```

```
1576 {%
```

```
1577 \gdef\glsdefaultttype{#2}%
```

```
1578 }{}%
```

Add this to the list of glossary types:

```
1579 \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.

```
1580 \expandafter\gdef\csname glolist@#2\endcsname{,}%
```

Store the file extensions:

```
1581 \expandafter\edef\csname @glotype@#2@log\endcsname{#1}%
```

```
1582 \expandafter\edef\csname @glotype@#2@in\endcsname{#3}%
```

```
1583 \expandafter\edef\csname @glotype@#2@out\endcsname{#4}%
```

```
1584 \expandafter\@gls@forbidtexext\csname @glotype@#2@log\endcsname
```

```
1585 \expandafter\@gls@forbidtexext\csname @glotype@#2@in\endcsname
```

```
1586 \expandafter\@gls@forbidtexext\csname @glotype@#2@out\endcsname
```

Store the title:

```
1587 \expandafter\def\csname @glotype@#2@title\endcsname{#5}%
```

```
1588 \@gls@provide@newglossary
```

```
1589 \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.

```

1590 \ifcsundef{gls@#2@entryfmt}%
1591 {%
1592   \defglsentryfmt[#2]{\glsentryfmt}%
1593 }%
1594 {}%

```

Define sort counter if required:

```

1595 \@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.)

```

1596 \@ifnextchar[{\@gls@setcounter{#2}}%
1597   {\@gls@setcounter{#2}[\glscounter]}%
1598 }%
1599 {%
1600   \gls@gobbleopt
1601 }%
1602 }

```

`\altnewglossary`

```

1603 \newcommand*{\altnewglossary}[3]{%
1604   \newglossary[#2-glg]{#1}{#2-gls}{#2-glo}{#3}%
1605 }

```

Only define new glossaries in the preamble:

```

1606 \@onlypreamble{\newglossary}

```

Only define new glossaries before `\makeglossaries`

```

1607 \@onlypremakeg\newglossary

```

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

`\@newglossary`

```

1608 \newcommand*{\@newglossary}[4]{}

```

Store counter to be used for given glossary type (the first argument is the glossary label, the second argument is the name of the counter):

`@gls@setcounter`

```

1609 \def\@gls@setcounter#1[#2]{%
1610   \expandafter\def\csname @glsotype@#1@counter\endcsname{#2}%

```

Add counter to xindy list, if not already added:

```

1611   \ifglsxindy
1612     \GlsAddXdyCounters{#2}%
1613   \fi
1614 }

```

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

@gls@getcounter

```
1615 \newcommand*{\@gls@getcounter}[1]{%
1616 \csname @gls@#1@counter\endcsname
1617 }
```

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

```
1618 \glsdefmain
```

Define the “acronym” glossaries if required.

```
1619 \@gls@do@acronymsdef
```

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

```
1620 \@gls@do@symbolsdef
```

```
1621 \@gls@do@numbersdef
```

```
1622 \@gls@do@indexdef
```

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

```
1623 \newcommand*{\newignoredglossary}[1]{%
1624 \ifdefempty\@ignored@glossaries
1625 {%
1626 \edef\@ignored@glossaries{#1}%
1627 }%
1628 {%
1629 \eappto\@ignored@glossaries{, #1}%
1630 }%
1631 \csgdef{glolist@#1}{,}%
1632 \ifcsundef{gls@#1@entryfmt}%
1633 {%
1634 \defglsentryfmt[#1]{\glsentryfmt}%
1635 }%
1636 {}%
1637 \ifdefempty\@gls@nohyperlist
1638 {%
1639 \renewcommand*{\@gls@nohyperlist}{#1}%
1640 }%
1641 {%
1642 \eappto\@gls@nohyperlist{, #1}%
1643 }%
1644 }
```

`ignored@glossaries` List of ignored glossaries.

```
1645 \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.

```

1646 \newcommand*{\ifignoredglossary}[3]{%
1647   \edef\@gls@igtype{#1}%
1648   \expandafter\DTLifinlist\expandafter
1649     {\@gls@igtype}{\@ignored@glossaries}{#2}{#3}%
1650 }

```

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.

```

1651 \define@key{glossentry}{name}{%
1652 \def\@glo@name{#1}%
1653 }

```

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

```

1654 \define@key{glossentry}{description}{%
1655 \def\@glo@desc{#1}%
1656 }

```

descriptionplural

```

1657 \define@key{glossentry}{descriptionplural}{%
1658 \def\@glo@descplural{#1}%
1659 }

```

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>*.

```

1660 \define@key{glossentry}{sort}{%
1661 \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.

```

1662 \define@key{glossentry}{text}{%
1663 \def\@glo@text{#1}%
1664 }

```

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.

```
1665 \define@key{glossentry}{plural}{%
1666 \def\@glo@plural{#1}%
1667 }
```

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.

```
1668 \define@key{glossentry}{first}{%
1669 \def\@glo@first{#1}%
1670 }
```

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.

```
1671 \define@key{glossentry}{firstplural}{%
1672 \def\@glo@firstplural{#1}%
1673 }
```

s@default@value

```
1674 \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).

```
1675 \define@key{glossentry}{symbol}{%
1676 \def\@glo@symbol{#1}%
1677 }
```

symbolplural

```
1678 \define@key{glossentry}{symbolplural}{%
1679 \def\@glo@symbolplural{#1}%
1680 }
```

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

```
1681 \define@key{glossentry}{type}{%
1682 \def\@glo@type{#1}}
```

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

```
1683 \define@key{glossentry}{counter}{%
1684 \ifcsundef{c@#1}%
```

```

1685 {%
1686   \PackageError{glossaries}%
1687   {There is no counter called ‘#1’}%
1688   {%
1689     The counter key should have the name of a valid counter
1690     as its value%
1691   }%
1692 }%
1693 {%
1694   \def\@glo@counter{#1}%
1695 }%
1696 }

```

see The see key specifies a list of cross-references

```

1697 \define@key{glossentry}{see}{%
1698   \gls@checkseeallowed
1699   \def\@glo@see{#1}%
1700   \@glo@seeautonumberlist
1701 }

```

checkseeallowed

```

1702 \newcommand*{\gls@checkseeallowed}{%
1703   \@gls@see@noindex
1704 }

```

ed@preambleonly

```

1705 \newcommand*{\gls@checkseeallowed@preambleonly}{%
1706   \GlossariesWarning{glossaries}%
1707   {‘see’ key doesn’t have any effect when used in the document
1708     environment. Move the definition to the preamble
1709     after \string\makeglossaries\space
1710     or \string\makenoidxglossaries}%
1711 }

```

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

```

1712 \define@key{glossentry}{parent}{%
1713   \def\@glo@parent{#1}}

```

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

```

1714 \define@choicekey{glossentry}{nonumberlist}[\val\nr]{true,false}[true]{%
1715   \ifcase\nr\relax
1716     \def\@glo@prefix{\glsnonextpages}%
1717     \@gls@savenonumberlist{true}%
1718   \else
1719     \def\@glo@prefix{\glsnextpages}%
1720     \@gls@savenonumberlist{false}%
1721   \fi
1722 }

```

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.

```
1723 \newcommand*{\@gls@savenonumberlist}[1]{}
```

initnonumberlist

```
1724 \newcommand*{\@gls@initnonumberlist}{}%
```

nitnonumberlist

```
1725 \newcommand*{\@gls@storenonumberlist}[1]{}
```

savenonumberlist Allow the nonumberlist value to be saved.

```
1726 \newcommand*{\@gls@enablesavenonumberlist}{%
1727   \renewcommand*{\@gls@initnonumberlist}{%
1728     \undef\@glo@nonumberlist
1729   }%
1730   \renewcommand*{\@gls@savenonumberlist}[1]{%
1731     \def\@glo@nonumberlist{##1}%
1732   }%
1733   \renewcommand*{\@gls@storenonumberlist}[1]{%
1734     \ifdef\@glo@nonumberlist
1735       {%
1736         \cslet{glo@glsdetoklabel{##1}@nonumberlist}{\@glo@nonumberlist}%
1737       }%
1738     }%
1739   }%
1740   \appto\@gls@keymap{,{nonumberlist}{nonumberlist}}%
1741 }
```

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

user1

```
1742 \define@key{glossentry}{user1}{%
1743   \def\@glo@useri{#1}%
1744 }
```

user2

```
1745 \define@key{glossentry}{user2}{%
1746   \def\@glo@userii{#1}%
1747 }
```

user3

```
1748 \define@key{glossentry}{user3}{%
1749   \def\@glo@useriii{#1}%
1750 }
```

user4

```
1751 \define@key{glossentry}{user4}{%  
1752   \def\@glo@useriv{#1}%  
1753 }
```

user5

```
1754 \define@key{glossentry}{user5}{%  
1755   \def\@glo@userv{#1}%  
1756 }
```

user6

```
1757 \define@key{glossentry}{user6}{%  
1758   \def\@glo@uservi{#1}%  
1759 }
```

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.

```
1760 \define@key{glossentry}{short}{%  
1761   \def\@glo@short{#1}%  
1762 }
```

shortplural This key is provided for use by \newacronym.

```
1763 \define@key{glossentry}{shortplural}{%  
1764   \def\@glo@shortpl{#1}%  
1765 }
```

long This key is provided for use by \newacronym.

```
1766 \define@key{glossentry}{long}{%  
1767   \def\@glo@long{#1}%  
1768 }
```

longplural This key is provided for use by \newacronym.

```
1769 \define@key{glossentry}{longplural}{%  
1770   \def\@glo@longpl{#1}%  
1771 }
```

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

```
1772 \newcommand*\@glsnname{%  
1773   \PackageError{glossaries}{name key required in  
1774   \string\newglossaryentry\space for entry '@glo@label'}{You  
1775   haven't specified the entry name}}
```

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

```
1776 \newcommand*\@glsnodesc{%  
1777   \PackageError{glossaries}  
1778   {%  
1779     description key required in \string\newglossaryentry\space  
1780     for entry '@glo@label'%  
1781   }}
```



```

1781 }%
1782 {%
1783     You haven't specified the entry description%
1784 }%
1785 }%

```

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

```

1786 \newcommand*{\@glsdefaultplural}{}

```

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

```

1787 \newcommand*{\@gls@missingnumberlist}[1]{%
1788   ??%
1789   \ifglssavenumberlist
1790     \GlossariesWarning{Missing number list for entry '#1'.
1791       Maybe makeglossaries + rerun required}%
1792   \else
1793     \PackageError{glossaries}%
1794       {Package option 'savenumberlist=true' required}%
1795     {%
1796       You must use the 'savenumberlist' package option
1797       to reference location lists.%
1798     }%
1799   \fi
1800 }

```

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

```

1801 \newcommand*{\@glsdefaultsort}{\@glo@name}

```

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

```

1802 \newcount\gls@level

```

`\@noexpand@field`

```

1803 \newcommand{\@@gls@noexpand@field}[3]{%
1804   \expandafter\global\expandafter
1805     \let\csname glo@#1@#2\endcsname#3%
1806 }

```

`\noexpand@fields`

```

1807 \newcommand{\@gls@noexpand@fields}[4]{%
1808   \ifcsdef{gls@assign@#3@field}
1809   {%
1810     \ifdefequal{#4}{\@gls@default@value}%
1811     {%
1812       \edef\@gls@value{\expandonce{#1}}%
1813       \csuse{gls@assign@#3@field}{#2}{\@gls@value}%
1814     }%
1815   }%
1816     \csuse{gls@assign@#3@field}{#2}{#4}%

```

```

1817     }%
1818 }%
1819 {%
1820     \ifdefequal{#4}{\@gls@default@value}%
1821     {%
1822         \edef\@gls@value{\expandonce{#1}}%
1823         \@gls@noexpand@field{#2}{#3}{\@gls@value}%
1824     }%
1825     {%
1826         \@gls@noexpand@field{#2}{#3}{#4}%
1827     }%
1828 }%
1829 }

```

ls@expand@field

```

1830 \newcommand{\@gls@expand@field}[3]{%
1831     \expandafter
1832     \protected@xdef\csname glo@#1@#2\endcsname{#3}%
1833 }

```

s@expand@fields

```

1834 \newcommand{\@gls@expand@fields}[4]{%
1835     \ifcsdef{gls@assign@#3@field}
1836     {%
1837         \ifdefequal{#4}{\@gls@default@value}%
1838         {%
1839             \edef\@gls@value{\expandonce{#1}}%
1840             \csuse{gls@assign@#3@field}{#2}{\@gls@value}%
1841         }%
1842         {%
1843             \expandafter\@gls@startswithexpandonce#4\relax\relax\gls@endcheck
1844             {%
1845                 \@gls@expand@field{#2}{#3}{#4}%
1846             }%
1847             {%
1848                 \csuse{gls@assign@#3@field}{#2}{#4}%
1849             }%
1850         }%
1851     }%
1852     {%
1853         \ifdefequal{#4}{\@gls@default@value}%
1854         {%
1855             \@gls@expand@field{#2}{#3}{#1}%
1856         }%
1857         {%
1858             \@gls@expand@field{#2}{#3}{#4}%
1859         }%
1860     }%
1861 }

```

swithexpandonce

```
1862 \def\@gls@expandonce{\expandonce}
1863 \def\@gls@startswithexpandonce#1#2\gls@endcheck#3#4{%
1864   \def\@gls@tmp{#1}%
1865   \ifdefequal{\@gls@expandonce}{\@gls@tmp}{#3}{#4}%
1866 }
```

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.

```
1867 \let\gls@assign@field\@gls@expand@fields
```

gls@expand@fields

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

```
1868 \newcommand*{\gls@expand@fields}{%
1869   \let\gls@assign@field\@gls@expand@fields
1870 }
```

snoexpand@fields

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

```
1871 \newcommand*{\gls@snoexpand@fields}{%
1872   \let\gls@assign@field\@gls@snoexpand@fields
1873 }
```

ewglossaryentry

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

```
1874 \newrobustcmd{\newglossaryentry}[2]{%
```

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

```
1875   \glsdoifnoexists{#1}%
1876   {%
1877     \gls@defglossaryentry{#1}{#2}%
1878   }%
1879 }
```

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.

```
1880 \newcommand*{\gls@defdocnewglossaryentry}{%
1881   \let\gls@checkseeallowed\gls@checkseeallowed@preambleonly
1882   \let\newglossaryentry\new@glossaryentry
1883 }
```

deglossaryentry

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

```
1884 \newrobustcmd{\provideglossaryentry}[2]{%
```

```

1885 \ifglentryexists{#1}%
1886 {}%
1887 {%
1888   \gls@defglossaryentry{#1}{#2}%
1889 }%
1890 }
1891 \onlypreamble{\provideglossaryentry}

```

`w@glossaryentry` For use in document environment.

```

1892 \newrobustcmd{\new@glossaryentry}[2]{%
1893   \ifundef\@gls@deffile
1894   {%
1895     \global\newwrite\@gls@deffile
1896     \immediate\openout\@gls@deffile=\jobname.glsdefs
1897   }%
1898   {}%
1899   \ifglentryexists{#1}{}%
1900   {%
1901     \gls@defglossaryentry{#1}{#2}%
1902   }%
1903   \@gls@writedef{#1}%
1904 }
1905 \AtBeginDocument
1906 {
1907   \@gls@enablesavenonumberlist
1908   \makeatletter
1909   \InputIfFileExists{\jobname.glsdefs}{-}{-}%
1910   \makeatother
1911   \gls@defdocnewglossaryentry
1912 }
1913 \AtEndDocument{\ifdef\@gls@deffile{\closeout\@gls@deffile}{-}}

```

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

```

1914 \newcommand*{\@gls@writedef}[1]{%
1915   \immediate\write\@gls@deffile
1916   {%
1917     \string\ifglentryexists{#1}{}\glspercentchar^^J%
1918     \expandafter\@gobble\string\{\glspercentchar^^J%
1919     \string\gls@defglossaryentry{\glsdetoklabel{#1}}\glspercentchar^^J%
1920     \expandafter\@gobble\string\{\glspercentchar%
1921   }%

```

Write key value information:

```

1922 \@for\@gls@map:=\@gls@keymap\do
1923 {%
1924   \letcs\glo@value{glo@\glsdetoklabel{#1}}\expandafter\@secondoftwo\@gls@map}%
1925   \ifdef\glo@value
1926   {%
1927     \@onelevel@sanitize\glo@value
1928     \immediate\write\@gls@deffile

```

```

1929      {%
1930          \expandafter\@firstoftwo\@gls@map
1931          =\expandafter\@gobble\string\{\@glo@value\expandafter\@gobble\string\},%
1932          \glspercentchar
1933      }%
1934  }%
1935  {}%
1936  }%

```

Provide hook:

```

1937  \gls.writedefhook
1938  \immediate\write\@gls@deffile
1939  {%
1940      \glspercentchar^^J%
1941      \expandafter\@gobble\string\}\glspercentchar^^J%
1942      \expandafter\@gobble\string\}\glspercentchar%
1943  }%
1944  }

```

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

```

1945 \newcommand*{\@gls@keymap}{%
1946   {name}{name},%
1947   {sort}{sortvalue},% unescaped sort value
1948   {type}{type},%
1949   {first}{first},%
1950   {firstplural}{firstpl},%
1951   {text}{text},%
1952   {plural}{plural},%
1953   {description}{desc},%
1954   {descriptionplural}{descplural},%
1955   {symbol}{symbol},%
1956   {symbolplural}{symbolplural},%
1957   {user1}{useri},%
1958   {user2}{userii},%
1959   {user3}{useriii},%
1960   {user4}{useriv},%
1961   {user5}{userv},%
1962   {user6}{uservi},%
1963   {long}{long},%
1964   {longplural}{longpl},%
1965   {short}{short},%
1966   {shortplural}{shortpl},%
1967   {counter}{counter},%
1968   {parent}{parent}%
1969 }

```

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

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

```
1970 \newcommand*{\@gls@fetchfield}[2]{%
```

Ensure user field name is fully expanded

```
1971 \edef\@gls@thisval{#2}%
```

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

```
1972 \@for\@gls@map:=\@gls@keymap\do{%
```

```
1973 \edef\@this@key{\expandafter\@firstoftwo\@gls@map}%
```

```
1974 \ifdefequal{\@this@key}{\@gls@thisval}%
```

```
1975 {%
```

Found it.

```
1976 \edef#1{\expandafter\@secondoftwo\@gls@map}%
```

Break out of loop.

```
1977 \@endfortrue
```

```
1978 }%
```

```
1979 {}%
```

```
1980 }%
```

```
1981 }
```

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.

```
1982 \newcommand*{\glsaddstoragekey}{\@ifstar\@sglsaddstoragekey\@glsaddstoragekey}
```

Starred version switches on expansion for this key.

```
1983 \newcommand*{\@sglsaddstoragekey}[1]{%
```

```
1984 \key@ifundefined{glossentry}{#1}%
```

```
1985 {%
```

```
1986 \expandafter\newcommand\expandafter*\expandafter
```

```
1987 {\csname gls@assign@#1@field\endcsname}[2]{%
```

```
1988 \@gls@expand@field{##1}{#1}{##2}%
```

```
1989 }%
```

```
1990 }%
```

```
1991 {}%
```

```
1992 \@glsaddstoragekey{#1}%
```

```
1993 }
```

Unstarred version doesn't override default expansion.

```
1994 \newcommand*{\@glsaddstoragekey}[3]{%
```

Check the specified key doesn't already exist.

```
1995 \key@ifundefined{glossentry}{#1}%
```

```
1996 {%
```

Set up the key.

```
1997 \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
```

```
1998 \appto\@gls@keymap{, #1}{#1}}%
```

Set the default value.

```
1999 \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
```

Assignment code.

```
2000 \appto\@newglossaryentryposthook{%  
2001 \letcs{\@glo@tmp}{@glo@#1}%  
2002 \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%  
2003 }%
```

Define the no-link commands.

```
2004 \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%  
2005 }%  
2006 {%  
2007 \PackageError{glossaries}{Key ‘#1’ already exists}{}%  
2008 }%  
2009 }
```

```
\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.

```
2010 \newcommand*{\glsaddkey}{\ifstar\@sglsaddkey\@glsaddkey}
```

Starred version switches on expansion for this key.

```
2011 \newcommand*{\@sglsaddkey}[1]{%  
2012 \key@ifundefined{glossentry}{#1}%  
2013 {%  
2014 \expandafter\newcommand\expandafter*\expandafter  
2015 {\csname gls@assign@#1@field\endcsname}[2]{%  
2016 \@gls@expand@field{##1}{#1}{##2}%  
2017 }%  
2018 }%  
2019 }%  
2020 \@glsaddkey{#1}%  
2021 }
```

Unstarred version doesn't override default expansion.

```
2022 \newcommand*{\@glsaddkey}[7]{%
```

Check the specified key doesn't already exist.

```
2023 \key@ifundefined{glossentry}{#1}%  
2024 {%
```

Set up the key.

```
2025 \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%  
2026 \appto\@gls@keymap{,}{#1}{#1}}%
```

Set the default value.

```
2027 \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
```

Assignment code.

```

2028 \appto\@newglossaryentryposthook{%
2029 \letcs{\@glo@tmp}{\@glo@#1}%
2030 \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
2031 }%

```

Define the no-link commands.

```

2032 \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
2033 \newcommand*{#4}[1]{\@Gls@entry@field{##1}{#1}}%

```

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

```

2034 \ifcsdef{@gls@user@#1@}%
2035 {%
2036 \PackageError{glossaries}%
2037 {Can't define '\string#5' as helper command
2038 '\expandafter\string\csname @gls@user@#1@endcsname' already exists}%
2039 }%
2040 }%
2041 {%
2042 \expandafter\newcommand\expandafter*\expandafter
2043 {\csname @gls@user@#1@endcsname}[2][1]{%
2044 \new@ifnextchar[%
2045 {\csuse{@gls@user@#1@}{##1}{##2}}%
2046 {\csuse{@gls@user@#1@}{##1}{##2}[]}}%
2047 \csdef{@gls@user@#1@}##1##2[##3]{%
2048 \@gls@field@link{##1}{##2}{#3{##2}##3}%
2049 }%
2050 \newrobustcmd*{#5}{%
2051 \expandafter\@gls@hyp@opt\csname @gls@user@#1@endcsname}%
2052 }%

```

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

```

2053 \ifcsdef{@Gls@user@#1@}%
2054 {%
2055 \PackageError{glossaries}%
2056 {Can't define '\string#6' as helper command
2057 '\expandafter\string\csname @Gls@user@#1@endcsname' already exists}%
2058 }%
2059 }%
2060 {%
2061 \expandafter\newcommand\expandafter*\expandafter
2062 {\csname @Gls@user@#1@endcsname}[2][1]{%
2063 \new@ifnextchar[%
2064 {\csuse{@Gls@user@#1@}{##1}{##2}}%
2065 {\csuse{@Gls@user@#1@}{##1}{##2}[]}}%
2066 \csdef{@Gls@user@#1@}##1##2[##3]{%
2067 \@gls@field@link{##1}{##2}{#4{##2}##3}%
2068 }%
2069 \newrobustcmd*{#6}{%

```



```

2070      \expandafter\@gls@hyp@opt\csname @Gls@user@#1\endcsname}%
2071  }%

  Finally the all caps version:
2072  \ifcsdef{@GLS@user@#1@}%
2073  {%
2074    \PackageError{glossaries}%
2075    {Can't define '\string#7' as helper command
2076    '\expandafter\string\csname @GLS@user@#1@\endcsname' already exists}%
2077    }%
2078  }%
2079  {%

2080  \expandafter\newcommand\expandafter*\expandafter
2081    {\csname @GLS@user@#1\endcsname}[2][{}]{%
2082    \new@ifnextchar[%
2083      {\csuse{@GLS@user@#1@}{##1}{##2}}%
2084      {\csuse{@GLS@user@#1@}{##1}{##2}[{}]}%
2085    \csdef{@GLS@user@#1@}##1##2[##3]{%
2086      \@gls@field@link{##1}{##2}{\mfirstucMakeUppercase{#3{##2}##3}}%
2087    }%
2088    \newrobustcmd*{#7}{%
2089      \expandafter\@gls@hyp@opt\csname @GLS@user@#1\endcsname}%
2090    }%
2091  }%
2092  {%
2093    \PackageError{glossaries}{Key '#1' already exists}{}%
2094  }%
2095 }

```

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

```

2096 \newcommand{\glsfieldxdef}[3]{%
2097   \glsdoifexists{#1}%
2098   {%
2099     \edef\@glo@label{\glsdetoklabel{#1}}%
2100     \ifcsdef{glo@\@glo@label @#2}%
2101     {%
2102       \expandafter\xdef\csname glo@\@glo@label @#2\endcsname{#3}%
2103     }%
2104     {%
2105       \PackageError{glossaries}{Key '#2' doesn't exist}{}%
2106     }%
2107   }%
2108 }

```

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

```

2109 \newcommand{\glsfielddedef}[3]{%
2110   \glsdoifexists{#1}%
2111   {%
2112     \edef\@glo@label{\glsdetoklabel{#1}}%
2113     \ifcsdef{glo@\@glo@label @#2}%
2114     {%
2115       \expandafter\edef\csname glo@\@glo@label @#2\endcsname{#3}%
2116     }%
2117     {%
2118       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2119     }%
2120   }%
2121 }

```

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

```

2122 \newcommand{\glsfieldgdef}[3]{%
2123   \glsdoifexists{#1}%
2124   {%
2125     \edef\@glo@label{\glsdetoklabel{#1}}%
2126     \ifcsdef{glo@\@glo@label @#2}%
2127     {%
2128       \expandafter\gdef\csname glo@\@glo@label @#2\endcsname{#3}%
2129     }%
2130     {%
2131       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2132     }%
2133   }%
2134 }

```

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

```

2135 \newcommand{\glsfieldddef}[3]{%
2136   \glsdoifexists{#1}%
2137   {%
2138     \edef\@glo@label{\glsdetoklabel{#1}}%
2139     \ifcsdef{glo@\@glo@label @#2}%
2140     {%
2141       \expandafter\def\csname glo@\@glo@label @#2\endcsname{#3}%
2142     }%
2143     {%
2144       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2145     }%
2146   }%

```

2147 }

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

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

```
2148 \newcommand{\glsfieldfetch}[3]{%
2149   \glsdoifexists{#1}%
2150   {%
2151     \edef\@glo@label{\glsdetoklabel{#1}}%
2152     \ifcsdef{glo@\@glo@label @#2}%
2153     {%
2154       \letcs#3{glo@\@glo@label @#2}%
2155     }%
2156     {%
2157       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2158     }%
2159   }%
2160 }
```

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

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

```
2161 \newcommand{\ifglsfieldeq}[5]{%
2162   \glsdoifexists{#1}%
2163   {%
2164     \edef\@glo@label{\glsdetoklabel{#1}}%
2165     \ifcsdef{glo@\@glo@label @#2}%
2166     {%
2167       \ifcsstring{glo@\@glo@label @#2}{#3}{#4}{#5}%
2168     }%
2169     {%
2170       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2171     }%
2172   }%
2173 }
```

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

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

```
2174 \newcommand{\ifglsfielddefeq}[5]{%
2175   \glsdoifexists{#1}%
2176   {%
2177     \edef\@glo@label{\glsdetoklabel{#1}}%
2178     \ifcsdef{glo@\@glo@label @#2}%
2179     {%
```

```

2180 \expandafter\ifdefstrequal
2181 \csname glo@\@glo@label @#2\endcsname{#3}{#4}{#5}%
2182 }%
2183 {%
2184 \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2185 }%
2186 }%
2187 }

```

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

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

```

2188 \newcommand{\ifglsfieldcseq}[5]{%
2189 \glsdoifexists{#1}%
2190 {%
2191 \edef\@glo@label{\glsdetoklabel{#1}}%
2192 \ifcsdef{glo@\@glo@label @#2}%
2193 {%
2194 \ifcsstrequal{glo@\@glo@label @#2}{#3}{#4}{#5}%
2195 }%
2196 {%
2197 \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2198 }%
2199 }%
2200 }

```

`gls.writedefhook`

```

2201 \newcommand*{\gls.writedefhook}{}

```

`gls@assign@desc`

```

2202 \newcommand*{\gls@assign@desc}[1]{%
2203 \gls@assign@field{#1}{desc}{\@glo@desc}%
2204 \gls@assign@field{\@glo@desc}{#1}{descplural}{\@glo@descplural}%
2205 }

```

`ewglossaryentry`

```

2206 \newcommand{\longnewglossaryentry}[3]{%
2207 \glsdoifnoexists{#1}%
2208 {%
2209 \bgroup
2210 \let\@org@newglossaryentryprehook\@newglossaryentryprehook
2211 \long\def\@newglossaryentryprehook{%
2212 \long\def\@glo@desc{#3\leavevmode\unskip\nopostdesc}%
2213 \@org@newglossaryentryprehook
2214 }%
2215 \renewcommand*{\gls@assign@desc}[1]{%
2216 \global\cslet{glo@\glsdetoklabel{#1}@desc}{\@glo@desc}%

```

```

2217         \global\cslet{glo@\glsdetoklabel{#1}@descplural}{\@glo@desc}%
2218     }
2219     \gls@defglossaryentry{#1}{#2}%
2220 \egroup
2221 }
2222 }

```

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

```
2223 \@onlypreamble{\longnewglossaryentry}
```

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

```

2224 \newcommand{\longprovideglossaryentry}[3]{%
2225   \ifglentryexists{#1}{}%
2226   {\longnewglossaryentry{#1}{#2}{#3}}%
2227 }
2228 \@onlypreamble{\longprovideglossaryentry}

```

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

Defines a new entry without checking if it already exists.

```
2229 \newcommand{\gls@defglossaryentry}[2]{%
```

Prevent any further use of `\GlsSetQuote`:

```
2230   \let\GlsSetQuote\gls@nosetquote
```

Store label

```
2231   \edef\@glo@label{\glsdetoklabel{#1}}%
```

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

```
2232   \let\glslabel\@glo@label
```

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

```
2233   \let\@glo@name\@glsnname
```

```
2234   \let\@glo@desc\@glsnodesc
```

```
2235   \let\@glo@descplural\@gls@default@value
```

```
2236   \let\@glo@type\@gls@default@value
```

```
2237   \let\@glo@symbol\@gls@default@value
```

```
2238   \let\@glo@symbolplural\@gls@default@value
```

```
2239   \let\@glo@text\@gls@default@value
```

```
2240   \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.)

```
2241   \let\@glo@first\@gls@default@value
```

```
2242   \let\@glo@firstplural\@gls@default@value
```

Set the default sort:

```
2243 \let\@glo@sort\@gls@default@value
```

Set the default counter:

```
2244 \let\@glo@counter\@gls@default@value
```

```
2245 \def\@glo@see{}%
```

```
2246 \def\@glo@parent{}%
```

```
2247 \def\@glo@prefix{}%
```

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

```
2248 \@gls@initnonnumberlist
```

```
2249 \def\@glo@useri{}%
```

```
2250 \def\@glo@userii{}%
```

```
2251 \def\@glo@useriii{}%
```

```
2252 \def\@glo@useriv{}%
```

```
2253 \def\@glo@userv{}%
```

```
2254 \def\@glo@uservi{}%
```

```
2255 \def\@glo@short{}%
```

```
2256 \def\@glo@shortpl{}%
```

```
2257 \def\@glo@long{}%
```

```
2258 \def\@glo@longpl{}%
```

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

```
2259 \@newglossaryentryprehook
```

Extract key-val information from third parameter:

```
2260 \setkeys{glossentry}{#2}%
```

Check there is a default glossary.

```
2261 \ifundef\glsdefaultttype
```

```
2262 {%
```

```
2263 \PackageError{glossaries}%
```

```
2264 {No default glossary type (have you used 'nomain' by mistake?)}%
```

```
2265 {If you use package option 'nomain' you must define
```

```
2266 a new glossary before you can define entries}%
```

```
2267 }%
```

```
2268 {}}%
```

Assign type. This must be fully expandable

```
2269 \gls@assign@field{\glsdefaultttype}{\@glo@label}{type}{\@glo@type}%
```

```
2270 \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.

```
2271 \ifcsundef{glolist@\@glo@type}%
```

```
2272 {%
```

```
2273 \PackageError{glossaries}%
```

```

2274      {Glossary type ‘\@glo@type’ has not been defined}%
2275      {You need to define a new glossary type, before making entries
2276      in it}%
2277  }%
2278  {%

  Check if it's an ignored glossary
2279      \ifignoredglossary\@glo@type
2280      {%

    The description may be omitted for an entry in an ignored glossary.
2281      \ifx\@glo@desc\@glsnodesc
2282      \let\@glo@desc\@empty
2283      \fi
2284  }%
2285  {%
2286  }%
2287      \protected@edef\@glo@list@{\csname glo@list@\@glo@type\endcsname}%
2288      \expandafter\edef\csname glo@list@\@glo@type\endcsname{%
2289      \@glo@list@{\@glo@label},}%
2290  }%

  Initialise level to 0.
2291      \gls@level=0\relax

  Has this entry been assigned a parent?
2292      \ifx\@glo@parent\@empty

    Doesn't have a parent. Set \glo@<label>@parent to empty.
2293      \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{%
2294      \else

    Has a parent. Check to ensure this entry isn't its own parent.
2295      \ifdefequal\@glo@label\@glo@parent%
2296      {%
2297          \PackageError{glossaries}{Entry ‘\@glo@label’ can't be its own parent}{}%
2298          \def\@glo@parent{}%
2299          \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{%
2300          }%
2301          {%

    Check the parent exists:
2302          \ifglsentryexists{\@glo@parent}%
2303          {%

    Parent exists. Set \glo@<label>@parent.
2304          \expandafter\edef\csname glo@\@glo@label @parent\endcsname{%
2305          \@glo@parent}%

    Determine level.
2306          \gls@level=\csname glo@\@glo@parent @level\endcsname\relax
2307          \advance\gls@level by 1\relax

```

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

```
2308      \ifx\@glo@name\@glsnoname
2309      \expandafter\let\expandafter\@glo@name
2310      \csname glo@\@glo@parent @name\endcsname
```

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

```
2311      \ifx\@glo@plural\@gls@default@value
2312      \expandafter\let\expandafter\@glo@plural
2313      \csname glo@\@glo@parent @plural\endcsname
2314      \fi
2315      \fi
2316      }%
2317      {%
```

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

```
2318      \PackageError{glossaries}%
2319      {%
2320      Invalid parent '\@glo@parent'
2321      for entry '\@glo@label' - parent doesn't exist%
2322      }%
2323      {%
2324      Parent entries must be defined before their children%
2325      }%
2326      \def\@glo@parent{}%
2327      \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%
2328      }%
2329      }%
2330      \fi
```

Set the level for this entry

```
2331      \expandafter\xdef\csname glo@\@glo@label @level\endcsname{\number\gls@level}%
```

Define commands associated with this entry:

```
2332      \gls@assign@field{\@glo@name}{\@glo@label}{sortvalue}{\@glo@sort}%
2333      \letcs\@glo@sort{glo@\@glo@label @sortvalue}%
2334      \gls@assign@field{\@glo@name}{\@glo@label}{text}{\@glo@text}%
2335      \expandafter\gls@assign@field\expandafter
2336      {\csname glo@\@glo@label @text\endcsname\glspluralsuffix}%
2337      {\@glo@label}{plural}{\@glo@plural}%
2338      \expandafter\gls@assign@field\expandafter
2339      {\csname glo@\@glo@label @text\endcsname}%
2340      {\@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.

```
2341      \ifx\@glo@first\@gls@default@value
2342      \expandafter\gls@assign@field\expandafter
2343      {\csname glo@\@glo@label @plural\endcsname}%
2344      {\@glo@label}{firstpl}{\@glo@firstplural}%
2345      \else
2346      \expandafter\gls@assign@field\expandafter
```



```

2347         {\csname glo@\@glo@label @first\endcsname\glspluralsuffix}%
2348         {\@glo@label}{firstpl}{\@glo@firstplural}%
2349     \fi

2350     \ifcsundef{@glo@type@\@glo@type @counter}%
2351     {%
2352         \def\@glo@defaultcounter{\glscounter}%
2353     }%
2354     {%
2355         \letcs\@glo@defaultcounter{@glo@type@\@glo@type @counter}%
2356     }%
2357     \gls@assign@field{\@glo@defaultcounter}{\@glo@label}{counter}{\@glo@counter}%
2358     \gls@assign@field{}{\@glo@label}{useri}{\@glo@useri}%
2359     \gls@assign@field{}{\@glo@label}{userii}{\@glo@userii}%
2360     \gls@assign@field{}{\@glo@label}{useriii}{\@glo@useriii}%
2361     \gls@assign@field{}{\@glo@label}{useriv}{\@glo@useriv}%
2362     \gls@assign@field{}{\@glo@label}{userv}{\@glo@userv}%
2363     \gls@assign@field{}{\@glo@label}{uservi}{\@glo@uservi}%
2364     \gls@assign@field{}{\@glo@label}{short}{\@glo@short}%
2365     \gls@assign@field{}{\@glo@label}{shortpl}{\@glo@shortpl}%
2366     \gls@assign@field{}{\@glo@label}{long}{\@glo@long}%
2367     \gls@assign@field{}{\@glo@label}{longpl}{\@glo@longpl}%
2368     \ifx\@glo@name\@glsnoname
2369         \@glsnoname
2370         \let\@glo@name\@gls@default@value
2371     \fi
2372     \gls@assign@field{}{\@glo@label}{name}{\@glo@name}%

```

Set default numberlist if not defined:

```

2373     \ifcsundef{glo@\@glo@label @numberlist}%
2374     {%
2375         \csxdef{glo@\@glo@label @numberlist}{%
2376             \noexpand\@gls@missingnumberlist{\@glo@label}}%
2377     }%
2378     {}%

```

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

```

2379     \@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.

```

2380     \def\@glo@@desc{\@glo@first}%
2381     \ifx\@glo@desc\@glo@@desc
2382         \let\@glo@desc\@glo@first
2383     \fi
2384     \ifx\@glo@desc\@glsnodesc
2385         \@glsnodesc
2386         \let\@glo@desc\@gls@default@value
2387     \fi
2388     \gls@assign@desc{\@glo@label}%

```

Set the sort key for this entry:

```
2389 \gls@defsort{\@glo@type}{\@glo@label}%
2390 \def\@glo@@symbol{\@glo@text}%
2391 \ifx\@glo@symbol\@glo@@symbol
2392 \let\@glo@symbol\@glo@text
2393 \fi
2394 \gls@assign@field{\relax}{\@glo@label}{symbol}{\@glo@symbol}%
2395 \expandafter
2396 \gls@assign@field\expandafter
2397 {\csname glo@\@glo@label @symbol\endcsname}
2398 {\@glo@label}{symbolplural}{\@glo@symbolplural}%
```

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

```
2399 \expandafter\xdef\csname glo@\@glo@label @flagfalse\endcsname{%
2400 \noexpand\global
2401 \noexpand\let\expandafter\noexpand
2402 \csname ifglo@\@glo@label @flag\endcsname\noexpand\iffalse
2403 }%
2404 \expandafter\xdef\csname glo@\@glo@label @flagtrue\endcsname{%
2405 \noexpand\global
2406 \noexpand\let\expandafter\noexpand
2407 \csname ifglo@\@glo@label @flag\endcsname\noexpand\iftrue
2408 }%
2409 \csname glo@\@glo@label @flagfalse\endcsname
```

Sort out any cross-referencing if required.

```
2410 \ifdefvoid\@glo@see
2411 {}%
2412 {%
2413 \protected@edef\@do@glsee{%
2414 \noexpand\@gls@fixbraces\noexpand\@glo@list\@glo@see
2415 \noexpand\@nil
2416 \noexpand\expandafter\noexpand\@glsee\noexpand\@glo@list{\@glo@label}}%
2417 \@do@glsee
2418 }%
```

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

```
2419 \ifignoredglossary\@glo@type
2420 {%
2421 \csdef{glo@\@glo@label @index}{}%
2422 }
2423 {%
2424 \do@glo@storeentry{\@glo@label}%
2425 }%
```

Define entry counters if enabled:

```
2426 \@newglossaryentry@defcounters
```

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

```

2427 \newglossaryentryposthook
2428 }

aryentryprehook Allow extra information to be added to glossary entries:
2429 \newcommand*{\@newglossaryentryprehook}{}

ryentryposthook Allow extra information to be added to glossary entries:
2430 \newcommand*{\@newglossaryentryposthook}{}

try@defcounters
2431 \newcommand*{\@newglossaryentry@defcounters}{}

\glsmoveentry Moves entry whose label is given by first argument to the glossary named in the second argu-
ment.
2432 \newcommand*{\glsmoveentry}[2]{%
2433 \edef\@glo@thislabel{\glsdetoklabel{#1}}%
2434 \edef\@glo@type{\csname glo@\@glo@thislabel @type\endcsname}%
2435 \def\@glo@list{,%}
2436 \for\glsentries[\@glo@type]{\@glo@label}%
2437 {%
2438 \ifdefequal\@glo@thislabel\@glo@label
2439 {}{\eappto\@glo@list{\@glo@label,%}}%
2440 }%
2441 \cslet\@glo@list\@glo@type{\@glo@list}%
2442 \csdef\@glo@\@glo@thislabel @type{#2}%
2443 }

ssaryentryfield Indicate what command should be used to display each entry in the glossary. (This enables
the glossaries-accsupp package to use \accsuppglossaryentryfield instead.)
2444 \ifglxindy
2445 \newcommand*{\@glossaryentryfield}{\string\glossentry}
2446 \else
2447 \newcommand*{\@glossaryentryfield}{\string\glossentry}
2448 \fi

rysubentryfield Indicate what command should be used to display each subentry in the glossary. (This en-
ables the glossaries-accsupp package to use \accsuppglossarysubentryfield instead.)
2449 \ifglxindy
2450 \newcommand*{\@glossarysubentryfield}{%
2451 \string\subglossentry}
2452 \else
2453 \newcommand*{\@glossarysubentryfield}{%
2454 \string\subglossentry}
2455 \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.)

```
2456 \newcommand{\@glo@storeentry}[1]{%
```

Escape makeindex/xindy special characters in the label:

```
2457 \edef\@glo@esclabel{#1}%
```

```
2458 \@gls@checkmkidxchars\@glo@esclabel
```

Get the sort string and escape any special characters

```
2459 \protected@edef\@glo@sort{\csname glo@#1@sort\endcsname}%
```

```
2460 \@gls@checkmkidxchars\@glo@sort
```

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

```
2461 \@gls@checkmkidxchars\@glo@prefix
```

Get the parent, if one exists

```
2462 \edef\@glo@parent{\csname glo@#1@parent\endcsname}%
```

Write the information to the glossary file.

```
2463 \ifglxindy
```

Store using xindy syntax.

```
2464 \ifx\@glo@parent\@empty
```

Entry doesn't have a parent

```
2465 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
```

```
2466 (\string"\@glo@sort\string" %
```

```
2467 \string"\@glo@prefix\@glossaryentryfield{\@glo@esclabel}\string") %
```

```
2468 }%
```

```
2469 \else
```

Entry has a parent

```
2470 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
```

```
2471 \csname glo@\@glo@parent @index\endcsname
```

```
2472 (\string"\@glo@sort\string" %
```

```
2473 \string"\@glo@prefix\@glossarysubentryfield
```

```
2474 {\csname glo@#1@level\endcsname}{\@glo@esclabel}\string") %
```

```
2475 }%
```

```
2476 \fi
```

```
2477 \else
```

Store using makeindex syntax.

```
2478 \ifx\@glo@parent\@empty
```

Sanitize \@glo@prefix

```
2479 \@onelevel@sanitize\@glo@prefix
```

Entry doesn't have a parent

```
2480 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
```

```
2481 \@glo@sort\@gls@actualchar\@glo@prefix
```

```
2482 \@glossaryentryfield{\@glo@esclabel}%
```

```

2483     }%
2484   \else
      Entry has a parent
2485     \expandafter\protected\xdef\csname glo@#1@index\endcsname{%
2486       \csname glo@\@glo@parent @index\endcsname\@gls@levelchar
2487       \@glo@sort\@gls@actualchar\@glo@prefix
2488       \@glossarysubentryfield
2489       {\csname glo@#1@level\endcsname}{\@glo@esclabel}%
2490     }%
2491   \fi
2492 \fi
2493 }

```

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`

```

2494 \AtBeginDocument{%
2495   \ifpackageloaded{amsmath}%
2496   {\let\gls@ifnotmeasuring\@gls@ifnotmeasuring}%
2497   }%
2498 }
2499 \newcommand*\@gls@ifnotmeasuring[1]{%
2500   \ifmeasuring@
2501   \else
2502     #1%
2503   \fi
2504 }
2505 \newcommand*\gls@ifnotmeasuring[1]{#1}

```

`\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.

```

2506 \newcommand*\glsreset[1]{%
2507   \gls@ifnotmeasuring
2508   {%
2509     \glsdoifexists{#1}%
2510     {%
2511       \@glsreset{#1}%
2512     }%
2513   }%
2514 }

```

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

```

2515 \newcommand*{\glslocalreset}[1]{%
2516   \gls@ifnotmeasuring
2517   {%
2518     \glsdoifexists{#1}%
2519     {%
2520       \@glslocalreset{#1}%
2521     }%
2522   }%
2523 }

```

`\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.

```

2524 \newcommand*{\glsunset}[1]{%
2525   \gls@ifnotmeasuring
2526   {%
2527     \glsdoifexists{#1}%
2528     {%
2529       \@glsunset{#1}%
2530     }%
2531   }%
2532 }

```

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

```

2533 \newcommand*{\glslocalunset}[1]{%
2534   \gls@ifnotmeasuring
2535   {%
2536     \glsdoifexists{#1}%
2537     {%
2538       \@glslocalunset{#1}%
2539     }%
2540   }%
2541 }

```

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

```

2542 \newcommand*{\@glslocalunset}{\@glslocalunset}

```

`@@glslocalunset` Local unset without checks.

```

2543 \newcommand*{\@glslocalunset}[1]{%
2544   \expandafter\let\csname ifglo@glsdetoklabel{#1}@flag\endcsname\iftrue
2545 }

```

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

```

2546 \newcommand*{\@glsunset}{\@glsunset}

```

`\@@glsunset` Global unset without checks.

```

2547 \newcommand*{\@@glsunset}[1]{%
2548   \expandafter\global\csname glo@glsdetoklabel{#1}@flagtrue\endcsname
2549 }

```

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

```
2550 \newcommand*{\@glslocalreset}{\@@glslocalreset}
```

`\@@glslocalreset` Local reset without checks.

```
2551 \newcommand*{\@@glslocalreset}[1]{%
2552   \expandafter\let\csname ifglo@glsdetoklabel{#1}@flag\endcsname\iffalse
2553 }
```

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

```
2554 \newcommand*{\@glsreset}{\@@glsreset}
```

`\@@glsreset` Global reset without checks.

```
2555 \newcommand*{\@@glsreset}[1]{%
2556   \expandafter\global\csname glo@glsdetoklabel{#1}@flagfalse\endcsname
2557 }
```

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

`\glsresetall`

```
2558 \newcommand*{\glsresetall}[1][\@glo@types]{%
2559   \forallglsentries[#1]{\@glsentry}%
2560   {%
2561     \glsreset{\@glsentry}%
2562   }%
2563 }
```

As above, but with only a local effect:

`\glslocalresetall`

```
2564 \newcommand*{\glslocalresetall}[1][\@glo@types]{%
2565   \forallglsentries[#1]{\@glsentry}%
2566   {%
2567     \glslocalreset{\@glsentry}%
2568   }%
2569 }
```

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

`\glsunsetall`

```
2570 \newcommand*{\glsunsetall}[1][\@glo@types]{%
2571   \forallglsentries[#1]{\@glsentry}%
2572   {%
2573     \glsunset{\@glsentry}%
2574   }%
2575 }
```

As above, but with only a local effect:

lslocalunsetall

```
2576 \newcommand*{\glslocalunsetall}[1][\@glo@types]{%
2577   \forallglsentries[#1]{\@glsentry}%
2578   {%
2579     \glslocalunset{\@glsentry}%
2580   }%
2581 }
```

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.

```
2582 \newcommand*{\@newglossaryentry@defcounters}{%
2583   \csdef{glo@\@glo@label @currcount}{0}%
2584   \csdef{glo@\@glo@label @prevcount}{0}%
2585 }
```

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

```
2586 \newcommand*{\glsenableentrycount}{%
```

Enable new entry fields.

```
2587   \let\@newglossaryentry@defcounters\@newglossaryentry@defcounters
```

Disable `\newglossaryentry` in the document environment.

```
2588   \renewcommand*{\gls@defdocnewglossaryentry}{%
2589     \renewcommand*\newglossaryentry[2]{%
2590       \PackageError{glossaries}{\string\newglossaryentry\space
2591         may only be used in the preamble when entry counting has
2592         been activated}{If you use \string\glsenableentrycount\space
2593         you must place all entry definitions in the preamble not in
2594         the document environment}%
2595     }%
2596   }%
```

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

```
2597   \newcommand*{\glsentrycurrcount}[1]{%
2598     \ifcsundef{glo@\glsdetoklabel{##1}@currcount}%
2599     {0}{\@gls@entry@field{##1}{currcount}}%
2600   }%
2601   \newcommand*{\glsentryprevcount}[1]{%
```



```

2602 \ifcsundef{glo@\glsdetoklabel{##1}@prevcount}%
2603 {0}{\@gls@entry@field{##1}{prevcount}}}%
2604 }%

```

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

```

2605 \renewcommand*{\@glsunset}[1]{%
2606   \@glsunset{##1}%
2607   \@gls@increment@currcount{##1}%
2608 }%
2609 \renewcommand*{\@glslocalunset}[1]{%
2610   \@glslocalunset{##1}%
2611   \@gls@local@increment@currcount{##1}%
2612 }%
2613 \renewcommand*{\@glsreset}[1]{%
2614   \@glsreset{##1}%
2615   \csgdef{glo@\glsdetoklabel{##1}@currcount}{0}%
2616 }%
2617 \renewcommand*{\@glslocalreset}[1]{%
2618   \@glslocalreset{##1}%
2619   \csdef{glo@\glsdetoklabel{##1}@currcount}{0}%
2620 }%

```

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.)

```

2621 \def\@cgl's@##1##2[##3]{%
2622   \ifnum\glsentryprevcount{##2}=1\relax
2623     \cgl'sformat{##2}{##3}%
2624     \glsunset{##2}%
2625   \else
2626     \@gls@{##1}{##2}[##3]%
2627   \fi
2628 }%

```

Similarly for the analogous commands. No case change plural:

```

2629 \def\@cgl'spl@##1##2[##3]{%
2630   \ifnum\glsentryprevcount{##2}=1\relax
2631     \cgl'splformat{##2}{##3}%
2632     \glsunset{##2}%
2633   \else
2634     \@cgl'spl@{##1}{##2}[##3]%
2635   \fi
2636 }%

```

First letter uppercase singular:

```

2637 \def\@cGls@##1##2[##3]{%
2638   \ifnum\glsentryprevcount{##2}=1\relax
2639     \cGlsformat{##2}{##3}%
2640     \glsunset{##2}%
2641   \else
2642     \@Gls@{##1}{##2}[##3]%
2643   \fi

```

2644 }%

First letter uppercase plural:

```
2645 \def\@cGlspl@##1##2[##3]{%
2646   \ifnum\glsentryprevcount{##2}=1\relax
2647     \cGlsplformat{##2}{##3}%
2648     \glsunset{##2}%
2649   \else
2650     \@Glspl@{##1}{##2}[##3]%
2651   \fi
2652 }%
```

Write information to aux file at the end of the document

```
2653 \AtEndDocument{\@gls@write@entrycounts}%
```

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

```
2654 \renewcommand*{\@gls@entry@count}[2]{%
2655   \csxdef{glo@\glsdetoklabel{##1}@prevcount}{##2}%
2656 }%
```

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

```
2657 \let\glsenableentrycount\relax
2658 }
2659 \@onlypreamble\glsenableentrycount
```

ement@currcount

```
2660 \newcommand*{\@gls@increment@currcount}[1]{%
2661   \csxdef{glo@\glsdetoklabel{##1}@currcount}{%
2662     \number\numexpr\glsentrycurrcount{##1}+1}%
2663 }
```

ement@currcount

```
2664 \newcommand*{\@gls@local@increment@currcount}[1]{%
2665   \csxdef{glo@\glsdetoklabel{##1}@currcount}{%
2666     \number\numexpr\glsentrycurrcount{##1}+1}%
2667 }
```

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.)

```
2668 \newcommand*{\@gls@write@entrycounts}{%
2669   \immediate\write\@auxout
2670     {\string\providecommand*{\string\@gls@entry@count}[2]{}}%
2671   \forallglsentries{\@glsentry}{%
2672     \ifglsused{\@glsentry}%
2673     {\immediate\write\@auxout
2674       {\string\@gls@entry@count{\@glsentry}{\glsentrycurrcount{\@glsentry}}}%
2675     }%
2676 }
```

```
2676 }%
2677 }
```

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

```
2678 \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.)

```
2679 \newrobustcmd*{\cgl}{\@gls@hyp@opt\@cgl}
```

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

```
2680 \newcommand*{\@cgl}[2][{}]{%
2681   \new@ifnextchar[{\@cgl@{#1}{#2}}{\@cgl@{#1}{#2}[]}%
2682 }
```

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

```
2683 \def\@cgl@#1#2[#3]{%
2684   \GlossariesWarning{\string\cgl\space is defaulting to
2685     \string\gls\space since you haven't enabled entry counting}%
2686   \@gls@{#1}{#2}[#3]%
2687 }
```

`\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.

```
2688 \newcommand*{\cglformat}[2][{}]{%
2689   \ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}#2%
2690 }
```

`\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.)

```
2691 \newrobustcmd*{\cGl}{\@gls@hyp@opt\@cGl}
```

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

```
2692 \newcommand*{\@cGl}[2][{}]{%
2693   \new@ifnextchar[{\@cGl@{#1}{#2}}{\@cGl@{#1}{#2}[]}%
2694 }
```

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

```
2695 \def\@cGl@#1#2[#3]{%
2696   \GlossariesWarning{\string\cGl\space is defaulting to
2697     \string\Gls\space since you haven't enabled entry counting}%
2698   \@Gls@{#1}{#2}[#3]%
2699 }
```

`\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.

```
2700 \newcommand*{\cGlsformat}[2]{%
2701   \ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}#2%
2702 }
```

`\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.)

```
2703 \newrobustcmd*{\cglsp1}{\@gls@hyp@opt\@cglsp1}
```

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

```
2704 \newcommand*{\@cglsp1}[2][ ]{%
2705   \new@ifnextchar[\@cglsp1@{#1}{#2}}{\@cglsp1@{#1}{#2}[ ]}%
2706 }
```

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

```
2707 \def\@cglsp1@#1#2[#3]{%
2708   \GlossariesWarning{\string\cglsp1\space is defaulting to
2709     \string\glsp1\space since you haven't enabled entry counting}%
2710   \@glsp1@{#1}{#2}[#3]%
2711 }
```

`\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.

```
2712 \newcommand*{\cglsp1format}[2]{%
2713   \ifglshaslong{#1}{\glsp1long{#1}}{\glsp1firstplural{#1}}#2%
2714 }
```

`\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.)

```
2715 \newrobustcmd*{\cGlspl}{\@gls@hyp@opt\@cGlspl}
```

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

```
2716 \newcommand*{\@cGlspl}[2][ ]{%
2717   \new@ifnextchar[\@cGlspl@{#1}{#2}}{\@cGlspl@{#1}{#2}[ ]}%
2718 }
```

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

```
2719 \def\@cGlspl@#1#2[#3]{%
2720   \GlossariesWarning{\string\cGlspl\space is defaulting to
2721     \string\Glspl\space since you haven't enabled entry counting}%
2722   \@Glspl@{#1}{#2}[#3]%
2723 }
```

`\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.

```
2724 \newcommand*{\cGlsplformat}[2]{%
2725   \ifglshaslong{#1}{\Glsentrylongpl{#1}}{\Glsentryfirstplural{#1}}#2%
2726 }
```

1.10 Loading files containing glossary entries

Glossary entries can be defined in an external file. These external files can contain `\newglossaryentry` and `\newacronym` commands.¹

`\loadglsentries[⟨type⟩]{⟨filename⟩}`

This command will input the file using `\input`. The optional argument specifies to which glossary the entries should be assigned if they haven't used the type key. If the optional argument is not specified, the default glossary is used. Only those entries used in the document (via `\glslink`, `\gls`, `\glspl` and uppercase variants or `\glsadd` and `\glsaddall` will appear in the glossary). The mandatory argument is the filename (with or without `.tex` extension).

`\loadglsentries`

```
2727 \newcommand*{\loadglsentries}[2][\@gls@default]{%
2728   \let\@gls@default\glsdefaulttype
2729   \def\glsdefaulttype{#1}\input{#2}%
2730   \let\glsdefaulttype\@gls@default
2731 }
```

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

```
2732 \@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`

```
2733 \newcommand*{\glstextformat}[1]{#1}
```

`\glsentryfmt` As from version 3.11a, the way in which an entry is displayed is now governed by `\glsentryfmt`. This doesn't take any arguments. The required information is set by commands like `\gls`. To

¹and any other valid \TeX code that can be used in the preamble.

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

```
2734 \newcommand*{\glsentryfmt}{%
2735   \@gls@default@entryfmt\glsdisplayfirst\glsdisplay
2736 }
```

Format that provides backwards compatibility:

```
2737 \newcommand*{\@gls@default@entryfmt}[2]{%
2738   \ifdefempty\glscustomtext
2739   {%
2740     \glsifplural
2741     {%
```

Plural form

```
2742     \glscapscase
2743     {%
```

Don't adjust case

```
2744     \ifglsused\glslabel
2745     {%
```

Subsequent use

```
2746         #2{\glsentryplural{\glslabel}}%
2747         {\glsentrydescplural{\glslabel}}%
2748         {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2749     }%
2750     {%
```

First use

```
2751         #1{\glsentryfirstplural{\glslabel}}%
2752         {\glsentrydescplural{\glslabel}}%
2753         {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2754     }%
2755     }%
2756     {%
```

Make first letter upper case

```
2757     \ifglsused\glslabel
2758     {%
```

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.)

```
2759     \ifbool{glscompatible-3.07}%
2760     {%
2761       \protected@edef\@glo@etext{%
2762         #2{\glsentryplural{\glslabel}}%
2763         {\glsentrydescplural{\glslabel}}%
2764         {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2765       \xmakefirstuc\@glo@etext
2766     }%
```

```

2767      {%
2768      #2{\Glsentryplural{\glslabel}}%
2769      {\glsentrydescplural{\glslabel}}%
2770      {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2771      }%
2772      }%
2773      {%

```

First use

```

2774      \ifbool{glscompatible-3.07}%
2775      {%
2776      \protected@edef\@glo@etext{%
2777      #1{\Glsentryfirstplural{\glslabel}}%
2778      {\glsentrydescplural{\glslabel}}%
2779      {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2780      \xmakefirstuc\@glo@etext
2781      }%
2782      {%
2783      #1{\Glsentryfirstplural{\glslabel}}%
2784      {\glsentrydescplural{\glslabel}}%
2785      {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2786      }%
2787      }%
2788      }%
2789      {%

```

Make all upper case

```

2790      \ifglsused\glslabel
2791      {%

```

Subsequent use

```

2792      \mfirstucMakeUppercase{#2{\glsentryplural{\glslabel}}%
2793      {\glsentrydescplural{\glslabel}}%
2794      {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2795      }%
2796      {%

```

First use

```

2797      \mfirstucMakeUppercase{#1{\glsentryfirstplural{\glslabel}}%
2798      {\glsentrydescplural{\glslabel}}%
2799      {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2800      }%
2801      }%
2802      }%
2803      {%

```

Singular form

```

2804      \glscapscase
2805      {%

```

Don't adjust case

```

2806      \ifglsused\glslabel

```

```

2807      {%
Subsequent use
2808      #2{\glsentrytext{\glslabel}}%
2809      {\glsentrydesc{\glslabel}}%
2810      {\glsentrysymbol{\glslabel}}{\glsinsert}%
2811      }%
2812      {%

First use
2813      #1{\glsentryfirst{\glslabel}}%
2814      {\glsentrydesc{\glslabel}}%
2815      {\glsentrysymbol{\glslabel}}{\glsinsert}%
2816      }%
2817      }%
2818      {%

Make first letter upper case
2819      \ifglused\glslabel
2820      {%

Subsequent use
2821      \ifbool{glscompatible-3.07}%
2822      {%
2823      \protected@edef\@glo@etext{%
2824      #2{\glsentrytext{\glslabel}}%
2825      {\glsentrydesc{\glslabel}}%
2826      {\glsentrysymbol{\glslabel}}{\glsinsert}}%
2827      \xmakefirstuc\@glo@etext
2828      }%
2829      {%
2830      #2{\Glsentrytext{\glslabel}}%
2831      {\glsentrydesc{\glslabel}}%
2832      {\glsentrysymbol{\glslabel}}{\glsinsert}%
2833      }%
2834      }%
2835      {%

First use
2836      \ifbool{glscompatible-3.07}%
2837      {%
2838      \protected@edef\@glo@etext{%
2839      #1{\glsentryfirst{\glslabel}}%
2840      {\glsentrydesc{\glslabel}}%
2841      {\glsentrysymbol{\glslabel}}{\glsinsert}}%
2842      \xmakefirstuc\@glo@etext
2843      }%
2844      {%
2845      #1{\Glsentryfirst{\glslabel}}%
2846      {\glsentrydesc{\glslabel}}%
2847      {\glsentrysymbol{\glslabel}}{\glsinsert}%

```



```

2848     }%
2849     }%
2850     }%
2851     {%

```

Make all upper case

```

2852     \ifglsused\glslabel
2853     {%

```

Subsequent use

```

2854         \mfirstucMakeUppercase{#2{\glsentrytext{\glslabel}}}%
2855         {\glsentrydesc{\glslabel}}}%
2856         {\glsentrysymbol{\glslabel}}{\glsinsert}}}%
2857     }%
2858     {%

```

First use

```

2859         \mfirstucMakeUppercase{#1{\glsentryfirst{\glslabel}}}%
2860         {\glsentrydesc{\glslabel}}}%
2861         {\glsentrysymbol{\glslabel}}{\glsinsert}}}%
2862     }%
2863     }%
2864     }%
2865     }%
2866     {%

```

Custom text provided in \glsdisp

```

2867     \ifglsused{\glslabel}%
2868     {%

```

Subsequent use

```

2869         #2{\glscustomtext}%
2870         {\glsentrydesc{\glslabel}}}%
2871         {\glsentrysymbol{\glslabel}}{}%
2872     }%
2873     {%

```

First use

```

2874         #1{\glscustomtext}%
2875         {\glsentrydesc{\glslabel}}}%
2876         {\glsentrysymbol{\glslabel}}{}%
2877     }%
2878     }%
2879 }

```

`\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.

```

2880 \newcommand*{\glsenentryfmt}{%
2881   \ifdefempty\glscustomtext
2882   {%
2883     \glsifplural
2884     {%

```

Plural form

2885 \glscapscase
2886 {%

Don't adjust case

2887 \ifglused\glslabel
2888 {%

Subsequent use

2889 \glstentryplural{\glslabel}\glinsert
2890 }%
2891 {%

First use

2892 \glstentryfirstplural{\glslabel}\glinsert
2893 }%
2894 }%
2895 {%

Make first letter upper case

2896 \ifglused\glslabel
2897 {%

Subsequent use.

2898 \Glstentryplural{\glslabel}\glinsert
2899 }%
2900 {%

First use

2901 \Glstentryfirstplural{\glslabel}\glinsert
2902 }%
2903 }%
2904 {%

Make all upper case

2905 \ifglused\glslabel
2906 {%

Subsequent use

2907 \mfirstucMakeUppercase
2908 {\glstentryplural{\glslabel}\glinsert}%
2909 }%
2910 {%

First use

2911 \mfirstucMakeUppercase
2912 {\glstentryfirstplural{\glslabel}\glinsert}%
2913 }%
2914 }%
2915 }%
2916 {%

Singular form

```
2917 \glscapscase
2918 {%
```

Don't adjust case

```
2919 \ifglused\glslabel
2920 {%
```

Subsequent use

```
2921 \glentrytext{\glslabel}\glsinsert
2922 }%
2923 {%
```

First use

```
2924 \glentryfirst{\glslabel}\glsinsert
2925 }%
2926 }%
2927 {%
```

Make first letter upper case

```
2928 \ifglused\glslabel
2929 {%
```

Subsequent use

```
2930 \Glentrytext{\glslabel}\glsinsert
2931 }%
2932 {%
```

First use

```
2933 \Glentryfirst{\glslabel}\glsinsert
2934 }%
2935 }%
2936 {%
```

Make all upper case

```
2937 \ifglused\glslabel
2938 {%
```

Subsequent use

```
2939 \mfirstucMakeUppercase{\glentrytext{\glslabel}\glsinsert}%
2940 }%
2941 {%
```

First use

```
2942 \mfirstucMakeUppercase{\glentryfirst{\glslabel}\glsinsert}%
2943 }%
2944 }%
2945 }%
2946 }%
2947 {%
```

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

```
2948 \glscustomtext\glsinsert
```

```

2949 }%
2950 }

```

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

```

2951 \newcommand*{\glsgenacfmt}{%
2952   \ifdefempty\glscustomtext
2953     {%
2954       \ifglused\glslabel
2955         {%

```

Subsequent use:

```

2956       \glsifplural
2957       {%

```

Subsequent plural form:

```

2958       \glscapscase
2959       {%

```

Subsequent plural form, don't adjust case:

```

2960       \acronymfont{\glentryshortpl{\glslabel}}\glsinsert
2961       }%
2962       {%

```

Subsequent plural form, make first letter upper case:

```

2963       \acronymfont{\Glentryshortpl{\glslabel}}\glsinsert
2964       }%
2965       {%

```

Subsequent plural form, all caps:

```

2966       \mfirstucMakeUppercase
2967       {\acronymfont{\glentryshortpl{\glslabel}}\glsinsert}%
2968       }%
2969       }%
2970       {%

```

Subsequent singular form

```

2971       \glscapscase
2972       {%

```

Subsequent singular form, don't adjust case:

```

2973       \acronymfont{\glentryshort{\glslabel}}\glsinsert
2974       }%
2975       {%

```

Subsequent singular form, make first letter upper case:

```

2976       \acronymfont{\Glentryshort{\glslabel}}\glsinsert
2977       }%
2978       {%

```

Subsequent singular form, all caps:

```

2979       \mfirstucMakeUppercase
2980       {\acronymfont{\glentryshort{\glslabel}}\glsinsert}%

```

```

2981      }%
2982      }%
2983      }%
2984      {%

```

First use:

```

2985      \glsifplural
2986      {%

```

First use plural form:

```

2987      \glscapscase
2988      {%

```

First use plural form, don't adjust case:

```

2989      \genplacrformat{\glslabel}{\glsinsert}%
2990      }%
2991      {%

```

First use plural form, make first letter upper case:

```

2992      \Genplacrformat{\glslabel}{\glsinsert}%
2993      }%
2994      {%

```

First use plural form, all caps:

```

2995      \mfirstucMakeUppercase
2996      {\genplacrformat{\glslabel}{\glsinsert}}%
2997      }%
2998      }%
2999      {%

```

First use singular form

```

3000      \glscapscase
3001      {%

```

First use singular form, don't adjust case:

```

3002      \genacrformat{\glslabel}{\glsinsert}%
3003      }%
3004      {%

```

First use singular form, make first letter upper case:

```

3005      \Genacrformat{\glslabel}{\glsinsert}%
3006      }%
3007      {%

```

First use singular form, all caps:

```

3008      \mfirstucMakeUppercase
3009      {\genacrformat{\glslabel}{\glsinsert}}%
3010      }%
3011      }%
3012      }%
3013      }%
3014      {%

```

User supplied text.

```
3015 \glscustomtext
3016 }%
3017 }
```

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

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

```
3018 \newcommand*{\genacrfullformat}[2]{%
3019 \glentrylong{#1}#2\space
3020 (\protect\firstacronymfont{\glentryshort{#1}})%
3021 }
```

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

As above but makes the first letter upper case.

```
3022 \newcommand*{\Genacrfullformat}[2]{%
3023 \protected@edef\gls@text{\genacrfullformat{#1}{#2}}%
3024 \xmakefirstuc\gls@text
3025 }
```

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

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

```
3026 \newcommand*{\genplacrfullformat}[2]{%
3027 \glentrylongpl{#1}#2\space
3028 (\protect\firstacronymfont{\glentryshortpl{#1}})%
3029 }
```

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

As above but makes the first letter upper case.

```
3030 \newcommand*{\Genplacrfullformat}[2]{%
3031 \protected@edef\gls@text{\genplacrfullformat{#1}{#2}}%
3032 \xmakefirstuc\gls@text
3033 }
```

glsdisplayfirst Deprecated. Kept for backward compatibility.

```
3034 \newcommand*{\glsdisplayfirst}[4]{#1#4}
```

`\glsdisplay` Deprecated. Kept for backward compatibility.

```
3035 \newcommand*{\glsdisplay}[4]{#1#4}
```

`\defglsdisplay` Deprecated. Kept for backward compatibility.

```
3036 \newcommand*{\defglsdisplay}[2][\glsdefaulttype]{%
3037   \GlossariesWarning{\string\defglsdisplay\space is now obsolete.^^J
3038   Use \string\defglsentryfmt\space instead}%
3039   \expandafter\def\csname gls@#1@display\endcsname##1##2##3##4{#2}%
3040   \edef\@gls@doentrydef{%
3041     \noexpand\defglsentryfmt[#1]{%
3042       \noexpand\ifcsdef{gls@#1@displayfirst}%
3043       {%
3044         \noexpand\@@gls@default@entryfmt
3045         {\noexpand\csuse{gls@#1@displayfirst}}}%
3046         {\noexpand\csuse{gls@#1@display}}}%
3047       }%
3048       {%
3049         \noexpand\@@gls@default@entryfmt
3050         {\noexpand\glsdisplayfirst}%
3051         {\noexpand\csuse{gls@#1@display}}}%
3052       }%
3053     }%
3054   }%
3055   \@gls@doentrydef
3056 }
```

`glsdisplayfirst` Deprecated. Kept for backward compatibility.

```
3057 \newcommand*{\defglsdisplayfirst}[2][\glsdefaulttype]{%
3058   \GlossariesWarning{\string\defglsdisplayfirst\space is now obsolete.^^J
3059   Use \string\defglsentryfmt\space instead}%
3060   \expandafter\def\csname gls@#1@displayfirst\endcsname##1##2##3##4{#2}%
3061   \edef\@gls@doentrydef{%
3062     \noexpand\defglsentryfmt[#1]{%
3063       \noexpand\ifcsdef{gls@#1@display}%
3064       {%
3065         \noexpand\@@gls@default@entryfmt
3066         {\noexpand\csuse{gls@#1@displayfirst}}}%
3067         {\noexpand\csuse{gls@#1@display}}}%
3068       }%
3069       {%
3070         \noexpand\@@gls@default@entryfmt
3071         {\noexpand\csuse{gls@#1@displayfirst}}}%
3072         {\noexpand\glsdisplay}%
3073       }%
3074     }%
3075   }%
3076   \@gls@doentrydef
3077 }
```

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.

```
3078 \define@key{glslink}{counter}{%
3079   \ifcsundef{c@#1}%
3080   {%
3081     \PackageError{glossaries}%
3082     {There is no counter called ‘#1’}%
3083     {%
3084       The counter key should have the name of a valid counter
3085       as its value%
3086     }%
3087   }%
3088   {%
3089     \def\@gls@counter{#1}%
3090   }%
3091 }
```

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.

```
3092 \define@key{glslink}{format}{%
3093   \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.

```
3094 \define@boolkey{glslink}{hyper}[true]{}
```

Initialise hyper key.

```
3095 \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.

```
3096 \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.

```
3097 \newcommand*{\glslinkvar}[3]{#1}
```

`\glsifhyper` Now deprecated.

```
3098 \newcommand*{\glsifhyper}[2]{%
3099 \glslinkvar{#1}{#2}{#1}%
3100 \GlossariesWarning{\string\glsifhyper\space is deprecated. Did
3101 you mean \string\glsifhyperon\space or \string\glslinkvar?}%
3102 }
```

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

```
3103 \newcommand*{\@gls@hyp@opt}[1]{%
3104 \let\glslinkvar\@firstofthree
3105 \let\@gls@hyp@opt@cs#1\relax
3106 \@ifstar{\s@gls@hyp@opt}%
3107 {\@ifnextchar+{\@firstoftwo{\p@gls@hyp@opt}}{#1}}%
3108 }
```

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

```
3109 \newcommand*{\s@gls@hyp@opt}[1] []{%
3110 \let\glslinkvar\@secondofthree
3111 \@gls@hyp@opt@cs[hyper=false,#1]}
```

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

```
3112 \newcommand*{\p@gls@hyp@opt}[1] []{%
3113 \let\glslinkvar\@thirdofthree
3114 \@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`

```
3115 \newrobustcmd*{\glslink}{%
3116 \@gls@hyp@opt\@gls@@link
3117 }
```

`\@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.

```
3118 \newcommand*{\@gls@@link}[3] [] {%
3119 \glsdoifexistsordo{#2}%
3120 {%
3121 \let\do@gls@link@checkfirsthyper\relax
3122 \@gls@link[#1]{#2}{#3}%
3123 }%
```

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

```
3124 \glstextformat{#3}%
3125 }%
```

```
3126 \glspostlinkhook
3127 }
```

`glspostlinkhook`

```
3128 \newcommand*{\glspostlinkhook}{}
3129 % \end{macrocode}
3130 %\end{macro}
3131 %
3132 %
3133 %\begin{macro}{\@gls@link@checkfirsthyper}
3134 % Check for first use and switch off \gloskey[glslink]{hyper} key
3135 % if hyperlink not wanted. (Should be off if first use and
3136 % hyper=false is on or if first use and both the entry is in an acronym
3137 % list and the acrfootnote setting is on.)
3138 % This assumes the glossary type is stored in \cs{glstype} and the
3139 % label is stored in \cs{glslabel}.
3140 %\changes{4.08}{2014-07-30}{new}
3141 % \begin{macrocode}
3142 \newcommand*{\@gls@link@checkfirsthyper}{%
3143 \ifglsused{\glslabel}%
3144 {%
3145 }%
3146 {%
3147 \gls@checkisacronymlist\glstype
3148 \ifglshyperfirst
3149 \if@glsisacronymlist
3150 \ifglsacrfootnote
3151 \KV@glslink@hyperfalse
3152 \fi
```

```

3153     \fi
3154     \else
3155         \KV@glslink@hyperfalse
3156     \fi
3157 }%

    Allow user to hook into this
3158 \glslinkcheckfirsthyperhook
3159 }

linkfirsthyperhook    Allow used to hook into the \@gls@link@checkfirsthyper macro
3160 \newcommand*{\glslinkcheckfirsthyperhook}{}

linkpostsetkeys
3161 \newcommand*{\glslinkpostsetkeys}{}

\glsifhyperon    Check the value of the hyper key:
3162 \newcommand{\glsifhyperon}[2]{\ifKV@glslink@hyper#1\else#2\fi}

ablehyperinlist    Disable hyperlink if in the “nohyper” list.
3163 \newcommand*{\do@glstdisablehyperinlist}{%
3164     \expandafter\DTLifinlist\expandafter{\glstype}{\@gls@nohyperlist}%
3165     {\KV@glslink@hyperfalse}}%
3166 }

lt@glslink@opts    Hook to set default options for \@glslink.
3167 \newcommand*{\@gls@setdefault@glslink@opts}{}

\@gls@link
3168 \def\@gls@link[#1]#2#3{%
    Inserting \leavevmode suggested by Donald Arseneau (avoids problem with tabularx).
3169     \leavevmode
3170     \edef\glslabel{\glsdetoklabel{#2}}%

    Save options in \@gls@link@opts and label in \@gls@link@label
3171     \def\@gls@link@opts{#1}%
3172     \let\@gls@link@label\glslabel

3173     \def\@glsnumberformat{glsnumberformat}%
3174     \edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%

    If this is in one of the “nohypertypes” glossaries, suppress the hyperlink by default
3175     \edef\glstype{\csname glo@\glslabel @type\endcsname}%

    Save original setting
3176     \let\org@ifKV@glslink@hyper\ifKV@glslink@hyper

    Set defaults:
3177     \@gls@setdefault@glslink@opts

```

Switch off hyper setting if the glossary type has been identified in nohyperlist.

```
3178 \do@gl:disablehyperinlist
```

Macros must set this before calling \@gls@link. The commands that check the first use flag should set this to \@gls@link@checkfirsthyper otherwise it should be set to \relax.

```
3179 \do@gls@link@checkfirsthyper
```

```
3180 \setkeys{glslink}{#1}%
```

Add a hook for the user to customise things after the keys have been set.

```
3181 \glslinkpostsetkeys
```

Store the entry's counter in \theglsentrycounter

```
3182 \@gls@saveentrycounter
```

Define sort key if necessary:

```
3183 \@gls@setsort{\glslabel}%
```

(De-tok'ing done by \@do@wrglossary)

```
3184 \@do@wrglossary{#2}%
```

```
3185 \ifKV@glslink@hyper
```

```
3186 \glslink{\glolinkprefix\glslabel}{\glstextformat{#3}}%
```

```
3187 \else
```

```
3188 \glsdonohyperlink{\glolinkprefix\glslabel}{\glstextformat{#3}}%
```

```
3189 \fi
```

Restore original setting

```
3190 \let\ifKV@glslink@hyper\org@ifKV@glslink@hyper
```

```
3191 }
```

\glolinkprefix

```
3192 \newcommand*{\glolinkprefix}{glo:}
```

glsentrycounter Set default value of entry counter

```
3193 \def\glsentrycounter{\glscounter}%
```

aveentrycounter Need to check if using equation counter in align environment:

```
3194 \newcommand*{\@gls@saveentrycounter}{%
```

```
3195 \def\@gls@Hcounter{}}%
```

Are we using equation counter?

```
3196 \ifthenelse{\equal{\@gls@counter}{equation}}{%
```

```
3197 {
```

If we're in align environment, \xatlevel@ will be defined. (Can't test for \@currentvir as may be inside an inner environment.)

```
3198 \ifcsundef{xatlevel@}%
```

```
3199 {%
```

```
3200 \edef\theglsentrycounter{\expandafter\noexpand
```

```
3201 \csname the\@gls@counter\endcsname}%
```

```
3202 }%
```

```

3203   {%
3204     \ifx\xatlevel@\@empty
3205       \edef\theglentrycounter{\expandafter\noexpand
3206         \csname the\@gls@counter\endcsname}%
3207     \else
3208       \savecounters@
3209       \advance\c@equation by 1\relax
3210       \edef\theglentrycounter{\csname the\@gls@counter\endcsname}%

```

Check if hyperref version of this counter

```

3211     \ifcsundef{theH\@gls@counter}%
3212     {%
3213       \def\@gls@Hcounter{\theglentrycounter}%
3214     }%
3215     {%
3216       \def\@gls@Hcounter{\csname theH\@gls@counter\endcsname}%
3217     }%
3218     \protected@edef\theHglentrycounter{\@gls@Hcounter}%
3219     \restorecounters@
3220   \fi
3221 }%
3222 }%
3223 {%

```

Not using equation counter so no special measures:

```

3224   \edef\theglentrycounter{\expandafter\noexpand
3225     \csname the\@gls@counter\endcsname}%
3226 }%

```

Check if hyperref version of this counter

```

3227 \ifx\@gls@Hcounter\@empty
3228 \ifcsundef{theH\@gls@counter}%
3229 {%
3230   \def\theHglentrycounter{\theglentrycounter}%
3231 }%
3232 {%
3233   \protected@edef\theHglentrycounter{\expandafter\noexpand
3234     \csname theH\@gls@counter\endcsname}%
3235 }%
3236 \fi
3237 }

```

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.

```

3238 \def\@set@glo@numformat#1#2#3#4{%
3239   \expandafter\@glo@check@mkiidxrangechar#3\@nil
3240   \protected@edef#1{%

```

```

3241 \@glo@prefix setentrycounter[#4]{#2}%
3242 \expandafter\string\csname\@glo@suffix\endcsname
3243 }%
3244 \@gls@checkmkidxchars#1%
3245 }

```

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.

