

Documented Code For glossaries v4.19

Nicola L.C. Talbot

Dickimaw Books

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

2015-11-22

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 v4.19: L^AT_EX2e 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	84
1.9 Keeping Track of How Many Times an Entry Has Been Unset	87
1.10 Loading files containing glossary entries	92
1.11 Using glossary entries in the text	93
1.11.1 Links to glossary entries	103
1.11.2 Displaying entry details without adding information to the glossary	145
1.12 Adding an entry to the glossary without generating text	154
1.13 Creating associated files	155
1.14 Writing information to associated files	170
1.15 Glossary Entry Cross-References	177
1.16 Displaying the glossary	179
1.17 Acronyms	208
1.18 Predefined acronym styles	212
1.19 Predefined Glossary Styles	244
1.20 Debugging Commands	245
1.21 Compatibility with version 2.07 and below	250
2 Prefix Support (glossaries-prefix Code)	251
3 Glossary Styles	257
3.1 Glossary hyper-navigation definitions (glossary-hypernav package)	257
3.2 In-line Style (glossary-inline.sty)	260
3.3 List Style (glossary-list.sty)	262
3.4 Glossary Styles using longtable (the glossary-long package)	265
3.5 Glossary Styles using longtable (the glossary-longragged package)	271
3.6 Glossary Styles using multicol (glossary-mcols.sty)	277
3.7 Glossary Styles using supertabular environment (glossary-super package)	281
3.8 Glossary Styles using supertabular environment (glossary-superragged package)	287
3.9 Tree Styles (glossary-tree.sty)	293
4 glossaries-compatible-207	301

5 Accessibility Support (glossaries-accsupp Code)	321
5.1 Defining Replacement Text	322
5.2 Accessing Replacement Text	325
5.3 Displaying the Glossary	341
5.4 Acronyms	342
5.5 Debugging Commands	356
6 Multi-Lingual Support	358
6.1 Polyglossia Captions	359
Glossary	359
Change History	359
Index	384

1 Main Package Code

1.1 Package Definition

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

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries}[2015/11/22 v4.19 (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.

```
\if@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.

\glsorg@theglossary

```
21 \let\glsorg@theglossary\theglossary
```

sorg@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

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.

```
32 \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.

```
33 \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.

```

34 \ifcsundef{chapter}%
35   {\newcommand*{\@@glossarysec}{section}}%
36   {\newcommand*{\@@glossarysec}{chapter}}

```

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

```

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

```

Determine whether or not to use numbered sections.

\@@glossarysecstar

```

40 \newcommand*{\@@glossarysecstar}{*}

```

\@@glossaryseclabel

```

41 \newcommand*{\@@glossaryseclabel}{}

```

\glsautoprefix Prefix to add before label if automatically generated:

```

42 \newcommand*{\glsautoprefix}{}

```

numberedsection

```

43 \define@choicekey{glossaries.sty}{numberedsection}[\val\nr]{%
44 false,nolabel,autolabel,nameref}[nolabel]{%
45   \ifcase\nr\relax
46     \renewcommand*{\@@glossarysecstar}{*}%
47     \renewcommand*{\@@glossaryseclabel}{}%
48   \or
49     \renewcommand*{\@@glossarysecstar}{}%
50     \renewcommand*{\@@glossaryseclabel}{}%
51   \or
52     \renewcommand*{\@@glossarysecstar}{}%
53     \renewcommand*{\@@glossaryseclabel}{%
54       \label{\glsautoprefix@glo@type}}%
55   \or
56     \renewcommand*{\@@glossarysecstar}{*}%
57     \renewcommand*{\@@glossaryseclabel}{%
58       \protected@edef\@currentlabelname{\glossarytoctitle}%
59       \label{\glsautoprefix@glo@type}}%
60   \fi
61 }

```

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 [subsection 1.19](#).)

ssary@default@style

```
62 \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 [subsection 1.19](#).

```
63 \define@key{glossaries.sty}{style}{%
64   \renewcommand*{\@glossary@default@style}{#1}%
65 }
```

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.

\@gls@declareoption

```
66 \newcommand*{\@gls@declareoption}[2]{%
67   \DeclareOptionX{#1}{#2}%
68   \DeclareOption{#1}{#2}%
69 }
```

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”:

lossaryentrynumbers

```
70 \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 ignores its argument).

```
71 \@gls@declareoption{nonumberlist}{%
72   \renewcommand*{\glossaryentrynumbers}[1]{\gls@save@numberlist{#1}}%
73 }
```

savenumberlist Provide means to store the number list for entries.

```
74 \define@boolkey{glossaries.sty}[gls]{savenumberlist}[true]{}
75 \glssavenumberlistfalse
```

o@seeautonumberlist

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

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

```

77 \@gls@declareoption{seeautonumberlist}{%
78   \renewcommand*{\@glo@seeautonumberlist}{%
79     \def\@glo@prefix{\glsnextpages}%
80   }%
81 }

```

\@gls@loadlong

```

82 \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.

```

83 \@gls@declareoption{nolong}{\renewcommand*{\@gls@loadlong}{}}

```

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

```

84 \IfFileExists{supertabular.sty}{%
85   \newcommand*{\@gls@loadsuper}{\RequirePackage{glossary-super}}}%
86   \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.

```

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

```

\@gls@loadlist

```

88 \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.

```

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

```

\@gls@loadtree

```

90 \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.

```

91 \@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).

```

92 \@gls@declareoption{nostyles}{%
93   \renewcommand*{\@gls@loadlong}{}%
94   \renewcommand*{\@gls@loadsuper}{}%
95   \renewcommand*{\@gls@loadlist}{}%
96   \renewcommand*{\@gls@loadtree}{}%
97   \let\@glossary@default@style\relax
98 }

```


`\glspostdescription` 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.)

```

99 \newcommand*{\glspostdescription}{%
100 \ifglsnopostdot\else.\spacefactor\sfcode'\. \fi
101 }

```

`nopostdot` Boolean option to suppress post description dot

```

102 \define@boolkey{glossaries.sty}[gls]{nopostdot}[true]{}
103 \glsnopostdotfalse

```

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

```

104 \define@boolkey{glossaries.sty}[gls]{nogroupskip}[true]{}
105 \glsnogroupskipfalse

```

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

```

106 \define@boolkey{glossaries.sty}[gls]{ucmark}[true]{}
107 \@ifclassloaded{memoir}
108 {%
109 \glsucmarktrue
110 }%
111 {%
112 \glsucmarkfalse
113 }

```

`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`.

```

114 \define@boolkey{glossaries.sty}[gls]{entrycounter}[true]{}
115 \glsentrycounterfalse

```

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

```

116 \define@key{glossaries.sty}{counterwithin}{%
117 \renewcommand*{\@gls@counterwithin}{#1}%
118 \glsentrycountertrue
119 }

```

`\@gls@counterwithin` The default value is no parent counter:

```

120 \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`.

```

121 \define@boolkey{glossaries.sty}[gls]{subentrycounter}[true]{}
122 \glssubentrycounterfalse

```

`\glo@default@sorttype` Initialise default sort for `\printnoidxglossary`

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

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

```
124 \define@choicekey{glossaries.sty}{sort}{standard,def,use}{%
125   \renewcommand*{\@glo@default@sorttype}{#1}%
126   \csname @gls@setupsort@#1\endcsname
127 }
```

`\glsprestandardsort` `\glsprestandardsort{<sort cs>}{<type>}{<label>}`

Allow user to hook into sort mechanism. The first argument `<sort cs>` is the temporary control sequence containing the sort value before it has been sanitized and had `makeindex/xindy` special characters escaped.

```
128 \newcommand*{\glsprestandardsort}[3]{%
129   \glsdosanitizesort
130 }
```

`@setupsort@standard` Set up the macros for default sorting.

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

Store entry information when it's defined.

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

No count register required for standard sort.

```
133   \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.)

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

```
135     \ifx\@glo@sort\@glsdefaultsort
```

```
136       \let\@glo@sort\@glo@name
```

```
137     \fi
```

```
138     \let\glsdosanitizesort\@gls@sanitizesort
```

```
139     \glsprestandardsort{\@glo@sort}{##1}{##2}%
```

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

```
141   }%
```

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

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

```
143 }
```

Set standard sort as the default:

```
144 \@gls@setupsort@standard
```

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

```

145 \newcommand*\glssortnumberfmt[1]{%
146   \ifnum#1<100000 0\fi
147   \ifnum#1<10000 0\fi
148   \ifnum#1<1000 0\fi
149   \ifnum#1<100 0\fi
150   \ifnum#1<10 0\fi
151   \number#1%
152 }

```

`\@gls@setupsort@def` Set up the macros for order of definition sorting.

```

153 \newcommand*\@gls@setupsort@def{%

```

Store entry information when it's defined.

```

154   \def\do@glo@storeentry{\@glo@storeentry}%

```

Defined count register associated with the glossary.

```

155   \def\@gls@defsortcount##1{%
156     \expandafter\global
157     \expandafter\newcount\csname glossary@##1@sortcount\endcsname
158   }%

```

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

```

159   \def\@gls@defsort##1##2{%
160     \expandafter\global\expandafter
161     \advance\csname glossary@##1@sortcount\endcsname by 1\relax
162     \expandafter\protected@xdef\csname glo@##2@sort\endcsname{%
163       \expandafter\glssortnumberfmt
164       {\csname glossary@##1@sortcount\endcsname}}%
165   }%

```

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

```

166   \def\@gls@setsort##1{%
167 }

```

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

```

168 \newcommand*\@gls@setupsort@use{%

```

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

```

169   \let\do@glo@storeentry\@gobble

```

Defined count register associated with the glossary.

```

170   \def\@gls@defsortcount##1{%
171     \expandafter\global
172     \expandafter\newcount\csname glossary@##1@sortcount\endcsname
173   }%

```

Initialise the sort key to empty.

```

174   \def\@gls@defsort##1##2{%
175     \expandafter\gdef\csname glo@##2@sort\endcsname{}%
176   }%

```

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

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

Get the parent, if one exists

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

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

```
179 \ifx\@glo@parent\@empty
```

```
180 \else
```

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

```
182 \fi
```

Set index information for this entry

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

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

```
185 \ifx\@gls@tmp\@empty
```

```
186 \expandafter\global\expandafter
```

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

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

```
189 \expandafter\glssortnumberfmt
```

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

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

```
192 \fi
```

```
193 }%
```

```
194 }
```

`\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.)

```
195 \newcommand*\glsdefmain{%
```

```
196 \if@gls@docloaded
```

```
197 \newglossary[glg2]{main}{gls2}{glo2}{\glossaryname}%
```

```
198 \else
```

```
199 \newglossary{main}{gls}{glo}{\glossaryname}%
```

```
200 \fi
```

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

```
201 \newcommand*\gls@tr@set@main@toctitle{%
```

```
202 \translatelet{\glossarytoctitle}{Glossary}%
```

```
203 }%
```

```
204 }
```

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 [subsection 1.10](#)).

`\glsdefaulttype`

```
205 \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`

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

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

```
207 \@gls@declareoption{nomain}{%
208   \let\glsdefaulttype\relax
209   \renewcommand*{\glsdefmain}{}%
210 }
```

`acronym` The `acronym` option sets an associated conditional which is used in [subsection 1.17](#) to determine whether or not to define a separate glossary for acronyms.

```
211 \define@boolkey{glossaries.sty}[gls]{acronym}[true]{%
212   \ifglsacronym
213     \renewcommand*{\@gls@do@acronymsdef}{%
214       \DeclareAcronymList{acronym}%
215       \newglossary[alg]{acronym}{acr}{acn}{\acronymname}%
216       \renewcommand*{\acronymtype}{acronym}%
217     }%
218   }%
219   \else
220     \let\@gls@do@acronymsdef\relax
221   \fi
222 }
```

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

```
217   \newcommand*{\gls@tr@set@acronym@toctitle}{%
218     \translatelet{\glossarytoctitle}{Acronyms}%
219   }%
220   }%
221   \else
222     \let\@gls@do@acronymsdef\relax
223   \fi
224 }
```

`\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.

```
225 \AtBeginDocument{%
226   \ifglsacronym
227     \ifbool{glscompatible-3.07}{%
228       {}%
229     }%
230     \providecommand*{\printacronyms}[1][{}]{%
231       \printglossary[type=\acronymtype,#1]}%
232   }
```

```

233 \fi
234 }

```

`@gls@do@acronymsdef` Set default value

```

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

```

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

```

236 \@gls@declareoption{acronyms}{%
237   \glsacronymtrue
238   \renewcommand{\@gls@do@acronymsdef}{%
239     \DeclareAcronymList{acronym}%
240     \newglossary[alg]{acronym}{acr}{acn}{\acronymname}%
241     \renewcommand*{\acronymtype}{acronym}%

```

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

```

242     \newcommand*{\gls@tr@set@acronym@toctitle}{%
243       \translatelet{\glossarytoctitle}{Acronyms}%
244     }%
245   }%
246 }

```

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

```

247 \newcommand*{\@glsacronymlists}{}

```

`\@addtoacronymlists`

```

248 \newcommand*{\@addtoacronymlists}[1]{%
249   \ifx\@glsacronymlists\@empty
250     \protected@xdef\@glsacronymlists{#1}%
251   \else
252     \protected@xdef\@glsacronymlists{\@glsacronymlists,#1}%
253   \fi
254 }

```

`\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.)

```

255 \newcommand*{\DeclareAcronymList}[1]{%
256   \glsIfListOfAcronyms{#1}{\@addtoacronymlists{#1}}%
257 }

```

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

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

```

258 \newcommand{\glsIfListOfAcronyms}[1]{%
259   \edef\@do@gls@islistofacronyms{%
260     \noexpand\@gls@islistofacronyms{#1}\@glsacronymlists}}%
261   \@do@gls@islistofacronyms
262 }

```

Internal command requires label and list to be expanded:

```

263 \newcommand{\@gls@islistofacronyms}[4]{%
264   \def\gls@islistofacronyms##1,#1,##2\end@gls@islistofacronyms{%
265     \def\@before{##1}\def\@after{##2}}%
266   \gls@islistofacronyms,#2,#1,\@nil\end@gls@islistofacronyms
267   \ifx\@after\@nnil

```

Not found

```

268     #4%
269   \else

```

Found

```

270     #3%
271   \fi
272 }

```

`\if@glsisacronymlist` Convenient boolean.