```

3246 \def\@glo@check@mkidxrangechar#1#2\@nil{%
3247 \if#1(\relax
3248 \def\@glo@prefix{(%
3249 \if\relax#2\relax
3250 \def\@glo@suffix{glsnumberformat}%
3251 \else
3252 \def\@glo@suffix{#2}%
3253 \fi
3254 \else
3255 \if#1)\relax
3256 \def\@glo@prefix{)%
3257 \if\relax#2\relax
3258 \def\@glo@suffix{glsnumberformat}%
3259 \else
3260 \def\@glo@suffix{#2}%
3261 \fi
3262 \else
3263 \def\@glo@prefix{}\def\@glo@suffix{#1#2}%
3264 \fi
3265 \fi}

```

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

```

3266 \newcommand*{\@gls@escbsdq}[1]{%
3267 \def\@gls@checkedmkidx{%
3268 \let\gls@xdystring=#1\relax
3269 \@onelevel@sanitize\gls@xdystring
3270 \edef\do@gls@xdycheckbackslash{%
3271 \noexpand\@gls@xdycheckbackslash\gls@xdystring\noexpand\@nil
3272 \@backslashchar\@backslashchar\noexpand\null}%
3273 \do@gls@xdycheckbackslash
3274 \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdystring}%
3275 \def\@gls@checkedmkidx{%
3276 \expandafter\@gls@xdycheckquote\gls@xdystring\@nil""\null
3277 \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdystring}%

```

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

```

3278 \@for\@gls@tmp:=\gls@protected@pagefmts\do
3279 {%
3280 \edef\@gls@sanitized@tmp{\expandafter\@gobble\string\\ \expandonce\@gls@tmp}%

```

```

3281 \@onelevel@sanitize\@gls@sanitized@tmp
3282 \edef\gls@dosubst{%
3283   \noexpand\DTLsubstituteall\noexpand\gls@xdyststring
3284   {\@gls@sanitized@tmp}{\expandonce\@gls@tmp}%
3285 }%
3286 \gls@dosubst
3287 }%

```

Assign to required control sequence

```

3288 \let#1=\gls@xdyststring
3289 }

```

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

checkmkidxchars

```

3290 \newcommand{\@gls@checkmkidxchars}[1]{%
3291   \ifglxsindy
3292     \@gls@escbsdq{#1}%
3293   \else
3294     \def\@gls@checkedmkidx{%
3295       \expandafter\@gls@checkquote#1\@nil""\null
3296       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3297     \def\@gls@checkedmkidx{%
3298       \expandafter\@gls@checkescquote#1\@nil\""\null
3299       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3300     \def\@gls@checkedmkidx{%
3301       \expandafter\@gls@checkescactual#1\@nil\?\?\null
3302       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3303     \def\@gls@checkedmkidx{%
3304       \expandafter\@gls@checkactual#1\@nil??\null
3305       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3306     \def\@gls@checkedmkidx{%
3307       \expandafter\@gls@checkbar#1\@nil||\null
3308       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3309     \def\@gls@checkedmkidx{%
3310       \expandafter\@gls@checkescbar#1\@nil\\|\null
3311       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3312     \def\@gls@checkedmkidx{%
3313       \expandafter\@gls@checklevel#1\@nil!!\null
3314       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3315     \fi
3316 }

```

Update the control sequence and strip trailing \@nil:

s@updatechecked

```

3317 \def\@gls@updatechecked#1\@nil#2{\def#2{#1}}

```

\@gls@tmpb Define temporary token

```

3318 \newtoks\@gls@tmpb

```

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

```
3319 \def\@gls@checkquote#1"#2"#3\null{%
3320   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3321   \toks@={#1}%
3322   \ifx\null#2\null
3323   \ifx\null#3\null
3324     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3325     \def\@gls@checkquote{\relax}%
3326   \else
3327     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3328       \@gls@quotechar\@gls@quotechar\@gls@quotechar\@gls@quotechar}%
3329     \def\@gls@checkquote{\@gls@checkquote#3\null}%
3330   \fi
3331 \else
3332   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3333     \@gls@quotechar\@gls@quotechar}%
3334   \ifx\null#3\null
3335     \def\@gls@checkquote{\@gls@checkquote#2""\null}%
3336   \else
3337     \def\@gls@checkquote{\@gls@checkquote#2"#3\null}%
3338   \fi
3339 \fi
3340 \@gls@checkquote
3341 }
```

s@checkescquote Do the same for \":

```
3342 \def\@gls@checkescquote#1\"#2\"#3\null{%
3343   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3344   \toks@={#1}%
3345   \ifx\null#2\null
3346   \ifx\null#3\null
3347     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3348     \def\@gls@checkescquote{\relax}%
3349   \else
3350     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3351       \@gls@quotechar\string\"@gls@quotechar
3352       \@gls@quotechar\string\"@gls@quotechar}%
3353     \def\@gls@checkescquote{\@gls@checkescquote#3\null}%
3354   \fi
3355 \else
3356   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3357     \@gls@quotechar\string\"@gls@quotechar}%
3358   \ifx\null#3\null
3359     \def\@gls@checkescquote{\@gls@checkescquote#2\"\" \null}%
3360   \else
3361     \def\@gls@checkescquote{\@gls@checkescquote#2\"#3\null}%
3362   \fi
3363 \fi
3364 \@gls@checkescquote
```


3365 }

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

```
3366 \def\@gls@checkescactual#1\?#2\?#3\null{%
3367 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3368 \toks@={#1}%
3369 \ifx\null#2\null
3370 \ifx\null#3\null
3371 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3372 \def\@gls@checkescactual{\relax}%
3373 \else
3374 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3375 \@gls@quotechar\string\" \@gls@actualchar
3376 \@gls@quotechar\string\" \@gls@actualchar}%
3377 \def\@gls@checkescactual{\@gls@checkescactual#3\null}%
3378 \fi
3379 \else
3380 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3381 \@gls@quotechar\string\" \@gls@actualchar}%
3382 \ifx\null#3\null
3383 \def\@gls@checkescactual{\@gls@checkescactual#2\?\?\null}%
3384 \else
3385 \def\@gls@checkescactual{\@gls@checkescactual#2\?#3\null}%
3386 \fi
3387 \fi
3388 \@gls@checkescactual
3389 }
```

gls@checkeschar Similarly for \|:

```
3390 \def\@gls@checkeschar#1\|#2\|#3\null{%
3391 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3392 \toks@={#1}%
3393 \ifx\null#2\null
3394 \ifx\null#3\null
3395 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3396 \def\@gls@checkeschar{\relax}%
3397 \else
3398 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3399 \@gls@quotechar\string\" \@gls@encapchar
3400 \@gls@quotechar\string\" \@gls@encapchar}%
3401 \def\@gls@checkeschar{\@gls@checkeschar#3\null}%
3402 \fi
3403 \else
3404 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3405 \@gls@quotechar\string\" \@gls@encapchar}%
3406 \ifx\null#3\null
3407 \def\@gls@checkeschar{\@gls@checkeschar#2\|\|\null}%
3408 \else
3409 \def\@gls@checkeschar{\@gls@checkeschar#2\|#3\null}%

```

```

3410 \fi
3411 \fi
3412 \@@gls@checkescbar
3413 }

```

s@checkesclevel Similarly for \!:

```

3414 \def\@gls@checkesclevel#1\!#2\!#3\null{%
3415 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3416 \toks@={#1}%
3417 \ifx\null#2\null
3418 \ifx\null#3\null
3419 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3420 \def\@@gls@checkesclevel{\relax}%
3421 \else
3422 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3423 \@gls@quotechar\string"\@gls@levelchar
3424 \@gls@quotechar\string"\@gls@levelchar}%
3425 \def\@@gls@checkesclevel{\@gls@checkesclevel#3\null}%
3426 \fi
3427 \else
3428 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3429 \@gls@quotechar\string"\@gls@levelchar}%
3430 \ifx\null#3\null
3431 \def\@@gls@checkesclevel{\@gls@checkesclevel#2\!\!\null}%
3432 \else
3433 \def\@@gls@checkesclevel{\@gls@checkesclevel#2\!#3\null}%
3434 \fi
3435 \fi
3436 \@@gls@checkesclevel
3437 }

```

\@gls@checkbar and for |:

```

3438 \def\@gls@checkbar#1|#2|#3\null{%
3439 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3440 \toks@={#1}%
3441 \ifx\null#2\null
3442 \ifx\null#3\null
3443 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3444 \def\@@gls@checkbar{\relax}%
3445 \else
3446 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3447 \@gls@quotechar\@gls@encapchar\@gls@quotechar\@gls@encapchar}%
3448 \def\@@gls@checkbar{\@gls@checkbar#3\null}%
3449 \fi
3450 \else
3451 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3452 \@gls@quotechar\@gls@encapchar}%
3453 \ifx\null#3\null
3454 \def\@@gls@checkbar{\@gls@checkbar#2||\null}%

```

```

3455 \else
3456 \def\@gls@checkbar{\@gls@checkbar#2|#3\null}%
3457 \fi
3458 \fi
3459 \@gls@checkbar
3460 }

```

@gls@checklevel and for !:

```

3461 \def\@gls@checklevel#1!#2!#3\null{%
3462 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3463 \toks@={#1}%
3464 \ifx\null#2\null
3465 \ifx\null#3\null
3466 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3467 \def\@gls@checklevel{\relax}%
3468 \else
3469 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3470 \@gls@quotechar\@gls@levelchar\@gls@quotechar\@gls@levelchar}%
3471 \def\@gls@checklevel{\@gls@checklevel#3\null}%
3472 \fi
3473 \else
3474 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3475 \@gls@quotechar\@gls@levelchar}%
3476 \ifx\null#3\null
3477 \def\@gls@checklevel{\@gls@checklevel#2!!\null}%
3478 \else
3479 \def\@gls@checklevel{\@gls@checklevel#2!#3\null}%
3480 \fi
3481 \fi
3482 \@gls@checklevel
3483 }

```

gls@checkactual and for ?:

```

3484 \def\@gls@checkactual#1?#2?#3\null{%
3485 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3486 \toks@={#1}%
3487 \ifx\null#2\null
3488 \ifx\null#3\null
3489 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3490 \def\@gls@checkactual{\relax}%
3491 \else
3492 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3493 \@gls@quotechar\@gls@actualchar\@gls@quotechar\@gls@actualchar}%
3494 \def\@gls@checkactual{\@gls@checkactual#3\null}%
3495 \fi
3496 \else
3497 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3498 \@gls@quotechar\@gls@actualchar}%
3499 \ifx\null#3\null

```

```

3500     \def\@gls@checkactual{\@gls@checkactual#2??\null}%
3501     \else
3502     \def\@gls@checkactual{\@gls@checkactual#2?#3\null}%
3503     \fi
3504     \fi
3505     \@gls@checkactual
3506 }

```

s@xdycheckquote As before but for use with xindy

```

3507 \def\@gls@xdycheckquote#1"#2"#3\null{%
3508   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3509   \toks@={#1}%
3510   \ifx\null#2\null
3511     \ifx\null#3\null
3512       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3513       \def\@gls@xdycheckquote{\relax}%
3514     \else
3515       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3516         \string"\string"}%
3517       \def\@gls@xdycheckquote{\@gls@xdycheckquote#3\null}%
3518     \fi
3519   \else
3520     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3521       \string"}%
3522     \ifx\null#3\null
3523       \def\@gls@xdycheckquote{\@gls@xdycheckquote#2""\null}%
3524     \else
3525       \def\@gls@xdycheckquote{\@gls@xdycheckquote#2"#3\null}%
3526     \fi
3527     \fi
3528     \@gls@xdycheckquote
3529 }

```

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

```

3530 \edef\def\@gls@xdycheckbackslash{%
3531   \noexpand\def\noexpand\@gls@xdycheckbackslash##1\@backslashchar
3532     ##2\@backslashchar##3\noexpand\null{%
3533     \noexpand\@gls@tmpb=\noexpand\expandafter
3534       {\noexpand\@gls@checkedmkidx}%
3535     \noexpand\toks@={##1}%
3536     \noexpand\ifx\noexpand\null##2\noexpand\null
3537       \noexpand\ifx\noexpand\null##3\noexpand\null
3538         \noexpand\edef\noexpand\@gls@checkedmkidx{%
3539           \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
3540         \noexpand\def\noexpand\@gls@xdycheckbackslash{\relax}%
3541       \noexpand\else
3542         \noexpand\edef\noexpand\@gls@checkedmkidx{%
3543           \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
3544           \@backslashchar\@backslashchar\@backslashchar\@backslashchar}%

```

```

3545 \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3546   \noexpand\@gls@xdycheckbackslash##3\noexpand\null}%
3547 \noexpand\fi
3548 \noexpand\else
3549 \noexpand\edef\noexpand\@gls@checkedmkidx{%
3550   \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
3551   \@backslashchar\@backslashchar}%
3552 \noexpand\ifx\noexpand\null##3\noexpand\null
3553 \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3554   \noexpand\@gls@xdycheckbackslash##2\@backslashchar
3555   \@backslashchar\noexpand\null}%
3556 \noexpand\else
3557 \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3558   \noexpand\@gls@xdycheckbackslash##2\@backslashchar
3559   ##3\noexpand\null}%
3560 \noexpand\fi
3561 \noexpand\fi
3562 \noexpand\@gls@xdycheckbackslash
3563 }%
3564 }

```

Now go ahead and define \@gls@xdycheckbackslash

```

3565 \def\@gls@xdycheckbackslash

```

lsdohypertarget

```

3566 \newlength\gls@tmplen
3567 \newcommand*\@glsdohypertarget}[2]{%
3568   \settoheight{\gls@tmplen}{#2}%
3569   \raisebox{\gls@tmplen}{\hypertarget{#1}{}}#2%
3570 }

```

\glsdohyperlink

```

3571 \newcommand*\@glsdohyperlink}[2]{\hyperlink{#1}{#2}}

```

lsdonohyperlink

```

3572 \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.

```

3573 \ifcsundef{hyperlink}%
3574 {%
3575   \let\@glslink\glsdonohyperlink
3576 }%
3577 {%
3578   \let\@glslink\glsdohyperlink
3579 }

```

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

```
3580 \ifcsundef{hypertarget}%
3581 {%
3582   \let\@glstarget\@secondoftwo
3583 }%
3584 {%
3585   \let\@glstarget\glsdohypertarget
3586 }
```

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

`\glsdisablehyper`

```
3587 \newcommand{\glsdisablehyper}{%
3588   \KV@glslink@hyperfalse
3589   \let\@glslink\glsdonohyperlink
3590   \let\@glstarget\@secondoftwo
3591 }
```

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

`\glsenablehyper`

```
3592 \newcommand{\glsenablehyper}{%
3593   \KV@glslink@hypertrue
3594   \let\@glslink\glsdohyperlink
3595   \let\@glstarget\glsdohypertarget
3596 }
```

Provide some convenience commands if not already defined:

```
3597 \providecommand{\@firstofthree}[3]{#1}
3598 \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`

```
3599 \newrobustcmd*{\gls}{\@glshyp@opt\@gls}
```

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

`\@gls`

```
3600 \newcommand*{\@gls}[2] [] {%
3601   \new@ifnextchar[{\@gls@{#1}{#2}}{\@gls@{#1}{#2} []}]%
3602 }
```

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

```
3603 \def\@gls@#1#2[#3]{%
3604   \glsdoifexists{#2}%
3605   {%
3606     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3607     \let\glsifplural\@secondoftwo
3608     \let\glsupcase\@firstofthree
3609     \let\glscustomtext\@empty
3610     \def\glsinsert{#3}%
```

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

```
3611   \def\@glo@text{\csname gls@\gls@type @entryfmt\endcsname}%
```

Call `\@gls@link`. If footnote package option has been used and the glossary type is `\acronymtype`, suppress hyperlink for first use. Likewise if the `hyperfirst=false` package option is used.

```
3612   \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3613   \ifKV@glslink@local
3614     \glslocalunset{#2}%
3615   \else
3616     \glsunset{#2}%
3617   \fi
3618   }%
```

```
3619   \glspostlinkhook
3620 }
```

`\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`

```
3621 \newrobustcmd*{\Gls}{\@gls@hyp@opt\@Gls}
```

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

```
3622 \newcommand*{\@Gls}[2] [] {%
3623   \new@ifnextchar[{\@Gls@{#1}{#2}}{\@Gls@{#1}{#2} []}]%
3624 }
```

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

```
3625 \def\@Gls@#1#2[#3]{%
3626   \glsdoifexists{#2}%
3627   {%
3628     \let\do@glsl@link@checkfirsthyper\@glsl@link@checkfirsthyper

3629     \let\glsl@plural\@secondoftwo
3630     \let\glscapscase\@secondofthree
3631     \let\glscustomtext\@empty
3632     \def\glslinsert{#3}%
```

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

```
3633   \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%
```

Call `\@glsl@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.

```
3634   \@glsl@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3635   \ifKV@glsl@link@local
3636     \glsl@localunset{#2}%
3637   \else
3638     \glslunset{#2}%
3639   \fi
3640 }%

3641 \glspostlinkhook
3642 }
```

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

`\GLS`

```
3643 \newrobustcmd*{\GLS}{\@glsl@hyp@opt\@GLS}
```

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

```
3644 \newcommand*{\@GLS}[2][\@GLS@#1]{%
3645   \new@ifnextchar[\@GLS@{#1}{#2}}{\@GLS@{#1}{#2}[]}%
3646 }
```

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

```
3647 \def\@GLS@#1#2[#3]{%
3648   \glsdoifexists{#2}%
3649   {%
3650     \let\do@glsl@link@checkfirsthyper\@glsl@link@checkfirsthyper

3651     \let\glsl@plural\@secondoftwo
3652     \let\glscapscase\@thirdofthree
3653     \let\glscustomtext\@empty
3654     \def\glslinsert{#3}%
```


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

```
3655 \def\@glo@text{\csname gls@\gls@type @entryfmt\endcsname}%
```

Call `\@gls@link`. If footnote package option has been used and the glossary type is `\acronym@type`, suppress hyperlink for first use. Likewise if the `hyperfirst=false` package option is used.

```
3656 \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3657 \ifKV@gls@link@local
3658 \glslocalunset{#2}%
3659 \else
3660 \glsunset{#2}%
3661 \fi
3662 }%
```

```
3663 \gls@postlinkhook
3664 }
```

`\glspl` behaves in the same way as `\gls` except it uses the plural form.

`\glspl`

```
3665 \newrobustcmd*{\glspl}{\@gls@hyp@opt\@glspl}
```

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

```
3666 \newcommand*{\@glspl}[2] [] {%
3667 \new@ifnextchar[\@glspl@{#1}{#2}]{\@glspl@{#1}{#2} []}%
3668 }
```

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

```
3669 \def\@glspl@#1#2[#3] {%
3670 \glsdoifexists{#2}%
3671 {%
3672 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3673 \let\glsifplural\@firstoftwo
3674 \let\gls@scapscase\@firstofthree
3675 \let\gls@customtext\@empty
3676 \def\glsinsert{#3}%
```

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

```
3677 \def\@glo@text{\csname gls@\gls@type @entryfmt\endcsname}%
```

Call `\@gls@link`. If footnote package option has been used and the glossary type is `\acronym@type`, suppress hyperlink for first use. Likewise if the `hyperfirst=false` package option is used.

```
3678 \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3679 \ifKV@glslink@local
3680 \glsllocalunset{#2}%
3681 \else
3682 \glunset{#2}%
3683 \fi
3684 }%

3685 \glspostlinkhook
3686 }
```

`\Glspl` behaves in the same way as `\glsp1`, except that the first letter of the link text is converted to uppercase (as with `\Gls`, if the first letter has an accent, it will need to be grouped).

`\Glspl`

```
3687 \newrobustcmd*{\Glspl}{\@gls@hyp@opt\@Glspl}
```

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

```
3688 \newcommand*{\@Glspl}[2][{}]{%
3689 \new@ifnextchar[{\@Glspl@{#1}{#2}}{\@Glspl@{#1}{#2}[]}%
3690 }
```

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

```
3691 \def\@Glspl@#1#2[#3]{%
3692 \glsoifexists{#2}%
3693 {%
3694 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3695 \let\glsifplural\@firstoftwo
3696 \let\glscapscase\@secondofthree
3697 \let\glscustomtext\@empty
3698 \def\glsinsert{#3}%
```

Determine what the link text should be (this is stored in `\@glo@text`). This needs to be expanded so that the `\@glo@text` can be passed to `\xmakefirstuc`. Note that `\@gls@link` sets `\glstype`.

```
3699 \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.

```
3700 \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3701 \ifKV@glslink@local
3702 \glsllocalunset{#2}%
3703 \else
3704 \glunset{#2}%
3705 \fi
3706 }%
```

```

3707 \glspostlinkhook
3708 }

```

\GLSp1 behaves like \glsp1 except that all the link text is converted to uppercase.

\GLSp1

```

3709 \newrobustcmd*{\GLSp1}{\@gls@hyp@opt\@GLSp1}

```

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

```

3710 \newcommand*{\@GLSp1}[2][\@GLSp1@{#1}{#2}]{\@GLSp1@{#1}{#2}[\@GLSp1@{#1}{#2}]}%
3711 \new@ifnextchar[\@GLSp1@{#1}{#2}]{\@GLSp1@{#1}{#2}[\@GLSp1@{#1}{#2}]}%
3712 }

```

\@GLSp1 Read in the final optional argument:

```

3713 \def\@GLSp1@#1#2[#3]{%
3714   \glsdoifexists{#2}%
3715   {%
3716     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
3717     \let\glsifplural\@firstoftwo
3718     \let\glsapscase\@thirdofthree
3719     \let\glscustomtext\@empty
3720     \def\glsinsert{#3}%

```

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

```

3721 \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.

```

3722 \@gls@link[#1]{#2}{\@glo@text}%

```

Indicate that this entry has now been used

```

3723 \ifKV@glslink@local
3724   \glslocalunset{#2}%
3725   \else
3726     \glsunset{#2}%
3727   \fi
3728 }%

```

```

3729 \glspostlinkhook
3730 }

```

\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:

```

3731 \newrobustcmd*{\glsdisp}{\@gls@hyp@opt\@glsdisp}

```

Defined the un-starred form.

`\@glsdisp`

```

3732 \newcommand*\@glsdisp}[3] [] {%
3733   \glsdoifexists{#2}{%

3734     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3735     \let\glsifplural\@secondoftwo
3736     \let\glscapscase\@firstofthree
3737     \def\glscustomtext{#3}%
3738     \def\glsinsert{}%

```

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

```

3739   \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.

```

3740   \@gls@link[#1]{#2}{\@glo@text}%

```

Indicate that this entry has now been used

```

3741   \ifKV@glslink@local
3742     \glslocalunset{#2}%
3743   \else
3744     \glsunset{#2}%
3745   \fi
3746 }%

```

```

3747 \glspostlinkhook
3748 }

```

`checkfirsthyper` Instead of just setting `\do@gls@link@checkfirsthyper` to `\relax` in `\@gls@field@link`, set it to `\@gls@link@nocheckfirsthyper` in case some other action needs to take place.

```

3749 \newcommand*\@gls@link@nocheckfirsthyper{}

```

`@gls@field@link`

```

3750 \newcommand*\@gls@field@link}[3] {%
3751   \glsdoifexists{#2}%
3752   {%
3753     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
3754     \@gls@link[#1]{#2}{#3}%
3755   }%

3756   \glspostlinkhook
3757 }

```

`\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`

```
3758 \newrobustcmd*{\glstext}{\@gls@hyp@opt\@glstext}
```

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

```
3759 \newcommand*{\@glstext}[2] [] {%
```

```
3760   \new@ifnextchar[{\@glstext@{#1}{#2}}{\@glstext@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3761 \def\@glstext@#1#2[#3] {%
```

```
3762   \@gls@field@link{#1}{#2}{\glstrytext{#2}#3}%
```

```
3763 }
```

`\GLStext` behaves like `\glstext` except the text is converted to uppercase.

`\GLStext`

```
3764 \newrobustcmd*{\GLStext}{\@gls@hyp@opt\@GLStext}
```

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

```
3765 \newcommand*{\@GLStext}[2] [] {%
```

```
3766   \new@ifnextchar[{\@GLStext@{#1}{#2}}{\@GLStext@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3767 \def\@GLStext@#1#2[#3] {%
```

```
3768   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glstrytext{#2}#3}}%
```

```
3769 }
```

`\Glstext` behaves like `\glstext` except that the first letter of the text is converted to uppercase.

`\Glstext`

```
3770 \newrobustcmd*{\Glstext}{\@gls@hyp@opt\@Glstext}
```

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

```
3771 \newcommand*{\@Glstext}[2] [] {%
```

```
3772   \new@ifnextchar[{\@Glstext@{#1}{#2}}{\@Glstext@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3773 \def\@Glstext@#1#2[#3] {%
```

```
3774   \@gls@field@link{#1}{#2}{\Glstrytext{#2}#3}%
```

```
3775 }
```

`\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`

```
3776 \newrobustcmd*{\glsfirst}{\@gls@hyp@opt\@glsfirst}
```

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

```
3777 \newcommand*{\@glsfirst}[2] [] {%
```

```
3778   \new@ifnextchar[{\@glsfirst@{#1}{#2}}{\@glsfirst@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3779 \def\@glsfirst@#1#2[#3]{%
3780   \@gls@field@link{#1}{#2}{\glsentryfirst{#2}#3}%
3781 }
```

\Glsfirst behaves like \glsfirst except it displays the first letter in uppercase.

\Glsfirst

```
3782 \newrobustcmd*{\Glsfirst}{\@gls@hyp@opt\@Glsfirst}
```

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

```
3783 \newcommand*{\@Glsfirst}[2] [] {%
3784   \new@ifnextchar[{\@Glsfirst@{#1}{#2}}{\@Glsfirst@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3785 \def\@Glsfirst@#1#2[#3]{%
3786   \@gls@field@link{#1}{#2}{\glsentryfirst{#2}#3}%
3787 }
```

\GLSfirst behaves like \Glsfirst except it displays the text in uppercase.

\GLSfirst

```
3788 \newrobustcmd*{\GLSfirst}{\@gls@hyp@opt\@GLSfirst}
```

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

```
3789 \newcommand*{\@GLSfirst}[2] [] {%
3790   \new@ifnextchar[{\@GLSfirst@{#1}{#2}}{\@GLSfirst@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3791 \def\@GLSfirst@#1#2[#3]{%
3792   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryfirst{#2}#3}}%
3793 }
```

\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

```
3794 \newrobustcmd*{\glsplural}{\@gls@hyp@opt\@glsplural}
```

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

```
3795 \newcommand*{\@glsplural}[2] [] {%
3796   \new@ifnextchar[{\@glsplural@{#1}{#2}}{\@glsplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3797 \def\@glsplural@#1#2[#3]{%
3798   \@gls@field@link{#1}{#2}{\glsentryplural{#2}#3}%
3799 }
```

\Glsplural behaves like \glsplural except that the first letter is converted to uppercase.

\Glsplural

```
3800 \newrobustcmd*{\Glsplural}{\@gls@hyp@opt\@Glsplural}
```

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

```
3801 \newcommand*{\@Glsplural}[2] [] {%  
3802   \new@ifnextchar [{\@Glsplural@{#1}{#2}}{\@Glsplural@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3803 \def\@Glsplural@#1#2[#3] {%  
3804   \@gls@field@link{#1}{#2}{\Glsentryplural{#2}#3}%  
3805 }
```

\Glsplural behaves like \glsplural except that the text is converted to uppercase.

\Glsplural

```
3806 \newrobustcmd*{\Glsplural}{\@gls@hyp@opt\@Glsplural}
```

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

```
3807 \newcommand*{\@Glsplural}[2] [] {%  
3808   \new@ifnextchar [{\@Glsplural@{#1}{#2}}{\@Glsplural@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3809 \def\@Glsplural@#1#2[#3] {%  
3810   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryplural{#2}#3}}%  
3811 }
```

\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

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

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

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

Read in the final optional argument:

```
3815 \def\@glsfirstplural@#1#2[#3] {%  
3816   \@gls@field@link{#1}{#2}{\glsentryfirstplural{#2}#3}%  
3817 }
```

\Glsfirstplural behaves like \glsfirstplural except that the first letter is converted to uppercase.

\Glsfirstplural

```
3818 \newrobustcmd*{\Glsfirstplural}{\@gls@hyp@opt\@Glsfirstplural}
```

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

```
3819 \newcommand*{\@Glsfirstplural}[2] [] {%  
3820   \new@ifnextchar [{\@Glsfirstplural@{#1}{#2}}{\@Glsfirstplural@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3821 \def\@Glsfirstplural@#1#2[#3] {%  
3822   \@gls@field@link{#1}{#2}{\Glsentryfirstplural{#2}#3}%  
3823 }
```

`\GLSfirstplural` behaves like `\glsfirstplural` except that the link text is converted to uppercase.

`\GLSfirstplural`

```
3824 \newrobustcmd*{\GLSfirstplural}{\@gls@hyp@opt\@GLSfirstplural}
```

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

```
3825 \newcommand*{\@GLSfirstplural}[2] [] {%
```

```
3826   \new@ifnextchar[{\@GLSfirstplural@{#1}{#2}}{\@GLSfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3827 \def\@GLSfirstplural@#1#2[#3] {%
```

```
3828   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryfirstplural{#2}#3}}%
```

```
3829 }
```

`\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`

```
3830 \newrobustcmd*{\glsname}{\@gls@hyp@opt\@glsname}
```

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

```
3831 \newcommand*{\@glsname}[2] [] {%
```

```
3832   \new@ifnextchar[{\@glsname@{#1}{#2}}{\@glsname@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3833 \def\@glsname@#1#2[#3] {%
```

```
3834   \@gls@field@link{#1}{#2}{\glsentryname{#2}#3}}%
```

```
3835 }
```

`\Glsname` behaves like `\glsname` except that the first letter is converted to uppercase.

`\Glsname`

```
3836 \newrobustcmd*{\Glsname}{\@gls@hyp@opt\@Glsname}
```

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

```
3837 \newcommand*{\@Glsname}[2] [] {%
```

```
3838   \new@ifnextchar[{\@Glsname@{#1}{#2}}{\@Glsname@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3839 \def\@Glsname@#1#2[#3] {%
```

```
3840   \@gls@field@link{#1}{#2}{\Glsentryname{#2}#3}}%
```

```
3841 }
```

`\GLSname` behaves like `\glsname` except that the link text is converted to uppercase.

`\GLSname`

```
3842 \newrobustcmd*{\GLSname}{\@gls@hyp@opt\@GLSname}
```

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

```
3843 \newcommand*{\@GLSname}[2] [] {%
```

```
3844   \new@ifnextchar[{\@GLSname@{#1}{#2}}{\@GLSname@{#1}{#2} []}]}
```


Read in the final optional argument:

```
3845 \def\@GLSname@#1#2[#3]{%
3846   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryname{#2}#3}}%
3847 }
```

\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

```
3848 \newrobustcmd*{\glsdesc}{\@gls@hyp@opt\@glsdesc}
```

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

```
3849 \newcommand*{\@glsdesc}[2][]{%
3850   \new@ifnextchar[{\@glsdesc@{#1}{#2}}{\@glsdesc@{#1}{#2}[]}]}
```

Read in the final optional argument:

```
3851 \def\@glsdesc@#1#2[#3]{%
3852   \@gls@field@link{#1}{#2}{\glsentrydesc{#2}#3}%
3853 }
```

\Glsdesc behaves like \glsdesc except that the first letter is converted to uppercase.

\Glsdesc

```
3854 \newrobustcmd*{\Glsdesc}{\@gls@hyp@opt\@Glsdesc}
```

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

```
3855 \newcommand*{\@Glsdesc}[2][]{%
3856   \new@ifnextchar[{\@Glsdesc@{#1}{#2}}{\@Glsdesc@{#1}{#2}[]}]}
```

Read in the final optional argument:

```
3857 \def\@Glsdesc@#1#2[#3]{%
3858   \@gls@field@link{#1}{#2}{\Glsentrydesc{#2}#3}%
3859 }
```

\GLSdesc behaves like \glsdesc except that the link text is converted to uppercase.

\GLSdesc

```
3860 \newrobustcmd*{\GLSdesc}{\@gls@hyp@opt\@GLSdesc}
```

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

```
3861 \newcommand*{\@GLSdesc}[2][]{%
3862   \new@ifnextchar[{\@GLSdesc@{#1}{#2}}{\@GLSdesc@{#1}{#2}[]}]}
```

Read in the final optional argument:

```
3863 \def\@GLSdesc@#1#2[#3]{%
3864   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrydesc{#2}#3}}%
3865 }
```

\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

```
3866 \newrobustcmd*{\glsdescplural}{\@gls@hyp@opt\@glsdescplural}
```

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

```
3867 \newcommand*{\@glsdescplural}[2] [] {%
3868   \new@ifnextchar [{\@glsdescplural@{#1}{#2}}{\@glsdescplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3869 \def\@glsdescplural@#1#2[#3] {%
3870   \@gls@field@link{#1}{#2}{\glsentrydescplural{#2}#3}%
3871 }
```

\Glsdescplural behaves like \glsdescplural except that the first letter is converted to uppercase.

\Glsdescplural

```
3872 \newrobustcmd*{\Glsdescplural}{\@gls@hyp@opt\@Glsdescplural}
```

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

```
3873 \newcommand*{\@GLSdescplural}[2] [] {%
3874   \new@ifnextchar [{\@GLSdescplural@{#1}{#2}}{\@GLSdescplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3875 \def\@GLSdescplural@#1#2[#3] {%
3876   \@gls@field@link{#1}{#2}{\Glsentrydescplural{#2}#3}%
3877 }
```

\GLSdescplural behaves like \glsdescplural except that the link text is converted to uppercase.

\GLSdescplural

```
3878 \newrobustcmd*{\GLSdescplural}{\@gls@hyp@opt\@GLSdescplural}
```

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

```
3879 \newcommand*{\@GLSdescplural}[2] [] {%
3880   \new@ifnextchar [{\@GLSdescplural@{#1}{#2}}{\@GLSdescplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3881 \def\@GLSdescplural@#1#2[#3] {%
3882   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrydescplural{#2}#3}}%
3883 }
```

\glssymbol behaves like \gls except it always uses the value given by the symbol key and it doesn't mark the entry as used.

\glssymbol

```
3884 \newrobustcmd*{\glssymbol}{\@gls@hyp@opt\@glssymbol}
```

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

```
3885 \newcommand*{\@glssymbol}[2] [] {%
3886   \new@ifnextchar [{\@glssymbol@{#1}{#2}}{\@glssymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3887 \def\@glssymbol@#1#2[#3] {%
3888   \@gls@field@link{#1}{#2}{\glsentrysymbol{#2}#3}%
3889 }
```

`\Glssymbol` behaves like `\glssymbol` except that the first letter is converted to uppercase.

`\Glssymbol`

```
3890 \newrobustcmd*{\Glssymbol}{\@gls@hyp@opt\@Glssymbol}
```

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

```
3891 \newcommand*{\@Glssymbol}[2] [] {%
```

```
3892   \new@ifnextchar[{\@Glssymbol@{#1}{#2}}{\@Glssymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3893 \def\@Glssymbol@#1#2[#3] {%
```

```
3894   \@gls@field@link{#1}{#2}{\glstentrysymbol{#2}#3}%
```

```
3895 }
```

`\GLSsymbol` behaves like `\glssymbol` except that the link text is converted to uppercase.

`\GLSsymbol`

```
3896 \newrobustcmd*{\GLSsymbol}{\@gls@hyp@opt\@GLSsymbol}
```

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

```
3897 \newcommand*{\@GLSsymbol}[2] [] {%
```

```
3898   \new@ifnextchar[{\@GLSsymbol@{#1}{#2}}{\@GLSsymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3899 \def\@GLSsymbol@#1#2[#3] {%
```

```
3900   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glstentrysymbol{#2}#3}%
```

```
3901 }
```

`\glssymbolplural` behaves like `\gls` except it always uses the value given by the symbol-plural key and it doesn't mark the entry as used.

`glssymbolplural`

```
3902 \newrobustcmd*{\glssymbolplural}{\@gls@hyp@opt\@glssymbolplural}
```

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

```
3903 \newcommand*{\@glssymbolplural}[2] [] {%
```

```
3904   \new@ifnextchar[{\@glssymbolplural@{#1}{#2}}{\@glssymbolplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3905 \def\@glssymbolplural@#1#2[#3] {%
```

```
3906   \@gls@field@link{#1}{#2}{\glstentrysymbolplural{#2}#3}%
```

```
3907 }
```

`\Glssymbolplural` behaves like `\glssymbolplural` except that the first letter is converted to uppercase.

`Glssymbolplural`

```
3908 \newrobustcmd*{\Glssymbolplural}{\@gls@hyp@opt\@Glssymbolplural}
```

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

```
3909 \newcommand*{\@Glssymbolplural}[2] [] {%
```

```
3910   \new@ifnextchar[{\@Glssymbolplural@{#1}{#2}}{\@Glssymbolplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3911 \def\@Glsymbolplural@#1#2[#3]{%
3912   \@gls@field@link{#1}{#2}{\Glsentrysymbolplural{#2}#3}%
3913 }
```

\Glsymbolplural behaves like \glsymbolplural except that the link text is converted to uppercase.

\Glsymbolplural

```
3914 \newrobustcmd*{\Glsymbolplural}{\@gls@hyp@opt\@Glsymbolplural}
```

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

```
3915 \newcommand*{\@Glsymbolplural}[2] [] {%
3916   \new@ifnextchar[{\@Glsymbolplural@{#1}{#2}}{\@Glsymbolplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3917 \def\@Glsymbolplural@#1#2[#3]{%
3918   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentrysymbolplural{#2}#3}}%
3919 }
```

\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

```
3920 \newrobustcmd*{\glsuseri}{\@gls@hyp@opt\@glsuseri}
```

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

```
3921 \newcommand*{\@glsuseri}[2] [] {%
3922   \new@ifnextchar[{\@glsuseri@{#1}{#2}}{\@glsuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3923 \def\@glsuseri@#1#2[#3]{%
3924   \@gls@field@link{#1}{#2}{\Glsentryuseri{#2}#3}%
3925 }
```

\Glsuseri behaves like \glsuseri except that the first letter is converted to uppercase.

\Glsuseri

```
3926 \newrobustcmd*{\Glsuseri}{\@gls@hyp@opt\@Glsuseri}
```

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

```
3927 \newcommand*{\@Glsuseri}[2] [] {%
3928   \new@ifnextchar[{\@Glsuseri@{#1}{#2}}{\@Glsuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3929 \def\@Glsuseri@#1#2[#3]{%
3930   \@gls@field@link{#1}{#2}{\Glsentryuseri{#2}#3}%
3931 }
```

\GLSuseri behaves like \glsuseri except that the link text is converted to uppercase.

\GLSuseri

```
3932 \newrobustcmd*{\GLSuseri}{\@gls@hyp@opt\@GLSuseri}
```

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

```
3933 \newcommand*{\@GLSuseri}[2] [] {%  
3934   \new@ifnextchar[{\@GLSuseri@{#1}{#2}}{\@GLSuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3935 \def\@GLSuseri@#1#2[#3] {%  
3936   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseri{#2}#3}}%  
3937 }
```

\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

```
3938 \newrobustcmd*{\glsuserii}{\@gls@hyp@opt\@glsuserii}
```

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

```
3939 \newcommand*{\@glsuserii}[2] [] {%  
3940   \new@ifnextchar[{\@glsuserii@{#1}{#2}}{\@glsuserii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3941 \def\@glsuserii@#1#2[#3] {%  
3942   \@gls@field@link{#1}{#2}{\glsentryuserii{#2}#3}}%  
3943 }
```

\Glsuserii behaves like \glsuserii except that the first letter is converted to uppercase.

\Glsuserii

```
3944 \newrobustcmd*{\Glsuserii}{\@gls@hyp@opt\@Glsuserii}
```

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

```
3945 \newcommand*{\@Glsuserii}[2] [] {%  
3946   \new@ifnextchar[{\@Glsuserii@{#1}{#2}}{\@Glsuserii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3947 \def\@Glsuserii@#1#2[#3] {%  
3948   \@gls@field@link{#1}{#2}{\Glsentryuserii{#2}#3}}%  
3949 }
```

\GLSuserii behaves like \glsuserii except that the link text is converted to uppercase.

\GLSuserii

```
3950 \newrobustcmd*{\GLSuserii}{\@gls@hyp@opt\@GLSuserii}
```

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

```
3951 \newcommand*{\@GLSuserii}[2] [] {%  
3952   \new@ifnextchar[{\@GLSuserii@{#1}{#2}}{\@GLSuserii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3953 \def\@GLSuserii@#1#2[#3] {%  
3954   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuserii{#2}#3}}%  
3955 }
```

\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`

```
3956 \newrobustcmd*{\glsuseriii}{\@gls@hyp@opt\@glsuseriii}
```

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

```
3957 \newcommand*{\@glsuseriii}[2] [] {%
```

```
3958   \new@ifnextchar[{\@glsuseriii@{#1}{#2}}{\@glsuseriii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3959 \def\@glsuseriii@#1#2[#3] {%
```

```
3960   \@gls@field@link{#1}{#2}{\glsentryuseriii{#2}#3}%
```

```
3961 }
```

`\Glsuseriii` behaves like `\glsuseriii` except that the first letter is converted to upper-case.

`\Glsuseriii`

```
3962 \newrobustcmd*{\Glsuseriii}{\@gls@hyp@opt\@Glsuseriii}
```

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

```
3963 \newcommand*{\@Glsuseriii}[2] [] {%
```

```
3964   \new@ifnextchar[{\@Glsuseriii@{#1}{#2}}{\@Glsuseriii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3965 \def\@Glsuseriii@#1#2[#3] {%
```

```
3966   \@gls@field@link{#1}{#2}{\Glsentryuseriii{#2}#3}%
```

```
3967 }
```

`\GLSuseriii` behaves like `\glsuseriii` except that the link text is converted to uppercase.

`\GLSuseriii`

```
3968 \newrobustcmd*{\GLSuseriii}{\@gls@hyp@opt\@GLSuseriii}
```

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

```
3969 \newcommand*{\@GLSuseriii}[2] [] {%
```

```
3970   \new@ifnextchar[{\@GLSuseriii@{#1}{#2}}{\@GLSuseriii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3971 \def\@GLSuseriii@#1#2[#3] {%
```

```
3972   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseriii{#2}#3}}%
```

```
3973 }
```

`\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`

```
3974 \newrobustcmd*{\glsuseriv}{\@gls@hyp@opt\@glsuseriv}
```

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

```
3975 \newcommand*{\@glsuseriv}[2] [] {%
```

```
3976   \new@ifnextchar[{\@glsuseriv@{#1}{#2}}{\@glsuseriv@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3977 \def\@glsuseriv@#1#2[#3]{%
3978   \@gls@field@link{#1}{#2}{\glsentryuseriv{#2}#3}%
3979 }
```

\Glsuseriv behaves like \glsuseriv except that the first letter is converted to uppercase.

\Glsuseriv

```
3980 \newrobustcmd*{\Glsuseriv}{\@gls@hyp@opt\@Glsuseriv}
```

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

```
3981 \newcommand*{\@Glsuseriv}[2][ ]{%
3982   \new@ifnextchar[{\@Glsuseriv@{#1}{#2}}{\@Glsuseriv@{#1}{#2}[ ]}]}
```

Read in the final optional argument:

```
3983 \def\@Glsuseriv@#1#2[#3]{%
3984   \@gls@field@link{#1}{#2}{\glsentryuseriv{#2}#3}%
3985 }
```

\GLSuseriv behaves like \glsuseriv except that the link text is converted to uppercase.

\GLSuseriv

```
3986 \newrobustcmd*{\GLSuseriv}{\@gls@hyp@opt\@GLSuseriv}
```

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

```
3987 \newcommand*{\@GLSuseriv}[2][ ]{%
3988   \new@ifnextchar[{\@GLSuseriv@{#1}{#2}}{\@GLSuseriv@{#1}{#2}[ ]}]}
```

Read in the final optional argument:

```
3989 \def\@GLSuseriv@#1#2[#3]{%
3990   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseriv{#2}#3}}%
3991 }
```

\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

```
3992 \newrobustcmd*{\glsuserv}{\@gls@hyp@opt\@glsuserv}
```

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

```
3993 \newcommand*{\@glsuserv}[2][ ]{%
3994   \new@ifnextchar[{\@glsuserv@{#1}{#2}}{\@glsuserv@{#1}{#2}[ ]}]}
```

Read in the final optional argument:

```
3995 \def\@glsuserv@#1#2[#3]{%
3996   \@gls@field@link{#1}{#2}{\glsentryuserv{#2}#3}%
3997 }
```

\Glsuserv behaves like \glsuserv except that the first letter is converted to uppercase.

\Glsuserv

```
3998 \newrobustcmd*{\Glsuserv}{\@gls@hyp@opt\@Glsuserv}
```

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

```
3999 \newcommand*{\@Glsuserv}[2] [] {%  
4000 \new@ifnextchar [{\@Glsuserv@{#1}{#2}}{\@Glsuserv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4001 \def\@Glsuserv@#1#2[#3] {%  
4002   \@gls@field@link{#1}{#2}{\Glsentryuserv{#2}#3}%  
4003 }
```

\Glsuserv behaves like \glsuserv except that the link text is converted to uppercase.

\Glsuserv

```
4004 \newrobustcmd*{\Glsuserv}{\@gls@hyp@opt\@Glsuserv}
```

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

```
4005 \newcommand*{\@Glsuserv}[2] [] {%  
4006 \new@ifnextchar [{\@Glsuserv@{#1}{#2}}{\@Glsuserv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4007 \def\@Glsuserv@#1#2[#3] {%  
4008   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuserv{#2}#3}}%  
4009 }
```

\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

```
4010 \newrobustcmd*{\glsuservi}{\@gls@hyp@opt\@glsuservi}
```

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

```
4011 \newcommand*{\@glsuservi}[2] [] {%  
4012   \new@ifnextchar [{\@glsuservi@{#1}{#2}}{\@glsuservi@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4013 \def\@glsuservi@#1#2[#3] {%  
4014   \@gls@field@link{#1}{#2}{\glsentryuservi{#2}#3}%  
4015 }
```

\Glsuservi behaves like \glsuservi except that the first letter is converted to uppercase.

\Glsuservi

```
4016 \newrobustcmd*{\Glsuservi}{\@gls@hyp@opt\@Glsuservi}
```

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

```
4017 \newcommand*{\@Glsuservi}[2] [] {%  
4018   \new@ifnextchar [{\@Glsuservi@{#1}{#2}}{\@Glsuservi@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4019 \def\@Glsuservi@#1#2[#3] {%  
4020   \@gls@field@link{#1}{#2}{\Glsentryuservi{#2}#3}%  
4021 }
```

\Glsuservi behaves like \glsuservi except that the link text is converted to uppercase.

\GLSuservi

```
4022 \newrobustcmd*{\GLSuservi}{\@gls@hyp@opt\@GLSuservi}
```

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

```
4023 \newcommand*{\@GLSuservi}[2] [] {%
```

```
4024   \new@ifnextchar[{\@GLSuservi@{#1}{#2}}{\@GLSuservi@{#1}{#2} []}]
```

Read in the final optional argument:

```
4025 \def\@GLSuservi@#1#2[#3] {%
```

```
4026   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glentryuservi{#2}{#3}}}%
```

```
4027 }
```

Now deal with acronym related keys. First the short form:

\acrshort

```
4028 \newrobustcmd*{\acrshort}{\@gls@hyp@opt\@ns@acrshort}
```

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

```
4029 \newcommand*{\@ns@acrshort}[2] [] {%
```

```
4030   \new@ifnextchar[{\@acrshort{#1}{#2}}{\@acrshort{#1}{#2} []}]
```

```
4031 }
```

Read in the final optional argument:

```
4032 \def\@acrshort#1#2[#3] {%
```

```
4033   \glsdoifexists{#2}%
```

```
4034   {%
```

```
4035     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
```

```
4036     \let\glsifplural\@secondoftwo
```

```
4037     \let\gls@caps@case\@firstofthree
```

```
4038     \let\glsinsert\@empty
```

```
4039     \def\gls@customtext{%
```

```
4040       \acronymfont{\glentryshort{#2}}#3%
```

```
4041     }%
```

Call \@gls@link Note that \@gls@link sets \glstype.

```
4042   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
```

```
4043   }%
```

```
4044   \glspostlinkhook
```

```
4045 }
```

\Acrshort

```
4046 \newrobustcmd*{\Acrshort}{\@gls@hyp@opt\@ns@Acrshort}
```

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

```
4047 \newcommand*{\@ns@Acrshort}[2] [] {%
```

```
4048   \new@ifnextchar[{\@Acrshort{#1}{#2}}{\@Acrshort{#1}{#2} []}]
```

```
4049 }
```

Read in the final optional argument:

```
4050 \def\@Acrshort#1#2[#3]{%
4051   \glsdoifexists{#2}%
4052   {%

4053     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper

4054     \def\glslabel{#2}%
4055     \let\glsifplural\@secondoftwo
4056     \let\glscapscase\@thirdofthree
4057     \let\glsinsert\@empty
4058     \def\glscustomtext{%
4059       \acronymfont{\Glsentryshort{#2}}#3%
4060     }%

    Call \@gl@link Note that \@gl@link sets \glstype.
4061     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4062     }%

4063   \glspostlinkhook
4064 }
```

\ACRshort

```
4065 \newrobustcmd*{\ACRshort}{\@gl@hyp@opt\@ns@ACRshort}
```

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

```
4066 \newcommand*{\@ns@ACRshort}[2][ ]{%
4067   \new@ifnextchar[{\@ACRshort{#1}{#2}}{\@ACRshort{#1}{#2}[]}%
4068 }
```

Read in the final optional argument:

```
4069 \def\@ACRshort#1#2[#3]{%
4070   \glsdoifexists{#2}%
4071   {%

4072     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper

4073     \def\glslabel{#2}%
4074     \let\glsifplural\@secondoftwo
4075     \let\glscapscase\@thirdofthree
4076     \let\glsinsert\@empty
4077     \def\glscustomtext{%
4078       \mfirstucMakeUppercase{\acronymfont{\Glsentryshort{#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 }
```

Short plural:

\acrshortpl

```
4084 \newrobustcmd*{\acrshortpl}{\@gls@hyp@opt\ns@acrshortpl}
```

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

```
4085 \newcommand*{\ns@acrshortpl}[2][\%  
4086 \new@ifnextchar[\@acrshortpl{#1}{#2}]{\@acrshortpl{#1}{#2}[]}%  
4087 }
```

Read in the final optional argument:

```
4088 \def\@acrshortpl#1#2[#3]{%  
4089 \glsdoifexists{#2}%  
4090 {%  
  
4091 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper  
  
4092 \def\glslabel{#2}%  
4093 \let\glsifplural\@firstoftwo  
4094 \let\glsupcase\@firstofthree  
4095 \let\glsinsert\@empty  
4096 \def\glscustomtext{%  
4097 \acronymfont{\glsentryshortpl{#2}}#3%  
4098 }%
```

Call \@gls@link Note that \@gls@link sets \glsstyle.

```
4099 \@gls@link[#1]{#2}{\csname gls@\glsstyle @entryfmt\endcsname}%  
4100 }%  
  
4101 \glspostlinkhook  
4102 }
```

\Acrshortpl

```
4103 \newrobustcmd*{\Acrshortpl}{\@gls@hyp@opt\ns@Acrshortpl}
```

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

```
4104 \newcommand*{\ns@Acrshortpl}[2][\%  
4105 \new@ifnextchar[\@Acrshortpl{#1}{#2}]{\@Acrshortpl{#1}{#2}[]}%  
4106 }
```

Read in the final optional argument:

```
4107 \def\@Acrshortpl#1#2[#3]{%  
4108 \glsdoifexists{#2}%  
4109 {%  
  
4110 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper  
  
4111 \def\glslabel{#2}%  
4112 \let\glsifplural\@firstoftwo  
4113 \let\glsupcase\@secondofthree  
4114 \let\glsinsert\@empty
```

```

4115 \def\glscustomtext{%
4116 \acronymfont{\Glsentryshortpl{#2}}#3%
4117 }%

Call \@gls@link Note that \@gls@link sets \glstype.
4118 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4119 }%

4120 \glspostlinkhook
4121 }

```

\ACRshortpl

```

4122 \newrobustcmd*{\ACRshortpl}{\@gls@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 \glstoifexists{#2}%
4128 {%

4129 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

4130 \def\glslabel{#2}%
4131 \let\glsifplural\@firstoftwo
4132 \let\glscapscase\@thirdofthree
4133 \let\glsinsert\@empty
4134 \def\glscustomtext{%
4135 \mfirstucMakeUppercase{\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 }

```

\acrlong

```

4141 \newrobustcmd*{\acrlong}{\@gls@hyp@opt\@ns@acrlong}

Define the un-starred form. Need to determine if there is a final optional argument
4142 \newcommand*{\ns@acrlong}[2][{}]{%
4143 \new@ifnextchar[{\@acrlong{#1}{#2}}{\@acrlong{#1}{#2}[]}%
4144 }

```

Read in the final optional argument:

```
4145 \def\@acrlong#1#2[#3]{%
4146   \glsdoifexists{#2}%
4147   {%
4148     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4149     \def\glslabel{#2}%
4150     \let\glsifplural\@secondoftwo
4151     \let\glscapscase\@firstofthree
4152     \let\glsinsert\@empty
```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```
4153   \def\glscustomtext{%
4154     \glstrylong{#2}#3%
4155   }%
```

Call \@gl@link Note that \@gl@link sets \glstype.

```
4156   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4157   }%
4158   \glspostlinkhook
4159 }
```

\Acrlong

```
4160 \newrobustcmd*{\Acrlong}{\@gl@hyp@opt\@ns@Acrlong}
```

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

```
4161 \newcommand*{\ns@Acrlong}[2][ ]{%
4162   \new@ifnextchar{\@Acrlong{#1}{#2}}{\@Acrlong{#1}{#2}[ ]}%
4163 }
```

Read in the final optional argument:

```
4164 \def\@Acrlong#1#2[#3]{%
4165   \glsdoifexists{#2}%
4166   {%
4167     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4168     \def\glslabel{#2}%
4169     \let\glsifplural\@secondoftwo
4170     \let\glscapscase\@secondofthree
4171     \let\glsinsert\@empty
```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```
4172   \def\glscustomtext{%
4173     \Glsentrylong{#2}#3%
4174   }%
```

Call \@gls@link. Note that \@gls@link sets \gls@type.

```
4175 \gls@link[#1]{#2}{\csname gls@\gls@type @entryfmt\endcsname}%
4176 }%

4177 \gls@postlinkhook
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\gls@scapscase\@thirdofthree
4190 \let\glsinsert\@empty
```

Bug fix v4.02 removed \acronymfont from \gls@customtext (\acronymfont only designed for short form).

```
4191 \def\gls@customtext{%
4192 \mfirstucMakeUppercase{\gls@entrylong{#2}{#3}}%
4193 }%
```

Call \@gls@link. Note that \@gls@link sets \gls@type.

```
4194 \gls@link[#1]{#2}{\csname gls@\gls@type @entryfmt\endcsname}%
4195 }%

4196 \gls@postlinkhook
4197 }
```

Short plural:

\acrlongpl

```
4198 \newrobustcmd*{\acrlongpl}{\@gls@hyp@opt\@ns@acrlongpl}
```

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

```
4199 \newcommand*{\ns@acrlongpl}[2][\%]
4200 \new@ifnextchar[\@acrlongpl{#1}{#2}}{\@acrlongpl{#1}{#2}[]}%
4201 }
```

Read in the final optional argument:

```
4202 \def\@acrlongpl#1#2[#3]{%
4203   \glsdoifexists{#2}%
4204   {%
4205     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4206     \def\glslabel{#2}%
4207     \let\glsifplural\@firstoftwo
4208     \let\glscapscase\@firstofthree
4209     \let\glsinsert\@empty
```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```
4210   \def\glscustomtext{%
4211     \glstrylongpl{#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 }
```

\Acrlongpl

```
4217 \newrobustcmd*{\Acrlongpl}{\@gl@hyp@opt\@ns@Acrlongpl}
```

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

```
4218 \newcommand*{\ns@Acrlongpl}[2][\@ns@Acrlongpl]{%
4219   \new@ifnextchar[\@Acrlongpl{#1}{#2}}{\@Acrlongpl{#1}{#2}}}%
4220 }
```

Read in the final optional argument:

```
4221 \def\@Acrlongpl#1#2[#3]{%
4222   \glsdoifexists{#2}%
4223   {%
4224     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4225     \def\glslabel{#2}%
4226     \let\glsifplural\@firstoftwo
4227     \let\glscapscase\@secondofthree
4228     \let\glsinsert\@empty
```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```
4229   \def\glscustomtext{%
4230     \Glsentrylongpl{#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 }
```

`\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 \glsdoifexists{#2}%  
4242 {%  
  
4243 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper  
  
4244 \def\glslabel{#2}%  
4245 \let\glsifplural\@firstoftwo  
4246 \let\gls caps case\@thirdofthree  
4247 \let\glsinsert\@empty
```

Bug fix v4.02 removed `\acronymfont` from `\glscustomtext` (`\acronymfont` only designed for short form).

```
4248 \def\glscustomtext{%  
4249 \mfirstucMakeUppercase{\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 }
```

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>}`

```
4255 \newcommand*{\@gls@entry@field}[2]{%
```



```

4256 \csname glo@\glsdetoklabel{#1}@#2\endcsname
4257 }

```

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

```

4258 \newcommand*{\glsletentryfield}[3]{%
4259   \letcs{#1}{glo@\glsdetoklabel{#2}@#3}%
4260 }

```

`\Gls@entry@field` Generic first letter uppercase version.

`\@Gls@entry@field{<label>}{<field>}`

```

4261 \newcommand*{\@Gls@entry@field}[2]{%
4262   \glsdoifexistsordo{#1}%
4263   {%
4264     \letcs{@glo@text}{glo@\glsdetoklabel{#1}@#2}%
4265     \ifdef{@glo@text
4266       {%
4267         \xmakefirstuc{@glo@text}%
4268       }%
4269       {%
4270         ??\PackageError{glossaries}{The field ‘#2’ doesn’t exist for glossary
4271           entry ‘\glsdetoklabel{#1}’}{Check you have correctly spelt the entry
4272             label and the field name}%
4273       }%
4274     }%
4275     {%
4276       ??%
4277     }%
4278 }

```

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`

```

4279 \newcommand*{\glsentryname}[1]{\@gls@entry@field{#1}{name}}

```

`\Glsentryname`

```

4280 \newrobustcmd*{\Glsentryname}[1]{%
4281   \@Gls@entryname{#1}%
4282 }

```

`\@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:

```

4283 \newcommand*{\@Gls@entryname}[1]{%
4284   \@Gls@entry@field{#1}{name}%
4285 }

```

ls@acrentryname Now the behaviour when \setacronymstyle is used:

```

4286 \newcommand*{\@Gls@acrentryname}[1]{%
4287   \ifglshaslong{#1}%
4288   {%
4289     \letcs\@glo@text{glo@glsetoklabel{#1}@name}%
4290     \expandafter\@gls@getbody\@glo@text{}\@nil
4291     \expandafter\ifx\@gls@body\glsetrylong\relax
4292       \expandafter\Glsentrylong\@gls@rest
4293     \else
4294       \expandafter\ifx\@gls@body\glsetryshort\relax
4295         \expandafter\Glsentryshort\@gls@rest
4296       \else
4297         \expandafter\ifx\@gls@body\acronymfont\relax

```

Temporarily make \glsetryshort behave like \Glsentryshort. (This is on the assumption that the argument of \acronymfont is \glsetryshort{<label>}, as that's the behaviour of the predefined acronym styles.) This is scoped to localise the effect of the assignment.

```

4298     {%
4299       \let\glsetryshort\Glsentryshort
4300       \@glo@text
4301     }%
4302   \else
4303     \xmakefirstuc{\@glo@text}%
4304   \fi
4305 \fi
4306 \fi
4307 }%
4308 {%

```

Not an acronym

```

4309   \@Gls@entry@field{#1}{name}%
4310 }%
4311 }

```

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.

\glsetrydesc

```

4312 \newcommand*{\glsetrydesc}[1]{\@gls@entry@field{#1}{desc}}

```

\Glsentrydesc

```

4313 \newrobustcmd*{\Glsentrydesc}[1]{%
4314   \@Gls@entry@field{#1}{desc}%
4315 }

```

Plural form:

entrydescplural

```
4316 \newcommand*{\glentrydescplural}[1]{%
4317   \@gls@entry@field{#1}{descplural}%
4318 }
```

entrydescplural

```
4319 \newrobustcmd*{\Glsentrydescplural}[1]{%
4320   \@Gls@entry@field{#1}{descplural}%
4321 }
```

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

```
4322 \newcommand*{\glentrytext}[1]{\@gls@entry@field{#1}{text}}
```

\Glsentrytext

```
4323 \newrobustcmd*{\Glsentrytext}[1]{%
4324   \@Gls@entry@field{#1}{text}%
4325 }
```

Get the plural form:

\glentryplural

```
4326 \newcommand*{\glentryplural}[1]{%
4327   \@gls@entry@field{#1}{plural}%
4328 }
```

\Glsentryplural

```
4329 \newrobustcmd*{\Glsentryplural}[1]{%
4330   \@Gls@entry@field{#1}{plural}%
4331 }
```

Get the symbol associated with this entry. The argument is the label associated with the entry.

\glentrysymbol

```
4332 \newcommand*{\glentrysymbol}[1]{%
4333   \@gls@entry@field{#1}{symbol}%
4334 }
```

\Glsentrysymbol

```
4335 \newrobustcmd*{\Glsentrysymbol}[1]{%
4336   \@Gls@entry@field{#1}{symbol}%
4337 }
```

Plural form:

trysymbolplural

```
4338 \newcommand*{\glsentrysymbolplural}[1]{%
4339   \@gls@entry@field{#1}{symbolplural}%
4340 }
```

trysymbolplural

```
4341 \newrobustcmd*{\Glsentrysymbolplural}[1]{%
4342   \@Gls@entry@field{#1}{symbolplural}%
4343 }
```

Get the entry text to be used when the entry is first used in the document (as specified by the first key when the entry was defined).

\glsentryfirst

```
4344 \newcommand*{\glsentryfirst}[1]{%
4345   \@gls@entry@field{#1}{first}%
4346 }
```

\Glsentryfirst

```
4347 \newrobustcmd*{\Glsentryfirst}[1]{%
4348   \@Gls@entry@field{#1}{first}%
4349 }
```

Get the plural form (as specified by the firstplural key when the entry was defined).

ntryfirstplural

```
4350 \newcommand*{\glsentryfirstplural}[1]{%
4351   \@gls@entry@field{#1}{firstpl}%
4352 }
```

ntryfirstplural

```
4353 \newrobustcmd*{\Glsentryfirstplural}[1]{%
4354   \@Gls@entry@field{#1}{firstpl}%
4355 }
```

sentrytitlecase

```
4356 \newrobustcmd*{@glsentrytitlecase}[2]{%
4357   \glsfieldfetch{#1}{#2}{\@gls@value}%
4358   \xcapitalisewords{\@gls@value}%
4359 }
4360 \ifdef\texorpdfstring
4361 {
4362   \newcommand*{\glsentrytitlecase}[2]{%
4363     \texorpdfstring
4364       {\@glsentrytitlecase{#1}{#2}}%
4365       {\@gls@entry@field{#1}{#2}}%
4366   }
4367 }
4368 {
```

```

4369 \newcommand*{\glsentrytitlecase}[2]{\@glsentrytitlecase{#1}{#2}}
4370 }

```

Display the glossary type with which this entry is associated (as specified by the type key used when the entry was defined)

`\glsentrytype`

```

4371 \newcommand*{\glsentrytype}[1]{\@gls@entry@field{#1}{type}}

```

Display the sort text used for this entry. Note that the sort key is sanitized, so unexpected results may occur if the sort key contained commands.

`\glsentrysort`

```

4372 \newcommand*{\glsentrysort}[1]{%
4373   \@gls@entry@field{#1}{sort}%
4374 }

```

`\glsentryuseri` Get the first user key (as specified by the user1 when the entry was defined). The argument is the label associated with the entry.

```

4375 \newcommand*{\glsentryuseri}[1]{%
4376   \@gls@entry@field{#1}{useri}%
4377 }

```

`\Glsentryuseri`

```

4378 \newrobustcmd*{\Glsentryuseri}[1]{%
4379   \@Gls@entry@field{#1}{useri}%
4380 }

```

`\glsentryuserii` Get the second user key (as specified by the user2 when the entry was defined). The argument is the label associated with the entry.

```

4381 \newcommand*{\glsentryuserii}[1]{%
4382   \@gls@entry@field{#1}{userii}%
4383 }

```

`\Glsentryuserii`

```

4384 \newrobustcmd*{\Glsentryuserii}[1]{%
4385   \@Gls@entry@field{#1}{userii}%
4386 }

```

`\glsentryuseriii` Get the third user key (as specified by the user3 when the entry was defined). The argument is the label associated with the entry.

```

4387 \newcommand*{\glsentryuseriii}[1]{%
4388   \@gls@entry@field{#1}{useriii}%
4389 }

```

`\Glsentryuseriii`

```

4390 \newrobustcmd*{\Glsentryuseriii}[1]{%
4391   \@Gls@entry@field{#1}{useriii}%
4392 }

```

`\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.

```
4393 \newcommand*{\glentryuseriv}[1]{%
4394   \@gls@entry@field{#1}{useriv}%
4395 }
```

`\Glsentryuseriv`

```
4396 \newrobustcmd*{\Glsentryuseriv}[1]{%
4397   \@Gls@entry@field{#1}{useriv}%
4398 }
```

`\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.

```
4399 \newcommand*{\glentryuserv}[1]{%
4400   \@gls@entry@field{#1}{userv}%
4401 }
```

`\Glsentryuserv`

```
4402 \newrobustcmd*{\Glsentryuserv}[1]{%
4403   \@Gls@entry@field{#1}{userv}%
4404 }
```

`\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.

```
4405 \newcommand*{\glentryuservi}[1]{%
4406   \@gls@entry@field{#1}{uservi}%
4407 }
```

`\Glsentryuservi`

```
4408 \newrobustcmd*{\Glsentryuservi}[1]{%
4409   \@Gls@entry@field{#1}{uservi}%
4410 }
```

`\glentryshort` Get the short key (as specified by the short the entry was defined). The argument is the label associated with the entry.

```
4411 \newcommand*{\glentryshort}[1]{\@gls@entry@field{#1}{short}}
```

`\Glsentryshort`

```
4412 \newrobustcmd*{\Glsentryshort}[1]{%
4413   \@Gls@entry@field{#1}{short}%
4414 }
```

`glentryshortpl` Get the short plural key (as specified by the shortplural the entry was defined). The argument is the label associated with the entry.

```
4415 \newcommand*{\glentryshortpl}[1]{\@gls@entry@field{#1}{shortpl}}
```

Glsentryshortpl

```
4416 \newrobustcmd*{\Glsentryshortpl}[1]{%  
4417   \@Gls@entry@field{#1}{shortpl}%  
4418 }
```

\glsentrylong Get the long key (as specified by the long the entry was defined). The argument is the label associated with the entry.

```
4419 \newcommand*{\glsentrylong}[1]{\@Gls@entry@field{#1}{long}}
```

\Glsentrylong

```
4420 \newrobustcmd*{\Glsentrylong}[1]{%  
4421   \@Gls@entry@field{#1}{long}%  
4422 }
```

\glsentrylongpl Get the long plural key (as specified by the longplural the entry was defined). The argument is the label associated with the entry.

```
4423 \newcommand*{\glsentrylongpl}[1]{\@Gls@entry@field{#1}{longpl}}
```

\Glsentrylongpl

```
4424 \newrobustcmd*{\Glsentrylongpl}[1]{%  
4425   \@Gls@entry@field{#1}{longpl}%  
4426 }
```

Short cut macros to access full form:

\glsentryfull

```
4427 \newcommand*{\glsentryfull}[1]{%  
4428   \acrfullformat{\glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%  
4429 }
```

\Glsentryfull

```
4430 \newrobustcmd*{\Glsentryfull}[1]{%  
4431   \acrfullformat{\Glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%  
4432 }
```

\glsentryfullpl

```
4433 \newcommand*{\glsentryfullpl}[1]{%  
4434   \acrfullformat{\glsentrylongpl{#1}}{\acronymfont{\glsentryshortpl{#1}}}%  
4435 }
```

\Glsentryfullpl

```
4436 \newrobustcmd*{\Glsentryfullpl}[1]{%  
4437   \acrfullformat{\Glsentrylongpl{#1}}{\acronymfont{\glsentryshortpl{#1}}}%  
4438 }
```

entrynumberlist Displays the number list as is.

```

4439 \newcommand*\glsentrynumberlist}[1]{%
4440   \glsdoifexists{#1}%
4441   {%
4442     \@gls@entry@field{#1}{numberlist}%
4443   }%
4444 }

```

splaynumberlist Formats the number list for the given entry label. Doesn't work with hyperref.

```

4445 \@ifpackageloaded{hyperref} {%
4446   \newcommand*\glsdisplaynumberlist}[1]{%
4447     \GlossariesWarning
4448     {%
4449       \string\glsdisplaynumberlist\space
4450       doesn't work with hyperref.^^JUsing
4451       \string\glsentrynumberlist\space instead%
4452     }%
4453     \glsentrynumberlist{#1}%
4454   }%
4455 }%
4456 {%
4457   \newcommand*\glsdisplaynumberlist}[1]{%
4458     \glsdoifexists{#1}%
4459     {%
4460       \bgroup

4461         \edef\@glo@label{\glsdetoklabel{#1}}%
4462         \let\@org@glsnumberformat\glsnumberformat
4463         \def\glsnumberformat##1{##1}%
4464         \protected@edef\the@numberlist{%
4465           \csname glo@\@glo@label @numberlist\endcsname}%
4466         \def\@gls@numlist@sep{%
4467           \def\@gls@numlist@nextsep{%
4468             \def\@gls@numlist@lastsep{%
4469               \def\@gls@thislist{%
4470                 \def\@gls@donext@def{%
4471                   \renewcommand\do[1]{%
4472                     \protected@edef\@gls@thislist{%
4473                       \@gls@thislist
4474                       \noexpand\@gls@numlist@sep
4475                       ##1%
4476                     }%
4477                     \let\@gls@numlist@sep\@gls@numlist@nextsep
4478                     \def\@gls@numlist@nextsep{\glsnumlistsep}%
4479                     \@gls@donext@def
4480                     \def\@gls@donext@def{%
4481                       \def\@gls@numlist@lastsep{\glsnumlistlastsep}%
4482                     }%
4483                   }%

```



```

4484         \expandafter \glsnumlistparser \expandafter{\the@numberlist}%
4485         \let\@gls@numlist@sep\@gls@numlist@lastsep
4486         \@gls@thislist
4487     \egroup
4488 }%
4489 }
4490 }

```

`\glsnumlistsep`

```

4491 \newcommand*{\glsnumlistsep}{, }

```

`\glsnumlistlastsep`

```

4492 \newcommand*{\glsnumlistlastsep}{ \& }

```

`\gls hyperlink` Provide a hyperlink to a glossary entry without adding information to the glossary file. The entry needs to be added using a command like `\gls link` or `\gls add` to ensure that the target is defined. The first (optional) argument specifies the link text. The entry name is used by default. The second argument is the entry label.

```

4493 \newcommand*{\gls hyperlink}[2][\gls entrytext{\@glo@label}]{%
4494   \def\@glo@label{#2}%
4495   \@gls link{\glo link prefix\gls detok label{#2}}{#1}}

```

1.12 Adding an entry to the glossary without generating text

The following keys are provided for `\gls add` and `\gls add all`:

```

4496 \define@key{gloss add}{counter}{\def\@gls@counter{#1}}
4497 \define@key{gloss add}{format}{\def\@gls number format{#1}}

```

This key is only used by `\gls add all`:

```

4498 \define@key{gloss add}{types}{\def\@glo@type{#1}}

```

`\gls add[<options>]{<label>}`

Add a term to the glossary without generating any link text. The optional argument indicates which counter to use, and how to format it (using a key-value list) the second argument is the entry label. Note that *<options>* only has two keys: counter and format (the types key will be ignored).

`\gls add`

```

4499 \newrobustcmd*{\gls add}[2][ ]{%

```

Need to move to horizontal mode if not already in it, but only if not in preamble.

```

4500   \@gls@adjustmode
4501   \gls do if exists{#2}%
4502   {%
4503     \def\@gls number format{gls number format}%

```

```

4504 \edef\@gls@counter{\csname glo@\glsdetoklabel{#2}@counter\endcsname}%
4505 \setkeys{glossadd}{#1}%

Store the entry's counter in \theglsentrycounter
4506 \@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.
4507 \@do@wrglossary{#2}%
4508 }%
4509 }

```

@gls@adjustmode

```

4510 \newcommand*{\@gls@adjustmode}{}
4511 \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

```

4512 \newrobustcmd*{\glsaddall}[1][]{%
4513 \edef\@glo@type{\@glo@types}%
4514 \setkeys{glossadd}{#1}%
4515 \forallglsentries[\@glo@type]{\@glo@entry}{%
4516 \glsadd[#1]{\@glo@entry}%
4517 }%
4518 }

```

\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.

```

4519 \newrobustcmd*{\glsaddallunused}[1][\@glo@types]{%
4520 \forallglsentries[#1]{\@glo@entry}%
4521 {%
4522 \ifglsused{\@glo@entry}{\glsadd[format=glsignore]{\@glo@entry}}%
4523 }%
4524 }

```

\glsignore

```

4525 \newcommand*{\glsignore}[1]{}

```

1.13 Creating associated files

The `\writeist` command creates the associated customized `.ist` `makeindex` style file. While defining this command, some characters have their catcodes temporarily changed to ensure they get written to the `.ist` file correctly. The `makeindex` actual character (usually `@`) is redefined to be a `?`, to allow internal commands to be written to the glossary file output file.

The special characters are stored in `\@gls@actualchar`, `\@gls@encapchar`, `\@gls@levelchar` and `\@gls@quotechar` to make them easier to use later, but don't change these values, because the characters are encoded in the command definitions that are used to escape the special characters (which means that the user no longer needs to worry about `makeindex` special characters).

The symbols and numbers label for group headings are hardwired into the `.ist` file as `glssymbols` and `glsnumbers`, the group titles can be translated (so that `\glssymbolsgroupname` replaces `glssymbols` and `\glsnumbersgroupname` replaces `glsnumbers`) using the command `\glsgetgrouptitle` which is defined in `.` This is done to prevent any problem characters in `\glssymbolsgroupname` and `\glsnumbersgroupname` from breaking hyperlinks.

```
\glsopenbrace  Define \glsopenbrace to make it easier to write an opening brace to a file.
4526 \edef\glsopenbrace{\expandafter\@gobble\string\{}}

\glsclosebrace Define \glsclosebrace to make it easier to write an opening brace to a file.
4527 \edef\glsclosebrace{\expandafter\@gobble\string\}}

\glsbackslash  Define \glsbackslash to make it easier to write a backslash to a file.
4528 \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.
4529 \edef\glsquote#1{\string"#1\string"}

\glspercentchar Define \glspercentchar to make it easier to write a percent character to a file.
4530 \edef\glspercentchar{\expandafter\@gobble\string\%}

\glstildechar  Define \glstildechar to make it easier to write a tilde character to a file.
4531 \edef\glstildechar{\string~}

\@glsfirstletter Define the first letter to come after the digits 0,...,9. Only required for xindy.
4532 \ifglsxindy
4533   \newcommand*{\@glsfirstletter}{A}
4534 \fi

\glsfirstletterAfterDigits Sets the first letter to come after the digits 0,...,9.
4535 \ifglsxindy
4536   \newcommand*{\GlsSetXdyFirstLetterAfterDigits}[1]{%
4537     \renewcommand*{\@glsfirstletter}{#1}}
4538 \else
```

```

4539 \newcommand*{\GlsSetXdyFirstLetterAfterDigits}[1]{%
4540 \glsnoxywarning\GlsSetXdyFirstLetterAfterDigits}
4541 \fi

```

`\@glsminrange` Define the minimum number of successive location references to merge into a range.

```

4542 \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.

```

4543 \ifglsxindy
4544 \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
4545 \renewcommand*{\@glsminrange}{#1}}
4546 \else
4547 \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
4548 \glsnoxywarning\GlsSetXdyMinRangeLength}
4549 \fi

```

`\writeist`

```

4550 \ifglsxindy

```

Code to use if xindy is required.

```

4551 \def\writeist{%

```

Define write register if not already defined

```

4552 \ifundef{\glswrite}{\newwrite\glswrite}{}%

```

Update attributes list

```

4553 \@gls@addpredefinedattributes

```

Open the file.

```

4554 \openout\glswrite=\istfilename

```

Write header comment at the start of the file

```

4555 \write\glswrite{;; xindy style file created by the glossaries
4556 package}%
4557 \write\glswrite{;; for document '\jobname' on
4558 \the\year-\the\month-\the\day}%

```

Specify the required styles

```

4559 \write\glswrite{^^J; required styles^^J}
4560 \@for\@xdystyle:=\@xdyrequiredstyles\do{%
4561 \ifx\@xdystyle\@empty
4562 \else
4563 \protected@write\glswrite{{(require
4564 \string"\@xdystyle.xdy\string")}}%
4565 \fi
4566 }%

```

List the allowed attributes (possible values used by the format key)

```

4567 \write\glswrite{^^J%
4568 ; list of allowed attributes (number formats)^^J}%
4569 \write\glswrite{(define-attributes ((\@xdyattributes)))}%

```

Define any additional alphabets

```
4570 \write\glswrite{^^J; user defined alphabets^^J}%
4571 \write\glswrite{\@xdyuseralphabets}%
```

Define location classes.

```
4572 \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.

```
4573 \@for\@gls@classI:=\@gls@xdy@locationlist\do{%
```

Case were $\langle Hprefix \rangle$ is empty:

```
4574 \protected@write\glswrite{}\{(define-location-class
4575 \string"\@gls@classI\string"^^J\space\space\space
4576 (
4577 :sep "{"
4578 \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
4579 :sep "}"
4580 )
4581 ^^J\space\space\space
4582 :min-range-length \@glsminrange^^J%
4583 )
4584 }%
```

Nested iteration over all classes:

```
4585 {%
4586 \@for\@gls@classII:=\@gls@xdy@locationlist\do{%
4587 \protected@write\glswrite{}\{(define-location-class
4588 \string"\@gls@classII-\@gls@classI\string"
4589 ^^J\space\space\space
4590 (
4591 :sep "{"
4592 \csname @gls@xdy@Lclass@\@gls@classII\endcsname\space
4593 :sep "}"
4594 \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
4595 :sep "}"
4596 )
4597 ^^J\space\space\space
4598 :min-range-length \@glsminrange^^J%
4599 )
4600 }%
4601 }%
4602 }%
4603 }%
```

User defined location classes (needs checking for new location format).

```
4604 \write\glswrite{^^J; user defined location classes}%
4605 \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.)

```

4606 \write\glswrite{^^J; define cross-reference class^^J}%
4607 \write\glswrite{(define-crossref-class \string"see\string"
4608 :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.)

```

4609 \write\glswrite{(markup-crossref-list
4610 :class \string"see\string"^^J\space\space\space
4611 :open \string"\string\glsseeformat\string"
4612 :close \string"{\string")}%

```

List the order to sort the classes.

```

4613 \write\glswrite{^^J; define the order of the location classes}%
4614 \write\glswrite{(define-location-class-order
4615 (\@xdylocationclassorder))}%

```

Specify what to write to the start and end of the glossary file.

```

4616 \write\glswrite{^^J; define the glossary markup^^J}%

4617 \write\glswrite{(markup-index^^J\space\space\space
4618 :open \string"\string
4619 \glossarysection[\string\glossarytoctitle]{\string
4620 \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)

```

4621 \@for\@this@ctr:=\@xdycounters\do{%
4622   {%
4623     \@for\@this@attr:=\@xdyattributelist\do{%
4624       \protected\write\glswrite{}\{\string\providecommand*%
4625       \expandafter\string
4626       \csname glsX\@this@ctr X\@this@attr\endcsname[2]%
4627       {%
4628         \string\setentrycounter
4629         [\expandafter\@gobble\string\#1]{\@this@ctr}%
4630         \expandafter\string
4631         \csname\@this@attr\endcsname
4632         {\expandafter\@gobble\string\#2}%
4633       }%
4634     }%
4635   }%
4636 }%
4637 }%

```

Add the end part of the open tag and the rest of the markup-index information:

```

4638 \write\glswrite{%
4639 \string\begin
4640 {theglossary}\string\glossaryheader\glstildechar n\string" ^^J\space
4641 \space\space:close \string"\glspercentchar\glstildechar n\string

```

```

4642         \end{theglossary}\string\glossarypostamble
4643         \glstildechar n\string" ^^J\space\space\space
4644         :tree)}}%

```

Specify what to put between letter groups

```

4645     \write\glswrite{(markup-letter-group-list
4646         :sep \string"\string\glsgroupskip\glstildechar n\string"}}%

```

Specify what to put between entries

```

4647     \write\glswrite{(markup-indexentry
4648         :open \string"\string\relax \string\glresetentrylist
4649         \glstildechar n\string"}}%

```

Specify how to format entries

```

4650     \write\glswrite{(markup-locclass-list :open
4651         \string"\glsoopenbrace\string\glossaryentrynumbers
4652         \glsoopenbrace\string\relax\space \string"^^J\space\space\space
4653         :sep \string", \string"
4654         :close \string"\glsclosebrace\glsclosebrace\string"}}%

```

Specify how to separate location numbers

```

4655     \write\glswrite{(markup-locref-list
4656         :sep \string"\string\delimN\space\string"}}%

```

Specify how to indicate location ranges

```

4657     \write\glswrite{(markup-range
4658         :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.

```

4659     \@onelevel@sanitize\gls@suffixF
4660     \@onelevel@sanitize\gls@suffixFF
4661     \ifx\gls@suffixF\@empty
4662     \else
4663         \write\glswrite{(markup-range
4664             :close "\gls@suffixF" :length 1 :ignore-end)}}%
4665     \fi
4666     \ifx\gls@suffixFF\@empty
4667     \else
4668         \write\glswrite{(markup-range
4669             :close "\gls@suffixFF" :length 2 :ignore-end)}}%
4670     \fi

```

Specify how to format locations.

```

4671     \write\glswrite{^^J; define format to use for locations^^J}%
4672     \write\glswrite{@xdylocref}%

```

Specify how to separate letter groups.

```

4673     \write\glswrite{^^J; define letter group list format^^J}%
4674     \write\glswrite{(markup-letter-group-list
4675         :sep \string"\string\glsgroupskip\glstildechar n\string"}}%

```

Define letter group headings.

```
4676 \write\glswrite{^^J; letter group headings^^J}%
4677 \write\glswrite{(markup-letter-group
4678   :open-head \string\string\glsgroupheading
4679   \glsoopenbrace\string"^^J\space\space\space
4680   :close-head \string\glsclosebrace\string")}%
```

Define additional letter groups.

```
4681 \write\glswrite{^^J; additional letter groups^^J}%
4682 \write\glswrite{\@xdylettergroups}%
```

Define additional sort rules

```
4683 \write\glswrite{^^J; additional sort rules^^J}
4684 \write\glswrite{\@xdysortrules}%
```

Hook for any additional information:

```
4685 \@gls@writeisthook
```

Close the style file

```
4686 \closeout\glswrite
```

Suppress any further calls.

```
4687 \let\writeist\relax
4688 }
4689 \else
```

Code to use if makeindex is required.

```
4690 \edef\@gls@actualchar{\string?}
4691 \edef\@gls@encapchar{\string|}
4692 \edef\@gls@levelchar{\string!}
4693 \edef\@gls@quotechar{\string"}%
4694 \let\GlsSetQuote\gls@nosetquote
4695 \def\writeist{\relax
4696   \ifundef{\glswrite}{\newwrite\glswrite}{}\relax
4697   \openout\glswrite=\istfilename
4698   \write\glswrite{\glspersentchar\space makeindex style file
4699     created by the glossaries package}
4700   \write\glswrite{\glspersentchar\space for document
4701     '\jobname' on \the\year-\the\month-\the\day}
4702   \write\glswrite{actual '@gls@actualchar'}
4703   \write\glswrite{encap '@gls@encapchar'}
4704   \write\glswrite{level '@gls@levelchar'}
4705   \write\glswrite{quote '@gls@quotechar'}
4706   \write\glswrite{keyword \string\string\glossaryentry\string"}
4707   \write\glswrite{preamble \string\string\glossarysection[\string
4708     \glossarytoctitle]{\string\glossarytitle}\string
4709     \glossarypreamble\string\n\string\begin{theglossary}\string
4710     \glossaryheader\string\n\string"}
4711   \write\glswrite{postamble \string\string%\string\n\string
4712     \end{theglossary}\string\glossarypostamble\string\n
4713     \string"}
4714   \write\glswrite{group_skip \string\string\glsgroupskip\string\n
```



```

4715     \string"}
4716     \write\glswrite{item_0 \string"\string%\string\n\string"}
4717     \write\glswrite{item_1 \string"\string%\string\n\string"}
4718     \write\glswrite{item_2 \string"\string%\string\n\string"}
4719     \write\glswrite{item_01 \string"\string%\string\n\string"}
4720     \write\glswrite{item_x1
4721         \string"\string\relax \string\glresetentrylist\string\n
4722         \string"}
4723     \write\glswrite{item_12 \string"\string%\string\n\string"}
4724     \write\glswrite{item_x2
4725         \string"\string\relax \string\glresetentrylist\string\n
4726         \string"}

4727     \write\glswrite{delim_0 \string"\string\{\string
4728         \glossaryentrynumbers\string\{\string\relax \string"}
4729     \write\glswrite{delim_1 \string"\string\{\string
4730         \glossaryentrynumbers\string\{\string\relax \string"}
4731     \write\glswrite{delim_2 \string"\string\{\string
4732         \glossaryentrynumbers\string\{\string\relax \string"}
4733     \write\glswrite{delim_t \string"\string\}\string\}\string"}
4734     \write\glswrite{delim_n \string"\string\delimN \string"}
4735     \write\glswrite{delim_r \string"\string\delimR \string"}
4736     \write\glswrite{headings_flag 1}
4737     \write\glswrite{heading_prefix
4738         \string"\string\glsgroupheading\string\{\string"}
4739     \write\glswrite{heading_suffix
4740         \string"\string\}\string\relax
4741         \string\glresetentrylist \string"}
4742     \write\glswrite{symhead_positive \string"glssymbols\string"}
4743     \write\glswrite{numhead_positive \string"glnumbers\string"}
4744     \write\glswrite{page_compositor \string"glscpositor\string"}
4745     \@gls@escbsdq\gls@suffixF
4746     \@gls@escbsdq\gls@suffixFF
4747     \ifx\gls@suffixF\@empty
4748     \else
4749         \write\glswrite{suffix_2p \string"\gls@suffixF\string"}
4750     \fi
4751     \ifx\gls@suffixFF\@empty
4752     \else
4753         \write\glswrite{suffix_3p \string"\gls@suffixFF\string"}
4754     \fi

```

Hook for any additional information:

```

4755     \@gls@writeisthook

```

Close the file and disable \writeist.

```

4756     \closeout\glswrite
4757     \let\writeist\relax
4758 }
4759 \fi

```

SetWriteIstHook Allow user to append information to the style file.

```
4760 \newcommand*\GlsSetWriteIstHook}[1]{\renewcommand*\@gls@writeisthook}{#1}}
4761 \@onlypremakeg\GlsSetWriteIstHook
```

ls@writeisthook

```
4762 \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.

```
4763 \ifglxsindy
4764 \newcommand*\GlsSetQuote}[1]{\glsnomakeindexwarning\GlsSetQuote}
4765 \newcommand*\gls@nosetquote}[1]{\glsnomakeindexwarning\GlsSetQuote}
4766 \else
4767 \newcommand*\GlsSetQuote}[1]{\edef\@gls@quotechar{\string#1}}%
```

If German is in use, set the extra makeindex option so makeglossaries can pick it up.