```

273 \newif\if@glsisacronymlist

```

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

```

274 \newcommand*\@gls@checkisacronymlist[1]{%
275   \glsIfListOfAcronyms{#1}%
276   {\@glsisacronymlisttrue}\@glsisacronymlistfalse}%
277 }

```

`\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.)

```

278 \newcommand*\@SetAcronymLists[1]{%
279   \renewcommand*\@glsacronymlists{#1}%
280 }

```

`acronymlists`

```

281 \define@key{glossaries.sty}{acronymlists}{%
282   \DeclareAcronymList{#1}%
283 }

```

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 [subsection 1.6](#)).

`\glscounter`

```

284 \newcommand{\glscounter}{page}

```

```

counter    The counter option changes the default counter. (This just redefines \glscounter.)
285 \define@key{glossaries.sty}{counter}{%
286   \renewcommand*{\glscounter}{#1}%
287 }

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

sDeclareNoHyperList
289 \newcommand*{\GlsDeclareNoHyperList}[1]{%
290   \ifdefempty\@gls@nohyperlist
291     {%
292       \renewcommand*{\@gls@nohyperlist}{#1}%
293     }%
294     {%
295       \appto\@gls@nohyperlist{,#1}%
296     }%
297 }

nohypertypes
298 \define@key{glossaries.sty}{nohypertypes}{%
299   \GlsDeclareNoHyperList{#1}%
300 }

\GlossariesWarning    Prints a warning message.
301 \newcommand*{\GlossariesWarning}[1]{%
302   \PackageWarning{glossaries}{#1}%
303 }

sariesWarningNoLine    Prints a warning message without the line number.
304 \newcommand*{\GlossariesWarningNoLine}[1]{%
305   \PackageWarningNoLine{glossaries}{#1}%
306 }

nowarn    Define package option to suppress warnings
307 \@gls@declareoption{nowarn}{%
308   \renewcommand*{\GlossariesWarning}[1]{}%
309   \renewcommand*{\GlossariesWarningNoLine}[1]{}%
310 }

@warnonglossdefined    Issue a warning if overriding \printglossary
311 \newcommand*{\@gls@warnonglossdefined}{%
312   \GlossariesWarning{Overriding \string\printglossary}%
313 }

rnontheglossdefined    Issue a warning if overriding theglossary
314 \newcommand*{\@gls@warnontheglossdefined}{%
315   \GlossariesWarning{Overriding 'theglossary' environment}%
316 }

```


noredefwarn Suppress warning on redefinition of \printglossary

```

317 \@gls@declareoption{noredefwarn}{%
318   \renewcommand*{\@gls@warnonglossdefined}{}%
319   \renewcommand*{\@gls@warnonthe glossdefined}{}%
320 }
```

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.

\@gls@sanitizedesc

```

321 \newcommand*{\@gls@sanitizedesc}{%
322 }
```

\glssetexpandfield `\glssetexpandfield{<field>}`

Sets field to always expand.

```

323 \newcommand*{\glssetexpandfield}[1]{%
324   \csdef{gls@assign@#1@field}##1##2{%
325     \@gls@expand@field{##1}{#1}{##2}%
326   }%
327 }
```

\glssetnoexpandfield `\glssetnoexpandfield{<field>}`

Sets field to never expand.

```

328 \newcommand*{\glssetnoexpandfield}[1]{%
329   \csdef{gls@assign@#1@field}##1##2{%
330     \@gls@noexpand@field{##1}{#1}{##2}%
331   }%
332 }
```

s@assign@type@field The type must always be expandable.

```

333 \glssetexpandfield{type}
```

s@assign@desc@field The description is not expanded by default:

```

334 \glssetnoexpandfield{desc}
```

gn@descplural@field

```

335 \glssetnoexpandfield{descplural}
```

\@gls@sanitizename

```

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

s@assign@name@field Don't expand name by default.
 337 \glsssetnoexpandfield{name}

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

assign@symbol@field Don't expand symbol by default.
 339 \glsssetnoexpandfield{symbol}

@symbolplural@field
 340 \glsssetnoexpandfield{symbolplural}

Sanitizing stuff:

\@gls@sanitizesort
 341 \newcommand*{\@gls@sanitizesort}{%
 342 \ifglsssanitizesort
 343 \@gls@sanitizesort
 344 \else
 345 \@gls@nosanitizesort
 346 \fi
 347 }

\@@gls@sanitizesort
 348 \newcommand*\@@gls@sanitizesort{%
 349 \@onelevel@sanitize\@glo@sort
 350 }

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

@noidx@sanitizesort Remove braces around first character (if present) before sanitizing.
 352 \newcommand*\@gls@noidx@sanitizesort{%
 353 \ifdefvoid\@glo@sort
 354 {}%
 355 {%
 356 \expandafter\@gls@noidx@sanitizesort\@glo@sort\gls@end@sanitizesort
 357 }%
 358 }
 359 \def\@@gls@noidx@sanitizesort#1#2\gls@end@sanitizesort{%
 360 \def\@glo@sort{#1#2}%
 361 \@onelevel@sanitize\@glo@sort
 362 }

noidx@nosanitizesort
 363 \newcommand*\@@gls@noidx@nosanitizesort{%
 364 \ifdefvoid\@glo@sort
 365 {}%

```

366 {%
367   \expandafter\@@gls@noidx@no@sanitizesort\@glo@sort\gls@end@sanitizesort
368 }%
369 }
370 \def\@@gls@noidx@no@sanitizesort#1#2\gls@end@sanitizesort{%
371   \bgroup
372     \glsnoidxstripaccents
373     \protected@xdef\@@glo@sort{#1#2}%
374   \egroup
375   \let\@glo@sort\@@glo@sort
376 }

```

lsglsnoidxstripaccents

```

377 \newcommand*\glsnoidxstripaccents{%
378   \let\IeC\@firstofone
379   \let\''\@firstofone
380   \let\'\@firstofone
381   \let\~\@firstofone
382   \let\""\@firstofone
383   \let\u\@firstofone
384   \let\t\@firstofone
385   \let\d\@firstofone
386   \let\r\@firstofone
387   \let\=\@firstofone
388   \let\.\@firstofone
389   \let\~\@firstofone
390   \let\v\@firstofone
391   \let\H\@firstofone
392   \let\c\@firstofone
393   \let\b\@firstofone
394   \def\AE{AE}%
395   \def\ae{ae}%
396   \def\OE{OE}%
397   \def\oe{oe}%
398   \def\AA{AA}%
399   \def\aa{aa}%
400   \def\L{L}%
401   \def\l{l}%
402   \def\O{O}%
403   \def\o{o}%
404   \def\SS{SS}%
405   \def\ss{ss}%
406   \def\th{th}%
407 }

```

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.

```

408 \define@boolkey[gls]{sanitize}{description}[true]{%

```

```

409 \GlossariesWarning{sanitize={description} package option deprecated}%
410 \ifgls@sanitize@description
411   \glssetnoexpandfield{desc}%
412   \glssetnoexpandfield{descplural}%
413 \else
414   \glssetexpandfield{desc}%
415   \glssetexpandfield{descplural}%
416 \fi
417 }

418 \define@boolkey[gls]{sanitize}{name}[true]{%
419   \GlossariesWarning{sanitize={name} package option deprecated}%
420   \ifgls@sanitize@name
421     \glssetnoexpandfield{name}%
422   \else
423     \glssetexpandfield{name}%
424   \fi
425 }

426 \define@boolkey[gls]{sanitize}{symbol}[true]{%
427   \GlossariesWarning{sanitize={symbol} package option deprecated}%
428   \ifgls@sanitize@symbol
429     \glssetnoexpandfield{symbol}%
430     \glssetnoexpandfield{symbolplural}%
431   \else
432     \glssetexpandfield{symbol}%
433     \glssetexpandfield{symbolplural}%
434   \fi
435 }

```

sanitizesort

```

436 \define@boolkey{glossaries.sty}[gls]{sanitizesort}[true]{%
437   \ifglssanitizesort
438     \glssetnoexpandfield{sortvalue}%
439     \renewcommand*{\@gls@noidx@setsanitizesort}{%
440       \glssanitizesorttrue
441       \glssetnoexpandfield{sortvalue}%
442     }%
443   \else
444     \glssetexpandfield{sortvalue}%
445     \renewcommand*{\@gls@noidx@setsanitizesort}{%
446       \glssanitizesortfalse
447       \glssetexpandfield{sortvalue}%
448     }%
449   \fi
450 }

```

Default setting:

```

451 \glssanitizesorttrue
452 \glssetnoexpandfield{sortvalue}%

```

idx@setsanitizesort Default behaviour for \makenoidxglossaries is sanitizesort=false.

```
453 \newcommand*{\@gls@noidx@setsanitizesort}{%
454   \glssanitizesortfalse
455   \glssetexpandfield{sortvalue}%
456 }

457 \define@choicekey[gls]{sanitize}{sort}{true,false}[true]{%
458   \setbool{glssanitizesort}{#1}%
459   \ifglssanitizesort
460     \glssetnoexpandfield{sortvalue}%
461   \else
462     \glssetexpandfield{sortvalue}%
463   \fi
464   \GlossariesWarning{sanitize={sort} package option
465     deprecated. Use sanitizesort instead}%
466 }
```

sanitize

```
467 \define@key{glossaries.sty}{sanitize}[description=true,symbol=true,name=true]{%
468   \ifthenelse{\equal{#1}{none}}{%
469     {%
470       \GlossariesWarning{sanitize package option deprecated}%
471       \glssetexpandfield{name}%
472       \glssetexpandfield{symbol}%
473       \glssetexpandfield{symbolplural}%
474       \glssetexpandfield{desc}%
475       \glssetexpandfield{descplural}%
476     }%
477     {%
478       \setkeys[gls]{sanitize}{#1}%
479     }%
480 }
```

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

```
481 \newif\ifglstranslate
```

ls@nottranslatorhook \@gls@nottranslatorhook has been removed.

\@gls@usetranslator

```
482 \newcommand*\@gls@usetranslator{%
  polyglossia tricks \@ifpackageloaded into thinking that babel has been loaded,
  so check for polyglossia as well.
483   \@ifpackageloaded{polyglossia}%
484   {%
485     \let\glsifusetranslator\@secondoftwo
486   }%
487   {%
```

```

488 \@ifpackageloaded{babel}%
489 {%
490 \IfFileExists{translator.sty}%
491 {%
492 \RequirePackage{translator}%
493 \let\glsifusetranslator\@firstoftwo
494 }%
495 {}%
496 }%
497 {}%
498 }%
499 }

```

fusedtranslatordict Checks if given translator dictionary has been loaded.

```

500 \newcommand{\glsifusedtranslatordict}[3]{%
501 \glsifusetranslator
502 {\ifcsdef{ver@glossaries-dictionary-#1.dict}{#2}{#3}}%
503 {#3}%
504 }

```

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

```

505 \@gls@declareoption{notranslate}{%
506 \glstranslatefalse
507 \let\@gls@usetranslator\relax
508 \let\glsifusetranslator\@secondoftwo
509 }

```

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

```

510 \define@choicekey{glossaries.sty}{translate}[\val\nr]%
511 {true,false,babel}[true]%
512 {%
513 \ifcase\nr\relax
514 \glstranslatetrue
515 \renewcommand*\@gls@usetranslator{%
516 \@ifpackageloaded{polyglossia}%
517 {%
518 \let\glsifusetranslator\@secondoftwo
519 }%
520 {}%
521 \@ifpackageloaded{babel}%
522 {%
523 \IfFileExists{translator.sty}%
524 {%
525 \RequirePackage{translator}%
526 \let\glsifusetranslator\@firstoftwo
527 }%
528 {}%
529 }%

```

```

530     {}%
531   }%
532 }%
533 \or
534   \glstranslatefalse
535   \let\@gls@usetranslator\relax
536   \let\glsifusetranslator\@secondoftwo
537 \or
538   \glstranslatetrue
539   \let\@gls@usetranslator\relax
540   \let\glsifusetranslator\@secondoftwo
541 \fi
542 }

```

Set the default value:

```

543 \glstranslatefalse
544 \let\glsifusetranslator\@secondoftwo
545 \@ifpackageloaded{translator}%
546 {%
547   \glstranslatetrue
548   \let\glsifusetranslator\@firstoftwo
549 }%
550 {%
551   \@for\gls@thissty:=tracklang,babel,ngerman,polyglossia\do
552   {
553     \@ifpackageloaded{\gls@thissty}%
554     {%
555       \glstranslatetrue
556       \@endfortrue
557     }%
558   }%
559 }
560 }

```

indexonlyfirst Set whether to only index on first use.

```

561 \define@boolkey{glossaries.sty}[gls]{indexonlyfirst}[true]{}
562 \glsindexonlyfirstfalse

```

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

```

563 \define@boolkey{glossaries.sty}[gls]{hyperfirst}[true]{}
564 \glshyperfirsttrue

```

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

```

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

```

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

```

566 \define@boolkey{glossaries.sty}[glsacr]{footnote}[true]{}

```

```

567 \ifbool{glsacrdescription}%
568 {}%
569 {%
570 \renewcommand*{\@gls@sanitizedesc}{}%
571 }%
572 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
573 }

```

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

```

574 \define@boolkey{glossaries.sty}[glsacr]{description}[true]{%
575 \renewcommand*{\@gls@sanitizesymbol}{}%
576 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
577 }

```

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

```

578 \define@boolkey{glossaries.sty}[glsacr]{smallcaps}[true]{%
579 \renewcommand*{\@gls@sanitizesymbol}{}%
580 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
581 }

```

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

```

582 \define@boolkey{glossaries.sty}[glsacr]{smaller}[true]{%
583 \renewcommand*{\@gls@sanitizesymbol}{}%
584 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
585 }

```

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

```

586 \define@boolkey{glossaries.sty}[glsacr]{dua}[true]{%
587 \renewcommand*{\@gls@sanitizesymbol}{}%
588 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
589 }

```

shortcuts Define acronym shortcuts.

```

590 \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.

```

591 \newcommand*{\glsorder}{word}

```

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

```

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

```

order

```

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

```


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

```
595 \newif\ifglxindy
```

The default is `makeindex`:

```
596 \glxindyfalse
```

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

```
597 \@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.

```
598 \define@boolkey[gls]{xindy}{glsnumbers}[true]{}
```

```
599 \gls{xindy@glsglsnumberstrue}
```

`\@xdy@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.)

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

Define key to set the language

```
601 \define@key[gls]{xindy}{language}{\def\@xdy@main@language{#1}}
```

`\gls@codepage` Define the code page. If `\inputencodingname` is defined use that, otherwise have initialise with no codepage.

```
602 \ifcsundef{inputencodingname}{%
```

```
603 \def\gls@codepage{}}{%
```

```
604 \def\gls@codepage{\inputencodingname}
```

```
605 }
```

Define a key to set the code page.

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

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

```
607 \define@key{glossaries.sty}{xindy}[]{%
```

```
608 \glxindytrue
```

```
609 \setkeys[gls]{xindy}{#1}%
```

```
610 }
```

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

```
611 \@gls@declareoption{xindygloss}{%
```

```
612 \glxindytrue
```

```
613 }
```

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

```
614 \@gls@declareoption{xindynoglsnumbers}{%
615   \glsxindytrue
616   \gls@xindy@glsnumbersfalse
617 }
```

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

```
618 \define@boolkey{glossaries.sty}[gls]{automake}[true]{%
619   \ifglsautomake
620     \renewcommand*{\@gls@doautomake}{%
621       \PackageError{glossaries}{You must use
622       \string\makeglossaries\space with automake=true}
623       {%
624         Either remove the automake=true setting or
625         add \string\makeglossaries\space to your document preamble.%
626       }%
627     }%
628   \else
629     \renewcommand*{\@gls@doautomake}{}%
630   \fi
631 }
632 \glsautomakefalse
```

`\@gls@doautomake`

```
633 \newcommand*{\@gls@doautomake}{%
634   \AtEndDocument{\@gls@doautomake}
```

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

```
635 \define@boolkey{glossaries.sty}[gls]{savewrites}[true]{%
636   \ifglssavewrites
637     \renewcommand*{\glswritefiles}{\@glswritefiles}%
638   \else
639     \let\glswritefiles\@empty
640   \fi
641 }
```

Set default:

```
642 \glssavewritesfalse
643 \let\glswritefiles\@empty
```

`compatible-3.07`

```
644 \define@boolkey{glossaries.sty}[gls]{compatible-3.07}[true]{%
645   \boolfalse{glscompatible-3.07}
```

`compatible-2.07`

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

Also set 3.07 compatibility if this option is set.

```
647 \ifbool{glscompatible-2.07}%  
648 {%  
649   \booltrue{glscompatible-3.07}%  
650 }%  
651 {}%  
652 }  
653 \boolfalse{glscompatible-2.07}
```

symbols Create a “symbols” glossary type

```
654 \@gls@declareoption{symbols}{%  
655   \let\@gls@do@symbolsdef\@gls@symbolsdef  
656 }
```

Default is not to define the symbols glossary:

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

\@gls@symbolsdef

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

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

```
661   \newcommand*{\gls@tr@set@symbols@toctitle}{%  
662     \translatelet{\glossarytoctitle}{Symbols (glossaries)}%  
663   }%  
664 }%
```

numbers Create a “symbols” glossary type

```
665 \@gls@declareoption{numbers}{%  
666   \let\@gls@do@numbersdef\@gls@numbersdef  
667 }
```

Default is not to define the numbers glossary:

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

\@gls@numbersdef

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

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

```
672   \newcommand*{\gls@tr@set@numbers@toctitle}{%  
673     \translatelet{\glossarytoctitle}{Numbers (glossaries)}%  
674   }%  
675 }%
```

index Create an “index” glossary type

```
676 \@gls@declareoption{index}{%  
677   \let\@gls@do@indexdef\@gls@indexdef  
678 }
```

Default is not to define index glossary:

```
679 \newcommand*{\@gls@do@indexdef}{}%
```

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

```
680 \newcommand*{\@gls@indexdef}{%
681   \newglossary[ilg]{index}{ind}{idx}{\indexname}%
682   \newcommand*{\printindex}[1][\printglossary[type=index,##1]]{%
683     \newcommand*{\newterm}[2][\newglossaryentry{##2}%
684       {type={index},name={##2},description={\nopostdesc},##1}}%
685   }%
686 }
```

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

```
687 \@for\CurrentOption := \@declaredoptions\do{%
688   \ifx\CurrentOption\@empty
689   \else
690     \@expandtwoargs
691     \in@ {,\CurrentOption ,}{,\@classoptionslist,\@curroptions,}%
692     \ifin@
693     \@use@ption
694     \expandafter \let\csname ds@\CurrentOption\endcsname\@empty
695   \fi
696 \fi
697 }
```

Now process options passed to the package:

```
698 \ProcessOptionsX
```

Load backward compatibility stuff:

```
699 \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:

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

Now define \setupglossaries:

```
703 \newcommand*{\setupglossaries}[1]{%
704   \renewcommand*{\@gls@setacrstyle}{}%
705   \ifglsacrshortcuts
706     \def\@gls@setupshortcuts{\glsacrshortcutstrue}%
707   \else
708     \def\@gls@setupshortcuts{%
709       \ifglsacrshortcuts
710         \DefineAcronymSynonyms
711       \fi
712     }%
713   \fi
```

```

714 \glsacrshortcutsfalse
715 \let\@gls@do@numbersdef\relax
716 \let\@gls@do@symbolssdef\relax
717 \let\@gls@do@indexdef\relax
718 \let\@gls@do@acronymsdef\relax
719 \setkeys{glossaries.sty}{#1}%
720 \@gls@setacrstyle
721 \@gls@setupshortcuts
722 \@gls@do@acronymsdef
723 \@gls@do@numbersdef
724 \@gls@do@symbolssdef
725 \@gls@do@indexdef
726 }

```

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

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

```

727 \ifthenelse{\equal{\glscounter}{section}}{%
728 {%
729   \ifcsundef{chapter}{}%
730   {%
731     \let\@gls@old@chapter\@chapter
732     \def\@chapter[#1]#2{\@gls@old@chapter[#1]{#2}%
733       \ifcsundef{hyperdef}{}{\hyperdef{section}{\thesection}{}}}%
734   }%
735 }%
736 {}

```

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

```

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

```

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

```

738 \newcommand*{\@onlypremakeg}[1]{%
739   \ifx\@gls@onlypremakeg\@empty
740     \def\@gls@onlypremakeg{#1}%
741   \else
742     \expandafter\toks@\expandafter{\@gls@onlypremakeg}%
743     \edef\@gls@onlypremakeg{\the\toks@,\noexpand#1}%
744   \fi
745 }

```

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

```
746 \newcommand*{\@disable@onlypremakeg}{%
747 \@for\@thiscs:=\@gls@onlypremakeg\do{%
748   \expandafter\@disable@premakecs\@thiscs%
749 }}
```

`\@disable@premakecs` Disables the given command.

```
750 \newcommand*{\@disable@premakecs}[1]{%
751   \def#1{\PackageError{glossaries}{\string#1\space may only be
752     used before \string\makeglossaries}{You can't use
753     \string#1\space after \string\makeglossaries}}%
754 }
```

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`

```
755 \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`

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

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

```
757 \newcommand*{\glssettoctitle}[1]{%
758   \def\glossarytoctitle{\curname @glotype@#1@title\endcurname}}
```

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`

```
759 \providecommand*{\entryname}{Notation}
```

`\descriptionname`

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

`\symbolname`

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

`\pagelistname`

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

Labels for makeindex's symbol and number groups:

glssymbolsgroupname

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

glsnumpersgroupname

```
764 \providecommand*{\glsnumpersgroupname}{Numbers}
```

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

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

\glsacrpluralsuffix Default plural suffix for acronyms

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

glsupacrpluralsuffix

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

\seename

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

\andname

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

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

requireGlossariesLang

```
770 \newcommand*{\RequireGlossariesLang}[1]{%
771   \@ifundefined{ver@glossaries-#1.1df}{\input{glossaries-#1.1df}}{}%
772 }
```

providesGlossariesLang

```
773 \newcommand*{\ProvidesGlossariesLang}[1]{%
774   \ProvidesFile{glossaries-#1.1df}%
775 }
```

addglossarytocaptions Does nothing if translator hasn't been loaded.

```
776 \newcommand*{\addglossarytocaptions}[1]{}%
```

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

```
777 \ifglstranslate
```

Load tracklang

```
778 \RequirePackage{tracklang}
```

Load translator if required.

```
779 \@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.)

```
780 \ifpackageloaded{translator}
781 {%
```

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.

```
782 \ifboolexpr
783 {
784   test {\ifdefstring{\trans@languages}{English}}
785   and not
786   test {\ifdefstring{\bbl@loaded}{english}}
787 }
788 {%
789 \let\glsifusetranslator\@secondoftwo
790 }%
791 {%
792 \usedictionary{glossaries-dictionary}%
793 \renewcommand*{\addglossarytocaptions}[1]{%
794 \ifcsundef{captions#1}{}%
795 {%
796 \expandafter\let\expandafter\@gls@tmp\csname captions#1\endcsname
797 \expandafter\toks@\expandafter{\@gls@tmp
798 \renewcommand*{\glossaryname}{\translate{Glossary}}}%
799 }%
800 \expandafter\edef\csname captions#1\endcsname{\the\toks@}%
801 }%
802 }%
803 }%
804 }%
805 }%
```

Check for tracked languages

```
806 \AnyTrackedLanguages
807 {%
808 \ForEachTrackedDialect{\this@dialect}{%
809 \IfTrackedLanguageFileExists{\this@dialect}%
810 {glossaries-}% prefix
811 {.ldf}%
812 {%
813 \RequireGlossariesLang{\CurrentTrackedTag}%
814 }%
815 {%
816 \PackageWarningNoLine{glossaries}%
817 {No language module detected for '\this@dialect'.\MessageBreak
```



```

818         Language modules need to be installed separately.\MessageBreak
819         Please check on CTAN for a bundle called\MessageBreak
820         ‘glossaries-\CurrentTrackedLanguage’ or similar}%
821     }%
822 }%
823 }%
824 {}%

    if using translator use translator interface.
825 \glsifusetranslator
826 {%
827     \renewcommand*{\glssettoctitle}[1]{%
828         \ifcsdef{gls@tr@set@#1@toctitle}%
829         {%
830             \csuse{gls@tr@set@#1@toctitle}%
831         }%
832         {%
833             \def\glossarytoctitle{\csname @glotype@#1@title\endcsname}%
834         }%
835     }%
836     \renewcommand*{\glossaryname}{\translate{Glossary}}%
837     \renewcommand*{\acronymname}{\translate{Acronyms}}%
838     \renewcommand*{\entryname}{\translate{Notation (glossaries)}}%
839     \renewcommand*{\descriptionname}{%
840         \translate{Description (glossaries)}}%
841     \renewcommand*{\symbolname}{\translate{Symbol (glossaries)}}%
842     \renewcommand*{\pagelistname}{%
843         \translate{Page List (glossaries)}}%
844     \renewcommand*{\glssymbolsgroupname}{%
845         \translate{Symbols (glossaries)}}%
846     \renewcommand*{\glsnumbersgroupname}{%
847         \translate{Numbers (glossaries)}}%
848 }{}%
849 \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.

```
850 \DeclareRobustCommand*{\nopostdesc}{}
```

`\@nopostdesc` Suppress next description terminator.

```

851 \newcommand*{\@nopostdesc}{%
852     \let\org@glspostdescription\glspostdescription
853     \def\glspostdescription{%
854         \let\glspostdescription\org@glspostdescription}%
855 }

```

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

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

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

```
857 \newcommand{\glspar}{\par}
```

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

```
858 \newcommand{\setStyleFile}[1]{%
859   \renewcommand*{\gls@istfilebase}{#1}%
   Just in case \istfilename has been modified.
860   \ifglsxindy
861     \def\istfilename{\gls@istfilebase.xdy}
862   \else
863     \def\istfilename{\gls@istfilebase.ist}
864   \fi
865 }
```

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

```
866 \@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 re-defining 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`

```
867 \ifglsxindy
868   \def\istfilename{\gls@istfilebase.xdy}
869 \else
870   \def\istfilename{\gls@istfilebase.ist}
871 \fi
```

`\gls@istfilebase`

```
872 \newcommand*{\gls@istfilebase}{\jobname}
```

The `makeglossaries` Perl script picks up this name from the auxiliary file. If the name ends with `.xdy` it calls `xindy` otherwise it calls `makeindex`. Since its not required by \TeX , `\@istfilename` ignores its argument.

`\@istfilename`

```
873 \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`

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

`\glsSetCompositor` Sets the compositor.

```

875 \newcommand*\glsSetCompositor[1]{%
876   \renewcommand*\glscompositor{#1}}

```

Only use before `\makeglossaries`

```

877 \@onlypremakeg\glsSetCompositor

```

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

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

```

878 \newcommand*\@glsAlphacompositor{\glscompositor}

```

`sSetAlphaCompositor` Sets the alpha compositor.

```

879 \ifglsxindy
880   \newcommand*\glsSetAlphaCompositor[1]{%
881     \renewcommand*\@glsAlphacompositor{#1}}
882 \else
883   \newcommand*\glsSetAlphaCompositor[1]{%
884     \glsnoxindywarning\glsSetAlphaCompositor}
885 \fi

```

Can only be used before `\makeglossaries`

```

886 \@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.