```
4768 \@ifpackageloaded{tracklang}%
4769 {%
4770 \IfTrackedLanguage{german}%
4771 {%
4772 \def\@gls@extramakeindexopts{-g}%
4773 }%
4774 }%
4775 }%
4776 {}%
```

Need to redefine \@gls@checkquote

```
4777 \edef\@gls@docheckquotedef{%
4778 \noexpand\def\noexpand\@gls@checkquote####1#1####2#1####3\noexpand\null{%
4779 \noexpand\@gls@tmpb=\noexpand\expandafter\noexpand\@gls@checkedmkidx}%
4780 \noexpand\toks@={####1}%
4781 \noexpand\ifx\noexpand\null####2\noexpand\null
4782 \noexpand\ifx\noexpand\null####3\noexpand\null
4783 \noexpand\edef\noexpand\@gls@checkedmkidx{%
4784 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
4785 \noexpand\def\noexpand\@gls@checkquote{\noexpand\relax}%
4786 \noexpand\else
4787 \noexpand\edef\noexpand\@gls@checkedmkidx{%
4788 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
4789 \noexpand\@gls@quotechar\noexpand\@gls@quotechar
4790 \noexpand\@gls@quotechar\noexpand\@gls@quotechar}%
4791 \noexpand\def\noexpand\@gls@checkquote{%
4792 \noexpand\@gls@checkquote####3\noexpand\null}%
4793 \noexpand\fi
4794 \noexpand\else
4795 \noexpand\edef\noexpand\@gls@checkedmkidx{%
4796 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
4797 \noexpand\@gls@quotechar\noexpand\@gls@quotechar}%
4798 \noexpand\ifx\noexpand\null####3\noexpand\null
4799 \noexpand\def\noexpand\@gls@checkquote{%
```

```

4800         \noexpand\@gls@checkquote####2#1#1\noexpand\null}%
4801     \noexpand\else
4802         \noexpand\def\noexpand\@gls@checkquote{%
4803             \noexpand\@gls@checkquote####2#1###3\noexpand\null}%
4804     \noexpand\fi
4805 \noexpand\fi
4806 \noexpand\@gls@checkquote
4807 }%
4808 }%
4809 \@gls@docheckquotedef
4810 \edef\@gls@docheckquotedef{%
4811     \noexpand\renewcommand{\noexpand\@gls@checkmkidxchars}[1]{%
4812         \noexpand\def\noexpand\@gls@checkedmkidx{%
4813             \noexpand\expandafter\noexpand\@gls@checkquote####1\noexpand\@nil
4814             #1#1\noexpand\null
4815             \noexpand\expandafter\noexpand\@gls@updatechecked
4816             \noexpand\@gls@checkedmkidx{####1}%
4817             \noexpand\def\noexpand\@gls@checkedmkidx{%
4818                 \noexpand\expandafter\noexpand\@gls@checkescquote####1\noexpand\@nil
4819                 \expandonce{\csname#1\endcsname}\expandonce{\csname#1\endcsname}%
4820                 \noexpand\null
4821                 \noexpand\expandafter\noexpand\@gls@updatechecked
4822                 \noexpand\@gls@checkedmkidx{####1}%
4823                 \noexpand\def\noexpand\@gls@checkedmkidx{%
4824                     \noexpand\expandafter\noexpand\@gls@checkescactual####1\noexpand\@nil
4825                     \noexpand\?\noexpand\?\noexpand\null
4826                     \noexpand\expandafter\noexpand\@gls@updatechecked
4827                     \noexpand\@gls@checkedmkidx{####1}%
4828                     \noexpand\def\noexpand\@gls@checkedmkidx{%
4829                         \noexpand\expandafter\noexpand\@gls@checkactual####1\noexpand\@nil
4830                         \noexpand?\noexpand?\noexpand\null
4831                         \noexpand\expandafter\noexpand\@gls@updatechecked
4832                         \noexpand\@gls@checkedmkidx{####1}%
4833                         \noexpand\def\noexpand\@gls@checkedmkidx{%
4834                             \noexpand\expandafter\noexpand\@gls@checkbar####1\noexpand\@nil
4835                             \noexpand|\noexpand|\noexpand\null
4836                             \noexpand\expandafter\noexpand\@gls@updatechecked
4837                             \noexpand\@gls@checkedmkidx{####1}%
4838                             \noexpand\def\noexpand\@gls@checkedmkidx{%
4839                                 \noexpand\expandafter\noexpand\@gls@checkescbar####1\noexpand\@nil
4840                                 \noexpand\\|\noexpand\\|\noexpand\null
4841                                 \noexpand\expandafter\noexpand\@gls@updatechecked
4842                                 \noexpand\@gls@checkedmkidx{####1}%
4843                                 \noexpand\def\noexpand\@gls@checkedmkidx{%
4844                                     \noexpand\expandafter\noexpand\@gls@checklevel####1\noexpand\@nil
4845                                     \noexpand!\noexpand!\noexpand\null
4846                                     \noexpand\expandafter\noexpand\@gls@updatechecked
4847                                     \noexpand\@gls@checkedmkidx{####1}%
4848                                 }%

```

```

4849 }%
4850 \@gls@docheckquotedef
4851 \edef\@gls@docheckquotedef{%
4852   \noexpand\def\noexpand\@gls@checkescquote####1%
4853     \expandonce{\csname#1\endcsname}####2\expandonce{\csname#1\endcsname}%
4854     ####3\noexpand\null{%
4855       \noexpand\@gls@tmpb=\noexpand\expandafter{\noexpand\@gls@checkedmkidx}%
4856       \noexpand\toks@={####1}%
4857       \noexpand\ifx\noexpand\null####2\noexpand\null
4858       \noexpand\ifx\noexpand\null####3\noexpand\null
4859       \noexpand\edef\noexpand\@gls@checkedmkidx{%
4860         \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
4861       \noexpand\def\noexpand\@gls@checkescquote{\noexpand\relax}%
4862       \noexpand\else
4863       \noexpand\edef\noexpand\@gls@checkedmkidx{%
4864         \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
4865         \noexpand\@gls@quotechar\noexpand\string\expandonce{%
4866           \csname#1\endcsname}\noexpand\@gls@quotechar
4867         \noexpand\@gls@quotechar\noexpand\string\expandonce{%
4868           \csname#1\endcsname}\noexpand\@gls@quotechar}%
4869       \noexpand\def\noexpand\@gls@checkescquote{%
4870         \noexpand\@gls@checkescquote####3\noexpand\null}%
4871       \noexpand\fi
4872       \noexpand\else
4873       \noexpand\edef\noexpand\@gls@checkedmkidx{%
4874         \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
4875         \noexpand\@gls@quotechar\noexpand\string
4876         \expandonce{\csname#1\endcsname}\noexpand\@gls@quotechar}%
4877       \noexpand\ifx\noexpand\null####3\noexpand\null
4878       \noexpand\def\noexpand\@gls@checkescquote{%
4879         \noexpand\@gls@checkescquote####2\expandonce{\csname#1\endcsname}%
4880         \expandonce{\csname#1\endcsname}\noexpand\null}%
4881       \noexpand\else
4882       \noexpand\def\noexpand\@gls@checkescquote{%
4883         \noexpand\@gls@checkescquote####2\expandonce{\csname#1\endcsname}%
4884         ####3\noexpand\null}%
4885       \noexpand\fi
4886       \noexpand\fi
4887       \noexpand\@gls@checkescquote
4888     }%
4889 }%
4890 \@gls@docheckquotedef
4891 }
4892 \newcommand*{\gls@nosetquote}[1]{\PackageError{glossaries}%
4893   {\string\GlsSetQuote\space not permitted here}%
4894   {Move \string\GlsSetQuote\space earlier in the preamble, as
4895    soon as possible after glossaries.sty has been loaded}}
4896 \fi

```

ramakeindexopts

```
4897 \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`

```
4898 \newcommand{\noist}{%
```

Update attributes list

```
4899 \@gls@addpredefinedattributes
```

```
4900 \let\writeist\relax
```

```
4901 }
```

`\@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`

```
4902 \newcommand*{\@makeglossary}[1]{%
```

```
4903 \ifglossaryexists{#1}%
```

```
4904 {%
```

Only create a new write if `savewrites=false` otherwise create a token to collect the information.

```
4905 \ifglssavewrites
```

```
4906 \expandafter\newtoks\csname glo@#1@filetok\endcsname
```

```
4907 \else
```

```
4908 \expandafter\newwrite\csname glo@#1@file\endcsname
```

```
4909 \expandafter\@glsopenfile\csname glo@#1@file\endcsname{#1}%
```

```
4910 \fi
```

```
4911 \@gls@renewglossary
```

```
4912 \writeist
```

```
4913 }%
```

```
4914 {%
```

```
4915 \PackageError{glossaries}%
```

```
4916 {Glossary type ‘#1’ not defined}%
```

```
4917 {New glossaries must be defined before using \string\makeglossary}%
```

```
4918 }%
```

```
4919 }
```

`\@glsopenfile` Open write file associated with the given glossary.

```
4920 \newcommand*{\@glsopenfile}[2]{%
```

```

4921 \immediate\openout#1=\jobname.\csname @glotype@#2@out\endcsname
4922 \PackageInfo{glossaries}{Writing glossary file
4923   \jobname.\csname @glotype@#2@out\endcsname}%
4924 }

```

\@closegls

```

4925 \newcommand*{\@closegls}[1]{%
4926   \closeout\csname glo@#1@file\endcsname
4927 }
4928 % \end{macrocode}
4929 %\end{macro}
4930 %
4931 %\begin{macro}{\@gls@automake}
4932 %\changes{4.08}{2014-07-30}{new}
4933 % \begin{macrocode}
4934 \ifglxindy
4935 \newcommand*{\@gls@automake}[1]{%
4936   \ifglossaryexists{#1}
4937   {%
4938     \@closegls{#1}%
4939     \ifdefstring{\glsorder}{letter}%
4940     {\def\@gls@order{-M ord/letorder }}%
4941     {\let\@gls@order\@empty}%
4942     \ifcsundef{\xdy@#1@language}%
4943     {\let\@gls@langmod\@xdy@main@language}%
4944     {\letcs\@gls@langmod{\xdy@#1@language}}%
4945     \edef\@gls@dothiswrite{\noexpand\write18{xindy
4946       -I xindy
4947       \@gls@order
4948       -L \@gls@langmod\space
4949       -M \gls@istfilebase\space
4950       -C \gls@codepage\space
4951       -t \jobname.\csuse{@glotype@#1@log}
4952       -o \jobname.\csuse{@glotype@#1@in}
4953       \jobname.\csuse{@glotype@#1@out}}}%
4954     }%
4955     \@gls@dothiswrite
4956   }%
4957   {%
4958     \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
4959   }%
4960 }
4961 \else
4962 \newcommand*{\@gls@automake}[1]{%
4963   \ifglossaryexists{#1}
4964   {%
4965     \@closegls{#1}%
4966     \ifdefstring{\glsorder}{letter}%
4967     {\def\@gls@order{-l }}%

```

```

4968     {\let\@gls@order\@empty}%
4969     \edef\@gls@dothiswrite{\noexpand\write18{makeindex \@gls@order
4970     -s \istfilename\space
4971     -t \jobname.\csuse{\@glotype@#1@log}
4972     -o \jobname.\csuse{\@glotype@#1@in}
4973     \jobname.\csuse{\@glotype@#1@out}}}%
4974     }%
4975     \@gls@dothiswrite
4976   }%
4977   {%
4978     \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
4979   }%
4980 }
4981 \fi

```

`\makeglossaries` Issue warning that `\makeglossaries` hasn't been used.

```

4982 \newcommand*{\@warn@nomakeglossaries}{}

```

Only use this if warning if `\printglossary` has been used without `\makeglossaries`

```

4983 \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`

```

4984 \newcommand*{\makeglossaries}{%

```

Define the write used for style file also used for all other output files if `savewrites=true`.

```

4985   \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.

```

4986   \protected@write\@auxout{}{\string\providecommand\string\@glsorder[1]{} }
4987   \protected@write\@auxout{}{\string\providecommand\string\@istfilename[1]{} }

```

If `\@gls@extramakeindexopts` has been defined, write it:

```

4988   \ifundef\@gls@extramakeindexopts
4989   {%
4990     {%
4991       \protected@write\@auxout{}{\string\providecommand
4992       \string\@gls@extramakeindexopts[1]{} }
4993       \protected@write\@auxout{}{\string\@gls@extramakeindexopts
4994       {\@gls@extramakeindexopts}}%
4995     }%

```

Write the name of the style file to the aux file (needed by `makeglossaries`)

```

4996   \protected@write\@auxout{}{\string\@istfilename{\istfilename}}%
4997   \protected@write\@auxout{}{\string\@glsorder{\glsorder}}

```

Iterate through each glossary type and activate it.

```
4998 \for\@glo@type:=\@glo@types\do{%
4999 \ifthenelse{\equal{\@glo@type}{}}{}}{%
5000 \makeglossary{\@glo@type}}%
5001 }%
```

New glossaries must be created before `\makeglossaries` so disable `\newglossary`.

```
5002 \renewcommand*\newglossary[4][]{%
5003 \PackageError{glossaries}{New glossaries
5004 must be created before \string\makeglossaries}{You need
5005 to move \string\makeglossaries\space after all your
5006 \string\newglossary\space commands}}%
```

Any subsequence instances of this command should have no effect

```
5007 \let\makeglossary\relax
5008 \let\makeglossary\relax
5009 \let\makeglossaries\relax
```

Disable all commands that have no effect after `\makeglossaries`

```
5010 \@disable@onlypremakeg
```

Allow see key:

```
5011 \let\gls@checkseeallowed\relax
```

Suppress warning about no `\makeglossaries`

```
5012 \let\warn@nomakeglossaries\relax
```

Activate warning about missing `\printglossary`

```
5013 \def\warn@noprintglossary{%
5014 \ifdefstring{\@glo@types}{,}%
5015 {%
5016 \GlossariesWarningNoLine{No glossaries have been defined}%
5017 }%
5018 {%
5019 \GlossariesWarningNoLine{No \string\printglossary\space
5020 or \string\printglossaries\space
5021 found. ^^J(Remove \string\makeglossaries\space if you
5022 don't want any glossaries.) ^^JThis document will not
5023 have a glossary}%
5024 }%
5025 }%
```

Declare list parser for `\glsdisplaynumberlist`

```
5026 \ifglssavenumberlist
5027 \edef\@gls@doddeflistparser{\noexpand\DeclareListParser
5028 {\noexpand\glsnumlistparser}{\delimN}}%
5029 \@gls@doddeflistparser
5030 \fi
```

Prevent user from also using `\makenoidxglossaries`

```
5031 \let\makenoidxglossaries\@no@makeglossaries
```


Prohibit sort key in printgloss family:

```
5032 \renewcommand*{\@printgloss@setsort}{%
5033 \let\@glo@assign@sortkey\@glo@no@assign@sortkey
5034 }%
```

Check the automake setting:

```
5035 \ifglsautomake
5036 \renewcommand*{\@gls@doautomake}{%
5037 \@for\@gls@type:=\@glo@types\do{%
5038 \ifdefempty{\@gls@type}{}%
5039 {\@gls@automake{\@gls@type}}%
5040 }%
5041 }%
5042 \fi
5043 }
```

Must occur in the preamble:

```
5044 \@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`

```
5045 \let\makeglossary\makeglossaries
```

If `\makeglossaries` hasn't been used, issue a warning. Also issue a warning if neither `\printglossaries` nor `\printglossary` have been used.

```
5046 \AtEndDocument{%
5047 \warn@nomakeglossaries
5048 \warn@noprintglossary
5049 }
```

`noidxglossaries` Analogous to `\makeglossaries` this activates the commands needed for `\printnoidxglossary`

```
5050 \newcommand*{\makenoidxglossaries}{%
```

Redefine empty glossary warning:

```
5051 \renewcommand{\@gls@noref@warn}[1]{%
5052 \GlossariesWarning{Empty glossary for
5053 \string\printnoidxglossary[type={##1}].
5054 Rerun may be required (or you may have forgotten to use
5055 commands like \string\gls)}}%
5056 }%
```

Don't escape makeindex/xindy characters

```
5057 \let\@gls@checkmkidxchars\@gobble
```

Write glossary information to aux instead of glossary files

```
5058 \let\@do@wrglossary\gls@noidxglossary
```

Switch on group headings that use the character code:

```
5059 \let\@gls@getgrouptitle\@gls@noidx@getgrouptitle
```

Allow see key:

```
5060 \let\gls@checkseeallowed\relax
```

Redefine cross-referencing macro:

```
5061 \renewcommand{\@do@seeglossary}[2]{%
5062   \edef\@gls@label{\glsdetoklabel{##1}}%
5063   \protected@write\@auxout{}{%
5064     \string\@gls@reference
5065     {\csname glo@\@gls@label @type\endcsname}%
5066     {\@gls@label}%
5067     {%
5068       \string\glsseeformat##2}%
5069     }%
5070   }%
5071 }%
```

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.

```
5072 \AtBeginDocument
5073 {%
5074   \write\@auxout{\string\providecommand\string\@gls@reference[3]{}}%
5075 }%
```

Change warning about no glossaries

```
5076 \def\warn@noprintglossary{%
5077   \GlossariesWarningNoLine{No \string\printnoidxglossary\space
5078     or \string\printnoidxglossaries ^^J
5079     found. (Remove \string\makenoidxglossaries\space if you
5080     don't want any glossaries.)^^JThis document will not have a glossary}%
5081 }%
```

Suppress warning about no \makeglossaries

```
5082 \let\warn@nomakeglossaries\relax
```

Prevent user from also using \makeglossaries

```
5083 \let\makeglossaries\@no@makeglossaries
```

Allow sort key in printgloss family:

```
5084 \renewcommand*{\@printgloss@setsort}{%
5085   \let\@glo@assign@sortkey\@glo@assign@sortkey
```

Initialise default sort order:

```
5086   \def\@glo@sorttype{\@glo@default@sorttype}%
5087 }%
```

All entries must be defined in the preamble:

```

5088 \renewcommand*{\new@glossaryentry}[2]{%
5089 \PackageError{glossaries}{Glossary entries must be
5090 defined in the preamble^^Jwhen you use
5091 \string\makenoidxglossaries}%
5092 {Either move your definitions to the preamble or use
5093 \string\makeglossaries}%
5094 }%

Redefine \glsentrynumberlist
5095 \renewcommand*{\glsentrynumberlist}[1]{%
5096 \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
5097 \ifdef\@gls@loclist
5098 {%
5099 \glsnoidxloclist{\@gls@loclist}%
5100 }%
5101 {%
5102 ??\glsdoifexists{##1}%
5103 {%
5104 \GlossariesWarning{Missing location list for ‘##1’. Either
5105 a rerun is required or you haven’t referenced the entry}%
5106 }%
5107 }%
5108 }%

Redefine \glsdisplaynumberlist
5109 \renewcommand*{\glsdisplaynumberlist}[1]{%
5110 \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
5111 \ifdef\@gls@loclist
5112 {%
5113 \def\@gls@noidxloclist@sep{%
5114 \def\@gls@noidxloclist@sep{%
5115 \def\@gls@noidxloclist@sep{%
5116 \glsnumlistsep
5117 }%
5118 \def\@gls@noidxloclist@finalsep{\glsnumlistlastsep}%
5119 }%
5120 }%
5121 \def\@gls@noidxloclist@finalsep{}}%
5122 \def\@gls@noidxloclist@prev{}}%
5123 \forlistloop{\glsnoidxdisplayloclisthandler}{\@gls@loclist}%
5124 \@gls@noidxloclist@finalsep
5125 \@gls@noidxloclist@prev
5126 }%
5127 {%
5128 ??\glsdoifexists{##1}%
5129 {%
5130 \GlossariesWarning{Missing location list for ‘##1’. Either
5131 a rerun is required or you haven’t referenced the entry}%
5132 }%

```

```

5133 }%
5134 }%

```

Provide a generic way of iterating through the number list:

```

5135 \renewcommand*{\glsnumberlistloop}[3]{%
5136   \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
5137   \let\@gls@org@glsnoidxdisplayloc\glsnoidxdisplayloc
5138   \let\@gls@org@glssееformat\glssееformat
5139   \let\glsnoidxdisplayloc##2\relax
5140   \let\glssееformat##3\relax
5141   \ifdef\@gls@loclist
5142   {%
5143     \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
5144   }%
5145   {%
5146     ??\glsdoifexists{##1}%
5147     {%
5148       \GlossariesWarning{Missing location list for ‘##1’. Either
5149       a rerun is required or you haven’t referenced the entry}%
5150     }%
5151   }%
5152   \let\glsnoidxdisplayloc\@gls@org@glsnoidxdisplayloc
5153   \let\glssееformat\@gls@org@glssееformat
5154 }%

```

Modify sanitize sort function

```

5155 \let\@gls@sanitizesort\@gls@noidx@sanitizesort
5156 \let\@gls@nosanitizesort\@gls@noidx@nosanitizesort
5157 \@gls@noidx@setsanitizesort
5158 }

```

Preamble-only command:

```

5159 \@onlypreamble{\makenoidxglossaries}

```

`\glsnumberlistloop` `\glsnumberlistloop{<label>}{<handler>}`

```

5160 \newcommand*{\glsnumberlistloop}[2]{%
5161   \PackageError{glossaries}{\string\glsnumberlistloop\space
5162   only works with \string\makenoidxglossaries}{}%
5163 }

```

`\listloophandler` Handler macro for `\glsnumberlistloop`. (The argument should be in the form `\glsnoidxdisplayloc {<prefix>}{<counter>}{<format>}{<n>}`)

```

5164 \newcommand*{\glsnoidxnumberlistloophandler}[1]{%
5165   #1%
5166 }

```

`@makeglossaries` Can’t use both `\makeglossaries` and `\makenoidxglossaries`

```

5167 \newcommand*{\@no@makeglossaries}{%
5168   \PackageError{glossaries}{You can't use both
5169   \string\makeglossaries\space and \string\makenoidxglossaries}%
5170   {Either use one or other (or none) of those commands but not both
5171   together.}%
5172 }

```

`@gls@noref@warn` Warning when no instances of `\@gls@reference` found.

```

5173 \newcommand{\@gls@noref@warn}[1]{%
5174   \GlossariesWarning{\string\makenoidxglossaries\space
5175   is required to make \string\printnoidxglossary[type={#1}] work}%
5176 }

```

`s@noidxglossary` Write the glossary information to the aux file:

```

5177 \newcommand*{\@gls@noidxglossary}{%
5178   \protected@write\@auxout{}{%
5179     \string\@gls@reference
5180     {\csname glo@\@gls@label @type\endcsname}%
5181     {\@gls@label}%
5182     {\string\glsnoidxdisplayloc
5183      {\@glo@counterprefix}%
5184      {\@gls@counter}%
5185      {\@glsnumberformat}%
5186      {\@glslocref}%
5187     }%
5188   }%
5189 }

```

1.14 Writing information to associated files

`\istfile` Deprecated.

```

5190 \def\istfile{\glswrite}

```

At the end of the document, the files should be created if `savewrites=true`.

```

5191 \AtEndDocument{%
5192   \glswritefiles
5193 }

```

`\@glswritefiles` Only write the files if `savewrites=true`

```

5194 \newcommand*{\@glswritefiles}{%

```

Iterate through all the glossaries

```

5195   \forallglossaries{\@glo@type}{%

```

Check for empty glossaries (patch provided by Patrick Häcker)

```

5196     \ifcsundef{glo@\@glo@type @filetok}%
5197     {%
5198       \def\gls@tmp{}%

```

```

5199     }%
5200     {%
5201         \edef\gls@tmp{\expandafter\the
5202             \csname glo@%glo@type @filetok\endcsname}%
5203     }%
5204     \ifx\gls@tmp\@empty
5205         \ifx\@glo@type\glsdefaulttype
5206             \GlossariesWarningNoLine{Glossary ‘\@glo@type’ has no
5207                 entries.^^JRemember to use package option ‘nomain’ if
5208 you
5209                 don’t want to^^Juse the main glossary}%
5210         \else
5211             \GlossariesWarningNoLine{Glossary ‘\@glo@type’ has no
5212                 entries}%
5213         \fi
5214     \else
5215         \@glsopenfile{\glswrite}{\@glo@type}%
5216         \immediate\write\glswrite{%
5217             \expandafter\the
5218                 \csname glo@%glo@type @filetok\endcsname}%
5219         \immediate\closeout\glswrite
5220     \fi
5221 }%
5222 }

```

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`

```

5223 \if@gls@docloaded
5224 \else
5225   \renewcommand*{\glossary}[1][main]{\gls@glossary{#1}}
5226 \fi

```

The associated number should be stored in `\theglentrycounter` before using `\gls@glossary`.

`\gls@glossary`

```

5227 \newcommand*{\gls@glossary}[1]{%
5228   \@gls@glossary{#1}%
5229 }

```

`\@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.

```
5230 \newcommand*{\@gls@glossary}[2]{%
5231   \if@gls@debug
5232     \PackageInfo{glossaries}{wrglossary(#1)(#2)}%
5233   \fi
5234   \index{#2}%
5235 }
```

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`

```
5236 \newcommand{\@gls@renewglossary}{%
5237   \gdef\@gls@glossary##1{\@bsphack\begin@group\gls@wrglossary{##1}}%
5238   \let\@gls@renewglossary\@empty
5239 }
```

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`

```
5240 \newcommand*{\gls@wrglossary}[2]{%
5241   \ifglssavewrites
5242     \protected@edef\@gls@tmp{\the\csname glo@#1@filetok\endcsname#2}%
5243     \expandafter\global\expandafter\csname glo@#1@filetok\endcsname
5244       \expandafter{\@gls@tmp^^J}%
5245   \else
5246     \ifcsdef{glo@#1@file}%
5247     {%
5248       \expandafter\protected@write\csname glo@#1@file\endcsname{%
5249         \gls@disablepagerefexpansion}{#2}%
5250     }%
5251     {%
5252       \ifignoredglossary{#1}{}%
5253       {%
5254         \GlossariesWarning{No file defined for glossary ‘#1’}%
5255       }%
5256     }%
5257   \fi
5258   \endgroup\@esphack
5259 }
```

`\@do@wrglossary`

```
5260 \newcommand*{\@do@wrglossary}[1]{%
5261   \glswriteentry{#1}{\@do@wrglossary{#1}}%
5262 }
```

`\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.

```
5263 \newcommand*{\glswriteentry}[2]{%
5264   \ifglindexonlyfirst
5265     \ifglused{#1}{#2}%
5266   \else
5267     #2%
5268   \fi
5269 }
```

`protected@pagefmts` List of page formats to be protected against expansion.

```
5270 \newcommand{\gls@protected@pagefmts}{%
5271   \gls@numberpage,\gls@alphpage,\gls@Alphpage,\gls@romanpage,\gls@Romanpage,\gls@arabicpage%
5272 }
```

`pagerefexpansion`

```
5273 \newcommand*{\gls@disablepagerefexpansion}{%
5274   \@for\@gls@this:=\gls@protected@pagefmts\do
5275   {%
5276     \expandafter\let\@gls@this\relax
5277   }%
5278 }
```

`\gls@alphpage`

```
5279 \newcommand*{\gls@alphpage}{\@alph\c@page}
```

`\gls@Alphpage`

```
5280 \newcommand*{\gls@Alphpage}{\@Alph\c@page}
```

`\gls@numberpage`

```
5281 \newcommand*{\gls@numberpage}{\number\c@page}
```

`\gls@arabicpage`

```
5282 \newcommand*{\gls@arabicpage}{\@arabic\c@page}
```

`\gls@romanpage`

```
5283 \newcommand*{\gls@romanpage}{\romannumeral\c@page}
```

`\gls@Romanpage`

```
5284 \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).


```

5285 \newcommand*{\glsaddprotectedpagefmt}[1]{%
5286   \eappto\gls@protected@pagefmts{,\expandonce{\csname gls#1page\endcsname}}}%
5287   \csedef{gls#1page}{\expandonce{\csname#1\endcsname}\noexpand\c@page}%
5288   \eappto\@wrglossarynumberhook{%
5289     \noexpand\let\expandonce{\csname org@gls#1\endcsname}%
5290     \expandonce{\csname#1\endcsname}%
5291     \noexpand\def\expandonce{\csname#1\endcsname}{%
5292       \noexpand\@wrglossary@pageformat
5293       \expandonce{\csname gls#1page\endcsname}%
5294       \expandonce{\csname org@gls#1\endcsname}%
5295     }%
5296   }%
5297 }

```

`ssarynumberhook` Hook used by `\@do@wrglossary`

```

5298 \newcommand*\@wrglossarynumberhook{}

```

`sary@pageformat`

```

5299 \newcommand{\@wrglossary@pageformat}[3]{%
5300   \ifx#3\c@page #1\else #2#3\fi
5301 }

```

`owprimitivemods` Conditional to determine whether or not `\@do@wrglossary` should be allowed to temporarily redefine `\the` and `\number`.

```

5302 \newif\ifglswrallowprimitivemods
5303 \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.

```

5304 \newcommand*{\@do@wrglossary}[1]{%
5305   \begingroup

```

First a bit of hackery to prevent premature expansion of `\c@page`. Store original definitions:

```

5306   \let\orgthe\the
5307   \let\orgnumber\number

5308   \let\orgarabic\@arabic
5309   \let\orgromannumeral\romannumeral
5310   \let\orgalph\@alph
5311   \let\orgAlph\@Alph
5312   \let\orgRoman\@Roman

```

Redefine:

```

5313   \ifglswrallowprimitivemods
5314     \def\the##1{%
5315       \ifx##1\c@page \gls@numberpage\else\orgthe##1\fi}%
5316     \def\number##1{%
5317       \ifx##1\c@page \gls@numberpage\else\orgnumber##1\fi}%
5318   \fi

```

```

5319 \def\@arabic##1{%
5320 \ifx##1\c@page \gls@arabicpage\else\orgarabic##1\fi}%
5321 \def\romannumeral##1{%
5322 \ifx##1\c@page \gls@romanpage\else\orgromannumeral##1\fi}%
5323 \def\@Roman##1{%
5324 \ifx##1\c@page \gls@Romanpage\else\orgRoman##1\fi}%
5325 \def\@alph##1{%
5326 \ifx##1\c@page \gls@alphpage\else\orgalph##1\fi}%
5327 \def\@Alph##1{%
5328 \ifx##1\c@page \gls@Alphpage\else\orgAlph##1\fi}%

```

Add hook to allow for other number formats:

```
5329 \@wrglossarynumberhook
```

Prevent expansion:

```
5330 \gls@disablepagerefexpansion
```

Now store location in \@glslocref:

```

5331 \protected@xdef\@glslocref{\theHglentrycounter}%
5332 \endgroup

```

Escape any special characters

```
5333 \@gls@checkmkidxchars\@glslocref
```

Check if the hyper-location is the same as the location and set the hyper prefix.

```

5334 \expandafter\ifx\theHglentrycounter\theHglentrycounter\relax
5335 \def\@glo@counterprefix{}%
5336 \else
5337 \protected@edef\@glsHlocref{\theHglentrycounter}%
5338 \@gls@checkmkidxchars\@glsHlocref
5339 \edef\@do@gls@getcounterprefix{\noexpand\@gls@getcounterprefix
5340 {\@glslocref}{\@glsHlocref}%
5341 }%
5342 \@do@gls@getcounterprefix
5343 \fi

```

De-tok label if required

```
5344 \edef\@gls@label{\glsdetoklabel{#1}}%
```

Write the information to file:

```

5345 \@do@wrglossary
5346 }

```

@do@wrglossary

```
5347 \newcommand*{\@do@wrglossary}{%
```

Determine whether to use xindy or makeindex syntax

```
5348 \ifglsxindy
```

Need to determine if the formatting information starts with a (or) indicating a range.

```

5349 \expandafter\@glo@check@mkidxrangechar\@glsnumberformat\@nil
5350 \def\@glo@range{}%
5351 \expandafter\if\@glo@prefix(\relax

```

```

5352     \def\@glo@range{:open-range}%
5353 \else
5354     \expandafter\if\@glo@prefix)\relax
5355     \def\@glo@range{:close-range}%
5356 \fi
5357 \fi

Write to the glossary file using xindy syntax.
5358 \gls@glossary{\csname glo@\@gls@label @type\endcsname}{%
5359 (indexentry :tkey (\csname glo@\@gls@label @index\endcsname)

5360 :locref \string"\@glo@counterprefix}{\@gls@locref}\string" %
5361 :attr \string"\@gls@counter\@glo@suffix\string"
5362 \@glo@range
5363 )
5364 }%
5365 \else

```

Convert the format information into the format required for makeindex

```

5366 \@set@glo@numformat{\@glo@numfmt}{\@gls@counter}{\@gls@numberformat}%
5367 {\@glo@counterprefix}%

```

Write to the glossary file using makeindex syntax.

```

5368 \gls@glossary{\csname glo@\@gls@label @type\endcsname}{%
5369 \string\glossaryentry{\csname glo@\@gls@label @index\endcsname
5370 \@gls@encapchar\@glo@numfmt}{\@gls@locref}}}%
5371 \fi
5372 }

```

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.)

```

5373 \newcommand*\@gls@getcounterprefix[2]{%
5374 \edef\@gls@thisloc{#1}\edef\@gls@thisHloc{#2}%
5375 \ifx\@gls@thisloc\@gls@thisHloc
5376 \def\@glo@counterprefix{}%
5377 \else
5378 \def\@gls@get@counterprefix##1.#1##2\end@getprefix{%
5379 \def\@glo@tmp{##2}%
5380 \ifx\@glo@tmp\@empty
5381 \def\@glo@counterprefix{}%
5382 \else
5383 \def\@glo@counterprefix{##1}%
5384 \fi
5385 }%
5386 \@gls@get@counterprefix#2.#1\end@getprefix

```

Warn if no prefix can be formed.

```

5387 \ifx\@glo@counterprefix\@empty
5388 \GlossariesWarning{Hyper target ‘#2’ can’t be formed by
5389 prefixing^^Jlocation ‘#1’. You need to modify the
5390 definition of \string\theH\@gls@counter^^Jotherwise you
5391 will get the warning: “name{\@gls@counter.#1}’ has been^^J
5392 referenced but does not exist”}%
5393 \fi
5394 \fi
5395 }

```

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 `[\langle tag \rangle]{\langle list \rangle}`, where `\langle tag \rangle` is a tag such as “see” and `\langle list \rangle` is a list of labels.

```

5396 \newcommand{\do@seeglossary}[2]{%
5397 \def\@gls@xref{#2}%
5398 \@onelevel@sanitize\@gls@xref
5399 \@gls@checkmkidxchars\@gls@xref
5400 \ifglxindy
5401 \gls@glossary{\csname glo@#1@type\endcsname}{%
5402 (indexentry
5403 :tkey (\csname glo@#1@index\endcsname)
5404 :xref (\string"\@gls@xref\string")
5405 :attr \string"see\string"
5406 )
5407 }%
5408 \else
5409 \gls@glossary{\csname glo@#1@type\endcsname}{%
5410 \string@glossaryentry{\csname glo@#1@index\endcsname
5411 \@gls@encapchar glsseeformat\@gls@xref}{Z}}%
5412 \fi
5413 }

```

`\@gls@fixbraces` If no optional argument is specified, list needs to be enclosed in a set of braces.

```

5414 \def\@gls@fixbraces#1#2#3\@nil{%
5415 \ifx#2[\relax
5416 \@gls@fixbraces#1#2#3\@end@fixbraces
5417 \else
5418 \def#1{{#2#3}}%
5419 \fi
5420 }

```

`@@gls@fixbraces`

```

5421 \def\@@gls@fixbraces#1[#2]#3\@end@fixbraces{%
5422 \def#1{[#2]{#3}}%
5423 }

```

```

\glssee \glssee{<label>}{<cross-ref list>}
5424 \DeclareRobustCommand*\glssee}[3][\seename]{%
5425 \do@seeglossary{#2}{#1}{#3}}
5426 \newcommand*\@glssee}[3][\seename]{%
5427 \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.
5428 \DeclareRobustCommand*\glsseeformat}[3][\seename]{%
5429 \emph{#1} \glsseelist{#2}}

\glsseelist \glsseelist{<list>} formats list of entry labels.
5430 \DeclareRobustCommand*\glsseelist}[1]{%
    If there is only one item in the list, set the last separator to do nothing.
5431 \let\@gls@dolast\relax
    Don’t display separator on the first iteration of the loop
5432 \let\@gls@donext\relax
    Iterate through the labels
5433 \@for\@gls@thislabel:=#1\do{%
    Check if on last iteration of loop
5434 \ifx\@xfor@nextelement\@nnil
5435 \@gls@dolast
5436 \else
5437 \@gls@donext
5438 \fi
    Display the entry for this label. (Expanding label as it’s a temporary control sequence that’s
    used elsewhere.)
5439 \expandafter\glsseeitem\expandafter{\@gls@thislabel}%
    Update separators
5440 \let\@gls@dolast\glsseelastsep
5441 \let\@gls@donext\glsseesep
5442 }%
5443 }

\glsseelastsep Separator to use between penultimate and ultimate entries in a cross-referencing list.
5444 \newcommand*\glsseelastsep}{\space\andname\space}

\glsseesep Separator to use between entires in a cross-referencing list.
5445 \newcommand*\glsseesep}{, }

\glsseeitem \glsseeitem{<label>} formats individual entry in a cross-referencing list.
5446 \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.)
5447 \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.

```
5448 \newcommand*{\gls@save@numberlist}[1]{%
5449   \ifglssavenumberlist
5450     \toks@{#1}%
5451     \edef\@do@writeaux@info{%
5452       \noexpand\csgdef{glo@\glscurrententrylabel @numberlist}{\the\toks@}%
5453     }%
5454     \@onelevel@sanitize\@do@writeaux@info
5455     \protected@write\@auxout{}\@do@writeaux@info}%
5456   \fi
5457 }
```

`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.)

```
5458 \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.

```
5459 \ifcsundef{printglossary}{}%
5460 {%
```

If `\printglossary` is already defined, issue a warning and undefine it.

```
5461 \@gls@warnonglossdefined
5462 \undef\printglossary
5463 }
```

`\printglossary` has an optional argument. The default value is to set the glossary type to the main glossary.

```
5464 \newcommand*{\printglossary}[1][type=\glsdefaulttype]{%
5465   \@printglossary{#1}{\@print@glossary}%
5466 }
```

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

```
5467 \newcommand*{\printglossaries}{%
5468   \forallglossaries{\@glo@type}{\printglossary[type=\@glo@type]}%
5469 }
```

printnoidxglossary Provide an alternative to \printglossary that doesn't require an external indexing application. Entries won't be sorted and the location list will be empty.

```
5470 \newcommand*{\printnoidxglossary}[1][type=\glsdefaulttype]{%
5471   \@printglossary{#1}{\@printnoidxglossary}%
5472 }
```

printnoidxglossaries Analogous to \printglossaries

```
5473 \newcommand*{\printnoidxglossaries}{%
5474   \forallglossaries{\@glo@type}{\printnoidxglossary[type=\@glo@type]}%
5475 }
```

printgloss@setsort Initialise to do nothing.

```
5476 \newcommand*{\@printgloss@setsort}{{}}
```

preglossaryhook

```
5477 \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.

```
5478 \newcommand{\@printglossary}[2]{%
```

Set up defaults.

```
5479   \def\@glo@type{\glsdefaulttype}%
5480   \def\glossarytitle{\csname @glotype@\@glo@type @title\endcsname}%

5481   \def\glossarytoctitle{\glossarytitle}%
5482   \let\org@glossarytitle\glossarytitle

5483   \def\@glossarystyle{%
5484     \ifx\@glossary@default@style\relax
5485       \GlossariesWarning{No default glossary style provided \MessageBreak
5486         for the glossary '\@glo@type'. \MessageBreak
5487         Using deprecated fallback. \MessageBreak
5488         To fix this set the style with \MessageBreak
5489         \string\setglossarystyle\space or use the \MessageBreak
5490         style key=value option}%
5491     \fi
5492   }%
5493   \def\gls@dotoc@title{\glssettoc@title{\@glo@type}}%
```

Store current value of \glossaryentrynumbers. (This may be changed via the optional argument)

```
5494   \let\@org@glossaryentrynumbers\glossaryentrynumbers
```

Localise the effects of the optional argument

```
5495 \bgroup
```

Activate or deactivate sort key:

```
5496 \@printgloss@setsort
```

Determine settings specified in the optional argument.

```
5497 \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)

```
5498 \ifx\glossarytitle\org@glossarytitle
```

```
5499 \else
```

```
5500 \expandafter\let\csname @glo@type\@glo@type @title\endcsname
```

```
5501 \glossarytitle
```

```
5502 \fi
```

Allow a high-level user command to indicate the current glossary

```
5503 \let\currentglossary\@glo@type
```

Enable individual number lists to be suppressed.

```
5504 \let\org@glossaryentrynumbers\glossaryentrynumbers
```

```
5505 \let\glsnonextpages\@glsnonextpages
```

Enable individual number list to be activated:

```
5506 \let\glsnextpages\@glsnextpages
```

Enable suppression of description terminators.

```
5507 \let\nopostdesc\@nopostdesc
```

Set up the entry for the TOC

```
5508 \gls@dotoc@title
```

Set the glossary style

```
5509 \@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):

```
5510 \let\gls@org@glossaryentryfield\glossentry
```

```
5511 \let\gls@org@glossarysubentryfield\subglossentry
```

```
5512 \renewcommand{\glossentry}[1]{%
```

```
5513 \xdef\glscurrententrylabel{\glsdetoklabel{##1}}%
```

```
5514 \gls@org@glossaryentryfield{##1}%
```

```
5515 }%
```

```
5516 \renewcommand{\subglossentry}[2]{%
```

```
5517 \xdef\glscurrententrylabel{\glsdetoklabel{##2}}%
```

```
5518 \gls@org@glossarysubentryfield{##1}{##2}%
```

```
5519 }%
```

```
5520 \@gls@preglossaryhook
```

Now do the handler macro that deals with the actual glossary:

```
5521 #2%
```


End the current scope

```
5522 \egroup
```

Reset \glossaryentrynumbers

```
5523 \global\let\glossaryentrynumbers\@org@glossaryentrynumbers
```

Suppress warning about no \printglossary

```
5524 \global\let\warn@noprntglossary\relax
```

```
5525 }
```

@print@glossary Internal workings of \printglossary dealing with reading the external file.

```
5526 \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.)

```
5527 \makeatletter
```

Input the glossary file, if it exists.

```
5528 \@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.

```
5529 \IfFileExists{\jobname.\csname @glo@type@\@glo@type @in\endcsname}%
```

```
5530 {}%
```

```
5531 {\null}%
```

If xindy is being used, need to write the language dependent information to the .aux file for makeglossaries.

```
5532 \ifglxindy
```

```
5533 \ifcsundef{@xdy@\@glo@type @language}%
```

```
5534 {%
```

```
5535 \edef\@do@auxoutstuff{%
```

```
5536 \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.

```
5537 \noexpand\immediate\noexpand\write\@auxout{%
```

```
5538 \string\providecommand\string\@xdylanguage[2]{}}%
```

```
5539 \noexpand\immediate\noexpand\write\@auxout{%
```

```
5540 \string\@xdylanguage{\@glo@type}{\@xdy@main@language}}%
```

```
5541 }%
```

```
5542 }%
```

```
5543 }%
```

```
5544 {%
```

```
5545 \edef\@do@auxoutstuff{%
```

```
5546 \noexpand\AtEndDocument{%
```

```
5547 \noexpand\immediate\noexpand\write\@auxout{%
```

```
5548 \string\providecommand\string\@xdylanguage[2]{}}%
```

```
5549 \noexpand\immediate\noexpand\write\@auxout{%
```

```
5550 \string\@xdylanguage{\@glo@type}{\csname @xdy@\@glo@type
```

```

5551         @language\endcsname}}}%
5552     }%
5553 }%
5554 }%
5555 \do@auxoutstuff
5556 \edef\do@auxoutstuff{%
5557     \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.

```

5558     \noexpand\immediate\noexpand\write\@auxout{%
5559         \string\providecommand\string\@gls@codepage[2]{}}}%
5560     \noexpand\immediate\noexpand\write\@auxout{%
5561         \string\@gls@codepage{\@glo@type}{\@gls@codepage}}}%
5562     }%
5563 }%
5564 \do@auxoutstuff
5565 \fi

```

Activate warning if \makeglossaries hasn't been used.

```

5566 \renewcommand*{\@warn@nomakeglossaries}{%
5567     \GlossariesWarningNoLine{\string\makeglossaries\space
5568         hasn't been used,^^Jthe glossaries will not be updated}%
5569 }%
5570 }

```

The sort macros all have the syntax:

```
\@glo@sortmacro@<order>{<type>}
```

where <order> is the sort order as specified by the sort key and <type> is the glossary type. (The referenced entry list is stored in \@glsref@<type>. The actual sorting is done by \@glo@sortentries{<handler>}{<type>}.

glo@sortentries

```

5571 \newcommand*{\@glo@sortentries}[2]{%
5572     \def\@glo@sortinglist{}%
5573     \def\@glo@sortinghandler{#1}%
5574     \edef\@glo@type{#2}%
5575     \forlistcsloop{\@glo@do@sortentries}{\@glsref@#2}%
5576     \csdef{\@glsref@#2}{}%
5577     \@for\@this@label:=\@glo@sortinglist\do{%

```

Has this entry already been added?

```

5578         \xifinlistcs{\@this@label}{\@glsref@#2}%
5579         {}%
5580         {%
5581             \listcsxadd{\@glsref@#2}{\@this@label}%
5582         }%
5583         \ifcsdef{\@glo@sortingchildren@\@this@label}%

```

```

5584    {%
5585        \@glo@addchildren{#2}{\@this@label}%
5586    }%
5587    {}%
5588 }%
5589 }

```

@glo@addchildren

```
\@glo@addchildren{<type>}{<parent>}
```

```
5590 \newcommand*{\@glo@addchildren}[2]{%
```

Scope to allow nesting.

```

5591 \bgroup
5592 \letcs{\@glo@childlist}{\@glo@sortingchildren@#2}%
5593 \@for\@this@childlabel:=\@glo@childlist\do
5594 {%

```

Check this label hasn't already been added.

```

5595 \xifinlistcs{\@this@childlabel}{\@glsref@#1}%
5596 {}%
5597 {%
5598 \listcsxadd{\@glsref@#1}{\@this@childlabel}%
5599 }%

```

Does this child have children?

```

5600 \ifcsdef{\@glo@sortingchildren@\@this@childlabel}%
5601 {%
5602 \@glo@addchildren{#1}{\@this@childlabel}%
5603 }%
5604 {%
5605 }%
5606 }%
5607 \egroup
5608 }

```

@do@sortentries

```

5609 \newcommand*{\@glo@do@sortentries}[1]{%
5610 \ifglshasparent{#1}%
5611 {%

```

This entry has a parent, so add it to the child list

```

5612 \edef\@glo@parent{\csuse{glo@glstetoklabel{#1}@parent}}%
5613 \ifcsundef{\@glo@sortingchildren@\@glo@parent}%
5614 {%
5615 \csdef{\@glo@sortingchildren@\@glo@parent}{}%
5616 }%
5617 {}%
5618 \expandafter\@glo@sortedinsert
5619 \csname @glo@sortingchildren@\@glo@parent\endcsname{#1}%

```

Has the parent been added?

```
5620 \xifinlistcs{\@glo@parent}{\@glsref{\@glo@type}}%
5621 {%
```

Yes, it has so do nothing.

```
5622 }%
5623 {%
```

No, it hasn't so add it now.

```
5624 \expandafter\@glo@do@sortentries\expandafter{\@glo@parent}%
5625 }%
5626 }%
5627 {%
5628 \@glo@sortedinsert{\@glo@sortinglist}{#1}%
5629 }%
5630 }
```

```
glo@sortedinsert \@glo@sortedinsert{\<list>}{\<entry label>}
```

Insert into list.

```
5631 \newcommand*{\@glo@sortedinsert}[2]{%
5632 \dtl@insertinto{#2}{#1}{\@glo@sortinghandler}%
5633 }%
```

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

```
5634 \newcommand*{\@glo@sorthandler@word}[2]{%
5635 \letcs\@gls@sort@A{glo\glsdetoklabel{#1}@sort}%
5636 \letcs\@gls@sort@B{glo\glsdetoklabel{#2}@sort}%
5637 \edef\@glo@do@compare{%
5638 \noexpand\dtlwordindexcompare{\noexpand\dtl@sortresult}%
5639 {\expandonce\@gls@sort@B}%
5640 {\expandonce\@gls@sort@A}%
5641 }%
5642 \@glo@do@compare
5643 }
```

thandler@letter

```
5644 \newcommand*{\@glo@sorthandler@letter}[2]{%
5645 \letcs\@gls@sort@A{glo\glsdetoklabel{#1}@sort}%
5646 \letcs\@gls@sort@B{glo\glsdetoklabel{#2}@sort}%
5647 \edef\@glo@do@compare{%
5648 \noexpand\dtlletterindexcompare{\noexpand\dtl@sortresult}%
5649 {\expandonce\@gls@sort@B}%
5650 {\expandonce\@gls@sort@A}%
5651 }
```

```

5651 }%
5652 \glo@do@compare
5653 }

```

orthandler@case Case-sensitive sort.

```

5654 \newcommand*{\@glo@sorthandler@case}[2]{%
5655   \letcs\@gls@sort@A{glo\glsdetoklabel{#1}@sort}%
5656   \letcs\@gls@sort@B{glo\glsdetoklabel{#2}@sort}%
5657   \edef\glo@do@compare{%
5658     \noexpand\dtlcompare{\noexpand\dtl@sortresult}%
5659     {\expandonce\@gls@sort@B}%
5660     {\expandonce\@gls@sort@A}%
5661   }%
5662   \glo@do@compare
5663 }

```

thandler@nocase Case-insensitive sort.

```

5664 \newcommand*{\@glo@sorthandler@nocase}[2]{%
5665   \letcs\@gls@sort@A{glo\glsdetoklabel{#1}@sort}%
5666   \letcs\@gls@sort@B{glo\glsdetoklabel{#2}@sort}%
5667   \edef\glo@do@compare{%
5668     \noexpand\dtlcompare{\noexpand\dtl@sortresult}%
5669     {\expandonce\@gls@sort@B}%
5670     {\expandonce\@gls@sort@A}%
5671   }%
5672   \glo@do@compare
5673 }

```

@sortmacro@word Sort macro for ‘word’

```

5674 \newcommand*{\@glo@sortmacro@word}[1]{%
5675   \ifdefstring{\@glo@default@sorttype}{standard}%
5676   {%
5677     \@glo@sortentries{\@glo@sorthandler@word}{#1}%
5678   }%
5679   {%
5680     \PackageError{glossaries}{Conflicting sort options:^^J
5681       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5682       \string\printnoidxglossary[sort=word]}{ }%
5683   }%
5684 }

```

ortmacro@letter Sort macro for ‘letter’

```

5685 \newcommand*{\@glo@sortmacro@letter}[1]{%
5686   \ifdefstring{\@glo@default@sorttype}{standard}%
5687   {%
5688     \@glo@sortentries{\@glo@sorthandler@letter}{#1}%
5689   }%
5690   {%
5691     \PackageError{glossaries}{Conflicting sort options:^^J

```

```

5692     \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5693     \string\printnoidxglossary[sort=letter]}{}%
5694 }%
5695 }

```

tmacro@standard Sort macro for ‘standard’. (Use either ‘word’ or ‘letter’ order.)

```

5696 \newcommand*{\@glo@sortmacro@standard}[1]{%
5697   \ifdefstring{\@glo@default@sorttype}{standard}%
5698   {%
5699     \ifcsdef{\@glo@sorthandler@\glsorder}%
5700     {%
5701       \@glo@sortentries{\csuse{\@glo@sorthandler@\glsorder}}{#1}%
5702     }%
5703     {%
5704       \PackageError{glossaries}{Unknown sort handler ‘\glsorder’}{}%
5705     }%
5706   }%
5707   {%
5708     \PackageError{glossaries}{Conflicting sort options:^^J
5709       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5710       \string\printnoidxglossary[sort=standard]}{}%
5711   }%
5712 }

```

@sortmacro@case Sort macro for ‘case’

```

5713 \newcommand*{\@glo@sortmacro@case}[1]{%
5714   \ifdefstring{\@glo@default@sorttype}{standard}%
5715   {%
5716     \@glo@sortentries{\@glo@sorthandler@case}{#1}%
5717   }%
5718   {%
5719     \PackageError{glossaries}{Conflicting sort options:^^J
5720       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5721       \string\printnoidxglossary[sort=case]}{}%
5722   }%
5723 }

```

ortmacro@nocase Sort macro for ‘nocase’

```

5724 \newcommand*{\@glo@sortmacro@nocase}[1]{%
5725   \ifdefstring{\@glo@default@sorttype}{standard}%
5726   {%
5727     \@glo@sortentries{\@glo@sorthandler@nocase}{#1}%
5728   }%
5729   {%
5730     \PackageError{glossaries}{Conflicting sort options:^^J
5731       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5732       \string\printnoidxglossary[sort=nocase]}{}%
5733   }%
5734 }

```

o@sortmacro@def Sort macro for ‘def’. The order of definition is given in \glo@list@{type}.

```
5735 \newcommand*{\@glo@sortmacro@def}[1]{%
5736   \def\@glo@sortinglist{}%
5737   \for@gl@sentries[#1]{\@gl@thislabel}%
5738   {%
5739     \xifinlistcs{\@gl@thislabel}{\@gl@sref@#1}%
5740     {%
5741       \list@add{\@glo@sortinglist}{\@gl@thislabel}%
5742     }%
5743   }%
5744   }%
5745   }%
5746   \cslet{\@gl@sref@#1}{\@glo@sortinglist}%
5747 }
```

Hasn't been referenced.

ortmacro@def@do This won't include parent entries that haven't been referenced.

```
5748 \newcommand*{\@glo@sortmacro@def@do}[1]{%
5749   \ifinlistcs{#1}{\@gl@sref@\@glo@type}%
5750   {}%
5751   {%
5752     \listcsadd{\@gl@sref@\@glo@type}{#1}%
5753   }%
5754   \ifcsdef{\@glo@sortingchildren@#1}%
5755   {%
5756     \@glo@addchildren{\@glo@type}{#1}%
5757   }%
5758   {}%
5759 }
```

o@sortmacro@use Sort macro for ‘use’. (No sorting is required, as the entries are already in order of use, so do nothing.)

```
5760 \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.

```
5761 \newcommand*{\@print@noidx@glossary}{%
5762   \ifcsdef{\@gl@sref@\@glo@type}%
5763   {%
```

Sort the entries:

```
5764   \ifcsdef{\@glo@sortmacro@\@glo@sorttype}%
5765   {%
5766     \csuse{\@glo@sortmacro@\@glo@sorttype}{\@glo@type}%
5767   }%
```

```

5768     {%
5769     \PackageError{glossaries}{Unknown sort handler ‘\@glo@sorttype’}{}%
5770     }%

```

Do the glossary heading and preamble

```

5771     \glossarysection[\glossarytoctitle]{\glossarytitle}%
5772     \glossarypreamble
5773     \begin{theglossary}%
5774     \glossaryheader
5775     \glsresetentrylist
5776     \def\@gls@currentlettergroup{}%

```

Iterate through the entries.

```

5777     \forlistcsloop{\@gls@noidx@do}{\@glsref@{\@glo@type}%

```

Finally end the glossary and do the postamble:

```

5778     \end{theglossary}%
5779     \glossarypostamble
5780     }%
5781     {%
5782     \@gls@noref@warn{\@glo@type}%
5783     }%
5784 }

```

\glo@grabfirst

```

5785 \def\glo@grabfirst#1#2\@nil{%
5786   \def\@gls@firsttok{#1}%
5787   \ifdefempty\@gls@firsttok
5788   {%
5789     \def\@glo@thislettergrp{0}%
5790   }%
5791   {%

```

Sanitize it:

```

5792     \@onelevel@sanitize\@gls@firsttok

```

Fetch the first letter:

```

5793     \expandafter\@glo@grabfirst\@gls@firsttok{}{}\@nil
5794     }%
5795 }

```

\@glo@grabfirst

```

5796 \def\@glo@grabfirst#1#2\@nil{%
5797   \ifdefempty\@glo@thislettergrp
5798   {%
5799     \def\@glo@thislettergrp{glssymbols}%
5800   }%
5801   {%
5802     \count@=\uccode‘#1\relax
5803     \ifnum\count@=0\relax
5804     \def\@glo@thislettergrp{glssymbols}%

```



```

5805 \else
5806 \ifdefstring\@glo@sorttype{case}%
5807 {%
5808 \count@=#1\relax
5809 }%
5810 {%
5811 }%
5812 \edef\@glo@thislettergrp{\the\count@}%
5813 \fi
5814 }%
5815 }

```

\@gls@noidx@do Handler for list iteration used by \@print@noidx@glossary. The argument is the entry label. This only allows one sublevel.

```

5816 \newcommand{\@gls@noidx@do}[1]{%

```

Get this entry's location list

```

5817 \global\letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%

```

Does this entry have a parent?

```

5818 \ifglshasparent{#1}%

```

```

5819 {%

```

Has a parent.

```

5820 \gls@level=\csuse{glo@\glsdetoklabel{#1}@level}\relax

```

```

5821 \ifdefvoid{\@gls@loclist}

```

```

5822 {%

```

```

5823 \subglossentry{\gls@level}{#1}{}%

```

```

5824 }%

```

```

5825 {%

```

```

5826 \subglossentry{\gls@level}{#1}%

```

```

5827 {%

```

```

5828 \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%

```

```

5829 }%

```

```

5830 }%

```

```

5831 }%

```

```

5832 {%

```

Doesn't have a parent Get this entry's sort key

```

5833 \letcs{\@gls@sort}{glo@\glsdetoklabel{#1}@sort}%

```

Fetch the first letter:

```

5834 \expandafter\glo@grabfirst\@gls@sort{}{}\@nil

```

```

5835 \ifdefequal{\@glo@thislettergrp}{\@gls@currentlettergroup}%

```

```

5836 {}%

```

```

5837 {%

```

Do the group header:

```

5838 \ifdefempty{\@gls@currentlettergroup}{\@gls@groupskip}%

```

```

5839 \gls@groupheading{\@glo@thislettergrp}%

```

```

5840 }%

```

```

5841 \let\@gls@currentlettergroup\@glo@thislettergrp

```

Do this entry:

```

5842 \ifdefvoid{\@gls@loclist}
5843 {%
5844 \glossentry{#1}{}%
5845 }%
5846 {%
5847 \glossentry{#1}%
5848 {%
5849 \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
5850 }%
5851 }%
5852 }%
5853 }

```

\glsnoidxloclist `\glsnoidxloclist{<list cs>}`

Display location list.

```

5854 \newcommand*{\glsnoidxloclist}[1]{%
5855 \def\@gls@noidxloclist@sep{}%
5856 \def\@gls@noidxloclist@prev{}%
5857 \forlistloop{\glsnoidxloclisthandler}{#1}%
5858 }

```

xloclisthandler Handler for location list iterator.

```

5859 \newcommand*{\glsnoidxloclisthandler}[1]{%
5860 \ifdefstring{\@gls@noidxloclist@prev}{#1}%
5861 {%
5862 }%
5863 {%
5864 \@gls@noidxloclist@sep
5865 #1%
5866 \def\@gls@noidxloclist@sep{\delimN}%
5867 \def\@gls@noidxloclist@prev{#1}%
5868 }%
5869 }

```

yloclisthandler Handler for location list iterator when used with \glsdisplaynumberlist.

```

5870 \newcommand*{\glsnoidxdisplayloclisthandler}[1]{%
5871 \ifdefstring{\@gls@noidxloclist@prev}{#1}%
5872 {%

```

Same as previous location so skip.

```

5873 }%
5874 {%
5875 \@gls@noidxloclist@sep
5876 \@gls@noidxloclist@prev

```

```

5877 \def\@gls@noidxloclist@prev{#1}%
5878 }%
5879 }

```

`\glsnoidxdisplayloc` `\glsnoidxdisplayloc{<prefix>}{<counter>}{<format>}{<location>}`

Display a location in the location list.

```

5880 \newcommand*\glsnoidxdisplayloc[4]{%
5881 \setentrycounter[#1]{#2}%
5882 \csuse{#3}{#4}%
5883 }

```

`\@gls@reference` `\@gls@reference{<type>}{<label>}{<loc>}`

Identifies that a reference has been used (for use in the aux file). All entries must be defined in the preamble.

```

5884 \newcommand*\@gls@reference}[3]{%

```

Add to label list

```

5885 \glsdoifexistsorwarn{#2}%
5886 {%
5887 \ifcsundef{@glsref@#1}{\csgdef{@glsref@#1}{}}{}%
5888 \ifinlistcs{#2}{@glsref@#1}%
5889 {}%
5890 {\listcsgadd{@glsref@#1}{#2}}%

```

Add to location list

```

5891 \ifcsundef{glo@glstdetoklabel{#2}@loclist}%
5892 {\csgdef{glo@glstdetoklabel{#2}@loclist}{}}%
5893 {}%
5894 \listcsgadd{glo@glstdetoklabel{#2}@loclist}{#3}%
5895 }%
5896 }

```

The keys that can be used in the optional argument to `\printglossary` or `\printnoidxglossary` are as follows: The type key sets the glossary type.

```

5897 \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`.

```

5898 \define@key{printgloss}{title}{%
5899 \def\glossarytitle{#1}%
5900 \let\gls@dotocitle\relax
5901 }

```

The toctitle sets the text used for the relevant entry in the table of contents.

```

5902 \define@key{printgloss}{toctitle}{%

```

```

5903 \def\glossarytoctitle{#1}%
5904 \let\gls@dotoc\title\relax
5905 }

```

The style key sets the glossary style (but only for the given glossary).

```

5906 \define@key{printgloss}{style}{%
5907   \ifcsundef{@glsstyle@#1}%
5908   {%
5909     \PackageError{glossaries}%
5910     {Glossary style ‘#1’ undefined}{}%
5911   }%
5912   {%
5913     \def\@glossarystyle{\setglossentrycompatibility
5914       \csname @glsstyle@#1\endcsname}%
5915   }%
5916 }

```

The numberedsection key determines if this glossary should be in a numbered section.

```

5917 \define@choicekey{printgloss}{numberedsection}[\val\nr]{%
5918 false,nolabel,autolabel,nameref}[nolabel]{%
5919   \ifcase\nr\relax
5920     \renewcommand*{\@glossarysecstar}{*}%
5921     \renewcommand*{\@glossaryseclabel}{}%
5922   \or
5923     \renewcommand*{\@glossarysecstar}{}%
5924     \renewcommand*{\@glossaryseclabel}{}%
5925   \or
5926     \renewcommand*{\@glossarysecstar}{}%
5927     \renewcommand*{\@glossaryseclabel}{\label{\glsautoprefix\@glo@type}}%
5928   \or
5929     \renewcommand*{\@glossarysecstar}{*}%
5930     \renewcommand*{\@glossaryseclabel}{%
5931       \protected@edef\@currentlabelname{\glossarytoctitle}%
5932       \label{\glsautoprefix\@glo@type}}%
5933   \fi
5934 }

```

The nogroupskip key determines whether or not there should be a vertical gap between glossary groups.

```

5935 \define@choicekey{printgloss}{nogroupskip}{true,false}[true]{%
5936   \csuse{glsnogroupskip#1}%
5937 }

```

The nopostdot key has the same effect as the package option of the same name.

```

5938 \define@choicekey{printgloss}{nopostdot}{true,false}[true]{%
5939   \csuse{glsnopostdot#1}%
5940 }

```

The entrycounter key is the same as the package option but localised to the current glossary.

```

5941 \define@choicekey{printgloss}{entrycounter}{true,false}[true]{%
5942   \csuse{glsentrycounter#1}%

```

```

5943 \ifglentrycounter
5944   \ifx\@gls@counterwithin\@empty
5945     \newcounter{glossaryentry}%
5946   \else
5947     \newcounter{glossaryentry}[\@gls@counterwithin]%
5948   \fi
5949   \def\theHglossaryentry{\currentglossary.\theglossaryentry}%
5950   \renewcommand*\@glsresetentrycounter{%
5951     \setcounter{glossaryentry}{0}%
5952   }%
5953   \renewcommand*\@glsstepentry{[1]{%
5954     \refstepcounter{glossaryentry}%
5955     \label{glentry-\@glsdetoklabel{##1}}%
5956   }%
5957   \renewcommand*\@glentrycounterlabel{\theglossaryentry.\space}%
5958   \renewcommand*\@glentryitem{[1]{%
5959     \@glsstepentry{##1}\glentrycounterlabel
5960   }%
5961 \else
5962   \renewcommand*\@glsresetentrycounter{%
5963   \renewcommand*\@glsstepentry{[1]{}%
5964   \renewcommand*\@glentrycounterlabel{%
5965   \renewcommand*\@glentryitem{[1]{\@glsresetsubentrycounter}
5966 \fi
5967 }

```

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.

```

5968 \define@choicekey{printgloss}{subentrycounter}{true,false}[true]{%
5969   \csuse{glssubentrycounter#1}%
5970   \ifglssubentrycounter
5971     \ifundef\c@glossarysubentry
5972     {%
5973       \ifglentrycounter
5974         \newcounter{glossarysubentry}[glossaryentry]%
5975       \else
5976         \newcounter{glossarysubentry}
5977       \fi
5978     }{}%
5979     \renewcommand*\@glsstepsubentry{[1]{%
5980       \edef\currentglssubentry{\@glsdetoklabel{##1}}%
5981       \refstepcounter{glossarysubentry}%
5982       \label{glentry-\currentglssubentry}%
5983     }%
5984     \renewcommand*\@glsresetsubentrycounter{%
5985       \setcounter{glossarysubentry}{0}%
5986     }%
5987     \renewcommand*\@glssubentryitem{[1]{%
5988       \@glsstepsubentry{##1}\glssubentrycounterlabel

```

```

5989 }%
5990 \renewcommand*{\glssubentrycounterlabel}{\theglossarysubentry}\space}%
5991 \def\theHglossarysubentry{\currentglssubentry.\theglossarysubentry}
5992 \else
5993 \renewcommand*{\glssubentryitem}[1]{}%
5994 \renewcommand*{\glstepsubentry}[1]{}%
5995 \renewcommand*{\glsresetsubentrycounter}{}%
5996 \renewcommand*{\glssubentrycounterlabel}{}%
5997 \fi
5998 }

```

The nonnumberlist key determines if this glossary should have a number list.

```

5999 \define@boolkey{printgloss}[gls]{nonnumberlist}[true]{%
6000 \ifglslnonnumberlist
6001 \def\glossaryentrynumbers##1{%
6002 \else
6003 \def\glossaryentrynumbers##1{##1}%
6004 \fi}

```

The sort key sets the glossary sort handler (\printnoidxglossary only).

```

6005 \define@key{printgloss}{sort}{\@glo@assign@sortkey{#1}}

```

@assign@sortkey Issue error if used with \printglossary

```

6006 \newcommand*{\@glo@no@assign@sortkey}[1]{%
6007 \PackageError{glossaries}{'sort' key not permitted with
6008 \string\printglossary}%
6009 {The 'sort' key may only be used with \string\printnoidxglossary}%
6010 }

```

@assign@sortkey For use with \printnoidxglossary

```

6011 \newcommand*{\@glo@assign@sortkey}[1]{%
6012 \def\@glo@sorttype{#1}%
6013 }

```

@glslnonextpages Suppresses the next number list only. Global assignments required as it may not occur in the same level of grouping as the next numberlist. (For example, if \glslnonextpages is place in the entry's description and 3 column tabular style glossary is used.) \org@glossaryentrynumbers needs to be set at the start of each glossary, in the event that \glossaryentrynumber is re-defined.

```

6014 \newcommand*{\@glslnonextpages}{%
6015 \gdef\glossaryentrynumbers##1{%
6016 \glsresetentrylist
6017 }%
6018 }

```

\@glslnextpages 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 \glslnextpages 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.

```
6019 \newcommand*{\@glsnextpages}{%
6020   \gdef\glossaryentrynumbers##1{%
6021     ##1\glsresetentrylist}}
```

`\glsresetentrylist` Resets `\glossaryentrynumbers`

```
6022 \newcommand*{\glsresetentrylist}{%
6023   \global\let\glossaryentrynumbers\org@glossaryentrynumbers}
```

`\glsnonextpages` Outside of `\printglossary` this does nothing.

```
6024 \newcommand*{\glsnonextpages}{}%
```

`\glsnextpages` Outside of `\printglossary` this does nothing.

```
6025 \newcommand*{\glsnextpages}{}%
```

`\glossaryentry` If the `entrycounter` package option has been used, define a counter to number each level 0 entry.

```
6026 \ifglentrycounter
6027   \ifx\@gls@counterwithin\@empty
6028     \newcounter{glossaryentry}
6029   \else
6030     \newcounter{glossaryentry}[\@gls@counterwithin]
6031   \fi
6032   \def\theHglossaryentry{\currentglossary.\theglossaryentry}
6033 \fi%
```

`\glossarysubentry` If the `subentrycounter` package option has been used, define a counter to number each level 1 entry.

```
6034 \ifglssubentrycounter
6035   \ifglentrycounter
6036     \newcounter{glossarysubentry}[glossaryentry]
6037   \else
6038     \newcounter{glossarysubentry}
6039   \fi
6040   \def\theHglossarysubentry{\currentglssubentry.\theglossarysubentry}
6041 \fi%
```

`\glsresetentrylist` Resets the `\glossarysubentry` counter.

```
6042 \ifglssubentrycounter
6043   \newcommand*{\glsresetentrylist}{%
6044     \setcounter{glossarysubentry}{0}%
6045   }
6046 \else
6047   \newcommand*{\glsresetentrylist}{}%
6048 \fi%
```

subentrycounter Resets the glossaryentry counter.

```
6049 \ifglentrycounter
6050   \newcommand*{\glsresetentrycounter}{%
6051     \setcounter{glossaryentry}{0}%
6052   }
6053 \else
6054   \newcommand*{\glsresetentrycounter}{}
6055 \fi
```

\glsstepentry Advance the glossaryentry counter if in use. The argument is the label associated with the entry.

```
6056 \ifglentrycounter
6057   \newcommand*{\glsstepentry}[1]{%
6058     \refstepcounter{glossaryentry}%
6059     \label{glentry-\glsdetoklabel{#1}}%
6060   }
6061 \else
6062   \newcommand*{\glsstepentry}[1]{}
6063 \fi
```

glsstepsubentry Advance the glossarysubentry counter if in use. The argument is the label associated with the subentry.

```
6064 \ifglssubentrycounter
6065   \newcommand*{\glsstepsubentry}[1]{%
6066     \edef\currentglssubentry{\glsdetoklabel{#1}}%
6067     \refstepcounter{glossarysubentry}%
6068     \label{glentry-\currentglssubentry}%
6069   }
6070 \else
6071   \newcommand*{\glsstepsubentry}[1]{}
6072 \fi
```

\glsrefentry Reference the entry or sub-entry counter if in use, otherwise just do \gls.

```
6073 \ifglentrycounter
6074   \newcommand*{\glsrefentry}[1]{\ref{glentry-\glsdetoklabel{#1}}}
6075 \else
6076   \ifglssubentrycounter
6077     \newcommand*{\glsrefentry}[1]{\ref{glentry-\glsdetoklabel{#1}}}
6078   \else
6079     \newcommand*{\glsrefentry}[1]{\gls{#1}}
6080   \fi
6081 \fi
```

trycounterlabel Defines how to display the glossaryentry counter.

```
6082 \ifglentrycounter
6083   \newcommand*{\glsentrycounterlabel}{\theglossaryentry.\space}
6084 \else
6085   \newcommand*{\glsentrycounterlabel}{}
6086 \fi
```


trycounterlabel Defines how to display the glossarysubentry counter.

```
6087 \ifglssubentrycounter
6088   \newcommand*{\glssubentrycounterlabel}{\theglossarysubentry}\space}
6089 \else
6090   \newcommand*{\glssubentrycounterlabel}{}
6091 \fi
```

\glstentryitem Step and display glossaryentry counter, if appropriate.

```
6092 \ifglstentrycounter
6093   \newcommand*{\glstentryitem}[1]{%
6094     \glststepentry{#1}\glstentrycounterlabel
6095   }
6096 \else
6097   \newcommand*{\glstentryitem}[1]{\glstresetsubentrycounter}
6098 \fi
```

glssubentryitem Step and display glossarysubentry counter, if appropriate.

```
6099 \ifglssubentrycounter
6100   \newcommand*{\glssubentryitem}[1]{%
6101     \glststepsubentry{#1}\glssubentrycounterlabel
6102   }
6103 \else
6104   \newcommand*{\glssubentryitem}[1]{}
6105 \fi
```

theglossary If the theglossary environment has already been defined, a warning will be issued. This environment should be redefined by glossary styles.

```
6106 \ifcsundef{theglossary}%
6107 {%
6108   \newenvironment{theglossary}{}{}}%
6109 }%
6110 {%
6111   \@gls@warnontheglossdefined
6112   \renewenvironment{theglossary}{}{}}%
6113 }
```

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

```
6114 \newcommand*{\glossaryheader}{}%
```

\glstarget \glstarget{<label>}{<name>}

Provide user interface to \@glstarget to make it easier to modify the glossary style in the document.

```
6115 \newcommand*{\glstarget}[2]{\@glstarget{\glolinkprefix#1}{#2}}
```

As from version 3.08, glossary information is now written to the external files using `\glossentry` and `\subglossentry` instead of `\glossaryentryfield` and `\glossarysubentryfield`. The default definition provides backward compatibility for glossary styles that use the old forms.

```
\compatibleglossentry \glossentry{<label>}{<page-list>}
```

```
6116 \providecommand*{\compatibleglossentry}[2]{%
6117   \toks@{#2}%
6118   \protected@edef\@do@glossentry{\noexpand\glossaryentryfield{#1}%
6119     {\noexpand\glsnamefont
6120       {\expandafter\expandonce\csname glo@#1@name\endcsname}}}%
6121     {\expandafter\expandonce\csname glo@#1@desc\endcsname}%
6122     {\expandafter\expandonce\csname glo@#1@symbol\endcsname}%
6123     {\the\toks@}}%
6124   }%
6125   \@do@glossentry
6126 }
```

`\glossentryname`

```
6127 \newcommand*{\glossentryname}[1]{%
6128   \glsdoifexistsorwarn{#1}%
6129   {%
6130     \letcs{\glo@name}{\glo@\glsdetoklabel{#1}@name}%
6131     \expandafter\glsnamefont\expandafter{\glo@name}%
6132   }%
6133 }
```

`\Glossentryname`

```
6134 \newcommand*{\Glossentryname}[1]{%
6135   \glsdoifexistsorwarn{#1}%
6136   {%
6137     \glsnamefont{\Glsentryname{#1}}%
6138   }%
6139 }
```

`\glossentrydesc`

```
6140 \newcommand*{\glossentrydesc}[1]{%
6141   \glsdoifexistsorwarn{#1}%
6142   {%
6143     \glsentrydesc{#1}%
6144   }%
6145 }
```

`\Glossentrydesc`

```

6146 \newcommand*{\Glossentrydesc}[1]{%
6147   \glsdoifexistsorwarn{#1}%
6148   {%
6149     \Glsentrydesc{#1}%
6150   }%
6151 }

```

lossentrysymbol

```

6152 \newcommand*{\glossentrysymbol}[1]{%
6153   \glsdoifexistsorwarn{#1}%
6154   {%
6155     \glsentrysymbol{#1}%
6156   }%
6157 }

```

lossentrysymbol

```

6158 \newcommand*{\Glossentrysymbol}[1]{%
6159   \glsdoifexistsorwarn{#1}%
6160   {%
6161     \Glsentrysymbol{#1}%
6162   }%
6163 }

```

blesubglossentry `\subglossentry{<level>}{<label>}{<page-list>}`

```

6164 \providecommand*{\compatiblesubglossentry}[3]{%
6165   \toks@{#3}%
6166   \protected@edef\@do@subglossentry{\noexpand\glossarysubentryfield{\number#1}%
6167     {#2}%
6168     {\noexpand\glsnamefont
6169       {\expandafter\expandonce\csname glo@#2@name\endcsname}}%
6170     {\expandafter\expandonce\csname glo@#2@desc\endcsname}%
6171     {\expandafter\expandonce\csname glo@#2@symbol\endcsname}%
6172     {\the\toks@}%
6173   }%
6174   \@do@subglossentry
6175 }

```

rycompatibility

```

6176 \newcommand*{\setglossentrycompatibility}{%
6177   \let\glossentry\compatibleglossentry
6178   \let\subglossentry\compatiblesubglossentry
6179 }
6180 \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.

```
6181 \newcommand{\glossaryentryfield}[5]{%
6182   \GlossariesWarning
6183   {Deprecated use of \string\glossaryentryfield.^^J
6184    I recommend you change to \string\glossentry.^^J
6185    If you've just upgraded, try removing your gls auxiliary
6186    files^^J and recompile}%
6187   \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.)

```
6188 \newcommand*{\glossarysubentryfield}[6]{%
6189   \GlossariesWarning
6190   {Deprecated use of \string\glossarysubentryfield.^^J
6191    I recommend you change to \string\subglossentry.^^J
6192    If you've just upgraded, try removing your gls auxiliary
6193    files^^J and recompile}%
6194   \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

```
6195 \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

```
6196 \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 `glsymbols` produces `\glsymbolsgroupname`, the group labelled `glsnumbers` produces `\glsnumbersgroupname` and all the other groups simply produce their label. As mentioned above, the group labels are: `glsymbols`, `glsnumbers`, `A`, ..., `Z`. If you want to redefine the group titles, you will need to redefine this command. Languages other than English may produce labels that are non-expandable, so we need to check for that otherwise it will create a “missing `\endcsname` inserted” error.

`\glsgetgrouptitle`

```
6197 \newcommand*{\glsgetgrouptitle}[1]{%
6198   \@gls@getgrouptitle{#1}{\@gls@grptitle}%
6199   \@gls@grptitle
6200 }
```

`\@gls@getgrouptitle` Gets the group title specified by the label (first argument) and stores in the second argument, which must be a control sequence.

```
6201 \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.
6202   \dtl@ifsingle{#1}%
6203   {%
6204     \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
6205   }%
6206   {%
6207     \ifboolexpr{test{\ifstrequal{#1}{glsymbols}}
6208                 or test{\ifstrequal{#1}{glsnumbers}}}%
6209     {%
6210       \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
6211     }%
6212     {%
6213       \def#2{#1}%
6214     }%
6215   }%
6216 }
```

`\othergrouptitle` Version for the no-indexing app option:

```

6217 \newcommand*{\@gls@noidx@getgrouptitle}[2]{%
6218   \DTLifint{#1}%
6219   {\edef#2{\char#1\relax}}%
6220   {%
6221     \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
6222   }%
6223 }

```

`\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`

```

6224 \newcommand*{\glsgetgrouplabel}[1]{%
6225 \ifthenelse{\equal{#1}{\glssymbolsgroupname}}{\glssymbols}{%
6226 \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`

```

6227 \newcommand*{\setentrycounter}[2][ ]{%
6228   \def\@glo@counterprefix{#1}%
6229   \ifx\@glo@counterprefix\empty
6230     \def\@glo@counterprefix{.}%
6231   \else
6232     \def\@glo@counterprefix{.#1.}%
6233   \fi
6234   \def\glsentrycounter{#2}%
6235 }

```

The current glossary style can be set using `\setglossarystyle{<style>}`.

`\setglossarystyle`

```

6236 \newcommand*{\setglossarystyle}[1]{%
6237   \ifcsundef{@glsstyle@#1}%
6238   {%
6239     \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
6240   }%
6241   {%
6242     \csname @glsstyle@#1\endcsname
6243   }%

```

Set the default style if it's not already set.

```

6244 \ifx\@glossary@default@style\relax
6245   \protected@edef\@glossary@default@style{#1}%

```

```

6246 \fi
6247 }

```

`\glossarystyle`

```

6248 \newcommand*{\glossarystyle}[1]{%
6249   \ifcsundef{@glsstyle@#1}%
6250   {%
6251     \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
6252   }%
6253   {%
6254     \GlossariesWarning
6255     {Deprecated command \string\glossarystyle.^^J
6256      I recommend you switch to \string\setglossarystyle\space unless
6257      you want to maintain backward compatibility}%
6258     \setglossentrycompatibility
6259     \csname @glsstyle@#1\endcsname

6260   \ifcsdef{@glscompstyle@#1}%
6261     {\setglossentrycompatibility\csuse{@glscompstyle@#1}}%
6262   }%
6263 }%

```

Set the default style if it isn't already set so that `\printglossary` can warn if the fallback style is in use.

```

6264 \ifx\@glossary@default@style\relax
6265   \protected@edef\@glossary@default@style{#1}%
6266 \fi
6267 }

```

`\ewglossarystyle` 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.

```

6268 \newcommand{\newglossarystyle}[2]{%
6269   \ifcsundef{@glsstyle@#1}%
6270   {%
6271     \expandafter\def\csname @glsstyle@#1\endcsname{#2}%
6272   }%
6273   {%
6274     \PackageError{glossaries}{Glossary style ‘#1’ is already defined}{}%
6275   }%
6276 }

```

`\ewglossarystyle` Code for this macro supplied by Marco Daniel.

```

6277 \newcommand{\renewglossarystyle}[2]{%
6278   \ifcsundef{@glsstyle@#1}%
6279   {%
6280     \PackageError{glossaries}{Glossary style ‘#1’ isn’t already defined}{}%
6281   }%
6282   {%
6283     \csdef{@glsstyle@#1}{#2}%
6284   }%
6285 }

```

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`

```

6286 \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`

```

6287 \ifcsundef{hyperlink}%
6288 {%
6289   \def\glshypernumber#1{#1}%
6290 }%
6291 {%
6292   \def\glshypernumber#1{\@glshypernumber#1\nohyperpage{}}\@nil}
6293 }

```

`@glshypernumber` This code was provided by Heiko Oberdiek to allow material to be attached to the location.

```

6294 \def\@glshypernumber#1\nohyperpage#2#3\@nil{%
6295   \ifx\#1\%
6296   \else
6297     \@delimR#1\delimR\delimR\%
6298   \fi
6299   \ifx\#2\%

```



```

6300 \else
6301   #2%
6302 \fi
6303 \ifx\\#3\\%
6304 \else
6305   \@glshypernumber#3\@nil
6306 \fi
6307 }

```

`\@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`

```

6308 \def\@delimR#1\delimR #2\delimR #3\\{%
6309 \ifx\\#2\\%
6310   \@delimN{#1}%
6311 \else
6312   \@gls@numberlink{#1}\delimR\@gls@numberlink{#2}%
6313 \fi}

```

`\@delimN` displays a list of individual numbers, instead of a range:

`\@delimN`

```

6314 \def\@delimN#1{\@delimN#1\delimN \delimN\\}
6315 \def\@delimN#1\delimN #2\delimN#3\\{%
6316 \ifx\\#3\\%
6317   \@gls@numberlink{#1}%
6318 \else
6319   \@gls@numberlink{#1}\delimN\@gls@numberlink{#2}%
6320 \fi
6321 }

```

The following code is modified from `hyperref's \HyInd@pagelink` where the name of the counter being used is given by `\@gls@counter`.

```

6322 \def\@gls@numberlink#1{%
6323 \begingroup
6324 \toks@={}%
6325 \@gls@removespaces#1 \@nil
6326 \endgroup}

6327 \def\@gls@removespaces#1 #2\@nil{%
6328 \toks@=\expandafter{\the\toks@#1}%
6329 \ifx\\#2\\%
6330   \edef\x{\the\toks@}%
6331   \ifx\x\empty
6332   \else

6333     \hyperlink{\glstrycounter\@gls@counterprefix\the\toks@}%
6334     {\the\toks@}%
6335   \fi

```

```

6336 \else
6337   \@gls@ReturnAfterFi{%
6338     \@gls@removespaces#2\@nil
6339   }%
6340 \fi
6341 }
6342 \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`

```
6343 \newcommand*\hyperrm[1]{\textrm{\glshypernumber{#1}}}
```

`\hypersf`

```
6344 \newcommand*\hypersf[1]{\textsf{\glshypernumber{#1}}}
```

`\hypertt`

```
6345 \newcommand*\hypertt[1]{\texttt{\glshypernumber{#1}}}
```

`\hyperbf`

```
6346 \newcommand*\hyperbf[1]{\textbf{\glshypernumber{#1}}}
```

`\hypermd`

```
6347 \newcommand*\hypermd[1]{\textmd{\glshypernumber{#1}}}
```

`\hyperit`

```
6348 \newcommand*\hyperit[1]{\textit{\glshypernumber{#1}}}
```

`\hypersl`

```
6349 \newcommand*\hypersl[1]{\textsl{\glshypernumber{#1}}}
```

`\hyperup`

```
6350 \newcommand*\hyperup[1]{\textup{\glshypernumber{#1}}}
```

`\hypersc`

```
6351 \newcommand*\hypersc[1]{\textsc{\glshypernumber{#1}}}
```

`\hyperemph`

```
6352 \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.

```
6353 \newcommand{\oldacronym}[4][\gls@label]{%
6354   \def\gls@label{#2}%
6355   \newacronym[#4]{#1}{#2}{#3}%
6356   \ifcsundef{xspace}%
6357   {%
6358     \expandafter\edef\csname#1\endcsname{%
6359       \noexpand@ifstar{\noexpand\Gls{#1}}{\noexpand\gls{#1}}}%
6360   }%
6361 }%
6362 {%
6363   \expandafter\edef\csname#1\endcsname{%
6364     \noexpand@ifstar{\noexpand\Gls{#1}\noexpand\xspace}%
6365     \noexpand\gls{#1}\noexpand\xspace}%
6366   }%
6367 }%
6368 }
```

`\newacronym[⟨key-val list⟩]{⟨label⟩}{⟨abbrv⟩}{⟨long⟩}`

This is a quick way of defining acronyms, using `\newglossaryentry` with the appropriate values. It sets the glossary type to `\acronymtype` which will be `acronym` if the package option `acronym` has been used, otherwise it will be the default glossary. Since `\newacronym` merely calls `\newglossaryentry`, the acronym is treated like any other glossary entry.

If you prefer a different format, you can redefine `\newacronym` as required. The optional argument can be used to override any of the settings.

This is just a stub. It's redefined by commands like `\SetDefaultAcronymStyle`.

`\newacronym`

```
6369 \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.

```
6370 \newcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}
```

If `garamondx` has been loaded, need to use `\textulc` instead of `\textup`.

`\glstextup`

```
6371 \newrobustcmd*{\glstextup}[1]{\ifdef\textulc{\textulc{#1}}{\textup{#1}}}
```

The following are defined for compatibility with version 2.07 and earlier.

`\glsshortkey`

```
6372 \newcommand*{\glsshortkey}{short}
```

`\sshortpluralkey`

```
6373 \newcommand*{\glsshortpluralkey}{shortplural}
```

`\glslongkey`

```
6374 \newcommand*{\glslongkey}{long}
```

`\lslongpluralkey`

```
6375 \newcommand*{\glslongpluralkey}{longplural}
```

`\acrfull` Full form of the acronym.

```
6376 \newrobustcmd*{\acrfull}{\@gls@hyp@opt\@ns@acrfull}
```

```
6377 \newcommand*{\ns@acrfull}[2][]{\%
```

```
6378 \new@ifnextchar[\@acrfull{#1}{#2}}%
```

```
6379 \@acrfull{#1}{#2}[]}%
```

```
6380 }
```

`\@acrfull` Low-level macro:

```
6381 \def\@acrfull#1#2[#3]{\%
```

Make it easier for acronym styles to change this:

```
6382 \acrfullfmt{#1}{#2}{#3}%
```

```
6383 }
```

Using `\acrlinkfullformat` and `\acrfullformat` is now deprecated as it can cause complications with the first letter upper case variants, but the package needs to provide backward compatibility support.

```

\acrfullfmt  No case change full format.
6384 \newcommand*{\acrfullfmt}[3]{%
6385   \acrlinkfullformat{\@acrlong}{\@acrshort}{#1}{#2}{#3}%
6386 }

\acrlinkfullformat  Format for full links like \acrfull. Syntax: \acrlinkfullformat{<long cs>}{<short cs>}{<options>}{<label>}{<insert>}
6387 \newcommand{\acrlinkfullformat}[5]{%
6388   \acrfullformat{#1}{#3}{#4}[#5]{#2}{#3}{#4}[]}%
6389 }

\acrfullformat  Default full form is <long> (<short>).
6390 \newcommand{\acrfullformat}[2]{#1\glsspace(#2)}

\glsspace  Robust space to ensure it's written to the .glsdefs file.
6391 \newrobustcmd{\glsspace}{\space}

        Default format for full acronym

\Acrfull
6392 \newrobustcmd*{\Acrfull}{\@gls@hyp@opt\ns@Acrfull}

6393 \newcommand*{\ns@Acrfull}[2][]{%
6394   \new@ifnextchar[{\@Acrfull{#1}{#2}}%
6395     {\@Acrfull{#1}{#2}[]}%
6396 }

        Low-level macro:
6397 \def\@Acrfull#1#2[#3]{%

        Make it easier for acronym styles to change this:
6398   \Acrfullfmt{#1}{#2}{#3}%
6399 }

\Acrfullfmt  First letter upper case full format.
6400 \newcommand*{\Acrfullfmt}[3]{%
6401   \acrlinkfullformat{\@acrlong}{\@acrshort}{#1}{#2}{#3}%
6402 }

\ACRfull
6403 \newrobustcmd*{\ACRfull}{\@gls@hyp@opt\ns@ACRfull}

6404 \newcommand*{\ns@ACRfull}[2][]{%
6405   \new@ifnextchar[{\@ACRfull{#1}{#2}}%
6406     {\@ACRfull{#1}{#2}[]}%
6407 }

        Low-level macro:
6408 \def\@ACRfull#1#2[#3]{%

```

Make it easier for acronym styles to change this:

```
6409 \ACRfullfmt{#1}{#2}{#3}%  
6410 }
```

\ACRfullfmt All upper case full format.

```
6411 \newcommand*{\ACRfullfmt}[3]{%  
6412 \acrlinkfullformat{\@ACRlong}{\@ACRshort}{#1}{#2}{#3}%  
6413 }
```

Plural:

\acrfullpl

```
6414 \newrobustcmd*{\acrfullpl}{\@gls@hyp@opt\ns@acrfullpl}  
  
6415 \newcommand*\ns@acrfullpl[2][{}]{%  
6416 \new@ifnextchar[{\@acrfullpl{#1}{#2}}%  
6417 {\@acrfullpl{#1}{#2}[]}%  
6418 }
```

Low-level macro:

```
6419 \def\@acrfullpl#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6420 \acrfullplfmt{#1}{#2}{#3}%  
6421 }
```

\acrfullplfmt No case change plural full format.

```
6422 \newcommand*{\acrfullplfmt}[3]{%  
6423 \acrlinkfullformat{\@acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%  
6424 }
```

\Acrfullpl

```
6425 \newrobustcmd*{\Acrfullpl}{\@gls@hyp@opt\ns@Acrfullpl}  
  
6426 \newcommand*\ns@Acrfullpl[2][{}]{%  
6427 \new@ifnextchar[{\@Acrfullpl{#1}{#2}}%  
6428 {\@Acrfullpl{#1}{#2}[]}%  
6429 }
```

Low-level macro:

```
6430 \def\@Acrfullpl#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6431 \Acrfullplfmt{#1}{#2}{#3}%  
6432 }
```

\Acrfullplfmt First letter upper case plural full format.

```
6433 \newcommand*{\Acrfullplfmt}[3]{%  
6434 \acrlinkfullformat{\@Acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%  
6435 }
```

```

\ACRfullpl
6436 \newrobustcmd*{\ACRfullpl}{\@gls@hyp@opt\@ns@ACRfullpl}

6437 \newcommand*\ns@ACRfullpl[2][{}]{%
6438   \new@ifnextchar[{\@ACRfullpl{#1}{#2}}{%
6439     {\@ACRfullpl{#1}{#2}[]}%
6440 }

```

Low-level macro:

```

6441 \def\@ACRfullpl#1#2[#3]{%
  Make it easier for acronym styles to change this:
6442   \ACRfullplfmt{#1}{#2}{#3}%
6443 }

```

```

\ACRfullplfmt  All upper case plural full format.
6444 \newcommand*{\ACRfullplfmt}[3]{%
6445   \acrlinkfullformat{\@ACRlongpl}{\@ACRshortpl}{#1}{#2}{#3}%
6446 }

```

1.18 Predefined acronym styles

`\acronymfont` This is only used with the additional acronym styles:

```
6447 \newcommand{\acronymfont}[1]{#1}
```

`\firstacronymfont` This is only used with the additional acronym styles:

```
6448 \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.