```

887 \newcommand*\gls@suffixF{}

```

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

```

888 \newcommand*\glsSetSuffixF[1]{%
889   \renewcommand*\gls@suffixF{#1}}

```

Only has an effect when used before `\makeglossaries`

```

890 \@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.

```

891 \newcommand*\gls@suffixFF{}

```

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

```

892 \newcommand*\glsSetSuffixFF[1]{%
893   \renewcommand*\gls@suffixFF{#1}%
894 }

```

`\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".

```
895 \ifcsundef{hyperlink}%
896 {%
897   \newcommand*{\glsnumberformat}[1]{#1}%
898 }%
899 {%
900   \newcommand*{\glsnumberformat}[1]{\glshypernumber{#1}}%
901 }
```

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
902 \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
903 \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
904 \newcommand*{\glossarypreamble}{%
905   \csuse{@glossarypreamble@currentglossary}%
906 }
```

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

Code provided by Michael Pock.

```

907 \newcommand{\setglossarypreamble}[2][\glsdefaultttype]{%
908   \ifglossaryexists{#1}{%
909     \csgdef{@glossarypreamble@#1}{#2}%
910   }{%
911     \GlossariesWarning{%
912       Glossary ‘#1’ is not defined%
913     }%
914   }%
915 }

```

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`

```

916 \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`.

```

917 \newcommand*{\glossarysection}[2][\@gls@title]{%
918   \def\@gls@title{#2}%
919   \ifcsundef{phantomsection}%
920   {%
921     \@glossarysection{#1}{#2}%
922   }%
923   {%
924     \p@glossarysection{#1}{#2}%
925   }%

926   \glsglossarymark{\glossarytoctitle}%
927 }

```

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

```

928 \ifcsundef{glossarymark}%
929 {%
930   \newcommand{\glsglossarymark}[1]{\glossarymark{#1}}
931 }%
932 {%
933   \@ifclassloaded{memoir}

```

```

934  {%
935    \newcommand{\glsglossarymark}[1]{%
936      \ifglsucmark
937        \markboth{\memUHead{#1}}{\memUHead{#1}}%
938      \else
939        \markboth{#1}{#1}%
940      \fi
941    }
942  }%
943  {%
944    \newcommand{\glsglossarymark}[1]{%
945      \ifglsucmark
946        \@mkboth{\mfirstucMakeUppercase{#1}}{\mfirstucMakeUppercase{#1}}%
947      \else
948        \@mkboth{#1}{#1}%
949      \fi
950    }
951  }
952 }

```

`\glossarymark` Provided for backward compatibility:

```

953 \providecommand{\glossarymark}[1]{%
954   \ifglsucmark
955     \@mkboth{\mfirstucMakeUppercase{#1}}{\mfirstucMakeUppercase{#1}}%
956   \else
957     \@mkboth{#1}{#1}%
958   \fi
959 }

```

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.

`\setglossarysection`

```

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

```

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

`\@glossarysection`

```

962 \newcommand*\@glossarysection[2]{%
963   \ifdefempty\@glossarysecstar
964   {%
965     \csname\@glossarysec\endcsname[#1]{#2}%
966   }%
967   {%

```

```

968 \csname\@glossarysec\endcsname*{#2}%
969 \gls@toc{#1}{\@glossarysec}%
970 }%

```

Do automatic labelling if required

```

971 \@glossaryseclabel
972 }

```

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.

\@p@glossarysection

```

973 \newcommand*\@p@glossarysection}[2]{%
974 \glsclearpage
975 \phantomsection
976 \ifdefempty\@glossarysecstar
977 {%
978 \csname\@glossarysec\endcsname*{#2}%
979 }%
980 {%
981 \gls@toc{#1}{\@glossarysec}%
982 \csname\@glossarysec\endcsname*{#2}%
983 }%

```

Do automatic labelling if required

```

984 \@glossaryseclabel
985 }

```

\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.

```

986 \newcommand*\gls@doclearpage{%
987 \ifthenelse{\equal{\@glossarysec}{chapter}}{%
988 {%
989 \ifcsundef{cleardoublepage}%
990 {%
991 \clearpage
992 }%
993 {%
994 \ifcsdef{if@openright}%
995 {%
996 \if@openright
997 \cleardoublepage
998 \else
999 \clearpage
1000 \fi
1001 }%
1002 {%

```

```

1003      \cleardoublepage
1004    }%
1005  }%
1006 }%
1007 {}%
1008 }

```

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

```

1009 \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`

```

1010 \newcommand*{\@gls@toc}[2]{%
1011   \ifglstoc
1012     \ifglsnumberline
1013       \addcontentsline{toc}{#2}{\protect\numberline{}}#1}%
1014   \else
1015     \addcontentsline{toc}{#2}{#1}%
1016   \fi
1017 \fi
1018 }

```

1.4 Xindy

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

`\glsnoxindywarning` Issues a warning if `xindy` hasn't been specified. These warnings can be suppressed by redefining `\glsnoxindywarning` to ignore its argument

```

1019 \newcommand*{\glsnoxindywarning}[1]{%
1020   \GlossariesWarning{Not in xindy mode --- ignoring \string#1}%
1021 }

```

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

```

1022 \ifglsxindy
1023   \edef\@xdyattributes{\string"default\string"}%
1024 \fi

```

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

```

1025 \ifglsxindy
1026   \edef\@xdyattributelist{}%
1027 \fi

```


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

```
1028 \ifglxindy
1029   \def\@xdylocref{}
1030 \fi
```

`\@gls@ifinlist`

```
1031 \newcommand*\@gls@ifinlist[4]{%
1032   \def\@do@ifinlist##1,##2\end@do@ifinlist{%
1033     \def\@gls@listsuffix{##2}%
1034     \ifx\@gls@listsuffix\@empty
1035       #4%
1036     \else
1037       #3%
1038     \fi
1039   }%
1040   \@do@ifinlist,##2,##1,\end@do@ifinlist
1041 }
```

`\GlsAddXdyCounters` 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.

```
1042 \ifglxindy
1043   \newcommand*\@xdycounters{\@glscounter}
1044   \newcommand*\GlsAddXdyCounters[1]{%
1045     \@for\@gls@ctr:=#1\do{%
1046       \edef\@do@addcounter{%
1047         \noexpand\@gls@ifinlist{\@gls@ctr}{\@xdycounters}{}%
1048         {%
1049           \noexpand\edef\noexpand\@xdycounters{\@xdycounters,%
1050             \noexpand\@gls@ctr}%
1051         }%
1052       }%
1053       \@do@addcounter
1054     }
1055   }
```

Only has an effect before `\writeist`:

```
1056   \@onlypremakeg\GlsAddXdyCounters
1057 \else
1058   \newcommand*\GlsAddXdyCounters[1]{%
1059     \glsnoxindywarning\GlsAddXdyAttribute
1060   }
1061 \fi
```

`d@glsaddxdycounters` Counters must all be identified before adding attributes.

```
1062 \newcommand*\@disabled@glsaddxdycounters{%
1063   \PackageError{glossaries}{\string\GlsAddXdyCounters\space
```

```

1064   can't be used after \string\GlsAddXdyAttribute}{Move all
1065   occurrences of \string\GlsAddXdyCounters\space before the first
1066   instance of \string\GlsAddXdyAttribute}%
1067 }

```

`\GlsAddXdyAttribute` Adds an attribute.

```

1068 \ifglxindy

```

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

```

1069 \newcommand*\@glsaddxdyattribute[2]{%

```

Add to xindy attribute list

```

1070 \edef\@xdyattributes{\@xdyattributes ^^J \string"#1\string" ^^J
1071 \string"#2#1\string"}%

```

Add to xindy markup location.

```

1072 \expandafter\toks@\expandafter{\@xdylocref}%
1073 \edef\@xdylocref{\the\toks@ ^^J%
1074 (markup-locref
1075 :open \string"\glstildechar n%
1076 \expandafter\string\csname glsX#2X#1\endcsname
1077 \string" ^^J
1078 :close \string"\string" ^^J
1079 :attr \string"#2#1\string")}%

```

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

```

1080 \expandafter\gdef\csname glsX#2X#1\endcsname##1##2{%
1081 \setentrycounter[##1]{#2}\csname #1\endcsname{##2}%
1082 }%
1083 }

```

High-level command:

```

1084 \newcommand*\GlsAddXdyAttribute[1]{%

```

Add to comma-separated attribute list

```

1085 \ifx\@xdyattributelist\@empty
1086 \edef\@xdyattributelist{#1}%
1087 \else
1088 \edef\@xdyattributelist{\@xdyattributelist,#1}%
1089 \fi

```

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

```

1090 \@for\@this@counter:=\@xdycounters\do{%
1091 \protected@edef\gls@do@addxdyattribute{%
1092 \noexpand\@glsaddxdyattribute{#1}{\@this@counter}%
1093 }
1094 \gls@do@addxdyattribute
1095 }%

```

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

```

1096 \let\GlsAddXdyCounters\@disabled@glsaddxdycounters
1097 }

```

Only has an effect before `\writeist`:

```

1098 \@onlypremakeg\GlsAddXdyAttribute
1099 \else
1100 \newcommand*\GlsAddXdyAttribute[1]{%
1101 \glsnnoxindywarning\GlsAddXdyAttribute}
1102 \fi

```

`redefinedattributes` Add known attributes for all defined counters

```

1103 \ifglsxindy
1104 \newcommand*\@glsgls@addpredefinedattributes{%
1105 \GlsAddXdyAttribute{glsnumberformat}
1106 \GlsAddXdyAttribute{textrm}
1107 \GlsAddXdyAttribute{textsf}
1108 \GlsAddXdyAttribute{texttt}
1109 \GlsAddXdyAttribute{textbf}
1110 \GlsAddXdyAttribute{textmd}
1111 \GlsAddXdyAttribute{textit}
1112 \GlsAddXdyAttribute{textup}
1113 \GlsAddXdyAttribute{textsl}
1114 \GlsAddXdyAttribute{textsc}
1115 \GlsAddXdyAttribute{emph}
1116 \GlsAddXdyAttribute{glshypernumber}
1117 \GlsAddXdyAttribute{hyperrm}
1118 \GlsAddXdyAttribute{hypersf}
1119 \GlsAddXdyAttribute{hypertt}
1120 \GlsAddXdyAttribute{hyperbf}
1121 \GlsAddXdyAttribute{hypermd}
1122 \GlsAddXdyAttribute{hyperit}
1123 \GlsAddXdyAttribute{hyperup}
1124 \GlsAddXdyAttribute{hypersl}
1125 \GlsAddXdyAttribute{hypersc}
1126 \GlsAddXdyAttribute{hyperemph}

1127 \GlsAddXdyAttribute{glsglignore}
1128 }
1129 \else
1130 \let\@glsgls@addpredefinedattributes\relax
1131 \fi

```

`\@xdyuseralphabets` List of additional alphabets

```

1132 \def\@xdyuseralphabets{}

```

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

```

1133 \ifglsxindy
1134 \newcommand*\@GlsAddXdyAlphabet[2]{%
1135 \edef\@xdyuseralphabets{%
1136 \@xdyuseralphabets ^^J
1137 (define-alphabet "#1" (#2))}}

```

```

1138 \else
1139   \newcommand*{\GlsAddXdyAlphabet}[2]{%
1140     \glsnxindywarning\GlsAddXdyAlphabet}
1141 \fi

```

This code is only required for xindy:

```

1142 \ifglsxindy

```

`\ls@xdy@locationlist` List of predefined location names.

```

1143   \newcommand*{\@gl@xdy@locationlist}{%
1144     roman-page-numbers,%
1145     Roman-page-numbers,%
1146     arabic-page-numbers,%
1147     alpha-page-numbers,%
1148     Alpha-page-numbers,%
1149     Appendix-page-numbers,%
1150     arabic-section-numbers%
1151   }

```

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

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

```

1152   \protected@edef\@gl@roman{\@roman{0}\string"
1153     \string"roman-numbers-lowercase\string" :sep \string"}}%
1154   \@onelevel@sanitize\@gl@roman
1155   \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
1156     :sep \string"}%
1157   \@onelevel@sanitize\@tmp
1158   \ifx\@tmp\@gl@roman
1159     \expandafter
1160     \edef\csname @gl@xdy@Lclass@roman-page-numbers\endcsname{%
1161       \string"roman-numbers-lowercase\string"%
1162     }%
1163   \else
1164     \expandafter
1165     \edef\csname @gl@xdy@Lclass@roman-page-numbers\endcsname{
1166       :sep \string"\@gl@roman\string"%
1167     }%
1168   \fi

```

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

```

1169   \expandafter\def\csname @gl@xdy@Lclass@Roman-page-numbers\endcsname{%
1170     \string"roman-numbers-uppercase\string"%
1171   }%

```

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

```

1172 \expandafter\def\csname @gls@xdy@Lclass@arabic-page-numbers\endcsname{%
1173 \string"arabic-numbers\string"%
1174 }%

```

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

```

1175 \expandafter\def\csname @gls@xdy@Lclass@alpha-page-numbers\endcsname{%
1176 \string"alpha\string"%
1177 }%

```

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

```

1178 \expandafter\def\csname @gls@xdy@Lclass@Alpha-page-numbers\endcsname{%
1179 \string"ALPHA\string"%
1180 }%

```

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

```

1181 \expandafter\def\csname @gls@xdy@Lclass@Appendix-page-numbers\endcsname{%
1182 \string"ALPHA\string"
1183 :sep \string"\@glsAlphacompositor\string"
1184 \string"arabic-numbers\string"%
1185 }

```

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

```

1186 \expandafter\def\csname @gls@xdy@Lclass@arabic-section-numbers\endcsname{%
1187 \string"arabic-numbers\string"
1188 :sep \string"\glscompositor\string"
1189 \string"arabic-numbers\string"%
1190 }%

```

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

```

1191 \def\@xdyuserlocationdefs{}

```

xdyuserlocationnames List of additional user location names

```

1192 \def\@xdyuserlocationnames{}

```

End of xindy-only block:

```

1193 \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.)

```

1194 \ifglsxindy
1195 \newcommand*\GlsAddXdyLocation[3][{}]{%
1196 \def\@gls@tmp{#1}%

```

```

1197 \ifx\@gls@tmp\@empty
1198 \edef\@xdyuserlocationdefs{%
1199 \@xdyuserlocationdefs ^^J%
1200 (define-location-class \string"#2\string"^^J\space\space
1201 \space(:sep \string"{} \glsoopenbrace\string" #3
1202 :sep \string"\glsclosebrace\string"))
1203 }%
1204 \else
1205 \edef\@xdyuserlocationdefs{%
1206 \@xdyuserlocationdefs ^^J%
1207 (define-location-class \string"#2\string"^^J\space\space
1208 \space(:sep "\glsoopenbrace"
1209 #1
1210 :sep "\glsclosebrace\glsoopenbrace" #3
1211 :sep "\glsclosebrace"))
1212 }%
1213 \fi
1214 \edef\@xdyuserlocationnames{%
1215 \@xdyuserlocationnames^^J\space\space\space
1216 \string"#1\string"}%
1217 }

```

Only has an effect before `\writeist`:

```

1218 \@onlypremakeg\GlsAddXdyLocation
1219 \else
1220 \newcommand*{\GlsAddXdyLocation}[2]{%
1221 \glsnoxindywarning\GlsAddXdyLocation}
1222 \fi

```

`ylocationclassorder` Define location class order

```

1223 \ifglxindy
1224 \edef\@xdylocationclassorder{^^J\space\space\space
1225 \string"roman-page-numbers\string"^^J\space\space\space
1226 \string"arabic-page-numbers\string"^^J\space\space\space
1227 \string"arabic-section-numbers\string"^^J\space\space\space
1228 \string"alpha-page-numbers\string"^^J\space\space\space
1229 \string"Roman-page-numbers\string"^^J\space\space\space
1230 \string"Alpha-page-numbers\string"^^J\space\space\space
1231 \string"Appendix-page-numbers\string"
1232 \@xdyuserlocationnames^^J\space\space\space
1233 \string"see\string"
1234 }
1235 \fi

```

Change the location order.