```
6449 \newcommand*{\acrnameformat}[2]{\acronymfont{#1}}
```

Define some tokens used by `\newacronym`:

`\glskeylisttok`

```
6450 \newtoks\glskeylisttok
```

`\glslabeltok`

```
6451 \newtoks\glslabeltok
```

`\glsshorttok`

```
6452 \newtoks\glsshorttok
```

`\gslongtok`

```
6453 \newtoks\gslongtok
```

`\newacronymhook` Provide a hook for `\newacronym`:

```
6454 \newcommand*{\newacronymhook}{}
```

genericNewAcronym New improved version of setting the acronym style.

```
6455 \newcommand*{\SetGenericNewAcronym}{%
```

Change the behaviour of \Glsentryname to workaround expansion issues that cause a problem for \makefirstuc

```
6456 \let\@Gls@entryname\@Gls@acrentryname
```

Change the way acronyms are defined:

```
6457 \renewcommand{\newacronym}[4][{}]{%
6458   \ifdefempty{\@glsacronymlists}%
6459   {%
6460     \def\@glo@type{\acronymtype}%
6461     \setkeys{glossentry}{##1}%
6462     \DeclareAcronymList{\@glo@type}%
6463   }%
6464 }%
6465 \glskeylisttok{##1}%
6466 \glslabeltok{##2}%
6467 \glsshorttok{##3}%
6468 \glslongtok{##4}%
6469 \newacronymhook
6470 \protected@edef\@do@newglossaryentry{%
6471   \noexpand\newglossaryentry{\the\glslabeltok}%
6472   {%
6473     type=\acronymtype,%
6474     name={\expandonce{\acronymentry{##2}}},%
6475     sort={\acronymssort{\the\glsshorttok}{\the\glslongtok}},%
6476     text={\the\glsshorttok},%
6477     short={\the\glsshorttok},%
6478     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6479     long={\the\glslongtok},%
6480     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6481     \GenericAcronymFields,%
6482     \the\glskeylisttok
6483   }%
6484 }%
6485 \@do@newglossaryentry
6486 }%
```

Make sure that \acrfull etc reflects the new style:

```
6487 \renewcommand*{\acrfullfmt}[3]{%
6488   \glslink[##1]{##2}{\genacrfullformat{##2}{##3}}}%
6489 \renewcommand*{\Acrfullfmt}[3]{%
6490   \glslink[##1]{##2}{\Genacrfullformat{##2}{##3}}}%
6491 \renewcommand*{\ACRfullfmt}[3]{%
6492   \glslink[##1]{##2}{%
6493     \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}%
6494 \renewcommand*{\acrfullplfmt}[3]{%
6495   \glslink[##1]{##2}{\genplacrfullformat{##2}{##3}}}%
6496 \renewcommand*{\Acrfullplfmt}[3]{%
```



```

6497 \glslink{##1}{##2}{\Genplacrformat{##2}{##3}}}%
6498 \renewcommand*{\ACRfullplfmt}[3]{%
6499 \glslink{##1}{##2}{%
6500 \mfirstucMakeUppercase{\genplacrformat{##2}{##3}}}}%

```

Make sure that `\glsentryfull` etc reflects the new style:

```

6501 \renewcommand*{\glsentryfull}[1]{\genacrformat{##1}{}}%
6502 \renewcommand*{\Glsentryfull}[1]{\Genacrformat{##1}{}}%
6503 \renewcommand*{\glsentryfullpl}[1]{\genplacrformat{##1}{}}%
6504 \renewcommand*{\Glsentryfullpl}[1]{\Genplacrformat{##1}{}}%
6505 }

```

`\icAcronymFields` Fields used by `\SetGenericNewAcronym` that can be changed by the acronym style.

```

6506 \newcommand*{\GenericAcronymFields}{description={\the\glslongtok}}

```

`\acronymentry` `\acronymentry{<label>}`

Display style for the name field in the list of acronyms.

```

6507 \newcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{#1}}}

```

`\acronymsort` `\acronymsort{<short>}{<long>}`

Default sort format for acronyms.

```

6508 \newcommand*{\acronymsort}[2]{#1}

```

`\setacronymstyle` `\setacronymstyle{<style name>}`

```

6509 \newcommand*{\setacronymstyle}[1]{%
6510 \ifcsundef{@glsacr@dispstyle@#1}
6511 {%
6512 \PackageError{glossaries}{Undefined acronym style ‘#1’}{%
6513 }%
6514 {%
6515 \ifdefempty{\@glsacronymlists}%
6516 {%
6517 \DeclareAcronymList{\acronymtype}%
6518 }%
6519 }%
6520 \SetGenericNewAcronym
6521 \GlsUseAcrStyleDefs{#1}%
6522 \@for\@gls@type:=\@glsacronymlists\do{%
6523 \defglsentryfmt[\@gls@type]{\GlsUseAcrEntryDispStyle{#1}}%
6524 }%
6525 }%
6526 }

```

`\newacronymstyle`

```
\newacronymstyle{<style name>}{<entry format definition>}{<display
definitions>}
```

Defines a new acronym style called *<style name>*.

```
6527 \newcommand*{\newacronymstyle}[3]{%
6528   \ifcsdef{@glsacr@dispstyle@#1}%
6529   {%
6530     \PackageError{glossaries}{Acronym style ‘#1’ already exists}{}%
6531   }%
6532   {%
6533     \csdef{@glsacr@dispstyle@#1}{#2}%
6534     \csdef{@glsacr@styledefs@#1}{#3}%
6535   }%
6536 }
```

`\renewacronymstyle` Redefines the given acronym style.

```
6537 \newcommand*{\renewacronymstyle}[3]{%
6538   \ifcsdef{@glsacr@dispstyle@#1}%
6539   {%
6540     \csdef{@glsacr@dispstyle@#1}{#2}%
6541     \csdef{@glsacr@styledefs@#1}{#3}%
6542   }%
6543   {%
6544     \PackageError{glossaries}{Acronym style ‘#1’ doesn’t exist}{}%
6545   }%
6546 }
```

`\GlsUseAcrEntryDispStyle`

```
6547 \newcommand*{\GlsUseAcrEntryDispStyle}[1]{\csuse{@glsacr@dispstyle@#1}}
```

`\GlsUseAcrStyleDefs`

```
6548 \newcommand*{\GlsUseAcrStyleDefs}[1]{\csuse{@glsacr@styledefs@#1}}
```

Predefined acronym styles:

`long-short` *<long>* (*<short>*) acronym style.

```
6549 \newacronymstyle{long-short}%
6550 {%
```

Check for long form in case this is a mixed glossary.

```
6551   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6552 }%
6553 {%
6554   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6555   \renewcommand*{\genacrfullformat}[2]{%
6556     \glsentrylong{##1}##2\space
6557     (\protect\firstacronymfont{\glsentryshort{##1}})%
6558   }%
6559   \renewcommand*{\Genacrfullformat}[2]{%
```

```

6560 \Glsentrylong{##1}##2\space
6561 (\protect\firstacronymfont{\glsentryshort{##1}})%
6562 }%
6563 \renewcommand*{\genplacrfullformat}[2]{%
6564 \glsentrylongpl{##1}##2\space
6565 (\protect\firstacronymfont{\glsentryshortpl{##1}})%
6566 }%
6567 \renewcommand*{\Genplacrfullformat}[2]{%
6568 \Glsentrylongpl{##1}##2\space
6569 (\protect\firstacronymfont{\glsentryshortpl{##1}})%
6570 }%
6571 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6572 \renewcommand*{\acronymsort}[2]{##1}%
6573 \renewcommand*{\acronymfont}[1]{##1}%
6574 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
6575 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6576 }

```

long-sp-short Similar to the previous style but allows the space between the long and short form to be customized.

```

6577 \newacronymstyle{long-sp-short}%
6578 {%
  Check for long form in case this is a mixed glossary.
6579 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6580 }%
6581 {%
6582 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6583 \renewcommand*{\genacrfullformat}[2]{%
6584 \glsentrylong{##1}##2\glsacspace{##1}%
6585 (\protect\firstacronymfont{\glsentryshort{##1}})%
6586 }%
6587 \renewcommand*{\Genacrfullformat}[2]{%
6588 \Glsentrylong{##1}##2\glsacspace{##1}%
6589 (\protect\firstacronymfont{\glsentryshort{##1}})%
6590 }%
6591 \renewcommand*{\genplacrfullformat}[2]{%
6592 \glsentrylongpl{##1}##2\glsacspace{##1}%
6593 (\protect\firstacronymfont{\glsentryshortpl{##1}})%
6594 }%
6595 \renewcommand*{\Genplacrfullformat}[2]{%
6596 \Glsentrylongpl{##1}##2\glsacspace{##1}%
6597 (\protect\firstacronymfont{\glsentryshortpl{##1}})%
6598 }%
6599 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6600 \renewcommand*{\acronymsort}[2]{##1}%
6601 \renewcommand*{\acronymfont}[1]{##1}%
6602 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
6603 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6604 }

```

`\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.

```
6605 \newcommand*{\glsacspace}[1]{%
6606   \settowidth{\dimen@}{(\firstacronymfont{\glsentryshort{#1}})}%
6607   \ifdim\dimen@<3em~\else\space\fi
6608 }
```

`short-long` *<short>* (*<long>*) acronym style.

```
6609 \newacronymstyle{short-long}%
6610 {%
```

Check for long form in case this is a mixed glossary.

```
6611   \ifglshaslong{\glslabel}{\glsacspace}{\glsacspace}%
6612 }%
6613 {%
6614   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6615   \renewcommand*{\genacrfullformat}[2]{%
6616     \protect\firstacronymfont{\glsentryshort{##1}}##2\space
6617     (\glsentrylong{##1})%
6618   }%
6619   \renewcommand*{\Genacrfullformat}[2]{%
6620     \protect\firstacronymfont{\Glsentryshort{##1}}##2\space
6621     (\glsentrylong{##1})%
6622   }%
6623   \renewcommand*{\genplacrfullformat}[2]{%
6624     \protect\firstacronymfont{\glsentryshortpl{##1}}##2\space
6625     (\glsentrylongpl{##1})%
6626   }%
6627   \renewcommand*{\Genplacrfullformat}[2]{%
6628     \protect\firstacronymfont{\Glsentryshortpl{##1}}##2\space
6629     (\glsentrylongpl{##1})%
6630   }%
6631   \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6632   \renewcommand*{\acronymsort}[2]{##1}%
6633   \renewcommand*{\acronymfont}[1]{##1}%
6634   \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
6635   \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6636 }
```

`long-sc-short` *<long>* (*\textsc{<short>}*) acronym style.

```
6637 \newacronymstyle{long-sc-short}%
6638 {%
6639   \GlsUseAcrEntryDispStyle{long-short}%
6640 }%
6641 {%
6642   \GlsUseAcrStyleDefs{long-short}%
6643   \renewcommand*{\acronymfont}[1]{\textsc{##1}}%
6644   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6645 }
```

long-sm-short *<long>* (\textsmaller{<short>}) acronym style.

```

6646 \newacronymstyle{long-sm-short}%
6647 {%
6648   \GlsUseAcrEntryDisplayStyle{long-short}%
6649 }%
6650 {%
6651   \GlsUseAcrStyleDefs{long-short}%
6652   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6653   \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6654 }

```

sc-short-long *<short>* (\textsc{<long>}) acronym style.

```

6655 \newacronymstyle{sc-short-long}%
6656 {%
6657   \GlsUseAcrEntryDisplayStyle{short-long}%
6658 }%
6659 {%
6660   \GlsUseAcrStyleDefs{short-long}%
6661   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6662   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6663 }

```

sm-short-long *<short>* (\textsmaller{<long>}) acronym style.

```

6664 \newacronymstyle{sm-short-long}%
6665 {%
6666   \GlsUseAcrEntryDisplayStyle{short-long}%
6667 }%
6668 {%
6669   \GlsUseAcrStyleDefs{short-long}%
6670   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6671   \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6672 }

```

long-short-desc *<long>* ({<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

6673 \newacronymstyle{long-short-desc}%
6674 {%
6675   \GlsUseAcrEntryDisplayStyle{long-short}%
6676 }%
6677 {%
6678   \GlsUseAcrStyleDefs{long-short}%
6679   \renewcommand*{\GenericAcronymFields}{}%
6680   \renewcommand*{\acronymsort}[2]{##2}%
6681   \renewcommand*{\acronymentry}[1]{%
6682     \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6683 }

```

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.

```

6684 \newacronymstyle{long-sp-short-desc}%
6685 {%
6686   \GlsUseAcrEntryDispStyle{long-sp-short}%
6687 }%
6688 {%
6689   \GlsUseAcrStyleDefs{long-sp-short}%
6690   \renewcommand*{\GenericAcronymFields}{}%
6691   \renewcommand*{\acronymsort}[2]{##2}%
6692   \renewcommand*{\acronymentry}[1]{%
6693     \glentrylong{##1}\glspacespace{##1}(\acronymfont{\glentryshort{##1}})}%
6694 }

```

g-sc-short-desc *<long>* (\textsc{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

6695 \newacronymstyle{long-sc-short-desc}%
6696 {%
6697   \GlsUseAcrEntryDispStyle{long-sc-short}%
6698 }%
6699 {%
6700   \GlsUseAcrStyleDefs{long-sc-short}%
6701   \renewcommand*{\GenericAcronymFields}{}%
6702   \renewcommand*{\acronymsort}[2]{##2}%
6703   \renewcommand*{\acronymentry}[1]{%
6704     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6705 }

```

g-sm-short-desc *<long>* (\textsmaller{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

6706 \newacronymstyle{long-sm-short-desc}%
6707 {%
6708   \GlsUseAcrEntryDispStyle{long-sm-short}%
6709 }%
6710 {%
6711   \GlsUseAcrStyleDefs{long-sm-short}%
6712   \renewcommand*{\GenericAcronymFields}{}%
6713   \renewcommand*{\acronymsort}[2]{##2}%
6714   \renewcommand*{\acronymentry}[1]{%
6715     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6716 }

```

short-long-desc *<short>* ({<long>}) acronym style that has an accompanying description (which the user needs to supply).

```

6717 \newacronymstyle{short-long-desc}%
6718 {%
6719   \GlsUseAcrEntryDispStyle{short-long}%
6720 }%
6721 {%
6722   \GlsUseAcrStyleDefs{short-long}%
6723   \renewcommand*{\GenericAcronymFields}{}%

```

```

6724 \renewcommand*{\acronymsort}[2]{##2}%
6725 \renewcommand*{\acronymentry}[1]{%
6726     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6727 }

```

short-long-desc *<long>* (\textsc{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

6728 \newacronymstyle{sc-short-long-desc}%
6729 {%
6730     \GlsUseAcrEntryDispStyle{sc-short-long}%
6731 }%
6732 {%
6733     \GlsUseAcrStyleDefs{sc-short-long}%
6734     \renewcommand*{\GenericAcronymFields}{}%
6735     \renewcommand*{\acronymsort}[2]{##2}%
6736     \renewcommand*{\acronymentry}[1]{%
6737         \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6738 }

```

short-long-desc *<long>* (\textsmaller{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

6739 \newacronymstyle{sm-short-long-desc}%
6740 {%
6741     \GlsUseAcrEntryDispStyle{sm-short-long}%
6742 }%
6743 {%
6744     \GlsUseAcrStyleDefs{sm-short-long}%
6745     \renewcommand*{\GenericAcronymFields}{}%
6746     \renewcommand*{\acronymsort}[2]{##2}%
6747     \renewcommand*{\acronymentry}[1]{%
6748         \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6749 }

```

dua *<long>* only acronym style.

```

6750 \newacronymstyle{dua}%
6751 {%

```

Check for long form in case this is a mixed glossary.

```

6752 \ifdefempty\glscustomtext
6753 {%
6754     \ifglshaslong{\glslabel}%
6755     {%
6756         \glcifplural
6757         {%

```

Plural form:

```

6758         \glscapscase
6759         {%

```

Plural form, don't adjust case:

```
6760      \glsentrylongpl{\glslabel}\glsinsert
6761      }%
6762      {%
```

Plural form, make first letter upper case:

```
6763      \Glsentrylongpl{\glslabel}\glsinsert
6764      }%
6765      {%
```

Plural form, all caps:

```
6766      \mfirstucMakeUppercase
6767      {\glsentrylongpl{\glslabel}\glsinsert}%
6768      }%
6769      }%
6770      {%
```

Singular form

```
6771      \glscapscase
6772      {%
```

Singular form, don't adjust case:

```
6773      \glsentrylong{\glslabel}\glsinsert
6774      }%
6775      {%
```

Subsequent singular form, make first letter upper case:

```
6776      \Glsentrylong{\glslabel}\glsinsert
6777      }%
6778      {%
```

Subsequent singular form, all caps:

```
6779      \mfirstucMakeUppercase
6780      {\glsentrylong{\glslabel}\glsinsert}%
6781      }%
6782      }%
6783      }%
6784      {%
```

Not an acronym:

```
6785      \glsgenentryfmt
6786      }%
6787      }%
6788      {\glscustomtext\glsinsert}%
6789      }%
6790      {%
6791      \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%

6792      \renewcommand*{\acrfullfmt}[3]{%
6793      \glslink{##1}{##2}{\glsentrylong{##2}##3\space
6794      (\acronymfont{\glsentryshort{##2}})}}%
6795      \renewcommand*{\Acrfullfmt}[3]{%
```



```

6796 \glslink{##1}{##2}{\Glsentrylong{##2}##3\space
6797 (\acronymfont{\glsentryshort{##2}})}}}%
6798 \renewcommand*{\ACRfullfmt}[3]{%
6799 \glslink{##1}{##2}{%
6800 \mfirstucMakeUppercase{\glsentrylong{##2}##3\space
6801 (\acronymfont{\glsentryshort{##2}})}}}%

6802 \renewcommand*{\acrfullplfmt}[3]{%
6803 \glslink{##1}{##2}{\Glsentrylongpl{##2}##3\space
6804 (\acronymfont{\glsentryshortpl{##2}})}}}%

6805 \renewcommand*{\Acrfullplfmt}[3]{%
6806 \glslink{##1}{##2}{\Glsentrylongpl{##2}##3\space
6807 (\acronymfont{\glsentryshortpl{##2}})}}}%
6808 \renewcommand*{\ACRfullplfmt}[3]{%
6809 \glslink{##1}{##2}{%
6810 \mfirstucMakeUppercase{\glsentrylongpl{##2}##3\space
6811 (\acronymfont{\glsentryshortpl{##2}})}}}%
6812 \renewcommand*{\glsentryfull}[1]{%
6813 \glsentrylong{##1}\space(\acronymfont{\glsentryshort{##1}})%
6814 }%
6815 \renewcommand*{\Glsentryfull}[1]{%
6816 \Glsentrylong{##1}\space(\acronymfont{\glsentryshort{##1}})%
6817 }%
6818 \renewcommand*{\glsentryfullpl}[1]{%
6819 \glsentrylongpl{##1}\space(\acronymfont{\glsentryshortpl{##1}})%
6820 }%
6821 \renewcommand*{\Glsentryfullpl}[1]{%
6822 \Glsentrylongpl{##1}\space(\acronymfont{\glsentryshortpl{##1}})%
6823 }%
6824 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6825 \renewcommand*{\acronymsort}[2]{##1}%
6826 \renewcommand*{\acronymfont}[1]{##1}%
6827 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6828 }

```

dua-desc <long> only acronym style with user-supplied description.

```

6829 \newacronymstyle{dua-desc}%
6830 {%
6831 \GlsUseAcrEntryDispStyle{dua}%
6832 }%
6833 {%
6834 \GlsUseAcrStyleDefs{dua}%
6835 \renewcommand*{\GenericAcronymFields}{}%

6836 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentrylong{##1}}}%
6837 \renewcommand*{\acronymsort}[2]{##2}%
6838 }%

```

footnote *<short>*\footnote{*<long>*} acronym style.

```
6839 \newacronymstyle{footnote}%  
6840 {%
```

Check for long form in case this is a mixed glossary.

```
6841 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%  
6842 }%  
6843 {%  
6844 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
```

Need to ensure hyperlinks are switched off on first use:

```
6845 \glshyperfirstfalse  
6846 \renewcommand*{\genacrfullformat}[2]{%  
6847 \protect\firstacronymfont{\glsentryshort{##1}}##2%  
6848 \protect\footnote{\glsentrylong{##1}}%  
6849 }%  
6850 \renewcommand*{\Genacrfullformat}[2]{%  
6851 \firstacronymfont{\Glsentryshort{##1}}##2%  
6852 \protect\footnote{\glsentrylong{##1}}%  
6853 }%  
6854 \renewcommand*{\genplacrfullformat}[2]{%  
6855 \protect\firstacronymfont{\glsentryshortpl{##1}}##2%  
6856 \protect\footnote{\glsentrylongpl{##1}}%  
6857 }%  
6858 \renewcommand*{\Genplacrfullformat}[2]{%  
6859 \protect\firstacronymfont{\Glsentryshortpl{##1}}##2%  
6860 \protect\footnote{\glsentrylongpl{##1}}%  
6861 }%  
6862 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%  
6863 \renewcommand*{\acronymsort}[2]{##1}%  
6864 \renewcommand*{\acronymfont}[1]{##1}%  
6865 \renewcommand*{\acrpluralsuffix}{\glssacrpluralsuffix}%
```

Don't use footnotes for \acrfull:

```
6866 \renewcommand*{\acrfullfmt}[3]{%  
6867 \glslink[##1]{##2}{\acronymfont{\glsentryshort{##2}}##3\space  
6868 (\glsentrylong{##2})}%  
6869 \renewcommand*{\Acrfullfmt}[3]{%  
6870 \glslink[##1]{##2}{\acronymfont{\Glsentryshort{##2}}##3\space  
6871 (\glsentrylong{##2})}%  
6872 \renewcommand*{\ACRfullfmt}[3]{%  
6873 \glslink[##1]{##2}{%  
6874 \mfirstucMakeUppercase{\acronymfont{\glsentryshort{##2}}##3\space  
6875 (\glsentrylong{##2})}}}%  
6876 \renewcommand*{\acrfullplfmt}[3]{%  
6877 \glslink[##1]{##2}{\acronymfont{\glsentryshortpl{##2}}##3\space  
6878 (\glsentrylongpl{##2})}%  
6879 \renewcommand*{\Acrfullplfmt}[3]{%  
6880 \glslink[##1]{##2}{\acronymfont{\Glsentryshortpl{##2}}##3\space  
6881 (\glsentrylongpl{##2})}%
```

```

6882 \renewcommand*{\ACRfullplfmt}[3]{%
6883   \glslink{##1}{##2}{%
6884     \mfirstucMakeUppercase{\acronymfont{\glsentryshortpl{##2}}##3\space
6885     (\glsentrylongpl{##2})}}}%

```

Similarly for \glsentryfull etc:

```

6886 \renewcommand*{\glsentryfull}[1]{%
6887   \acronymfont{\glsentryshort{##1}}\space(\glsentrylong{##1})}%
6888 \renewcommand*{\Glsentryfull}[1]{%
6889   \acronymfont{\Glsentryshort{##1}}\space(\glsentrylong{##1})}%
6890 \renewcommand*{\glsentryfullpl}[1]{%
6891   \acronymfont{\glsentryshortpl{##1}}\space(\glsentrylongpl{##1})}%
6892 \renewcommand*{\Glsentryfullpl}[1]{%
6893   \acronymfont{\Glsentryshortpl{##1}}\space(\glsentrylongpl{##1})}%
6894 }

```

footnote-sc \textsc{<short>}\footnote{<long>} acronym style.

```

6895 \newacronymstyle{footnote-sc}%
6896 {%
6897   \GlsUseAcrEntryDisplayStyle{footnote}%
6898 }%
6899 {%
6900   \GlsUseAcrStyleDefs{footnote}%
6901   \renewcommand{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}
6902   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6903   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6904 }%

```

footnote-sm \textsmaller{<short>}\footnote{<long>} acronym style.

```

6905 \newacronymstyle{footnote-sm}%
6906 {%
6907   \GlsUseAcrEntryDisplayStyle{footnote}%
6908 }%
6909 {%
6910   \GlsUseAcrStyleDefs{footnote}%
6911   \renewcommand{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}
6912   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6913   \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6914 }%

```

footnote-desc <short>\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

6915 \newacronymstyle{footnote-desc}%
6916 {%
6917   \GlsUseAcrEntryDisplayStyle{footnote}%
6918 }%
6919 {%
6920   \GlsUseAcrStyleDefs{footnote}%
6921   \renewcommand*{\GenericAcronymFields}{}%

```

```

6922 \renewcommand*{\acronymsort}[2]{##2}%
6923 \renewcommand*{\acronymentry}[1]{%
6924     \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6925 }

```

ootnote-sc-desc \textsc{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

6926 \newacronymstyle{footnote-sc-desc}%
6927 {%
6928     \GlsUseAcrEntryDispStyle{footnote-sc}%
6929 }%
6930 {%
6931     \GlsUseAcrStyleDefs{footnote-sc}%
6932     \renewcommand*{\GenericAcronymFields}{}%
6933     \renewcommand*{\acronymsort}[2]{##2}%
6934     \renewcommand*{\acronymentry}[1]{%
6935         \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6936 }

```

ootnote-sm-desc \textsmaller{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

6937 \newacronymstyle{footnote-sm-desc}%
6938 {%
6939     \GlsUseAcrEntryDispStyle{footnote-sm}%
6940 }%
6941 {%
6942     \GlsUseAcrStyleDefs{footnote-sm}%
6943     \renewcommand*{\GenericAcronymFields}{}%
6944     \renewcommand*{\acronymsort}[2]{##2}%
6945     \renewcommand*{\acronymentry}[1]{%
6946         \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6947 }

```

AcronymSynonyms

```

6948 \newcommand*{\DefineAcronymSynonyms}{%

```

Short form

\acs

```

6949 \let\acs\acrshort

```

First letter uppercase short form

\Acs

```

6950 \let\Acs\Acrshort

```

Plural short form

\acsp

```

6951 \let\acsp\acrshortpl

```

First letter uppercase plural short form

\Acsp

6952 \let\Acsp\Acrshorttpl

Long form

\acl

6953 \let\acl\acrlong

Plural long form

\aclp

6954 \let\aclp\acrlongpl

First letter upper case long form

\Acl

6955 \let\Acl\Acrlong

First letter upper case plural long form

\Aclp

6956 \let\Aclp\Acrlongpl

Full form

\acf

6957 \let\acf\acrfull

Plural full form

\acfp

6958 \let\acfp\acrfullpl

First letter upper case full form

\Acf

6959 \let\Acf\Acrfull

First letter upper case plural full form

\Acfp

6960 \let\Acfp\Acrfullpl

Standard form

\ac

6961 \let\ac\gls

First upper case standard form

\Ac

6962 \let\Ac\Gls

Standard plural form

`\acp`

```
6963 \let\acp\glspl
```

Standard first letter upper case plural form

`\Acp`

```
6964 \let\Acp\Glspl
```

```
6965 }
```

Define synonyms if required

```
6966 \ifglsacrshortcuts
```

```
6967 \DefineAcronymSynonyms
```

```
6968 \fi
```

These commands for setting the style are now deprecated but are kept for backward compatibility.

`\glsacronymDisplayStyle` Sets the default acronym display style for given glossary.

```
6969 \newcommand*{\SetDefaultAcronymDisplayStyle}[1]{%
```

```
6970 \defglsentryfmt[#1]{\glsentryfmt}%
```

```
6971 }
```

`\glsnewacronym` Sets up the acronym definition for the default style. The information is provided by the tokens

`\glslabeltok`, `\glsshorttok`, `\gslongtok` and `\glskeylisttok`.

```
6972 \newcommand*{\DefaultNewAcronymDef}{%
```

```
6973 \edef\@do@newglossaryentry{%
```

```
6974 \noexpand\newglossaryentry{\the\glslabeltok}%
```

```
6975 {%
```

```
6976 type=\acronymtype,%
```

```
6977 name={\the\glsshorttok},%
```

```
6978 sort={\the\glsshorttok},%
```

```
6979 text={\the\glsshorttok},%
```

```
6980 first={\acrfullformat{\the\gslongtok}{\the\glsshorttok}},%
```

```
6981 plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
```

```
6982 firstplural={\acrfullformat{\noexpand\expandonce\noexpand\@glo@longpl}%
```

```
6983 {\noexpand\expandonce\noexpand\@glo@shortpl}},%
```

```
6984 short={\the\glsshorttok},%
```

```
6985 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
```

```
6986 long={\the\gslongtok},%
```

```
6987 longplural={\the\gslongtok\noexpand\acrpluralsuffix},%
```

```
6988 description={\the\gslongtok},%
```

```
6989 descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
```

Remaining options specified by the user:

```
6990 \the\glskeylisttok
```

```
6991 }%
```

```
6992 }%
```

```
6993 \let\@org@gls@assign@firstpl\gls@assign@firstpl
```

```

6994 \let\@org@gls@assign@plural\gls@assign@plural
6995 \let\@org@gls@assign@descplural\gls@assign@descplural
6996 \def\gls@assign@firstpl##1##2{%
6997   \@@gls@expand@field{##1}{firstpl}{##2}%
6998 }%
6999 \def\gls@assign@plural##1##2{%
7000   \@@gls@expand@field{##1}{plural}{##2}%
7001 }%
7002 \def\gls@assign@descplural##1##2{%
7003   \@@gls@expand@field{##1}{descplural}{##2}%
7004 }%
7005 \do@newglossaryentry
7006 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7007 \let\gls@assign@plural\@org@gls@assign@plural
7008 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7009 }

```

ultAcronymStyle Set up the default acronym style:

```
7010 \newcommand*{\SetDefaultAcronymStyle}{%
```

Set the display style:

```

7011   \@for\@gls@type:=\@glsacronymlists\do{%
7012     \SetDefaultAcronymDisplayStyle{\@gls@type}%
7013   }%

```

Set up the definition of \newacronym:

```
7014 \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).

```

7015   \ifx\@glsacronymlists\@empty
7016     \def\@glo@type{\acronymtype}%
7017     \setkeys{glossentry}{##1}%
7018     \DeclareAcronymList{\@glo@type}%
7019     \SetDefaultAcronymDisplayStyle{\@glo@type}%
7020   \fi
7021   \glskeylisttok{##1}%
7022   \glslabeltok{##2}%
7023   \glsshorttok{##3}%
7024   \glslongtok{##4}%
7025   \newacronymhook
7026   \DefaultNewAcronymDef
7027 }%
7028 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
7029 }

```

\acrfootnote Used by the footnote acronym styles.

```
7030 \newcommand*{\acrfootnote}[3]{\acrlinkfootnote{#1}{#2}{#3}}
```

acrlinkfootnote

```

7031 \newcommand*{\acrlinkfootnote}[3]{%
7032   \footnote{\glslink[#1]{#2}{#3}}%
7033 }

```

acrnolinkfootnote

```

7034 \newcommand*{\acrnolinkfootnote}[3]{%
7035   \footnote{#3}%
7036 }

```

acronymDisplayStyle Sets the acronym display style for given glossary for the description and footnote combination.

```

7037 \newcommand*{\SetDescriptionFootnoteAcronymDisplayStyle}[1]{%
7038   \def\glsentryfmt[#1]{%

7039     \ifdefempty\glscustomtext
7040     {%
7041       \ifglsused{\glslabel}%
7042       {%
7043         \acronymfont{\glsentryfmt}%
7044       }%
7045       {%
7046         \firstacronymfont{\glsentryfmt}%
7047         \ifgls hassymbol{\glslabel}%
7048         {%
7049           \expandafter\protect\expandafter\acrfootnote\expandafter
7050             {\@gls@link@opts}{\@gls@link@label}%
7051           {%
7052             \glsifplural
7053               {\glsentrysymbolplural{\glslabel}}%
7054               {\glsentrysymbol{\glslabel}}%
7055             }%
7056           }%
7057         }%
7058       }%
7059     {\glscustomtext\glsinsert}%
7060   }%
7061 }

```

acronymNewAcronymDef

```

7062 \newcommand*{\DescriptionFootnoteNewAcronymDef}{%
7063   \edef\@do@newglossaryentry{%
7064     \noexpand\newglossaryentry{\the\glslabeltok}%
7065     {%
7066       type=\acronymtype,%
7067       name={\noexpand\acronymfont{\the\glsshorttok}},%
7068       sort={\the\glsshorttok},%
7069       first={\the\glsshorttok},%
7070       firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7071       text={\the\glsshorttok},%

```



```

7072 plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7073 short={\the\glsshorttok},%
7074 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7075 long={\the\glslongtok},%
7076 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7077 symbol={\the\glslongtok},%
7078 symbolplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7079 \the\glskeylisttok
7080 }%
7081 }%
7082 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7083 \let\@org@gls@assign@plural\gls@assign@plural
7084 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7085 \def\gls@assign@firstpl##1##2{%
7086   \@@gls@expand@field{##1}{firstpl}{##2}%
7087 }%
7088 \def\gls@assign@plural##1##2{%
7089   \@@gls@expand@field{##1}{plural}{##2}%
7090 }%
7091 \def\gls@assign@symbolplural##1##2{%
7092   \@@gls@expand@field{##1}{symbolplural}{##2}%
7093 }%
7094 \do@newglossaryentry
7095 \let\gls@assign@plural\@org@gls@assign@plural
7096 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7097 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7098 }

```

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.

```

7099 \newcommand*{\SetDescriptionFootnoteAcronymStyle}{%
7100   \renewcommand{\newacronym}[4][\]{%
7101     \ifx\@glsacronymlists\@empty
7102       \def\@glo@type{\acronymtype}%
7103       \setkeys{glossentry}{##1}%
7104       \DeclareAcronymList{\@glo@type}%
7105       \SetDescriptionFootnoteAcronymDisplayStyle{\@glo@type}%
7106     \fi
7107     \glskeylisttok{##1}%
7108     \glslabeltok{##2}%
7109     \glsshorttok{##3}%
7110     \glslongtok{##4}%
7111     \newacronymhook
7112     \DescriptionFootnoteNewAcronymDef
7113   }%

```

If footnote package option is specified, set the first use to append the long form (stored in

symbol) as a footnote.

```

7114 \for\@gls@type:=\@glsacronymlists\do{%
7115   \SetDescriptionFootnoteAcronymDisplayStyle{\@gls@type}%
7116 }%
```

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.

```

7117 \ifglsacrsmallcaps
7118   \renewcommand*{\acronymfont}[1]{\textsc{##1}}%
7119   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7120 \else
7121   \ifglsacrsmaller
7122     \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
7123   \fi
7124 \fi
```

Check for package option clash

```

7125 \ifglsacrdua
7126   \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’
7127     can’t both be set}{}%
7128 \fi
7129 }%
```

nymDisplayStyle Sets the acronym display style for given glossary with description and dua combination.

```

7130 \newcommand*{\SetDescriptionDUAAcronymDisplayStyle}[1]{%
7131   \defglsentryfmt[##1]{\glsentryfmt}%
7132 }
```

UANewAcronymDef

```

7133 \newcommand*{\DescriptionDUANewAcronymDef}{%
7134   \edef\@do@newglossaryentry{%
7135     \noexpand\newglossaryentry{\the\glslabeltok}%
7136     {%
7137       type=\acronymtype,%
7138       name={\the\glslongtok},%
7139       sort={\the\glslongtok},%
7140       text={\the\glslongtok},%
7141       first={\the\glslongtok},%
7142       plural={\noexpand\expandonce\noexpand\@glo@longpl},%
7143       firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7144       short={\the\glsshorttok},%
7145       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7146       long={\the\glslongtok},%
7147       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7148       symbol={\the\glsshorttok},%
7149       symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7150       \the\glskeylisttok
7151     }%
7152   }%
```

```

7153 \let\org@gl@assign@firstpl\gl@assign@firstpl
7154 \let\org@gl@assign@plural\gl@assign@plural
7155 \let\org@gl@assign@symbolplural\gl@assign@symbolplural
7156 \def\gl@assign@firstpl##1##2{%
7157   \@gl@expand@field{##1}{firstpl}{##2}%
7158 }%
7159 \def\gl@assign@plural##1##2{%
7160   \@gl@expand@field{##1}{plural}{##2}%
7161 }%
7162 \def\gl@assign@symbolplural##1##2{%
7163   \@gl@expand@field{##1}{symbolplural}{##2}%
7164 }%
7165 \do@newglossaryentry
7166 \let\gl@assign@firstpl\org@gl@assign@firstpl
7167 \let\gl@assign@plural\org@gl@assign@plural
7168 \let\gl@assign@symbolplural\org@gl@assign@symbolplural
7169 }

```

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.

```

7170 \newcommand*\SetDescriptionDUAACronymStyle{%
7171   \ifgl@smallcaps
7172     \PackageError{glossaries}{Option clash: 'smallcaps' and 'dua'
7173       can't both be set}{}%
7174   \else
7175     \ifgl@smaller
7176       \PackageError{glossaries}{Option clash: 'smaller' and 'dua'
7177         can't both be set}{}%
7178     \fi
7179   \fi
7180   \renewcommand{\newacronym}[4][{}]{%
7181     \ifx\@gl@acronymlists\@empty
7182       \def\@glo@type{\acronymtype}%
7183       \setkeys{glossentry}{##1}%
7184       \DeclareAcronymList{\@glo@type}%
7185       \SetDescriptionDUAACronymDisplayStyle{\@glo@type}%
7186     \fi
7187     \gl@keylisttok{##1}%
7188     \gl@labeltok{##2}%
7189     \gl@shorttok{##3}%
7190     \gl@longtok{##4}%
7191     \newacronymhook
7192     \DescriptionDUANewAcronymDef
7193   }%

```

Set display.

```

7194 \@for\@gl@type:=\@gl@acronymlists\do{%
7195   \SetDescriptionDUAACronymDisplayStyle{\@gl@type}%

```

```

7196 }%
7197 }%

```

`\acronymDisplayStyle` Sets the acronym display style for given glossary using the description setting (but not footnote or dua).

```

7198 \newcommand*{\SetDescriptionAcronymDisplayStyle}[1]{%
7199   \def\glsentryfmt[#1]{%

7200     \ifdefempty\glscustomtext
7201     {%
7202       \ifglshassymbol{\glslabel}%
7203       {%

```

Move the inserted text outside of `\acronymfont`

```

7204       \let\gls@org@insert\glsinsert
7205       \let\glsinsert\@empty
7206       \acronymfont{\glsentryfmt}\gls@org@insert
7207     }%
7208     {%
7209       \glsentryfmt
7210       \ifglshassymbol{\glslabel}%
7211       {%
7212         \glsifplural
7213         {%
7214           \def\@glo@symbol{\glsentrysymbolplural{\glslabel}}%
7215           }%
7216           {%
7217             \def\@glo@symbol{\glsentrysymbol{\glslabel}}%
7218             }%
7219             \space(\protect\firstacronymfont
7220             {\glscapscase
7221              {\@glo@symbol}
7222              {\@glo@symbol}
7223              {\mfirstucMakeUppercase{\@glo@symbol}}})%
7224           }%
7225         }%
7226       }%
7227     }%
7228     {\glscustomtext\glsinsert}%
7229   }%
7230 }

```

`\onNewAcronymDef`

```

7231 \newcommand*{\DescriptionNewAcronymDef}{%
7232   \edef\@do@newglossaryentry{%
7233     \noexpand\newglossaryentry{\the\glslabeltok}%
7234     {%
7235       type=\acronymtype,%
7236       name={\noexpand

```

```

7237     \acronymformat{\the\glsshorttok}{\the\glslongtok}},%
7238     sort={\the\glsshorttok},%
7239     first={\the\glslongtok},%
7240     firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7241     text={\the\glsshorttok},%
7242     plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7243     short={\the\glsshorttok},%
7244     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7245     long={\the\glslongtok},%
7246     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7247     symbol={\noexpand\@glo@text},%
7248     symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7249     \the\glskeylisttok}%
7250 }%
7251 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7252 \let\@org@gls@assign@plural\gls@assign@plural
7253 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7254 \def\gls@assign@firstpl##1##2{%
7255   \@@gls@expand@field{##1}{firstpl}{##2}%
7256 }%
7257 \def\gls@assign@plural##1##2{%
7258   \@@gls@expand@field{##1}{plural}{##2}%
7259 }%
7260 \def\gls@assign@symbolplural##1##2{%
7261   \@@gls@expand@field{##1}{symbolplural}{##2}%
7262 }%
7263 \do@newglossaryentry
7264 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7265 \let\gls@assign@plural\@org@gls@assign@plural
7266 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7267 }

```

ionAcronymStyle Option description is used, but not dua or footnote. Store long form in first key and short form in text and symbol key. The name is stored using \acronymformat to allow the user to override the way the name is displayed in the list of acronyms.

```

7268 \newcommand*{\SetDescriptionAcronymStyle}{%
7269   \renewcommand{\newacronym}[4][]{%
7270     \ifx\@glsacronymlists\@empty
7271       \def\@glo@type{\acronymtype}%
7272       \setkeys{glossentry}{##1}%
7273       \DeclareAcronymList{\@glo@type}%
7274       \SetDescriptionAcronymDisplayStyle{\@glo@type}%
7275     \fi
7276     \glskeylisttok{##1}%
7277     \glslabeltok{##2}%
7278     \glsshorttok{##3}%
7279     \glslongtok{##4}%
7280     \newacronymhook
7281     \DescriptionNewAcronymDef

```

7282 }%

Set display.

7283 \@for\@gls@type:=\@glsacronymlists\do{%

7284 \SetDescriptionAcronymDisplayStyle{\@gls@type}%

7285 }%

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.

7286 \ifglsacrsmallcaps

7287 \renewcommand{\acronymfont}[1]{\textsc{##1}}

7288 \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%

7289 \else

7290 \ifglsacrsmaller

7291 \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%

7292 \fi

7293 \fi

7294 }%

nymDisplayStyle Sets the acronym display style for given glossary with footnote setting (but not description or dua).

7295 \newcommand*{\SetFootnoteAcronymDisplayStyle}[1]{%

7296 \defglsentryfmt[#1]{%

7297 \ifdefempty\glscustomtext

7298 {%

Move the inserted text outside of \acronymfont

7299 \let\gls@org@insert\glsinsert

7300 \let\glsinsert\@empty

7301 \ifglsused{\glslabel}%

7302 {%

7303 \acronymfont{\glsgenentryfmt}\gls@org@insert

7304 }%

7305 {%

7306 \firstacronymfont{\glsgenentryfmt}\gls@org@insert

7307 \ifglschaslong{\glslabel}%

7308 {%

7309 \expandafter\protect\expandafter\acrfootnote\expandafter

7310 {\@gls@link@opts}{\@gls@link@label}%

7311 {%

7312 \glsifplural

7313 {\glsentrylongpl{\glslabel}}%

7314 {\glsentrylong{\glslabel}}%

7315 }%

7316 }%

7317 {}%

7318 }%

7319 }%

```

7320     {\glscustomtext\glsinsert}%
7321 }%
7322 }

```

teNewAcronymDef

```

7323 \newcommand*{\FootnoteNewAcronymDef}{%
7324   \edef\@do@newglossaryentry{%
7325     \noexpand\newglossaryentry{\the\glslabeltok}%
7326     {%
7327       type=\acronymtype,%
7328       name={\noexpand\acronymfont{\the\glsshorttok}},%
7329       sort={\the\glsshorttok},%
7330       text={\the\glsshorttok},%
7331       plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7332       first={\the\glsshorttok},%
7333       firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7334       short={\the\glsshorttok},%
7335       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7336       long={\the\glslongtok},%
7337       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7338       description={\the\glslongtok},%
7339       descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7340       \the\glskeylisttok
7341     }%
7342   }%
7343   \let\@org@gls@assign@plural\gls@assign@plural
7344   \let\@org@gls@assign@firstpl\gls@assign@firstpl
7345   \let\@org@gls@assign@descplural\gls@assign@descplural
7346   \def\gls@assign@firstpl##1##2{%
7347     \@@gls@expand@field{##1}{firstpl}{##2}%
7348   }%
7349   \def\gls@assign@plural##1##2{%
7350     \@@gls@expand@field{##1}{plural}{##2}%
7351   }%
7352   \def\gls@assign@descplural##1##2{%
7353     \@@gls@expand@field{##1}{descplural}{##2}%
7354   }%
7355   \@do@newglossaryentry
7356   \let\gls@assign@plural\@org@gls@assign@plural
7357   \let\gls@assign@firstpl\@org@gls@assign@firstpl
7358   \let\gls@assign@descplural\@org@gls@assign@descplural
7359 }

```

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.

```

7360 \newcommand*{\SetFootnoteAcronymStyle}{%
7361   \renewcommand{\newacronym}[4][\]{%
7362     \ifx\@glsacronymlists\@empty
7363       \def\@glo@type{\acronymtype}%

```

```

7364     \setkeys{glossentry}{##1}%
7365     \DeclareAcronymList{\@glo@type}%
7366     \SetFootnoteAcronymDisplayStyle{\@glo@type}%
7367     \fi
7368     \glskeylisttok{##1}%
7369     \glslabeltok{##2}%
7370     \glsshorttok{##3}%
7371     \glslongtok{##4}%
7372     \newacronymhook
7373     \FootnoteNewAcronymDef
7374 }%

```

Set display

```

7375 \@for\@gls@type:=\@glsacronymlists\do{%
7376     \SetFootnoteAcronymDisplayStyle{\@gls@type}%
7377 }%

```

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.

```

7378 \ifglsacrsmallcaps
7379     \renewcommand*{\acronymfont}[1]{\textsc{##1}}%
7380     \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7381 \else
7382     \ifglsacrsmaller
7383         \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
7384     \fi
7385 \fi

```

Check for option clash

```

7386 \ifglsacrdua
7387     \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’
7388         can’t both be set}{}%
7389 \fi
7390 }%

```

`\parenifnotempty` Do a space followed by the argument if the argument doesn't expand to empty or `\relax`. If argument isn't empty (or `\relax`), apply the macro to it given in the second argument.

```

7391 \DeclareRobustCommand*\glsdoparenifnotempty[2]{%
7392     \protected@edef\gls@tmp{##1}%
7393     \ifdefempty\gls@tmp
7394     {}%
7395     {%
7396         \ifx\gls@tmp\@gls@default@value
7397         \else
7398             \space (#2{##1})%
7399         \fi
7400     }%
7401 }

```


`\acronymDisplayStyle` Sets the acronym display style for given glossary where neither footnote nor description is required, but smallcaps or smaller specified.

```

7402 \newcommand*{\SetSmallAcronymDisplayStyle}[1]{%
7403   \def\glsentryfmt[#1]{%

7404     \ifdefempty\glscustomtext
7405     {%

      Move the inserted text outside of \acronymfont

7406       \let\gls@org@insert\glsinsert
7407       \let\glsinsert\@empty
7408       \ifglsused{\glslabel}%
7409       {%
7410         \acronymfont{\glsentryfmt}\gls@org@insert
7411       }%
7412       {%
7413         \glsentryfmt
7414         \ifgls hassymbol{\glslabel}%
7415         {%
7416           \glsifplural
7417           {%
7418             \def\@glo@symbol{\glsentrysymbolplural{\glslabel}}%
7419           }%
7420           {%
7421             \def\@glo@symbol{\glsentrysymbol{\glslabel}}%
7422           }%
7423           \space
7424           (\glscapscase
7425             {\firstacronymfont{\@glo@symbol}}%
7426             {\firstacronymfont{\@glo@symbol}}%
7427             {\firstacronymfont{\mfirstucMakeUppercase{\@glo@symbol}}})%
7428           }%
7429         }%
7430       }%
7431     }%
7432     {\glscustomtext\glsinsert}%
7433   }%
7434 }

```

`\allNewAcronymDef`

```

7435 \newcommand*{\SmallNewAcronymDef}{%
7436   \edef\@do@newglossaryentry{%
7437     \noexpand\newglossaryentry{\the\glslabeltok}%
7438     {%
7439       type=\acronymtype,%
7440       name={\noexpand\acronymfont{\the\glsshorttok}},%
7441       sort={\the\glsshorttok},%
7442       text={\the\glsshorttok},%

```

Default to the short plural.

```

7443 plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7444 first={\the\glslongtok},%
    Default to the long plural.
7445 firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7446 short={\the\glsshorttok},%
7447 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7448 long={\the\glslongtok},%
7449 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7450 description={\noexpand\@glo@first},%
7451 descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7452 symbol={\the\glsshorttok},%
    Default to the short plural.
7453 symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7454 \the\glskeylisttok
7455 }%
7456 }%
7457 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7458 \let\@org@gls@assign@plural\gls@assign@plural
7459 \let\@org@gls@assign@descplural\gls@assign@descplural
7460 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7461 \def\gls@assign@firstpl##1##2{%
7462   \@gls@expand@field{##1}{firstpl}{##2}%
7463 }%
7464 \def\gls@assign@plural##1##2{%
7465   \@gls@expand@field{##1}{plural}{##2}%
7466 }%
7467 \def\gls@assign@descplural##1##2{%
7468   \@gls@expand@field{##1}{descplural}{##2}%
7469 }%
7470 \def\gls@assign@symbolplural##1##2{%
7471   \@gls@expand@field{##1}{symbolplural}{##2}%
7472 }%
7473 \do@newglossaryentry
7474 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7475 \let\gls@assign@plural\@org@gls@assign@plural
7476 \let\gls@assign@descplural\@org@gls@assign@descplural
7477 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7478 }

```

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.

```

7479 \newcommand*{\SetSmallAcronymStyle}{%
7480   \renewcommand{\newacronym}[4][\]{%
7481     \ifx\@glsacronymlists\@empty
7482       \def\@glo@type{\acronymtype}%
7483       \setkeys{glossentry}{##1}%
7484       \DeclareAcronymList{\@glo@type}%
7485       \SetSmallAcronymDisplayStyle{\@glo@type}%

```

```

7486 \fi
7487 \glskeylisttok{##1}%
7488 \glslabeltok{##2}%
7489 \glsshorttok{##3}%
7490 \glslongtok{##4}%
7491 \newacronymhook
7492 \SmallNewAcronymDef
7493 }%

```

Change the display since first only contains long form.

```

7494 \@for\@gls@type:=\@glsacronymlists\do{%
7495 \SetSmallAcronymDisplayStyle{\@gls@type}%
7496 }%

```

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.

```

7497 \ifglsacrsmallcaps
7498 \renewcommand*\acronymfont[1]{\textsc{##1}}
7499 \renewcommand*\acrpluralsuffix{\glsupacrpluralsuffix}%
7500 \else
7501 \renewcommand*\acronymfont[1]{\textsmaller{##1}}
7502 \fi

```

check for option clash

```

7503 \ifglsacrdua
7504 \ifglsacrsmallcaps
7505 \PackageError{glossaries}{Option clash: ‘smallcaps’ and ‘dua’
7506 can’t both be set}{}%
7507 \else
7508 \PackageError{glossaries}{Option clash: ‘smaller’ and ‘dua’
7509 can’t both be set}{}%
7510 \fi
7511 \fi
7512 }%

```

DUADisplayStyle Sets the acronym display style for given glossary with dua setting.

```

7513 \newcommand*\SetDUADisplayStyle[1]{%
7514 \defglsentryfmt[1]{\glsentryfmt}%
7515 }

```

UANewAcronymDef

```

7516 \newcommand*\DUANewAcronymDef{%
7517 \edef\@do@newglossaryentry{%
7518 \noexpand\newglossaryentry{\the\glslabeltok}%
7519 {%
7520 type=\acronymtype,%
7521 name={\the\glsshorttok},%
7522 text={\the\glslongtok},%
7523 first={\the\glslongtok},%
7524 plural={\noexpand\expandonce\noexpand\@glo@longpl},%

```

```

7525     firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7526     short={\the\glsshorttok},%
7527     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7528     long={\the\glslongtok},%
7529     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7530     description={\the\glslongtok},%
7531     descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7532     symbol={\the\glsshorttok},%
7533     symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7534     \the\glskeylisttok
7535   }%
7536 }%
7537 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7538 \let\@org@gls@assign@plural\gls@assign@plural
7539 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7540 \let\@org@gls@assign@descplural\gls@assign@descplural
7541 \def\gls@assign@firstpl##1##2{%
7542   \@gls@expand@field{##1}{firstpl}{##2}%
7543 }%
7544 \def\gls@assign@plural##1##2{%
7545   \@gls@expand@field{##1}{plural}{##2}%
7546 }%
7547 \def\gls@assign@symbolplural##1##2{%
7548   \@gls@expand@field{##1}{symbolplural}{##2}%
7549 }%
7550 \def\gls@assign@descplural##1##2{%
7551   \@gls@expand@field{##1}{descplural}{##2}%
7552 }%
7553 \do@newglossaryentry
7554 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7555 \let\gls@assign@plural\@org@gls@assign@plural
7556 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7557 \let\gls@assign@descplural\@org@gls@assign@descplural
7558 }

```

\SetDUASStyle Always expand acronyms.

```

7559 \newcommand*{\SetDUASStyle}{%
7560   \renewcommand{\newacronym}[4][]{%
7561     \ifx\@glsacronymlists\@empty
7562       \def\@glo@type{\acronymtype}%
7563       \setkeys{glossentry}{##1}%
7564       \DeclareAcronymList{\@glo@type}%
7565       \SetDUADisplayStyle{\@glo@type}%
7566     \fi
7567     \glskeylisttok{##1}%
7568     \glslabeltok{##2}%
7569     \glsshorttok{##3}%
7570     \glslongtok{##4}%
7571     \newacronymhook

```

```

7572 \DUANewAcronymDef
7573 }%
    Set the display
7574 \@for\@gls@type:=\@glsacronymlists\do{%
7575 \SetDUADisplayStyle{\@gls@type}%
7576 }%
7577 }

```

SetAcronymStyle

```

7578 \newcommand*\SetAcronymStyle{%
7579 \SetDefaultAcronymStyle
7580 \ifglsacrdescription
7581 \ifglsacrfootnote
7582 \SetDescriptionFootnoteAcronymStyle
7583 \else
7584 \ifglsacrdua
7585 \SetDescriptionDUAAcronymStyle
7586 \else
7587 \SetDescriptionAcronymStyle
7588 \fi
7589 \fi
7590 \else
7591 \ifglsacrfootnote
7592 \SetFootnoteAcronymStyle
7593 \else
7594 \ifthenelse{\boolean{glsacrsmalldcaps}\OR
7595 \boolean{glsacrsmaller}}{%
7596 {%
7597 \SetSmallAcronymStyle
7598 }%
7599 }%
7600 \ifglsacrdua
7601 \SetDUASyle
7602 \fi
7603 }%
7604 \fi
7605 \fi
7606 }

```

Set the acronym style according to the package options

```
7607 \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.

`tomDisplayStyle` Sets the acronym display style.

```
7608 \newcommand*\SetCustomDisplayStyle}[1]{%
```

```

7609 \defglentryfmt[#1]{\glsgenentryfmt}%
7610 }

```

omAcronymFields

```

7611 \newcommand*{\CustomAcronymFields}{%
7612   name={\the\glsshorttok},%
7613   description={\the\glslongtok},%
7614   first={\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
7615   firstplural={\acrfullformat
7616     {\noexpand\glentrylongpl{\the\glslabeltok}}%
7617     {\noexpand\glentryshortpl{\the\glslabeltok}}},%

7618   text={\the\glsshorttok},%
7619   plural={\the\glsshorttok\noexpand\acrpluralsuffix}%
7620 }

```

omNewAcronymDef

```

7621 \newcommand*{\CustomNewAcronymDef}{%
7622   \protected@edef\@do@newglossaryentry{%
7623     \noexpand\newglossaryentry{\the\glslabeltok}%
7624     {%
7625       type=\acronymtype,%
7626       short={\the\glsshorttok},%
7627       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7628       long={\the\glslongtok},%
7629       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7630       user1={\the\glsshorttok},%
7631       user2={\the\glsshorttok\noexpand\acrpluralsuffix},%
7632       user3={\the\glslongtok},%
7633       user4={\the\glslongtok\noexpand\acrpluralsuffix},%
7634       \CustomAcronymFields,%
7635       \the\glskeylisttok
7636     }%
7637   }%
7638   \@do@newglossaryentry
7639 }

```

\SetCustomStyle

```

7640 \newcommand*{\SetCustomStyle}{%
7641   \renewcommand{\newacronym}[4][\{%
7642     \ifx\@glsacronymlists\@empty
7643       \def\@glo@type{\acronymtype}%
7644       \setkeys{glossentry}{##1}%
7645       \DeclareAcronymList{\@glo@type}%
7646       \SetCustomDisplayStyle{\@glo@type}%
7647     \fi
7648     \glskeylisttok{##1}%
7649     \glslabeltok{##2}%
7650     \glsshorttok{##3}%

```

```

7651 \glslongtok{##4}%
7652 \newacronymhook
7653 \CustomNewAcronymDef
7654 }%
Set the display
7655 \@for\@gls@type:=\@glsacronymlists\do{%
7656 \SetCustomDisplayStyle{\@gls@type}%
7657 }%
7658 }

```

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.

```
7659 \RequirePackage{glossary-hypermnav}
```

The styles that use list-like environments. These are not loaded if the nolist option is used:

```
7660 \@gls@loadlist
```

The styles that use the longtable environment. These are not loaded if the nolong package option is used.

```
7661 \@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.

```
7662 \@gls@loadsuper
```

The tree-like styles. These are not loaded if the notree package option is used.

```
7663 \@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.

```

7664 \ifx\@glossary@default@style\relax
7665 \else
7666 \setglossarystyle{\@glossary@default@style}
7667 \fi

```

1.20 Debugging Commands

\showgloparent `\showgloparent{\label}`

```

7668 \newcommand*{\showgloparent}[1]{%
7669 \expandafter\show\csname glo@\glsdetoklabel{#1}@parent\endcsname
7670 }

```

`\showglolevel` `\showglolevel{<label>}`

```
7671 \newcommand*{\showglolevel}[1]{%
7672   \expandafter\show\csname glo@glstetoklabel{#1}@level\endcsname
7673 }
```

`\showglotext` `\showglotext{<label>}`

```
7674 \newcommand*{\showglotext}[1]{%
7675   \expandafter\show\csname glo@glstetoklabel{#1}@text\endcsname
7676 }
```

`\showgloplural` `\showgloplural{<label>}`

```
7677 \newcommand*{\showgloplural}[1]{%
7678   \expandafter\show\csname glo@glstetoklabel{#1}@plural\endcsname
7679 }
```

`\showglofirst` `\showglofirst{<label>}`

```
7680 \newcommand*{\showglofirst}[1]{%
7681   \expandafter\show\csname glo@glstetoklabel{#1}@first\endcsname
7682 }
```

`\showglofirstpl` `\showglofirstpl{<label>}`

```
7683 \newcommand*{\showglofirstpl}[1]{%
7684   \expandafter\show\csname glo@glstetoklabel{#1}@firstpl\endcsname
7685 }
```

`\showglotype` `\showglotype{<label>}`

```
7686 \newcommand*{\showglotype}[1]{%
7687   \expandafter\show\csname glo@glstetoklabel{#1}@type\endcsname
7688 }
```


`\showglocounter` `\showglocounter{<label>}`

```
7689 \newcommand*{\showglocounter}[1]{%
7690   \expandafter\show\csname glo@glstdetoklabel{#1}@counter\endcsname
7691 }
```

`\showglouser` `\showglouser{<label>}`

```
7692 \newcommand*{\showglouser}[1]{%
7693   \expandafter\show\csname glo@glstdetoklabel{#1}@useri\endcsname
7694 }
```

`\showglouserii` `\showglouserii{<label>}`

```
7695 \newcommand*{\showglouserii}[1]{%
7696   \expandafter\show\csname glo@glstdetoklabel{#1}@userii\endcsname
7697 }
```

`\showglouseriii` `\showglouseriii{<label>}`

```
7698 \newcommand*{\showglouseriii}[1]{%
7699   \expandafter\show\csname glo@glstdetoklabel{#1}@useriii\endcsname
7700 }
```

`\showglouseriv` `\showglouseriv{<label>}`

```
7701 \newcommand*{\showglouseriv}[1]{%
7702   \expandafter\show\csname glo@glstdetoklabel{#1}@useriv\endcsname
7703 }
```

`\showglouserv` `\showglouserv{<label>}`

```
7704 \newcommand*{\showglouserv}[1]{%
7705   \expandafter\show\csname glo@glstdetoklabel{#1}@userv\endcsname
7706 }
```

\showglouservi \showglouservi{<label>}

```
7707 \newcommand*{\showglouservi}[1]{%
7708   \expandafter\show\csname glo@glstdetoklabel{#1}@uservi\endcsname
7709 }
```

\showgloname \showgloname{<label>}

```
7710 \newcommand*{\showgloname}[1]{%
7711   \expandafter\show\csname glo@glstdetoklabel{#1}@name\endcsname
7712 }
```

\showglodesc \showglodesc{<label>}

```
7713 \newcommand*{\showglodesc}[1]{%
7714   \expandafter\show\csname glo@glstdetoklabel{#1}@desc\endcsname
7715 }
```

howglodescplural \showglodescplural{<label>}

```
7716 \newcommand*{\showglodescplural}[1]{%
7717   \expandafter\show\csname glo@glstdetoklabel{#1}@descplural\endcsname
7718 }
```

\showglosort \showglosort{<label>}

```
7719 \newcommand*{\showglosort}[1]{%
7720   \expandafter\show\csname glo@glstdetoklabel{#1}@sort\endcsname
7721 }
```

\showglosymbol \showglosymbol{<label>}

```
7722 \newcommand*{\showglosymbol}[1]{%
7723   \expandafter\show\csname glo@glstdetoklabel{#1}@symbol\endcsname
7724 }
```

wglosymbolplural `\showglosymbolplural{<label>}`

```
7725 \newcommand*{\showglosymbolplural}[1]{%  
7726   \expandafter\show\csname glo@glstetoklabel{#1}@symbolplural\endcsname  
7727 }
```

\showgloshort `\showgloshort{<label>}`

```
7728 \newcommand*{\showgloshort}[1]{%  
7729   \expandafter\show\csname glo@glstetoklabel{#1}@short\endcsname  
7730 }
```

\showglolong `\showglolong{<label>}`

```
7731 \newcommand*{\showglolong}[1]{%  
7732   \expandafter\show\csname glo@glstetoklabel{#1}@long\endcsname  
7733 }
```

\showgloindex `\showgloindex{<label>}`

```
7734 \newcommand*{\showgloindex}[1]{%  
7735   \expandafter\show\csname glo@glstetoklabel{#1}@index\endcsname  
7736 }
```

\showgloflag `\showgloflag{<label>}`

```
7737 \newcommand*{\showgloflag}[1]{%  
7738   \expandafter\show\csname ifglo@glstetoklabel{#1}@flag\endcsname  
7739 }
```

\showgloloclist `\showgloloclist{<label>}`

```
7740 \newcommand*{\showgloloclist}[1]{%  
7741   \expandafter\show\csname glo@glstetoklabel{#1}@loclist\endcsname  
7742 }
```

`\showglofield` `\showglofield{<label>}{<field>}`

```
7743 \newcommand*{\showglofield}[2]{%
7744   \csshow{glo@glstetoklabel{#1}@#2}%
7745 }
```

`showacronymlists` `\showacronymlists`

Show list of glossaries that have been flagged as a list of acronyms.

```
7746 \newcommand*{\showacronymlists}{%
7747   \show\@glsacronymlists
7748 }
```

`\showglossaries` `\showglossaries`

Show list of defined glossaries.

```
7749 \newcommand*{\showglossaries}{%
7750   \show\@glo@types
7751 }
```

`\showglossaryin` `\showglossaryin{<glossary-label>}`

Show the ‘in’ extension for the given glossary.

```
7752 \newcommand*{\showglossaryin}[1]{%
7753   \expandafter\show\csname @glotype@#1@in\endcsname
7754 }
```

`\showglossaryout` `\showglossaryout{<glossary-label>}`

Show the ‘out’ extension for the given glossary.

```
7755 \newcommand*{\showglossaryout}[1]{%
7756   \expandafter\show\csname @glotype@#1@out\endcsname
7757 }
```

`showglossarytitle` `\showglossarytitle{<glossary-label>}`

Show the title for the given glossary.

```
7758 \newcommand*{\showglossarytitle}[1]{%
7759   \expandafter\show\csname @glotype@#1@title\endcsname
7760 }
```

wglossarycounter `\showglossarycounter{<glossary-label>}`

Show the counter for the given glossary.

```
7761 \newcommand*{\showglossarycounter}[1]{%
7762   \expandafter\show\csname @glotype@#1@counter\endcsname
7763 }
```

wglossaryentries `\showglossaryentries{<glossary-label>}`

Show the list of entry labels for the given glossary.

```
7764 \newcommand*{\showglossaryentries}[1]{%
7765   \expandafter\show\csname glolist@#1\endcsname
7766 }
```

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.

```
7767 \csname ifglcompatible-2.07\endcsname
7768   \RequirePackage{glossaries-compatible-207}
7769 \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.

```
7770 \NeedsTeXFormat{LaTeX2e}
```

```
7771 \ProvidesPackage{glossaries-prefix}[2016/12/16 v4.27 (NLCT)]
```

Pass all options to glossaries:

```
7772 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
7773 \ProcessOptions
```

Load glossaries:

```
7774 \RequirePackage{glossaries}
```

Add the new keys:

```
7775 \define@key{glossentry}{prefixfirst}{\def\@glo@entryprefixfirst{#1}}%
```

```
7776 \define@key{glossentry}{prefixfirstplural}{\def\@glo@entryprefixfirstplural{#1}}%
```

```
7777 \define@key{glossentry}{prefix}{\def\@glo@entryprefix{#1}}%
```

```
7778 \define@key{glossentry}{prefixplural}{\def\@glo@entryprefixplural{#1}}%
```

Add them to `\@gls@keymap`:

```
7779 \appto\@gls@keymap{,%
```

```
7780   {prefixfirst}{prefixfirst},%
```

```
7781   {prefixfirstplural}{prefixfirstplural},%
```

```
7782   {prefix}{prefix},%
```

```
7783   {prefixplural}{prefixplural}}%
```

```
7784 }
```

Set the default values:

```
7785 \appto\@newglossaryentryprehook{%
```

```
7786   \def\@glo@entryprefix{}%
```

```
7787   \def\@glo@entryprefixplural{}%
```

```
7788   \let\@glo@entryprefixfirst\@gls@default@value
```

```
7789   \let\@glo@entryprefixfirstplural\@gls@default@value
```

```
7790 }
```

Set the assignment code:

```
7791 \appto\@newglossaryentryposthook{%
```

```
7792   \gls@assign@field{}{\@glo@label}{prefix}{\@glo@entryprefix}%
```

```
7793   \gls@assign@field{}{\@glo@label}{prefixplural}{\@glo@entryprefixplural}%
```

If `prefixfirst` has not been supplied, make it the same as `prefix`.

```
7794 \expandafter\gls@assign@field\expandafter
```

```
7795   {\csname glo@\@glo@label @prefix\endcsname}{\@glo@label}{prefixfirst}%
```

```
7796   {\@glo@entryprefixfirst}}%
```

If prefixfirstplural has not been supplied, make it the same as prefixplural.

```

7797 \expandafter\gls@assign@field\expandafter
7798   {\csname glo@\@glo@label @prefixplural\endcsname}{\@glo@label}%
7799   {prefixfirstplural}{\@glo@entryprefixfirstplural}%
7800 }

```

Define commands to access these fields:

entryprefixfirst

```

7801 \newcommand*{\glsentryprefixfirst}[1]{\csuse{glo@#1@prefixfirst}}

```

entryfirstplural

```

7802 \newcommand*{\glsentryprefixfirstplural}[1]{\csuse{glo@#1@prefixfirstplural}}

```

\glsentryprefix

```

7803 \newcommand*{\glsentryprefix}[1]{\csuse{glo@#1@prefix}}

```

entryprefixplural

```

7804 \newcommand*{\glsentryprefixplural}[1]{\csuse{glo@#1@prefixplural}}

```

Now for the initial upper case variants:

entryprefixfirst

```

7805 \newrobustcmd*{\Glsentryprefixfirst}[1]{%
7806   \protected@edef\@glo@text{\csname glo@#1@prefixfirst\endcsname}%
7807   \xmakefirstuc\@glo@text
7808 }

```

entryfirstplural

```

7809 \newrobustcmd*{\Glsentryprefixfirstplural}[1]{%
7810   \protected@edef\@glo@text{\csname glo@#1@prefixfirstplural\endcsname}%
7811   \xmakefirstuc\@glo@text
7812 }

```

\Glsentryprefix

```

7813 \newrobustcmd*{\Glsentryprefix}[1]{%
7814   \protected@edef\@glo@text{\csname glo@#1@prefix\endcsname}%
7815   \xmakefirstuc\@glo@text
7816 }

```

entryprefixplural

```

7817 \newrobustcmd*{\Glsentryprefixplural}[1]{%
7818   \protected@edef\@glo@text{\csname glo@#1@prefixplural\endcsname}%
7819   \xmakefirstuc\@glo@text
7820 }

```

Define commands to determine if the prefix keys have been set:

\ifglshasprefix

```
7821 \newcommand*{\ifglshasprefix}[3]{%
7822   \ifcempty{glo@#1@prefix}%
7823   {#3}%
7824   {#2}%
7825 }
```

hasprefixplural

```
7826 \newcommand*{\ifglshasprefixplural}[3]{%
7827   \ifcempty{glo@#1@prefixplural}%
7828   {#3}%
7829   {#2}%
7830 }
```

shasprefixfirst

```
7831 \newcommand*{\ifglshasprefixfirst}[3]{%
7832   \ifcempty{glo@#1@prefixfirst}%
7833   {#3}%
7834   {#2}%
7835 }
```

efixfirstplural

```
7836 \newcommand*{\ifglshasprefixfirstplural}[3]{%
7837   \ifcempty{glo@#1@prefixfirstplural}%
7838   {#3}%
7839   {#2}%
7840 }
```

Define commands that insert the prefix before commands like \gls:

\pgls

```
7841 \newrobustcmd{\pgls}{\@gls@hyp@opt\@pgls}
```

\@pgls Unstarred version.

```
7842 \newcommand*{\@pgls}[2][ ]{%
7843   \new@ifnextchar[%
7844   {\@pgls@{#1}{#2}}%
7845   {\@pgls@{#1}{#2}[ ]}%
7846 }
```

\@pgls@ Read in the final optional argument:

```
7847 \def\@pgls@#1#2[#3]{%
7848   \glsdoifexists{#2}%
7849   {%
7850     \ifglsused{#2}%
7851     {%
7852       \glsentryprefix{#2}%
7853     }%

```



```

7854     {%
7855     \glsentryprefixfirst{#2}%
7856     }%
7857     \@gls@{#1}{#2}[#3]%
7858     }%
7859 }

```

Similarly for the plural version:

```

\pglsp1
7860 \newrobustcmd{\pglsp1}{\@gls@hyp@opt\@pglsp1}

```

\@pglsp1 Unstarred version.

```

7861 \newcommand*{\@pglsp1}[2][{}]{%
7862   \new@ifnextchar[%
7863   {\@pglsp1@{#1}{#2}}%
7864   {\@pglsp1@{#1}{#2}[]}%
7865 }

```

\@pglsp1@ Read in the final optional argument:

```

7866 \def\@pglsp1@#1#2[#3]{%
7867   \glsdoifexists{#2}%
7868   {%
7869     \ifglsused{#2}%
7870     {%
7871       \glsentryprefixplural{#2}%
7872     }%
7873     {%
7874       \glsentryprefixfirstplural{#2}%
7875     }%
7876     \@glspl@{#1}{#2}[#3]%
7877   }%
7878 }

```

Now for the first letter upper case versions:

```

\Pgls
7879 \newrobustcmd{\Pgls}{\@gls@hyp@opt\@Pgls}

```

\@Pgls Unstarred version.

```

7880 \newcommand*{\@Pgls}[2][{}]{%
7881   \new@ifnextchar[%
7882   {\@Pgls@{#1}{#2}}%
7883   {\@Pgls@{#1}{#2}[]}%
7884 }

```

\@Pgls@ Read in the final optional argument:

```

7885 \def\@Pgls@#1#2[#3]{%

```

```

7886 \glsdoifexists{#2}%
7887 {%
7888   \ifglsused{#2}%
7889   {%
7890     \ifglshasprefix{#2}%
7891     {%
7892       \Glsentryprefix{#2}%
7893       \@gls@{#1}{#2}[#3]%
7894     }%
7895     {\@Gls@{#1}{#2}[#3]}%
7896   }%
7897   {%
7898     \ifglshasprefixfirst{#2}%
7899     {%
7900       \Glsentryprefixfirst{#2}%
7901       \@gls@{#1}{#2}[#3]%
7902     }%
7903     {\@Gls@{#1}{#2}[#3]}%
7904   }%
7905 }%
7906 }

```

Similarly for the plural version:

```

\Pglspl
7907 \newrobustcmd{\Pglspl}{\@gls@hyp@opt\@Pglspl}

```

\@Pglspl Unstarred version.

```

7908 \newcommand*{\@Pglspl}[2] [] {%
7909   \new@ifnextchar[%
7910   {\@Pglspl@{#1}{#2}}%
7911   {\@Pglspl@{#1}{#2} []}%
7912 }

```

\@Pglspl@ Read in the final optional argument:

```

7913 \def\@Pglspl@#1#2[#3] {%
7914   \glsdoifexists{#2}%
7915   {%
7916     \ifglsused{#2}%
7917     {%
7918       \ifglshasprefixplural{#2}%
7919       {%
7920         \Glsentryprefixplural{#2}%
7921         \@glspl@{#1}{#2}[#3]%
7922       }%
7923       {\@Glspl@{#1}{#2}[#3]}%
7924     }%
7925     {%
7926       \ifglshasprefixfirstplural{#2}%

```

```

7927      {%
7928      \Glsentryprefixfirstplural{#2}%
7929      \@glsp1@{#1}{#2}[#3]%
7930      }%
7931      {\@Glspl@{#1}{#2}[#3]}%
7932      }%
7933  }%
7934 }

```

Finally the all upper case versions:

\PGLS

```

7935 \newrobustcmd{\PGLS}{\@gls@hyp@opt\PGLS}

```

\@PGLS Unstarred version.

```

7936 \newcommand*{\@PGLS}[2][{}]{%
7937   \new@ifnextchar[%
7938   {\@PGLS@{#1}{#2}}%
7939   {\@PGLS@{#1}{#2}[]}%
7940 }

```

\@PGLS@ Read in the final optional argument:

```

7941 \def\@PGLS@#1#2[#3]{%
7942   \glsdoifexists{#2}%
7943   {%
7944     \ifglsused{#2}%
7945     {%
7946       \mfirstucMakeUppercase{\glsentryprefix{#2}}%
7947     }%
7948     {%
7949       \mfirstucMakeUppercase{\glsentryprefixfirst{#2}}%
7950     }%
7951     \@GLS@{#1}{#2}[#3]%
7952   }%
7953 }

```

Plural version:

\PGLSp1

```

7954 \newrobustcmd{\PGLSp1}{\@gls@hyp@opt\PGLSp1}

```

\@PGLSp1 Unstarred version.

```

7955 \newcommand*{\@PGLSp1}[2][{}]{%
7956   \new@ifnextchar[%
7957   {\@PGLSp1@{#1}{#2}}%
7958   {\@PGLSp1@{#1}{#2}[]}%
7959 }

```

\@PGLSp1@ Read in the final optional argument:

```
7960 \def\@PGLSp1@#1#2[#3]{%
7961   \glsdoifexists{#2}%
7962   {%
7963     \ifglsused{#2}%
7964     {%
7965       \mfirstucMakeUppercase{\glsentryprefixplural{#2}}%
7966     }%
7967     {%
7968       \mfirstucMakeUppercase{\glsentryprefixfirstplural{#2}}%
7969     }%
7970     \@GLSp1@{#1}{#2}[#3]%
7971   }%
7972 }
```

3 Glossary Styles

3.1 Glossary hyper-navigation definitions (glossary-hypernav package)

Package Definition:

```
7973 \ProvidesPackage{glossary-hypernav}[2016/12/16 v4.27 (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`

```
7974 \newcommand*{\glsnavhyperlink}[3][\@glo@type]{%
7975   \edef\gls@grplabel{#2}\protected@edef\gls@grptitle{#3}%
7976   \@glslink{glsn:#1@#2}{#3}}
```

```
\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`

```
7977 \newcommand*{\glsnavhypertarget}[3][\@glo@type]{%
  Add this group to the aux file for re-run check.
7978   \protected@write\auxout{}{\string\@gls@hypergroup{#1}{#2}}%
  Add the target.
7979   \@glstarget{glsn:#1@#2}{#3}%
  Check list of know groups to determine if a re-run is required.
7980   \expandafter\let
7981     \expandafter\@gls@list\csname @gls@hypergroup@#1\endcsname
  Iterate through list and terminate loop if this group is found.
7982   \@for\@gls@elem:=\@gls@list\do{%
7983     \ifthenelse{equal{\@gls@elem}{#2}}{\@endfortrue}{}}%
```

Check if list terminated prematurely.

```
7984 \if@endfor
7985 \else
```

This group was not included in the list, so issue a warning.

```
7986 \GlossariesWarningNoLine{Navigation panel
7987     for glossary type ‘#1’^^Jmissing group ‘#2’}%
7988 \gdef\gls@hypergrouprerun{%
7989     \GlossariesWarningNoLine{Navigation panel
7990     has changed. Rerun LaTeX}}%
7991 \fi
7992 }
```

hypergrouprerun Give a warning at the end if re-run required

```
7993 \let\gls@hypergrouprerun\relax
7994 \AtEndDocument{\gls@hypergrouprerun}
```

@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.

```
7995 \newcommand*{\@gls@hypergroup}[2]{%
7996 \@ifundefined{gls@hypergrouplist@#1}{%
7997     \expandafter\xdef\csname @gls@hypergrouplist@#1\endcsname{#2}%
7998 }{%
7999     \expandafter\let\expandafter\@gls@tmp
8000         \csname @gls@hypergrouplist@#1\endcsname
8001     \expandafter\xdef\csname @gls@hypergrouplist@#1\endcsname{%
8002         \@gls@tmp,#2}%
8003 }%
8004 }
```

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

```
8005 \newcommand*{\glsnavigation}{%
8006     \def\@gls@between{}%
8007     \ifcsundef{gls@hypergrouplist@\@glo@type}%
8008     {%
8009         \def\@gls@list{}%
8010     }%
8011     {%
8012         \expandafter\let\expandafter\@gls@list
8013         \csname @gls@hypergrouplist@\@glo@type\endcsname
8014     }%
8015     \@for\@gls@tmp:=\@gls@list\do{%
```

```

8016 \gls@between

8017 \gls@getgrouptitle{\gls@tmp}{\gls@grptitle}%
8018 \glsnavhyperlink{\gls@tmp}{\gls@grptitle}%
8019 \let\gls@between\glshypernavsep
8020 }%
8021 }

```

`\glshypernavsep` Separator for the hyper navigation bar.

```

8022 \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`

```

8023 \newcommand*{\glssymbolnav}{%
8024 \glsnavhyperlink{glssymbols}{\glsgetgrouptitle{glssymbols}}%
8025 \glshypernavsep
8026 \glsnavhyperlink{glsnumbers}{\glsgetgrouptitle{glsnumbers}}%
8027 \glshypernavsep
8028 }

```

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.

```

8029 \ProvidesPackage{glossary-inline}[2016/12/16 v4.27 (NLCT)]

```

`inline` Define the inline style.

```

8030 \newglossarystyle{inline}{%
    Start of glossary sets up first empty separator between entries. (This is then changed by
    \glossentry)
8031 \renewenvironment{theglossary}%
8032 {%
8033 \def\gls@inlinesep{}%
8034 \def\gls@inlinesubsep{}%
8035 \def\gls@inlinepostchild{}%
8036 }%
8037 {\glspostinline}%

    No header:
8038 \renewcommand*{\glossaryheader}{}%

    No group headings (if heading is required, add \glsinlinedopostchild to start definition
    in case heading follows a child entry):
8039 \renewcommand*{\glsgroupheading}[1]{}%

```

Just display separator followed by name and description:

```

8040 \renewcommand{\glossentry}[2]{%
8041   \glsinlinedopostchild
8042   \gls@inlinesep
8043   \glsentryitem{##1}%
8044   \glsinlinenameformat{##1}{%
8045     \glossentryname{##1}%
8046   }%
8047   \ifglshasdescsuppressed{##1}%
8048   {%
8049     \glsinlineemptydescformat
8050     {%
8051       \glossentrysymbol{##1}%
8052     }%
8053     {%
8054       ##2%
8055     }%
8056   }%
8057   {%
8058     \ifglshasdesc{##1}%
8059     {\glsinlinedescformat{\glossentrydesc{##1}}{\glossentrysymbol{##1}}{##2}}%
8060     {\glsinlineemptydescformat{\glossentrysymbol{##1}}{##2}}%
8061   }%
8062   \ifglshaschildren{##1}%
8063   {%
8064     \glsresetsubentrycounter
8065     \glsinlineparentchildseparator
8066     \def\gls@inlinesubsep{%
8067       \def\gls@inlinepostchild{\glsinlinepostchild}%
8068     }%
8069     {}%
8070   \def\gls@inlinesep{\glsinlineseparator}%
8071 }%
```

Sub-entries display description:

```

8072 \renewcommand{\subglossentry}[3]{%
8073   \gls@inlinesubsep%
8074   \glsinlinesubnameformat{##2}{%
8075     \glossentryname{##2}%
8076   }%
8077   \glsinlinesubdescformat{\glossentrydesc{##2}}{\glossentrysymbol{##2}}{##3}%
8078   \def\gls@inlinesubsep{\glsinlinesubseparator}%
8079 }%
```

Nothing special between groups:

```

8080 \renewcommand*{\glsgroupskip}{}%
8081 }
```

linedopostchild

```

8082 \newcommand*{\glsinlinedopostchild}{%
```



```

8083 \gls@inlinepostchild
8084 \def\gls@inlinepostchild{}%
8085 }

```

`inlineseparator` Separator to use between entries.

```
8086 \newcommand*\glsinlineseparator{}\space
```

`inlinesubseparator` Separator to use between sub-entries.

```
8087 \newcommand*\glsinlinesubseparator{}\space
```

`parentchildseparator` Separator to use between parent and children.

```
8088 \newcommand*\glsinlineparentchildseparator{}\space
```

`inlinepostchild` Hook to use between child and next entry

```
8089 \newcommand*\glsinlinepostchild{}
```

`\glspostinline` Terminator for inline glossary.

```
8090 \newcommand*\glspostinline{\glspostdescription\space}
```

`inlinenameformat` Formats the name of the entry (first argument label, second argument name):

```
8091 \newcommand*\glsinlinenameformat}[2]{\glstarget{#1}{#2}}
```

`inlinedescformat` Formats the entry's description, symbol and location list:

```
8092 \newcommand*\glsinlinedescformat}[3]{\space#1}
```

`emptydescformat` Formats the entry's symbol and location list when the description is empty:

```
8093 \newcommand*\glsinlineemptydescformat}[2]{}
```

`inlinesubnameformat` Formats the name of the subentry (first argument label, second argument name):

```
8094 \newcommand*\glsinlinesubnameformat}[2]{\glstarget{#1}{}}
```

`inlinesubdescformat` Formats the subentry's description, symbol and location list:

```
8095 \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.

```
8096 \ProvidesPackage{glossary-list}[2016/12/16 v4.27 (NLCT)]
```

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```

8097 \providecommand{\indexspace}{%
8098 \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
8099 }

```

tgrouphaderfmt Provide a way of adjusting the format of the group headings.

```
8100 \newcommand*{\glslistgrouphaderfmt}[1]{#1}
```

tnavigationitem Provide a way of adjusting the format of the navigation header. This puts the navigation line inside the optional argument of item to prevent unwanted space occurring at the start, but this can cause a problem if the navigation line is too long. With this command, it makes it easier for the user to customise the style without having to remember to modify `\glossaryheader` after the style has been set.