`yLocationClassOrder`

```

1236 \ifglxindy
1237 \newcommand*\GlsSetXdyLocationClassOrder[1]{%
1238 \def\@xdylocationclassorder{#1}}

```

```

1239 \else
1240   \newcommand*\GlsSetXdyLocationClassOrder[1]{%
1241     \glsnoxywarning\GlsSetXdyLocationClassOrder}
1242 \fi

\@xdysortrules   Define sort rules

1243 \ifglxindy
1244   \def\@xdysortrules{}
1245 \fi

\GlsAddSortRule  Add a sort rule

1246 \ifglxindy
1247   \newcommand*\GlsAddSortRule[2]{%
1248     \expandafter\toks@\expandafter{\@xdysortrules}%
1249     \protected@edef\@xdysortrules{\the\toks@ ^^J
1250       (sort-rule \string"#1\string" \string"#2\string")}%
1251   }
1252 \else
1253   \newcommand*\GlsAddSortRule[2]{%
1254     \glsnoxywarning\GlsAddSortRule}
1255 \fi

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

1256 \ifglxindy
1257   \def\@xdyrequiredstyles{tex}
1258 \fi

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

1259 \ifglxindy
1260   \newcommand*\GlsAddXdyStyle[1]{%
1261     \edef\@xdyrequiredstyles{\@xdyrequiredstyles,#1}}%
1262 \else
1263   \newcommand*\GlsAddXdyStyle[1]{%
1264     \glsnoxywarning\GlsAddXdyStyle}
1265 \fi

\GlsSetXdyStyles  Reset the list of required styles

1266 \ifglxindy
1267   \newcommand*\GlsSetXdyStyles[1]{%
1268     \edef\@xdyrequiredstyles{#1}}
1269 \else
1270   \newcommand*\GlsSetXdyStyles[1]{%
1271     \glsnoxywarning\GlsSetXdyStyles}
1272 \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.

```
1273 \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.

```
1274 \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.

```
1275 \ifglsxindy
1276   \newcommand*\GlsSetXdyLanguage[2][\glsdefaulttype]{%
1277     \ifglossaryexists{#1}{%
1278       \expandafter\def\csname @xdy@#1@language\endcsname{#2}%
1279     }{%
1280       \PackageError{glossaries}{Can't set language type for
1281         glossary type '#1' --- no such glossary}{%
1282         You have specified a glossary type that doesn't exist}}
1283 \else
1284   \newcommand*\GlsSetXdyLanguage[2][]{%
1285     \glsnoxywarning\GlsSetXdyLanguage}
1286 \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.

```
1287 \def\@gls@codepage#1#2{}
```

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

```
1288 \ifglsxindy
1289   \newcommand*\GlsSetXdyCodePage[1]{%
1290     \renewcommand*\@gls@codepage{#1}%
1291   }

   Suggested by egreg:
1292   \AtBeginDocument{%
1293     \ifx\@gls@codepage\@empty
1294       \@ifpackageloaded{fontspec}{\def\@gls@codepage{utf8}}{}%
1295     \fi
1296   }
```



```

1297 \else
1298   \newcommand*{\GlsSetXdyCodePage}[1]{%
1299     \glsnoxindywarning\GlsSetXdyCodePage}
1300 \fi

```

`\@xdylettergroups` Store letter group definitions.

```

1301 \ifglxindy
1302   \ifglx@xindy@glslnumbers
1303     \def\@xdylettergroups{(define-letter-group
1304       \string"glslnumbers\string"^^J\space\space\space
1305       :prefixes (\string"0\string" \string"1\string"
1306       \string"2\string" \string"3\string" \string"4\string"
1307       \string"5\string" \string"6\string" \string"7\string"
1308       \string"8\string" \string"9\string")^^J\space\space\space
1309       :before \string"@glslfirstletter\string"))}
1310   \else
1311     \def\@xdylettergroups{}
1312   \fi
1313 \fi

```

`\GlsAddLetterGroup` 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.

```

1314   \newcommand*\GlsAddLetterGroup[2]{%
1315     \expandafter\toks@\expandafter{\@xdylettergroups}%
1316     \protected@edef\@xdylettergroups{\the\toks@^^J%
1317       (define-letter-group \string"#1\string"^^J\space\space\space#2)}%
1318   }%

```

1.5 Loops and conditionals

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

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

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

```

1319 \newcommand*\forallglossaries[3][\@glo@types]{%
1320   \@for#2:=#1\do{\ifx#2\@empty\else#3\fi}%
1321 }

```

`\forallacronyms`

```

1322 \newcommand*\forallacronyms[2]{%
1323   \@for#1:=\@glslacronymlists\do{\ifx#1\@empty\else#2\fi}%
1324 }

```

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

```
\forglsentries[<type>]{<cmd>}{<code>}
```

where $\langle type \rangle$ is the glossary label and $\langle cmd \rangle$ is a control sequence which will be set to the entry label in the current iteration.

```

1325 \newcommand*{\forlgsentries}[3][\glsdefaulttype]{%
1326   \edef\@glo@list{\csname glolist@#1\endcsname}%
1327   \@for#2:=\@glo@list\do
1328     {%
1329       \ifdefempty{#2}{-}{#3}%
1330     }%
1331 }

```

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

`\foralllgsentries[$\langle glossary list \rangle$]{ $\langle cmd \rangle$ }{ $\langle code \rangle$ }`

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

```

1332 \newcommand*{\foralllgsentries}[3][\@glo@types]{%
1333   \expandafter\foralllglossaries\expandafter[#1]{\@this@glo@}%
1334   {%
1335     \forlgsentries[\@this@glo@]{#2}{#3}%
1336   }%
1337 }

```

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

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

where $\langle type \rangle$ is the glossary's label.

```

1338 \newcommand{\ifglossaryexists}[3]{%
1339   \ifcsundef{@glo@type@#1@out}{#3}{#2}%
1340 }

```

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 `\glstrytext` 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 redefining `\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`

```
1341 \newcommand*{\glsdetoklabel}[1]{#1}
```

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

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

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

```
1342 \newcommand{\ifglentryexists}[3]{%
1343   \ifcsundef{glo@\glstoklabel{#1}@name}{#3}{#2}%
1344 }
```

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

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

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

```
1345 \newcommand*\ifglused[3]{%
1346   \ifbool{glo@\glstoklabel{#1}@flag}{#2}{#3}%
1347 }
```

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

`\glsoifexists`

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

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

```
1348 \newcommand{\glsoifexists}[2]{%
1349   \ifglentryexists{#1}{#2}{%
1350     \PackageError{glossaries}{Glossary entry '\glstoklabel{#1}'
1351       has not been defined}{You need to define a glossary entry before you
1352       can use it.}}%
1353 }
```

`\glsoifnoexists`

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

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

```
1354 \newcommand{\glsoifnoexists}[2]{%
1355   \ifglentryexists{#1}{%
1356     \PackageError{glossaries}{Glossary entry '\glstoklabel{#1}' has already
1357       been defined}{}}{#2}%
1358 }
```

`\glsoifexistsorwarn`

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

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

```

1359 \newcommand{\glsdoifexistsorwarn}[2]{%
1360   \ifglentryexists{#1}{#2}{%
1361     \GlossariesWarning{Glossary entry ' $\glsdetoklabel{#1}$ '
1362       has not been defined}%
1363   }%
1364 }
```

\glsdoifexistsordo $\glsdoifexistsordo{\langle label \rangle}{\langle code \rangle}{\langle undef code \rangle}$

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

```

1365 \newcommand{\glsdoifexistsordo}[3]{%
1366   \ifglentryexists{#1}{#2}{%
1367     \PackageError{glossaries}{Glossary entry ' $\glsdetoklabel{#1}$ '
1368       has not been defined}{You need to define a glossary entry before you
1369       can use it.}%
1370   #3%
1371 }%
1372 }
```

\glossarynoexistsordo $\doifglossarynoexistsordo{\langle label \rangle}{\langle code \rangle}{\langle else code \rangle}$

If glossary given by $\langle label \rangle$ doesn't exist do $\langle code \rangle$ otherwise generate an error and do $\langle else code \rangle$.

```

1373 \newcommand{\doifglossarynoexistsordo}[3]{%
1374   \ifglossaryexists{#1}%
1375   {%
1376     \PackageError{glossaries}{Glossary type ' $\#1$ ' already exists}{}%
1377     #3%
1378   }%
1379   {#2}%
1380 }
```

\ifglshaschildren $\ifglshaschildren{\langle label \rangle}{\langle true part \rangle}{\langle false part \rangle}$

```

1381 \newcommand{\ifglshaschildren}[3]{%
1382   \glsdoifexists{#1}%
1383   {%
1384     \def\do@glshaschildren{#3}%
1385     \edef\@gls@thislabel{\glsdetoklabel{#1}}%
1386     \expandafter\forglentries\expandafter
1387       [\csname glo@\@gls@thislabel @type\endcsname]
1388     {\glo@label}%
1389     {%
```

```

1390      \letcs\glo@parent{glo@\glo@label @parent}%
1391      \ifdefequal\@gls@thislabel\glo@parent
1392      {%
1393          \def\do@glshaschildren{#2}%
1394          \@endfortrue
1395      }%
1396      {}%
1397  }%
1398  \do@glshaschildren
1399  }%
1400 }

```

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

```

1401 \newcommand{\ifglshasparent}[3]{%
1402   \glsdoifexists{#1}%
1403   {%
1404     \ifcsemtyp{glo@\glsdetoklabel{#1}@parent}{#3}{#2}%
1405   }%
1406 }

```

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

```

1407 \newcommand*\ifglshasdesc[3]{%
1408   \ifcsemtyp{glo@\glsdetoklabel{#1}@desc}%
1409   {#3}%
1410   {#2}%
1411 }

```

\ifglsdessuppressed \ifglsdessuppressed{<label>}{<true part>}{<false part>} Does <true part>
if the description is just \nopostdesc otherwise does <false part>.

```

1412 \newcommand*\ifglsdessuppressed[3]{%
1413   \ifcsequal{glo@\glsdetoklabel{#1}@desc}{@no@post@desc}%
1414   {#2}%
1415   {#3}%
1416 }

```

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

```

1417 \newcommand*\ifglshassymbol[3]{%
1418   \letcs{\@glo@symbol}{glo@\glsdetoklabel{#1}@symbol}%
1419   \ifdefempty\@glo@symbol
1420   {#3}%
1421   {%
1422     \ifdefequal\@glo@symbol\@gls@default@value
1423     {#3}%
1424     {#2}%
1425   }%
1426 }

```

```

\ifglshaslong \ifglshaslong{<label>}{<true part>}{<false part>}
1427 \newcommand*{\ifglshaslong}[3]{%
1428 \letcs{\@glo@long}{glo@glstdetoklabel{#1}@long}%
1429 \ifdefempty\@glo@long
1430 {#3}%
1431 {%
1432 \ifdefequal\@glo@long\@gls@default@value
1433 {#3}%
1434 {#2}%
1435 }%
1436 }

```

```

\ifglshasshort \ifglshasshort{<label>}{<true part>}{<false part>}
1437 \newcommand*{\ifglshasshort}[3]{%
1438 \letcs{\@glo@short}{glo@glstdetoklabel{#1}@short}%
1439 \ifdefempty\@glo@short
1440 {#3}%
1441 {%
1442 \ifdefequal\@glo@short\@gls@default@value
1443 {#3}%
1444 {#2}%
1445 }%
1446 }

```

```

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

```

```

1447 \newcommand*{\ifglshasfield}[4]{%
1448 \glsdoifexists{#2}%
1449 {%
1450 \letcs{\@glo@thisvalue}{glo@glstdetoklabel{#2}@#1}%

```

First check supplied field label is defined.

```

1451 \ifdef\@glo@thisvalue
1452 {%

```

Is defined, so now check if empty.

```

1453 \ifdefempty\@glo@thisvalue
1454 {%

```

Is empty, so doesn't have field set.

```

1455 #4%
1456 }%
1457 {%

```

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

```

1458 \ifdefequal\@glo@thisvalue\@gls@default@value{#4}{#3}%
1459 }%
1460 }%
1461 {%

```

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

```

1462      \@gls@fetchfield{\@gls@thisfield}{#1}%
      If \@gls@thisfield is defined, we've found a map. If not, the field supplied
      doesn't exist.
1463      \ifdef\@gls@thisfield
1464      {%
      Is defined, so now check if empty.
1465      \letcs{\@glo@thisvalue}{glo\@glsdetoklabel{#2}\@gls@thisfield}%
1466      \ifdefempty\@glo@thisvalue
1467      {%
      Is empty so field hasn't been set.
1468      #4%
1469      }%
1470      {%
      Isn't empty so check if it's been set to \@gls@default@value.
1471      \ifdequal\@glo@thisvalue\@gls@default@value{#4}{#3}%
1472      }%
1473      }%
1474      {%
      Not defined.
1475      \GlossariesWarning{Unknown entry field '#1'}%
1476      #4%
1477      }%
1478      }%
1479      }%
1480 }

```

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`

```

1481 \newcommand*{\@glo@types}{,}

```

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

```

1482 \newcommand*\@gls@provide@newglossary{%
1483   \protected@write\@auxout{}\string\providecommand\string\@newglossary[4]{}%

```

Only need to do this once.

```

1484   \let\@gls@provide@newglossary\relax
1485 }

```

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

```
1486 \newcommand*{\defglentryfmt}[2][\glsdefaulttype]{%
1487   \csgdef{gls@#1@entryfmt}{#2}%
1488 }
```

`\gls@doentryfmt`

```
1489 \newcommand*{\gls@doentryfmt}[1]{\csuse{gls@#1@entryfmt}}
```

`\@gls@forbidtext` 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.

```
1490 \newcommand*{\@gls@forbidtext}[1]{%
1491   \ifboolexpr{test {\ifdefstring{#1}{tex}}
1492             or test {\ifdefstring{#1}{TEX}}}
1493   {%
1494     \def#1{nottex}%
1495     \PackageError{glossaries}%
1496       {Forbidden '.tex' extension replaced with '.nottex'}%
1497       {I'm sorry, I can't allow you to do something so reckless.\MessageBreak
1498       Don't use '.tex' as an extension for a temporary file.}%
1499   }%
1500   {%
1501   }%
1502 }
```

`\gls@gobbleopt` Discard optional argument.

```
1503 \newcommand*{\gls@gobbleopt}{\new@ifnextchar[{\@gls@gobbleopt}{}]}
1504 \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`

```
1505 \newcommand*{\newglossary}{\@ifstar\s@newglossary\@ns@newglossary}
```


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

```
1506 \newcommand*{\s@newglossary}[2]{%
1507 \ns@newglossary[#1-glg]{#1}{#1-gls}{#1-glo}{#2}%
1508 }
```

`\ns@newglossary` Define the unstarred version.

```
1509 \newcommand*{\ns@newglossary}[5][glg]{%
1510 \doifglossarynoexistsordo{#2}%
1511 {%
```

Check if default has been set

```
1512 \ifundef\glsdefaulttype
1513 {%
1514 \gdef\glsdefaulttype{#2}%
1515 }{ }%
```

Add this to the list of glossary types:

```
1516 \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.

```
1517 \expandafter\gdef\csname glolist@#2\endcsname{,}%
```

Store the file extensions:

```
1518 \expandafter\edef\csname @glotype@#2@log\endcsname{#1}%
1519 \expandafter\edef\csname @glotype@#2@in\endcsname{#3}%
1520 \expandafter\edef\csname @glotype@#2@out\endcsname{#4}%
1521 \expandafter\@gls@forbidtexext\csname @glotype@#2@log\endcsname
1522 \expandafter\@gls@forbidtexext\csname @glotype@#2@in\endcsname
1523 \expandafter\@gls@forbidtexext\csname @glotype@#2@out\endcsname
```

Store the title:

```
1524 \expandafter\def\csname @glotype@#2@title\endcsname{#5}%
```

```
1525 \@gls@provide@newglossary
```

```
1526 \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 redefined by the user later if required (see `\defglsentry`). This may already have been defined if this has been specified as a list of acronyms.

```
1527 \ifcsundef{gls@#2@entryfmt}%
1528 {%
1529 \defglsentryfmt[#2]{\glsentryfmt}%
1530 }%
1531 {}%
```

Define sort counter if required:

```
1532 \@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.)

```

1533 \ifnextchar[{\@gls@setcounter{#2}}{%
1534   {\@gls@setcounter{#2}[\glscounter]}}%
1535 }%
1536 {%
1537   \gls@gobbleopt
1538 }%
1539 }

```

`\altnewglossary`

```

1540 \newcommand*{\altnewglossary}[3]{%
1541   \newglossary[#2-glg]{#1}{#2-gls}{#2-glo}{#3}%
1542 }

```

Only define new glossaries in the preamble:

```

1543 \@onlypreamble{\newglossary}

```

Only define new glossaries before `\makeglossaries`

```

1544 \@onlypremakeg\newglossary

```

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

`\@newglossary`

```

1545 \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`

```

1546 \def\@gls@setcounter#1[#2]{%
1547   \expandafter\def\csname @glotype@#1@counter\endcsname{#2}%
1548   \ifglxindy
1549     \GlsAddXdyCounters{#2}%
1550   \fi
1551 }

```

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

`\@gls@getcounter`

```

1552 \newcommand*{\@gls@getcounter}[1]{%
1553   \csname @glotype@#1@counter\endcsname
1554 }

```

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

```
1555 \glsdefmain
```

Define the “acronym” glossaries if required.

```
1556 \@gls@do@acronymsdef
```

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

```
1557 \@gls@do@symbolsdef
```

```
1558 \@gls@do@numbersdef
```

```
1559 \@gls@do@indexdef
```

`\newignoredglossary` 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.

```
1560 \newcommand*{\newignoredglossary}[1]{%
1561   \ifdefempty\@ignored@glossaries
1562   {%
1563     \edef\@ignored@glossaries{#1}%
1564   }%
1565   {%
1566     \eappto\@ignored@glossaries{,#1}%
1567   }%
1568   \csgdef{glolist@#1}{,}%
1569   \ifcsundef{gls@#1@entryfmt}%
1570   {%
1571     \defglsentryfmt[#1]{\glsentryfmt}%
1572   }%
1573   {}%
1574   \ifdefempty\@gls@nohyperlist
1575   {%
1576     \renewcommand*{\@gls@nohyperlist}{#1}%
1577   }%
1578   {%
1579     \eappto\@gls@nohyperlist{,#1}%
1580   }%
1581 }
```

`@ignored@glossaries` List of ignored glossaries.

```
1582 \newcommand*{\@ignored@glossaries}{}
```

`\ifignoredglossary` Tests if the given glossary is an ignored glossary. Expansion is used in case the first argument is a control sequence.

```
1583 \newcommand*{\ifignoredglossary}[3]{%
1584   \edef\@gls@igtype{#1}%
1585   \expandafter\DTLifinlist\expandafter
1586   {\@gls@igtype}{\@ignored@glossaries}{#2}{#3}%
1587 }
```

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.

```
1588 \define@key{glossentry}{name}{%
1589 \def\@glo@name{#1}%
1590 }
```

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`.

```
1591 \define@key{glossentry}{description}{%
1592 \def\@glo@desc{#1}%
1593 }
```

descriptionplural

```
1594 \define@key{glossentry}{descriptionplural}{%
1595 \def\@glo@descplural{#1}%
1596 }
```

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>`.

```
1597 \define@key{glossentry}{sort}{%
1598 \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.

```
1599 \define@key{glossentry}{text}{%
1600 \def\@glo@text{#1}%
1601 }
```

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.

```

1602 \define@key{glossentry}{plural}{%
1603 \def\@glo@plural{#1}%
1604 }

```

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.

```

1605 \define@key{glossentry}{first}{%
1606 \def\@glo@first{#1}%
1607 }

```

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.

```

1608 \define@key{glossentry}{firstplural}{%
1609 \def\@glo@firstplural{#1}%
1610 }

```

`\@gls@default@value`

```

1611 \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).

```

1612 \define@key{glossentry}{symbol}{%
1613 \def\@glo@symbol{#1}%
1614 }

```

symbolplural

```

1615 \define@key{glossentry}{symbolplural}{%
1616 \def\@glo@symbolplural{#1}%
1617 }

```

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

```

1618 \define@key{glossentry}{type}{%
1619 \def\@glo@type{#1}}

```

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

```

1620 \define@key{glossentry}{counter}{%
1621 \ifcsundef{c@#1}%
1622 {%
1623 \PackageError{glossaries}%

```

```

1624     {There is no counter called ‘#1’}%
1625     {%
1626         The counter key should have the name of a valid counter
1627         as its value%
1628     }%
1629 }%
1630 {%
1631     \def\@glo@counter{#1}%
1632 }%
1633 }

```

see The see key specifies a list of cross-references

```

1634 \define@key{glossentry}{see}{%
1635     \gls@checkseeallowed
1636     \def\@glo@see{#1}%
1637     \@glo@seeautonumberlist
1638 }

```

gls@checkseeallowed

```

1639 \newcommand*{\gls@checkseeallowed}{%
1640     \PackageError{glossaries}%
1641     {‘see’ key may only be used after \string\makeglossaries\space
1642     or \string\makenoidxglossaries}%
1643     {You must use \string\makeglossaries\space
1644     or \string\makenoidxglossaries\space before defining
1645     any entries that have a ‘see’ key}%
1646 }

```

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

```

1647 \define@key{glossentry}{parent}{%
1648     \def\@glo@parent{#1}}

```

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

```

1649 \define@choicekey{glossentry}{nonumberlist}[\val\nr]{true,false}[true]{%
1650     \ifcase\nr\relax
1651         \def\@glo@prefix{\glsnonextpages}%
1652     \else
1653         \def\@glo@prefix{\glsnextpages}%
1654     \fi
1655 }

```

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

user1

```

1656 \define@key{glossentry}{user1}{%
1657     \def\@glo@useri{#1}%
1658 }

```

user2

```
1659 \define@key{glossentry}{user2}{%  
1660   \def\@glo@userii{#1}%  
1661 }
```

user3

```
1662 \define@key{glossentry}{user3}{%  
1663   \def\@glo@useriii{#1}%  
1664 }
```

user4

```
1665 \define@key{glossentry}{user4}{%  
1666   \def\@glo@useriv{#1}%  
1667 }
```

user5

```
1668 \define@key{glossentry}{user5}{%  
1669   \def\@glo@userv{#1}%  
1670 }
```

user6

```
1671 \define@key{glossentry}{user6}{%  
1672   \def\@glo@uservi{#1}%  
1673 }
```

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.

```
1674 \define@key{glossentry}{short}{%  
1675   \def\@glo@short{#1}%  
1676 }
```

shortplural This key is provided for use by \newacronym.

```
1677 \define@key{glossentry}{shortplural}{%  
1678   \def\@glo@shortpl{#1}%  
1679 }
```

long This key is provided for use by \newacronym.

```
1680 \define@key{glossentry}{long}{%  
1681   \def\@glo@long{#1}%  
1682 }
```

longplural This key is provided for use by \newacronym.

```
1683 \define@key{glossentry}{longplural}{%  
1684   \def\@glo@longpl{#1}%  
1685 }
```

```

\@glsnoname  Define command to generate error if name key is missing.
1686 \newcommand*{\@glsnoname}{%
1687   \PackageError{glossaries}{name key required in
1688   \string\newglossaryentry\space for entry '\@glo@label'}{You
1689   haven't specified the entry name}}

\@glsnodesc  Define command to generate error if description key is missing.
1690 \newcommand*\@glsnodesc{%
1691   \PackageError{glossaries}
1692   {%
1693     description key required in \string\newglossaryentry\space
1694     for entry '\@glo@label'%
1695   }%
1696   {%
1697     You haven't specified the entry description%
1698   }%
1699 }%

\@glsdefaultplural  Now obsolete. Don't use.
1700 \newcommand*\@glsdefaultplural{}}

\@gls@missingnumberlist  Define a command to generate warning when numberlist not set.
1701 \newcommand*\@gls@missingnumberlist[1]{%
1702   ??%
1703   \ifglssavenumberlist
1704     \GlossariesWarning{Missing number list for entry '#1'.
1705     Maybe makeglossaries + rerun required.}%
1706   \else
1707     \PackageError{glossaries}%
1708     {Package option 'savenumberlist=true' required.}%
1709     {%
1710       You must use the 'savenumberlist' package option
1711       to reference location lists.%
1712     }%
1713   \fi
1714 }

\@glsdefaultsort  Define command to set default sort.
1715 \newcommand*\@glsdefaultsort{\@glo@name}

\gls@level  Register to increment entry levels.
1716 \newcount\gls@level

\@gls@noexpand@field
1717 \newcommand{\@gls@noexpand@field}[3]{%
1718   \expandafter\global\expandafter
1719   \let\csname glo@#1@#2\endcsname#3%
1720 }

```


gls@noexpand@fields

```
1721 \newcommand{\@gls@noexpand@fields}[4]{%
1722   \ifcsdef{gls@assign@#3@field}
1723   {%
1724     \ifdefequal{#4}{\@gls@default@value}%
1725     {%
1726       \edef\@gls@value{\expandonce{#1}}%
1727       \csuse{gls@assign@#3@field}{#2}{\@gls@value}%
1728     }%
1729     {%
1730       \csuse{gls@assign@#3@field}{#2}{#4}%
1731     }%
1732   }%
1733   {%
1734     \ifdefequal{#4}{\@gls@default@value}%
1735     {%
1736       \edef\@gls@value{\expandonce{#1}}%
1737       \@gls@noexpand@field{#2}{#3}{\@gls@value}%
1738     }%
1739     {%
1740       \@gls@noexpand@field{#2}{#3}{#4}%
1741     }%
1742   }%
1743 }
```

\@@gls@expand@field

```
1744 \newcommand{\@@gls@expand@field}[3]{%
1745   \expandafter
1746   \protected@xdef\csname glo@#1@#2\endcsname{#3}%
1747 }
```

@gls@expand@fields

```
1748 \newcommand{\@gls@expand@fields}[4]{%
1749   \ifcsdef{gls@assign@#3@field}
1750   {%
1751     \ifdefequal{#4}{\@gls@default@value}%
1752     {%
1753       \edef\@gls@value{\expandonce{#1}}%
1754       \csuse{gls@assign@#3@field}{#2}{\@gls@value}%
1755     }%
1756     {%
1757       \expandafter\@gls@startswitexpandonce#4\relax\relax\gls@endcheck
1758     }%
1759     \@gls@expand@field{#2}{#3}{#4}%
1760   }%
1761   {%
1762     \csuse{gls@assign@#3@field}{#2}{#4}%
1763   }%
```

```

1764     }%
1765 }%
1766 {%
1767     \ifdefequal{#4}{\@gls@default@value}%
1768     {%
1769         @@gls@expand@field{#2}{#3}{#1}%
1770     }%
1771     {%
1772         @@gls@expand@field{#2}{#3}{#4}%
1773     }%
1774 }%
1775 }

```

tartswithexpandonce

```

1776 \def\@gls@expandonce{\expandonce}
1777 \def\@gls@startswithexpandonce#1#2\gls@endcheck#3#4{%
1778     \def\@gls@tmp{#1}%
1779     \ifdefequal{\@gls@expandonce}{\@gls@tmp}{#3}{#4}%
1780 }

```

`\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.