```
8101 \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.

```
8102 \newglossarystyle{list}{%
  Use description environment:
8103   \renewenvironment{theglossary}%
8104     {\begin{description}}{\end{description}}%
  No header at the start of the environment:
8105   \renewcommand*{\glossaryheader}{}%
  No group headings:
8106   \renewcommand*{\glsgroupheading}[1]{}%
  Main (level 0) entries start a new item in the list:
8107   \renewcommand*{\glossentry}[2]{%
8108     \item[\glsentryitem{##1}]%
8109       \glstarget{##1}{\glossentryname{##1}}]
8110     \glossentrydesc{##1}\glspostdescription\space ##2}%
  Sub-entries continue on the same line:
8111   \renewcommand*{\subglossentry}[3]{%
8112     \glssubentryitem{##2}%
8113     \glstarget{##2}{\strut}\space
8114     \glossentrydesc{##2}\glspostdescription\space ##3.}%
8115 %   \end{macrocode}
8116 % Add vertical space between groups:
8117 %\changes{3.03}{2012/09/21}{added check for glsnogroupskip}
8118 %   \begin{macrocode}
8119   \renewcommand*{\glsgroupskip}{\ifglslsnogroupskip\else\indexspace\fi}%
8120 }
```

listgroup The listgroup style is like the list style, but the glossary groups have headings.

```
8121 \newglossarystyle{listgroup}{%
  Base it on the list style:
8122   \setglossarystyle{list}%

```

Each group has a heading:

```
8123 \renewcommand*{\glsgroupheading}[1]{%
8124   \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.

```
8125 \newglossarystyle{listhypergroup}{%
```

Base it on the list style:

```
8126 \setglossarystyle{list}%
```

Add navigation links at the start of the environment.

```
8127 \renewcommand*{\glossaryheader}{%
8128   \glslstnavigationitem{\glslnavigation}}%
```

Each group has a heading with a hypertarget:

```
8129 \renewcommand*{\glsgroupheading}[1]{%
8130   \item[\glslistgroupheaderfmt
8131         {\glslnavhypertarget{##1}{\glsgrouptitle{##1}}}]}
```

altlist The altlist glossary style is like the list style, but places the description on a new line. Sub-entries follow in separate paragraphs without the sub-entry name. This style does not use the entry's symbol.

```
8132 \newglossarystyle{altlist}{%
```

Base it on the list style:

```
8133 \setglossarystyle{list}%
```

Main (level 0) entries start a new item in the list with a line break after the entry name:

```
8134 \renewcommand*{\glossentry}[2]{%
8135   \item[\glssentryitem{##1}%
8136         \glstarget{##1}{\glossentryname{##1}}]}%
```

Version 3.04 changed \newline to the following paragraph break stuff (thanks to Daniel Gebhardt for supplying the fix) to prevent a page break occurring at this point.

```
8137   \mbox{} \par \nobreak \@afterheading
8138   \glossentrydesc{##1} \glspostdescription \space ##2}%
```

Sub-entries start a new paragraph:

```
8139 \renewcommand{\subglossentry}[3]{%
8140   \par
8141   \glssubentryitem{##2}%
8142   \glstarget{##2}{\strut} \glossentrydesc{##2} \glspostdescription \space ##3}%
8143 }
```

altlistgroup The altlistgroup glossary style is like the altlist style, but the glossary groups have headings.

```
8144 \newglossarystyle{altlistgroup}{%
```

Base it on the altlist style:

```
8145 \setglossarystyle{altlist}%
```

Each group has a heading:

```
8146 \renewcommand*{\glsgroupheading}[1]{%
8147 \item[\glslistgroupheaderfmt{\glsgetgrouptitle{##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.

```
8148 \newglossarystyle{altlisthypergroup}{%
```

Base it on the `altlist` style:

```
8149 \setglossarystyle{altlist}%
```

Add navigation links at the start of the environment.

```
8150 \renewcommand*{\glossaryheader}{%
8151 \glslistnavigationitem{\glsnavigation}}%
```

Each group has a heading with a `hypertarget`:

```
8152 \renewcommand*{\glsgroupheading}[1]{%
8153 \item[\glslistgroupheaderfmt
8154 {\glsnavhypertarget{##1}{\glsgetgrouptitle{##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.

```
8155 \newglossarystyle{listdotted}{%
```

Base it on the `list` style:

```
8156 \setglossarystyle{list}%
```

Each main (level 0) entry starts a new item:

```
8157 \renewcommand*{\glossentry}[2]{%
8158 \item[]\makebox[\glslistdottedwidth][l]{%
8159 \glsentryitem{##1}%
8160 \glstarget{##1}{\glossentryname{##1}}%
8161 \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##1}}%
```

Sub entries have the same format as main entries:

```
8162 \renewcommand*{\subglossentry}[3]{%
8163 \item[]\makebox[\glslistdottedwidth][l]{%
8164 \glssubentryitem{##2}%
8165 \glstarget{##2}{\glossentryname{##2}}%
8166 \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##2}}%
8167 }
```

`listdottedwidth`

```
8168 \newlength\glslistdottedwidth
8169 \setlength{\glslistdottedwidth}{.5\hsize}
```

`sublistdotted` This style is similar to the `glostylelistdotted` style, except that the main entries just have the name displayed.

```
8170 \newglossarystyle{sublistdotted}{%
```

Base it on the listdotted style:

```
8171 \setglossarystyle{listdotted}%
```

Main (level 0) entries just display the name:

```
8172 \renewcommand*{\glossentry}[2]{%
8173   \item[\glentryitem{##1}\glstarget{##1}{\glossentryname{##1}}}%
8174 }
```

3.4 Glossary Styles using longtable (the glossary-long package)

The glossary styles defined in the package used the longtable environment in the glossary.

```
8175 \ProvidesPackage{glossary-long}[2016/12/16 v4.27 (NLCT)]
```

Requires the package:

```
8176 \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`.)

```
8177 \@ifundefined{glsdescwidth}{%
8178   \newlength{glsdescwidth}
8179   \setlength{glsdescwidth}{0.6\hsize}
8180 }{}
```

`\glspagelistwidth` This is a length that governs the width of the page list column.

```
8181 \@ifundefined{glspagelistwidth}{%
8182   \newlength{glspagelistwidth}
8183   \setlength{glspagelistwidth}{0.1\hsize}
8184 }{}
```

`long` The long glossary style command which uses the longtable environment:

```
8185 \newglossarystyle{long}{%
```

Use longtable with two columns:

```
8186 \renewenvironment{theglossary}%
8187   {\begin{longtable}[lp{glsdescwidth}]}%
8188   {\end{longtable}}%
```

Do nothing at the start of the environment:

```
8189 \renewcommand*{\glossaryheader}{}%
```

No heading between groups:

```
8190 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries displayed in a row:

```
8191 \renewcommand{\glossentry}[2]{%
8192   \glentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8193   \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline
8194   }%
```

Sub entries displayed on the following row without the name:

```
8195 \renewcommand{\subglossentry}[3]{%
8196     &
8197     \glssubentryitem{##2}%
8198     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
8199     ##3\tabularnewline
8200 }%
```

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>)

```
8201 \ifglsgroupskip
8202 \renewcommand*{\glsgroupskip}{}%
8203 \else
8204 \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
8205 \fi
8206 }
```

longborder The longborder style is like the above, but with horizontal and vertical lines:

```
8207 \newglossarystyle{longborder}{%
```

Base it on the glostylelong style:

```
8208 \setglossarystyle{long}%
```

Use longtable with two columns with vertical lines between each column:

```
8209 \renewenvironment{theglossary}{%
8210 \begin{longtable}{|l|p{\glsgdescwidth}|}{\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
8211 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8212 }
```

longheader The longheader style is like the long style but with a header:

```
8213 \newglossarystyle{longheader}{%
```

Base it on the glostylelong style:

```
8214 \setglossarystyle{long}%
```

Set the table's header:

```
8215 \renewcommand*{\glossaryheader}{%
8216 \bfseries \entryname & \bfseries \descriptionname\tabularnewline\endhead}%
8217 }
```

longheaderborder The longheaderborder style is like the long style but with a header and border:

```
8218 \newglossarystyle{longheaderborder}{%
```

Base it on the glostylelongborder style:

```
8219 \setglossarystyle{longborder}%
```

Set the table's header and add horizontal line to table's foot:

```
8220 \renewcommand*{\glossaryheader}{%
8221 \hline\bfseries \entryname & \bfseries
8222 \descriptionname\tabularnewline\hline
```

```

8223 \endhead
8224 \hline\endfoot}%
8225 }

```

long3col The long3col style is like long but with 3 columns

```

8226 \newglossarystyle{long3col}{%
    Use a longtable with 3 columns:
8227 \renewenvironment{theglossary}%
8228 {\begin{longtable}{lp{\glstdescwidth}p{\glspagelistwidth}}}%
8229 {\end{longtable}}%

```

No table header:

```

8230 \renewcommand*\glossaryheader{}%

```

No headings between groups:

```

8231 \renewcommand*\glsgroupheading[1]{}%

```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```

8232 \renewcommand{\glossentry}[2]{%
8233 \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8234 \glossentrydesc{##1} & ##2\tabularnewline
8235 }%

```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```

8236 \renewcommand{\subglossentry}[3]{%
8237 &
8238 \glssubentryitem{##2}%
8239 \glstarget{##2}{\strut}\glossentrydesc{##2} &
8240 ##3\tabularnewline
8241 }%

```

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>)

```

8242 \ifglsnogroupskip
8243 \renewcommand*\glsgroupskip{}%
8244 \else
8245 \renewcommand*\glsgroupskip{{ & & \tabularnewline}%
8246 \fi
8247 }

```

long3colborder The long3colborder style is like the long3col style but with a border:

```

8248 \newglossarystyle{long3colborder}{%
    Base it on the glostylelong3col style:
8249 \setglossarystyle{long3col}%
    Use a longtable with 3 columns with vertical lines around them:
8250 \renewenvironment{theglossary}%
8251 {\begin{longtable}{|l|p{\glstdescwidth}|p{\glspagelistwidth}|}%
8252 {\end{longtable}}%

```

Place horizontal lines at the head and foot of the table:

```
8253 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8254 }
```

`long3colheader` The `long3colheader` style is like `long3col` but with a header row:

```
8255 \newglossarystyle{long3colheader}{%
```

Base it on the `glostylelong3col` style:

```
8256 \setglossarystyle{long3col}%
```

Set the table's header:

```
8257 \renewcommand*{\glossaryheader}{%
8258 \bfseries\entryname&\bfseries\descriptionname&
8259 \bfseries\pagelistname\tabularnewline\endhead}%
8260 }
```

`colheaderborder` The `long3colheaderborder` style is like the above but with a border

```
8261 \newglossarystyle{long3colheaderborder}{%
```

Base it on the `glostylelong3colborder` style:

```
8262 \setglossarystyle{long3colborder}%
```

Set the table's header and add horizontal line at table's foot:

```
8263 \renewcommand*{\glossaryheader}{%
8264 \hline
8265 \bfseries\entryname&\bfseries\descriptionname&
8266 \bfseries\pagelistname\tabularnewline\hline\endhead
8267 \hline\endfoot}%
8268 }
```

`long4col` The `long4col` style has four columns where the third column contains the value of the associated symbol key.

```
8269 \newglossarystyle{long4col}{%
```

Use a `longtable` with 4 columns:

```
8270 \renewenvironment{theglossary}%
8271 {\begin{longtable}{llll}}%
8272 {\end{longtable}}%
```

No table header:

```
8273 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8274 \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):

```
8275 \renewcommand{\glossentry}[2]{%
8276 \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8277 \glossentrydesc{##1} &
8278 \glossentrysymbol{##1} &
8279 ##2\tabularnewline
8280 }%
```


Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
8281 \renewcommand{\subglossentry}[3]{%
8282     &
8283     \glssubentryitem{##2}%
8284     \glstarget{##2}{\strut}\glossentrydesc{##2} &
8285     \glossentrysymbol{##2} & ##3\tabularnewline
8286 }%
```

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>)

```
8287 \ifglsgroupskip
8288 \renewcommand*{\glsgroupskip}{}%
8289 \else
8290 \renewcommand*{\glsgroupskip}{ & & \tabularnewline}%
8291 \fi
8292 }
```

long4colheader The long4colheader style is like long4col but with a header row.

```
8293 \newglossarystyle{long4colheader}{%
```

Base it on the glostylelong4col style:

```
8294 \setglossarystyle{long4col}{%
```

Table has a header:

```
8295 \renewcommand*{\glossaryheader}{%
8296 \bfseries\entryname&\bfseries\descriptionname&
8297 \bfseries \symbolname&
8298 \bfseries\pagelistname\tabularnewline\endhead}%
8299 }
```

long4colborder The long4colborder style is like long4col but with a border.

```
8300 \newglossarystyle{long4colborder}{%
```

Base it on the glostylelong4col style:

```
8301 \setglossarystyle{long4col}{%
```

Use a longtable with 4 columns surrounded by vertical lines:

```
8302 \renewenvironment{theglossary}%
8303 {\begin{longtable}{|l|l|l|l|}}%
8304 {\end{longtable}}%
```

Add horizontal lines to the head and foot of the table:

```
8305 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8306 }
```

colheaderborder The long4colheaderborder style is like the above but with a border.

```
8307 \newglossarystyle{long4colheaderborder}{%
```

Base it on the glostylelong4col style:

```
8308 \setglossarystyle{long4col}{%
```

Use a longtable with 4 columns surrounded by vertical lines:

```
8309 \renewenvironment{theglossary}%
8310 {\begin{longtable}{|l|l|l|l|}}%
8311 {\end{longtable}}%
```

Add table header and horizontal line at the table's foot:

```
8312 \renewcommand*{\glossaryheader}{%
8313 \hline\bfseries\entryname&\bfseries\descriptionname&
8314 \bfseries \symbolname&
8315 \bfseries\pagelistname\tabularnewline\hline\endhead
8316 \hline\endfoot}%
8317 }
```

altlong4col The altlong4col style is like the long4col style but can have multiline descriptions and page lists.

```
8318 \newglossarystyle{altlong4col}{%
```

Base it on the glostylelong4col style:

```
8319 \setglossarystyle{long4col}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8320 \renewenvironment{theglossary}%
8321 {\begin{longtable}{lp{\glsdescwidth}lp{\glspagelistwidth}}}%
8322 {\end{longtable}}%
8323 }
```

altlong4colheader The altlong4colheader style is like altlong4col but with a header row.

```
8324 \newglossarystyle{altlong4colheader}{%
```

Base it on the glostylelong4colheader style:

```
8325 \setglossarystyle{long4colheader}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8326 \renewenvironment{theglossary}%
8327 {\begin{longtable}{lp{\glsdescwidth}lp{\glspagelistwidth}}}%
8328 {\end{longtable}}%
8329 }
```

altlong4colborder The altlong4colborder style is like altlong4col but with a border.

```
8330 \newglossarystyle{altlong4colborder}{%
```

Base it on the glostylelong4colborder style:

```
8331 \setglossarystyle{long4colborder}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8332 \renewenvironment{theglossary}%
8333 {\begin{longtable}{|lp{\glsdescwidth}|lp{\glspagelistwidth}|}}%
8334 {\end{longtable}}%
8335 }
```

colheaderborder The altlong4colheaderborder style is like the above but with a header as well as a border.

```
8336 \newglossarystyle{altlong4colheaderborder}{%
```

Base it on the glostylelong4colheaderborder style:

```
8337 \setglossarystyle{long4colheaderborder}{%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8338 \renewenvironment{theglossary}{%
```

```
8339 {\begin{longtable}{|l|p{\glstdescwidth}|l|p{\glspagelistwidth}|}}%
```

```
8340 {\end{longtable}}%
```

```
8341 }
```

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>

```
8342 \ProvidesPackage{glossary-longbooktabs}[2016/12/16 v4.27 (NLCT)]
```

Requires booktabs package:

```
8343 \RequirePackage{booktabs}
```

and the base packages for long styles:

```
8344 \RequirePackage{glossary-long}
```

```
8345 \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.

```
8346 \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.

```
8347 \glspatchLToutput
```

As with the longheader style, use the long style as a base.

```
8348 \setglossarystyle{long}{%
```

Add a header with rules.

```
8349 \renewcommand*{\glossaryheader}{%
```

```
8350 \toprule \bfseries \entryname & \bfseries
```

```
8351 \descriptionname\tabularnewline\midrule\endhead
```

```
8352 \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.

```
8353 \ifglsnogroupskip
```

```

8354 \renewcommand*{\glsgroupskip}{}%
8355 \else
8356 \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
8357 \fi
8358 }

```

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.

```
8359 \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.

```
8360 \glspatchLToutput
```

Use the long3col style as a base.

```
8361 \setglossarystyle{long3col}{%
```

Add a header with rules.

```

8362 \renewcommand*{\glossaryheader}{%
8363 \toprule \bfseries \entryname &
8364 \bfseries \descriptionname &
8365 \bfseries \pagelistname
8366 \tabularnewline\midrule\endhead
8367 \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.

```

8368 \ifglsgroupskip
8369 \renewcommand*{\glsgroupskip}{}%
8370 \else
8371 \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
8372 \fi
8373 }

```

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.

```
8374 \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.

```
8375 \glspatchLToutput
```

Use the long4col style as a base.

```
8376 \setglossarystyle{long4col}{%
```

Add a header with rules.

```

8377 \renewcommand*{\glossaryheader}{%
8378 \toprule \bfseries \entryname &
8379 \bfseries \descriptionname &
8380 \bfseries \symbolname &

```

```

8381 \bfseries \pagelistname
8382 \tabularnewline\midrule\endhead
8383 \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.

```

8384 \ifglsgnogroupskip
8385 \renewcommand*{\glsgroupskip}{}%
8386 \else
8387 \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
8388 \fi
8389 }

```

`ng4col-booktabs` The `altlong4col-booktabs` style is similar to the `altlong4colheader` style but uses the `booktabs` rules and patches `longtable` to check for group skip occurring at a page break.

```

8390 \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.

```

8391 \glspatchLToutput

```

Use the `long4col-booktabs` style as a base.

```

8392 \setglossarystyle{long4col-booktabs}%

```

Change the column specifications:

```

8393 \renewenvironment{theglossary}%
8394 {\begin{longtable}{lp{\glsgdescwidth}lp{\glspagelistwidth}}}%
8395 {\end{longtable}}%
8396 }

```

Ragged styles.

`ragged-booktabs` The `longragged-booktabs` style is similar to the `longragged` style but uses the `booktabs` rules and patches `longtable` to check for group skip occurring at a page break.

```

8397 \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.

```

8398 \glspatchLToutput

```

Use the `long-booktabs` style as a base.

```

8399 \setglossarystyle{long-booktabs}%

```

Adjust the column specification.

```

8400 \renewenvironment{theglossary}%
8401 {\begin{longtable}{l>\raggedright}p{\glsgdescwidth}}}%
8402 {\end{longtable}}%
8403 }

```

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.

```
8404 \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.

```
8405 \glspatchLToutput
```

Use the long3col-booktabs style as a base.

```
8406 \setglossarystyle{long3col-booktabs}{%
```

Adjust the column specification.

```
8407 \renewenvironment{theglossary}{%
```

```
8408 {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}}%
```

```
8409 >{\raggedright}p{\glspagelistwidth}}}%
```

```
8410 {\end{longtable}}}%
```

```
8411 }
```

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.

```
8412 \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.

```
8413 \glspatchLToutput
```

Use the altlong4col-booktabs style as a base.

```
8414 \setglossarystyle{altlong4col-booktabs}{%
```

Adjust the column specification.

```
8415 \renewenvironment{theglossary}{%
```

```
8416 {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}l%
```

```
8417 >{\raggedright}p{\glspagelistwidth}}}%
```

```
8418 {\end{longtable}}}%
```

```
8419 }
```

sLTpenaltycheck

```
8420 \newcommand*{\glsLTpenaltycheck}{%
```

```
8421 \ifnum\outputpenalty=-50\vskip-\normalbaselineskip\relax\fi
```

```
8422 }
```

enaltygroupskip

```
8423 \newcommand{\glspenaltygroupskip}{%
```

```
8424 \noalign{\penalty-50\vskip\normalbaselineskip}}
```

restoreLToutput Provide a way of restoring \LT@output for the user.

```
8425 \let\@gls@org@LT@output\LT@output
```

```
8426 \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

```
8427 \newcommand*{\glspatchLToutput}{%
8428 \renewcommand*{\LT@output}{%
8429 \ifnum\outputpenalty <-\@Mi
8430 \ifnum\outputpenalty > -\LT@end@pen
8431 \LT@err{floats and marginpars not allowed in a longtable}\@ehc
8432 \else
8433 \setbox\z@\vbox{\unvbox\@cclv}%
8434 \ifdim \ht\LT@lastfoot>\ht\LT@foot
8435 \dimen@\pagegoal
8436 \advance\dimen@-\ht\LT@lastfoot
8437 \ifdim\dimen@<\ht\z@
8438 \setbox\@cclv\vbox{\unvbox\z@\copy\LT@foot\vss}%
8439 \@makecol
8440 \@outputpage
8441 \setbox\z@\vbox{\box\LT@head\glslTpenaltycheck}%
8442 \fi
8443 \fi
8444 \global\@colroom\@colht
8445 \global\vsizel\@colht
8446 {\unvbox\z@\box\ifvoid\LT@lastfoot\LT@foot\else\LT@lastfoot\fi}%
8447 \fi
8448 \else
8449 \setbox\@cclv\vbox{\unvbox\@cclv\copy\LT@foot\vss}%
8450 \@makecol
8451 \@outputpage
8452 \global\vsizel\@colroom
8453 \copy\LT@head
8454 \glslTpenaltycheck
8455 \nobreak
8456 \fi
8457 }%
8458 }
```

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.

```
8459 \ProvidesPackage{glossary-longragged}[2016/12/16 v4.27 (NLCT)]
```

Requires the package:

```
8460 \RequirePackage{array}
```

Requires the package:

```
8461 \RequirePackage{longtable}
```

`\glsdescwidth` This is a length that governs the width of the description column. This may have already been defined.

```

8462 \@ifundefined{glsdescwidth}{%
8463   \newlength{glsdescwidth
8464   \setlength{glsdescwidth}{0.6\hsize}
8465 }{}

```

`lspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```

8466 \@ifundefined{glspagelistwidth}{%
8467   \newlength{glspagelistwidth
8468   \setlength{glspagelistwidth}{0.1\hsize}
8469 }{}

```

`longragged` The longragged glossary style is like the long but uses ragged right formatting for the description column.

```

8470 \newglossarystyle{longragged}{%

```

Use longtable with two columns:

```

8471   \renewenvironment{theglossary}%
8472     {\begin{longtable}{l>{\raggedright}p{glsdescwidth}}}%
8473     {\end{longtable}}%

```

Do nothing at the start of the environment:

```

8474   \renewcommand*{\glossaryheader}{}%

```

No heading between groups:

```

8475   \renewcommand*{\glsgroupheading}[1]{}%

```

Main (level 0) entries displayed in a row:

```

8476   \renewcommand{\glossentry}[2]{%
8477     \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8478     \glossentrydesc{##1}\glspostdescription\space ##2%
8479     \tabularnewline
8480   }%

```

Sub entries displayed on the following row without the name:

```

8481   \renewcommand{\subglossentry}[3]{%
8482     &
8483     \glssubentryitem{##2}%
8484     \glstarget{##2}{\strut}\glossentrydesc{##2}%
8485     \glspostdescription\space ##3%
8486     \tabularnewline
8487   }%

```

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>)

```

8488   \ifglsnogroupskip
8489     \renewcommand*{\glsgroupskip}{}%
8490   \else
8491     \renewcommand*{\glsgroupskip}{ & \tabularnewline}%

```



```

8492 \fi
8493 }

```

`longraggedborder` The `longraggedborder` style is like the above, but with horizontal and vertical lines:

```

8494 \newglossarystyle{longraggedborder}{%
    Base it on the glostylelongragged style:
8495 \setglossarystyle{longragged}%
    Use longtable with two columns with vertical lines between each column:
8496 \renewenvironment{theglossary}{%
8497 \begin{longtable}{|l|>{\raggedright}p{\glsgdescwidth}|}%
8498 {\end{longtable}}%
    Place horizontal lines at the head and foot of the table:
8499 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8500 }

```

`longraggedheader` The `longraggedheader` style is like the `longragged` style but with a header:

```

8501 \newglossarystyle{longraggedheader}{%
    Base it on the glostylelongragged style:
8502 \setglossarystyle{longragged}%
    Set the table's header:
8503 \renewcommand*{\glossaryheader}{%
8504 \bfseries \entryname & \bfseries \descriptionname
8505 \tabularnewline\endhead}%
8506 }

```

`longraggedheaderborder` The `longraggedheaderborder` style is like the `longragged` style but with a header and border:

```

8507 \newglossarystyle{longraggedheaderborder}{%
    Base it on the glostylelongraggedborder style:
8508 \setglossarystyle{longraggedborder}%
    Set the table's header and add horizontal line to table's foot:
8509 \renewcommand*{\glossaryheader}{%
8510 \hline\bfseries \entryname & \bfseries \descriptionname
8511 \tabularnewline\hline
8512 \endhead
8513 \hline\endfoot}%
8514 }

```

`longragged3col` The `longragged3col` style is like `longragged` but with 3 columns

```

8515 \newglossarystyle{longragged3col}{%
    Use a longtable with 3 columns:
8516 \renewenvironment{theglossary}{%
8517 {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}%
8518 >{\raggedright}p{\glspagelistwidth}}}%
8519 {\end{longtable}}%

```

No table header:

```
8520 \renewcommand*\glossaryheader{}\%
```

No headings between groups:

```
8521 \renewcommand*\glsgroupheading[1]{}\%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
8522 \renewcommand\glossentry[2]{%
8523   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8524   \glossentrydesc{##1} & ##2\tabularnewline
8525 }
```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```
8526 \renewcommand\subglossentry[3]{%
8527   &
8528   \glssubentryitem{##2}%
8529   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8530   ##3\tabularnewline
8531 }
```

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>)

```
8532 \ifglsnogroupskip
8533 \renewcommand*\glsgroupskip{}\%
8534 \else
8535 \renewcommand*\glsgroupskip{ & & \tabularnewline}%
8536 \fi
8537 }
```

agged3colborder The longragged3colborder style is like the longragged3col style but with a border:

```
8538 \newglossarystyle{longragged3colborder}{%
```

Base it on the glostylelongragged3col style:

```
8539 \setglossarystyle{longragged3col}%
```

Use a longtable with 3 columns with vertical lines around them:

```
8540 \renewenvironment{theglossary}%
8541   {\begin{longtable}{|l|>{\raggedright}p{\glsdescwidth}|}%
8542    >{\raggedright}p{\glspagelistwidth}|}%
8543   {\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
8544 \renewcommand*\glossaryheader{\hline\endhead\hline\endfoot}%
8545 }
```

agged3colheader The longragged3colheader style is like longragged3col but with a header row:

```
8546 \newglossarystyle{longragged3colheader}{%
```

Base it on the glostylelongragged3col style:

```
8547 \setglossarystyle{longragged3col}%
```

Set the table's header:

```
8548 \renewcommand*{\glossaryheader}{%
8549     \bfseries\entryname&\bfseries\descriptionname&
8550     \bfseries\pagelistname\tabularnewline\endhead}%
8551 }
```

`colheaderborder` The `longragged3colheaderborder` style is like the above but with a border

```
8552 \newglossarystyle{longragged3colheaderborder}{%
```

Base it on the `glostylelongragged3colborder` style:

```
8553 \setglossarystyle{longragged3colborder}%
```

Set the table's header and add horizontal line at table's foot:

```
8554 \renewcommand*{\glossaryheader}{%
8555     \hline
8556     \bfseries\entryname&\bfseries\descriptionname&
8557     \bfseries\pagelistname\tabularnewline\hline\endhead
8558     \hline\endfoot}%
8559 }
```

`longragged4col` 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.

```
8560 \newglossarystyle{altlongragged4col}{%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8561 \renewenvironment{theglossary}%
8562     {\begin{longtable}{1>{\raggedright}p{\glstdescwidth}1%
8563         >{\raggedright}p{\glspagelistwidth}}}%
8564     {\end{longtable}}%
```

No table header:

```
8565 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8566 \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):

```
8567 \renewcommand{\glossentry}[2]{%
8568     \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8569     \glossentrydesc{##1} & \glossentrysymbol{##1} &
8570     ##2\tabularnewline
8571 }%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
8572 \renewcommand{\subglossentry}[3]{%
8573     &
8574     \glssubentryitem{##2}%
8575     \glstarget{##2}{\strut}\glossentrydesc{##2} &
```

```

8576 \glossentrysymbol{##2} & ##3\tabularnewline
8577 }%

```

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>)

```

8578 \ifglsgroupskip
8579 \renewcommand*{\glsgroupskip}{}%
8580 \else
8581 \renewcommand*{\glsgroupskip}{ & & \tabularnewline}%
8582 \fi
8583 }

```

agged4colheader The altlongragged4colheader style is like altlongragged4col but with a header row.

```
8584 \newglossarystyle{altlongragged4colheader}{%
```

Base it on the glostylealtlongragged4col style:

```
8585 \setglossarystyle{altlongragged4col}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```

8586 \renewenvironment{theglossary}%
8587 {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}l%
8588 >{\raggedright}p{\glspagelistwidth}}}%
8589 {\end{longtable}}%

```

Table has a header:

```

8590 \renewcommand*{\glossaryheader}{%
8591 \bfseries\entryname&\bfseries\descriptionname&
8592 \bfseries \symbolname&
8593 \bfseries\pagelistname\tabularnewline\endhead}%
8594 }

```

agged4colborder The altlongragged4colborder style is like altlongragged4col but with a border.

```
8595 \newglossarystyle{altlongragged4colborder}{%
```

Base it on the glostylealtlongragged4col style:

```
8596 \setglossarystyle{altlongragged4col}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```

8597 \renewenvironment{theglossary}%
8598 {\begin{longtable}{|l|>{\raggedright}p{\glsgdescwidth}|l|}%
8599 >{\raggedright}p{\glspagelistwidth}|}%
8600 {\end{longtable}}%

```

Add horizontal lines to the head and foot of the table:

```

8601 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8602 }

```

colheaderborder The altlongragged4colheaderborder style is like the above but with a header as well as a border.

```
8603 \newglossarystyle{altlongragged4colheaderborder}{%
```

Base it on the `glostylealtlongragged4col` style:

```
8604 \setglossarystyle{altlongragged4col}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8605 \renewenvironment{theglossary}%  
8606 {\begin{longtable}{|l|>{\raggedright}p{\glstdescwidth}|l|}%  
8607 >{\raggedright}p{\glspagelistwidth}|}}%  
8608 {\end{longtable}}%
```

Add table header and horizontal line at the table's foot:

```
8609 \renewcommand*{\glossaryheader}{%  
8610 \hline\bfseries\entryname&\bfseries\descriptionname&  
8611 \bfseries \symbolname&  
8612 \bfseries\pagelistname\tabularnewline\hline\endhead  
8613 \hline\endfoot}%  
8614 }
```

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.

```
8615 \ProvidesPackage{glossary-mcols}[2016/12/16 v4.27 (NLCT)]
```

Required packages:

```
8616 \RequirePackage{multicol}  
8617 \RequirePackage{glossary-tree}
```

`\indexspace` The are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```
8618 \providecommand{\indexspace}{%  
8619 \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax  
8620 }
```

`\glsmcols` Define macro in which to store the number of columns. (Defaults to 2.)

```
8621 \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.)

```
8622 \newglossarystyle{mcolindex}{%  
8623 \setglossarystyle{index}%  
8624 \renewenvironment{theglossary}%  
8625 {%  
  
8626 \begin{multicols}{\glsmcols}  
8627 \setlength{\parindent}{0pt}%  
8628 \setlength{\parskip}{0pt plus 0.3pt}%
```

```

8629     \let\item\glstreeitem
8630     \let\subitem\glstreesubitem
8631     \let\subsubitem\glstreesubsubitem
8632 }%
8633 {\end{multicols}}%
8634 }

```

`mcindexgroup` As `mcindex` but has headings:

```

8635 \newglossarystyle{mcindexgroup}{%
8636   \setglossarystyle{mcindex}%
8637   \renewcommand*{\glsgroupheading}[1]{%
8638     \item\glstreegroupheaderfmt{\glsgrouptitle{##1}}\indexspace}%
8639 }

```

`indexhypergroup` The `mcindexhypergroup` style is like the `mcindexgroup` style but has hyper navigation.

```

8640 \newglossarystyle{mcindexhypergroup}{%

```

Base it on the `glostylemcindex` style:

```

8641   \setglossarystyle{mcindex}%

```

Put navigation links to the groups at the start of the glossary:

```

8642   \renewcommand*{\glossaryheader}{%
8643     \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.

```

8644   \renewcommand*{\glsgroupheading}[1]{%
8645     \item\glstreegroupheaderfmt
8646       {\glsnavigationtarget{##1}{\glsgrouptitle{##1}}}%
8647     \indexspace}%
8648 }

```

`colindexspannav` Similar to `mcindexhypergroup`, but puts the navigation line in the optional argument of `multicols`.

```

8649 \newglossarystyle{colindexspannav}{%
8650   \setglossarystyle{index}%
8651   \renewenvironment{theglossary}%
8652   {%
8653     \begin{multicols}{\glsmcols}[\noindent\glstreenavigationfmt{\glsnavigation}]
8654     \setlength{\parindent}{0pt}%
8655     \setlength{\parskip}{0pt plus 0.3pt}%
8656     \let\item\glstreeitem}%
8657   {\end{multicols}}%

```

Add a heading for each group (with a target). The group's title is in bold followed by a vertical gap.

```

8658   \renewcommand*{\glsgroupheading}[1]{%
8659     \item\glstreegroupheaderfmt

```

```

8660      {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
8661      \indexspace}%
8662 }

```

mcoltree Multi-column index style. Same as the tree, but puts the glossary in multiple columns.

```

8663 \newglossarystyle{mcoltree}{%
8664   \setglossarystyle{tree}%
8665   \renewenvironment{theglossary}%
8666   {%
8667     \begin{multicols}{\glsmcols}
8668     \setlength{\parindent}{0pt}%
8669     \setlength{\parskip}{0pt plus 0.3pt}%
8670   }%
8671   {\end{multicols}}%
8672 }

```

mcoltreegroup Like the mcoltree style but the glossary groups have headings.

```

8673 \newglossarystyle{mcoltreegroup}{%
      Base it on the glostylemcoltree style:
8674   \setglossarystyle{mcoltree}%
      Each group has a heading (in bold) followed by a vertical gap):
8675   \renewcommand{\glsgroupheading}[1]{\par
8676     \noindent\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}\par\indexspace}%
8677 }

```

ltreehypergroup The mcoltreehypergroup style is like the treegroup style, but has a set of links to the groups at the start of the glossary.

```

8678 \newglossarystyle{mcoltreehypergroup}{%
      Base it on the glostylemcoltree style:
8679   \setglossarystyle{mcoltree}%
      Put navigation links to the groups at the start of the theglossary environment:
8680   \renewcommand*{\glossaryheader}{%
8681     \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
      Each group has a heading (in bold with a target) followed by a vertical gap):
8682   \renewcommand*{\glsgroupheading}[1]{%
8683     \par\noindent
8684     \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8685     \indexspace}%
8686 }

```

mcoltreespannav Similar to the mcoltreehypergroup style but the navigation line is put in the optional argument of the multicols environment.

```

8687 \newglossarystyle{mcoltreespannav}{%
8688   \setglossarystyle{tree}%
8689   \renewenvironment{theglossary}%
8690   {%

```

```

8691     \begin{multicols}{\glsmcols}[\noindent\glstreenavigationfmt{\glsnavigation}]
8692     \setlength{\parindent}{0pt}%
8693     \setlength{\parskip}{0pt plus 0.3pt}%
8694 }%
8695 {\end{multicols}}%

```

Each group has a heading (in bold with a target) followed by a vertical gap):

```

8696 \renewcommand*{\glsgroupheading}[1]{%
8697   \par\noindent
8698   \glstreegroupheaderfmt{\glsnahypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8699   \indexspace}%
8700 }

```

mcoltreenoname Multi-column index style. Same as the **treenoname**, but puts the glossary in multiple columns.

```

8701 \newglossarystyle{mcoltreenoname}{%
8702   \setglossarystyle{treenoname}%
8703   \renewenvironment{theglossary}%
8704   {%
8705     \begin{multicols}{\glsmcols}
8706     \setlength{\parindent}{0pt}%
8707     \setlength{\parskip}{0pt plus 0.3pt}%
8708   }%
8709   {\end{multicols}}%
8710 }

```

treenonamegroup Like the **mcoltreenoname** style but the glossary groups have headings.

```

8711 \newglossarystyle{mcoltreenonamegroup}{%
8712   Base it on the glostylemcoltreenoname style:
8713   \setglossarystyle{mcoltreenoname}%
8714   Give each group a heading:
8715   \renewcommand{\glsgroupheading}[1]{\par
8716     \noindent\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}\par\indexspace}%
8717 }

```

onamehypergroup The **mcoltreenonamehypergroup** style is like the **mcoltreenonamegroup** style, but has a set of links to the groups at the start of the glossary.

```

8716 \newglossarystyle{mcoltreenonamehypergroup}{%
8717   Base it on the glostylemcoltreenoname style:
8718   \setglossarystyle{mcoltreenoname}%
8719   Put navigation links to the groups at the start of the theglossary environment:
8720   \renewcommand*{\glossaryheader}{%
8721     \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
8722 }

```

Each group has a heading (in bold with a target) followed by a vertical gap):

```

8720 \renewcommand*{\glsgroupheading}[1]{%
8721   \par\noindent

```



```

8722 \glstreegroupheaderfmt{\glshypertarget{##1}{\glsgrouptitle{##1}}}\par
8723 \indexspace}%
8724 }

```

treenonamespannav Similar to the `mcoltreenonamehypergroup` style but the navigation line is put in the optional argument of the `multicols` environment.

```

8725 \newglossarystyle{mcoltreenonamespannav}{%
8726 \setglossarystyle{treenoname}%
8727 \renewenvironment{theglossary}%
8728 {%
8729 \begin{multicols}{\glsmcols}\noindent\glstreenavigationfmt{\glsnavigation}]
8730 \setlength{\parindent}{0pt}%
8731 \setlength{\parskip}{0pt plus 0.3pt}%
8732 }%
8733 {\end{multicols}}%

```

Each group has a heading (in bold with a target) followed by a vertical gap):

```

8734 \renewcommand*{\glsgroupheading}[1]{%
8735 \par\noindent
8736 \glstreegroupheaderfmt{\glshypertarget{##1}{\glsgrouptitle{##1}}}\par
8737 \indexspace}%
8738 }

```

mcolalmtree Multi-column index style. Same as the `almtree`, but puts the glossary in multiple columns.

```

8739 \newglossarystyle{mcolalmtree}{%
8740 \setglossarystyle{almtree}%
8741 \renewenvironment{theglossary}%
8742 {%
8743 \begin{multicols}{\glsmcols}
8744 \def\@gls@prevlevel{-1}%
8745 \mbox{}\par
8746 }%
8747 {\par\end{multicols}}%
8748 }

```

mcolalmtreegroup Like the `mcolalmtree` style but the glossary groups have headings.

```

8749 \newglossarystyle{mcolalmtreegroup}{%

```

Base it on the `glostylemcolalmtree` style:

```

8750 \setglossarystyle{mcolalmtree}%

```

Give each group a heading.

```

8751 \renewcommand{\glsgroupheading}[1]{\par
8752 \def\@gls@prevlevel{-1}%
8753 \hangindent0pt\relax
8754 \parindent0pt\relax
8755 \glstreegroupheaderfmt{\glsgrouptitle{##1}}\par\indexspace}%
8756 }

```

treehypergroup The mcolalmtreehypergroup style is like the mcolalmtreegroup style, but has a set of links to the groups at the start of the glossary.

```
8757 \newglossarystyle{mcolalmtreehypergroup}{%
```

Base it on the glostylemcolalmtree style:

```
8758 \setglossarystyle{mcolalmtree}%
```

Put the navigation links in the header

```
8759 \renewcommand*{\glossaryheader}{%
```

```
8760 \par
```

```
8761 \def\@gls@prevlevel{-1}%
```

```
8762 \hangindent0pt\relax
```

```
8763 \parindent0pt\relax
```

```
8764 \glstreenavigationfmt{\glsnavigation}\par\indexspace}%
```

Put a hypertarget at the start of each group

```
8765 \renewcommand*{\glsgroupheading}[1]{%
```

```
8766 \par
```

```
8767 \def\@gls@prevlevel{-1}%
```

```
8768 \hangindent0pt\relax
```

```
8769 \parindent0pt\relax
```

```
8770 \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
```

```
8771 \indexspace}%
```

```
8772 }
```

almtreeespannav Similar to the mcolalmtreehypergroup style but the navigation line is put in the optional argument of the multicols environment.

```
8773 \newglossarystyle{mcolalmtreeespannav}{%
```

```
8774 \setglossarystyle{almtree}%
```

```
8775 \renewenvironment{theglossary}%
```

```
8776 {%
```

```
8777 \begin{multicols}{\glsncols}[\noindent\glstreenavigationfmt{\glsnavigation}]
```

```
8778 \def\@gls@prevlevel{-1}%
```

```
8779 \mbox{}\par
```

```
8780 }%
```

```
8781 {\par\end{multicols}}}%
```

Put a hypertarget at the start of each group

```
8782 \renewcommand*{\glsgroupheading}[1]{%
```

```
8783 \par
```

```
8784 \def\@gls@prevlevel{-1}%
```

```
8785 \hangindent0pt\relax
```

```
8786 \parindent0pt\relax
```

```
8787 \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
```

```
8788 \indexspace}
```

```
8789 }
```

3.8 Glossary Styles using supertabular environment (glossary-super package)

The glossary styles defined in the package use the supertabular environment.

8790 \ProvidesPackage{glossary-super}[2016/12/16 v4.27 (NLCT)]

Requires the package:

8791 \RequirePackage{supertabular}

`\glsdescwidth` This is a length that governs the width of the description column. This may already have been defined if has been loaded.

8792 \@ifundefined{glsdescwidth}{%

8793 \newlength{glsdescwidth

8794 \setlength{glsdescwidth}{0.6\hsize}

8795 }{}

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined if has been loaded.

8796 \@ifundefined{glspagelistwidth}{%

8797 \newlength{glspagelistwidth

8798 \setlength{glspagelistwidth}{0.1\hsize}

8799 }{}

`super` The super glossary style uses the supertabular environment (it uses lengths defined in the package.)

8800 \newglossarystyle{super}{%

Put the glossary in a supertabular environment with two columns and no head or tail:

8801 \renewenvironment{theglossary}{%

8802 {\tablehead{}\tabletail{}}%

8803 \begin{supertabular}{lp{glsdescwidth}}%

8804 {\end{supertabular}}%

Do nothing at the start of the table:

8805 \renewcommand*{\glossaryheader}{}%

No group headings:

8806 \renewcommand*{\glsgroupheading}[1]{}%

Main (level 0) entries put in a row (name in first column, description and page list in second column):

8807 \renewcommand{\glossentry}[2]{%

8808 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &

8809 \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline

8810 }%

Sub entries put in a row (no name, description and page list in second column):

8811 \renewcommand{\subglossentry}[3]{%

8812 &

8813 \glssubentryitem{##2}}%

```

8814 \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
8815 ##3\tabularnewline
8816 }%

```

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>)

```

8817 \ifglsgroupskip
8818 \renewcommand*{\glsgroupskip}{}%
8819 \else
8820 \renewcommand*{\glsgroupskip}{& \tabularnewline}%
8821 \fi
8822 }

```

superborder The superborder style is like the above, but with horizontal and vertical lines:

```
8823 \newglossarystyle{superborder}{%
```

Base it on the glostylesuper style:

```
8824 \setglossarystyle{super}%
```

Put the glossary in a supertabular environment with two columns and a horizontal line in the head and tail:

```

8825 \renewenvironment{theglossary}%
8826 {\tablehead{\hline}\tabletail{\hline}%
8827 \begin{supertabular}{|l|p{\glsgdescwidth}|}%
8828 {\end{supertabular}}%
8829 }

```

superheader The superheader style is like the super style, but with a header:

```
8830 \newglossarystyle{superheader}{%
```

Base it on the glostylesuper style:

```
8831 \setglossarystyle{super}%
```

Put the glossary in a supertabular environment with two columns, a header and no tail:

```

8832 \renewenvironment{theglossary}%
8833 {\tablehead{\bfseries \entryname &
8834 \bfseries\descriptionname\tabularnewline}%
8835 \tabletail{}}%
8836 \begin{supertabular}{lp{\glsgdescwidth}}%
8837 {\end{supertabular}}%
8838 }

```

superheaderborder The superheaderborder style is like the super style but with a header and border:

```
8839 \newglossarystyle{superheaderborder}{%
```

Base it on the glostylesuper style:

```
8840 \setglossarystyle{super}%
```

Put the glossary in a supertabular environment with two columns, a header and horizontal lines above and below the table:

```
8841 \renewenvironment{theglossary}%
```

```

8842   {\tablehead{\hline\bfseries \entryname &
8843             \bfseries \descriptionname\tabularnewline\hline}%
8844   \tabletail{\hline}
8845   \begin{supertabular}{|l|p{\glstdescwidth}|}%
8846   {\end{supertabular}}%
8847 }

```

super3col The super3col style is like the super style, but with 3 columns:

```
8848 \newglossarystyle{super3col}{%
```

Put the glossary in a supertabular environment with three columns and no head or tail:

```

8849 \renewenvironment{theglossary}%
8850   {\tablehead{}\tabletail{}}%
8851   \begin{supertabular}{lp{\glstdescwidth}p{\glspagelistwidth}}%
8852   {\end{supertabular}}%

```

Do nothing at the start of the table:

```
8853 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8854 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```

8855 \renewcommand{\glossentry}[2]{%
8856   \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8857   \glossentrydesc{##1} & ##2\tabularnewline
8858 }%

```

Sub entries on a row (no name, description in second column, page list in last column):

```

8859 \renewcommand{\subglossentry}[3]{%
8860   &
8861   \glssubentryitem{##2}%
8862   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8863   ##3\tabularnewline
8864 }%

```

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>)

```

8865 \ifglsgroupskip
8866   \renewcommand*{\glsgroupskip}{}%
8867 \else
8868   \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
8869 \fi
8870 }

```

super3colborder The super3colborder style is like the super3col style, but with a border:

```
8871 \newglossarystyle{super3colborder}{%
```

Base it on the glostylesuper3col style:

```
8872 \setglossarystyle{super3col}%
```

Put the glossary in a supertabular environment with three columns and a horizontal line in the head and tail:

```
8873 \renewenvironment{theglossary}%
8874 {\tablehead{\hline}\tabletail{\hline}%
8875 \begin{supertabular}{|l|p{\glsgdescwidth}|p{\glspagelistwidth}|}%
8876 {\end{supertabular}}}%
8877 }
```

super3colheader The super3colheader style is like the super3col style but with a header row:

```
8878 \newglossarystyle{super3colheader}{%
Base it on the glostylessuper3col style:
8879 \setglossarystyle{super3col}%
Put the glossary in a supertabular environment with three columns, a header and no tail:
8880 \renewenvironment{theglossary}%
8881 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8882 \bfseries\pagelistname\tabularnewline}\tabletail{}}%
8883 \begin{supertabular}{lp{\glsgdescwidth}p{\glspagelistwidth}}}%
8884 {\end{supertabular}}}%
8885 }
```

colheaderborder The super3colheaderborder style is like the super3col style but with a header and border:

```
8886 \newglossarystyle{super3colheaderborder}{%
Base it on the glostylessuper3colborder style:
8887 \setglossarystyle{super3colborder}%
Put the glossary in a supertabular environment with three columns, a header with horizontal
lines and a horizontal line in the tail:
8888 \renewenvironment{theglossary}%
8889 {\tablehead{\hline
8890 \bfseries\entryname&\bfseries\descriptionname&
8891 \bfseries\pagelistname\tabularnewline\hline}%
8892 \tabletail{\hline}%
8893 \begin{supertabular}{|l|p{\glsgdescwidth}|p{\glspagelistwidth}|}%
8894 {\end{supertabular}}}%
8895 }
```

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.

```
8896 \newglossarystyle{super4col}{%
Put the glossary in a supertabular environment with four columns and no head or tail:
8897 \renewenvironment{theglossary}%
8898 {\tablehead{}\tabletail{}}%
8899 \begin{supertabular}{llll}}{%
8900 \end{supertabular}}}%
Do nothing at the start of the table:
8901 \renewcommand*{\glossaryheader}{}%

```

No group headings:

```
8902 \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:

```
8903 \renewcommand{\glossentry}[2]{%
8904   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8905   \glossentrydesc{##1} &
8906   \glossentrysymbol{##1} & ##2\tabularnewline
8907 }%
```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```
8908 \renewcommand{\subglossentry}[3]{%
8909   &
8910   \glssubentryitem{##2}%
8911   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8912   \glossentrysymbol{##2} & ##3\tabularnewline
8913 }%
```

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>)

```
8914 \ifglsnogroupskip
8915 \renewcommand*\glsgroupskip{}{}%
8916 \else
8917 \renewcommand*\glsgroupskip{& & & \tabularnewline}%
8918 \fi
8919 }
```

super4colheader The super4colheader style is like the super4col but with a header row.

```
8920 \newglossarystyle{super4colheader}{%
```

Base it on the glostylesuper4col style:

```
8921 \setglossarystyle{super4col}%
```

Put the glossary in a supertabular environment with four columns, a header and no tail:

```
8922 \renewenvironment{theglossary}%
8923   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8924     \bfseries\symbolname &
8925     \bfseries\pagelistname\tabularnewline}%
8926   \tabletail{}}%
8927   \begin{supertabular}{1111}}%
8928   {\end{supertabular}}%
8929 }
```

super4colborder The super4colborder style is like the super4col but with a border.

```
8930 \newglossarystyle{super4colborder}{%
```

Base it on the glostylesuper4col style:

```
8931 \setglossarystyle{super4col}%
```

Put the glossary in a supertabular environment with four columns and a horizontal line in the head and tail:

```
8932 \renewenvironment{theglossary}%
8933   {\tablehead{\hline}\tabletail{\hline}%
8934    \begin{supertabular}{|l|l|l|l|}%
8935    {\end{supertabular}}}%
8936 }
```

colheaderborder The super4colheaderborder style is like the super4col but with a header and border.

```
8937 \newglossarystyle{super4colheaderborder}{%
```

Base it on the glostylessuper4col style:

```
8938 \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:

```
8939 \renewenvironment{theglossary}%
8940   {\tablehead{\hline\bfseries\entryname&\bfseries\descriptionname&
8941     \bfseries\symbolname &
8942     \bfseries\pagelistname\tabularnewline\hline}%
8943    \tabletail{\hline}%
8944    \begin{supertabular}{|l|l|l|l|}%
8945    {\end{supertabular}}}%
8946 }
```

altsuper4col The altsuper4col glossary style is like super4col but has provision for multiline descriptions.

```
8947 \newglossarystyle{altsuper4col}{%
```

Base it on the glostylessuper4col style:

```
8948 \setglossarystyle{super4col}%
```

Put the glossary in a supertabular environment with four columns and no head or tail:

```
8949 \renewenvironment{theglossary}%
8950   {\tablehead{}\tabletail{}\%
8951    \begin{supertabular}{lp{\glsgdescwidth}lp{\glspagelistwidth}}}%
8952   {\end{supertabular}}}%
8953 }
```

super4colheader The altsuper4colheader style is like the altsuper4col but with a header row.

```
8954 \newglossarystyle{altsuper4colheader}{%
```

Base it on the glostylessuper4colheader style:

```
8955 \setglossarystyle{super4colheader}%
```

Put the glossary in a supertabular environment with four columns, a header and no tail:

```
8956 \renewenvironment{theglossary}%
8957   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8958     \bfseries\symbolname &
8959     \bfseries\pagelistname\tabularnewline}\tabletail{}\%
8960    \begin{supertabular}{lp{\glsgdescwidth}lp{\glspagelistwidth}}}%
8961   {\end{supertabular}}}%
8962 }
```


`super4colborder` The `altsuper4colborder` style is like the `altsuper4col` but with a border.

```
8963 \newglossarystyle{altsuper4colborder}{%
```

Base it on the `glostylesuper4colborder` style:

```
8964 \setglossarystyle{super4colborder}{%
```

Put the glossary in a `supertabular` environment with four columns and a horizontal line in the head and tail:

```
8965 \renewenvironment{theglossary}{%
8966   {\tablehead{\hline}\tabletail{\hline}%
8967   \begin{supertabular}%
8968     {lllp{\glsgdescwidth}lllp{\glspagelistwidth}}}%
8969   {\end{supertabular}}}%
8970 }
```

`colheaderborder` The `altsuper4colheaderborder` style is like the `altsuper4col` but with a header and border.

```
8971 \newglossarystyle{altsuper4colheaderborder}{%
```

Base it on the `glostylesuper4colheaderborder` style:

```
8972 \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:

```
8973 \renewenvironment{theglossary}{%
8974   {\tablehead{\hline
8975     \bfseries\entryname &
8976     \bfseries\descriptionname &
8977     \bfseries\symbolname &
8978     \bfseries\pagelistname\tabularnewline\hline}%
8979   \tabletail{\hline}%
8980   \begin{supertabular}%
8981     {lllp{\glsgdescwidth}lllp{\glspagelistwidth}}}%
8982   {\end{supertabular}}}%
8983 }
```

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.

```
8984 \ProvidesPackage{glossary-superragged}[2016/12/16 v4.27 (NLCT)]
```

Requires the package:

```
8985 \RequirePackage{array}
```

Requires the package:

```
8986 \RequirePackage{supertabular}
```

`\glsdescwidth` This is a length that governs the width of the description column. This may already have been defined.

```

8987 \@ifundefined{glsdescwidth}{%
8988   \newlength{glsdescwidth
8989   \setlength{glsdescwidth}{0.6\hsize}
8990 }{}

```

`lspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```

8991 \@ifundefined{glspagelistwidth}{%
8992   \newlength{glspagelistwidth
8993   \setlength{glspagelistwidth}{0.1\hsize}
8994 }{}

```

`superragged` The superragged glossary style uses the supertabular environment.

```

8995 \newglossarystyle{superragged}{%

```

Put the glossary in a supertabular environment with two columns and no head or tail:

```

8996   \renewenvironment{theglossary}%
8997     {\tablehead{}\tabletail{}}%
8998     \begin{supertabular}{1>{\raggedright}p{glsdescwidth}}}%
8999     {\end{supertabular}}%

```

Do nothing at the start of the table:

```

9000   \renewcommand*{\glossaryheader}{}%

```

No group headings:

```

9001   \renewcommand*{\glsgroupheading}[1]{}%

```

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```

9002   \renewcommand{\glossentry}[2]{%
9003     \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
9004     \glossentrydesc{##1}\glspostdescription\space ##2%
9005     \tabularnewline
9006   }%

```

Sub entries put in a row (no name, description and page list in second column):

```

9007   \renewcommand{\subglossentry}[3]{%
9008     &
9009     \glssubentryitem{##2}%
9010     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
9011     ##3%
9012     \tabularnewline
9013   }%

```

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>)

```

9014   \ifglsnogroupskip
9015     \renewcommand*{\glsgroupskip}{}%
9016   \else

```

```

9017 \renewcommand*{\glsgroupskip}{& \tabularnewline}%
9018 \fi
9019 }

```

superraggedborder The superraggedborder style is like the above, but with horizontal and vertical lines:

```

9020 \newglossarystyle{superraggedborder}{%
    Base it on the glostylesuperragged style:
9021 \setglossarystyle{superragged}%
    Put the glossary in a supertabular environment with two columns and a horizontal line in the
    head and tail:
9022 \renewenvironment{theglossary}%
9023 {\tablehead{\hline}\tabletail{\hline}%
9024 \begin{supertabular}{|l|>{\raggedright}p{\glsgdescwidth}|}}%
9025 {\end{supertabular}}%
9026 }

```

superraggedheader The superraggedheader style is like the super style, but with a header:

```

9027 \newglossarystyle{superraggedheader}{%
    Base it on the glostylesuperragged style:
9028 \setglossarystyle{superragged}%
    Put the glossary in a supertabular environment with two columns, a header and no tail:
9029 \renewenvironment{theglossary}%
9030 {\tablehead{\bfseries \entryname & \bfseries \descriptionname
9031 \tabularnewline}%
9032 \tabletail{}}%
9033 \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}}}%
9034 {\end{supertabular}}%
9035 }

```

superraggedheaderborder The superraggedheaderborder style is like the superragged style but with a header and border:

```

9036 \newglossarystyle{superraggedheaderborder}{%
    Base it on the glostylesuper style:
9037 \setglossarystyle{superragged}%
    Put the glossary in a supertabular environment with two columns, a header and horizontal
    lines above and below the table:
9038 \renewenvironment{theglossary}%
9039 {\tablehead{\hline\bfseries \entryname &
9040 \bfseries \descriptionname\tabularnewline\hline}%
9041 \tabletail{\hline}
9042 \begin{supertabular}{|l|>{\raggedright}p{\glsgdescwidth}|}}%
9043 {\end{supertabular}}%
9044 }

```

superragged3col The superragged3col style is like the superragged style, but with 3 columns:

```

9045 \newglossarystyle{superragged3col}{%

```

Put the glossary in a supertabular environment with three columns and no head or tail:

```
9046 \renewenvironment{theglossary}%
9047   {\tablehead{}\tabletail{}}%
9048   \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}%
9049     >{\raggedright}p{\glspagelistwidth}}}%
9050   {\end{supertabular}}%
```

Do nothing at the start of the table:

```
9051 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9052 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
9053 \renewcommand{\glossentry}[2]{%
9054   \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
9055   \glossentrydesc{##1} &
9056   ##2\tabularnewline
9057 }%
```

Sub entries on a row (no name, description in second column, page list in last column):

```
9058 \renewcommand{\subglossentry}[3]{%
9059   &
9060   \glssubentryitem{##2}%
9061   \glstarget{##2}{\strut}\glossentrydesc{##2} &
9062   ##3\tabularnewline
9063 }%
```

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>)

```
9064 \ifglsgroupskip
9065 \renewcommand*{\glsgroupskip}{}%
9066 \else
9067 \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
9068 \fi
9069 }
```

ragged3colborder The superragged3colborder style is like the superragged3col style, but with a border:

```
9070 \newglossarystyle{superragged3colborder}{%
```

Base it on the glostylessuperragged3col style:

```
9071 \setglossarystyle{superragged3col}%
```

Put the glossary in a supertabular environment with three columns and a horizontal line in the head and tail:

```
9072 \renewenvironment{theglossary}%
9073   {\tablehead{\hline}\tabletail{\hline}%
9074   \begin{supertabular}{l|>{\raggedright}p{\glsgdescwidth}|%
9075     >{\raggedright}p{\glspagelistwidth}|}%
9076   {\end{supertabular}}%
9077 }
```

ragged3colheader The superragged3colheader style is like the superragged3col style but with a header row:

```
9078 \newglossarystyle{superragged3colheader}{%
```

Base it on the glostylesuperragged3col style:

```
9079 \setglossarystyle{superragged3col}{%
```

Put the glossary in a supertabular environment with three columns, a header and no tail:

```
9080 \renewenvironment{theglossary}{%
9081   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
9082     \bfseries\pagelistname\tabularnewline}\tabletail{}}%
9083   \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}%
9084     >{\raggedright}p{\glspagelistwidth}}}%
9085   {\end{supertabular}}}%
9086 }
```

colheaderborder The superragged3colheaderborder style is like the superragged3col style but with a header and border:

```
9087 \newglossarystyle{superragged3colheaderborder}{%
```

Base it on the glostylesuperragged3colborder style:

```
9088 \setglossarystyle{superragged3colborder}{%
```

Put the glossary in a supertabular environment with three columns, a header with horizontal lines and a horizontal line in the tail:

```
9089 \renewenvironment{theglossary}{%
9090   {\tablehead{\hline
9091     \bfseries\entryname&\bfseries\descriptionname&
9092     \bfseries\pagelistname\tabularnewline\hline}%
9093   \tabletail{\hline}%
9094   \begin{supertabular}{|l|>{\raggedright}p{\glsdescwidth}|%
9095     >{\raggedright}p{\glspagelistwidth}|}%
9096   {\end{supertabular}}}%
9097 }
```

superragged4col The altsuperragged4col glossary style is like altsuper4col style in the package but uses ragged right formatting in the description and page list columns.

```
9098 \newglossarystyle{altsuperragged4col}{%
```

Put the glossary in a supertabular environment with four columns and no head or tail:

```
9099 \renewenvironment{theglossary}{%
9100   {\tablehead{}\tabletail{}}%
9101   \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}l%
9102     >{\raggedright}p{\glspagelistwidth}}}%
9103   {\end{supertabular}}%
```

Do nothing at the start of the table:

```
9104 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9105 \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:

```

9106 \renewcommand{\glossentry}[2]{%
9107   \glentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
9108   \glossentrydesc{##1} &
9109   \glossentrysymbol{##1} & ##2\tabularnewline
9110 }%
```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```

9111 \renewcommand{\subglossentry}[3]{%
9112   &
9113   \glssubentryitem{##2}%
9114   \glstarget{##2}{\strut}\glossentrydesc{##2} &
9115   \glossentrysymbol{##2} & ##3\tabularnewline
9116 }%
```

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>)

```

9117 \ifglsgroupskip
9118   \renewcommand*{\glsgroupskip}{}%
9119 \else
9120   \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
9121 \fi
9122 }
```

agged4colheader The altsuperragged4colheader style is like the altsuperragged4col style but with a header row.

```

9123 \newglossarystyle{altsuperragged4colheader}{%
```

Base it on the glostylealtsuperragged4col style:

```

9124 \setglossarystyle{altsuperragged4col}{%
```

Put the glossary in a supertabular environment with four columns, a header and no tail:

```

9125 \renewenvironment{theglossary}%
9126   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
9127     \bfseries\symbolname &
9128     \bfseries\pagelistname\tabularnewline}\tabletail{}}%
9129   \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}l%
9130     >{\raggedright}p{\glspagelistwidth}}}%
9131   {\end{supertabular}}%
9132 }
```

agged4colborder The altsuperragged4colborder style is like the altsuperragged4col style but with a border.

```

9133 \newglossarystyle{altsuperragged4colborder}{%
```

Base it on the glostylealtsuperragged4col style:

```

9134 \setglossarystyle{altsuper4col}{%
```

Put the glossary in a supertabular environment with four columns and a horizontal line in the head and tail:

```

9135 \renewenvironment{theglossary}{%
```

```

9136   {\tablehead{\hline}\tabletail{\hline}%
9137   \begin{supertabular}%
9138     {||>{\raggedright}p{\glsgdescwidth}||}%
9139     >{\raggedright}p{\glspagelistwidth}||}%
9140   {\end{supertabular}}}%
9141 }

```

`colheaderborder` The `altsuperragged4colheaderborder` style is like the `altsuperragged4col` style but with a header and border.

```

9142 \newglossarystyle{altsuperragged4colheaderborder}{%

```

Base it on the `glostylealtsuperragged4col` style:

```

9143 \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:

```

9144 \renewenvironment{theglossary}%
9145   {\tablehead{\hline
9146     \bfseries\entryname &
9147     \bfseries\descriptionname &
9148     \bfseries\symbolname &
9149     \bfseries\pagelistname\tabularnewline\hline}%
9150   \tabletail{\hline}%
9151   \begin{supertabular}%
9152     {||>{\raggedright}p{\glsgdescwidth}||}%
9153     >{\raggedright}p{\glspagelistwidth}||}%
9154   {\end{supertabular}}}%
9155 }

```

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.

```

9156 \ProvidesPackage{glossary-tree}[2016/12/16 v4.27 (NLCT)]

```

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```

9157 \providecommand{\indexspace}{%
9158   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
9159 }

```

`\glstreenamefmt` Format used to display the name in the tree styles. (This may be counteracted by `\glsglfont`.) This command was previously also used to format the group headings.

```

9160 \newcommand*{\glstreenamefmt}[1]{\textbf{#1}}

```

`egroupheaderfmt` 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.

```
9161 \newcommand*{\glstreegroupheaderfmt}[1]{\glstreenamefmt{#1}}
```

`eenavigationfmt` Format used to display the navigation header in the tree styles.

```
9162 \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.

```
9163 \ifdef\@idxitem
9164 {\newcommand{\glstreeitem}{\@idxitem}}
9165 {\newcommand{\glstreeitem}{\par\hangindent40\p@}}
```

`\glstreesubitem` Level 1 item used in index style.

```
9166 \ifdef\subitem
9167 {\let\glstreesubitem\subitem}
9168 {\newcommand\glstreesubitem{\glstreeitem\hspace*{20\p@}}}
```

`streesubsubitem` Level 1 item used in index style.

```
9169 \ifdef\subsubitem
9170 {\let\glstreesubsubitem\subsubitem}
9171 {\newcommand\glstreesubsubitem{\glstreeitem\hspace*{30\p@}}}
```

`\glstreepredesc` Allow the user to adjust the space before the description (except for the `alttree` style).

```
9172 \newcommand{\glstreepredesc}{\space}
```

`reechildpredesc` Allow the user to adjust the space before the description for sub-entries (except for the `treenoname` and `alttree` style).

```
9173 \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.

```
9174 \newglossarystyle{index}{%
```

Set the paragraph indentation and skip and define `\item` to be the same as that used by `theindex`:

```
9175 \renewenvironment{theglossary}%
9176 {\setlength{\parindent}{0pt}%
9177 \setlength{\parskip}{0pt plus 0.3pt}%
9178 \let\item\glstreeitem
9179 \let\subitem\glstreesubitem
9180 \let\subsubitem\glstreesubsubitem
9181 }%
```


9182 {\par}%

Do nothing at the start of the environment:

9183 \renewcommand*{\glossaryheader}{}%

No group headers:

9184 \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.

9185 \renewcommand*{\glossentry}[2]{%

9186 \item\glstentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%

9187 \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%

9188 \glstreepredesc \glossentrydesc{##1}\glspostdescription\space ##2%

9189 }%

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.

9190 \renewcommand{\subglossentry}[3]{%

9191 \ifcase##1\relax

9192 % level 0

9193 \item

9194 \or

9195 % level 1

9196 \subitem

9197 \glssubentryitem{##2}%

9198 \else

9199 % all other levels

9200 \subsubitem

9201 \fi

9202 \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%

9203 \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%

9204 \glstreechildpredesc\glossentrydesc{##2}\glspostdescription\space ##3%

9205 }%

Vertical gap between groups is the same as that used by indices:

9206 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}}

indexgroup The indexgroup style is like the index style but has headings.

9207 \newglossarystyle{indexgroup}{%

Base it on the glostyleindex style:

9208 \setglossarystyle{index}%

Add a heading for each group. This puts the group's title in bold followed by a vertical gap.

9209 \renewcommand*{\glsgroupheading}[1]{%

9210 \item\glstreegroupheaderfmt{\glsgrouptitle{##1}}%

9211 \indexspace

9212 }%

9213 }

indexhypergroup The indexhypergroup style is like the indexgroup style but has hyper navigation.

```
9214 \newglossarystyle{indexhypergroup}{%
```

Base it on the glostyleindex style:

```
9215 \setglossarystyle{index}{%
```

Put navigation links to the groups at the start of the glossary:

```
9216 \renewcommand*{\glossaryheader}{%
```

```
9217 \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.

```
9218 \renewcommand*{\glsgroupheading}[1]{%
```

```
9219 \item\glstreegroupheaderfmt
```

```
9220 {\glsnahypertarget{##1}{\glsgrouptitle{##1}}}%
```

```
9221 \indexspace}%
```

```
9222 }
```

tree The tree glossary style is similar in style to the index style, but can have arbitrary levels.

```
9223 \newglossarystyle{tree}{%
```

Set the paragraph indentation and skip:

```
9224 \renewenvironment{theglossary}{%
```

```
9225 {\setlength{\parindent}{0pt}%
```

```
9226 \setlength{\parskip}{0pt plus 0.3pt}}%
```

```
9227 {}%
```

Do nothing at the start of the theglossary environment:

```
9228 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9229 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries: name in bold, followed by symbol in brackets (if it exists), the description and the page list:

```
9230 \renewcommand{\glossentry}[2]{%
```

```
9231 \hangindent0pt\relax
```

```
9232 \parindent0pt\relax
```

```
9233 \glstryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
```

```
9234 \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
```

```
9235 \glstreepredesc\glossentrydesc{##1}\glspostdescription\space##2\par
```

```
9236 }%
```

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.

```
9237 \renewcommand{\subglossentry}[3]{%
```

```
9238 \hangindent##1\glstreeindent\relax
```

```
9239 \parindent##1\glstreeindent\relax
```

```
9240 \ifnum##1=1\relax
```

```
9241 \glssubentryitem{##2}%
```

```
9242 \fi
```

```
9243 \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
```

```

9244 \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
9245 \glstreechildpredesc\glossentrydesc{##2}\glspostdescription\space ##3\par
9246 }%

```

Vertical gap between groups is the same as that used by indices:

```

9247 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}}

```

treegroup Like the tree style but the glossary groups have headings.

```

9248 \newglossarystyle{treegroup}{%

```

Base it on the `glostyletree` style:

```

9249 \setglossarystyle{tree}%

```

Each group has a heading (in bold) followed by a vertical gap):

```

9250 \renewcommand{\glsgroupheading}[1]{\par
9251 \noindent\glstreegroupheaderfmt{\glsgrouptitle{##1}}\par
9252 \indexspace}%
9253 }

```

treehypergroup The `treehypergroup` style is like the `treegroup` style, but has a set of links to the groups at the start of the glossary.

```

9254 \newglossarystyle{treehypergroup}{%

```

Base it on the `glostyletree` style:

```

9255 \setglossarystyle{tree}%

```

Put navigation links to the groups at the start of the `theglossary` environment:

```

9256 \renewcommand*{\glossaryheader}{%
9257 \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%

```

Each group has a heading (in bold with a target) followed by a vertical gap):

```

9258 \renewcommand*{\glsgroupheading}[1]{%
9259 \par\noindent
9260 \glstreegroupheaderfmt
9261 {\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}\par
9262 \indexspace}%
9263 }

```

\glstreeindent Length governing left indent for each level of the tree style.

```

9264 \newlength\glstreeindent
9265 \setlength{\glstreeindent}{10pt}

```

treenoname The `treenoname` glossary style is like the tree style, but doesn't print the name or symbol for sub-levels.

```

9266 \newglossarystyle{treenoname}{%

```

Set the paragraph indentation and skip:

```

9267 \renewenvironment{theglossary}%
9268 {\setlength{\parindent}{0pt}%
9269 \setlength{\parskip}{0pt plus 0.3pt}}%
9270 {}%

```

No header:

```
9271 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9272 \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.

```
9273 \renewcommand{\glossentry}[2]{%
9274   \hangindent0pt\relax
9275   \parindent0pt\relax
9276   \glstentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
9277   \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
9278   \glstreepredesc\glossentrydesc{##1}\glspostdescription\space##2\par
9279 }%
```

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.

```
9280 \renewcommand{\subglossentry}[3]{%
9281   \hangindent##1\glstreeindent\relax
9282   \parindent##1\glstreeindent\relax
9283   \ifnum##1=1\relax
9284     \glssubentryitem{##2}%
9285     \fi
9286     \glstarget{##2}{\strut}%
9287     \glossentrydesc{##2}\glspostdescription\space##3\par
9288 }%
```

Vertical gap between groups is the same as that used by indices:

```
9289 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
9290 }
```

treenonamegroup Like the `treenoname` style but the glossary groups have headings.

```
9291 \newglossarystyle{treenonamegroup}{%
```

Base it on the `glostyletreenoname` style:

```
9292 \setglossarystyle{treenoname}%
```

Give each group a heading:

```
9293 \renewcommand{\glsgroupheading}[1]{\par
9294   \noindent\glstreegroupheaderfmt
9295   {\glsgetgrouptitle{##1}}\par\indexspace}%
9296 }
```

onamehypergroup The `treenonamehypergroup` style is like the `treenonamegroup` style, but has a set of links to the groups at the start of the glossary.

```
9297 \newglossarystyle{treenonamehypergroup}{%
```

Base it on the `glostyletreenoname` style:

```
9298 \setglossarystyle{treenoname}%
```

Put navigation links to the groups at the start of the theglossary environment:

```
9299 \renewcommand*{\glossaryheader}{%
9300   \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
9301 \renewcommand*{\glsgroupheading}[1]{%
9302   \par\noindent
9303   \glstreegroupheaderfmt
9304     {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
9305   \indexspace}%
9306 }
```

esttoplevelname Find the widest name over all parentless entries in the given glossary or glossaries.

```
9307 \newrobustcmd*{\glsfindwidesttoplevelname}[1][\@glo@types]{%
9308   \dimen@=0pt\relax
9309   \gls@tmplen=0pt\relax
9310   \forallglossaries[#1]{\@gls@type}%
9311   {%
9312     \forglsentries[\@gls@type]{\@glo@label}%
9313     {%
9314       \ifglshasparent{\@glo@label}%
9315       }%
9316     {%
9317       \settowidth{\dimen@}%
9318         {\glstreenamefmt{\glsentryname{\@glo@label}}}%
9319       \ifdim\dimen@>\gls@tmplen
9320         \gls@tmplen=\dimen@
9321       \letcs{\@glswidestname}{glo@\glsdetoklabel{\@glo@label}@name}%
9322       \fi
9323     }%
9324   }%
9325 }%
9326 }
```

\glssetwidest `\glssetwidest[<level>]{<text>}` sets the widest text for the given level. It is used by the alt-tree glossary styles to determine the indentation of each level.

```
9327 \newcommand*{\glssetwidest}[2][0]{%
9328   \expandafter\def\csname @glswidestname\romannumeral#1\endcsname{%
9329     #2}%
9330 }
```

\@glswidestname Initialise \@glswidestname.

```
9331 \newcommand*{\@glswidestname}{}
```

\glstreenamebox Used by the alttree style to create the box for the name and associated information.

```
9332 \newcommand*{\glstreenamebox}[2]{%
9333   \makebox[#1][l]{#2}%
9334 }
```

alttree The alttree glossary style is similar in style to the tree style, but the indentation is obtained from the width of `\@glswidestname` which is set using `\glssetwidest`.

```
9335 \newglossarystyle{alttree}{%
```

Redefine theglossary environment.

```
9336 \renewenvironment{theglossary}{%
```

```
9337   {\def\@gls@prevlevel{-1}%
```

```
9338     \mbox{}\par}%
```

```
9339   {\par}%
```

Set the header and group headers to nothing.

```
9340 \renewcommand*{\glossaryheader}{}%
```

```
9341 \renewcommand*{\glsgroupheading}[1]{}%
```

Redefine the way that the level 0 entries are displayed.

```
9342 \renewcommand{\glossentry}[2]{%
```

```
9343   \ifnum\@gls@prevlevel=0\relax
```

```
9344   \else
```

Find out how big the indentation should be by measuring the widest entry.

```
9345     \settowidth{\glstreeindent}{\glstreenamfmt{\@glswidestname\space}}%
```

```
9346   \fi
```

Set the hangindent and paragraph indent.

```
9347   \hangindent\glstreeindent
```

```
9348   \parindent\glstreeindent
```

Put the name to the left of the paragraph block.

```
9349   \makebox[0pt][r]{\glstreenambox{\glstreeindent}{%
```

```
9350     \glsentryitem{##1}\glstreenamfmt{\glstarget{##1}{\glossentryname{##1}}}}%
```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
9351   \ifglshassymbol{##1}{(\glossentrysymbol{##1})\space}{}%
```

Do the description followed by the description terminator and location list.

```
9352   \glossentrydesc{##1}\glspostdescription \space ##2\par
```

Set the previous level to 0.

```
9353   \def\@gls@prevlevel{0}%
```

```
9354 }%
```

Redefine the way sub-entries are displayed.

```
9355 \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.

```
9356   \ifnum##1=1\relax
```

```
9357     \glssubentryitem{##2}%
```

```
9358   \fi
```

If the level hasn't changed, keep the same settings, otherwise adjust `\glstreeindent` accordingly.

```
9359   \ifnum\@gls@prevlevel=##1\relax
```

```
9360   \else
```

Compute the widest entry for this level, or for level 0 if not defined for this level. Store in `\gls@tmplen`

```

9361      \@ifundefined{glswidestname\romannumeral##1}{%
9362          \settowidth{\gls@tmplen}{\glstreenamefmt{\glswidestname\space}}{%
9363          \settowidth{\gls@tmplen}{\glstreenamefmt{%
9364              \csname @glswidestname\romannumeral##1\endcsname\space}}}%

```

Determine if going up or down a level

```

9365      \ifnum\@gls@prevlevel<##1\relax

```

Depth has increased, so add the width of the widest entry to `\glstreeindent`.

```

9366          \setlength\glstreeindent\gls@tmplen
9367          \addtolength\glstreeindent\parindent
9368          \parindent\glstreeindent
9369      \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`.

```

9370          \@ifundefined{glswidestname\romannumeral\@gls@prevlevel}{%
9371              \settowidth{\glstreeindent}{\glstreenamefmt{%
9372                  \glswidestname\space}}}%
9373          \settowidth{\glstreeindent}{\glstreenamefmt{%
9374              \csname @glswidestname\romannumeral\@gls@prevlevel
9375                  \endcsname\space}}}%

```

Subtract this length from the previous level's paragraph indent and set to `\glstreeindent`.

```

9376          \addtolength\parindent{-\glstreeindent}%
9377          \setlength\glstreeindent\parindent
9378      \fi
9379  \fi

```

Set the hanging indentation.

```

9380      \hangindent\glstreeindent

```

Put the name to the left of the paragraph block

```

9381      \makebox[0pt][r]{\glstreenamebox{\gls@tmplen}{%
9382          \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}}}%

```

If the symbol is missing, ignore it, otherwise put it in brackets.

```

9383      \ifglshassymbol{##2}{(\glossentrysymbol{##2})\space}{}%

```

Do the description followed by the description terminator and location list.

```

9384      \glossentrydesc{##2}\glspostdescription\space ##3\par

```

Set the previous level macro to the current level.

```

9385      \def\@gls@prevlevel{##1}%
9386  }%

```

Vertical gap between groups is the same as that used by indices:

```

9387      \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
9388  }

```

alttreegroup Like the **alttree** style but the glossary groups have headings.

```
9389 \newglossarystyle{alttreegroup}{%  
    Base it on the glostylealttree style:  
9390 \setglossarystyle{alttree}%  
    Give each group a heading.  
9391 \renewcommand{\glsgroupheading}[1]{\par  
9392 \def\@gls@prevlevel{-1}%  
9393 \hangindent0pt\relax  
9394 \parindent0pt\relax  
9395 \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}%  
9396 \par\indexspace}%  
9397 }
```

alttreehypergroup The **alttreehypergroup** style is like the **alttreegroup** style, but has a set of links to the groups at the start of the glossary.