```
1781 \let\gls@assign@field\@gls@expand@fields
```

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

```

1782 \newcommand*\glsexpandfields{%
1783     \let\gls@assign@field\@gls@expand@fields
1784 }

```

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

```

1785 \newcommand*\glsnoexpandfields{%
1786     \let\gls@assign@field\@gls@noexpand@fields
1787 }

```

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

```
1788 \newrobustcmd{\newglossaryentry}[2]{%
```

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

```

1789     \glsdoifnoexists{#1}%
1790     {%
1791         \gls@defglossaryentry{#1}{#2}%

```

```

1792 }%
1793 }

docnewglossaryentry The definition of \newglossaryentry is changed at the start of the document
environment.
1794 \newcommand*{\gls@defdocnewglossaryentry}{%
1795 \let\newglossaryentry\new@glossaryentry
1796 }

provideglossaryentry Like \newglossaryentry but does nothing if the entry has already been de-
fined.
1797 \newrobustcmd{\provideglossaryentry}[2]{%
1798 \ifglentryexists{#1}%
1799 }%
1800 {%
1801 \gls@defglossaryentry{#1}{#2}%
1802 }%
1803 }
1804 \@onlypreamble{\provideglossaryentry}

\new@glossaryentry For use in document environment.
1805 \newrobustcmd{\new@glossaryentry}[2]{%
1806 \ifundef\@gls@deffile
1807 {%
1808 \global\newwrite\@gls@deffile
1809 \immediate\openout\@gls@deffile=\jobname.glsdefs
1810 }%
1811 }%
1812 \ifglentryexists{#1}{}%
1813 {%
1814 \gls@defglossaryentry{#1}{#2}%
1815 }%
1816 \@gls@writedef{#1}%
1817 }
1818 \AtBeginDocument
1819 {
1820 \makeatletter
1821 \InputIfFileExists{\jobname.glsdefs}{\input}{\typeout}%
1822 \makeatother
1823 \gls@defdocnewglossaryentry
1824 }
1825 \AtEndDocument{\ifdef\@gls@deffile{\closeout\@gls@deffile}{}}

\@gls@writedef Writes glossary entry definition to \@gls@deffile.
1826 \newcommand*{\@gls@writedef}[1]{%
1827 \immediate\write\@gls@deffile
1828 {%
1829 \string\ifglentryexists{#1}{\glspercentchar^^J%

```

```

1830 \expandafter\@gobble\string\{\glpercentchar^^J%
1831 \string\gls@defglossaryentry{\glsetoklabel{#1}}\glpercentchar^^J%
1832 \expandafter\@gobble\string\{\glpercentchar%
1833 }%

Write key value information:

1834 \@for\@gls@map:=\@gls@keymap\do
1835 {%
1836 \edef\glo@value{\expandafter\expandonce
1837 \csname glo@\glsetoklabel{#1}\@expandafter
1838 \@secondoftwo\@gls@map\endcsname}%
1839 \@onelevel@sanitize\glo@value
1840 \immediate\write\@gls@deffile
1841 {%
1842 \expandafter\@firstoftwo\@gls@map
1843 =\expandafter\@gobble\string\{\glo@value\expandafter\@gobble\string\},%
1844 \glpercentchar%
1845 }%
1846 }%

Provide hook:

1847 \glswritedefhook
1848 \immediate\write\@gls@deffile
1849 {%
1850 \glpercentchar^^J%
1851 \expandafter\@gobble\string\}\glpercentchar^^J%
1852 \expandafter\@gobble\string\}\glpercentchar%
1853 }%
1854 }

```

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

```

1855 \newcommand*{\@gls@keymap}{%
1856 {name}{name},%
1857 {sort}{sortvalue},% unescaped sort value
1858 {type}{type},%
1859 {first}{first},%
1860 {firstplural}{firstpl},%
1861 {text}{text},%
1862 {plural}{plural},%
1863 {description}{desc},%
1864 {descriptionplural}{descplural},%
1865 {symbol}{symbol},%
1866 {symbolplural}{symbolplural},%
1867 {user1}{useri},%
1868 {user2}{userii},%
1869 {user3}{useriii},%
1870 {user4}{useriv},%
1871 {user5}{userv},%
1872 {user6}{uservi},%

```

```

1873 {long}{long},%
1874 {longplural}{longpl},%
1875 {short}{short},%
1876 {shortplural}{shortpl},%
1877 {counter}{counter},%
1878 {parent}{parent}%
1879 }

```

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

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

```
1880 \newcommand*{\@gls@fetchfield}[2]{%
```

Ensure user field name is fully expanded

```
1881 \edef\@gls@thisval{#2}%
```

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

```

1882 \@for\@gls@map:=\@gls@keymap\do{%
1883 \edef\@this@key{\expandafter\@firstoftwo\@gls@map}%
1884 \ifdefequal{\@this@key}{\@gls@thisval}%
1885 {%

```

Found it.

```
1886 \edef#1{\expandafter\@secondoftwo\@gls@map}%
```

Break out of loop.

```

1887 \@endfortrue
1888 }%
1889 {}%
1890 }%
1891 }

```

`\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.

```
1892 \newcommand*{\glsaddstoragekey}{\@ifstar\@sglsaddstoragekey\@glsaddstoragekey}
```

Starred version switches on expansion for this key.

```

1893 \newcommand*{\@sglsaddstoragekey}[1]{%
1894 \key@ifundefined{glossentry}{#1}%
1895 {%
1896 \expandafter\newcommand\expandafter*\expandafter
1897 {\csname gls@assign@#1@field\endcsname}[2]{%
1898 \@gls@expand@field{##1}{#1}{##2}%
1899 }%
1900 }%
1901 {}%

```

```

1902 \glsaddstoragekey{#1}%
1903 }

```

Unstarred version doesn't override default expansion.

```

1904 \newcommand*{\glsaddstoragekey}[3]{%

```

Check the specified key doesn't already exist.

```

1905 \key@ifundefined{glossentry}{#1}%
1906 {%

```

Set up the key.

```

1907 \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
1908 \appto\gls@keymap{, #1}{#1}}%

```

Set the default value.

```

1909 \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%

```

Assignment code.

```

1910 \appto\@newglossaryentryposthook{%
1911 \letcs{\@glo@tmp}{@glo@#1}%
1912 \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
1913 }%

```

Define the no-link commands.

```

1914 \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
1915 }%
1916 {%
1917 \PackageError{glossaries}{Key ‘#1’ already exists}{}%
1918 }%
1919 }

```

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

Allow user to add their own custom keys.

```

1920 \newcommand*{\glsaddkey}{\@ifstar\sglsaddkey\glsaddkey}

```

Starred version switches on expansion for this key.

```

1921 \newcommand*{\sglsaddkey}[1]{%
1922 \key@ifundefined{glossentry}{#1}%
1923 {%
1924 \expandafter\newcommand\expandafter*\expandafter
1925 {\csname gls@assign@#1@field\endcsname}[2]{%
1926 \@@gls@expand@field{##1}{#1}{##2}%
1927 }%
1928 }%
1929 }%
1930 \glsaddkey{#1}%
1931 }

```

Unstarred version doesn't override default expansion.

```
1932 \newcommand*{\@glsaddkey}[7]{%
```

Check the specified key doesn't already exist.

```
1933   \key@ifundefined{glossentry}{#1}%
```

```
1934   {%
```

Set up the key.

```
1935     \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
```

```
1936     \appto\@gls@keymap{, {#1}{#1}}%
```

Set the default value.

```
1937     \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
```

Assignment code.

```
1938     \appto\@newglossaryentryposthook{%
```

```
1939       \letcs{\@glo@tmp}{@glo@#1}%
```

```
1940       \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
```

```
1941     }%
```

Define the no-link commands.

```
1942     \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
```

```
1943     \newcommand*{#4}[1]{\@Gls@entry@field{##1}{#1}}%
```

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

```
1944     \ifcsdef{@gls@user@#1@}{%
```

```
1945     {%
```

```
1946       \PackageError{glossaries}%
```

```
1947       {Can't define '\string#5' as helper command
```

```
1948       '\expandafter\string\csname @gls@user@#1@\endcsname' already exists}%
```

```
1949     {}%
```

```
1950   }%
```

```
1951   {%
```

```
1952     \expandafter\newcommand\expandafter*\expandafter
```

```
1953     {\csname @gls@user@#1@\endcsname}[2][1]{%
```

```
1954       \new@ifnextchar[%
```

```
1955         {\csuse{@gls@user@#1@}{##1}{##2}}%
```

```
1956         {\csuse{@gls@user@#1@}{##1}{##2}[1]}}%
```

```
1957     \csdef{@gls@user@#1@}##1##2[##3]{%
```

```
1958       \@gls@field@link{##1}{##2}{#3{##2}##3}%
```

```
1959     }%
```

```
1960     \newrobustcmd*{#5}{%
```

```
1961       \expandafter\@gls@hyp@opt\csname @gls@user@#1@\endcsname}%
```

```
1962     }%
```

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

```
1963     \ifcsdef{@Gls@user@#1@}{%
```

```
1964     {%
```

```
1965       \PackageError{glossaries}%
```

```
1966       {Can't define '\string#6' as helper command
```

```
1967       '\expandafter\string\csname @Gls@user@#1@\endcsname' already exists}%
```

```

1968     {}%
1969   }%
1970   {%

1971   \expandafter\newcommand\expandafter*\expandafter
1972     {\csname @Gls@user@#1\endcsname}[2][\{%
1973     \new@ifnextchar[%
1974     {\csuse{@Gls@user@#1@}{##1}{##2}}}%
1975     {\csuse{@Gls@user@#1@}{##1}{##2}[]}}}%
1976   \csdef{@Gls@user@#1@}##1##2[##3]{%
1977     \@gls@field@link{##1}{##2}{#4{##2}##3}%
1978   }%
1979   \newrobustcmd*{#6}{%
1980     \expandafter\@gls@hyp@opt\csname @Gls@user@#1\endcsname}%
1981   }%

```

Finally the all caps version:

```

1982   \ifcsdef{@GLS@user@#1@}%
1983   {%
1984     \PackageError{glossaries}%
1985     {Can't define 'string#7' as helper command
1986     '\expandafter\string\csname @GLS@user@#1\endcsname' already exists}%
1987   }%
1988   }%
1989   {%

1990   \expandafter\newcommand\expandafter*\expandafter
1991     {\csname @GLS@user@#1\endcsname}[2][\{%
1992     \new@ifnextchar[%
1993     {\csuse{@GLS@user@#1@}{##1}{##2}}}%
1994     {\csuse{@GLS@user@#1@}{##1}{##2}[]}}}%
1995   \csdef{@GLS@user@#1@}##1##2[##3]{%
1996     \@gls@field@link{##1}{##2}{\mfirstucMakeUppercase{#3{##2}##3}}}%
1997   }%
1998   \newrobustcmd*{#7}{%
1999     \expandafter\@gls@hyp@opt\csname @GLS@user@#1\endcsname}%
2000   }%
2001   }%
2002   {%
2003     \PackageError{glossaries}{Key '#1' already exists}{}%
2004   }%
2005 }

```

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

```

2006 \newcommand{\glsfieldxdef}[3]{%
2007   \glsdoifexists{#1}%
2008   {%

```



```

2009 \edef\@glo@label{\glsdetoklabel{#1}}%
2010 \ifcsdef{glo@\@glo@label @#2}%
2011 {%
2012 \expandafter\xdef\csname glo@\@glo@label @#2\endcsname{#3}%
2013 }%
2014 {%
2015 \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2016 }%
2017 }%
2018 }

```

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

```

2019 \newcommand{\glsfieldeddef}[3]{%
2020 \glsdoifexists{#1}%
2021 {%
2022 \edef\@glo@label{\glsdetoklabel{#1}}%
2023 \ifcsdef{glo@\@glo@label @#2}%
2024 {%
2025 \expandafter\edef\csname glo@\@glo@label @#2\endcsname{#3}%
2026 }%
2027 {%
2028 \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2029 }%
2030 }%
2031 }

```

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

```

2032 \newcommand{\glsfieldgdef}[3]{%
2033 \glsdoifexists{#1}%
2034 {%
2035 \edef\@glo@label{\glsdetoklabel{#1}}%
2036 \ifcsdef{glo@\@glo@label @#2}%
2037 {%
2038 \expandafter\gdef\csname glo@\@glo@label @#2\endcsname{#3}%
2039 }%
2040 {%
2041 \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2042 }%
2043 }%
2044 }

```

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

```

2045 \newcommand{\glsfielddef}[3]{%
2046   \glsdoifexists{#1}%
2047   {%
2048     \edef\@glo@label{\glsdetoklabel{#1}}%
2049     \ifcsdef{glo@\@glo@label @#2}%
2050     {%
2051       \expandafter\def\csname glo@\@glo@label @#2\endcsname{#3}%
2052     }%
2053     {%
2054       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2055     }%
2056   }%
2057 }

```

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

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

```

2058 \newcommand{\glsfieldfetch}[3]{%
2059   \glsdoifexists{#1}%
2060   {%
2061     \edef\@glo@label{\glsdetoklabel{#1}}%
2062     \ifcsdef{glo@\@glo@label @#2}%
2063     {%
2064       \letcs#3{glo@\@glo@label @#2}%
2065     }%
2066     {%
2067       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2068     }%
2069   }%
2070 }

```

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

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

```

2071 \newcommand{\ifglsfieldeq}[5]{%
2072   \glsdoifexists{#1}%
2073   {%
2074     \edef\@glo@label{\glsdetoklabel{#1}}%
2075     \ifcsdef{glo@\@glo@label @#2}%
2076     {%
2077       \ifcsstring{glo@\@glo@label @#2}{#3}{#4}{#5}%
2078     }%
2079     {%
2080       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2081     }%

```

```
2082 }%
2083 }
```

`\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.