```
9398 \newglossarystyle{alttreehypergroup}{%  
    Base it on the glostylealttree style:  
9399 \setglossarystyle{alttree}%  
    Put the navigation links in the header  
9400 \renewcommand*{\glossaryheader}{%  
9401 \par  
9402 \def\@gls@prevlevel{-1}%  
9403 \hangindent0pt\relax  
9404 \parindent0pt\relax  
9405 \glstreenavigationfmt{\glsnavigation}\par\indexspace}%  
    Put a hypertarget at the start of each group  
9406 \renewcommand*{\glsgroupheading}[1]{%  
9407 \par  
9408 \def\@gls@prevlevel{-1}%  
9409 \hangindent0pt\relax  
9410 \parindent0pt\relax  
9411 \glstreegroupheaderfmt  
9412 {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par  
9413 \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.

```
9414 \NeedsTeXFormat{LaTeX2e}
9415 \ProvidesPackage{glossaries-compatible-207}[2016/12/16 v4.27 (NLCT)]
```

AddXdyAttribute Adds an attribute in old format.

```
9416 \ifglxsindy
9417   \renewcommand*\GlsAddXdyAttribute[1]{%
9418     \edef\@xdyattributes{\@xdyattributes ^^J \string"#1\string"}%
9419     \expandafter\toks@\expandafter{\@xdylocref}%
9420     \edef\@xdylocref{\the\toks@ ^^J}%
9421     (markup-locref
9422     :open \string"\string~n\string\setentrycounter
9423       {\noexpand\glscounter}%
9424       \expandafter\string\csname#1\endcsname
9425       \expandafter\@gobble\string\{\string" ^^J
9426       :close \string"\expandafter\@gobble\string\}\string" ^^J
9427       :attr \string"#1\string"))}}
```

Only has an effect before `\writeist`:

```
9428 \fi
```

sAddXdyCounters

```
9429 \renewcommand*\GlsAddXdyCounters[1]{%
9430   \GlossariesWarning{\string\GlsAddXdyCounters\space not available
9431     in compatibility mode.}%
9432 }
```

Add predefined attributes

```
9433 \GlsAddXdyAttribute{glsnumberformat}
9434 \GlsAddXdyAttribute{textrm}
9435 \GlsAddXdyAttribute{textsf}
9436 \GlsAddXdyAttribute{texttt}
9437 \GlsAddXdyAttribute{textbf}
9438 \GlsAddXdyAttribute{textmd}
9439 \GlsAddXdyAttribute{textit}
9440 \GlsAddXdyAttribute{textup}
9441 \GlsAddXdyAttribute{textsl}
```

```

9442 \GlsAddXdyAttribute{textsc}
9443 \GlsAddXdyAttribute{emph}
9444 \GlsAddXdyAttribute{glshypernumber}
9445 \GlsAddXdyAttribute{hyperrm}
9446 \GlsAddXdyAttribute{hypersf}
9447 \GlsAddXdyAttribute{hypertt}
9448 \GlsAddXdyAttribute{hyperbf}
9449 \GlsAddXdyAttribute{hypermd}
9450 \GlsAddXdyAttribute{hyperit}
9451 \GlsAddXdyAttribute{hyperup}
9452 \GlsAddXdyAttribute{hypersl}
9453 \GlsAddXdyAttribute{hypersc}
9454 \GlsAddXdyAttribute{hyperemph}

```

sAddXdyLocation Restore v2.07 definition:

```

9455 \ifglxindy
9456 \renewcommand*{\GlsAddXdyLocation}[2]{%
9457 \edef\@xdyuserlocationdefs{%
9458 \@xdyuserlocationdefs ^^J%
9459 (define-location-class \string"#1\string"^^J\space\space
9460 \space(#2))
9461 }%
9462 \edef\@xdyuserlocationnames{%
9463 \@xdyuserlocationnames^^J\space\space\space
9464 \string"#1\string"}%
9465 }
9466 \fi

```

\@do@wrglossary

```

9467 \renewcommand{\@do@wrglossary}[1]{%
  Determine whether to use xindy or makeindex syntax
9468 \ifglxindy
  Need to determine if the formatting information starts with a ( or ) indicating a range.
9469 \expandafter\@glo@check@mkidxrangechar\@glsnumberformat\@nil
9470 \def\@glo@range{}%
9471 \expandafter\if\@glo@prefix(\relax
9472 \def\@glo@range{:open-range}%
9473 \else
9474 \expandafter\if\@glo@prefix)\relax
9475 \def\@glo@range{:close-range}%
9476 \fi
9477 \fi

  Get the location and escape any special characters
9478 \protected@edef\@glslocref{\theglsentrycounter}%
9479 \@gls@checkmkidxchars\@glslocref

  Write to the glossary file using xindy syntax.
9480 \glossary[\csname glo@#1@type\endcsname]{%

```

```

9481 (indexentry :tkey (\csname glo@#1@index\endcsname)
9482   :locoref \string"\@glslocoref\string" %
9483   :attr \string"\@glo@suffix\string" \@glo@range
9484 )
9485 }%
9486 \else

```

Convert the format information into the format required for makeindex

```

9487 \@set@glo@numformat\@glo@numfmt\@gls@counter\@glsnumberformat

```

Write to the glossary file using makeindex syntax.

```

9488 \glossary[\csname glo@#1@type\endcsname]{%
9489 \string\glossaryentry{\csname glo@#1@index\endcsname
9490   \@gls@encapchar\@glo@numfmt}{\theglsentrycounter}}%
9491 \fi
9492 }

```

t@glo@numformat Only had 3 arguments in v2.07

```

9493 \def\@set@glo@numformat#1#2#3{%
9494   \expandafter\@glo@check@mkidxrangechar#3\@nil
9495   \protected@edef#1{%
9496     \@glo@prefix setentrycounter[]{\#2}%
9497     \expandafter\string\csname\@glo@suffix\endcsname
9498   }%
9499   \@gls@checkmkidxchars#1%
9500 }

```

\writeist Redefine \writeist back to the way it was in v2.07, but change \istfile to \glswrite.

```

9501 \ifglxsindy
9502   \def\writeist{%
9503     \openout\glswrite=\istfilename
9504     \write\glswrite{;; xindy style file created by the glossaries
9505       package in compatible-2.07 mode}%
9506     \write\glswrite{;; for document '\jobname' on
9507       \the\year-\the\month-\the\day}%
9508     \write\glswrite{^^J; required styles^^J}
9509     \@for\@xdystyle:=\@xdyrequiredstyles\do{%
9510       \ifx\@xdystyle\@empty
9511       \else
9512         \protected@write\glswrite{{(require
9513           \string"\@xdystyle.xdy\string")}}%
9514       \fi
9515     }%
9516     \write\glswrite{^^J%
9517       ; list of allowed attributes (number formats)^^J}%
9518     \write\glswrite{(define-attributes ((\@xdyattributes)))}%
9519     \write\glswrite{^^J; user defined alphabets^^J}%
9520     \write\glswrite{\@xdyuseralphabets}%
9521     \write\glswrite{^^J; location class definitions^^J}%
9522     \protected@edef\@gls@roman{\@roman{0}\string"

```

```

9523     \string"roman-numbers-lowercase\string" :sep \string"}}%
9524 \@onelevel@sanitize\@gls@roman
9525 \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
9526     :sep \string"}}%
9527 \@onelevel@sanitize\@tmp
9528 \ifx\@tmp\@gls@roman
9529     \write\glswrite{(define-location-class
9530         \string"roman-page-numbers\string"^^J\space\space\space
9531         (\string"roman-numbers-lowercase\string")
9532         :min-range-length \@glsminrange)}}%
9533 \else
9534     \write\glswrite{(define-location-class
9535         \string"roman-page-numbers\string"^^J\space\space\space
9536         (:sep "\@gls@roman")
9537         :min-range-length \@glsminrange)}}%
9538 \fi
9539 \write\glswrite{(define-location-class
9540     \string"Roman-page-numbers\string"^^J\space\space\space
9541     (\string"roman-numbers-uppercase\string")
9542     :min-range-length \@glsminrange)}}%
9543 \write\glswrite{(define-location-class
9544     \string"arabic-page-numbers\string"^^J\space\space\space
9545     (\string"arabic-numbers\string")
9546     :min-range-length \@glsminrange)}}%
9547 \write\glswrite{(define-location-class
9548     \string"alpha-page-numbers\string"^^J\space\space\space
9549     (\string"alpha\string")
9550     :min-range-length \@glsminrange)}}%
9551 \write\glswrite{(define-location-class
9552     \string"Alpha-page-numbers\string"^^J\space\space\space
9553     (\string"ALPHA\string")
9554     :min-range-length \@glsminrange)}}%
9555 \write\glswrite{(define-location-class
9556     \string"Appendix-page-numbers\string"^^J\space\space\space
9557     (\string"ALPHA\string"
9558     :sep \string"\@glsAlphacompositor\string"
9559     \string"arabic-numbers\string")
9560     :min-range-length \@glsminrange)}}%
9561 \write\glswrite{(define-location-class
9562     \string"arabic-section-numbers\string"^^J\space\space\space
9563     (\string"arabic-numbers\string"
9564     :sep \string"\glscompositor\string"
9565     \string"arabic-numbers\string")
9566     :min-range-length \@glsminrange)}}%
9567 \write\glswrite{^^J; user defined location classes}%
9568 \write\glswrite{\@xdyuserlocationdefs}%
9569 \write\glswrite{^^J; define cross-reference class^^J}%
9570 \write\glswrite{(define-crossref-class \string"see\string"
9571     :unverified )}%

```

```

9572 \write\glswrite{(markup-crossref-list
9573 :class \string"see\string"^^J\space\space\space
9574 :open \string"\string\glssseeformat\string"
9575 :close \string"{}\string")}%
9576 \write\glswrite{^^J; define the order of the location classes}%
9577 \write\glswrite{(define-location-class-order
9578 (\@xdylocationclassorder))}%
9579 \write\glswrite{^^J; define the glossary markup^^J}%
9580 \write\glswrite{(markup-index^^J\space\space\space
9581 :open \string"\string
9582 \glossarysection[\string\glossarytoctitle]{\string
9583 \glossarytitle}\string\glossarypreamble\string~n\string\begin
9584 {theglossary}\string\glossaryheader\string~n\string" ^^J\space
9585 \space\space:close \string"\expandafter\@gobble
9586 \string%\string~n\string
9587 \end{theglossary}\string\glossarypostamble
9588 \string~n\string" ^^J\space\space\space
9589 :tree)}}%
9590 \write\glswrite{(markup-letter-group-list
9591 :sep \string"\string\glsgroupskip\string~n\string")}%
9592 \write\glswrite{(markup-indexentry
9593 :open \string"\string\relax \string\glresetentrylist
9594 \string~n\string")}%
9595 \write\glswrite{(markup-locclass-list :open
9596 \string"\glsoopenbrace\string\glossaryentrynumbers
9597 \glsoopenbrace\string\relax\space \string"^^J\space\space\space
9598 :sep \string", \string"
9599 :close \string"\glsclosebrace\glsclosebrace\string")}%
9600 \write\glswrite{(markup-locref-list
9601 :sep \string"\string\delimN\space\string")}%
9602 \write\glswrite{(markup-range
9603 :sep \string"\string\delimR\space\string")}%
9604 \@onelevel@sanitize\gls@suffixF
9605 \@onelevel@sanitize\gls@suffixFF
9606 \ifx\gls@suffixF\@empty
9607 \else
9608 \write\glswrite{(markup-range
9609 :close "\gls@suffixF" :length 1 :ignore-end)}%
9610 \fi
9611 \ifx\gls@suffixFF\@empty
9612 \else
9613 \write\glswrite{(markup-range
9614 :close "\gls@suffixFF" :length 2 :ignore-end)}%
9615 \fi
9616 \write\glswrite{^^J; define format to use for locations^^J}%
9617 \write\glswrite{\@xdylocref}%
9618 \write\glswrite{^^J; define letter group list format^^J}%
9619 \write\glswrite{(markup-letter-group-list
9620 :sep \string"\string\glsgroupskip\string~n\string")}%

```

```

9621 \write\glswrite{^^J; letter group headings^^J}%
9622 \write\glswrite{(markup-letter-group
9623   :open-head \string"\string\glsgroupheading
9624   \glsoopenbrace\string"^^J\space\space\space
9625   :close-head \string"\glsclosebrace\string")}%
9626 \write\glswrite{^^J; additional letter groups^^J}%
9627 \write\glswrite{\@xdylettergroups}%
9628 \write\glswrite{^^J; additional sort rules^^J}
9629 \write\glswrite{\@xdysortrules}%
9630 \noist}
9631 \else
9632 \edef\@gls@actualchar{\string?}
9633 \edef\@gls@encapchar{\string|}
9634 \edef\@gls@levelchar{\string!}
9635 \edef\@gls@quotechar{\string"}
9636 \def\writeist{\relax
9637   \openout\glswrite=\istfilename
9638   \write\glswrite{\expandafter\@gobble\string\% makeindex style file
9639     created by the glossaries package}
9640   \write\glswrite{\expandafter\@gobble\string\% for document
9641     '\jobname' on \the\year-\the\month-\the\day}
9642   \write\glswrite{actual '\@gls@actualchar'}
9643   \write\glswrite{encap '\@gls@encapchar'}
9644   \write\glswrite{level '\@gls@levelchar'}
9645   \write\glswrite{quote '\@gls@quotechar'}
9646   \write\glswrite{keyword \string"\string\glossaryentry\string"}
9647   \write\glswrite{preamble \string"\string\glossarysection[\string
9648     \glossarytoctitle]{\string\glossarytitle}\string
9649     \glossarypreamble\string\n\string\begin{theglossary}\string
9650     \glossaryheader\string\n\string"}
9651   \write\glswrite{postamble \string"\string%\string\n\string
9652     \end{theglossary}\string\glossarypostamble\string\n
9653     \string"}
9654   \write\glswrite{group_skip \string"\string\glsgroupskip\string\n
9655     \string"}
9656   \write\glswrite{item_0 \string"\string%\string\n\string"}
9657   \write\glswrite{item_1 \string"\string%\string\n\string"}
9658   \write\glswrite{item_2 \string"\string%\string\n\string"}
9659   \write\glswrite{item_01 \string"\string%\string\n\string"}
9660   \write\glswrite{item_x1
9661     \string"\string\relax \string\glsresetentrylist\string\n
9662     \string"}
9663   \write\glswrite{item_12 \string"\string%\string\n\string"}
9664   \write\glswrite{item_x2
9665     \string"\string\relax \string\glsresetentrylist\string\n
9666     \string"}
9667   \write\glswrite{delim_0 \string"\string\{\string
9668     \glossaryentrynumbers\string\{\string\relax \string"}
9669   \write\glswrite{delim_1 \string"\string\{\string

```

```

9670      \glossaryentrynumbers\string\{\string\relax \string}
9671      \write\glswrite{delim_2 \string"\string\{\string
9672      \glossaryentrynumbers\string\{\string\relax \string}
9673      \write\glswrite{delim_t \string"\string\}\string\}\string}
9674      \write\glswrite{delim_n \string"\string\delimN \string}
9675      \write\glswrite{delim_r \string"\string\delimR \string}
9676      \write\glswrite{headings_flag 1}
9677      \write\glswrite{heading_prefix
9678      \string"\string\glsgroupheading\string\{\string}
9679      \write\glswrite{heading_suffix
9680      \string"\string\}\string\relax
9681      \string\glsgroupresetentrylist \string}
9682      \write\glswrite{symhead_positive \string"glssymbols\string}
9683      \write\glswrite{numhead_positive \string"glslnumbers\string}
9684      \write\glswrite{page_compositor \string"glscpositor\string}
9685      \@gls@escbsdq\gls@suffixF
9686      \@gls@escbsdq\gls@suffixFF
9687      \ifx\gls@suffixF\@empty
9688      \else
9689      \write\glswrite{suffix_2p \string"\gls@suffixF\string}
9690      \fi
9691      \ifx\gls@suffixFF\@empty
9692      \else
9693      \write\glswrite{suffix_3p \string"\gls@suffixFF\string}
9694      \fi
9695      \noist
9696    }
9697  \fi

```

\noist

```

9698 \renewcommand*{\noist}{\let\writeist\relax}

```

4.2 glossaries-compatible-307

```

9699 \NeedsTeXFormat{LaTeX2e}
9700 \ProvidesPackage{glossaries-compatible-307}[2016/12/16 v4.27 (NLCT)]

```

Compatibility macros for predefined glossary styles:

`\atglossarystyle` Defines a compatibility glossary style.

```

9701 \newcommand{\compatglossarystyle}[2]{%
9702   \ifcsundef{@glscompstyle@#1}%
9703   {%
9704     \csdef{@glscompstyle@#1}{#2}%
9705   }%
9706   {%
9707     \PackageError{glossaries}{Glossary compatibility style ‘#1’ is already defined}{}%
9708   }%
9709 }

```

Backward compatible inline style.

```

9710 \compatglossarystyle{inline}{%
9711   \renewcommand{\glossaryentryfield}[5]{%
9712     \glsinlinedopostchild
9713     \gls@inlinessep
9714     \def\glo@desc{##3}%
9715     \def\@no@post@desc{\nopostdesc}%
9716     \glsentryitem{##1}\glsinlinenameformat{##1}{##2}%
9717     \ifx\glo@desc\@no@post@desc
9718       \glsinlineemptydescformat{##4}{##5}%
9719     \else
9720       \ifstrepty{##3}%
9721         {\glsinlineemptydescformat{##4}{##5}}%
9722         {\glsinlinedescformat{##3}{##4}{##5}}%
9723     \fi
9724     \ifglshaschildren{##1}%
9725     {%
9726       \glsresetsubentrycounter
9727       \glsinlineparentchildseparator
9728       \def\gls@inlinesubsep{}%
9729       \def\gls@inlinepostchild{\glsinlinepostchild}%
9730     }%
9731   }%
9732   \def\gls@inlinesep{\glsinlineseparator}%
9733 }%
```

Sub-entries display description:

```

9734 \renewcommand{\glossarysubentryfield}[6]{%
9735   \gls@inlinesubsep%
9736   \glsinlinesubnameformat{##2}{##3}%
9737   \glssubentryitem{##2}\glsinlinesubdescformat{##4}{##5}{##6}%
9738   \def\gls@inlinesubsep{\glsinlinesubseparator}%
9739 }%
9740 }
```

Backward compatible list style.

```

9741 \compatglossarystyle{list}{%
9742   \renewcommand*\glossaryentryfield}[5]{%
9743     \item[\glsentryitem{##1}\glstarget{##1}{##2}]
9744       ##3\glspostdescription\space ##5}%
9745 }
```

Sub-entries continue on the same line:

```

9745 \renewcommand*\glossarysubentryfield}[6]{%
9746   \glssubentryitem{##2}%
9747   \glstarget{##2}{\strut}##4\glspostdescription\space ##6.}%
9748 }
```

Backward compatible listgroup style.

```

9749 \compatglossarystyle{listgroup}{%
9750   \csuse{@glscompstyle@list}%
9751 }%
```


Backward compatible listhypergroup style.

```
9752 \compatglossarystyle{listhypergroup}{%
9753   \csuse{@glscompstyle@list}%
9754 }%
```

Backward compatible altlist style.

```
9755 \compatglossarystyle{altlist}{%
9756   \renewcommand*{\glossaryentryfield}[5]{%
9757     \item[\glstentryitem{##1}\glstarget{##1}{##2}]%
9758     \mbox{}\par\nobreak\@afterheading
9759     ##3\glspostdescription\space ##5}%
9760   \renewcommand{\glossarysubentryfield}[6]{%
9761     \par
9762     \glssubentryitem{##2}%
9763     \glstarget{##2}{\strut}##4\glspostdescription\space ##6}%
9764 }%
```

Backward compatible altlistgroup style.

```
9765 \compatglossarystyle{altlistgroup}{%
9766   \csuse{@glscompstyle@altlist}%
9767 }%
```

Backward compatible altlisthypergroup style.

```
9768 \compatglossarystyle{altlisthypergroup}{%
9769   \csuse{@glscompstyle@altlist}%
9770 }%
```

Backward compatible listdotted style.

```
9771 \compatglossarystyle{listdotted}{%
9772   \renewcommand*{\glossaryentryfield}[5]{%
9773     \item[\makebox[\glslistdottedwidth][l]{%
9774       \glstentryitem{##1}\glstarget{##1}{##2}%
9775       \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##3}%
9776   \renewcommand*{\glossarysubentryfield}[6]{%
9777     \item[\makebox[\glslistdottedwidth][l]{%
9778       \glssubentryitem{##2}%
9779       \glstarget{##2}{##3}%
9780       \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##4}%
9781 }%
```

Backward compatible sublistdotted style.

```
9782 \compatglossarystyle{sublistdotted}{%
9783   \csuse{@glscompstyle@listdotted}%
9784   \renewcommand*{\glossaryentryfield}[5]{%
9785     \item[\glstentryitem{##1}\glstarget{##1}{##2}]}%
9786 }%
```

Backward compatible long style.

```
9787 \compatglossarystyle{long}{%
9788   \renewcommand*{\glossaryentryfield}[5]{%
9789     \glstentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
9790   \renewcommand*{\glossarysubentryfield}[6]{%
9791     \glssubentryitem{##2}\glstarget{##2}{##3} & ##4\glspostdescription\space ##5\\}%
9792 }%
```

```

9791      &
9792      \glssubentryitem{##2}%
9793      \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
9794 }%

```

Backward compatible longborder style.

```

9795 \compatglossarystyle{longborder}{%
9796   \csuse{@glscmpstyle@long}%
9797 }%

```

Backward compatible longheader style.

```

9798 \compatglossarystyle{longheader}{%
9799   \csuse{@glscmpstyle@long}%
9800 }%

```

Backward compatible longheaderborder style.

```

9801 \compatglossarystyle{longheaderborder}{%
9802   \csuse{@glscmpstyle@long}%
9803 }%

```

Backward compatible long3col style.

```

9804 \compatglossarystyle{long3col}{%
9805   \renewcommand*{\glossaryentryfield}[5]{%
9806     \glstarget{##1}{\glstarget{##1}{##2} & ##3 & ##5\\}%
9807     \renewcommand*{\glossarysubentryfield}[6]{%
9808       &
9809       \glssubentryitem{##2}%
9810       \glstarget{##2}{\strut}##4 & ##6\\}%
9811 }%

```

Backward compatible long3colborder style.

```

9812 \compatglossarystyle{long3colborder}{%
9813   \csuse{@glscmpstyle@long3col}%
9814 }%

```

Backward compatible long3colheader style.

```

9815 \compatglossarystyle{long3colheader}{%
9816   \csuse{@glscmpstyle@long3col}%
9817 }%

```

Backward compatible long3colheaderborder style.

```

9818 \compatglossarystyle{long3colheaderborder}{%
9819   \csuse{@glscmpstyle@long3col}%
9820 }%

```

Backward compatible long4col style.

```

9821 \compatglossarystyle{long4col}{%
9822   \renewcommand*{\glossaryentryfield}[5]{%
9823     \glstarget{##1}{\glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%
9824     \renewcommand*{\glossarysubentryfield}[6]{%
9825       &
9826       \glssubentryitem{##2}%

```

9827 \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%
 9828 }%

Backward compatible long4colheader style.

9829 \compatglossarystyle{long4colheader}{%
 9830 \csuse{@glscompstyle@long4col}%
 9831 }%

Backward compatible long4colborder style.

9832 \compatglossarystyle{long4colborder}{%
 9833 \csuse{@glscompstyle@long4col}%
 9834 }%

Backward compatible long4colheaderborder style.

9835 \compatglossarystyle{long4colheaderborder}{%
 9836 \csuse{@glscompstyle@long4col}%
 9837 }%

Backward compatible altlong4col style.

9838 \compatglossarystyle{altlong4col}{%
 9839 \csuse{@glscompstyle@long4col}%
 9840 }%

Backward compatible altlong4colheader style.

9841 \compatglossarystyle{altlong4colheader}{%
 9842 \csuse{@glscompstyle@long4col}%
 9843 }%

Backward compatible altlong4colborder style.

9844 \compatglossarystyle{altlong4colborder}{%
 9845 \csuse{@glscompstyle@long4col}%
 9846 }%

Backward compatible altlong4colheaderborder style.

9847 \compatglossarystyle{altlong4colheaderborder}{%
 9848 \csuse{@glscompstyle@long4col}%
 9849 }%

Backward compatible long style.

9850 \compatglossarystyle{longragged}{%
 9851 \renewcommand*{\glossaryentryfield}[5]{%
 9852 \glssentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
 9853 \tabularnewline}%
 9854 \renewcommand*{\glossarysubentryfield}[6]{%
 9855 &
 9856 \glssubentryitem{##2}%
 9857 \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
 9858 \tabularnewline}%
 9859 }%

Backward compatible longraggedborder style.

9860 \compatglossarystyle{longraggedborder}{%
 9861 \csuse{@glscompstyle@longragged}%
 9862 }%

Backward compatible longraggedheader style.

```
9863 \compatglossarystyle{longraggedheader}{%  
9864 \csuse{@glscompstyle@longragged}%  
9865 }%
```

Backward compatible longraggedheaderborder style.

```
9866 \compatglossarystyle{longraggedheaderborder}{%  
9867 \csuse{@glscompstyle@longragged}%  
9868 }%
```

Backward compatible longragged3col style.

```
9869 \compatglossarystyle{longragged3col}{%  
9870 \renewcommand*{\glossaryentryfield}[5]{%  
9871 \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%  
9872 \renewcommand*{\glossarysubentryfield}[6]{%  
9873 &  
9874 \glssubentryitem{##2}%  
9875 \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%  
9876 }%
```

Backward compatible longragged3colborder style.

```
9877 \compatglossarystyle{longragged3colborder}{%  
9878 \csuse{@glscompstyle@longragged3col}%  
9879 }%
```

Backward compatible longragged3colheader style.

```
9880 \compatglossarystyle{longragged3colheader}{%  
9881 \csuse{@glscompstyle@longragged3col}%  
9882 }%
```

Backward compatible longragged3colheaderborder style.

```
9883 \compatglossarystyle{longragged3colheaderborder}{%  
9884 \csuse{@glscompstyle@longragged3col}%  
9885 }%
```

Backward compatible altlongragged4col style.

```
9886 \compatglossarystyle{altlongragged4col}{%  
9887 \renewcommand*{\glossaryentryfield}[5]{%  
9888 \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%  
9889 \renewcommand*{\glossarysubentryfield}[6]{%  
9890 &  
9891 \glssubentryitem{##2}%  
9892 \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%  
9893 }%
```

Backward compatible altlongragged4colheader style.

```
9894 \compatglossarystyle{altlongragged4colheader}{%  
9895 \csuse{@glscompstyle@altlong4col}%  
9896 }%
```

Backward compatible altlongragged4colborder style.

```
9897 \compatglossarystyle{altlongragged4colborder}{%
```

```

9898 \csuse{@glscompstyle@altlong4col}%
9899 }%

```

Backward compatible altlongragged4colheaderborder style.

```

9900 \compatglossarystyle{altlongragged4colheaderborder}{%
9901 \csuse{@glscompstyle@altlong4col}%
9902 }%

```

Backward compatible index style.

```

9903 \compatglossarystyle{index}{%
9904 \renewcommand*{\glossaryentryfield}[5]{%
9905 \item\glsentryitem{##1}\textbf{\glstarget{##1}{##2}}%
9906 \ifx\relax##4\relax
9907 \else
9908 \space{##4}%
9909 \fi
9910 \space ##3\glspostdescription \space ##5}%
9911 \renewcommand*{\glossarysubentryfield}[6]{%
9912 \ifcase##1\relax
9913 % level 0
9914 \item
9915 \or
9916 % level 1
9917 \subitem
9918 \glssubentryitem{##2}%
9919 \else
9920 % all other levels
9921 \subsubitem
9922 \fi
9923 \textbf{\glstarget{##2}{##3}}%
9924 \ifx\relax##5\relax
9925 \else
9926 \space{##5}%
9927 \fi
9928 \space##4\glspostdescription\space ##6}%
9929 }%

```

Backward compatible indexgroup style.

```

9930 \compatglossarystyle{indexgroup}{%
9931 \csuse{@glscompstyle@index}%
9932 }%

```

Backward compatible indexhypergroup style.

```

9933 \compatglossarystyle{indexhypergroup}{%
9934 \csuse{@glscompstyle@index}%
9935 }%

```

Backward compatible tree style.

```

9936 \compatglossarystyle{tree}{%
9937 \renewcommand{\glossaryentryfield}[5]{%
9938 \hangindent0pt\relax

```

```

9939 \parindent0pt\relax
9940 \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
9941 \ifx\relax##4\relax
9942 \else
9943 \space(##4)%
9944 \fi
9945 \space ##3\glspostdescription \space ##5\par}%
9946 \renewcommand{\glossarysubentryfield}[6]{%
9947 \hangindent##1\glstreeindent\relax
9948 \parindent##1\glstreeindent\relax
9949 \ifnum##1=1\relax
9950 \glssubentryitem{##2}%
9951 \fi
9952 \textbf{\glstarget{##2}{##3}}%
9953 \ifx\relax##5\relax
9954 \else
9955 \space(##5)%
9956 \fi
9957 \space##4\glspostdescription\space ##6\par}%
9958 }%

```

Backward compatible treegroup style.

```

9959 \compatglossarystyle{treegroup}{%
9960 \csuse{@glscmpstyle@tree}%
9961 }%

```

Backward compatible treehypergroup style.

```

9962 \compatglossarystyle{treehypergroup}{%
9963 \csuse{@glscmpstyle@tree}%
9964 }%

```

Backward compatible treenoname style.

```

9965 \compatglossarystyle{treenoname}{%
9966 \renewcommand{\glossaryentryfield}[5]{%
9967 \hangindent0pt\relax
9968 \parindent0pt\relax
9969 \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
9970 \ifx\relax##4\relax
9971 \else
9972 \space(##4)%
9973 \fi
9974 \space ##3\glspostdescription \space ##5\par}%
9975 \renewcommand{\glossarysubentryfield}[6]{%
9976 \hangindent##1\glstreeindent\relax
9977 \parindent##1\glstreeindent\relax
9978 \ifnum##1=1\relax
9979 \glssubentryitem{##2}%
9980 \fi
9981 \glstarget{##2}{\strut}%
9982 ##4\glspostdescription\space ##6\par}%
9983 }%

```

Backward compatible treenonamegroup style.

```
9984 \compatglossarystyle{treenonamegroup}{%
9985   \csuse{@glscompstyle@treenoname}%
9986 }%
```

Backward compatible treenonamehypergroup style.

```
9987 \compatglossarystyle{treenonamehypergroup}{%
9988   \csuse{@glscompstyle@treenoname}%
9989 }%
```

Backward compatible alttree style.

```
9990 \compatglossarystyle{alttree}{%
9991   \renewcommand{\glossaryentryfield}[5]{%
9992     \ifnum\@gls@prevlevel=0\relax
9993     \else
9994       \settowidth{\glstreeindent}{\textbf{\@glswidestname\space}}%
9995       \hangindent\glstreeindent
9996       \parindent\glstreeindent
9997     \fi
9998     \makebox[0pt][r]{\makebox[\glstreeindent][l]{%
9999       \glssentryitem{##1}\textbf{\glstarget{##1}{##2}}}%
10000   \ifx\relax##4\relax
10001   \else
10002     (##4)\space
10003   \fi
10004   ##3\glspostdescription \space ##5\par
10005   \def\@gls@prevlevel{0}%
10006 }%
10007 \renewcommand{\glossarysubentryfield}[6]{%
10008   \ifnum##1=1\relax
10009     \glssubentryitem{##2}%
10010   \fi
10011   \ifnum\@gls@prevlevel=##1\relax
10012   \else
10013     \@ifundefined{@glswidestname\romannumeral##1}{%
10014       \settowidth{\gls@tmplen}{\textbf{\@glswidestname\space}}{%
10015       \settowidth{\gls@tmplen}{\textbf{%
10016         \csname @glswidestname\romannumeral##1\endcsname\space}}}%
10017     \ifnum\@gls@prevlevel<##1\relax
10018       \setlength\glstreeindent\gls@tmplen
10019       \addtolength\glstreeindent\parindent
10020       \parindent\glstreeindent
10021     \else
10022       \@ifundefined{@glswidestname\romannumeral\@gls@prevlevel}{%
10023         \settowidth{\glstreeindent}{\textbf{%
10024           \@glswidestname\space}}{%
10025         \settowidth{\glstreeindent}{\textbf{%
10026           \csname @glswidestname\romannumeral\@gls@prevlevel
10027             \endcsname\space}}}%
10028         \addtolength\parindent{-\glstreeindent}%

```

```

10029      \setlength\glstreeindent\parindent
10030      \fi
10031      \fi
10032      \hangindent\glstreeindent
10033      \makebox[0pt][r]{\makebox[\glstemplen][l]{%
10034        \textbf{\glstarget{##2}{##3}}}%
10035      \ifx##5\relax\relax
10036      \else
10037        (##5)\space
10038      \fi
10039      ##4\glspostdescription\space ##6\par
10040      \def\@gls@prevlevel{##1}%
10041    }%
10042 }%

```

Backward compatible alttreegroup style.

```

10043 \compatglossarystyle{alttreegroup}{%
10044   \csuse{@glscompstyle@almtree}%
10045 }%

```

Backward compatible almtreehypergroup style.

```

10046 \compatglossarystyle{almtreehypergroup}{%
10047   \csuse{@glscompstyle@almtree}%
10048 }%

```

Backward compatible mcolindex style.

```

10049 \compatglossarystyle{mcolindex}{%
10050   \csuse{@glscompstyle@index}%
10051 }%

```

Backward compatible mcolindexgroup style.

```

10052 \compatglossarystyle{mcolindexgroup}{%
10053   \csuse{@glscompstyle@index}%
10054 }%

```

Backward compatible mcolindexhypergroup style.

```

10055 \compatglossarystyle{mcolindexhypergroup}{%
10056   \csuse{@glscompstyle@index}%
10057 }%

```

Backward compatible mcoltree style.

```

10058 \compatglossarystyle{mcoltree}{%
10059   \csuse{@glscompstyle@tree}%
10060 }%

```

Backward compatible mcoltreegroup style.

```

10061 \compatglossarystyle{mcolindextreegroup}{%
10062   \csuse{@glscompstyle@tree}%
10063 }%

```

Backward compatible mcoltreehypergroup style.

```

10064 \compatglossarystyle{mcolindextreehypergroup}{%

```


10065 \csuse{@glscompstyle@tree}%
 10066 }%

Backward compatible mcoltreenoname style.

10067 \compatglossarystyle{mcoltreenoname}{%
 10068 \csuse{@glscompstyle@tree}%
 10069 }%

Backward compatible mcoltreenonamegroup style.

10070 \compatglossarystyle{mcoltreenonamegroup}{%
 10071 \csuse{@glscompstyle@tree}%
 10072 }%

Backward compatible mcoltreenonamehypergroup style.

10073 \compatglossarystyle{mcoltreenonamehypergroup}{%
 10074 \csuse{@glscompstyle@tree}%
 10075 }%

Backward compatible mcolalmtree style.

10076 \compatglossarystyle{mcolalmtree}{%
 10077 \csuse{@glscompstyle@almtree}%
 10078 }%

Backward compatible mcolalmtreegroup style.

10079 \compatglossarystyle{mcolalmtreegroup}{%
 10080 \csuse{@glscompstyle@almtree}%
 10081 }%

Backward compatible mcolalmtreehypergroup style.

10082 \compatglossarystyle{mcolalmtreehypergroup}{%
 10083 \csuse{@glscompstyle@almtree}%
 10084 }%

Backward compatible superragged style.

10085 \compatglossarystyle{superragged}{%
 10086 \renewcommand*{\glossaryentryfield}[5]{%
 10087 \glsentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
 10088 \tabularnewline}%
 10089 \renewcommand*{\glossarysubentryfield}[6]{%
 10090 &
 10091 \glssubentryitem{##2}%
 10092 \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
 10093 \tabularnewline}%
 10094 }%

Backward compatible superraggedborder style.

10095 \compatglossarystyle{superraggedborder}{%
 10096 \csuse{@glscompstyle@superragged}%
 10097 }%

Backward compatible superraggedheader style.

10098 \compatglossarystyle{superraggedheader}{%
 10099 \csuse{@glscompstyle@superragged}%
 10100 }%

Backward compatible superraggedheaderborder style.

```
10101 \compatglossarystyle{superraggedheaderborder}{%
10102 \csuse{@glscompstyle@superragged}%
10103 }%
```

Backward compatible superragged3col style.

```
10104 \compatglossarystyle{superragged3col}{%
10105 \renewcommand*{\glossaryentryfield}[5]{%
10106 \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%
10107 \renewcommand*{\glossarysubentryfield}[6]{%
10108 &
10109 \glssubentryitem{##2}%
10110 \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%
10111 }%
```

Backward compatible superragged3colborder style.

```
10112 \compatglossarystyle{superragged3colborder}{%
10113 \csuse{@glscompstyle@superragged3col}%
10114 }%
```

Backward compatible superragged3colheader style.

```
10115 \compatglossarystyle{superragged3colheader}{%
10116 \csuse{@glscompstyle@superragged3col}%
10117 }%
```

Backward compatible superragged3colheaderborder style.

```
10118 \compatglossarystyle{superragged3colheaderborder}{%
10119 \csuse{@glscompstyle@superragged3col}%
10120 }%
```

Backward compatible altsuperragged4col style.

```
10121 \compatglossarystyle{altsuperragged4col}{%
10122 \renewcommand*{\glossaryentryfield}[5]{%
10123 \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%
10124 \renewcommand*{\glossarysubentryfield}[6]{%
10125 &
10126 \glssubentryitem{##2}%
10127 \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%
10128 }%
```

Backward compatible altsuperragged4colheader style.

```
10129 \compatglossarystyle{altsuperragged4colheader}{%
10130 \csuse{@glscompstyle@altsuperragged4col}%
10131 }%
```

Backward compatible altsuperragged4colborder style.

```
10132 \compatglossarystyle{altsuperragged4colborder}{%
10133 \csuse{@glscompstyle@altsuperragged4col}%
10134 }%
```

Backward compatible altsuperragged4colheaderborder style.

```
10135 \compatglossarystyle{altsuperragged4colheaderborder}{%
```

```
10136 \csuse{@glscompstyle@altsuperragged4col}%
10137 }%
```

Backward compatible super style.

```
10138 \compatglossarystyle{super}{%
10139   \renewcommand*{\glossaryentryfield}[5]{%
10140     \glentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
10141   \renewcommand*{\glossarysubentryfield}[6]{%
10142     &
10143     \glssubentryitem{##2}%
10144     \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
10145 }%
```

Backward compatible superborder style.

```
10146 \compatglossarystyle{superborder}{%
10147   \csuse{@glscompstyle@super}%
10148 }%
```

Backward compatible superheader style.

```
10149 \compatglossarystyle{superheader}{%
10150   \csuse{@glscompstyle@super}%
10151 }%
```

Backward compatible superheaderborder style.

```
10152 \compatglossarystyle{superheaderborder}{%
10153   \csuse{@glscompstyle@super}%
10154 }%
```

Backward compatible super3col style.

```
10155 \compatglossarystyle{super3col}{%
10156   \renewcommand*{\glossaryentryfield}[5]{%
10157     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\\}%
10158   \renewcommand*{\glossarysubentryfield}[6]{%
10159     &
10160     \glssubentryitem{##2}%
10161     \glstarget{##2}{\strut}##4 & ##6\\}%
10162 }%
```

Backward compatible super3colborder style.

```
10163 \compatglossarystyle{super3colborder}{%
10164   \csuse{@glscompstyle@super3col}%
10165 }%
```

Backward compatible super3colheader style.

```
10166 \compatglossarystyle{super3colheader}{%
10167   \csuse{@glscompstyle@super3col}%
10168 }%
```

Backward compatible super3colheaderborder style.

```
10169 \compatglossarystyle{super3colheaderborder}{%
10170   \csuse{@glscompstyle@super3col}%
10171 }%
```

Backward compatible super4col style.

```
10172 \compatglossarystyle{super4col}{%
10173   \renewcommand*{\glossaryentryfield}[5]{%
10174     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%
10175   \renewcommand*{\glossarysubentryfield}[6]{%
10176     &
10177     \glssubentryitem{##2}%
10178     \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%
10179 }%
```

Backward compatible super4colheader style.

```
10180 \compatglossarystyle{super4colheader}{%
10181   \csuse{@glscompstyle@super4col}%
10182 }%
```

Backward compatible super4colborder style.

```
10183 \compatglossarystyle{super4colborder}{%
10184   \csuse{@glscompstyle@super4col}%
10185 }%
```

Backward compatible super4colheaderborder style.

```
10186 \compatglossarystyle{super4colheaderborder}{%
10187   \csuse{@glscompstyle@super4col}%
10188 }%
```

Backward compatible altsuper4col style.

```
10189 \compatglossarystyle{altsuper4col}{%
10190   \csuse{@glscompstyle@super4col}%
10191 }%
```

Backward compatible altsuper4colheader style.

```
10192 \compatglossarystyle{altsuper4colheader}{%
10193   \csuse{@glscompstyle@super4col}%
10194 }%
```

Backward compatible altsuper4colborder style.

```
10195 \compatglossarystyle{altsuper4colborder}{%
10196   \csuse{@glscompstyle@super4col}%
10197 }%
```

Backward compatible altsuper4colheaderborder style.

```
10198 \compatglossarystyle{altsuper4colheaderborder}{%
10199   \csuse{@glscompstyle@super4col}%
10200 }%
```

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.

```
10201 \NeedsTeXFormat{LaTeX2e}
```

Package version number now in line with main glossaries package number.

```
10202 \ProvidesPackage{glossaries-accsupp}[2016/12/16 v4.27 (NLCT)]
```

```
10203 Experimental glossaries accessibility]
```

Pass all options to glossaries:

```
10204 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
10205 \ProcessOptions
```

This package should be loaded before glossaries-extra, so complain if that has already been loaded.

```
10206 \@ifpackageloaded{glossaries-extra}
```

```
10207 {%
```

```
10208 \PackageWarning{glossaries-accsupp}{The ‘glossaries-accsupp’
```

```
10209 package has been loaded after the ‘glossaries-extra’
```

```
10210 package. This can cause a failure to integrate both
```

```
10211 packages. Either use the ‘accsupp’ option when you
```

```
10212 load ‘glossaries-extra’ or load ‘glossaries-accsupp’
```

```
10213 before loading ‘glossaries-extra’}%
```

```
10214 }
```

```
10215 }
```

tibleglossentry Override style compatibility macros:

```
10216 \def\compatibleglossentry#1#2{%
```

```
10217 \toks@{#2}%
```

```
10218 \protected@edef\@do@glossentry{%
```

```
10219 \noexpand\accsuppglossaryentryfield{#1}%
```

```
10220 {\noexpand\glsnamefont
```

```
10221 {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@name\endcsname}}%
```

```
10222 {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@desc\endcsname}%
```

```
10223 {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@symbol\endcsname}%
```

```
10224 {\the\toks@}%
```

```
10225 }%
```

```
10226 \@do@glossentry
```

```
10227 }
```

lesubglossentry

```
10228 \def\compatiblesubglossentry#1#2#3{%
10229   \toks@{#3}%
10230   \protected@edef\@do@subglossentry{%
10231     \noexpand\accsuppglossarysubentryfield{\number#1}%
10232     {#2}%
10233     {\noexpand\glsnamefont
10234      {\expandafter\expandonce\csname glo@\glsdetoklabel{#2}@name\endcsname}}}%
10235     {\expandafter\expandonce\csname glo@\glsdetoklabel{#2}@desc\endcsname}%
10236     {\expandafter\expandonce\csname glo@\glsdetoklabel{#2}@symbol\endcsname}%
10237     {\the\toks@}%
10238   }%
10239   \@do@subglossentry
10240 }
```

Required packages:

```
10241 \RequirePackage{glossaries}
10242 \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:

```
10243 \define@key{glossentry}{access}{%
10244   \def\@glo@access{#1}%
10245 }
```

textaccess The replacement text corresponding to the text key:

```
10246 \define@key{glossentry}{textaccess}{%
10247   \def\@glo@textaccess{#1}%
10248 }
```

firstaccess The replacement text corresponding to the first key:

```
10249 \define@key{glossentry}{firstaccess}{%
10250   \def\@glo@firstaccess{#1}%
10251 }
```

pluralaccess The replacement text corresponding to the plural key:

```
10252 \define@key{glossentry}{pluralaccess}{%
10253   \def\@glo@pluralaccess{#1}%
10254 }
```

firstpluralaccess The replacement text corresponding to the firstplural key:

```
10255 \define@key{glossentry}{firstpluralaccess}{%
10256   \def\@glo@firstpluralaccess{#1}%
10257 }
```

symbolaccess The replacement text corresponding to the symbol key:

```
10258 \define@key{glossentry}{symbolaccess}{%
10259   \def\@glo@symbolaccess{#1}%
10260 }
```

symbolpluralaccess The replacement text corresponding to the symbolplural key:

```
10261 \define@key{glossentry}{symbolpluralaccess}{%
10262   \def\@glo@symbolpluralaccess{#1}%
10263 }
```

descriptionaccess The replacement text corresponding to the description key:

```
10264 \define@key{glossentry}{descriptionaccess}{%
10265   \def\@glo@descaccess{#1}%
10266 }
```

descriptionpluralaccess The replacement text corresponding to the descriptionplural key:

```
10267 \define@key{glossentry}{descriptionpluralaccess}{%
10268   \def\@glo@descpluralaccess{#1}%
10269 }
```

shortaccess The replacement text corresponding to the short key:

```
10270 \define@key{glossentry}{shortaccess}{%
10271   \def\@glo@shortaccess{#1}%
10272 }
```

shortpluralaccess The replacement text corresponding to the shortplural key:

```
10273 \define@key{glossentry}{shortpluralaccess}{%
10274   \def\@glo@shortpluralaccess{#1}%
10275 }
```

longaccess The replacement text corresponding to the long key:

```
10276 \define@key{glossentry}{longaccess}{%
10277   \def\@glo@longaccess{#1}%
10278 }
```

longpluralaccess The replacement text corresponding to the longplural key:

```
10279 \define@key{glossentry}{longpluralaccess}{%
10280   \def\@glo@longpluralaccess{#1}%
10281 }
```

There are no equivalent keys for the user1...user6 keys. The replacement text would have to be explicitly put in the value, e.g., user1={\glsacccsupp{inches}{in}}.

Append these new keys to \@gls@keymap:

```

10282 \appto\@gls@keymap{,%
10283   {access}{access},%
10284   {textaccess}{textaccess},%
10285   {firstaccess}{firstaccess},%
10286   {pluralaccess}{pluralaccess},%
10287   {firstpluralaccess}{firstpluralaccess},%
10288   {symbolaccess}{symbolaccess},%
10289   {symbolpluralaccess}{symbolpluralaccess},%
10290   {descaccess}{descaccess},%
10291   {descpluralaccess}{descpluralaccess},%
10292   {shortaccess}{shortaccess},%
10293   {shortpluralaccess}{shortpluralaccess},%
10294   {longaccess}{longaccess},%
10295   {longpluralaccess}{longpluralaccess}%
10296 }
```

\@gls@noaccess Indicates that no replacement text has been provided.

```

10297 \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):

```

10298 \let\@gls@oldnewglossaryentryprehook\@newglossaryentryprehook
10299 \renewcommand*{\@newglossaryentryprehook}{%
10300   \@gls@oldnewglossaryentryprehook
10301   \def\@glo@access{\@glo@symbol}%
```

Initialise the other keys:

```

10302 \def\@glo@textaccess{\@glo@access}%
10303 \def\@glo@firstaccess{\@glo@access}%
10304 \def\@glo@pluralaccess{\@glo@textaccess}%
10305 \def\@glo@firstpluralaccess{\@glo@pluralaccess}%
10306 \def\@glo@symbolaccess{\relax}%
10307 \def\@glo@symbolpluralaccess{\@glo@symbolaccess}%
10308 \def\@glo@descaccess{\relax}%
10309 \def\@glo@descpluralaccess{\@glo@descaccess}%
10310 \def\@glo@shortaccess{\relax}%
10311 \def\@glo@shortpluralaccess{\@glo@shortaccess}%
10312 \def\@glo@longaccess{\relax}%
10313 \def\@glo@longpluralaccess{\@glo@longaccess}%
10314 }
```

Add to the end hook:

```

10315 \let\@gls@oldnewglossaryentryposthook\@newglossaryentryposthook
10316 \renewcommand*{\@newglossaryentryposthook}{%
10317   \@gls@oldnewglossaryentryposthook
```

Store the access information:

```

10318 \expandafter
10319 \protected@xdef\csname glo@\@glo@label @access\endcsname{%
```



```

10320     \@glo@access}%
10321 \expandafter
10322     \protected@xdef\csname glo@\@glo@label @textaccess\endcsname{%
10323     \@glo@textaccess}%
10324 \expandafter
10325     \protected@xdef\csname glo@\@glo@label @firstaccess\endcsname{%
10326     \@glo@firstaccess}%
10327 \expandafter
10328     \protected@xdef\csname glo@\@glo@label @pluralaccess\endcsname{%
10329     \@glo@pluralaccess}%
10330 \expandafter
10331     \protected@xdef\csname glo@\@glo@label @firstpluralaccess\endcsname{%
10332     \@glo@firstpluralaccess}%
10333 \expandafter
10334     \protected@xdef\csname glo@\@glo@label @symbolaccess\endcsname{%
10335     \@glo@symbolaccess}%
10336 \expandafter
10337     \protected@xdef\csname glo@\@glo@label @symbolpluralaccess\endcsname{%
10338     \@glo@symbolpluralaccess}%
10339 \expandafter
10340     \protected@xdef\csname glo@\@glo@label @descaccess\endcsname{%
10341     \@glo@descaccess}%
10342 \expandafter
10343     \protected@xdef\csname glo@\@glo@label @descpluralaccess\endcsname{%
10344     \@glo@descpluralaccess}%
10345 \expandafter
10346     \protected@xdef\csname glo@\@glo@label @shortaccess\endcsname{%
10347     \@glo@shortaccess}%
10348 \expandafter
10349     \protected@xdef\csname glo@\@glo@label @shortpluralaccess\endcsname{%
10350     \@glo@shortpluralaccess}%
10351 \expandafter
10352     \protected@xdef\csname glo@\@glo@label @longaccess\endcsname{%
10353     \@glo@longaccess}%
10354 \expandafter
10355     \protected@xdef\csname glo@\@glo@label @longpluralaccess\endcsname{%
10356     \@glo@longpluralaccess}%
10357 }

```

5.2 Accessing Replacement Text

`\glsentryaccess` Get the value of the access key for the entry with the given label:

```

10358 \newcommand*{\glsentryaccess}[1]{%
10359   \@gls@entry@field{#1}{access}%
10360 }

```

`\glsentrytextaccess` Get the value of the textaccess key for the entry with the given label:

```

10361 \newcommand*{\glsentrytextaccess}[1]{%

```

```

10362 \@gls@entry@field{#1}{textaccess}%
10363 }

```

entryfirstaccess Get the value of the firstaccess key for the entry with the given label:

```

10364 \newcommand*{\glsentryfirstaccess}[1]{%
10365 \@gls@entry@field{#1}{firstaccess}%
10366 }

```

entrypluralaccess Get the value of the pluralaccess key for the entry with the given label:

```

10367 \newcommand*{\glsentrypluralaccess}[1]{%
10368 \@gls@entry@field{#1}{pluralaccess}%
10369 }

```

entryfirstpluralaccess Get the value of the firstpluralaccess key for the entry with the given label:

```

10370 \newcommand*{\glsentryfirstpluralaccess}[1]{%
10371 \csname glo#1@firstpluralaccess\endcsname
10372 }

```

entrysymbolaccess Get the value of the symbolaccess key for the entry with the given label:

```

10373 \newcommand*{\glsentrysymbolaccess}[1]{%
10374 \@gls@entry@field{#1}{symbolaccess}%
10375 }

```

entrysymbolpluralaccess Get the value of the symbolpluralaccess key for the entry with the given label:

```

10376 \newcommand*{\glsentrysymbolpluralaccess}[1]{%
10377 \@gls@entry@field{#1}{symbolpluralaccess}%
10378 }

```

entrydescaccess Get the value of the descriptionaccess key for the entry with the given label:

```

10379 \newcommand*{\glsentrydescaccess}[1]{%
10380 \@gls@entry@field{#1}{descaccess}%
10381 }

```

entrydescpluralaccess Get the value of the descriptionpluralaccess key for the entry with the given label:

```

10382 \newcommand*{\glsentrydescpluralaccess}[1]{%
10383 \@gls@entry@field{#1}{descaccess}%
10384 }

```

entryshortaccess Get the value of the shortaccess key for the entry with the given label:

```

10385 \newcommand*{\glsentryshortaccess}[1]{%
10386 \@gls@entry@field{#1}{shortaccess}%
10387 }

```

entryshortpluralaccess Get the value of the shortpluralaccess key for the entry with the given label:

```

10388 \newcommand*{\glsentryshortpluralaccess}[1]{%
10389 \@gls@entry@field{#1}{shortpluralaccess}%
10390 }

```

entrylongaccess Get the value of the longaccess key for the entry with the given label:

```
10391 \newcommand*{\glsentrylongaccess}[1]{%
10392   \@gls@entry@field{#1}{longaccess}%
10393 }
```

ongpluralaccess Get the value of the longpluralaccess key for the entry with the given label:

```
10394 \newcommand*{\glsentrylongpluralaccess}[1]{%
10395   \@gls@entry@field{#1}{longpluralaccess}%
10396 }
```

\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.)

```
10397 \newcommand*{\glsaccsupp}[2]{%
10398   \BeginAccSupp{ActualText=#1}#2\EndAccSupp{}%
10399 }
```

\xglsaccsupp Fully expands replacement text before calling \glsaccsupp

```
10400 \newcommand*{\xglsaccsupp}[2]{%
10401   \protected@edef\@gls@replacementtext{#1}%
10402   \expandafter\glsaccsupp\expandafter{\@gls@replacementtext}{#2}%
10403 }
```

@access@display

```
10404 \newcommand*{\@gls@access@display}[2]{%
10405   \protected@edef\@glo@access{#2}%
10406   \ifx\@glo@access\@gls@noaccess
10407     #1%
10408   \else
10409     \xglsaccsupp{\@glo@access}{#1}%
10410   \fi
10411 }
```

meaccessdisplay Displays the first argument with the accessibility text for the entry with the label given by the second argument (if set).

```
10412 \DeclareRobustCommand*{\glsnameaccessdisplay}[2]{%
10413   \@gls@access@display{#1}{\glsentryaccess{#2}}%
10414 }
```

xtaccessdisplay As above but for the textaccess replacement text.

```
10415 \DeclareRobustCommand*{\glsstextaccessdisplay}[2]{%
10416   \@gls@access@display{#1}{\glsentrytextaccess{#2}}%
10417 }
```

alaccessdisplay As above but for the pluralaccess replacement text.

```
10418 \DeclareRobustCommand*{\glspluralaccessdisplay}[2]{%
10419   \@gls@access@display{#1}{\glsentrypluralaccess{#2}}%
10420 }
```

staccessdisplay As above but for the firstaccess replacement text.

```
10421 \DeclareRobustCommand*\glfirstaccessdisplay}[2]{%
10422   \@gls@access@display{#1}{\glentryfirstaccess{#2}}%
10423 }
```

alaccessdisplay As above but for the firstpluralaccess replacement text.

```
10424 \DeclareRobustCommand*\glfirstpluralaccessdisplay}[2]{%
10425   \@gls@access@display{#1}{\glentryfirstpluralaccess{#2}}%
10426 }
```

olaccessdisplay As above but for the symbolaccess replacement text.

```
10427 \DeclareRobustCommand*\glssymbolaccessdisplay}[2]{%
10428   \@gls@access@display{#1}{\glentrysymbolaccess{#2}}%
10429 }
```

alaccessdisplay As above but for the symbolpluralaccess replacement text.

```
10430 \DeclareRobustCommand*\glssymbolpluralaccessdisplay}[2]{%
10431   \@gls@access@display{#1}{\glentrysymbolpluralaccess{#2}}%
10432 }
```

onaccessdisplay As above but for the descriptionaccess replacement text.

```
10433 \DeclareRobustCommand*\glsdescriptionaccessdisplay}[2]{%
10434   \@gls@access@display{#1}{\glentrydescaccess{#2}}%
10435 }
```

alaccessdisplay As above but for the descriptionpluralaccess replacement text.

```
10436 \DeclareRobustCommand*\glsdescriptionpluralaccessdisplay}[2]{%
10437   \@gls@access@display{#1}{\glentrydescpluralaccess{#2}}%
10438 }
```

rtaccessdisplay As above but for the shortaccess replacement text.

```
10439 \DeclareRobustCommand*\glsshortaccessdisplay}[2]{%
10440   \@gls@access@display{#1}{\glentryshortaccess{#2}}%
10441 }
```

alaccessdisplay As above but for the shortpluralaccess replacement text.

```
10442 \DeclareRobustCommand*\glsshortpluralaccessdisplay}[2]{%
10443   \@gls@access@display{#1}{\glentryshortpluralaccess{#2}}%
10444 }
```

ngaccessdisplay As above but for the longaccess replacement text.

```
10445 \DeclareRobustCommand*\glslongaccessdisplay}[2]{%
10446   \@gls@access@display{#1}{\glentrylongaccess{#2}}%
10447 }
```

alaccessdisplay As above but for the longpluralaccess replacement text.

```
10448 \DeclareRobustCommand*\glslongpluralaccessdisplay}[2]{%
10449   \@gls@access@display{#1}{\glentrylongpluralaccess{#2}}%
10450 }
```

`\glsaccessdisplay` Gets the replacement text corresponding to the named key given by the first argument and calls the appropriate command defined above.

```

10451 \DeclareRobustCommand*\glsaccessdisplay}[3]{%
10452   \@ifundefined{gls#1accessdisplay}%
10453   {%
10454     \PackageError{glossaries-accsupp}{No accessibility support
10455       for key ‘#1’}{}%
10456   }%
10457   {%
10458     \csname gls#1accessdisplay\endcsname{#2}{#3}%
10459   }%
10460 }

```

`\default@entryfmt` Redefine the default entry format to use accessibility information

```

10461 \renewcommand*\@@gls@default@entryfmt}[2]{%
10462   \ifdefempty\glscustomtext
10463   {%
10464     \glsifplural
10465     {%

```

Plural form

```

10466       \glscapscase
10467       {%

```

Don't adjust case

```

10468       \ifglsused\glslabel
10469       {%

```

Subsequent use

```

10470         #2{\glspluralaccessdisplay
10471           {\glsentryplural{\glslabel}}{\glslabel}}%
10472         {\glsdescriptionpluralaccessdisplay
10473           {\glsentrydescplural{\glslabel}}{\glslabel}}%
10474         {\glsymbolpluralaccessdisplay
10475           {\glsentrysymbolplural{\glslabel}}{\glslabel}}
10476         {\glsinsert}%
10477       }%
10478     {%

```

First use

```

10479         #1{\glsfirstpluralaccessdisplay
10480           {\glsentryfirstplural{\glslabel}}{\glslabel}}%
10481         {\glsdescriptionpluralaccessdisplay
10482           {\glsentrydescplural{\glslabel}}{\glslabel}}%
10483         {\glsymbolpluralaccessdisplay
10484           {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10485         {\glsinsert}%
10486       }%
10487     }%
10488   }%

```

Make first letter upper case

```
10489      \ifglsused\glslabel
10490      {%
```

Subsequent use.

```
10491      #2{\glspluralaccessdisplay
10492          {\Glsentryplural{\glslabel}}{\glslabel}}%
10493          {\glsdescriptionpluralaccessdisplay
10494          {\glsentrydescplural{\glslabel}}{\glslabel}}%
10495          {\glssymbolpluralaccessdisplay
10496          {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10497          {\glsinsert}}%
10498      }%
10499      {%
```

First use

```
10500      #1{\glsfirstpluralaccessdisplay
10501          {\Glsentryfirstplural{\glslabel}}{\glslabel}}%
10502          {\glsdescriptionpluralaccessdisplay
10503          {\glsentrydescplural{\glslabel}}{\glslabel}}%
10504          {\glssymbolpluralaccessdisplay
10505          {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10506          {\glsinsert}}%
10507      }%
10508      }%
10509      {%
```

Make all upper case

```
10510      \ifglsused\glslabel
10511      {%
```

Subsequent use

```
10512      \MakeUppercase{%
10513      #2{\glspluralaccessdisplay
10514          {\glsentryplural{\glslabel}}{\glslabel}}%
10515          {\glsdescriptionpluralaccessdisplay
10516          {\glsentrydescplural{\glslabel}}{\glslabel}}%
10517          {\glssymbolpluralaccessdisplay
10518          {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10519          {\glsinsert}}}%
10520      }%
10521      {%
```

First use

```
10522      \MakeUppercase{%
10523      #1{\glsfirstpluralaccessdisplay
10524          {\glsentryfirstplural{\glslabel}}{\glslabel}}%
10525          {\glsdescriptionpluralaccessdisplay
10526          {\glsentrydescplural{\glslabel}}{\glslabel}}%
10527          {\glssymbolpluralaccessdisplay
10528          {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
```

```

10529         {\glsinsert}}}%
10530     }%
10531 }%
10532 }%
10533 {%

```

Singular form

```

10534     \glscapscase
10535     {%

```

Don't adjust case

```

10536     \ifglsused\glslabel
10537     {%

```

Subsequent use

```

10538     #2{\glstextaccessdisplay
10539         {\glsentrytext{\glslabel}}{\glslabel}}%
10540     {\glsdescriptionaccessdisplay
10541         {\glsentrydesc{\glslabel}}{\glslabel}}%
10542     {\glssymbolaccessdisplay
10543         {\glsentrysymbol{\glslabel}}{\glslabel}}%
10544     {\glsinsert}}%
10545 }%
10546 {%

```

First use

```

10547     #1{\glsfirstaccessdisplay
10548         {\glsentryfirst{\glslabel}}{\glslabel}}%
10549     {\glsdescriptionaccessdisplay
10550         {\glsentrydesc{\glslabel}}{\glslabel}}%
10551     {\glssymbolaccessdisplay
10552         {\glsentrysymbol{\glslabel}}{\glslabel}}%
10553     {\glsinsert}}%
10554 }%
10555 }%
10556 {%

```

Make first letter upper case

```

10557     \ifglsused\glslabel
10558     {%

```

Subsequent use

```

10559     #2{\glstextaccessdisplay
10560         {\Glsentrytext{\glslabel}}{\glslabel}}%
10561     {\glsdescriptionaccessdisplay
10562         {\glsentrydesc{\glslabel}}{\glslabel}}%
10563     {\glssymbolaccessdisplay
10564         {\glsentrysymbol{\glslabel}}{\glslabel}}%
10565     {\glsinsert}}%
10566 }%
10567 {%

```

First use

```

10568      #1{\glsfirstaccessdisplay
10569          {\Glsentryfirst{\glslabel}}{\glslabel}}%
10570          {\glsdescriptionaccessdisplay
10571          {\glsentrydesc{\glslabel}}{\glslabel}}%
10572          {\glssymbolaccessdisplay
10573          {\glsentrysymbol{\glslabel}}{\glslabel}}%
10574          {\glsinsert}}%
10575      }%
10576  }%
10577  {%

```

Make all upper case

```

10578      \ifglsused\glslabel
10579      {%

```

Subsequent use

```

10580      \MakeUppercase{%
10581      #2{\glstextaccessdisplay
10582          {\glsentrytext{\glslabel}}{\glslabel}}%
10583          {\glsdescriptionaccessdisplay
10584          {\glsentrydesc{\glslabel}}{\glslabel}}%
10585          {\glssymbolaccessdisplay
10586          {\glsentrysymbol{\glslabel}}{\glslabel}}%
10587          {\glsinsert}}%
10588      }%
10589      {%

```

First use

```

10590      \MakeUppercase{%
10591      #1{\glsfirstaccessdisplay
10592          {\glsentryfirst{\glslabel}}{\glslabel}}%
10593          {\glsdescriptionaccessdisplay
10594          {\glsentrydesc{\glslabel}}{\glslabel}}%
10595          {\glssymbolaccessdisplay
10596          {\glsentrysymbol{\glslabel}}{\glslabel}}%
10597          {\glsinsert}}%
10598      }%
10599  }%
10600  }%
10601  }%
10602  {%

```

Custom text provided in \glsdisp

```

10603      \ifglsused{\glslabel}%
10604      {%

```

Subsequent use

```

10605      #2{\glscustomtext}%
10606      {\glsdescriptionaccessdisplay
10607      {\glsentrydesc{\glslabel}}{\glslabel}}%

```



```

10608      {\glssymbolaccessdisplay
10609      {\glentrysymbol{\glslabel}}{\glslabel}}%
10610      {\glsinsert}}%
10611  }%
10612  {%

```

First use

```

10613      #1{\glscustomtext}%
10614      {\glsdescriptionaccessdisplay
10615      {\glentrydesc{\glslabel}}{\glslabel}}%
10616      {\glssymbolaccessdisplay
10617      {\glentrysymbol{\glslabel}}{\glslabel}}%
10618      {\glsinsert}}%
10619  }%
10620  }%
10621 }

```

`\glsgenentryfmt` Redefine to use accessibility information.

```

10622 \renewcommand*{\glsgenentryfmt}{%
10623   \ifdefempty\glscustomtext
10624   {%
10625     \glsifplural
10626     {%

```

Plural form

```

10627     \glscapscase
10628     {%

```

Don't adjust case

```

10629     \ifglused\glslabel
10630     {%

```

Subsequent use

```

10631     \glspluralaccessdisplay
10632     {\glentryplural{\glslabel}}{\glslabel}}%
10633     \glsinsert
10634   }%
10635   {%

```

First use

```

10636     \glsfirstpluralaccessdisplay
10637     {\glentryfirstplural{\glslabel}}{\glslabel}}%
10638     \glsinsert
10639   }%
10640   }%
10641   {%

```

Make first letter upper case

```

10642     \ifglused\glslabel
10643     {%

```

Subsequent use.

```
10644      \glspluralaccessdisplay
10645      {\Glsentryplural{\glslabel}}{\glslabel}%
10646      \glsinsert
10647      }%
10648      {%
```

First use

```
10649      \glsfirstpluralaccessdisplay
10650      {\Glsentryfirstplural{\glslabel}}{\glslabel}%
10651      \glsinsert
10652      }%
10653      }%
10654      {%
```

Make all upper case

```
10655      \ifglused\glslabel
10656      {%
```

Subsequent use

```
10657      \glspluralaccessdisplay
10658      {\mfirstucMakeUppercase{\glsentryplural{\glslabel}}}%
10659      {\glslabel}%
10660      \mfirstucMakeUppercase{\glsinsert}%
10661      }%
10662      {%
```

First use

```
10663      \glsfirstpluralaccessdisplay
10664      {\mfirstucMakeUppercase{\glsentryfirstplural{\glslabel}}}%
10665      {\glslabel}%
10666      \mfirstucMakeUppercase{\glsinsert}%
10667      }%
10668      }%
10669      }%
10670      {%
```

Singular form

```
10671      \glscapscase
10672      {%
```

Don't adjust case

```
10673      \ifglused\glslabel
10674      {%
```

Subsequent use

```
10675      \glstextaccessdisplay{\glsentrytext{\glslabel}}{\glslabel}%
10676      \glsinsert
10677      }%
10678      {%
```

First use

```
10679      \glsfirstaccessdisplay{\glsentryfirst{\glslabel}}{\glslabel}%
10680      \glsinsert
10681      }%
10682      }%
10683      {%
```

Make first letter upper case

```
10684      \ifglsused\glslabel
10685      {%
```

Subsequent use

```
10686      \glstextaccessdisplay{\Glsentrytext{\glslabel}}{\glslabel}%
10687      \glsinsert
10688      }%
10689      {%
```

First use

```
10690      \glsfirstaccessdisplay{\Glsentryfirst{\glslabel}}{\glslabel}%
10691      \glsinsert
10692      }%
10693      }%
10694      {%
```

Make all upper case

```
10695      \ifglsused\glslabel
10696      {%
```

Subsequent use

```
10697      \glstextaccessdisplay
10698      {\mfirstucMakeUppercase{\glsentrytext{\glslabel}}}{\glslabel}%
10699      \mfirstucMakeUppercase{\glsinsert}%
10700      }%
10701      {%
```

First use

```
10702      \glsfirstaccessdisplay
10703      {\mfirstucMakeUppercase{\glsentryfirst{\glslabel}}}{\glslabel}%
10704      \mfirstucMakeUppercase{\glsinsert}%
10705      }%
10706      }%
10707      }%
10708      }%
10709      {%
```

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.

```
10710      \glscustomtext\glsinsert
10711      }%
10712 }
```

`\glsgenacfmt` Redefine to include accessibility information.

```
10713 \renewcommand*{\glsgenacfmt}{%
10714   \ifdefempty\glscustomtext
10715     {%
10716       \ifglused\glslabel
10717       {%
```

 Subsequent use:

```
10718       \glsifplural
10719       {%
```

 Subsequent plural form:

```
10720       \glscapscase
10721       {%
```

 Subsequent plural form, don't adjust case:

```
10722       \acronymfont
10723       {\glsshortpluralaccessdisplay
10724        {\glentryshortpl{\glslabel}}{\glslabel}}%
10725       \glsinsert
10726     }%
10727     {%
```

 Subsequent plural form, make first letter upper case:

```
10728       \acronymfont
10729       {\glsshortpluralaccessdisplay
10730        {\Glsentryshortpl{\glslabel}}{\glslabel}}%
10731       \glsinsert
10732     }%
10733     {%
```

 Subsequent plural form, all caps:

```
10734       \mfirstucMakeUppercase
10735       {\acronymfont
10736        {\glsshortpluralaccessdisplay
10737         {\glentryshortpl{\glslabel}}{\glslabel}}%
10738        \glsinsert}%
10739     }%
10740     }%
10741     {%
```

 Subsequent singular form

```
10742       \glscapscase
10743       {%
```

 Subsequent singular form, don't adjust case:

```
10744       \acronymfont
10745       {\glsshortaccessdisplay{\glentryshort{\glslabel}}{\glslabel}}%
10746       \glsinsert
10747     }%
10748     {%
```

Subsequent singular form, make first letter upper case:

```
10749      \acronymfont
10750      {\glsshortaccessdisplay{\Glsentryshort{\glslabel}}{\glslabel}}%
10751      \glsinsert
10752      }%
10753      {%
```

Subsequent singular form, all caps:

```
10754      \mfirstucMakeUppercase
10755      {\acronymfont{%
10756      \glsshortaccessdisplay{\Glsentryshort{\glslabel}}{\glslabel}}%
10757      \glsinsert}%
10758      }%
10759      }%
10760      }%
10761      {%
```

First use:

```
10762      \glsifplural
10763      {%
```

First use plural form:

```
10764      \glscapscase
10765      {%
```

First use plural form, don't adjust case:

```
10766      \genplacrfullformat{\glslabel}{\glsinsert}%
10767      }%
10768      {%
```

First use plural form, make first letter upper case:

```
10769      \Genplacrfullformat{\glslabel}{\glsinsert}%
10770      }%
10771      {%
```

First use plural form, all caps:

```
10772      \mfirstucMakeUppercase
10773      {\genplacrfullformat{\glslabel}{\glsinsert}}%
10774      }%
10775      }%
10776      {%
```

First use singular form

```
10777      \glscapscase
10778      {%
```

First use singular form, don't adjust case:

```
10779      \genacrfullformat{\glslabel}{\glsinsert}%
10780      }%
10781      {%
```

First use singular form, make first letter upper case:

```
10782      \Genacrfullformat{\glslabel}{\glsinsert}%
10783      }%
10784      {%
```

First use singular form, all caps:

```
10785      \mfirstucMakeUppercase
10786      {\genacrfullformat{\glslabel}{\glsinsert}}%
10787      }%
10788      }%
10789      }%
10790      }%
10791      {%
```

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.

```
10792      \glscustomtext
10793      }%
10794 }
```

enacrfullformat Redefine to include accessibility information.

```
10795 \renewcommand*{\genacrfullformat}[2]{%
10796   \glslongaccessdisplay{\glsentrylong{#1}}{#1}#2\space
10797   (\glsshortaccessdisplay{\protect\firstacronymfont{\glsentryshort{#1}}}{#1})%
10798 }
```

enacrfullformat Redefine to include accessibility information.

```
10799 \renewcommand*{\Genacrfullformat}[2]{%
10800   \glslongaccessdisplay{\Glsentrylong{#1}}{#1}#2\space
10801   (\glsshortaccessdisplay{\protect\firstacronymfont{\Glsentryshort{#1}}}{#1})%
10802 }
```

placrfullformat Redefine to include accessibility information.

```
10803 \renewcommand*{\genplacrfullformat}[2]{%
10804   \glslongpluralaccessdisplay{\glsentrylongpl{#1}}{#1}#2\space
10805   (\glsshortpluralaccessdisplay
10806     {\protect\firstacronymfont{\glsentryshortpl{#1}}}{#1})%
10807 }
```

placrfullformat Redefine to include accessibility information.

```
10808 \renewcommand*{\Genplacrfullformat}[2]{%
10809   \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}#2\space
10810   (\glsshortpluralaccessdisplay
10811     {\protect\firstacronymfont{\Glsentryshortpl{#1}}}{#1})%
10812 }
```

\@acrshort

```
10813 \def\@acrshort#1#2[#3]{%
10814   \glsdoifexists{#2}%
```

```

10815 {%
10816   \let\do@gls@link@checkfirsthyper\relax

10817   \let\glsifplural\@secondoftwo
10818   \let\glsupcase\@firstofthree
10819   \let\glsinsert\@empty
10820   \def\glscustomtext{%
10821     \acronymfont{\glsshortaccessdisplay{\glentryshort{#2}}{#2}}#3%
10822   }%

   Call \@gls@link
10823   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10824   }%

10825   \glspostlinkhook
10826 }

```

\@Acrshort

```

10827 \def\@Acrshort#1#2[#3]{%
10828   \glsdoifexists{#2}%
10829   {%
10830     \let\do@gls@link@checkfirsthyper\relax

10831     \let\glsifplural\@secondoftwo
10832     \let\glsupcase\@secondofthree
10833     \let\glsinsert\@empty
10834     \def\glscustomtext{%
10835       \acronymfont{\glsshortaccessdisplay{\Glsentryshort{#2}}{#2}}#3%
10836     }%

     Call \@gls@link
10837     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10838     }%

10839     \glspostlinkhook
10840 }

```

\@ACRshort

```

10841 \def\@ACRshort#1#2[#3]{%
10842   \glsdoifexists{#2}%
10843   {%
10844     \let\do@gls@link@checkfirsthyper\relax

10845     \let\glsifplural\@secondoftwo
10846     \let\glsupcase\@thirdofthree
10847     \let\glsinsert\@empty
10848     \def\glscustomtext{%
10849       \acronymfont{\glsshortaccessdisplay
10850         {\MakeUppercase{\glentryshort{#2}}}{#2}}#3%
10851     }%

```

```

      Call \@gls@link
10852   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10853   }%

10854   \glspostlinkhook
10855 }

```

\@acrlong

```

10856 \def\@acrlong#1#2[#3]{%
10857   \glsdoifexists{#2}%
10858   {%
10859     \let\do@gls@link@checkfirsthyper\relax

10860     \let\glsifplural\@secondoftwo
10861     \let\glscapscase\@firstofthree
10862     \let\glsinsert\@empty
10863     \def\glscustomtext{%
10864       \acronymfont{\glslongaccessdisplay{\glsentrylong{#2}}{#2}}#3%
10865     }%

```

```

      Call \@gls@link
10866   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10867   }%

10868   \glspostlinkhook
10869 }

```

\@Acrlong

```

10870 \def\@Acrlong#1#2[#3]{%
10871   \glsdoifexists{#2}%
10872   {%
10873     \let\do@gls@link@checkfirsthyper\relax

10874     \let\glsifplural\@secondoftwo
10875     \let\glscapscase\@firstofthree
10876     \let\glsinsert\@empty
10877     \def\glscustomtext{%
10878       \acronymfont{\glslongaccessdisplay{\Glsentrylong{#2}}{#2}}#3%
10879     }%

```

```

      Call \@gls@link
10880   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10881   }%

10882   \glspostlinkhook
10883 }

```

\@ACRlong

```

10884 \def\@ACRlong#1#2[#3]{%
10885   \glsdoifexists{#2}%
10886   {%
10887     \let\do@gls@link@checkfirsthyper\relax

```



```

10888 \let\glsifplural\@secondoftwo
10889 \let\glsifcaps\@firstofthree
10890 \let\glsinsert\@empty
10891 \def\glscustomtext{%
10892     \acronymfont{\glslongaccessdisplay{%
10893         \MakeUppercase{\glsentrylong{#2}}}{#2}#3}%
10894     }%

    Call \@gls@link
10895     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10896 }%

10897 \glspostlinkhook
10898 }

```

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.

```

10899 \renewcommand*{\glossentryname}[1]{%
10900     \glsdoifexists{#1}%
10901     {%
10902         \glsnamefont{\glsnameaccessdisplay{\glsentryname{#1}}{#1}}%
10903     }%
10904 }

10905 \renewcommand*{\glossentrydesc}[1]{%
10906     \glsdoifexists{#1}%
10907     {%
10908         \glsnamefont{\glsnameaccessdisplay{\Glsentryname{#1}}{#1}}%
10909     }%
10910 }

10911 \renewcommand*{\glossentrydesc}[1]{%
10912     \glsdoifexists{#1}%
10913     {%
10914         \glsdescriptionaccessdisplay{\glsentrydesc{#1}}{#1}%
10915     }%
10916 }

10917 \renewcommand*{\Glossentrydesc}[1]{%
10918     \glsdoifexists{#1}%
10919     {%
10920         \glsdescriptionaccessdisplay{\Glsentrydesc{#1}}{#1}%
10921     }%
10922 }

```

```

10923 \renewcommand*{\glossentrysymbol}[1]{%
10924   \glsdoifexists{#1}%
10925   {%
10926     \glssymbolaccessdisplay{\glsentrysymbol{#1}}{#1}%
10927   }%
10928 }

10929 \renewcommand*{\Glossentrysymbol}[1]{%
10930   \glsdoifexists{#1}%
10931   {%
10932     \glssymbolaccessdisplay{\Glsentrysymbol{#1}}{#1}%
10933   }%
10934 }

```

ssaryentryfield

```

10935 \newcommand*{\accsuppglossaryentryfield}[5]{%
10936   \glossaryentryfield{#1}%
10937   {\glsnameaccessdisplay{#2}{#1}}%
10938   {\glsdescriptionaccessdisplay{#3}{#1}}%
10939   {\glssymbolaccessdisplay{#4}{#1}}{#5}%
10940 }

```

rysubentryfield

```

10941 \newcommand*{\accsuppglossarysubentryfield}[6]{%
10942   \glossarysubentryfield{#1}{#2}%
10943   {\glsnameaccessdisplay{#3}{#2}}%
10944   {\glsdescriptionaccessdisplay{#4}{#2}}%
10945   {\glssymbolaccessdisplay{#5}{#2}}{#6}%
10946 }

```

5.4 Acronyms

Redefine acronym styles provided by glossaries:

long-short *<long>* (*<short>*) acronym style.

```

10947 \renewacronymstyle{long-short}%
10948 {%

```

Check for long form in case this is a mixed glossary.

```

10949   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
10950 }%
10951 {%
10952   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
10953   \renewcommand*{\genacrfullformat}[2]{%
10954     \glslongaccessdisplay{\glsentrylong{##1}}{##1}##2\space
10955     (\glsshortaccessdisplay
10956       {\protect\firstacronymfont{\glsentryshort{##1}}}{##1})%
10957   }%
10958   \renewcommand*{\Genacrfullformat}[2]{%

```

```

10959 \glslongaccessdisplay{\Glsentrylong{##1}}{##1}##2\space
10960 (\glsshortaccessdisplay
10961   {\protect\firstacronymfont{\glsentryshort{##1}}{##1}})%
10962 }%
10963 \renewcommand*{\genplacrfullformat}[2]{%
10964   \glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}##2\space
10965   (\glsshortpluralaccessdisplay
10966     {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}})%
10967 }%
10968 \renewcommand*{\Genplacrfullformat}[2]{%
10969   \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}##2\space
10970   (\glsshortpluralaccessdisplay
10971     {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}})%
10972 }%
10973 \renewcommand*{\acronymentry}[1]{%
10974   \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}{##1}}
10975 \renewcommand*{\acronymsort}[2]{##1}%
10976 \renewcommand*{\acronymfont}[1]{##1}%
10977 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
10978 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
10979 }

```

short-long (*short*) (*long*) acronym style.

```

10980 \renewacronymstyle{short-long}%
10981 {%

```

Check for long form in case this is a mixed glossary.

```

10982 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
10983 }%
10984 {%
10985 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
10986 \renewcommand*{\genacrfullformat}[2]{%
10987   \glsshortaccessdisplay
10988     {\protect\firstacronymfont{\glsentryshort{##1}}{##1}##2\space
10989     (\glslongaccessdisplay{\glsentrylong{##1}}{##1}})%
10990 }%
10991 \renewcommand*{\Genacrfullformat}[2]{%
10992   \glsshortaccessdisplay
10993     {\protect\firstacronymfont{\Glsentryshort{##1}}{##1}##2\space
10994     (\glslongaccessdisplay{\glsentrylong{##1}}{##1}})%
10995 }%
10996 \renewcommand*{\genplacrfullformat}[2]{%
10997   \glsshortpluralaccessdisplay
10998     {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}##2\space
10999     (\glslongpluralaccessdisplay
11000       {\glsentrylongpl{##1}}{##1}})%
11001 }%
11002 \renewcommand*{\Genplacrfullformat}[2]{%
11003   \glsshortpluralaccessdisplay
11004     {\protect\firstacronymfont{\Glsentryshortpl{##1}}{##1}##2\space

```

```

11005 (\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1})%
11006 }%
11007 \renewcommand*{\acronymentry}[1]{%
11008   \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}%
11009 \renewcommand*{\acronymsort}[2]{##1}%
11010 \renewcommand*{\acronymfont}[1]{##1}%
11011 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
11012 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11013 }

```

long-short-desc *<long>* (*<short>*) acronym style that has an accompanying description (which the user needs to supply).

```

11014 \renewacronymstyle{long-short-desc}%
11015 {%
11016   \GlsUseAcrEntryDispStyle{long-short}%
11017 }%
11018 {%
11019   \GlsUseAcrStyleDefs{long-short}%
11020   \renewcommand*{\GenericAcronymFields}{}%
11021   \renewcommand*{\acronymsort}[2]{##2}%
11022   \renewcommand*{\acronymentry}[1]{%
11023     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11024     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11025 }

```

g-sc-short-desc *<long>* (\textsc{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

11026 \renewacronymstyle{long-sc-short-desc}%
11027 {%
11028   \GlsUseAcrEntryDispStyle{long-sc-short}%
11029 }%
11030 {%
11031   \GlsUseAcrStyleDefs{long-sc-short}%
11032   \renewcommand*{\GenericAcronymFields}{}%
11033   \renewcommand*{\acronymsort}[2]{##2}%
11034   \renewcommand*{\acronymentry}[1]{%
11035     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11036     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11037 }

```

g-sm-short-desc *<long>* (\textsmaller{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

11038 \renewacronymstyle{long-sm-short-desc}%
11039 {%
11040   \GlsUseAcrEntryDispStyle{long-sm-short}%
11041 }%
11042 {%
11043   \GlsUseAcrStyleDefs{long-sm-short}%
11044   \renewcommand*{\GenericAcronymFields}{}%

```

```

11045 \renewcommand*{\acronymsort}[2]{##2}%
11046 \renewcommand*{\acronymentry}[1]{%
11047   \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11048   (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11049 }

```

short-long-desc *(short)* (*{<long>}*) acronym style that has an accompanying description (which the user needs to supply).

```

11050 \renewacronymstyle{short-long-desc}%
11051 {%
11052   \GlsUseAcrEntryDispStyle{short-long}%
11053 }%
11054 {%
11055   \GlsUseAcrStyleDefs{short-long}%
11056   \renewcommand*{\GenericAcronymFields}{}%
11057   \renewcommand*{\acronymsort}[2]{##2}%
11058   \renewcommand*{\acronymentry}[1]{%
11059     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11060     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11061 }

```

short-long-desc *(long)* (*\textsc{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

11062 \renewacronymstyle{sc-short-long-desc}%
11063 {%
11064   \GlsUseAcrEntryDispStyle{sc-short-long}%
11065 }%
11066 {%
11067   \GlsUseAcrStyleDefs{sc-short-long}%
11068   \renewcommand*{\GenericAcronymFields}{}%
11069   \renewcommand*{\acronymsort}[2]{##2}%
11070   \renewcommand*{\acronymentry}[1]{%
11071     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11072     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11073 }

```

short-long-desc *(long)* (*\textsmaller{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

11074 \renewacronymstyle{sm-short-long-desc}%
11075 {%
11076   \GlsUseAcrEntryDispStyle{sm-short-long}%
11077 }%
11078 {%
11079   \GlsUseAcrStyleDefs{sm-short-long}%
11080   \renewcommand*{\GenericAcronymFields}{}%
11081   \renewcommand*{\acronymsort}[2]{##2}%
11082   \renewcommand*{\acronymentry}[1]{%
11083     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11084     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%

```

11085 }

dua *<long>* only acronym style.

11086 \renewacronymstyle{dua}%
11087 {%

Check for long form in case this is a mixed glossary.

11088 \ifdefempty\glscustomtext
11089 {%
11090 \ifglshaslong{\glslabel}%
11091 {%
11092 \glsifplural
11093 {%

Plural form:

11094 \glscapscase
11095 {%

Plural form, don't adjust case:

11096 \glslongpluralaccessdisplay{\glentrylongpl{\glslabel}}{\glslabel}%
11097 \glsinsert
11098 }%
11099 {%

Plural form, make first letter upper case:

11100 \glslongpluralaccessdisplay{\Glentrylongpl{\glslabel}}{\glslabel}%
11101 \glsinsert
11102 }%
11103 {%

Plural form, all caps:

11104 \glslongpluralaccessdisplay
11105 {\mfirstucMakeUppercase{\glentrylongpl{\glslabel}}}{\glslabel}%
11106 \mfirstucMakeUppercase{\glsinsert}%
11107 }%
11108 }%
11109 {%

Singular form

11110 \glscapscase
11111 {%

Singular form, don't adjust case:

11112 \glslongaccessdisplay{\glentrylong{\glslabel}}{\glslabel}\glsinsert
11113 }%
11114 {%

Subsequent singular form, make first letter upper case:

11115 \glslongaccessdisplay{\Glentrylong{\glslabel}}{\glslabel}\glsinsert
11116 }%
11117 {%

Subsequent singular form, all caps:

```

11118      \glslongaccessdisplay
11119      {\mfirstucMakeUppercase
11120       {\glsentrylong{\glslabel}\glsinsert}}{\glslabel}%
11121      \mfirstucMakeUppercase{\glsinsert}%
11122      }%
11123      }%
11124      }%
11125      {%

```

Not an acronym:

```

11126      \glsgenentryfmt
11127      }%
11128      }%
11129      {\glscustomtext\glsinsert}%
11130      }%
11131      {%
11132      \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
11133      \renewcommand*{\acrfullfmt}[3]{%
11134        \glslink[##1]{##2}{%
11135          \glslongaccessdisplay{\glsentrylong{##2}}{##2}##3\space
11136          (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}})%
11137      \renewcommand*{\Acrfullfmt}[3]{%
11138        \glslink[##1]{##2}{%
11139          \glslongaccessdisplay{\Glsentrylong{##2}}{##2}##3\space
11140          (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}})%
11141      \renewcommand*{\ACRfullfmt}[3]{%
11142        \glslink[##1]{##2}{%
11143          \glslongaccessdisplay
11144          {\mfirstucMakeUppercase{\glsentrylong{##2}}{##2}##3\space
11145          (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}})%
11146      \renewcommand*{\acrfullplfmt}[3]{%
11147        \glslink[##1]{##2}{%
11148          \glslongpluralaccessdisplay
11149          {\glsentrylongpl{##2}}{##2}##3\space
11150          (\glsshortpluralaccessdisplay
11151          {\acronymfont{\glsentryshortpl{##2}}}{##2}})%
11152      \renewcommand*{\ACRfullplfmt}[3]{%
11153        \glslink[##1]{##2}{%
11154          \glslongpluralaccessdisplay
11155          {\Glsentrylongpl{##2}}{##2}##3\space
11156          (\glsshortpluralaccessdisplay
11157          {\acronymfont{\glsentryshortpl{##2}}}{##2}})%
11158      \renewcommand*{\ACRfullplplfmt}[3]{%
11159        \glslink[##1]{##2}{%
11160          \glslongpluralaccessdisplay
11161          {\mfirstucMakeUppercase{\glsentrylongpl{##2}}{##2}##3\space
11162          (\glsshortpluralaccessdisplay
11163          {\acronymfont{\glsentryshortpl{##2}}}{##2}})%
11164      \renewcommand*{\glsentryfull}[1]{%

```

```

11165 \glslongaccessdisplay{\glsentrylong{##1}}\space
11166 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})%
11167 }%
11168 \renewcommand*{\Glsentryfull}[1]{%
11169 \glslongaccessdisplay{\Glsentrylong{##1}}{##1}\space
11170 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})%
11171 }%
11172 \renewcommand*{\glsentryfullpl}[1]{%
11173 \glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}\space
11174 (\glsshortpluralaccessdisplay{\acronymfont{\glsentryshortpl{##1}}}{##1})%
11175 }%
11176 \renewcommand*{\Glsentryfullpl}[1]{%
11177 \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}\space
11178 (\glsshortpluralaccessdisplay{\acronymfont{\glsentryshortpl{##1}}}{##1})%
11179 }%
11180 \renewcommand*{\acronymentry}[1]{%
11181 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}%
11182 \renewcommand*{\acronymsort}[2]{##1}%
11183 \renewcommand*{\acronymfont}[1]{##1}%
11184 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11185 }

```

dua-desc *<long>* only acronym style with user-supplied description.

```

11186 \renewacronymstyle{dua-desc}%
11187 {%
11188 \GlsUseAcrEntryDispStyle{dua}%
11189 }%
11190 {%
11191 \GlsUseAcrStyleDefs{dua}%
11192 \renewcommand*{\GenericAcronymFields}{}%
11193 \renewcommand*{\acronymentry}[1]{%
11194 \glslongaccessdisplay{\acronymfont{\glsentrylong{##1}}}{##1}}%
11195 \renewcommand*{\acronymsort}[2]{##2}%
11196 }%

```

footnote *<short>*\footnote{*<long>*} acronym style.

```

11197 \renewacronymstyle{footnote}%
11198 {%
11199 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
11200 }%
11201 {%
11202 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%

```

Need to ensure hyperlinks are switched off on first use:

```

11203 \glshyperfirstfalse
11204 \renewcommand*{\genacrfullformat}[2]{%
11205 \glsshortaccessdisplay
11206 {\protect\firstacronymfont{\glsentryshort{##1}}}{##1}##2%

```



```

11207 \protect\footnote{\glslongaccessdisplay{\glsentrylong{##1}}{##1}}%
11208 }%
11209 \renewcommand*{\Genacrfullformat}[2]{%
11210 \glsshortaccessdisplay
11211 {\firstacronymfont{\Glsentryshort{##1}}{##1}##2%
11212 \protect\footnote{\glslongaccessdisplay{\glsentrylong{##1}}{##1}}%
11213 }%
11214 \renewcommand*{\genplacrfullformat}[2]{%
11215 \glsshortpluralaccessdisplay
11216 {\protect\firstacronymfont{\Glsentryshortpl{##1}}{##1}##2%
11217 \protect\footnote{\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}}%
11218 }%
11219 \renewcommand*{\Genplacrfullformat}[2]{%
11220 \glsshortpluralaccessdisplay
11221 {\protect\firstacronymfont{\Glsentryshortpl{##1}}{##1}##2%
11222 \protect\footnote{\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}}%
11223 }%
11224 \renewcommand*{\acronymentry}[1]{%
11225 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}{##1}}%
11226 \renewcommand*{\acronymsort}[2]{##1}%
11227 \renewcommand*{\acronymfont}[1]{##1}%
11228 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%

```

Don't use footnotes for \acrfull:

```

11229 \renewcommand*{\acrfullfmt}[3]{%
11230 \glslink{##1}{##2}{%
11231 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}{##2}##3\space
11232 (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}}%
11233 \renewcommand*{\Acrfullfmt}[3]{%
11234 \glslink{##1}{##2}{%
11235 \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##2}}{##2}##3\space
11236 (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}}%
11237 \renewcommand*{\ACRfullfmt}[3]{%
11238 \glslink{##1}{##2}{%
11239 \glsshortaccessdisplay
11240 {\mfirstucMakeUppercase
11241 {\acronymfont{\glsentryshort{##2}}{##2}##3\space
11242 (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}}%
11243 \renewcommand*{\acrfullplfmt}[3]{%
11244 \glslink{##1}{##2}{%
11245 \glsshortpluralaccessdisplay
11246 {\acronymfont{\glsentryshortpl{##2}}{##2}##3\space
11247 (\glslongpluralaccessdisplay{\glsentrylongpl{##2}}{##2})}}}%
11248 \renewcommand*{\Acrfullplfmt}[3]{%
11249 \glslink{##1}{##2}{%
11250 \glsshortpluralaccessdisplay
11251 {\acronymfont{\Glsentryshortpl{##2}}{##2}##3\space
11252 (\glslongpluralaccessdisplay{\glsentrylongpl{##2}}{##2})}}}%
11253 \renewcommand*{\ACRfullplfmt}[3]{%
11254 \glslink{##1}{##2}{%

```

```

11255 \glsshortpluralaccessdisplay
11256 {\mfirstucMakeUppercase
11257 {\acronymfont{\glentryshortpl{##2}}{##2}##3\space
11258 (\glslongpluralaccessdisplay{\glentrylongpl{##2}}{##2}}}%

```

Similarly for \glentryfull etc:

```

11259 \renewcommand*{\glentryfull}[1]{%
11260 \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}{##1}\space
11261 (\glslongaccessdisplay{\glentrylong{##1}}{##1}}}%
11262 \renewcommand*{\Glsentryfull}[1]{%
11263 \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##1}}{##1}\space
11264 (\glslongaccessdisplay{\glentrylong{##1}}{##1}}}%
11265 \renewcommand*{\glentryfullpl}[1]{%
11266 \glsshortpluralaccessdisplay
11267 {\acronymfont{\glentryshortpl{##1}}{##1}\space
11268 (\glslongpluralaccessdisplay{\glentrylongpl{##1}}{##1}}}%
11269 \renewcommand*{\Glsentryfullpl}[1]{%
11270 \glsshortpluralaccessdisplay
11271 {\acronymfont{\Glsentryshortpl{##1}}{##1}\space
11272 (\glslongpluralaccessdisplay{\glentrylongpl{##1}}{##1}}}%
11273 }

```

footnote-sc \textsc{<short>}\footnote{<long>} acronym style.

```

11274 \renewacronymstyle{footnote-sc}%
11275 {%
11276 \GlsUseAcrEntryDispStyle{footnote}%
11277 }%
11278 {%
11279 \GlsUseAcrStyleDefs{footnote}%
11280 \renewcommand{\acronymentry}[1]{%
11281 \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}{##1}}
11282 \renewcommand{\acronymfont}[1]{\textsc{##1}}%
11283 \renewcommand*{\acrpluralsuffix}{\glstextup{\glspluralsuffix}}%
11284 }%

```

footnote-sm \textsmaller{<short>}\footnote{<long>} acronym style.

```

11285 \renewacronymstyle{footnote-sm}%
11286 {%
11287 \GlsUseAcrEntryDispStyle{footnote}%
11288 }%
11289 {%
11290 \GlsUseAcrStyleDefs{footnote}%
11291 \renewcommand{\acronymentry}[1]{%
11292 \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}{##1}}
11293 \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
11294 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11295 }%

```

footnote-desc <short>\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

11296 \renewacronymstyle{footnote-desc}%
11297 {%
11298   \GlsUseAcrEntryDisplayStyle{footnote}%
11299 }%
11300 {%
11301   \GlsUseAcrStyleDefs{footnote}%
11302   \renewcommand*{\GenericAcronymFields}{}%
11303   \renewcommand*{\acronymsort}[2]{##2}%
11304   \renewcommand*{\acronymentry}[1]{%
11305     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11306     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11307 }

```

ootnote-sc-desc \textsc{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

11308 \renewacronymstyle{footnote-sc-desc}%
11309 {%
11310   \GlsUseAcrEntryDisplayStyle{footnote-sc}%
11311 }%
11312 {%
11313   \GlsUseAcrStyleDefs{footnote-sc}%
11314   \renewcommand*{\GenericAcronymFields}{}%
11315   \renewcommand*{\acronymsort}[2]{##2}%
11316   \renewcommand*{\acronymentry}[1]{%
11317     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11318     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11319 }

```

ootnote-sm-desc \textsmaller{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

11320 \renewacronymstyle{footnote-sm-desc}%
11321 {%
11322   \GlsUseAcrEntryDisplayStyle{footnote-sm}%
11323 }%
11324 {%
11325   \GlsUseAcrStyleDefs{footnote-sm}%
11326   \renewcommand*{\GenericAcronymFields}{}%
11327   \renewcommand*{\acronymsort}[2]{##2}%
11328   \renewcommand*{\acronymentry}[1]{%
11329     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11330     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11331 }

```

Use \newacronymhook to modify the key list to set the access text to the long version by default.

```

11332 \renewcommand*{\newacronymhook}{%
11333   \edef\@gls@keylist{shortaccess=\the\glslongtok,%
11334     \the\glskeylisttok}%
11335   \expandafter\glskeylisttok\expandafter{\@gls@keylist}%

```

11336 }

1tNewAcronymDef Modify default style to use access text:

```
11337 \renewcommand*{\DefaultNewAcronymDef}{%
11338   \edef\@do@newglossaryentry{%
11339     \noexpand\newglossaryentry{\the\glslabeltok}%
11340     {%
11341       type=\acronymtype,%
11342       name={\the\glsshorttok},%
11343       description={\the\glslongtok},%
11344       descriptionaccess=\relax,%
11345       text={\the\glsshorttok},%
11346       access={\noexpand\@glo@textaccess},%
11347       sort={\the\glsshorttok},%
11348       short={\the\glsshorttok},%
11349       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11350       shortaccess={\the\glslongtok},%
11351       long={\the\glslongtok},%
11352       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11353       descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11354       first={\noexpand\glslongaccessdisplay
11355         {\the\glslongtok}{\the\glslabeltok}\space
11356         (\noexpand\glsshortaccessdisplay
11357           {\the\glsshorttok}{\the\glslabeltok})},%
11358       plural={\the\glsshorttok\acrpluralsuffix},%
11359       firstplural={\noexpand\glslongpluralaccessdisplay
11360         {\noexpand\@glo@longpl}{\the\glslabeltok}\space
11361         (\noexpand\glsshortpluralaccessdisplay
11362           {\noexpand\@glo@shortpl}{\the\glslabeltok})},%
11363       firstaccess=\relax,%
11364       firstpluralaccess=\relax,%
11365       textaccess={\noexpand\@glo@shortaccess},%
11366       \the\glskeylisttok
11367     }%
11368   }%
11369   \let\@org@gls@assign@firstpl\gls@assign@firstpl
11370   \let\@org@gls@assign@plural\gls@assign@plural
11371   \let\@org@gls@assign@descplural\gls@assign@descplural
11372   \def\gls@assign@firstpl##1##2{%
11373     \@gls@expand@field{##1}{firstpl}{##2}%
11374   }%
11375   \def\gls@assign@plural##1##2{%
11376     \@gls@expand@field{##1}{plural}{##2}%
11377   }%
11378   \def\gls@assign@descplural##1##2{%
11379     \@gls@expand@field{##1}{descplural}{##2}%
11380   }%
11381   \@do@newglossaryentry
11382   \let\gls@assign@firstpl\@org@gls@assign@firstpl
```

```

11383 \let\gls@assign@plural\@org@gls@assign@plural
11384 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11385 }

```

teNewAcronymDef

```

11386 \renewcommand*{\DescriptionFootnoteNewAcronymDef}{%
11387   \edef\@do@newglossaryentry{%
11388     \noexpand\newglossaryentry{\the\glslabeltok}%
11389     {%
11390       type=\acronymtype,%
11391       name={\noexpand\acronymfont{\the\glsshorttok}},%
11392       sort={\the\glsshorttok},%
11393       text={\the\glsshorttok},%
11394       short={\the\glsshorttok},%
11395       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11396       shortaccess={\the\glslongtok},%
11397       long={\the\glslongtok},%
11398       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11399       access={\noexpand\@glo@textaccess},%
11400       plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11401       symbol={\the\glslongtok},%
11402       symbolplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11403       firstpluralaccess=\relax,
11404       textaccess={\noexpand\@glo@shortaccess},%
11405       \the\glskeylisttok
11406     }%
11407   }%
11408   \let\@org@gls@assign@firstpl\gls@assign@firstpl
11409   \let\@org@gls@assign@plural\gls@assign@plural
11410   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
11411   \def\gls@assign@firstpl##1##2{%
11412     \@@gls@expand@field{##1}{firstpl}{##2}%
11413   }%
11414   \def\gls@assign@plural##1##2{%
11415     \@@gls@expand@field{##1}{plural}{##2}%
11416   }%
11417   \def\gls@assign@symbolplural##1##2{%
11418     \@@gls@expand@field{##1}{symbolplural}{##2}%
11419   }%
11420   \@do@newglossaryentry
11421   \let\gls@assign@plural\@org@gls@assign@plural
11422   \let\gls@assign@firstpl\@org@gls@assign@firstpl
11423   \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11424 }

```

onNewAcronymDef

```

11425 \renewcommand*{\DescriptionNewAcronymDef}{%
11426   \edef\@do@newglossaryentry{%
11427     \noexpand\newglossaryentry{\the\glslabeltok}%

```

```

11428 {%
11429     type=\acronymtype,%
11430     name={\noexpand
11431         \acrnameformat{\the\glsshorttok}{\the\glslongtok}},%
11432     access={\noexpand\@glo@textaccess},%
11433     sort={\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     first={\the\glslongtok},%
11440     firstaccess=\relax,
11441     firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11442     text={\the\glsshorttok},%
11443     textaccess={\the\glslongtok},%
11444     plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11445     symbol={\noexpand\@glo@text},%
11446     symbolaccess={\noexpand\@glo@textaccess},%
11447     symbolplural={\noexpand\@glo@plural},%
11448     firstpluralaccess=\relax,
11449     textaccess={\noexpand\@glo@shortaccess},%
11450     \the\glskeylisttok}%
11451 }%
11452 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11453 \let\@org@gls@assign@plural\gls@assign@plural
11454 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
11455 \def\gls@assign@firstpl##1##2{%
11456     \@gls@expand@field{##1}{firstpl}{##2}%
11457 }%
11458 \def\gls@assign@plural##1##2{%
11459     \@gls@expand@field{##1}{plural}{##2}%
11460 }%
11461 \def\gls@assign@symbolplural##1##2{%
11462     \@gls@expand@field{##1}{symbolplural}{##2}%
11463 }%
11464 \@do@newglossaryentry
11465 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11466 \let\gls@assign@plural\@org@gls@assign@plural
11467 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11468 }

```

teNewAcronymDef

```

11469 \renewcommand*{\FootnoteNewAcronymDef}{%
11470     \edef\@do@newglossaryentry{%
11471         \noexpand\newglossaryentry{\the\glslabeltok}%
11472         {%
11473             type=\acronymtype,%
11474             name={\noexpand\acronymfont{\the\glsshorttok}},%

```

```

11475     sort={\the\glsshorttok},%
11476     text={\the\glsshorttok},%
11477     textaccess={\the\glslongtok},%
11478     access={\noexpand\@glo@textaccess},%
11479     plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11480     short={\the\glsshorttok},%
11481     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11482     long={\the\glslongtok},%
11483     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11484     description={\the\glslongtok},%
11485     descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11486     \the\glskeylisttok
11487   }%
11488 }%
11489 \let\@org@gls@assign@plural\gls@assign@plural
11490 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11491 \let\@org@gls@assign@descplural\gls@assign@descplural
11492 \def\gls@assign@firstpl##1##2{%
11493   \@@gls@expand@field{##1}{firstpl}{##2}%
11494 }%
11495 \def\gls@assign@plural##1##2{%
11496   \@@gls@expand@field{##1}{plural}{##2}%
11497 }%
11498 \def\gls@assign@descplural##1##2{%
11499   \@@gls@expand@field{##1}{descplural}{##2}%
11500 }%
11501 \do@newglossaryentry
11502 \let\gls@assign@plural\@org@gls@assign@plural
11503 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11504 \let\gls@assign@descplural\@org@gls@assign@descplural
11505 }

```

11NewAcronymDef

```

11506 \renewcommand*{\SmallNewAcronymDef}{%
11507   \edef\@do@newglossaryentry{%
11508     \noexpand\newglossaryentry{\the\glslabeltok}%
11509     {%
11510       type=\acronymtype,%
11511       name={\noexpand\acronymfont{\the\glsshorttok}},%
11512       access={\noexpand\@glo@symbolaccess},%
11513       sort={\the\glsshorttok},%
11514       short={\the\glsshorttok},%
11515       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11516       shortaccess={\the\glslongtok},%
11517       long={\the\glslongtok},%
11518       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11519       text={\noexpand\@glo@short},%
11520       textaccess={\noexpand\@glo@shortaccess},%
11521       plural={\noexpand\@glo@shortpl},%

```

```

11522     first={\the\glslongtok},%
11523     firstaccess=\relax,
11524     firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11525     description={\noexpand\@glo@first},%
11526     descriptionplural={\noexpand\@glo@firstplural},%
11527     symbol={\the\glsshorttok},%
11528     symbolaccess={\the\glslongtok},%
11529     symbolplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11530     \the\glskeylisttok
11531 }%
11532 }%
11533 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11534 \let\@org@gls@assign@plural\gls@assign@plural
11535 \let\@org@gls@assign@descplural\gls@assign@descplural
11536 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
11537 \def\gls@assign@firstpl##1##2{%
11538   \@@gls@expand@field{##1}{firstpl}{##2}%
11539 }%
11540 \def\gls@assign@plural##1##2{%
11541   \@@gls@expand@field{##1}{plural}{##2}%
11542 }%
11543 \def\gls@assign@descplural##1##2{%
11544   \@@gls@expand@field{##1}{descplural}{##2}%
11545 }%
11546 \def\gls@assign@symbolplural##1##2{%
11547   \@@gls@expand@field{##1}{symbolplural}{##2}%
11548 }%
11549 \do@newglossaryentry
11550 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11551 \let\gls@assign@plural\@org@gls@assign@plural
11552 \let\gls@assign@descplural\@org@gls@assign@descplural
11553 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11554 }

```

The following are kept for compatibility with versions before 3.0:

sshortaccesskey

```
11555 \newcommand*{\glsshortaccesskey}{\glsshortkey access}%
```

pluralaccesskey

```
11556 \newcommand*{\glsshortpluralaccesskey}{\glsshortpluralkey access}%
```

lslongaccesskey

```
11557 \newcommand*{\glslongaccesskey}{\glslongkey access}%
```

pluralaccesskey

```
11558 \newcommand*{\glslongpluralaccesskey}{\glslongpluralkey access}%
```


5.5 Debugging Commands

owglonameaccess

```
11559 \newcommand*{\showglonameaccess}[1]{%  
11560   \expandafter\show\csname glo@\glsdetoklabel{#1}@textaccess\endcsname  
11561 }
```

owglotextaccess

```
11562 \newcommand*{\showglotextaccess}[1]{%  
11563   \expandafter\show\csname glo@\glsdetoklabel{#1}@textaccess\endcsname  
11564 }
```

glopluralaccess

```
11565 \newcommand*{\showglopluralaccess}[1]{%  
11566   \expandafter\show\csname glo@\glsdetoklabel{#1}@pluralaccess\endcsname  
11567 }
```

wglofirstaccess

```
11568 \newcommand*{\showglofirstaccess}[1]{%  
11569   \expandafter\show\csname glo@\glsdetoklabel{#1}@firstaccess\endcsname  
11570 }
```

rstpluralaccess

```
11571 \newcommand*{\showglofirstpluralaccess}[1]{%  
11572   \expandafter\show\csname glo@\glsdetoklabel{#1}@firstpluralaccess\endcsname  
11573 }
```

glosymbolaccess

```
11574 \newcommand*{\showglosymbolaccess}[1]{%  
11575   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolaccess\endcsname  
11576 }
```

bolpluralaccess

```
11577 \newcommand*{\showglosymbolpluralaccess}[1]{%  
11578   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolpluralaccess\endcsname  
11579 }
```

owglodescaccess

```
11580 \newcommand*{\showglodescaccess}[1]{%  
11581   \expandafter\show\csname glo@\glsdetoklabel{#1}@descaccess\endcsname  
11582 }
```

escpluralaccess

```
11583 \newcommand*{\showglodescpluralaccess}[1]{%  
11584   \expandafter\show\csname glo@\glsdetoklabel{#1}@descpluralaccess\endcsname  
11585 }
```

wgloshortaccess

```
11586 \newcommand*{\showgloshortaccess}[1]{%
11587   \expandafter\show\csname glo@\glsdetoklabel{#1}@shortaccess\endcsname
11588 }
```

ortpluralaccess

```
11589 \newcommand*{\showgloshortpluralaccess}[1]{%
11590   \expandafter\show\csname glo@\glsdetoklabel{#1}@shortpluralaccess\endcsname
11591 }
```

owglolongaccess

```
11592 \newcommand*{\showglolongaccess}[1]{%
11593   \expandafter\show\csname glo@\glsdetoklabel{#1}@longaccess\endcsname
11594 }
```

ongpluralaccess

```
11595 \newcommand*{\showglolongpluralaccess}[1]{%
11596   \expandafter\show\csname glo@\glsdetoklabel{#1}@longpluralaccess\endcsname
11597 }
```

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.

```
11598 \NeedsTeXFormat{LaTeX2e}
11599 \ProvidesPackage{glossaries-babel}[2016/12/16 v4.27 (NLCT)]
```

Load tracklang to obtain language settings.

```
11600 \RequirePackage{tracklang}
11601 \let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
11602 \AnyTrackedLanguages
11603 {%
11604   \ForEachTrackedDialect{\this@dialect}{%
11605     \IfTrackedLanguageFileExists{\this@dialect}%
11606       {glossaries-}% prefix
11607       {.ldf}%
11608       {%
11609         \RequireGlossariesLang{\CurrentTrackedTag}%
11610       }%
11611     {%
11612       \PackageWarningNoLine{glossaries}%
11613         {No language module detected for ‘\this@dialect’.\MessageBreak
11614           Language modules need to be installed separately.\MessageBreak
11615           Please check on CTAN for a bundle called\MessageBreak
11616             ‘glossaries-\CurrentTrackedLanguage’ or similar}%
11617     }%
11618   }%
11619 }%
11620 {}%
```

6.1 Polyglossia Captions

Language support has now been split off into independent language modules.

```
11621 \NeedsTeXFormat{LaTeX2e}
11622 \ProvidesPackage{glossaries-polyglossia}[2016/12/16 v4.27 (NLCT)]
```

Load tracklang to obtain language settings.

```
11623 \RequirePackage{tracklang}
11624 \let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
11625 \AnyTrackedLanguages
```

```

11626 {%
11627     \ForEachTrackedDialect{\this@dialect}{%
11628         \IfTrackedLanguageFileExists{\this@dialect}%
11629         {glossaries-}% prefix
11630         {.ldf}%
11631         {%
11632             \RequireGlossariesLang{\CurrentTrackedTag}%
11633         }%
11634         {%
11635             \PackageWarningNoLine{glossaries}%
11636             {No language module detected for ‘\this@dialect’.\MessageBreak
11637             Language modules need to be installed separately.\MessageBreak
11638             Please check on CTAN for a bundle called\MessageBreak
11639             ‘glossaries-\CurrentTrackedLanguage’ or similar}%
11640         }%
11641     }%
11642 }%
11643 {}%

```

Glossary

`makeindex` An indexing application. [10](#), [25](#), [26](#), [174](#)

`xindy` An flexible indexing application with multilingual support written in Perl. [10](#), [25](#), [26](#), [174](#)

Change History

1.01 (2007-05-17)	numberline: numberline option added . . . 6
General: Added range facility in format key 109	1.12 (2008-03-08)
\writeist: Added spaces after \delimN and \delimR in ist file 156	\@GLSpl: now uses \glentrydescplural and \glentrysymbolplural instead of \glentrydesc and \glentrysymbol 123
1.04 (2007-08-03)	\@Glspl@: now uses \glentrydescplural and \glentrysymbolplural instead of \glentrydesc and \glentrysymbol 122
1.05 (2007-08-10)	\@Glspl@: now uses \glentrydescplural and \glentrysymbolplural instead of \glentrydesc and \glentrysymbol 121
\glossarysection: added \@mkboth to \glossarysection 37	General: added check for \hypertarget separate to \hyperlink (memoir defines \hyperlink but not \hypertarget) 117
\gls@defglossaryentry: Changed the default value of the sort key to just the value of the name key 78	descriptionplural: new 60
1.07 (2007-09-13)	\gls@defglossaryentry: Changed default first plural to be first key with s appended (was text key with s appended) 77
\@gls@link: fixed bug caused by \theglentrycounter setting the page number too soon 107	descriptionplural support added 77
\glsadd: fixed bug caused by \theglentrycounter setting the page number too soon 153	symbolplural support added 77
1.08 (2007-10-13)	\Glsentrydescplural: New 147
General: Added babel support 31	\Glsentrydescplural: New 147
listgroup: changed listgroup style to use \glsgetgrouptitle 266	\Glsentrysymbolplural: New 148
altlistgroup: changed altlistgroup style to use \glsgetgrouptitle 267	\Glsentrysymbolplural: New 148
1.1 (2008-02-22)	\SetDescriptionFootnoteAcronymStyle: Added \protect before \footnote and \glslink 234
\@glossarysection: numbered sections and auto label added 39	\SetFootnoteAcronymStyle: Added \protect before \footnote and \glslink 240
\@gls@tmpb: changed \toksdef to \newtoks 111	symbolplural: new 61
\@gls@toc: numberline added 40	
\@p@glossarysection: numbered sections and auto label added 39	
General: amsgen now loaded (\new@ifnextchar needed) 4	
translate: translate option added 23	
\setglossarysection: new 38	
numberedsection: numberedsection package option added 7	

1.13 (2008-05-10)	
General: fixed bug that ignored 3rd parameter	125-132
\ACRfullpl: new	215
\Acrfullpl: new	214
\acrfullpl: new	214
\acrpluralsuffix: New	212
\gls@defglossaryentry: Changed default first value	77
Changed default firstplural value	77
Removed restriction on only using \newglossaryentry in the preamble	83
\newacronym: Removed restriction on only using \newacronym in the preamble	212
1.14 (2008-06-17)	
\@gls@hypergroup: new	262
General: added nonumberlist key to \printglossary	198
added numberedsection key to \printglossary	196
\firstacronymfont: new	215
\glsautoprefix: new	7
\glsnavhyperlink: changed \edef to \protected@edef	261
\glsnavhypertarget: added write to aux file	261
\glsnavigation: changed to only use labels for groups that are present ..	262
1.15 (2008-08-15)	
\@gls@link: added \glslabel	107
\gls@defglossaryentry: check for \@glo@first in description	81
check for \@glo@text in symbol	82
\gls@hypergroup: new	262
\glsnavhypertarget: added check if rerun required	261
\glssettoctitle: new	31
\printglossary: changed the way the TOC title is set	182
1.16 (2008-08-27)	
\@GLS@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	121
\@GLSpl: Test glossary type is \acronymtype in addition to checking if footnote option has been used	123
\@GLS@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	120
\@GLspl@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	122
\@gls@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	119
\@glsdisp: Test glossary type is \acronymtype in addition to checking if footnote option has been used	124
\@GLspl@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	121
\@gls@target: raised the hypertarget so the target text doesn't scroll off the top of the page	118
\gls@defglossaryentry: Changed def to let	77
1.17 (2008-12-26)	
\@do@wrglossary: new	177
\@do@seeglossary: new	180
\@glo@storeentry: new	84
\@gls@glossary: changed definition to use \index instead of \@index	175
\@glsdefaultplural: new	65
\@glsdefaultsort: new	65
\@gls@hypernumber: new	208
\@gls@noname: new	64
\@gls@nonextpages: new	198
General: added xindy support	25
parent: new	62
see: new	62
\gls@defglossaryentry: added nonumberlist key	78
added parent key	78
added see key	78
Stored main part of entry format when entry is defined	82
\gls@suffixF: new	35
\gls@suffixFF: new	36
\gls@wrglossary: modified to allow for xindy support	175

\glshyperlink: new	153	\SetDescriptionFootnoteAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	234
\glshypernumber: modified to allow material to be attached to location	208	\SetFootnoteAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	240
\glshnavhyperlink: replaced \hyperlink to \@glslink	261	\SetSmallAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	243
\glshnavhypertarget: replaced \hypertarget to \@glstarget ...	261	2.01 (2009 May 30) \@glsl@link: moved \@do@wrglossary before term is displayed to prevent unwanted whatsit	108
\glsssee: new	181	\forall glossaries: replaced \ifthenelse with \ifx	49
\glssseeformat: new	181	\forall glossentries: replaced \ifthenelse with \ifx	49
\glssSetSuffixF: new	36	\glssdefmain: new	13
\glssSetSuffixFF: new	36	\glssdescwidth: changed \linewidth to \hsize	269, 291
\ifglssxindy: new	25	\glsslistdottedwidth: changed \linewidth to \hsize	268
\listfilename: added xindy support ...	34	\glsspagelistwidth: changed \linewidth to \hsize	269, 291
\newglossarystyle: made \newglossarystyle long	207	nomain: added nomain package option	13
\nopostdesc: new	34	\writeist: removed item_02 - no such makeindex key	160
nonumberlist: new	62	2.02 (2007-07-13) \@printglossary: suppressed warning globally rather than locally	185
\printglossary: added check to determine if \printglossary is already defined	182	2.02 (2009-07-13) \glossarysection: changed \@mkboth to \glossarymark	37
added print language to aux file	182	\glssglossarymark: New	38
order: order package option added ...	25	2.03 (2009-09-23) \@GLS@: Added check for hyperfirst ...	121
\writeist: added xindy support	156	\@GLSpl: Added check for hyperfirst ...	123
1.18 (2009-01-14) \@glsl@loadlist: new	9	\@GLS@: Added check for hyperfirst ...	120
\@glsl@loadlong: new	8	\@GLspl@: Added check for hyperfirst ..	122
\@glsl@loadsuper: new	9	\@glsl@: Added check for hyperfirst ...	119
\@glsl@loadtree: new	9	\@glsl@@link: new	106
\glsl@defglossaryentry: Changed default value of sort to \@glsldefaultsort	78	\@glsl@link: added \leavevmode ...	107
moved sort sanitization to \newglossaryentry	82	Moved entry existence check to avoid duplicate code	107
\glstarget: new	201	\@glsl@disp: Added check for hyperfirst	124
\oldacronym: new	211	\@glspl@: Added check for hyperfirst ..	121
nolist: new	9	\glssglossarymark: Added check to see if it's already defined	38
nolong: new	8	hyperfirst: new	24
sort: moved sanitization to \newglossaryentry	60		
nostyles: new	9		
nosuper: new	9		
notree: new	9		
1.19 (2009-03-02) \glsclearpage: new	40		
\glssdisp: new	123		
\SetDescriptionAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	238		

2.04 (2009-11-10)	
\@GLS@: Changed test to check if glossary type has been identified as a list of acronyms	121
\@GLSpl: Changed test to check if glossary type has been identified as a list of acronyms	123
\@GLs@: Changed test to check if glossary type has been identified as a list of acronyms	120
\@GLspl@: Changed test to check if glossary type has been identified as a list of acronyms	122
\@glossaryentryfield: new	83
\@glossarysubentryfield: new	83
\@gls@: Changed test to check if glossary type has been identified as a list of acronyms	119
\@glsacronymlists: new	15
\@glsdisp: Changed test to check if glossary type has been identified as a list of acronyms	124
\@glspl@: Changed test to check if glossary type has been identified as a list of acronyms	121
\@newglossaryentryposthook: new ..	83
\@newglossaryentryprehook: new ...	83
acronymlists: new	16
\DeclareAcronymList: new	15
\DefineAcronymSynonyms: new	228
\gls@defglossaryentry: added user1-6 keys	78
\glsadd: fixed bug that ignored counter	153
\Glsentryuseri: new	149
\glsentryuseri: new	149
\Glsentryuserii: new	149
\glsentryuserii: new	149
\Glsentryuseriii: new	149
\glsentryuseriii: new	149
\Glsentryuseriv: new	150
\glsentryuseriv: new	150
\Glsentryuserv: new	150
\glsentryuserv: new	150
\Glsentryuservi: new	150
\glsentryuservi: new	150
\ns@newglossary: added check to determine if \gls@<type>@display and \gls@<type>@displayfirst have been defined.	57
\SetAcronymLists: new	16
\SetDefaultAcronymDisplayStyle: new	230
\SetDefaultAcronymStyle: new	231
\SetDescriptionAcronymDisplayStyle: new	236
\SetDescriptionDUAAcronymDisplayStyle: new	234
\SetDescriptionFootnoteAcronymDisplayStyle: new	232
\SetDUADisplayStyle: new	243
\SetFootnoteAcronymDisplayStyle: new	238
\SetSmallAcronymDisplayStyle: new	241
2.05 (2010-02-06)	
\@glsdisp: Added closing brace. Patch provided by Sergiu Dotenco	124
Removed spurious brace. Patch provided by Sergiu Dotenco	124
\writeist: Added \string before opening and closing braces. Patch provided by Segiu Dotenco	161
2.06 (2010-06-14)	
\altnewglossary: new	58
\CustomAcronymFields: new	246
\CustomNewAcronymDef: new	246
\SetCustomDisplayStyle: new	245
\SetCustomStyle: new	246
2.07 (2010-07-10)	
General: glsadd format key stored in \@glsnumberformat (was mistakenly stored in \@glo@format)	153
3.0 (2010-07-12)	
\@makeglossary: Added check for savewrites	165
\gls@wrglossary: modified to take into account savewrites	175
3.0 (2010/03/31)	
\@set@glo@numformat: added 4th argument	109
3.0 (2011-04-02)	
\@do@wrglossary: added check for hyper location prefix	178
modified to use new format	177
\@@glossarysec: replaced \@ifundefined with \ifcsundef ...	6
\@do@seeglossary: Sanitize and escape cross-referencing information	180
\@gls@counterwithin: new	10

\@gls@ifinlist: new	41	\glsadd: added	
\@gls@link: added		\@gls@saveentrycounter	154
\@gls@saveentrycounter	108	\GlsAddXdyCounters: new	41
added \@gls@setsort	108	\glentrycounterlabel: new	200
\@gls@saveentrycounter: new	108	\glentryitem: new	201
\@gls@setupsort@def: new	11	\Glentrylong: new	151
\@gls@setupsort@standard: new	11	\glentrylong: new	151
\@gls@setupsort@use: new	12	\Glentrylongpl: new	151
\@gls@xdy@locationlist: new	44	\glentrylongpl: new	151
\@glslink: replaced \@ifundefined		\Glentryshort: new	150
with \ifcsundef	117	\glentryshort: new	150
\@glsnextpages: new	198	\Glentryshortpl: new	151
\@print@glossary: replaced		\glentryshortpl: new	150
\@ifundefined with \ifcsundef ..	185	\glsgetgrouptitle: replaced	
\@printglossary: added		\@ifundefined with \ifcsundef ..	205
\currentglossary	184	\gls glossarymark: replaced	
added \glsnextpages	184	\@ifundefined with \ifcsundef ..	38
make toctitle default to title	184	\glshyperlink: changed default from	
\@xdyattributelist: new	41	\glentryname to \glentrytext ..	153
General: added prefix to hyperlink	209	\glshypernumber: replaced	
etoolbox now loaded	4	\@ifundefined with \ifcsundef ..	208
replaced \@ifundefined with		\glsnumberformat: replaced	
\ifcsundef	29, 32, 104, 196	\@ifundefined with \ifcsundef ..	36
\acrfootnote: new	231	\glsrefentry: new	200
\ACRfull: added starred version	213	\glsresetsubentrycounter: new ...	199
\Acrfull: added starred version	213	\glsseeitem: hyperlink uses	
\acrfull: added starred version	212	\glsseeitemformat instead of	
\ACRfullpl: added starred version ...	215	\glentryname	181
\Acrfullpl: added starred version ...	214	\glsseeitemformat: new	181
\acrfullpl: added starred version ...	214	\glssortnumberfmt: new	11
\acrlinkfootnote: new	231	\glsstepentry: new	200
\acrno linkfootnote: new	232	\glsstepsubentry: new	200
save writes: new	27	\glssubentrycounterlabel: new ...	201
see: added \@glo@seeautonumberlist ..	62	\glssubentryitem: new	201
seeautonumberlist: new	8	theglossary: replaced \@ifundefined	
\glossarysection: replaced		with \ifcsundef	201
\@ifundefined with \ifcsundef ..	37	short: new	64
\glossarystyle: replaced		shortplural: new	64
\@ifundefined with \ifcsundef ..	207	\ifglossaryexists: replaced	
\gls@codepage: replaced		\@ifundefined with \ifcsundef ..	50
\@ifundefined with \ifcsundef ..	26	\ifglentryexists: replaced	
\gls@defglossaryentry: added		\@ifundefined with \ifcsundef ..	51
\@gls@defsort	82	\istfile: deprecated	173
added short and long keys	78	glossaryentry: new	199
replaced \@ifundefined with		glossarysubentry: new	199
\ifcsundef	78	\newglossaryentry: replaced	
\gls@doclearpage: replaced		\DeclareRobustCommand with	
\@ifundefined with \ifcsundef ..	39	\newrobustcmd	67

\newglossarystyle: replaced		\showglouseriii: new	249
\@ifundefined with \ifcsundef .	207	\showglouseriv: new	249
\ns@newglossary: added		\showglouserv: new	249
\@gls@defsortcount	58	\showglouservi: new	250
replaced \@ifundefined with		subentrycounter: new	10
\ifcsundef	57	\writeist: added xindy-only macro	
entrycounter: new	10	definitions to glossary open tag	158
entrycounterwithin: new	10	modified to support new format	156
\oldacronym: replaced \@ifundefined		3.01 (2011-04-12)	
with \ifcsundef	211	\@glswritefiles: added check for	
compatible-2.07: compatible-2.07		empty glossaries	173
option added	27	General: made robust	120
long: new	64	\ACRfull: made robust	213
longplural: new	64	\Acrfull: made robust	213
nonumberlist: now boolean	62	\acrfull: made robust	212
sort: new	10	\acrfullformat: removed	
counter: replaced \@ifundefined with		\acronymfont as it should already be	
\ifcsundef	61	set in the second argument.	213
\printglossary: replaced		\ACRfullpl: made robust	215
\@ifundefined with \ifcsundef .	182	\Acrfullpl: made robust	214
\SetDescriptionFootnoteAcronymDisplayStyle		\acrfullpl: made robust	214
expanded options link options	232	\ACRlong: made robust	142
\setentrycounter: added optional		\Acrlong: made robust	141
argument	206	\acrlong: made robust	140
\showacronymlists: new	252	\ACRlongpl: made robust	144
\showglocounter: new	249	\Acrlongpl: made robust	143
\showgloDESC: new	250	\acrlongpl: made robust	142
\showgloDESCplural: new	250	\ACRshort: made robust	138
\showglofirst: new	248	\Acrshort: made robust	137
\showglofirstpl: new	248	\acrshort: made robust	137
\showgloflag: new	251	\ACRshortpl: made robust	140
\showgloindex: new	251	\Acrshortpl: made robust	139
\showglolevel: new	248	\acrshortpl: made robust	139
\showglongame: new	250	\Gls: made robust	119
\showgloparent: new	247	\glsadd: made robust	153
\showgloplural: new	248	\glsaddall: made robust	154
\showglosort: new	250	\GLSdesc: made robust	129
\showglossaries: new	252	\Glsdesc: made robust	129
\showglossarycounter: new	253	\glsdesc: made robust	129
\showglossaryentries: new	253	\GLSdescplural: made robust	130
\showglossaryin: new	252	\Glsdescplural: made robust	130
\showglossaryout: new	252	\glsdescplural: made robust	129
\showglossarytitle: new	252	\glsfirst: made robust	125
\showglosymbol: new	250	\GLSfirstplural: made robust	128
\showglosymbolplural: new	251	\Glsfirstplural: made robust	127
\showglotext: new	248	\glsfirstplural: made robust	127
\showglotype: new	248	\glslink: made robust	106
\showglouserI: new	249	\GLSname: made robust	128
\showglouserII: new	249	\Glsname: made robust	128

\glsname: made robust	128	\@printglossary: add a way to fetch	
\GLSpl: made robust	123	current entry label	184
\Glspl: made robust	122	savenumberlist: new	8
\glspl: made robust	121	ucmark: new	10
\GLSplural: made robust	127	\gls@defglossaryentry: added	
\GLSsymbol: made robust	131	numberlist element	81
\Glsymbol: made robust	131	\gls@save@numberlist: new	182
\glssymbol: made robust	130	\gls@wrglossary: added check for	
\GLSsymbolplural: made robust	132	glossary file defined	175
\Glsymbolplural: made robust	131	\glsdisplaynumberlist: new	152
\glssymbolplural: made robust	131	\glsentrycounter: set default value ..	108
\Glstext: made robust	125	\Glsentryfull: fixed bug (replaced	
\glstext: made robust	125	\glsentryshortpl with	
\GLSuseri: made robust	132	\glsentryshort)	151
\Glsuseri: made robust	132	\glsentryfullpl: fixed bug (replaced	
\glsuseri: made robust	132	\glsentryshort with	
\GLSuserii: made robust	133	\glsentryshortpl)	151
\Glsuserii: made robust	133	\glsentrynumberlist: new	152
\glsuserii: made robust	133	\glsmoveentry: new	83
\GLSuseriii: made robust	134	\glsresetsubentrycounter: new ...	200
\Glsuseriii: made robust	134	\ifglshaschildren: new	52
\glsuseriii: made robust	134	\ifglshasparent: new	53
\GLSuseriv: made robust	135	\makeglossaries: added list parser ..	168
\Glsuseriv: made robust	135	indexonlyfirst: new	24
\glsuseriv: made robust	134	\renewglossarystyle: new	207
\GLSuserv: made robust	136	\showglossaryentries: fixed misspelt	
\Glsuserv: made robust	135	command	253
\glsuserv: made robust	135	\SmallNewAcronymDef: fixed broken	
\GLSuservi: made robust	137	short and long plural	241
\Glsuservi: made robust	136	3.03 (2012/09/21)	
\glsuservi: made robust	136	\@gls@sanitizesort: new	18
3.02 (2012-05-19)		\@gls@setupsort@standard: used	
\glsnumlistlastsep: new	153	\@gls@sanitizesort	11
\glsnumlistsep: new	153	\@printglossary: allow title to override	
3.02 (2012-05-21)		default toctitle	183
\@do@wrglossary: changed		General: allow title to set toctitle	195
\@glslocref to		\glsinlinedescformat: new	265
\theglsentrycounter	179	\glsinlineemptydescformat: new ..	265
\@do@wrglossary: changed		\glsinlinenameformat: new	265
\@do@wr@glossary to test for		\glsinlinepostchild: new	265
indexonlyfirst option; put old		\glsinlinesubdescformat: new	265
\@do@wr@glossary code into		\glsinlinesubnameformat: new	265
\@do@wrglossary	175	\glspostinline: replaced “.” with	
\@gls@missingnumberlist: new	65	\glspostdescription	265
\@glswritefiles: added check for		altlongragged4col: added check for	
existence of token in case		glsnogroupskip	284
\makeglossaries has been		altsuperragged4col: added check for	
omitted	173	glsnogroupskip	302

alttree: added check for		\gls@disablepagerefexpansion: new	176
glsnogroupskip	311	\gls@numberpage: new	176
index: added check for glsnogroupskip	305	\gls@protected@pagefmts: new	176
nogroupskip: new	9	\gls@romanpage: new	176
long: added check for glsnogroupskip	270	\glsdefmain: added check for doc	
long3col: added check for		package	13
glsnogroupskip	271	\glsorg@endtheglossary: new	5
long4col: added check for		\glsorg@theglossary: new	5
glsnogroupskip	273	\PrintChanges: new	5
longragged: added check for		3.05 (2013-04-21)	
glsnogroupskip	280	\@do@wrglossary: add Roman case.	
longragged3col: added check for		Fixed bugs in the else statements ..	177
glsnogroupskip	282	\@gls@link: added check for	
no postdot: new	9	“nohypertypes”	107
tree: added check for glsnogroupskip	307	mcolalttree: replaced ‘2’ with	
treenoname: added check for		\glsmcols	289
glsnogroupskip	308	mcolindex: replaced ‘2’ with \glsmcols	285
super: added check for glsnogroupskip	292	mcolindexspannav: replaced ‘2’ with	
super3col: added check for		\glsmcols	286
glsnogroupskip	293	mcoltree: replaced ‘2’ with \glsmcols	287
super4col: added check for		mcoltreenoname: replaced ‘2’ with	
glsnogroupskip	295	\glsmcols	288
superragged: added check for		mcoltreesspannav: replaced ‘2’ with	
glsnogroupskip	298	\glsmcols	288
superragged3col: added check for		\gls@protected@pagefmts: added	
glsnogroupskip	300	Roman to list	176
3.04 (2012-11-11)		\gls@Romanpage: new	176
altlist: replaced \newline with		\glsgetgrouplabel: fixed bug (typo in	
paragraph break	267	\equal)	206
3.04 (2012-11-18)		\nopostdesc: made robust	34
\@do@wrglossary: changed		3.05 (2013/04/21)	
\theglsentrycounter back to		\@gls@nohyperlist: new	16
\@glslocref	179	\GlsDeclareNoHyperList: new	16
\@do@wrglossary: modified to		nohypertypes: new	16
compensate for possible incorrect		3.06 (2013/06/17)	
page number	177	\@xdy@main@language: Changed back to	
\@gls@escbsdq: unsanitize		using \language	25
\gls@numberpage, \gls@alphpage,		\findrootlanguage: Obsoleted	48
\gls@Alphpage and		3.07 (2013-07-05)	
\gls@romanpage	110	\@gls@link: fixed bug that failed to find	
\@print@glossary: Moved aux write to		entry in list	107
end of document to prevent		\glossarypreamble: modified to work	
unwanted whatsit occurring here. ..	185	with \setglossarypreamble	37
General: Added check for doc package	4	\gls@docclearpage: added check for	
added datatool-base as a required		openright	39
package	4	\glspostdescription: Added	
added local key	104	spacefactor code	9
\gls@Alphpage: new	176	\GlsSetXdyCodePage: Added check for	
\gls@alphpage: new	176	fontspec	48

\SetDescriptionAcronymDisplayStyle: now using \glsdoparenifnotempty	236	\ifglshasdesc: new	53
\setglossarypreamble: new	37	\ifglshassymbol: new	53
3.08a (2013-08-30)		altlongragged4col: updated to use \glossentry and \subglossentry	283
list: updated list style to use \glossentry and \subglossentry	266	alttree: updated to use \glossentry and \subglossentry	310
listdotted: updated listdotted style to use \glossentry and \subglossentry	268	index: added paragraph break at end of environment	305
altlist: updated altlist style to use \glossentry and \subglossentry	267	updated to use \glossentry and \subglossentry	305
inline: updated inline style to use \glossentry and \subglossentry	264	long: updated to use \glossentry and \subglossentry	269
3.08a (2013-09-28)		longragged: updated to use \glossentry and \subglossentry	280
\@glo@storeentry: no longer need to check for special characters in any of the fields other than sort	84	longragged3col: updated to use \glossentry and \subglossentry	282
updated for \glossentry	84	tree: updated to use \glossentry and \subglossentry	306
\@glossaryentryfield: switched to \glossentry	83	\setglossarystyle: new	206
\@glossarysubentryfield: switched to \subglossentry	83	\setglossentrycompatibility: new	203
General: added nogroupskip key to \printglossary	196	superragged: updated to use \glossentry and \subglossentry	298
removed definition of \@glossaryentryfield	353	3.09a (2013-10-09)	
removed definition of \@glossarysubentryfield	353	\@gls@assign@symbolplural@field: new	18
\compatibleglossentry: new	202	\@gls@default@value: new	61
\compatiblesubglossentry: new	203	\Glsentrydesc: made robust	146
\glossaryentryfield: deprecated	204	\Glsentrydescplural: made robust	147
\Glossentrydesc: new	202	\Glsentryfirst: made robust	148
\glossentrydesc: new	202	\Glsentryfirstplural: made robust	148
\Glossentryname: new	202	\Glsentryfull: made robust	151
\glossentryname: new	202	\Glsentryfullpl: made robust	151
\Glossentrysymbol: new	203	\Glsentrylong: made robust	151
\glossentrysymbol: new	203	\Glsentrylongpl: made robust	151
\gls@assign@desc@field: new	18	\Glsentryname: made robust	145
\gls@assign@descplural@field: new	18	\Glsentryplural: made robust	147
\gls@assign@field: new	67	\Glsentryshort: made robust	150
\gls@ifnotmeasuring: new	85	\Glsentryshortpl: made robust	151
\glsaddallunused: new	154	\Glsentrysymbol: made robust	147
\glsexpandfields: new	67	\Glsentrysymbolplural: made robust	148
\glsnoexpandfields: new	67	\Glsentrytext: made robust	147
\glssee: made robust	181	\Glsentryuseri: made robust	149
\glsseeformat: made robust	181	\Glsentryuserii: made robust	149
\glsseeitem: made robust	181	\Glsentryuseriii: made robust	149
\glsseelist: made robust	181	\Glsentryuseriv: made robust	150
\ifglstdescsuppressed: new	53	\Glsentryuserv: made robust	150
		\glstextup: new	212

\ifglshassymbol: changed test to check for \@gls@default@symbol	53
3.10a (2013-09-28)	
\gls@assign@type@field: new	18
3.10a (2013-10-13)	
\@gls@keymap: new	69
\@gls@provide@newglossary: new ...	56
\@gls@writedef: new	68
\@glsdefaultplural: Obsolete	65
\@glsnodesc: new	64
\@print@glossary: Added providecommand code to aux file	185, 186
\gls@defglossaryentry: Changed to using \@gls@default@value	77, 78
new	77
\gls@writedefhook: new	76
\makeglossaries: Added providecommand code to aux file ..	167
\new@glossaryentry: new	68
\ns@newglossary: added \@gls@provide@newglossary	57
3.11a (2013-10-15)	
\@ACRlong: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	353
\@ACRshort: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	351
\@Acrlong: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	352
\@Acrshort: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	351
\@GLS@: add \glslabel, \glsifplural, \glscapscase, \glscustomtext and \glsinsert	120
change to using \glentryfmt style commands	121
removed \MakeUppercase (now moved to \glentryfmt)	121
\@GLSpl: add \glslabel, \glsifplural, \glscapscase, \glscustomtext and \glsinsert	123
change to using \glentryfmt style commands	123
removed \MakeUppercase as now dealt with in \glentryfmt	123
\@Gls@: add \glsifplural, \glscapscase, \glscustomtext and \glsinsert	120
change to using \glentryfmt style commands	120
removed \makefirstuc (now dealt with in \glentryfmt)	120
\@Glspl@: add \glsifplural, \glscapscase, \glscustomtext and \glsinsert	122
change to using \glentryfmt style commands	122
removed \makefirstuc (now dealt with in \glentryfmt)	122
\@acrlong: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	352
\@acrshort: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	351
\@gls@: add \glslabel, \glsifplural, \glscapscase, \glscustomtext and \glsinsert	119
change to using \glentryfmt style commands	119
\@gls@noexpand@fields: Fixed bug expand replaced with noexpand	66
\@glsdisp: add \glslabel, \glsifplural, \glscapscase, \glscustomtext and \glsinsert	124
change to using \glentryfmt style commands	124
\@glspl@: add \glslabel, \glsifplural, \glscapscase, \glscustomtext and \glsinsert	121
change to using \glentryfmt style commands	121
General: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	137–144
changed to just use \Glsentrydescplural	130
changed to just use \glentrydescplural	130
changed to just use \Glsentrydesc .	129
changed to just use \glentrydesc .	129
changed to just use \Glsentryfirstplural	127

changed to just use		<code>\glsdisplay</code> : obsoleted	102
<code>\glsentryfirstplural</code>	127, 128	<code>\glsdisplayfirst</code> : obsoleted	102
changed to just use <code>\Glsentryfirst</code>	126	<code>\glsgenentryfmt</code> : new	97
changed to just use <code>\glsentryfirst</code>	126	<code>\glsgetgrouptitle</code> : Added check in	
changed to just use <code>\Glsentryname</code>	128	case non-Latin alphabet in use	205
changed to just use		<code>\glsglossarymark</code> : replaced	
<code>\glsentryname</code>	128, 129	<code>\MakeUppercase</code> with	
changed to just use <code>\Glsentryplural</code>	127	<code>\mfirstucMakeUppercase</code>	38
changed to just use		<code>\glsnavigation</code> : switched to using	
<code>\glsentryplural</code>	126, 127	<code>\@gls@getgrouptitle</code>	263
changed to just use		<code>\ifglshasdesc</code> : replaced <code>\ifdefempty</code>	
<code>\Glsentrysymbolplural</code>	132	with <code>\ifcsempy</code>	53
changed to just use		<code>\ifglshaslong</code> : new	53
<code>\Glsentrysymbolplural</code>	131, 132	<code>\ifglshasshort</code> : new	54
changed to just use <code>\Glsentrysymbol</code>	131	<code>\ifglshassymbol</code> : replaced	
changed to just use		<code>\ifdefempty</code> with <code>\ifcsempy</code>	53
<code>\glsentrysymbol</code>	130, 131	<code>\ifglused</code> : replaced <code>\ifthenelse</code> with	
Changed to just use <code>\Glsentrytext</code>	125	<code>\ifbool</code>	51
changed to just use <code>\glsentrytext</code>	125	<code>\longnewglossaryentry</code> : new	76
changed to just use		<code>\ns@newglossary</code> : replaced	
<code>\Glsentryuseriii</code>	134	<code>\glsdisplay</code> and	
changed to just use		<code>\glsdisplayfirst</code> with	
<code>\glsentryuseriii</code>	134	<code>\glsentryfmt</code>	57
changed to just use <code>\Glsentryuserii</code>	133	compatible-3.07: <code>cnew</code>	27
changed to just use <code>\glsentryuserii</code>	133	<code>\SetCustomDisplayStyle</code> : updated to	
changed to just use <code>\Glsentryuseriv</code>	135	use <code>\defglsglsentryfmt</code>	245
changed to just use <code>\glsentryuseriv</code>	135	<code>\SetDefaultAcronymDisplayStyle</code> :	
changed to just use <code>\Glsentryuseri</code>	132	changed to use <code>\defglsglsentryfmt</code>	230
changed to just use		<code>\SetDescriptionAcronymDisplayStyle</code> :	
<code>\glsentryuseri</code>	132, 133	updated to use <code>\defglsglsentryfmt</code>	236
changed to just use <code>\Glsentryuservi</code>	136	<code>\SetDescriptionDUAAcronymDisplayStyle</code> :	
changed to just use		updated to use <code>\defglsglsentryfmt</code>	234
<code>\glsentryuservi</code>	136, 137	<code>\SetDescriptionFootnoteAcronymDisplayStyle</code> :	
changed to just use <code>\Glsentryuserv</code>	136	updated to use <code>\defglsglsentryfmt</code>	232
changed to just use		<code>\SetDUADisplayStyle</code> : updated to use	
<code>\glsentryuserv</code>	135, 136	<code>\defglsglsentryfmt</code>	243
Now requires <code>textcase</code>	4	<code>\SetFootnoteAcronymDisplayStyle</code> :	
acronymlists: replaced		updated to use <code>\defglsglsentryfmt</code>	238
<code>\@addtoacronymlists</code> with		<code>\SetSmallAcronymDisplayStyle</code> :	
<code>\DeclareAcronymList</code>	16	updated to use <code>\defglsglsentryfmt</code>	241
<code>\defglsglsdisplay</code> : obsoleted	103	<code>\setupglossaries</code> : new	29
<code>\defglsglsdisplayfirst</code> : obsoleted	103	<code>\showglolong</code> : new	251
<code>\defglsglsentryfmt</code> : new	56	<code>\showgloshort</code> : new	251
<code>\forglsglsentries</code> : replaced <code>\ifx</code> with		numbers: new	28
<code>\ifdefempty</code>	49	symbols: new	27
<code>\gls@assign@desc</code> : new	76		
<code>\gls@defglossaryentry</code> : Fixed default			
counter if none supplied	81	<code>\gls@defglossaryentry</code> : added	
<code>\gls@doentryfmt</code> : new	56	<code>\glslabel</code>	77
		<code>\glsaddkey</code> : new	71

3.13a (2013-11-05)

\@gls@assign@symbol@field: changed to use \glssetnoexpandfield	18
\@gls@assign@symbolplural@field: changed to use \glssetnoexpandfield	18
\@gls@link: removed \relax	108
\@gls@notranslatorhook: new	22
\@gls@setupsort@standard: moved \@gls@santizesort to \glsprestandardsort	11
ucmark: added check for memoir	10
see: added \gls@checkseeallowed	62
\glossarysection: changed \glossarymark to \gls glossarymark	38
\glossarystyle: fixed bug caused by using \ifdef instead of \ifcsdef	207
\gls@assign@desc@field: changed to use \glssetnoexpandfield	18
\gls@assign@descplural@field: changed to use \glssetnoexpandfield	18
\gls@assign@name@field: changed to use \glssetnoexpandfield	18
\gls@assign@type@field: changed to use \glssetexpandfield	18
\gls@checkseeallowed: new	62
\glsaddallunused: set default to \@glo@types	154
\Glsentryfull: changed to use \acrfullformat	151
\glsentryfull: changed to use \acrfullformat	151
\Glsentryfullpl: changed to use \acrfullformat	151
\glsentryfullpl: changed to use \acrfullformat	151
\gls glossarymark: renamed \glossarymark to \gls glossarymark to avoid conflict with memoir	38
\glsprestandardsort: new	10
\glssetexpandfield: new	17
\glssetnoexpandfield: new	18
altsuper4colheader: switched to \tabularnewline	296
altsuper4colheaderborder: switched to \tabularnewline	297

long: switched to \tabularnewline	269, 270
long3col: switched to \tabularnewline	271
long3colheader: switched to \tabularnewline	272
long3colheaderborder: switched to \tabularnewline	272
long4col: switched to \tabularnewline	272
long4colheader: switched to \tabularnewline	273
longheader: switched to \tabularnewline	270
longheaderborder: switched to \tabularnewline	270
\SetFootnoteAcronymDisplayStyle: fixed missing argument bug	238
super: switched to \tabularnewline	291
super3col: switched to \tabularnewline	293
super3colheader: switched to \tabularnewline	294
super4col: switched to \tabularnewline	295
super4colheader: switched to \tabularnewline	295
super4colheaderborder: switched to \tabularnewline	296
superheader: switched to \tabularnewline	292
superheaderborder: switched to \tabularnewline	292

3.14a (2013-11-12)

\@glswritefiles: renamed \glswritefiles to \@glswritefiles and used “savewrites” option to set \glswritefiles	173
General: new	254
acronyms: new	14
\gls@def glossaryentry: added check for existence of default glossary	78
set the default for firstplural to be the value of plural	80
xindy gloss: new	26
\longprovideglossaryentry: new	77
compatible-2.07: added check for 2.07 before setting 3.07 compatibility	27

notranslate: new	22	sm-short-long-desc: new	223
\provideglossaryentry: new	67	index: new	28
4.0 (2013-11-14)		\newacronymstyle: new	218
\gls@defglossaryentry: added check		long-sc-short: new	220
for first key	80	long-sc-short-desc: new	222
super: fixed typo in \subglossentry		long-short: new	218
(\glossentrydesc)	291	long-short-desc: new	221
4.01 (2013-11-16)		long-sm-short: new	221
General: fixed non-value options so that		long-sm-short-desc: new	222
they can be passed to document class .	8	long-sp-short-desc: new	221
\CustomAcronymFields: inserted		footnote: new	225
missing comma	246	footnote-desc: new	227
4.02 (2013-12-05)		footnote-sc: new	227
\@acrfull: now using \acrfullfmt ..	212	footnote-sc-desc: new	228
\@gls@indexdef: new	28	footnote-sm: new	227
\@gls@numbersdef: new	28	footnote-sm-desc: new	228
\@gls@symbolsdef: new	27	\setacronymstyle: new	217
General: Removed \acronymfont .	141–144	\SetDescriptionAcronymDisplayStyle:	
\ACRfullfmt: new	214	Moved check for empty custom text to	
\Acrfullfmt: new	213	prevent unwanted parenthetical	
\acrfullfmt: new	213	material	236
\ACRfullplfmt: new	215	\SetDescriptionFootnoteAcronymDisplayStyle:	
\Acrfullplfmt: new	214	Moved check for empty custom text to	
\acrfullplfmt: new	214	prevent unwanted parenthetical	
\acronymentry: new	217	material	232
sanitize: fixed bug that caused an error		\SetFootnoteAcronymDisplayStyle:	
here	21	Moved check for empty custom text to	
sc-short-long: new	221	prevent unwanted parenthetical	
sc-short-long-desc: new	223	material	238
\Genacrfullformat: new	102	\SetGenericNewAcronym: new	216
\genacrfullformat: new	102	\SetSmallAcronymDisplayStyle:	
\GenericAcronymFields: new	217	Moved check for empty custom text to	
\Genplacrfullformat: new	102	prevent unwanted parenthetical	
\genplacrfullformat: new	102	material	241
\Glsentryfull: bug fix: added missing		dua: new	223
\acronymfont	151	dua-desc: new	225
\glsentryfull: bug fix: added missing		numberedsection: added nameref	
\acronymfont	151	option	7
\Glsentryfullpl: bug fix: added		4.02 (2013-13-05)	
missing \acronymfont	151	\makeglossaries: made preamble only	169
\glsentryfullpl: bug fix: added		4.03 (2014-01-17)	
missing \acronymfont	151	General: changed default to \@empty	
\glsgenacfmt: new	100	instead of \relax	27
\GlsUseAcrEntryDispStyle: new ...	218	4.03 (2014-01-20)	
\GlsUseAcrStyleDefs: new	218	\@do@wrglossary: added	
short-long: new	220	\glsdetoklabel	178
short-long-desc: new	222	\@ACRlong: removed \glslabel	
xindynoglsnumbers: new	26	(defined in \@gls@link)	353
sm-short-long: new	221		

\@ACRshort: removed \glslabel (defined in \@gls@link)	351	\Genplacrfullformat: redefined to use accessibility information	350
\@Acrlong: removed \glslabel (defined in \@gls@link)	352	\genplacrfullformat: redefined to use accessibility information	350
\@Acrshort: removed \glslabel (defined in \@gls@link)	351	\glossentryname: added \glsdetoklabel	202
\@GLS@: removed \glslabel (defined in \@gls@link)	120	\gls@defglossaryentry: added \glsdetoklabel	77
\@GLSpl: removed \glslabel (defined in \@gls@link)	123	replaced #1 with \@gls@label	78
\@Gls@: removed \glslabel (defined in \@gls@link)	120	replaced \ifthenelse with \ifdefequal	79
\@Gls@entry@field: new	145	\glsadd: added \glsdetoklabel	153
\@Glspl@: removed \glslabel (defined in \@gls@link)	122	\glsaddkey: switched to using \@gls@field@link	72
\@acrlong: removed \glslabel (defined in \@gls@link)	352	\glsdetoklabel: new	50
\@acrshort: removed \glslabel (defined in \@gls@link)	351	\glsdisplaynumberlist: added \glsdetoklabel	152
\@gls@: removed \glslabel (defined in \@gls@link)	119	\glsdoifexistsorwarn: new	51
\@gls@access@display: new	339	\glsentryaccess: switched to using \@gls@entry@field	337
\@gls@entry@field: new	144	\glsentrydescaccess: switched to using \@gls@entry@field	338
\@gls@fetchfield: new	69	\glsentrydescpluralaccess: switched to using \@gls@entry@field	338
\@gls@field@link: new	124	\glsentryfirstaccess: switched to using \@gls@entry@field	338
\@gls@link: added \glsdetoklabel .	107	\glsentryfirstplural: added \glsdetoklabel	148
moved \@gls@link@opts and \@gls@link@label to \@gls@link	107	\glsentrylongaccess: switched to using \@gls@entry@field	339
\@gls@writedef: added \glsdetoklabel	68	\glsentrylongpluralaccess: switched to using \@gls@entry@field	339
\@glsdisp: removed \glslabel (defined in \@gls@link)	124	\glsentrypluralaccess: switched to using \@gls@entry@field	338
\@Glspl@: removed \glslabel (defined in \@gls@link)	121	\glsentryshortaccess: switched to using \@gls@entry@field	338
\@printglossary: added \glsdetoklabel	184	\glsentryshortpluralaccess: switched to using \@gls@entry@field	338
General: removed \glslabel (defined in \@gls@link)	137	\glsentrysymbolaccess: switched to using \@gls@entry@field	338
sc-short-long-desc: redefined to use accessibility information	357	\glsentrysymbolpluralaccess: switched to using \@gls@entry@field	338
\compatibleglossentry: added \glsdetoklabel	333	\glsentrytextaccess: switched to using \@gls@entry@field	337
\compatiblesubglossentry: added \glsdetoklabel	334	\glsngenacfmt: redefined to use accessibility information	348
\Genacrfullformat: redefined to use accessibility information	350		
\genacrfullformat: redefined to use accessibility information	350		

\glsgenentryfmt: redefined to use accessibility information	345	sm-short-long-desc: redefined to use accessibility information	357
\glshyperlink: added \glsdetoklabel	153	long-sc-short-desc: redefined to use accessibility information	356
\glslocalreset: added \glsdetoklabel	85	long-short: redefined to use accessibility information	354
\glslocalunset: added \glsdetoklabel	86	long-short-desc: redefined to use accessibility information	356
\glsmoveentry: added \glsdetoklabel	83	long-sm-short-desc: redefined to use accessibility information	356
replaced \ifthenelse with \ifdefequal	83	footnote: redefined to use accessibility information	360
\glsrefentry: added \glsdetoklabel	200	footnote-desc: redefined to use accessibility information	362
\glsreset: added \glsdetoklabel ...	85	footnote-sc: redefined to use accessibility information	362
\glsseelist: added \expandafter commands	181	footnote-sc-desc: redefined to use accessibility information	363
\glsstepentry: added \glsdetoklabel	200	footnote-sm: redefined to use accessibility information	362
\glsstepsubentry: added \glsdetoklabel	200	footnote-sm-desc: redefined to use accessibility information	363
\glsunset: added \glsdetoklabel ...	86	\renewacronymstyle: new	218
short-long: commented spurious EOL	220	\showglocounter: added \glsdetoklabel	249
redefined to use accessibility information	355	\showglodesc: added \glsdetoklabel	250
short-long-desc: redefined to use accessibility information	357	\showglodescaccess: added \glsdetoklabel	369
\ifglshdescsuppressed: added \glsdetoklabel	53	\showglodescplural: added \glsdetoklabel	250
fixed typo	53	\showglodescpluralaccess: added \glsdetoklabel	369
\ifglshentryexists: added \glsdetoklabel	51	\showglofirst: added \glsdetoklabel	248
\ifglshaschildren: added \glsdetoklabel	52	\showglofirstaccess: added \glsdetoklabel	369
\ifglshasdesc: added \glsdetoklabel	53	\showglofirstpl: added \glsdetoklabel	248
\ifglshasfield: new	54	\showglofirstpluralaccess: added \glsdetoklabel	369
\ifglshaslong: added \glsdetoklabel	53	\showgloflag: added \glsdetoklabel	251
\ifglshasparent: added \glsdetoklabel	53	\showgloindex: added \glsdetoklabel	251
\ifglshasshort: added \glsdetoklabel	54	\showglolevel: added \glsdetoklabel	248
\ifglshassymbol: added \glsdetoklabel	53	\showglolong: added \glsdetoklabel	251
replaced \ifcempty with \ifdefempty and replaced \ifx with \ifdefequal	53	\showglolongaccess: added \glsdetoklabel	370
\ifglshused: added \glsdetoklabel ..	51		

\showglolongpluralaccess: added		redefined to use accessibility	
\glsdetoklabel	370	information	360
\showglongname: added \glsdetoklabel	250	4.04 (2014-03-04)	
\showglongnameaccess: added		\@gls@getcounterprefix: added	
\glsdetoklabel	369	warning if no prefix can be formed .	179
\showgloparent: added		4.04 (2014-03-06)	
\glsdetoklabel	247	\@gls@noidx@nosanitizesort: new .	19
\showgloplural: added		\@gls@noidx@sanitizesort: new ...	19
\glsdetoklabel	248	\@gls@nosanitizesort: new	19
\showglopluralaccess: added		\@gls@sanitizesort: new	19
\glsdetoklabel	369	\@glo@addchildren: new	187
\showgloshort: added		\@glo@do@sortentries: new	187
\glsdetoklabel	251	\@glo@grabfirst: new	192
\showgloshortaccess: added		\@glo@sortedinsert: new	188
\glsdetoklabel	370	\@glo@sortentries: new	186
\showgloshortpluralaccess: added		\@glo@sorthandler@case: new	189
\glsdetoklabel	370	\@glo@sorthandler@letter: new ...	188
\showglosort: added \glsdetoklabel	250	\@glo@sorthandler@nocase: new ...	189
\showglosymbol: added		\@glo@sorthandler@word: new	188
\glsdetoklabel	250	\@glo@sortmacro@case: new	190
\showglosymbolaccess: added		\@glo@sortmacro@def: new	191
\glsdetoklabel	369	\@glo@sortmacro@def@do: new	191
\showglosymbolplural: added		\@glo@sortmacro@letter: new	189
\glsdetoklabel	251	\@glo@sortmacro@nocase: new	190
\showglosymbolpluralaccess: added		\@glo@sortmacro@standard: new ...	190
\glsdetoklabel	369	\@glo@sortmacro@use: new	191
\showglotext: added \glsdetoklabel	248	\@glo@sortmacro@word: new	189
\showglotextaccess: added		\@gls@getothergrouptitle: new ...	205
\glsdetoklabel	369	\@gls@noidx@do: new	193
\showglotype: added \glsdetoklabel	248	\@gls@noref@warn: new	173
\showglouseri: added		\@gls@reference: new	195
\glsdetoklabel	249	\@gls@warnonglossdefined: new	17
\showglouserii: added		\@gls@warnontheGLOSSdefined: new .	17
\glsdetoklabel	249	\@no@makeglossaries: new	172
\showglouseriii: added		\@print@glossary: new	185
\glsdetoklabel	249	\@print@noidx@glossary: new	191
\showglouseriv: added		\@print@gloss@setsort: new	183
\glsdetoklabel	249	\@print@glossary: new	183
\showglouserv: added		General: added sort key to printgloss	
\glsdetoklabel	249	group	198
\showglouservi: added		\compatibleglossentry: changed	
\glsdetoklabel	250	\newcommand to \def as is may or	
dua: fixed bug in \acrfullfmt	224	may not be defined	333
fixed bug in \Acrfullplfmt	225	\compatiblesubglossentry: changed	
fixed bug in \acrfullplfmt	225	\newcommand to \def as is may or	
redefined to use accessibility		may not be defined	334
information	358	\defglsdisplayfirst: fixed unwanted	
dua-desc: commented spurious EOL ..	225	space	103
		\glo@grabfirst: new	192

\gls@defglossaryentry: replaced \ifx with \ifdefvoid	82	4.08 (2014-07-30)	\@ACRlong: added \do@gls@link@checkfirsthyper	352
\glsnoidxdisplayloc: new	195		\@ACRshort: added \do@gls@link@checkfirsthyper	351
\glsnoidxdisplaylocclishandler: new	194		\@Acrlong: added \do@gls@link@checkfirsthyper	352
\glsnoidxlocclishandler: new	194		\@Acrshort: added \do@gls@link@checkfirsthyper	351
\glsnoidxstripaccents: new	19		\@GLS@: moved \glsifhyper	121
alttree: moved hangindent and parindent assignments outside level test	310		moved check for first use to \@gls@link	121
\makeglossaries: Moved definition of \glswrite to \makeglossaries ..	167		\@GLSpl: moved \glsifhyper	123
\makenoidxglossaries: new	169		moved check for first use to \@gls@link	123
\printglossary: changed to use new \@printglossary	182		\@GLS@: moved \glsifhyper	120
\printnoidxglossaries: new	183		moved check for first use to \@gls@link	120
\printnoidxglossary: new	183		\@GLspl@: moved \glsifhyper	122
\showgloclolist: new	251		moved check for first use to \@gls@link	122
\warn@noprintglossary: Activate warning in \makeglossaries	182		\@acrlong: added \do@gls@link@checkfirsthyper	352
\writeist: checked for definition of \glswrite	156, 160		\@acrshort: added \do@gls@link@checkfirsthyper	350
4.06 (2014-03-12)			\@closegls: new	166
\@GLS@: added \glsifhyper	121		\@gls@: moved \glsifhyper	119
\@GLSpl: added \glsifhyper	123		moved check for first use to \@gls@link	119
\@GLS@: added \glsifhyper	120		\@gls@doautomake: new	26
\@GLspl@: added \glsifhyper	122		\@gls@field@link: added assignment of \do@gls@link@checkfirsthyper	124
\@gls@: added \glsifhyper	119		\@gls@forbidtexext: new	56
\@gls@numbersdef: added hook to set toc title	28		\@gls@hyp@opt: new	105
\@gls@symbolsdef: added hook to set toc title	27		\@gls@link: removed redundancy	107
\@glsdisp: added \glsifhyper	124		renamed \gls@type to \glstype ...	107
\@glspl@: added \glsifhyper	121		\@glsdisp: moved \glsifhyper	124
General: added \glsifhyper	137–144		moved check for first use to \@gls@link	124
acronym: added hook to set toc title	14		\@GLspl@: moved \glsifhyper	121
acronyms: added hook to set toc title ...	14		moved check for first use to \@gls@link	121
\glsdefmain: added hook to set toc title	13		\@ignored@glossaries: new	59
4.07 (2014-04-04)			General: added entrycounter option to printgloss family	196
\@glossarysection: added optional argument when using unstarred version	39		added nopostdot option to printgloss family	196
\@gls@noidx@do: added \global in case it's used in a tabular-like style	193			
\Acrfullplfmt: fixed no case change bug	214			
\glsletentryfield: new	145			

added subentrycounter option to printgloss family	197	removed \@sGlsuseriii	134
explicitly initialise hyper key	104	removed \@sglsuseriii	134
moved \glsifhyper	137–144	removed \@sGLSuserii	133
removed \@sACRlongpl	144	removed \@sGlsuserii	133
removed \@sAcrlongpl	143	removed \@sglsuserii	133
removed \@sacrlongpl	142	removed \@sGLSuseriv	135
removed \@sACRlong	142	removed \@sGlsuseriv	135
removed \@sAcrlong	141	removed \@sglsuseriv	134
removed \@sacrlong	140	removed \@sGLSuseri	133
removed \@sACRshortpl	140	removed \@sGlsuseri	132
removed \@sAcrshortpl	139	removed \@sglsuseri	132
removed \@sacrshortpl	139	removed \@sGLSuservi	137
removed \@sACRshort	138	removed \@sGlsuservi	136
removed \@sAcrshort	137	removed \@sglsuservi	136
removed \@sacrshort	137	removed \@sGLSuserv	136
removed \@sgls@link	106	removed \@sGlsuserv	136
removed \@sGLSdescplural	130	removed \@sglsuserv	135
removed \@sGlsdescplural	130	removed \@sGLS	120
removed \@sglsdescplural	130	removed \@sGls	119
removed \@sGLSdesc	129	removed \@sgls	119
removed \@sGlsdesc	129	removed \@thirdofthree (defined in kernel)	118
removed \@sglsdesc	129	removed sPGLS	259
removed \@sglsdisp	124	removed sPglS	257
removed \@sGLSfirstplural	128	removed spglS	256
removed \@sGlsfirstplural	127	removed sPGLSpl	259
removed \@sglsfirstplural	127	removed sPglSpl	258
removed \@sGLSfirst	126	removed spglSpl	257
removed \@sGlsfirst	126	\ACRfull: removed \s@ACRfull	213
removed \@sglsfirst	125	switched to using \@gls@hyp@opt ..	213
removed \@sGLSname	128	\Acrfull: removed \@sAcrfull	213
removed \@sGlsname	128	switched to using \@gls@hyp@opt ..	213
removed \@sglsname	128	\acrfull: removed \@sacrfull	212
removed \@sGLSplural	127	switched to using \@gls@hyp@opt ..	212
removed \@sGlsplural	127	\ACRfullpl: removed \s@ACRfullpl ..	215
removed \@sglsplural	126	switched to using \@gls@hyp@opt ..	215
removed \@sGLSpl	123	\Acrfullpl: removed \s@Acrfullpl ..	214
removed \@sGlspl	122	switched to using \@gls@hyp@opt ..	214
removed \@sglspl	121	\acrfullpl: removed \s@acrfullpl ..	214
removed \@sGLSsymbolplural	132	switched to using \@gls@hyp@opt ..	214
removed \@sGlsymbolplural	131	\ACRlong: switched to using \@gls@hyp@opt	142
removed \@sglsymbolplural	131	\Acrlong: switched to using \@gls@hyp@opt	141
removed \@sGLSsymbol	131	\acrlong: switched to using \@gls@hyp@opt	140
removed \@sGlsymbol	131	\ACRlongpl: switched to using \@gls@hyp@opt	144
removed \@sglsymbol	130		
removed \@sGLStext	125		
removed \@sGlstext	125		
removed \@sglstext	125		
removed \@sGLSuseriii	134		

\Acrlongpl: switched to using \@gls@hyp@opt	143	definition	118
\acrlongpl: switched to using \@gls@hyp@opt	142	\GLSfirst: switched to using \@gls@hyp@opt	126
\ACRshort: switched to using \@gls@hyp@opt	138	\Glsfirst: switched to using \@gls@hyp@opt	126
\acrshort: switched to using \@gls@hyp@opt	137	\glsfirst: switched to using \@gls@hyp@opt	125
\acrshort: switched to using \@gls@hyp@opt	137	\GLSfirstplural: switched to using \@gls@hyp@opt	128
\ACRshorttpl: switched to using \@gls@hyp@opt	140	\Glsfirstplural: switched to using \@gls@hyp@opt	127
\acrshorttpl: switched to using \@gls@hyp@opt	139	\glsfirstplural: switched to using \@gls@hyp@opt	127
\acrshorttpl: switched to using \@gls@hyp@opt	139	\glsifhyper: deprecated	105
\forallacronyms: new	49	\glslink: switched to using \@gls@hyp@opt	106
\GLS: switched to using \@gls@hyp@opt	120	\glslinkcheckfirsthyperhook: new	107
\Gls: switched to using \@gls@hyp@opt	119	\glslinkvar: new	105
\gls: switched to using \@gls@hyp@opt	118	\GLSname: switched to using \@gls@hyp@opt	128
\gls@defglossaryentry: added check for ignored glossary	79	\Glsname: switched to using \@gls@hyp@opt	128
\gls@istfilebase: new	35	\glsname: switched to using \@gls@hyp@opt	128
\glsaddkey: removed \@sGLS@user@<key>	73	\GLSpl: switched to using \@gls@hyp@opt	123
removed \@sGls@user@<key>	72	\Glspl: switched to using \@gls@hyp@opt	122
removed \@sgls@user@<key>	72	\glspl: switched to using \@gls@hyp@opt	121
switched to using \@gls@hyp@opt	72, 73	\GLSplural: switched to using \@gls@hyp@opt	127
\GLSdesc: switched to using \@gls@hyp@opt	129	\Glsplural: switched to using \@gls@hyp@opt	126
\Glsdesc: switched to using \@gls@hyp@opt	129	\glsplural: switched to using \@gls@hyp@opt	126
\glsdesc: switched to using \@gls@hyp@opt	129	\glsspace: new	213
\GLSdescplural: switched to using \@gls@hyp@opt	130	\GLSsymbol: switched to using \@gls@hyp@opt	131
\Glsdescplural: switched to using \@gls@hyp@opt	130	\Glsymbol: switched to using \@gls@hyp@opt	131
\glsdescplural: switched to using \@gls@hyp@opt	129	\glssymbol: switched to using \@gls@hyp@opt	130
\glsdisablehyper: added \KV@glslink@hyperfalse to definition	118	\GLSsymbolplural: switched to using \@gls@hyp@opt	132
\glsdisp: switched to using \@gls@hyp@opt	123	\Glsymbolplural: switched to using \@gls@hyp@opt	131
\glsdohyperlink: new	117	\glssymbolplural: switched to using \@gls@hyp@opt	131
\glsdohypertarget: new	117		
\glsenablehyper: added \KV@glslink@hypertrue to			

\GLStext: switched to using \@gls@hyp@opt	125	\ns@newglossary: added \@gls@hyp@opt: new	57
\Glstext: switched to using \@gls@hyp@opt	125	\p@gls@hyp@opt: new	105
\glstext: switched to using \@gls@hyp@opt	125	\PGLS: changed to use \@gls@hyp@opt	259
\glstreenamefmt: new	303	\Pgls: changed to use \@gls@hyp@opt	257
\GLSuseri: switched to using \@gls@hyp@opt	132	\pgls: changed to use \@gls@hyp@opt	256
\Glsuseri: switched to using \@gls@hyp@opt	132	\PGLSpl: changed to use \@gls@hyp@opt	259
\glsuseri: switched to using \@gls@hyp@opt	132	\Pglspl: changed to use \@gls@hyp@opt	258
\GLSuserii: switched to using \@gls@hyp@opt	133	\pglspl: changed to use \@gls@hyp@opt	257
\Glsuserii: switched to using \@gls@hyp@opt	133	\s@gls@hyp@opt: new	105
\glsuserii: switched to using \@gls@hyp@opt	133	\s@newglossary: new	57
\GLSuseriii: switched to using \@gls@hyp@opt	134	automake: new	26
\Glsuseriii: switched to using \@gls@hyp@opt	134	4.09 (2014-08-12)	
\glsuseriii: switched to using \@gls@hyp@opt	134	\glsaddkey: fixed bug in user commands	72
\GLSuseriv: switched to using \@gls@hyp@opt	135	4.10 (2014-08-27)	
\Glsuseriv: switched to using \@gls@hyp@opt	135	\@Gls@acrentryname: new	146
\glsuseriv: switched to using \@gls@hyp@opt	134	\@Gls@entryname: new	145
\GLSuserv: switched to using \@gls@hyp@opt	136	\@gls@glossary: Renamed \@glossary to \@gls@glossary	175
\Glsuserv: switched to using \@gls@hyp@opt	135	\glspercentchar: new	155
\glsuserv: switched to using \@gls@hyp@opt	135	\glstildechar: new	155
\GLSuservi: switched to using \@gls@hyp@opt	137	alttree: moved space after symbol	310, 311
\Glsuservi: switched to using \@gls@hyp@opt	136	4.11 (2014-09-01)	
\ifignoredglossary: new	59	\@do@wrglossary: added hook	178
altlongragged4col: fixed bug that displayed description instead of symbol	283	sanitize: none option	21
\newglossary: added starred version ..	57	\gls@wrglossary: renamed from \@wrglossary to \gls@wrglossary	175
\newignoredglossary: new	59	\glsaddprotectedpagefmt: new	176
		\glsbackslash: new	155
		4.12 (2014-11-22)	
		\@gls@addpredefinedattributes: Added glsignore attribute	43
		\@gls@adjustmode: new	154
		\@gls@notranslatorhook: removed ...	22
		\@gls@toc: added \protect to \numberline	40
		\@gls@usetranslator: new	22
		\glsacrpluralsuffix: new	31
		\glsadd: added check for vertical mode	153
		\glsaddallunused: replaced @gobble with glsignore	154
		\glsifusedtranslator: new	22
		\glsignore: new	154
		\glsupacrpluralsuffix: new	31
		\ProvidesGlossariesLang: new	32

\RequireGlossariesLang: new	32	4.16 (2015-07-08)	
4.13 (2015-02-03)			\@ACRlong: added \glspostlinkhook 353
\indexspace: new	265, 285, 303		\@ACRshort: added \glspostlinkhook 352
4.14 (2015-02-28)			\@Acrlong: added \glspostlinkhook 352
\@glslocalreset: new	87		\@Acrshort: added \glspostlinkhook 351
\@glslocalunset: new	86		\@GLS@: added \glspostlinkhook ... 121
\@glsreset: new	87		\@GLSpl: added \glspostlinkhook .. 123
\@glsunset: new	86		\@Gls@: added \glspostlinkhook ... 120
\@newglossaryentry@defcounters:			\@GLspl@: added \glspostlinkhook . 123
new	88		\@acrlong: added \glspostlinkhook 352
\cGls: new	91		\@acrshort: added \glspostlinkhook 351
\cGls@: new	91		\@gls@: added \glspostlinkhook ... 119
\cGLspl@: new	92		\@gls@link: added
\cgls: new	91		\glspostlinkhook 106
\cgls@: new	91		\@gls@field@link: added
\cglspl: new	92		\glspostlinkhook 124
\cglspl@: new	92		\@gls@link: moved definition of
\@gls@entry@count: new	91		\glsifhyperon outside of this
\@gls@increment@currcount: new	90		macro 108
\@gls@local@increment@currcount:			\@glsdisp: added \glspostlinkhook 124
new	90		\@GLspl@: added \glspostlinkhook . 122
\@gls@write@entrycounts: new	90		General: added \glspostlinkhook 137–144
\@glslocalreset: new	87		\glsacspace: new 220
\@glslocalunset: new	86		\glsadd: changed \@do@wrglossary to
\@glsreset: new	87		\@do@wrglossary 154
\@glsunset: new	86		\glsfielddef: new 74
\@newglossaryentry@defcounters:			\glsfielddedef: new 74
new	83		\glsfieldfetch: new 75
\cGls: new	91		\glsfieldgdef: new 74
\cgls: new	91		\glsfieldxdef: new 73
\cGlsformat: new	92		\glsifhyperon: moved definition of
\cglsformat: new	91		\glsifhyperon 107
\cGLspl: new	92		\glslinkpostsetkeys: new 107
\cglspl: new	92		\glspostlinkhook: new 106
\cGLsplformat: new	93		\glswriteentry: new 176
\cglsplformat: new	92		\ifglsfieldcseq: new 76
\@gls@defdocnewglossaryentry: new	67		\ifglsfielddefeq: new 75
\@glsenableentrycount: new	88		\ifglsfieldefeq: new 75
\@glslocalreset: switched to			long-sp-short: new 219
\@glslocalreset	85		\showglofield: new 252
\@glslocalunset: switched to			
\@glslocalunset	86	4.18 (2015-09-09)	
\@glsreset: switched to \@glsreset	85		General: split mfirstuc into separate
\@glsunset: switched to \@glsunset	86		bundle 4
4.15 (2015-03-16)		4.19 (2015-10-31)	
General: bug fix replaced \@glo@type			\glstreenamibox: new 309
with \glstype	144	4.19 (2015-11-22)	
4.16 (2015-06-18)			\@gls@link@nocheckfirsthyper: new 124
\glsaddstoragekey: new	70		\@gls@preglossaryhook: new 183

\@printglossary: added		\glslistgroupheaderfmt: new	266
\@gls@preglossaryhook	184	\glslistnavigationitem: new	266
\do@glsglisablehyperinlist: new	107	\glstreegroupheaderfmt: new	303
\doifglossarynoexistsordo: new	52	\glstreenavigationfmt: new	304
\gls@gobbleopt: new	56	\ifglswrallowprimitivemods: new	177
\glsdoifexistsordo: new	52	list: fixed missing space before	
4.20 (2015-11-30)		description	266
\@gls@link: added		long: fixed typo in \glossentrydesc	270
\@gls@setdefault@glslink@opts	107	super4col: fixed bug in \glossentry	295
added \glsdonohyperlink when		4.23 (2016-04-30)	
hyperlink is suppressed	108	\glscurrentfieldvalue: new	55
\@gls@setdefault@glslink@opts:		\ifglshasfield: added	
new	107	\glscurrentfieldvalue	54, 55
\gls@checkseeallowed@preambleonly:		altlongragged4col: check for	
new	62	nogroupskip changed	284
\glsdonohyperlink: new	117	altsuperragged4col: check for	
4.21 (2016-01-24)		nogroupskip changed	302
\@printglossary: warn if no style has		long: check for nogroupskip changed	270
been set	183	long-booktabs: check for nogroupskip	
General: changed checkfirsthyper		changed	275
assignment	137–144	long3col: check for nogroupskip	
\glossarystyle: set default style if not		changed	271
already set	207	long3col-booktabs: check for	
\glsLTpenaltycheck: new	278	nogroupskip changed	276
\glspatchLToutput: new	279	long4col: check for nogroupskip	
\glspenaltygroupskip: new	278	changed	273
altlong4col-booktabs: new	277	long4col-booktabs: check for	
altlongragged4col-booktabs: new	278	nogroupskip changed	277
long-booktabs: new	275	longragged: check for nogroupskip	
long3col-booktabs: new	276	changed	280
long4col-booktabs: new	276	longragged3col: check for nogroupskip	
longragged-booktabs: new	277	changed	282
longragged3col-booktabs: new	278	super: check for nogroupskip changed	292
\setglossarystyle: set default style if		super3col: check for nogroupskip	
not already set	206	changed	293
4.22 (2016-04-19)		super4col: check for nogroupskip	
\@do@wrglossary: added check for		changed	295
\@arabic	177	superragged: check for nogroupskip	
added test to allow temporary primitive		changed	298
modifications and added arabic case	177	superragged3col: check for	
mcolalttreespannav: new	290	nogroupskip changed	300
mcolindexspannav: new	286	4.24 (2016-05-27)	
mcoltreenonamespannav: new	289	\@gls@extramakeindexopts: new	164
mcoltreespannav: new	287	\@gls@glossary: added check for debug	
\gls@arabicpage: new	176	mode	175
\gls@protected@pagefmts: added		\@gls@see@noindex: new	5
arabic to list	176	debug: new	5
\glsentrytitlecase: new	148	seenoinindex: new	6
\glsfindwidesttoplevelname: new	309	\glsnomakeindexwarning: new	40

\GlsSetQuote:new	162	mcolindex:replaced \@idxitem with	
\GlsSetWriteIstHook:new	162	\glstreeitem	286
4.25 (2016-06-09)		mcolindexspannav:replaced \@idxitem	
\@gls@enablesavenonumberlist:new	63	with \glstreeitem	286
\@gls@initnonumberlist:new	63	\glstreechildpredesc:new	304
\@gls@savenonumberlist:new	63	\glstreeitem:new	304
4.26 (2016-10-12)		\glstreepredesc:new	304
\@glossary@default@style:added		\glstreesubitem:new	304
check for classicthesis	7	\glstreesubsubitem:new	304

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	
\!	114
\"	19, 111–114, 116
\#	158
\%	155, 160, 161, 317, 318
\&	31, 153
\'	19
\.	9, 20
\=	19
\?	111, 113, 163
\@@delimN	209
\@@do@wrglossary	170, 178
\@@do@wrglossary	154, 175
\@@glo@assign@sortkey	170
\@@glo@list	50
\@@glo@sort	19
\@@glo@type	183
\@@glossarysec	6, 39
\@@glossaryseclabel	7, 39, 196
\@@glossarysecstar	7, 39, 196
\@@gls@checkactual	115, 116
\@@gls@checkbar	114, 115
\@@gls@checkescactual	113
\@@gls@checkescbar	113, 114
\@@gls@checkesclevel	114
\@@gls@checkescquote	112, 164
\@@gls@checklevel	115
\@@gls@checkquote	112, 162, 163
\@@gls@default@entryfmt	94, 103
\@@gls@expand@field	18, 66, 70, 71, 231, 233, 235, 237, 239, 242, 244, 364–368
\@@gls@extramakeindexopts	162, 167
\@@gls@fixbraces	180
\@@gls@noexpand@field	18, 65, 66
\@@gls@noidx@no@sanitizesort	19
\@@gls@noidx@nosanitizesort	172
\@@gls@nosanitizesort	18, 172
\@@gls@sanitizesort	18, 172
\@@gls@xdycheckbackslash	116, 117
\@@gls@xdycheckquote	116
\@@gls@localreset	87, 89
\@@gls@localunset	86, 89
\@@gls@reset	87, 89
\@@gls@unset	86, 89
\@@newglossaryentry@defcounters	88
\@@this@glo@	50
\@ACRfull	213
\@ACRfullpl	215
\@ACRlong	142, 214
\@ACRlongpl	144, 215
\@ACRshort	138, 214
\@ACRshortpl	140, 215
\@Acrfull	213
\@Acrfullpl	214
\@Acrlong	141, 213
\@Acrlongpl	143, 214
\@Acrshort	137, 138
\@Acrshortpl	139
\@Alph	176–178
\@GLS	120
\@GLS@	120, 259
\@GLSdesc	129
\@GLSdesc@	129
\@GLSdescplural	130
\@GLSdescplural@	130
\@GLSfirst	126
\@GLSfirst@	126
\@GLSfirstplural	128
\@GLSfirstplural@	128
\@GLSname	128
\@GLSname@	128, 129
\@GLSpl	123
\@GLSpl@	123, 260
\@GLSplural	127

\@GLSplural@	127	\@Glsuseriii@	134
\@GLSsymbol	131	\@Glsuseriv	135
\@GLSsymbol@	131	\@Glsuseriv@	135
\@GLSsymbolplural	132	\@Glsuserv	135, 136
\@GLSsymbolplural@	132	\@Glsuserv@	136
\@GLStext	125	\@Glsuservi	136
\@GLStext@	125	\@Glsuservi@	136
\@GLSuseri	132, 133	\@Mi	279
\@GLSuseri@	133	\@PGLS	259
\@GLSuserii	133	\@PGLS@	259
\@GLSuserii@	133	\@PGLSpl	259
\@GLSuseriii	134	\@PGLSpl@	259
\@GLSuseriii@	134	\@Pgls	257
\@GLSuseriv	135	\@Pgls@	257
\@GLSuseriv@	135	\@Pglspl	258
\@GLSuserv	136	\@Pglspl@	258
\@GLSuserv@	136	\@Roman	176–178
\@GLSuservi	137	\@acrfull	212
\@GLSuservi@	137	\@acrfullpl	214
\@Gls	119	\@acrlong	140, 141, 213
\@Gls@	89, 91, 119, 258	\@acrlongpl	142, 143, 214
\@Gls@acrentryname	216	\@acrshort	137, 213
\@Gls@entry@field	72, 146–151	\@acrshortpl	139, 214
\@Gls@entryname	145, 216	\@addtoacronymlists	15
\@Glsdesc	129	\@after	15
\@Glsdesc@	129	\@afterheading	267, 321
\@Glsdescplural	130	\@alph	176–178
\@Glsdescplural@	130	\@arabic	176–178
\@Glsfirst	126	\@auxout	56,
\@Glsfirst@	126		57, 90, 167, 170, 173, 182, 185, 186, 261
\@Glsfirstplural	127	\@backslashchar	110, 116, 117
\@Glsfirstplural@	127	\@before	15
\@Glsname	128	\@bsphack	175
\@Glsname@	128	\@cGls	91
\@Glspl	122	\@cGls@	89, 91
\@Glspl@	90, 92, 122, 258, 259	\@cGlspl	92
\@Glsplural	126, 127	\@cGlspl@	90, 92
\@Glsplural@	127	\@cclv	279
\@Glssymbol	131	\@cgls	91
\@Glssymbol@	131	\@cgls@	89, 91
\@Glssymbolplural	131	\@cglspl	92
\@Glssymbolplural@	131, 132	\@cglspl@	89, 92
\@Glstext	125	\@chapter	30
\@Glstext@	125	\@classoptionslist	28
\@Glsuseri	132	\@colht	279
\@Glsuseri@	132	\@colroom	279
\@Glsuserii	133	\@currentlabelname	7, 196
\@Glsuserii@	133	\@curroptions	28
\@Glsuseriii	134	\@declaredoptions	28

\@delimN	209	\@glo@desc	60, 76, 77, 79, 81
\@delimR	208	\@glo@descaccess	335–337
\@disable@onlypremakeg	168	\@glo@descplural	60, 76, 77
\@disable@premakecs	30	\@glo@descpluralaccess	335–337
\@disabled@gl saddxdycounters	43	\@glo@do@sortentries	186
\@do@addcounter	41	\@glo@entry	154
\@do@auxoutstuff	185, 186	\@glo@entryprefix	254
\@do@glossentry	202, 333	\@glo@entryprefixfirst	254
\@do@gl s@getcounterprefix	178	\@glo@entryprefixfirstplural ..	254, 255
\@do@gl s@islistofacronyms	15	\@glo@entryprefixplural	254
\@do@gl ssee	82	\@glo@esclabel	84, 85
\@do@ifinlist	41	\@glo@etext	94–96
\@do@newglossaryentry		\@glo@first	61, 77, 80, 81, 242, 368
	216, 230–237, 239, 241–244, 246, 364–368	\@glo@firstaccess	334, 336, 337
\@do@seeglossary	170, 181	\@glo@firstplural	61, 77, 80, 81, 368
\@do@subglossentry	203, 334	\@glo@firstpluralaccess	335–337
\@do@wrglossary	108	\@glo@grabfirst	192
\@do@writeaux@info	182	\@glo@label	64,
\@ehc	279		71–82, 88, 152, 153, 254, 255, 309, 336, 337
\@empty ...	12, 13, 15, 27, 28, 30, 41, 42, 46,	\@glo@list	82
	48, 49, 79, 84, 109, 119–123, 137–144,	\@glo@long	53, 64, 78, 81
	156, 159, 161, 166, 167, 174, 175, 179,	\@glo@longaccess	335–337
	180, 197, 199, 206, 231, 233, 235–239,	\@glo@longpl	64,
	241, 242, 244, 246, 315, 317, 319, 351–353		78, 81, 230, 233, 234, 237, 239, 242–244, 364
\@end@fixbraces	180	\@glo@longpluralaccess	335–337
\@endfortrue	24, 52, 70, 261	\@glo@name	11, 60, 65, 77, 80, 81
\@esphack	175	\@glo@no@assign@sortkey	169
\@expandtwoargs	28	\@glo@nonumberlist	63
\@firstofone	19, 20	\@glo@numfmt	179, 315
\@firstofthree	105, 118,	\@glo@parent ..	12, 62, 78–80, 84, 85, 187, 188
	119, 121, 124, 137, 139, 141, 143, 351–353	\@glo@plural	61, 77, 80, 366
\@firstoftwo	22,	\@glo@pluralaccess	334, 336, 337
	23, 69, 70, 105, 121–123, 139, 140, 143, 144	\@glo@prefix	
\@for	23, 28, 30,		8, 62, 78, 84, 85, 110, 178, 179, 314, 315
	41, 43, 49, 50, 68, 70, 110, 156–158, 168,	\@glo@range	178, 179, 314, 315
	169, 176, 181, 186, 187, 217, 231, 234,	\@glo@see	62, 78, 82
	235, 238, 240, 243, 245, 247, 261, 262, 315	\@glo@seeautonumberlist	8, 62
\@glo@@desc	81	\@glo@short	54, 64, 78, 81, 367
\@glo@@symbol	82	\@glo@shortaccess	335–337, 364–367
\@glo@access	334, 336, 337, 339	\@glo@shortpl	64, 78, 81,
\@glo@addchildren	187, 191		230, 232–234, 237, 239, 242, 244, 364, 367
\@glo@assign@sortkey	169, 170, 198	\@glo@shortpluralaccess	335–337
\@glo@check@mkidxrangechar		\@glo@sort	11, 19, 60, 78, 80, 84, 85
	109, 110, 178, 314, 315	\@glo@sortedinsert	187, 188
\@glo@childlist	187	\@glo@sortentries	189, 190
\@glo@counter	62, 78, 81	\@glo@sorthandler@case	190
\@glo@counterprefix ..	173, 178–180, 206, 209	\@glo@sorthandler@letter	189
\@glo@default@sorttype ..	10, 170, 189, 190	\@glo@sorthandler@nocase	190
\@glo@defaultcounter	81	\@glo@sorthandler@word	189

\@glo@sortinghandler	186, 188	\@gls@checkactual	111, 163
\@glo@sortinglist	186, 188, 191	\@gls@checkbar	111, 163
\@glo@sorttype	170, 191–193, 198	\@gls@checkedmkidx	110–117, 162–164
\@glo@storeentry	11, 13	\@gls@checkescactual	111, 163
\@glo@sufffix	110, 179, 315	\@gls@checkescbar	111, 163
\@glo@symbol	53, 61, 77, 82, 236, 241, 336	\@gls@checkescquote	111, 163, 164
\@glo@symbolaccess	335–337, 367	\@gls@checklevel	111, 163
\@glo@symbolplural	61, 77, 82	\@gls@checkmkidxchars	
\@glo@symbolpluralaccess	335–337	84, 110, 163, 169, 178, 180, 314, 315
\@glo@text	60,	\@gls@checkquote	111, 162, 163
77, 80, 82, 119–124, 145, 146, 237, 255, 366		\@gls@classI	157
\@glo@textaccess ...	334, 336, 337, 364–367	\@gls@classII	157
\@glo@thislabel	83	\@gls@codepage	186
\@glo@thislettergrp	192, 193	\@gls@counter	
\@glo@thisvalue	54, 55	104, 107–109, 153, 154, 173, 179, 180, 315	
\@glo@tmp	71, 72, 179	\@gls@counterwithin	10, 197, 199
\@glo@type	7, 12, 13, 61, 77–79,	\@gls@ctr	41
81, 82, 153, 154, 168, 173, 174, 183–186,		\@gls@currentlettergroup	192, 193
188, 191, 192, 195, 196, 216, 231, 233,		\@gls@declareoption	
235, 237, 239, 240, 242, 244, 246, 261, 262		8, 9, 13, 14, 17, 22, 25–28
\@glo@types		\@gls@default	93
. 49, 50, 57, 87, 88, 154, 168, 169, 252, 309		\@gls@default@value	
\@glo@useri	63, 78, 81	53–55, 65, 66, 77, 78, 80, 81, 240, 254
\@glo@useriii	63, 78, 81	\@gls@deffile	68, 69
\@glo@useriii	63, 78, 81	\@gls@defsort	11, 12, 82
\@glo@useriv	64, 78, 81	\@gls@defsortcount	11, 12, 58
\@glo@userv	64, 78, 81	\@gls@do@acronymsdef	14, 29, 59
\@glo@uservi	64, 78, 81	\@gls@do@indexdef	28, 29, 59
\@glodesc	81	\@gls@do@numbersdef	28, 29, 59
\@glolist@	79	\@gls@do@symbolsdef	27, 59
\@gloname	81	\@gls@do@symbolssdef	29
\@glossary@default@style		\@gls@doautomake	26, 169
.....	7, 9, 183, 206, 207, 247	\@gls@docheckquotedef	162–164
\@glossaryentryfield	84	\@gls@docloadedfalse	4
\@glossarysection	37	\@gls@docloadedtrue	4
\@glossarystyle	183, 184, 196	\@gls@dodeflistparser	168
\@glossarysubentryfield	84, 85	\@gls@doentrydef	103
\@gls	118	\@gls@dolast	181
\@gls@	89, 91, 119, 257, 258	\@gls@donext	181
\@gls@@link	106	\@gls@donext@def	152
\@gls@Hcounter	108, 109	\@gls@dothiswrite	166, 167
\@gls@ReturnAfterFi	210	\@gls@elem	261
\@gls@access@display	339, 340	\@gls@enablesavenonumberlist	68
\@gls@actualchar ..	84, 85, 113, 115, 160, 318	\@gls@encapchar	
\@gls@addpredefinedattributes ..	156, 165	113, 114, 160, 179, 180, 315, 318
\@gls@adjustmode	153	\@gls@entry@count	90
\@gls@automake	166, 169	\@gls@entry@field	
\@gls@between	262, 263	71, 72, 88, 89, 145–152, 337–339
\@gls@body	146	\@gls@escbsdq	111, 161, 319

<code>\@gls@expand@fields</code>	66, 67	<code>\@gls@noidx@do</code>	192
<code>\@gls@expandonce</code>	67	<code>\@gls@noidx@getgrouptitle</code>	170, 206
<code>\@gls@extramakeindexopts</code>	167	<code>\@gls@noidx@sanitizesort</code>	19, 172
<code>\@gls@fetchfield</code>	55	<code>\@gls@noidx@setsanitizesort</code>	21, 172
<code>\@gls@field@link</code>	72, 73, 125–137	<code>\@gls@noidx@loclist@finalsep</code>	171
<code>\@gls@firsttok</code>	192	<code>\@gls@noidx@loclist@prev</code>	171, 194, 195
<code>\@gls@fixbraces</code>	82	<code>\@gls@noidx@loclist@sep</code>	171, 194
<code>\@gls@forbidtexext</code>	57	<code>\@gls@noref@warn</code>	169, 192
<code>\@gls@get@counterprefix</code>	179	<code>\@gls@numberlink</code>	209
<code>\@gls@getbody</code>	146	<code>\@gls@numbersdef</code>	28
<code>\@gls@getcounterprefix</code>	178	<code>\@gls@numlist@lastsep</code>	152, 153
<code>\@gls@getgrouptitle</code>	170, 205, 263	<code>\@gls@numlist@nextsep</code>	152
<code>\@gls@glossary</code>	174, 175	<code>\@gls@numlist@sep</code>	152, 153
<code>\@gls@gobbleopt</code>	56	<code>\@gls@old@chapter</code>	30
<code>\@gls@grptitle</code>	205, 261, 263	<code>\@gls@oldnewglossaryentryposthook</code>	336
<code>\@gls@hyp@opt</code>	72, 73, 91, 92, 106, 118–123, 125–144, 212–215, 256–259	<code>\@gls@oldnewglossaryentryprehook</code>	336
<code>\@gls@hyp@opt@cs</code>	105	<code>\@gls@onlypremakeg</code>	30
<code>\@gls@hypergroup</code>	261	<code>\@gls@order</code>	166, 167
<code>\@gls@ifinlist</code>	41	<code>\@gls@org@LT@output</code>	278
<code>\@gls@ifnotmeasuring</code>	85	<code>\@gls@org@glsnoidx@displayloc</code>	172
<code>\@gls@igtype</code>	60	<code>\@gls@org@glssseeformat</code>	172
<code>\@gls@increment@currcount</code>	89	<code>\@gls@preglossaryhook</code>	184
<code>\@gls@indexdef</code>	28	<code>\@gls@prevlevel</code>	289, 290, 310–312, 327, 328
<code>\@gls@initnonumberlist</code>	63, 78	<code>\@gls@provide@newglossary</code>	57
<code>\@gls@islistofacronyms</code>	15	<code>\@gls@quotechar</code>	112–115, 160, 162, 164, 318
<code>\@gls@keylist</code>	363	<code>\@gls@reference</code>	170, 173
<code>\@gls@keymap</code>	63, 68, 70, 71, 254, 336	<code>\@gls@removespaces</code>	209, 210
<code>\@gls@label</code>	170, 173, 178, 179	<code>\@gls@renewglossary</code>	165
<code>\@gls@langmod</code>	166	<code>\@gls@replacementtext</code>	339
<code>\@gls@levelchar</code>	85, 114, 115, 160, 318	<code>\@gls@rest</code>	146
<code>\@gls@link</code>	106, 119–124, 137–144, 351–353	<code>\@gls@roman</code>	44, 45, 315, 316
<code>\@gls@link@checkfirsthyper</code>	106, 119–124	<code>\@gls@sanitized@tmp</code>	110, 111
<code>\@gls@link@label</code>	107, 232, 238	<code>\@gls@sanitizedesc</code>	24
<code>\@gls@link@nocheckfirsthyper</code>	124, 137–144	<code>\@gls@sanitizesort</code>	11
<code>\@gls@link@opts</code>	107, 232, 238	<code>\@gls@sanitizesymbol</code>	24, 25
<code>\@gls@list</code>	261, 262	<code>\@gls@saveentrycounter</code>	108, 154
<code>\@gls@listsuffix</code>	41	<code>\@gls@savenonumberlist</code>	62, 63
<code>\@gls@loadlist</code>	9, 247	<code>\@gls@see@noindex</code>	6, 62
<code>\@gls@loadlong</code>	8, 9, 247	<code>\@gls@setacrstyle</code>	24, 25, 29
<code>\@gls@loadsuper</code>	9, 247	<code>\@gls@setcounter</code>	58
<code>\@gls@loadtree</code>	9, 247	<code>\@gls@setdefault@glslink@opts</code>	107
<code>\@gls@local@increment@currcount</code>	89	<code>\@gls@setsort</code>	11, 12, 108
<code>\@gls@loclist</code>	171, 172, 193, 194	<code>\@gls@setupshortcuts</code>	29
<code>\@gls@map</code>	68–70	<code>\@gls@sort</code>	193
<code>\@gls@missingnumberlist</code>	81	<code>\@gls@sort@A</code>	188, 189
<code>\@gls@noaccess</code>	339	<code>\@gls@sort@B</code>	188, 189
<code>\@gls@noexpand@fields</code>	67	<code>\@gls@startswithexpandonce</code>	66
<code>\@gls@nohyperlist</code>	16, 59, 107	<code>\@gls@storenonumberlist</code>	63, 81
		<code>\@gls@symbolsdef</code>	27

\@gls@this	176	\@glslink	108, 118, 153, 261
\@gls@thisHloc	179	\@glslocalreset	86, 89
\@gls@thisfield	55	\@glslocalunset	86, 89
\@gls@thislabel	52, 181, 191	\@glslocref	173, 178, 179, 314, 315
\@gls@thislist	152, 153	\@glsminrange	156, 157, 316
\@gls@thisloc	179	\@glsname	128
\@gls@thisval	70	\@glsname@	128
\@gls@title	37	\@glsnextpages	184
\@gls@tmp	12, 13, 32, 46, 67, 110, 111, 175, 262, 263	\@glsnodesc	77, 79, 81
\@gls@tmpb	112–117, 162, 164	\@glsnoname	77, 80, 81
\@gls@toc	39	\@glsnonextpages	184
\@gls@type	169, 217, 231, 234, 235, 238, 240, 243, 245, 247, 309	\@glsnumberformat	104, 107, 153, 173, 178, 179, 314, 315
\@gls@updatechecked	110, 111, 163	\@glsopenfile	165, 174
\@gls@usetranslator	22, 23, 32	\@glsorder	167
\@gls@value	65, 66, 148	\@glspl	121
\@gls@warnonglossdefined	17, 182	\@glspl@	89, 92, 121, 257–259
\@gls@warnonthehglossdefined	17, 201	\@glsplural	126
\@gls@write@entrycounts	90	\@glsplural@	126
\@gls@writedef	68	\@glsreset	85, 89
\@gls@writeisthook	160–162	\@glssee	82, 181
\@gls@xdy@locationlist	157	\@glssymbol	130
\@gls@xdycheckbackslash	110	\@glssymbol@	130
\@gls@xdycheckquote	110	\@glssymbolplural	131
\@gls@xref	180	\@glssymbolplural@	131
\@glsAlpha compositor	35, 45, 316	\@glstarget	118, 202, 261
\@glsHlocref	178	\@glstext	125
\@glsacronymlists	15, 16, 49, 216, 217, 231, 233–235, 237–240, 242–247, 252	\@glstext@	125
\@glsaddkey	71	\@glsunset	86, 89
\@glsaddstoragekey	70	\@glsuseri	132
\@glsaddxdyattribute	42, 43	\@glsuseri@	132
\@glsdefaultsort	11	\@glsuserii	133
\@glsdesc	129	\@glsuserii@	133
\@glsdesc@	129	\@glsuseriii	134
\@glsdescplural	129, 130	\@glsuseriii@	134
\@glsdescplural@	130	\@glsuseriv	134
\@glsdisp	123	\@glsuseriv@	134, 135
\@glsentry	87, 88, 90	\@glsuserv	135
\@glsentrytitlecase	148, 149	\@glsuserv@	135
\@glsfirst	125	\@glsuservi	136
\@glsfirst@	125, 126	\@glsuservi@	136
\@glsfirstletter	49, 155	\@glswidestname	309–311, 327
\@glsfirstplural	127	\@glswritefiles	27
\@glsfirstplural@	127	\@gobble	12, 68, 69, 110, 155, 158, 169, 313, 317, 318
\@glshypernumber	208	\@idxitem	304
\@glsisacronymlistfalse	16	\@ifclassloaded	4, 10, 38
\@glsisacronymlisttrue	16	\@ifnextchar	58, 105

\@ifpackageloaded	\@printglossary	182, 183
..... 4, 7, 22, 23, 32, 48, 85, 152, 162, 333	\@roman	44, 315
\@ifstar	\@secondofthree	
\@ifundefined	105, 118, 120, 122, 138, 139, 141, 143, 351	
. 32, 262, 269, 280, 291, 298, 311, 327, 341	\@secondoftwo	22, 23, 32, 68, 70, 118–
\@ignored@glossaries	120, 124, 137, 138, 141, 142, 351–353, 371	
59, 60	\@set@glo@numformat	179, 315
\@input@	\@sglsaddkey	71
185	\@sglsaddstoragekey	70
\@istfilename	\@thirdofthree	
167 105, 120, 123, 138, 140, 142, 144, 351	
\@makecol	\@this@attr	158
279	\@this@childlabel	187
\@makeglossary	\@this@counter	43
168	\@this@ctr	158
\@minus	\@this@key	70
265, 285, 303	\@this@label	186, 187
\@mkboth	\@thiscs	30
38	\@tmp	44, 316
\@newglossary	\@use@option	28
56, 57	\@warn@nomakeglossaries	167, 186
\@newglossaryentry@defcounters ..	\@wrglossary@pageformat	177
82, 88	\@wrglossarynumberhook	177, 178
\@newglossaryentryposthook	\@xdy@main@language	25, 166, 185
..... 71, 72, 83, 254, 336	\@xdy@attributelist	42, 158
\@newglossaryentryprehook	\@xdy@attributes	42, 156, 313, 315
..... 71, 76, 78, 254, 336	\@xdy@counters	41, 43, 158
\@nil	\@xdy@language	185
15, 82, 109–111, 146,	\@xdy@lettergroups	49, 160, 318
163, 178, 180, 192, 193, 208–210, 314, 315	\@xdy@locationclassorder	47, 158, 317
\@nnil	\@xdy@locref	42, 159, 313, 317
15, 181	\@xdy@requiredstyles	47, 156, 315
\@no@makeglossaries	\@xdy@sortrules	47, 160, 318
168, 170	\@xdystyle	156, 315
\@no@post@desc	\@xdyuseralphabets	44, 157, 315
320	\@xdyuserlocationdefs ...	46, 157, 314, 316
\@nopostdesc	\@xdyuserlocationnames	46, 314
184	\@xfor@nextelement	181
\@onelevel@sanitize	\\	83, 110, 155, 160,
19, 44,	161, 208, 209, 318, 319, 321–323, 331, 332	
68, 84, 110, 111, 159, 180, 182, 192, 316, 317	\{	68, 69, 155, 161, 313, 318, 319
\@onlypreamble ...	\}	69, 155, 161, 313, 319
58, 68, 77, 90, 93, 169, 172	\^	19
\@onlypremakeg	\‘	19
34–36, 42, 43, 46, 58, 162	\ 	111, 113, 163
\@org@glossaryentrynumbers	\~	20
183, 185		
\@org@gls@assign@descplural		
..... 231, 239, 242, 244, 364, 367, 368		
\@org@gls@assign@firstpl		
230,		
231, 233, 235, 237, 239, 242, 244, 364–368		
\@org@gls@assign@plural		
..... 231, 233, 235, 237, 239, 242, 244, 364–368		
\@org@gls@assign@symbolplural		
..... 231, 233, 235, 237, 242, 244, 365, 366, 368		
\@org@gls@numberformat		
152		
\@org@newglossaryentryprehook		
76		
\@outputpage		
279		
\@p@glossarysection		
37		
\@pgls		
256		
\@pgls@		
256		
\@pglspl		
257		
\@pglspl@		
257		
\@plus		
265, 285, 303		
\@print@glossary		
182		
\@print@noidx@glossary		
183		
\@printgloss@setsort		
169, 170, 184		

A

\AA	20
-----------	----

<code>\cs</code>	106	<code>\defglentryfmt</code>	58, 59, 103, 217, 230, 232, 234, 236, 238, 241, 243, 246
<code>\csdef</code>	18, 70–73, 82, 83, 88, 89, 186, 187, 208, 218, 319	<code>\define@boolkey</code> 5, 6, 8–10, 14, 20, 21, 24–27, 104, 198
<code>\csedef</code>	90, 177	<code>\define@choicekey</code> 6, 7, 10, 21, 23, 25, 62, 196, 197
<code>\csgdef</code>	37, 56, 59, 89, 90, 182, 195	<code>\define@key</code> ...	7, 10, 16, 21, 25, 26, 60–64, 70, 71, 104, 153, 195, 196, 198, 254, 334, 335
<code>\cslet</code>	63, 76, 77, 83, 191	<code>\DefineAcronymSynonyms</code>	29, 230
<code>\csname</code>	10–13, 28, 31–33, 39, 42, 44, 45, 48, 50, 52, 57–59, 65, 66, 70– 74, 76, 79–87, 103, 107–110, 119–124, 137–145, 152, 154, 157, 158, 163–166, 170, 173–175, 177, 179, 180, 183–185, 187, 196, 202, 203, 206, 207, 211, 247– 255, 261, 262, 309, 311, 313–315, 327, 333, 334, 336–338, 341, 351–353, 369, 370	<code>\delimN</code>	159, 168, 194, 209, 317
<code>\csshow</code>	252	<code>\delimR</code>	159, 208, 209, 317
<code>\csuse</code>	33, 37, 56, 65, 66, 72, 73, 103, 166, 167, 187, 190, 191, 193, 195–197, 207, 218, 255, 320–332	<code>\DescriptionDUANewAcronymDef</code>	235
<code>\csxdef</code>	81, 90	<code>\DescriptionFootnoteNewAcronymDef</code> .	233
<code>\currentglossary</code>	37, 184, 197, 199	<code>\descriptionname</code>	33, 270, 272– 276, 281, 283–285, 292–297, 299, 301–303
<code>\currentglssubentry</code>	197–200	<code>\DescriptionNewAcronymDef</code>	237
<code>\CurrentOption</code>	28, 254, 333	<code>\dimen@</code>	220, 279, 309
<code>\CurrentTrackedLanguage</code>	33, 371, 372	<code>\disable@keys</code>	29
<code>\CurrentTrackedTag</code>	33, 371, 372	<code>\do</code>	23, 28, 30, 41, 43, 49, 50, 68, 70, 110, 152, 156–158, 168, 169, 176, 181, 186, 187, 217, 231, 234, 235, 238, 240, 243, 245, 247, 261, 262, 315
<code>\CustomAcronymFields</code>	246	<code>\do@glo@storeentry</code>	11, 12, 82
<code>\CustomNewAcronymDef</code>	247	<code>\do@gls@link@checkfirsthyper</code> 106, 108, 119–124, 137–144, 351, 352
D			
<code>\d</code>	19	<code>\do@gls@xdycheckbackslash</code>	110
<code>datatool package</code>	188	<code>\do@gls@disablehyperinlist</code>	108
<code>\day</code>	156, 160, 315, 318	<code>\do@gls@haschildren</code>	52
<code>\DeclareAcronymList</code>	14, 16, 216, 217, 231, 233, 235, 237, 240, 242, 244, 246	<code>doc package</code>	4, 5, 13
<code>\DeclareListParser</code>	168	<code>\doifglossarynoexistsordo</code>	57
<code>\DeclareOption</code>	8, 254, 333	<code>\dtl@ifsingle</code>	205
<code>\DeclareOptionX</code>	8	<code>\dtl@insertinto</code>	188
<code>\DeclareRobustCommand</code> 34, 181, 240, 339–341		<code>\dtl@sortresult</code>	188, 189
<code>\def</code>	8, 11, 12, 15, 19, 20, 25, 26, 29–31, 33, 34, 37, 41, 44–49, 52, 56–58, 60–64, 67, 74, 76, 78–83, 89–93, 103, 104, 107–117, 119–144, 152, 153, 156, 160, 162–164, 166, 168, 170, 171, 173, 177–180, 183, 186, 191, 192, 194– 199, 205–216, 231, 233, 235–237, 239, 241, 242, 244, 246, 254, 256–260, 262– 265, 289, 290, 309–312, 314, 315, 318, 320, 327, 328, 333–336, 350–353, 364–368	<code>\dtlcompare</code>	189
<code>\def@gls@xdycheckbackslash</code>	116, 117	<code>\dtlicompare</code>	189
<code>\DefaultNewAcronymDef</code>	231	<code>\DTLifinlist</code>	60, 107
		<code>\DTLifint</code>	206
		<code>\dtlletterindexcompare</code>	188
		<code>\DTLsubstituteall</code>	111
		<code>\dtlwordindexcompare</code>	188
		<code>\DUANewAcronymDef</code>	245
E			
<code>\eappto</code>	59, 83, 177	<code>\edef</code>	12, 15, 30, 33, 41, 42, 44–47, 50, 52, 57, 59, 60, 65, 66, 70, 73–78, 83, 84, 103, 107–117, 152, 154, 155, 160, 162–164, 166–168, 170, 174, 178, 179, 182, 185–189, 193, 197, 200,

206, 209, 211, 230, 232, 234, 236, 239, 241, 243, 261, 313, 314, 316, 318, 363–367	
\egroup	19, 77, 153, 185, 187
\else	5, 9, 12–15, 17, 18, 20, 21, 26–30, 34, 35, 38, 40–44, 46–49, 62, 65, 79, 80, 83–85, 89, 90, 107–117, 119–124, 146, 155, 156, 159–166, 174–181, 184, 193, 197–201, 206, 208–210, 220, 234, 235, 238, 240, 243, 245, 247, 262, 266, 270, 271, 273, 276, 277, 279, 280, 282, 284, 292, 293, 295, 298, 300, 302, 305, 307, 308, 310, 311, 314–320, 325–328, 339
\emph	181, 210
\empty	209
\end	106, 159, 166, 192, 266, 269–275, 277, 278, 280–303, 317
\end@doifinlist	41
\end@getprefix	179
\end@gl@s@islistofacronyms	15
\EndAccSupp	339
\endcsname	10–13, 28, 31–33, 39, 42, 44, 45, 48, 50, 52, 57–59, 65, 66, 70–74, 76, 79–87, 103, 107–110, 119– 124, 137–145, 152, 154, 157, 158, 163– 166, 170, 173–175, 177, 179, 180, 183– 187, 196, 202, 203, 206, 207, 211, 247– 255, 261, 262, 309, 311, 313–315, 327, 333, 334, 336–338, 341, 351–353, 369, 370
\endfoot	270–277, 281–285
\endgroup	5, 175, 178, 209
\endhead	270–277, 281–285
\endtheglossary	5
\entryname	33, 270, 272– 276, 281, 283–285, 292–297, 299, 301–303
\equal	21, 29, 39, 108, 168, 206, 261
equation (counter)	108, 109
etoolbox package	4
\expandafter	11–13, 19, 28, 30, 32, 33, 42, 44, 45, 47–50, 52, 57, 58, 60, 65, 66, 68–74, 76, 79, 80, 82, 84–87, 103, 107– 116, 146, 153, 155, 158, 162–165, 174– 176, 178, 179, 181, 184, 187, 188, 192, 193, 202, 203, 207, 209, 211, 232, 238, 247–255, 261, 262, 309, 313–315, 317, 318, 333, 334, 336, 337, 339, 363, 369, 370
\expandonce	65–67, 110, 111, 163, 164, 177, 188, 189, 202, 203, 216, 230, 232–234, 237, 239, 242–244, 333, 334
F	
\fi	5–7, 9, 11–15, 17, 18, 20, 21, 23, 26– 30, 34, 35, 38, 40–49, 58, 62, 65, 79–85, 89, 90, 106–117, 119–124, 146, 154–156, 159, 161–165, 167–169, 174–184, 186, 193, 196–201, 206–210, 220, 230, 231, 233–235, 237, 238, 240, 243–247, 253, 262, 266, 270, 271, 273, 276–279, 281, 282, 284, 292, 293, 295, 299, 300, 302, 305–311, 313–317, 319, 320, 325–328, 339
file types	
.aux	185
.glo	84
.ist	155, 165
.toc	40
.xdy	35
glo	253
\firstacronymfont	102, 218–220, 226, 232, 236, 238, 241, 350, 354–356, 360, 361
\footnote	226, 232, 361
\FootnoteNewAcronymDef	240
\foralllglossaries	50, 173, 183, 309
\foralllglsentries	87, 88, 90, 154
\ForEachTrackedDialect	33, 371, 372
\forlglsentries	50, 52, 83, 191, 309
\forlistcsloop	186, 192
\forlistloop	171, 172, 194
G	
garamondx package	212
\gdef	12, 42, 57, 74, 79, 80, 175, 198, 199, 262
\Genacrfullformat	101, 216–220, 226, 350, 354, 355, 361
\genacrfullformat	101, 102, 216–220, 226, 349, 350, 354, 355, 360
\GenericAcronymFields	216, 218–228, 354–357, 359, 360, 363
\Genplacrfullformat	101, 217, 219, 220, 226, 349, 355, 361
\genplacrfullformat	101, 102, 216, 217, 219, 220, 226, 349, 355, 361
\glo@desc	320
\glo@do@compare	188, 189
\glo@grabfirst	193
\glo@label	52, 83
\glo@list	83
\glo@name	202
\glo@parent	52
\glo@type	83

<code>\glo@value</code>	68, 69	<code>list</code>	7, 266–268, 320
<code>\global</code>	12, 13, 65, 68, 76, 77, 82, 86, 87, 175, 185, 193, 199, 279	<code>listdotted</code>	268, 269, 321
<code>\glo@linkprefix</code>	108, 153, 202	<code>listgroup</code>	266, 267, 320
<code>\gloskey</code>	106	<code>listhypergroup</code>	267, 321
<code>glossareentry (counter)</code>	200	<code>long</code>	269–271, 275, 280, 321, 323
<code>glossaries package</code>	28, 48, 156, 247, 254, 266, 313, 333	<code>long-booktabs</code>	275, 277
<code>glossaries-accsupp package</code>	83, 333	<code>long3col</code>	271, 272, 276, 322
<code>glossaries-extra package</code>	333	<code>long3col-booktabs</code>	276, 278
<code>\GlossariesWarning</code>	5, 6, 17, 20, 21, 37, 40, 51, 55, 62, 65, 91, 92, 103, 105, 152, 166, 167, 169, 171–173, 175, 180, 183, 204, 207, 313	<code>long3colborder</code>	271, 322
<code>\GlossariesWarningNoLine</code>	5, 17, 168, 170, 174, 186, 262	<code>long3colheader</code>	272, 276, 322
<code>\glossary</code>	314, 315	<code>long3colheaderborder</code>	272, 322
<code>glossary package</code>	1, 211	<code>long4col</code>	272–274, 276, 322
<code>glossary styles:</code>		<code>long4col-booktabs</code>	276, 277
<code>altlist</code>	267, 268, 321	<code>long4colborder</code>	273, 323
<code>altlistgroup</code>	267, 268, 321	<code>long4colheader</code>	273, 276, 323
<code>altlisthypergroup</code>	268, 321	<code>long4colheaderborder</code>	273, 323
<code>altlong4col</code>	274, 283, 323	<code>longborder</code>	270, 322
<code>altlong4col-booktabs</code>	277, 278	<code>longheader</code>	270, 275, 322
<code>altlong4colborder</code>	274, 323	<code>longheaderborder</code>	270, 322
<code>altlong4colheader</code>	274, 277, 323	<code>longragged</code>	277, 280, 281
<code>altlong4colheaderborder</code>	275, 323	<code>longragged-booktabs</code>	277
<code>altlongragged4col</code> ...	278, 283, 284, 324	<code>longragged3col</code>	278, 281, 282, 324
<code>altlongragged4col-booktabs</code>	278	<code>longragged3col-booktabs</code>	278
<code>altlongragged4colborder</code>	284, 324	<code>longragged3colborder</code>	282, 324
<code>altlongragged4colheader</code>	284, 324	<code>longragged3colheader</code>	282, 324
<code>altlongragged4colheaderborder</code>	284, 325	<code>longragged3colheaderborder</code> .	283, 324
<code>altsuper4col</code>	296, 297, 301, 332	<code>longraggedborder</code>	281, 323
<code>altsuper4colborder</code>	297, 332	<code>longraggedheader</code>	281, 324
<code>altsuper4colheader</code>	296, 332	<code>longraggedheaderborder</code>	281, 324
<code>altsuper4colheaderborder</code> ...	297, 332	<code>mcolalmtree</code>	289, 329
<code>altsuperragged4col</code>	301–303, 330	<code>mcolalmtreegroup</code>	290, 329
<code>altsuperragged4colborder</code> ...	302, 330	<code>mcolalmtreehypergroup</code>	290, 329
<code>altsuperragged4colheader</code> ...	302, 330	<code>mcolindex</code>	286, 328
<code>altsuperragged4colheaderborder</code> .	303, 330	<code>mcolindexgroup</code>	286, 328
<code>alttree</code>	289, 304, 309, 310, 312, 327	<code>mcolindexhypergroup</code>	286, 328
<code>alttreegroup</code>	312, 328	<code>mcoltree</code>	287, 328
<code>alttreehypergroup</code>	312, 328	<code>mcoltreegroup</code>	328
<code>index</code>	7, 285, 304–306, 325	<code>mcoltreehypergroup</code>	287, 328
<code>indexgroup</code>	305, 306, 325	<code>mcoltreenoname</code>	288, 329
<code>indexhypergroup</code>	306, 325	<code>mcoltreenonamegroup</code>	288, 329
<code>inline</code>	320	<code>mcoltreenonamehypergroup</code>	288, 289, 329
		<code>sublistdotted</code>	321
		<code>super</code>	291–293, 299, 331
		<code>super3col</code>	293, 294, 331
		<code>super3colborder</code>	293, 331
		<code>super3colheader</code>	294, 331
		<code>super3colheaderborder</code>	294, 331
		<code>super4col</code>	294–296, 332

super4colborder	295, 332	\glossentry ..	83, 184, 194, 203, 204, 264, 266–269, 271, 272, 280, 282, 283, 291, 293, 295, 298, 300, 302, 305, 306, 308, 310
super4colheader	295, 332	\Glossentrydesc	353
super4colheaderborder	296, 332	\glossentrydesc 264, 266–273, 280, 282, 283, 291–293, 295, 298, 300, 302, 305–308, 310, 311, 353
superborder	292, 331	\glossentryname	264, 266–269, 271, 272, 280, 282, 283, 291, 293, 295, 298, 300, 302, 305, 306, 308, 310, 311, 353
superheader	292, 331	\Glossentrysymbol	354
superheaderborder	292, 331	\glossentrysymbol	264, 272, 273, 283, 284, 295, 302, 305–308, 310, 311, 354
superragged	298, 299, 329	\Gls	91, 211, 229
superragged3col	299–301, 330	\gls	91, 169, 200, 211, 229
superragged3colborder	300, 330	\gls@Alphpage	176, 178
superragged3colheader	301, 330	\gls@alphpage	176, 178
superragged3colheaderborder	301, 330	\gls@arabicpage	176, 178
superraggedborder	299, 329	\gls@assign@desc	76, 81
superraggedheader	299, 329	\gls@assign@descplural	231, 239, 242, 244, 364, 367, 368
superraggedheaderborder	299, 330	\gls@assign@field	67, 71, 72, 76, 78, 80–82, 254, 255
tree	287, 306, 307, 310, 325	\gls@assign@firstpl	230, 231, 233, 235, 237, 239, 242, 244, 364–368
treegroup	287, 307, 326	\gls@assign@plural	231, 233, 235, 237, 239, 242, 244, 364–368
treehypergroup	307, 326	\gls@assign@symbolplural	231, 233, 235, 237, 242, 244, 365, 366, 368
treenoname	288, 304, 307, 308, 326	\gls@checkisacronymlist	106
treenonamegroup	308, 327	\gls@checkseeallowed	62, 67, 168, 170
treenonamehypergroup	308, 327	\gls@checkseeallowed@preambleonly ..	67
glossary-hypernav package	155	\gls@codepage	48, 166, 186
glossary-list package	7, 9, 265	\gls@defdocnewglossaryentry	68, 88
glossary-long package	8, 269, 283, 291	\gls@defglossaryentry	67, 68, 77
glossary-longragged package	279	\gls@disablepagerefexpansion ..	175, 178
glossary-mcols package	285	\gls@do@addxdyattribute	43
glossary-super package ...	9, 269, 291, 297, 301	\gls@doclearpage	40
glossary-superragged package	297	\gls@dosubst	111
glossary-tree package	9, 303	\gls@dotocitle	183, 184, 195, 196
\glossaryentry	179, 180, 315	\gls@end@sanitizesort	19
glossaryentry (counter)	10, 200, 201	\gls@endcheck	66, 67
\glossaryentryfield	202, 320–327, 329–332, 354	\gls@glossary	174, 179, 180
\glossaryentrynumbers	8, 159, 183–185, 193, 194, 198, 199, 317	\gls@gobbleopt	58
\glossaryheader	158, 192, 263, 266–276, 280–288, 290, 291, 293, 294, 298, 300, 301, 305–310, 312, 317	\gls@grplabel	261
\glossarymark	38	\gls@hypergroup prerun	262
\glossaryname	13, 33	\gls@ifnotmeasuring	85, 86
\glossarypostamble	159, 192, 317	\gls@inlinepostchild	263–265, 320
\glossarypreamble	158, 192, 317	\gls@inlinesep	263, 264, 320
\glossarysection	158, 192, 317		
glossarysubentry (counter)	10, 199–201		
\glossarysubentryfield	203, 320–327, 329–332, 354		
\glossarytitle ..	158, 183, 184, 192, 195, 317		
\glossarytoctitle	7, 13, 14, 27, 28, 31, 33, 38, 158, 183, 192, 196, 317		

<code>\gls@inlinesubsep</code>	263, 264, 320	<code>\gls@clearpage</code>	39
<code>\gls@islistofacronyms</code>	15	<code>\gls@closebrace</code>	46, 159, 160, 317, 318
<code>\gls@istfilebase</code>	34, 166	<code>\gls@compositor</code>	35, 45, 161, 316, 319
<code>\gls@label</code>	211	<code>\gls@counter</code>	16, 29, 41, 58, 81, 108, 313
<code>\gls@level</code>	79, 80, 193	<code>\gls@currententrylabel</code>	182, 184
<code>\gls@noidxglossary</code>	170	<code>\gls@currentfieldvalue</code>	54, 55
<code>\gls@nosetquote</code>	77, 160, 162, 164	<code>\gls@customtext</code>	
<code>\gls@numberpage</code>	176, 177	... 94, 97, 99, 100, 102, 119–124, 137–	
<code>\gls@org@glossaryentryfield</code>	184	144, 223, 224, 232, 236, 238, 239, 241,	
<code>\gls@org@glossarysubentryfield</code>	184	341, 344, 345, 347, 348, 350–353, 358, 359	
<code>\gls@org@insert</code>	236, 238, 241	<code>\GlsDeclareNoHyperList</code>	16
<code>\gls@protected@pagefmts</code>	110, 176, 177	<code>\gls@defaulttype</code>	13,
<code>\gls@Romanpage</code>	176, 178	37, 48, 50, 56, 57, 78, 93, 103, 174, 182, 183	
<code>\gls@romanpage</code>	176, 178	<code>\gls@defmain</code>	13, 59
<code>\gls@save@numberlist</code>	8	<code>\gls@descriptionaccessdisplay</code>	
<code>\gls@suffixF</code>	36, 159, 161, 317, 319 343–345, 353, 354	
<code>\gls@suffixFF</code>	36, 159, 161, 317, 319	<code>\gls@descriptionpluralaccessdisplay</code>	
<code>\gls@text</code>	102 341, 342	
<code>\gls@thissty</code>	23	<code>\gls@descwidth</code>	269–271, 274,
<code>\gls@tmp</code>	173, 174, 240	275, 277, 278, 280–285, 291–294, 296–303	
<code>\gls@tmplen</code>	117, 309, 311, 327, 328	<code>\gls@detoklabel</code>	51–55, 63, 68, 73–77,
<code>\gls@tr@set@acronym@toctitle</code>	14	83, 86–90, 107, 145, 146, 152–154, 170–	
<code>\gls@tr@set@main@toctitle</code>	13	172, 178, 184, 187–189, 193, 195, 197,	
<code>\gls@tr@set@numbers@toctitle</code>	28	200, 202, 247–252, 309, 333, 334, 369, 370	
<code>\gls@tr@set@symbols@toctitle</code>	27	<code>\gls@display</code>	94, 103
<code>\gls@wrglossary</code>	175	<code>\gls@displayfirst</code>	94, 103
<code>\gls@xdystring</code>	110, 111	<code>\gls@displaynumberlist</code>	171
<code>\gls@xindy@glsnumbersfalse</code>	26	<code>\gls@dohyperlink</code>	117, 118
<code>\gls@xindy@glsnumberstrue</code>	25	<code>\gls@dohypertarget</code>	118
<code>\gls@accsupp</code>	339	<code>\gls@doifexists</code>	
<code>\gls@acronymtrue</code>	14	.. 52–54, 73–76, 85, 86, 119–124, 137–	
<code>\gls@acrpluralsuffix</code>		144, 152, 153, 171, 172, 256–260, 350–354	
..... 31, 212, 221, 225–227, 231		<code>\gls@doifexistsordo</code>	106, 145
<code>\gls@acrshortcutsfalse</code>	29	<code>\gls@doifexistsorwarn</code>	195, 202, 203
<code>\gls@acrshortcutstrue</code>	29	<code>\gls@doifnoexists</code>	67, 76
<code>\gls@acspace</code>	219, 222	<code>\gls@donohyperlink</code>	108, 117, 118
<code>\gls@add</code>	154	<code>\gls@dosanitizesort</code>	11
<code>\gls@add options</code>		<code>\gls@entryaccess</code>	339
counter	153	<code>\gls@entrycounter</code>	206, 209
format	153, 208	<code>\gls@entrycounterfalse</code>	10
<code>\gls@addall options</code>		<code>\gls@entrycounterlabel</code>	197, 201
types	153, 154	<code>\gls@entrycountertrue</code>	10
<code>\GlsAddXdyAttribute</code>	42, 43, 313, 314	<code>\gls@entrycurrcount</code>	88, 90
<code>\GlsAddXdyCounters</code>	42, 43, 58	<code>\Gls@entrydesc</code>	129, 203, 353
<code>\gls@automakefalse</code>	26	<code>\gls@entrydesc . 96, 97, 129, 202, 343–345, 353</code>	
<code>\gls@autoprefix</code>	7, 196	<code>\gls@entrydescaccess</code>	340
<code>\gls@capscase</code>	94, 95, 98–101,	<code>\Gls@entrydescplural</code>	130
119–124, 137–144, 223, 224, 236, 241,		<code>\gls@entrydescplural . 94, 95, 130, 341, 342</code>	
341, 343, 345, 346, 348, 349, 351–353, 358		<code>\gls@entrydescpluralaccess</code>	340

<code>\Glsentryfirst</code>	92, 96, 99, 126, 344, 347	<code>\glsentryshortpl</code>	
<code>\glsentryfirst</code> 91, 96, 97, 99, 126, 343, 344, 347		100, 102, 139, 140, 151, 219,	
<code>\glsentryfirstaccess</code>	340	220, 225–227, 246, 348, 350, 355, 359–362	
<code>\Glsentryfirstplural</code> 93, 95, 98, 127, 342, 346		<code>\glsentryshortpluralaccess</code>	340
<code>\glsentryfirstplural</code>	92,	<code>\Glsentrysymbol</code>	131, 203, 354
94, 95, 98, 127, 128, 341, 342, 345, 346		<code>\glsentrysymbol</code>	96, 97,
<code>\glsentryfirstpluralaccess</code>	340	130, 131, 203, 232, 236, 241, 343–345, 354	
<code>\glsentryfmt</code>	58, 59	<code>\glsentrysymbolaccess</code>	340
<code>\Glsentryfull</code>	217, 225, 227, 360, 362	<code>\Glsentrysymbolplural</code>	132
<code>\glsentryfull</code>	217, 225, 227, 359, 362	<code>\glsentrysymbolplural</code>	
<code>\Glsentryfullpl</code>	217, 225, 227, 360, 362	.. 94, 95, 131, 132, 232, 236, 241, 341, 342	
<code>\glsentryfullpl</code>	217, 225, 227, 360, 362	<code>\glsentrysymbolpluralaccess</code>	340
<code>\glsentryitem</code> 197, 264, 266–269, 271, 272,		<code>\Glsentrytext</code>	96, 99, 125, 343, 347
280, 282, 283, 291, 293, 295, 298, 300,		<code>\glsentrytext</code>	96,
302, 305, 306, 308, 310, 320–327, 329–332		97, 99, 125, 153, 181, 343, 344, 346, 347	
<code>\Glsentrylong</code>	92, 141, 146,	<code>\glsentrytextaccess</code>	339
151, 219, 224, 225, 350, 352, 355, 358–360		<code>\glsentrytype</code>	78
<code>\glsentrylong</code>	91, 102, 141,	<code>\Glsentryuseri</code>	132
142, 146, 151, 218–228, 238, 350, 352–363		<code>\glsentryuseri</code>	132, 133
<code>\glsentrylongaccess</code>	340	<code>\Glsentryuserii</code>	133
<code>\Glsentrylongpl</code>	93,	<code>\glsentryuserii</code>	133
143, 151, 219, 224, 225, 350, 355, 358–360		<code>\Glsentryuseriii</code>	134
<code>\glsentrylongpl</code>		<code>\glsentryuseriii</code>	134
92, 102, 143, 144, 151, 219, 220,		<code>\Glsentryuseriv</code>	135
224–227, 238, 246, 350, 355, 356, 358–362		<code>\glsentryuseriv</code>	135
<code>\glsentrylongpluralaccess</code>	340	<code>\Glsentryuserv</code>	136
<code>\Glsentryname</code>	128, 202, 353	<code>\glsentryuserv</code>	135, 136
<code>\glsentryname</code>	128, 129, 309, 353	<code>\Glsentryuservi</code>	136
<code>\glsentrynumberlist</code>	152, 171	<code>\glsentryuservi</code>	136, 137
<code>\Glsentryplural</code>	95, 98, 127, 342, 346	<code>\glsfieldfetch</code>	148
<code>\glsentryplural</code>		<code>\glsfirstaccessdisplay</code>	343, 344, 347
94, 95, 98, 126, 127, 341, 342, 345, 346		<code>\glsfirstpluralaccessdisplay</code>	
<code>\glsentrypluralaccess</code>	339	341, 342, 345, 346	
<code>\Glsentryprefix</code>	258	<code>\glsfirstpluralacesdisplay</code>	346
<code>\glsentryprefix</code>	256, 259	<code>\glsacgenacfmt</code>	218–220, 226, 354, 355, 360
<code>\Glsentryprefixfirst</code>	258	<code>\glsacgenentryfmt</code>	
<code>\glsentryprefixfirst</code>	257, 259	218–220, 224, 226, 230, 232, 234,	
<code>\Glsentryprefixfirstplural</code>	259	236, 238, 241, 243, 246, 354, 355, 359, 360	
<code>\glsentryprefixfirstplural</code>	257, 260	<code>\glsgetgrouptitle</code>	
<code>\Glsentryprefixplural</code>	258	263, 267, 268, 286–290, 305–309, 312	
<code>\glsentryprefixplural</code>	257, 260	<code>\gls glossarymark</code>	38
<code>\glsentryprevcount</code>	88–90	<code>\gls groupheading</code>	160, 193, 263, 266–
<code>\Glsentryshort</code>	100, 138,	269, 271, 272, 280, 282, 283, 286–291,	
146, 220, 226, 227, 349–351, 355, 361, 362		293, 295, 298, 300, 301, 305–310, 312, 318	
<code>\glsentryshort</code>	100, 102, 137, 138, 146,	<code>\gls groupskip</code> 159, 193, 264, 266, 270, 271,	
151, 217–228, 348–351, 354–357, 359–363		273, 276, 277, 280, 282, 284, 292, 293,	
<code>\glsentryshortaccess</code>	340	295, 298–300, 302, 305, 307, 308, 311, 317	
<code>\Glsentryshortpl</code>		<code>\glshyperfirstfalse</code>	226, 360
100, 140, 220, 226, 227, 348, 355, 361, 362		<code>\glshyperfirsttrue</code>	24

<code>\glshyperlink</code>	181	<code>\glslongtok</code>	216–220, 224, 226, 230, 231, 233–235, 237, 239, 240, 242– 244, 246, 247, 354, 355, 359, 360, 363–368
<code>\glshypernavsep</code>	263	<code>\glsLTpenaltycheck</code>	279
<code>\glshypernumber</code>	36, 210	<code>\glsmcols</code>	285–290
<code>\glsifhyperon</code>	105	<code>\glsnameaccessdisplay</code>	353, 354
<code>\glsIfListOfAcronyms</code>	15, 16	<code>\glsnamefont</code>	202, 203, 333, 334, 353
<code>\glsifplural</code>	94, 97, 100, 101, 119–124, 137–144, 223, 232, 236, 238, 241, 341, 345, 348, 349, 351–353, 358	<code>\glsnavhyperlink</code>	263
<code>\glsifusetranslator</code>	22, 23, 32, 33, 371	<code>\glsnavhypertarget</code>	267, 268, 286–290, 306, 307, 309, 312
<code>\glsindexonlyfirstfalse</code>	24	<code>\glsnavigation</code>	267, 268, 286–290, 306, 307, 309, 312
<code>\glsinlinedescformat</code>	264, 320	<code>\glsnextpages</code>	8, 62, 184
<code>\glsinlinedopostchild</code>	264, 320	<code>\glsnogroupskipfalse</code>	9
<code>\glsinlineemptydescformat</code>	264, 320	<code>\glsnoidxdisplayloc</code>	172, 173
<code>\glsinlinenameformat</code>	264, 320	<code>\glsnoidxdisplaylocclsthandler</code>	171
<code>\glsinlineparentchildseparator</code>	264, 320	<code>\glsnoidxloclist</code>	171, 193, 194
<code>\glsinlinepostchild</code>	264, 320	<code>\glsnoidxlocclsthandler</code>	194
<code>\glsinlineseparator</code>	264, 320	<code>\glsnoidxnumberlistloophandler</code>	172
<code>\glsinlinesubdescformat</code>	264, 320	<code>\glsnoidxstripaccents</code>	19
<code>\glsinlinesubnameformat</code>	264, 320	<code>\glsnomakeindexwarning</code>	162
<code>\glsinlinesubseparator</code>	264, 320	<code>\glsnonextpages</code>	62, 184
<code>\glsinsert</code> 94–101, 119–124, 137–144, 224, 232, 236, 238, 239, 241, 341–353, 358, 359		<code>\glsnopostdotfalse</code>	9
<code>\glskeylisttok</code>	216, 230, 231, 233– 235, 237, 239, 240, 242–244, 246, 363–368	<code>\glsnoxindywarning</code> ..	35, 42–44, 46–48, 156
<code>\glslabel</code>	77, 94–101, 106–108, 138–144, 218–220, 223, 224, 226, 232, 236, 238, 241, 341–350, 354, 355, 358–360	<code>\glsnumberformat</code>	152
<code>\glslabeltok</code>	216, 230–237, 239–241, 243, 244, 246, 364–367	<code>\glsnumberlistloop</code>	172
<code>\glslink</code>	216, 217, 224–227, 232, 359, 361	<code>\glsnumbersgroupname</code>	28, 34, 206
<code>\glslink options</code>		<code>\glsnumlistlastsep</code>	152, 171
counter	104, 118, 253	<code>\glsnumlistparser</code>	153, 168
format	104, 118, 208	<code>\glsnumlistsep</code>	152, 171
hyper	104, 107, 118	<code>\glsopenbrace</code>	46, 159, 160, 317, 318
local	104	<code>\glsorder</code>	25, 166, 167, 190
<code>\glslinkcheckfirsthyperhook</code>	107	<code>\glsorg@endtheglossary</code>	5
<code>\glslinkpostsetkeys</code>	108	<code>\glsorg@PrintChanges</code>	5
<code>\glslinkvar</code>	105	<code>\glsorg@theglossary</code>	5
<code>\glslistdottedwidth</code>	268, 321	<code>\glspagelistwidth</code> ...	271, 274, 275, 277, 278, 281–285, 293, 294, 296, 297, 300–303
<code>\glslistgroupheaderfmt</code>	267, 268	<code>\glspatchLToutput</code>	275–278
<code>\glslistnavigationitem</code>	267, 268	<code>\glspenaltygroupskip</code>	276, 277
<code>\glslocalreset</code>	87	<code>\glspercentchar</code>	68, 69, 158, 160
<code>\glslocalunset</code>	88, 119–124	<code>\Glspl</code>	92, 230
<code>\glslongaccessdisplay</code>	350, 352–364	<code>\glspl</code>	92, 230
<code>\glslongkey</code>	368	<code>\glspluralaccessdisplay</code>	341, 342, 345, 346
<code>\glslongpluralaccessdisplay</code>	350, 355, 356, 358–362, 364	<code>\glspluralsuffix</code>	31, 80, 81, 219, 220, 355, 356, 360–362
<code>\glslongpluralkey</code>	368	<code>\glspostdescription</code>	34, 265–267, 269, 270, 280, 291, 292, 298, 305–308, 310, 311, 320–323, 325–329, 331
		<code>\glspostinline</code>	263

<code>\glspostlinkhook</code>	<code>\glstildechar</code>
..... 106, 119–124, 137–144, 351–353	42, 158, 159
<code>\glsprestandardsort</code>	<code>\glstranslatefalse</code>
11	22, 23
<code>\glreset</code>	<code>\glstranslatetrue</code>
87	23
<code>\glresetentrycounter</code>	<code>\glstreechildpredesc</code>
197, 200	305, 307
<code>\glresetentrylist</code> .	<code>\glstreegroupheaderfmt</code>
159, 192, 198, 199, 317 286–290, 305–309, 312
<code>\glresetsubentrycounter</code>	<code>\glstreeindent</code> ..
..... 197, 198, 201, 264, 320	306, 308, 310, 311, 326–328
<code>\glssanitizesortfalse</code>	<code>\glstreeitem</code>
21	286, 304
<code>\glssanitizesorttrue</code>	<code>\glstreenamebox</code>
21	310, 311
<code>\glssavenumberlistfalse</code>	<code>\glstreenamefmt</code>
8	304–306, 308–311
<code>\glssavewritesfalse</code>	<code>\glstreenavigationfmt</code>
27 286–290, 306, 307, 309, 312
<code>\glseeformat</code>	<code>\glstreepredesc</code>
158, 170, 172, 317	305, 306, 308
<code>\glseeitem</code>	<code>\glstreesubitem</code>
181	286, 304
<code>\glseeitemformat</code>	<code>\glstreesubsubitem</code>
181	286, 304
<code>\glseeelastsep</code>	<code>\glstype</code> .
181	106, 107, 119–124, 137–144, 351–353
<code>\glseeelist</code>	<code>\glsucmarkfalse</code>
181	10
<code>\glseeesep</code>	<code>\glsucmarktrue</code>
181	10
<code>\glsetexpandfield</code>	<code>\glunset</code>
18, 20–22	87, 89, 90, 119–124
<code>\glsetnoexpandfield</code>	<code>\glsupacrpluralsuffix</code>
18, 20, 21 220, 221, 227, 234, 238, 240, 243
<code>\GlsSetQuote</code>	<code>\GlsUseAcrEntryDisplayStyle</code> ...
77, 160	217, 220–
<code>\glsettoctitle</code>	223, 225, 227, 228, 356, 357, 360, 362, 363
33, 183	<code>\GlsUseAcrStyleDefs</code>
<code>\glsshortaccessdisplay</code>	217, 220–
..... 348–351, 354–357, 359–364	223, 225, 227, 228, 356, 357, 360, 362, 363
<code>\glsshortkey</code>	<code>\glswrallowprimitivemodstrue</code>
368	177
<code>\glsshortpluralaccessdisplay</code>	<code>\glswrite</code> ...
..... 348, 350, 355, 359–362, 364	156–161, 167, 173, 174, 315–319
<code>\glsshortpluralkey</code>	<code>\glswritedefhook</code>
368	69
<code>\glsshorttok</code>	<code>\glswriteentry</code>
216, 230–235, 237, 239–244, 246, 364–368	175
<code>\glssortnumberfmt</code>	<code>\glswritefiles</code>
12, 13	27, 173
<code>\glsspace</code>	<code>\glsxindyfalse</code>
213	25
<code>\glsstepeentry</code>	<code>\glsxindytrue</code>
197, 201	26
<code>\glssubentry</code>	
197, 198, 201	
<code>\glssubentrycounterfalse</code>	
10	
<code>\glssubentrycounterlabel</code> ...	
197, 198, 201	
<code>\glssubentryitem</code>	
. 197, 198, 264, 266–268, 270, 271, 273,	
280, 282, 283, 291, 293, 295, 298, 300,	
302, 305, 306, 308, 310, 320–327, 329–332	
<code>\glssymbolaccessdisplay</code>	
343–345, 354	
<code>\glssymbolpluralaccessdisplay</code> .	
341, 342	
<code>\glssymbolsgroupname</code>	
27, 34, 206	
<code>\glstarget</code>	
204, 265–	
273, 280, 282, 283, 291–293, 295, 298,	
300, 302, 305, 306, 308, 310, 311, 320–332	
<code>\glstextaccessdisplay</code> ..	
343, 344, 346, 347	
<code>\glstextformat</code>	
106, 108	
<code>\glstextup</code>	
31, 362	

H

<code>\H</code>	20
<code>\hangindent</code>	289, 290, 304, 306, 308, 310–312, 325–328
<code>\hbox</code>	268, 321
<code>\hfill</code>	268, 321
<code>\hline</code>	270–274,
281–285, 292–294, 296, 297, 299–301, 303	
<code>\hsize</code>	268, 269, 280, 291, 298
<code>\hspace</code>	304
<code>\hss</code>	268, 321
<code>\ht</code>	279
<code>\hyperdef</code>	30
<code>\hyperlink</code>	104, 117, 209
<code>hyperref</code> package	179, 182, 208, 253
<code>\hypertarget</code>	117

I

<code>\IeC</code>	19	<code>\ifglshaschildren</code>	264, 320
<code>\if</code>	110, 178, 179, 314	<code>\ifglshasdesc</code>	264
<code>\if@endfor</code>	262	<code>\ifglshaslong</code>	91–93, 146, 218–220, 223, 226, 238, 354, 355, 358, 360
<code>\if@gl@debug</code>	5, 17, 175	<code>\ifglshasparent</code>	187, 193, 309
<code>\if@gl@docloaded</code>	4, 13, 174	<code>\ifglshasprefix</code>	258
<code>\if@gl@isacronymlist</code>	106	<code>\ifglshasprefixfirst</code>	258
<code>\if@openright</code>	39	<code>\ifglshasprefixfirstplural</code>	258
<code>\ifbool</code>	14, 24, 27, 51, 94–96	<code>\ifglshasprefixplural</code>	258
<code>\ifboolexpr</code>	32, 56, 205	<code>\ifglshassymbol</code>	232, 236, 241, 305–308, 310, 311
<code>\ifcase</code>	6, 7, 23, 62, 196, 305, 325	<code>\ifglshyperfirst</code>	106
<code>\ifcsdef</code>	22, 33, 39, 65, 66, 72–76, 103, 175, 186, 187, 190, 191, 207, 218	<code>\ifgl@indexonlyfirst</code>	176
<code>\ifcsempy</code>	53, 256	<code>\ifgl@nogroupskip</code>	266, 270, 271, 273, 275–277, 280, 282, 284, 292, 293, 295, 298, 300, 302, 305, 307, 308, 311
<code>\ifcsequal</code>	53	<code>\ifgl@snonumberlist</code>	198
<code>\ifcsstrequal</code>	76	<code>\ifgl@snopostdot</code>	9
<code>\ifcsstring</code>	75	<code>\ifgl@snnumberline</code>	40
<code>\ifcsundef</code>	6, 26, 29, 30, 32, 36–39, 50, 51, 58, 59, 61, 78, 81, 88, 89, 104, 108, 109, 117, 118, 166, 173, 182, 185, 187, 195, 196, 201, 205–208, 211, 217, 262, 319	<code>\ifgl@ssanitizesort</code>	18, 21
<code>\ifdef</code>	54, 55, 63, 68, 104, 145, 148, 171, 172, 212, 304	<code>\ifgl@savenumberlist</code>	65, 168, 182
<code>\ifdefempty</code>	16, 39, 50, 53–55, 59, 94, 97, 100, 169, 192, 193, 216, 217, 223, 232, 236, 238, 240, 241, 341, 345, 348, 358	<code>\ifgl@savewrites</code>	27, 165, 175
<code>\ifdefequal</code>	52–55, 65–67, 70, 79, 83, 193	<code>\ifgl@subentrycounter</code>	197, 199–201
<code>\ifdefstrequal</code>	76	<code>\ifgl@stoc</code>	40
<code>\ifdefstring</code>	32, 56, 166, 168, 189, 190, 193, 194	<code>\ifgl@stranslate</code>	32
<code>\ifdefvoid</code>	19, 82, 193, 194	<code>\ifgl@sucmark</code>	38
<code>\ifdim</code>	220, 279, 309	<code>\ifgl@sused</code>	90, 94–100, 106, 154, 176, 232, 236, 238, 241, 256–260, 341–348
<code>\iffalse</code>	82, 87	<code>\ifgl@swrallowprimitivemods</code>	177
<code>\IfFileExists</code>	9, 22, 23, 185	<code>\ifgl@xindy</code>	34, 35, 41–44, 46–49, 58, 83, 84, 111, 155, 156, 162, 166, 178, 180, 185, 313–315
<code>\ifglossaryexists</code>	37, 48, 52, 165, 166	<code>\ifignoredglossary</code>	79, 82, 175
<code>\ifgl@sanitize@description</code>	20	<code>\ifin@</code>	28
<code>\ifgl@sanitize@name</code>	20	<code>\ifinlistcs</code>	191, 195
<code>\ifgl@sanitize@symbol</code>	20	<code>\ifKV@gl@link@hyper</code>	107, 108
<code>\ifgl@xindy@gl@numbers</code>	49	<code>\ifKV@gl@link@local</code>	119–124
<code>\ifgl@sacrdescription</code>	245	<code>\ifmeasuring@</code>	85
<code>\ifgl@sacrdua</code>	234, 240, 243, 245	<code>\ifnum</code>	11, 89, 90, 192, 278, 279, 306, 308, 310, 311, 326, 327
<code>\ifgl@sacrfootnote</code>	106, 245	<code>\ifstrempy</code>	320
<code>\ifgl@sacrronym</code>	14	<code>\ifstrequal</code>	205
<code>\ifgl@sacrshortcuts</code>	29, 230	<code>\ifthenelse</code>	21, 29, 39, 108, 168, 206, 245, 261
<code>\ifgl@sacrsmalldcaps</code>	234, 235, 238, 240, 243	<code>\IfTrackedLanguage</code>	162
<code>\ifgl@sacrsmaller</code>	234, 235, 238, 240	<code>\IfTrackedLanguageFileExists</code>	33, 371, 372
<code>\ifgl@sautomake</code>	26, 169	<code>\iftrue</code>	82, 86
<code>\ifgl@sdscsuppressed</code>	264	<code>\ifundef</code>	57, 68, 78, 156, 160, 167, 197
<code>\ifgl@sentrycounter</code>	197, 199–201	<code>\ifvmode</code>	154
<code>\ifgl@sentryexists</code>	51, 52, 68, 77, 79	<code>\ifvoid</code>	279

<code>\ifx</code>	11–13, 15, 28, 30, 41, 42, 44, 46, 48, 49, 79–82, 84, 109, 112– 117, 146, 156, 159, 161, 162, 164, 174, 177–181, 183, 184, 197, 199, 206–209, 231, 233, 235, 237, 239, 240, 242, 244, 246, 247, 315–317, 319, 320, 325–328, 339	<code>\longnewglossaryentry</code>	77
<code>\immediate</code>	68, 69, 90, 166, 174, 185, 186	<code>longtable</code> package	269, 275, 279
<code>\in@</code>	28	<code>\LT@end@open</code>	279
<code>\index</code>	175	<code>\LT@err</code>	279
<code>\indexname</code>	28	<code>\LT@foot</code>	279
<code>\indexspace</code> .	266, 286–290, 305–309, 311, 312	<code>\LT@head</code>	279
<code>\input</code>	32, 93	<code>\LT@lastfoot</code>	279
<code>\inputencodingname</code>	26	<code>\LT@output</code>	278, 279
<code>\InputIfFileExists</code>	68	M	
<code>\istfilename</code>	34, 156, 160, 167, 315, 318	<code>\makeatletter</code>	68, 185
<code>\item</code>	266–269, 286, 304–306, 320, 321, 325	<code>\makeatother</code>	68
J		<code>\makebox</code>	268, 309–311, 321, 327, 328
<code>\jobname</code> 35, 68, 156, 160, 166, 167, 185, 315, 318		<code>makeglossaries</code> ..	25, 35, 48, 57, 162, 167, 185
K		<code>\makeglossaries</code>	6, 26, 30, 62, 169–171, 173, 186
<code>\key@ifundefined</code>	70, 71	<code>\makeglossary</code>	165, 168
<code>\KV@glslink@hyperfalse</code> .	104, 106, 107, 118	<code>makeindex</code>	373
<code>\KV@glslink@hypertrue</code>	104, 118	<code>makeindex</code>	10, 25, 26, 31, 34–36, 40, 56–58, 60, 84, 109, 113, 155, 158, 160, 162, 165, 174, 178, 179, 204, 205, 314, 315
L		<code>delim_n</code>	36
<code>\L</code>	20	<code>delim_r</code>	36
<code>\l</code>	20	<code>page_compositor</code>	35
<code>\label</code>	7, 196, 197, 200	<code>special characters</code>	111, 112, 155
<code>\language</code>	25	<code>\makenoidxglossaries</code> ..	6, 62, 168, 172, 173
<code>\leaders</code>	268, 321	<code>\MakeTextUppercase</code>	4
<code>\leavevmode</code>	76, 107	<code>\MakeUppercase</code>	342, 344, 351, 353
<code>\let</code> ..	5, 9, 11–14, 19, 20, 22, 23, 27–30, 32, 34, 43, 54–56, 65, 67, 76–82, 85–88, 90, 93, 105–108, 110, 111, 117–124, 137– 144, 146, 152, 153, 160, 161, 165–170, 172, 175–177, 181, 183–185, 193, 195, 196, 199, 203, 216, 228–231, 233, 235– 239, 241, 242, 244, 254, 261–263, 278, 286, 304, 319, 336, 351–353, 364–368, 371	<code>\markboth</code>	38
<code>\letcs</code>	52– 55, 68, 71, 72, 75, 80, 81, 145, 146, 166, 171, 172, 187–189, 193, 202, 205, 206, 309	<code>\mbox</code>	154, 267, 289, 290, 310, 321
<code>link text</code>	93	<code>memoir class</code>	175
<code>\listcsadd</code>	191	<code>\memUchead</code>	38
<code>\listcsgadd</code>	195	<code>\MessageBreak</code>	33, 56, 183, 371, 372
<code>\listcsxadd</code>	186, 187	<code>mfirstuc package</code>	1
<code>\listead</code>	191	<code>\mfirstucMakeUppercase</code>	4, 38, 73, 95, 97–101, 125–138, 140, 142, 144, 216, 217, 224–227, 236, 241, 259, 260, 346–350, 358, 359, 361, 362
<code>\loadglsentries</code>	93	<code>\midrule</code>	275–277
<code>\long</code>	76, 210	<code>\month</code>	156, 160, 315, 318
		<code>multicol package</code>	285
		N	
		<code>\n</code>	160, 161, 318
		<code>\NeedsTeXFormat</code> ...	4, 254, 313, 319, 333, 371
		<code>\new@glossaryentry</code>	67, 171
		<code>\new@ifnextchar</code>	56, 72, 73, 91, 92, 119–123, 125–144, 212–215, 256–259
		<code>\newacronym</code>	211, 216, 231, 233, 235, 237, 239, 242, 244, 246

<code>\newacronymhook</code>	216,	<code>symbol</code>	60, 61,
	231, 233, 235, 237, 240, 243, 244, 247, 363		130, 233–235, 237, 242, 272, 294, 334–336
<code>\newacronymstyle</code>	218–223, 225–228	<code>symbolaccess</code>	338, 340
<code>\newcommand</code>		<code>symbolplural</code>	131, 335
... 5–19, 21, 22, 24–32, 34–44, 46–77,		<code>symbolpluralaccess</code>	338, 340
83–88, 90–94, 97, 100, 102, 103, 105–		<code>text</code>	60, 61, 118, 124, 147, 233, 237, 334
108, 110, 111, 117–156, 162, 164–167,		<code>textaccess</code>	337, 339
169, 172–183, 185–191, 193–195, 198–		<code>type</code>	13, 61, 93, 149
208, 210–218, 220, 228, 230–239, 241–		<code>user1</code>	132, 149, 335
253, 255–259, 261–266, 278, 279, 285,		<code>user2</code>	133, 149
303, 304, 309, 319, 337–339, 354, 368–370		<code>user3</code>	133, 149
<code>\newcount</code>	12, 65	<code>user4</code>	134, 150
<code>\newcounter</code>	197, 199	<code>user5</code>	135, 150
<code>\newenvironment</code>	201	<code>user6</code>	136, 150, 335
<code>\newglossary</code>	13, 14, 27, 28, 58, 168	<code>\newglossarystyle</code>	
<code>\newglossaryentry</code> 28, 64, 67, 88, 216, 230,		263, 266–278, 280–308, 310, 312
232, 234, 236, 239, 241, 243, 246, 364–367		<code>\newif</code>	4, 15, 22, 25, 177
<code>\newglossaryentry options</code>		<code>\newlength</code> ..	117, 268, 269, 280, 291, 298, 307
access	336, 337	<code>\newrobustcmd</code>	
counter	61	... 67, 68, 72, 73, 91, 92, 106, 118–123,	
description		125–151, 153, 154, 212–215, 255–259, 309	
. 24, 60, 64, 67, 77, 129, 146, 212, 239, 335		<code>\newterm</code>	28
<code>descriptionaccess</code>	338, 340	<code>\newtoks</code>	111, 165, 215
<code>descriptionplural</code>	129, 335	<code>\newwrite</code>	68, 156, 160, 165, 167
<code>descriptionpluralaccess</code>	338, 340	<code>ngerman package</code>	162
<code>first</code> . 61, 80, 118, 125, 148, 237, 242, 243, 334		<code>\noalign</code>	278
<code>firstaccess</code>	338, 340	<code>\nobreak</code>	267, 279, 321
<code>firstplural</code>	61, 127, 148, 335	<code>\noexpand</code> .	15, 30, 41, 43, 81, 82, 103, 108–
<code>firstpluralaccess</code>	338, 340	111, 116, 117, 152, 162–164, 166–168,	
<code>format</code>	156	177, 178, 182, 185, 186, 188, 189, 202,	
<code>long</code>	100, 151, 335	203, 211, 216, 230, 232–234, 236, 237,	
<code>longaccess</code>	339, 340	239, 241–244, 246, 313, 333, 334, 364–368	
<code>longplural</code>	151, 335	<code>\nohyperpage</code>	208
<code>longpluralaccess</code>	339, 340	<code>\noindent</code>	204, 286–290, 307–309
<code>name</code>	60, 64, 67, 77, 128, 145, 181, 334	<code>\noist</code>	318, 319
<code>nonumberlist</code>	62, 63	<code>\nopostdesc</code>	28, 34, 76, 184, 320
<code>parent</code>	62, 67	<code>\normalbaselineskip</code>	278
<code>plural</code>	61, 80, 126, 334	<code>\nr</code>	6, 7, 23, 62, 196
<code>pluralaccess</code>	338, 339	<code>\ns@ACRfull</code>	213
<code>prefix</code>	254	<code>\ns@Acrfull</code>	213
<code>prefixfirst</code>	254	<code>\ns@acrfull</code>	212
<code>prefixfirstplural</code>	255	<code>\ns@ACRfullpl</code>	215
<code>prefixplural</code>	255	<code>\ns@Acrfullpl</code>	214
<code>see</code>	5, 8, 62, 67, 168, 170	<code>\ns@acrfullpl</code>	214
<code>short</code>	100, 150, 335	<code>\ns@ACRlong</code>	142
<code>shortaccess</code>	338, 340	<code>\ns@Acrlong</code>	141
<code>shortplural</code>	150, 335	<code>\ns@acrlong</code>	140
<code>shortpluralaccess</code>	338, 340	<code>\ns@ACRlongpl</code>	144
<code>sort</code>	60, 149, 204, 205	<code>\ns@Acrlongpl</code>	143

<code>\ns@acrlongpl</code>	142	<code>makeindex</code>	158, 253
<code>\ns@ACRshort</code>	138	<code>nogroupskip</code>	270, 271, 273, 275–277, 280, 282, 284, 292, 293, 295, 298, 300, 302
<code>\ns@Acrshort</code>	137	<code>nolist</code>	247
<code>\ns@acrshort</code>	137	<code>nolong</code>	247, 269
<code>\ns@ACRshortpl</code>	140	<code>nomain</code>	13
<code>\ns@Acrshortpl</code>	139	<code>nonumberlist</code>	8
<code>\ns@acrshortpl</code>	139	<code>nosuper</code>	247
<code>\ns@newglossary</code>	57	<code>notree</code>	247
<code>\null</code>	110–117, 162–164, 185	<code>nowarn</code>	5
<code>\number</code>	11, 80, 90, 176, 177, 203, 334	<code>numberline</code>	6
<code>\numberline</code>	40	<code>sanitize</code>	20, 60, 145, 146
<code>\numexpr</code>	90	<code>sanitizesort</code>	17
O		<code>savewrites</code>	27, 377
<code>\O</code>	20	false	165
<code>\o</code>	20	true	167, 173
<code>\OE</code>	20	<code>section</code>	6, 38
<code>\oe</code>	20	<code>sort</code>	
<code>\openout</code>	68, 156, 160, 166, 315, 318	def	10, 11
<code>\OR</code>	245	standard	10
<code>\or</code>	6, 7, 23, 196, 305, 325	use	10, 11
<code>\org@glossaryentrynumbers</code>	184, 199	<code>style</code>	7, 247
<code>\org@glossarytitle</code>	183, 184	<code>subentrycounter</code>	197, 199
<code>\org@glspostdescription</code>	34	<code>toc</code>	6
<code>\org@ifKV@glslink@hyper</code>	107, 108	true	6
<code>\orgAlph</code>	177, 178	<code>translate</code>	23
<code>\orgalph</code>	177, 178	false	22
<code>\orgarabic</code>	177, 178	<code>translator</code>	22
<code>\orgnumber</code>	177	<code>xindy</code>	25, 26, 158, 253
<code>\orgRoman</code>	177, 178	<code>\PackageError</code>	5, 6, 26, 30, 42, 48, 51, 52, 56, 62, 64, 65, 71– 76, 78–80, 88, 104, 145, 164, 165, 168, 171–173, 189, 190, 192, 196, 198, 206– 208, 217, 218, 234, 235, 240, 243, 319, 341
<code>\orgromannumeral</code>	177, 178	<code>\PackageInfo</code>	5, 166, 175
<code>\orgthe</code>	177	<code>\PackageWarning</code>	5, 17, 333
<code>\outputpenalty</code>	278, 279	<code>\PackageWarningNoLine</code> ..	5, 17, 33, 371, 372
P		<code>\pagegoal</code>	279
<code>\p@</code>	265, 285, 303, 304	<code>\pagelistname</code>	33, 272– 274, 276, 277, 283–285, 294–297, 301–303
<code>\p@gl@hyp@opt</code>	105	<code>\par</code>	34, 204, 265, 267, 285, 287–290, 303–312, 321, 326–328
package options:		<code>\parindent</code>	285–290, 304, 306–308, 310–312, 326–328
acronym	13, 14, 30, 182, 211	<code>\parskip</code>	285–289, 304, 306, 307
true	14	<code>\PassOptionsToPackage</code>	254, 333
counter	16	<code>\penalty</code>	278
description	237, 238	<code>\phantomsection</code>	39
dua	236–238	<code>polyglossia package</code>	22, 32
entrycounter	197, 199		
true	10		
footnote	119–124, 233, 236, 237, 239		
hyperfirst			
false	119–124		
indexonlyfirst	380		

<code>\SetFootnoteAcronymStyle</code>	245	<code>\tabularnewline</code>	269–
<code>\SetGenericNewAcronym</code>	217		277, 280–285, 291–303, 323, 324, 329, 330
<code>\setglossarystyle</code> 183, 207, 247, 266–278,		<code>\texorpdfstring</code>	148
281–290, 292–297, 299–303, 305–308, 312		<code>\textbar</code>	263
<code>\setglossentrycompatibility</code> ...	196, 207	<code>\textbf</code>	204, 210, 303, 325–328
<code>\setkeys</code> .. 22, 26, 29, 38, 78, 108, 154, 184,		textcase package	4
216, 231, 233, 235, 237, 240, 242, 244, 246		<code>\textit</code>	210
<code>\setlength</code>	268, 269, 280, 285–	<code>\textmd</code>	210
289, 291, 298, 304, 306, 307, 311, 327, 328		<code>\textrm</code>	210
<code>\SetSmallAcronymDisplayStyle</code> ..	242, 243	<code>\textsc</code>	
<code>\SetSmallAcronymStyle</code>	245		210, 220, 221, 227, 234, 238, 240, 243, 362
<code>\settoheight</code>	117	<code>\textsf</code>	210
<code>\settowidth</code>	220, 309–311, 327	<code>\textsl</code>	210
<code>\sfcode</code>	9	<code>\textsmaller</code> 221, 227, 234, 238, 240, 243, 362	
<code>\show</code>	247–253, 369, 370	<code>\texttt</code>	210
<code>\SmallNewAcronymDef</code>	243	<code>\textulc</code>	212
<code>\space</code>	6, 26, 30, 42, 46,	<code>\textup</code>	210, 212
49, 62, 64, 88, 91, 92, 102, 103, 105, 152,		<code>\th</code>	20
157–160, 164, 166–168, 170, 172, 173,		<code>\the</code>	30, 33,
181, 183, 186, 197, 198, 200, 201, 207,			42, 47, 49, 57, 112–117, 156, 160, 162,
213, 218–228, 236, 240, 241, 263, 265–			164, 174, 175, 177, 182, 193, 202, 203,
267, 269, 270, 280, 291, 292, 298, 304–			209, 216–220, 224, 226, 230, 232–234,
308, 310, 311, 313, 314, 316–318, 320–			236, 237, 239, 241–244, 246, 313, 315,
323, 325–329, 331, 350, 354–357, 359–364			318, 333, 334, 354, 355, 359, 360, 363–368
<code>\spacefactor</code>	9	<code>\the@numberlist</code>	152, 153
<code>\SS</code>	20	<code>\theglossary</code>	5
<code>\ss</code>	20	<code>\theglossaryentry</code>	197, 199, 200
<code>\string</code>	6,	<code>\theglossarysubentry</code>	198, 199, 201
17, 26, 30, 40–42, 44–47, 49, 56, 57, 62,		<code>\theglentrycounter</code> 108, 109, 178, 314, 315	
64, 68, 69, 72, 73, 83, 84, 88, 90–92, 103,		<code>\theH</code>	180
105, 110, 112–114, 116, 152, 155–162,		<code>\theHglossaryentry</code>	197, 199
164, 165, 167–173, 179, 180, 183, 185,		<code>\theHglossarysubentry</code>	198, 199
186, 189, 190, 198, 204, 207, 261, 313–319		<code>\theHglentrycounter</code>	109, 178
<code>\strut</code>	204, 266–268, 270,	<code>\thesection</code>	30
271, 273, 280, 282, 283, 292, 293, 295,		<code>\this@dialect</code>	33, 371, 372
298, 300, 302, 308, 320–324, 326, 329–332		<code>\toks@</code> .. 30, 32, 33, 42, 47, 49, 57, 112–117,	
<code>\subglossentry</code>			162, 164, 182, 202, 203, 209, 313, 333, 334
..... 83, 184, 193, 203, 204, 264, 266–		<code>\toprule</code>	275, 276
268, 270, 271, 273, 280, 282, 283, 291,		tracklang package	32, 371
293, 295, 298, 300, 302, 305, 306, 308, 310		<code>\trans@languages</code>	32
<code>\subitem</code>	286, 304, 305, 325	<code>\translate</code>	33, 34
<code>\subsubitem</code>	286, 304, 305, 325	<code>\translatelet</code>	13, 14, 27, 28
supertabular package	9, 247, 291, 297	translator package .	13, 14, 22, 27, 28, 32, 33, 182
<code>\symbolname</code>	33,		
273, 274, 276, 284, 285, 295–297, 302, 303			
T			
<code>\t</code>	19	U	
<code>\tablehead</code>	291–303	<code>\u</code>	19
<code>\tabletail</code>	291–303	<code>\uccode</code>	192
		<code>\undef</code>	63, 182
		<code>\unskip</code>	76, 268, 321

\unvbox	279	X	
\usedictionary	32	\x	209
\usepackage	189, 190	\xatlevel@	109
V		\xcapitalisewords	148
\v	20	\xdef	73, 79, 80, 82, 184, 262
\val	6, 7, 23, 62, 196	\xglaccsupp	339
\vbox	279	\xifinlistcs	186–188, 191
\vsize	279	xindy	373
\vskip	265, 278, 285, 303	xindy 10, 25, 26, 34, 35, 40, 44, 46–49, 84, 116,	
\vss	279	155–157, 174, 178, 179, 185, 204, 253, 314	
W		\xmakefirstuc	94–96, 102, 145, 146, 255
\warn@nomakeglossaries	168–170	\xspace	211
\warn@noprintglossary	168–170, 185	xspace package	4, 211
\write	68, 69, 90, 156–	Y	
161, 166, 167, 170, 174, 185, 186, 315–319		\year	156, 160, 315, 318
\writeist	165, 319	Z	
		\z@	279