```
2084 \newcommand{\ifglsfielddefeq}[5]{%
2085   \glsdoifexists{#1}%
2086   {%
2087     \edef\@glo@label{\glsdetoklabel{#1}}%
2088     \ifcsdef{glo@\@glo@label @#2}%
2089     {%
2090       \expandafter\ifdefstrequal
2091       \csname glo@\@glo@label @#2\endcsname{#3}{#4}{#5}%
2092     }%
2093     {%
2094       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2095     }%
2096   }%
2097 }
```

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

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

```
2098 \newcommand{\ifglsfieldcseq}[5]{%
2099   \glsdoifexists{#1}%
2100   {%
2101     \edef\@glo@label{\glsdetoklabel{#1}}%
2102     \ifcsdef{glo@\@glo@label @#2}%
2103     {%
2104       \ifcsstrequal{glo@\@glo@label @#2}{#3}{#4}{#5}%
2105     }%
2106     {%
2107       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2108     }%
2109   }%
2110 }
```

`\glswritedefhook`

```
2111 \newcommand*{\glswritedefhook}{}%
```

`\gls@assign@desc`

```
2112 \newcommand*{\gls@assign@desc}[1]{%
2113   \gls@assign@field{#1}{desc}{\@glo@desc}%
2114   \gls@assign@field{\@glo@desc}{#1}{descplural}{\@glo@descplural}%
2115 }
```

2115 }

ongnewglossaryentry

```
2116 \newcommand{\longnewglossaryentry}[3]{%
2117   \glsdoifnoexists{#1}%
2118   {%
2119     \bgroup
2120     \let\@org@newglossaryentryprehook\@newglossaryentryprehook
2121     \long\def\@newglossaryentryprehook{%
2122       \long\def\@glo@desc{#3\leavevmode\unskip\nopostdesc}%
2123       \@org@newglossaryentryprehook
2124     }%
2125     \renewcommand*\@gls@assign@desc[1]{%
2126       \global\cslet{glo@glstdetoklabel{#1}@desc}{\@glo@desc}%
2127       \global\cslet{glo@glstdetoklabel{#1}@descplural}{\@glo@desc}%
2128     }
2129     \gls@defglossaryentry{#1}{#2}%
2130   \egroup
2131 }
2132 }
```

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

2133 \@onlypreamble{\longnewglossaryentry}

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

```
2134 \newcommand{\longprovideglossaryentry}[3]{%
2135   \ifglstryexists{#1}{}%
2136   {\longnewglossaryentry{#1}{#2}{#3}}%
2137 }
2138 \@onlypreamble{\longprovideglossaryentry}
```

gls@defglossaryentry

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

Defines a new entry without checking if it already exists.

2139 \newcommand{\gls@defglossaryentry}[2]{%

Store label

2140 \edef\@glo@label{\glstdetoklabel{#1}}%

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

2141 \let\glslabel\@glo@label

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

2142 \let\@glo@name\@glsnname

2143 \let\@glo@desc\@glsnodesc

2144 \let\@glo@descplural\@gls@default@value

```

2145 \let\@glo@type\@gls@default@value
2146 \let\@glo@symbol\@gls@default@value

2147 \let\@glo@symbolplural\@gls@default@value
2148 \let\@glo@text\@gls@default@value
2149 \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.)

```

2150 \let\@glo@first\@gls@default@value
2151 \let\@glo@firstplural\@gls@default@value

```

Set the default sort:

```

2152 \let\@glo@sort\@gls@default@value

```

Set the default counter:

```

2153 \let\@glo@counter\@gls@default@value
2154 \def\@glo@see{}%
2155 \def\@glo@parent{}%
2156 \def\@glo@prefix{}%

2157 \def\@glo@useri{}%
2158 \def\@glo@userii{}%
2159 \def\@glo@useriii{}%
2160 \def\@glo@useriv{}%
2161 \def\@glo@userv{}%
2162 \def\@glo@uservi{}%

2163 \def\@glo@short{}%
2164 \def\@glo@shortpl{}%
2165 \def\@glo@long{}%
2166 \def\@glo@longpl{}%

```

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

```

2167 \@newglossaryentryprehook

```

Extract key-val information from third parameter:

```

2168 \setkeys{glossentry}{#2}%

```

Check there is a default glossary.

```

2169 \ifundef\glsdefaulttype
2170 {%
2171   \PackageError{glossaries}%
2172   {No default glossary type (have you used ‘nomain’?)}%
2173   {If you use package option ‘nomain’ you must define
2174    a new glossary before you can define entries}%
2175 }%
2176 {}%

```

Assign type. This must be fully expandable

```
2177 \gls@assign@field{\glsdefaulttype}{\@glo@label}{type}{\@glo@type}%
2178 \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.

```
2179 \ifcsundef{glolist@\@glo@type}%
2180 {%
2181     \PackageError{glossaries}%
2182     {Glossary type ‘\@glo@type’ has not been defined}%
2183     {You need to define a new glossary type, before making entries
2184      in it}%
2185 }%
2186 {%
```

Check if it's an ignored glossary

```
2187 \ifignoredglossary\@glo@type
2188 {%
```

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

```
2189 \ifx\@glo@desc\@glsnodesc
2190 \let\@glo@desc\@empty
2191 \fi
2192 }%
2193 {%
2194 }%
2195 \protected@edef\@glolist@{\csname glolist@\@glo@type\endcsname}%
2196 \expandafter\xdef\csname glolist@\@glo@type\endcsname{%
2197 \@glolist@{\@glo@label},}%
2198 }%
```

Initialise level to 0.

```
2199 \gls@level=0\relax
```

Has this entry been assigned a parent?

```
2200 \ifx\@glo@parent\@empty
```

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

```
2201 \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%
2202 \else
```

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

```
2203 \ifdefequal\@glo@label\@glo@parent%
2204 {%
2205     \PackageError{glossaries}{Entry ‘\@glo@label’ can’t be its own parent}{}%
2206     \def\@glo@parent{}%
2207     \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%
2208 }%
2209 {%
```

Check the parent exists:

```
2210 \ifglsentryexists{\@glo@parent}%
2211 {%
```

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

```

2212      \expandafter\xdef\csname glo@\@glo@label @parent\endcsname{%
2213      \@glo@parent}%

Determine level.

2214      \gls@level=\csname glo@\@glo@parent @level\endcsname\relax
2215      \advance\gls@level by 1\relax

If name hasn't been specified, use same as the parent name
2216      \ifx\@glo@name\@gls@noname
2217      \expandafter\let\expandafter\@glo@name
2218      \csname glo@\@glo@parent @name\endcsname

If name and plural haven't been specified, use same as the parent
2219      \ifx\@glo@plural\@gls@default@value
2220      \expandafter\let\expandafter\@glo@plural
2221      \csname glo@\@glo@parent @plural\endcsname
2222      \fi
2223      \fi
2224      }%
2225      {%

Parent doesn't exist, so issue an error message and change this entry to have no
parent
2226      \PackageError{glossaries}%
2227      {%
2228      Invalid parent '\@glo@parent'
2229      for entry '\@glo@label' - parent doesn't exist%
2230      }%
2231      {%
2232      Parent entries must be defined before their children%
2233      }%
2234      \def\@glo@parent{%
2235      \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{%
2236      }%
2237      }%
2238      \fi

Set the level for this entry
2239      \expandafter\xdef\csname glo@\@glo@label @level\endcsname{\number\gls@level}%

Define commands associated with this entry:
2240      \gls@assign@field{\@glo@name}{\@glo@label}{sortvalue}{\@glo@sort}%
2241      \letcs\@glo@sort{glo@\@glo@label @sortvalue}%
2242      \gls@assign@field{\@glo@name}{\@glo@label}{text}{\@glo@text}%
2243      \expandafter\gls@assign@field\expandafter
2244      {\csname glo@\@glo@label @text\endcsname\gls@pluralsuffix}%
2245      {\@glo@label}{plural}{\@glo@plural}%
2246      \expandafter\gls@assign@field\expandafter
2247      {\csname glo@\@glo@label @text\endcsname}%
2248      {\@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.

```

2249 \ifx\@glo@first\@gls@default@value
2250 \expandafter\gls@assign@field\expandafter
2251 {\csname glo@\@glo@label @plural\endcsname}%
2252 {\@glo@label}{firstpl}{\@glo@firstplural}%
2253 \else
2254 \expandafter\gls@assign@field\expandafter
2255 {\csname glo@\@glo@label @first\endcsname\glspluralsuffix}%
2256 {\@glo@label}{firstpl}{\@glo@firstplural}%
2257 \fi

2258 \ifcsundef{@glotype@\@glo@type @counter}%
2259 {%
2260 \def\@glo@defaultcounter{\@glscounter}%
2261 }%
2262 {%
2263 \letcs\@glo@defaultcounter{@glotype@\@glo@type @counter}%
2264 }%
2265 \gls@assign@field{\@glo@defaultcounter}{\@glo@label}{counter}{\@glo@counter}%
2266 \gls@assign@field{}{\@glo@label}{useri}{\@glo@useri}%
2267 \gls@assign@field{}{\@glo@label}{userii}{\@glo@userii}%
2268 \gls@assign@field{}{\@glo@label}{useriii}{\@glo@useriii}%
2269 \gls@assign@field{}{\@glo@label}{useriv}{\@glo@useriv}%
2270 \gls@assign@field{}{\@glo@label}{userv}{\@glo@userv}%
2271 \gls@assign@field{}{\@glo@label}{uservi}{\@glo@uservi}%
2272 \gls@assign@field{}{\@glo@label}{short}{\@glo@short}%
2273 \gls@assign@field{}{\@glo@label}{shortpl}{\@glo@shortpl}%
2274 \gls@assign@field{}{\@glo@label}{long}{\@glo@long}%
2275 \gls@assign@field{}{\@glo@label}{longpl}{\@glo@longpl}%
2276 \ifx\@glo@name\@glsnoname
2277 \@glsnoname
2278 \let\@glo@name\@gls@default@value
2279 \fi
2280 \gls@assign@field{}{\@glo@label}{name}{\@glo@name}%

```

Set default numberlist if not defined:

```

2281 \ifcsundef{glo@\@glo@label @numberlist}%
2282 {%
2283 \csxdef{glo@\@glo@label @numberlist}{%
2284 \noexpand\@gls@missingnumberlist{\@glo@label}}%
2285 }%
2286 {}%

```

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.

```

2287 \def\@glo@@desc{\@glo@first}%
2288 \ifx\@glo@desc\@glo@@desc
2289 \let\@glo@desc\@glo@first

```



```

2290 \fi
2291 \ifx\@glo@desc\@glsnodedesc
2292 \@glsnodedesc
2293 \let\@glo@desc\@gls@default@value
2294 \fi
2295 \gls@assign@desc{\@glo@label}%

```

Set the sort key for this entry:

```

2296 \@gls@defsort{\@glo@type}{\@glo@label}%

2297 \def\@glo@@symbol{\@glo@text}%
2298 \ifx\@glo@symbol\@glo@@symbol
2299 \let\@glo@symbol\@glo@text
2300 \fi
2301 \gls@assign@field{\relax}{\@glo@label}{symbol}{\@glo@symbol}%
2302 \expandafter
2303 \gls@assign@field\expandafter
2304 {\csname glo@\@glo@label @symbol\endcsname}
2305 {\@glo@label}{symbolplural}{\@glo@symbolplural}%

```

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

```

2306 \expandafter\xdef\csname glo@\@glo@label @flagfalse\endcsname{%
2307 \noexpand\global
2308 \noexpand\let\expandafter\noexpand
2309 \csname ifglo@\@glo@label @flag\endcsname\noexpand\iffalse
2310 }%
2311 \expandafter\xdef\csname glo@\@glo@label @flagtrue\endcsname{%
2312 \noexpand\global
2313 \noexpand\let\expandafter\noexpand
2314 \csname ifglo@\@glo@label @flag\endcsname\noexpand\iftrue
2315 }%
2316 \csname glo@\@glo@label @flagfalse\endcsname

```

Sort out any cross-referencing if required.

```

2317 \ifdefined\@glo@see
2318 {}%
2319 {%
2320 \protected@edef\@do@glsssee{%
2321 \noexpand\@gls@fixbraces\noexpand\@glo@list\@glo@see
2322 \noexpand\@nil
2323 \noexpand\expandafter\noexpand\@glsssee\noexpand\@glo@list{\@glo@label}}%
2324 \@do@glsssee
2325 }%

```

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

```

2326 \ifignoredglossary\@glo@type
2327 {%
2328 \csdef{glo@\@glo@label @index}{}%
2329 }
2330 {%

```

```

2331 \do@glo@storeentry{\@glo@label}%
2332 }%

Define entry counters if enabled:
2333 \@newglossaryentry@defcounters

Add end hook in case another package wants to add extra keys.
2334 \@newglossaryentryposthook
2335 }

\glossaryentryprehook Allow extra information to be added to glossary entries:
2336 \newcommand*{\@newglossaryentryprehook}{}

\glossaryentryposthook Allow extra information to be added to glossary entries:
2337 \newcommand*{\@newglossaryentryposthook}{}

\glossaryentry@defcounters
2338 \newcommand*{\@newglossaryentry@defcounters}{}

\glsmoveentry Moves entry whose label is given by first argument to the glossary named in the
second argument.
2339 \newcommand*{\glsmoveentry}[2]{%
2340 \edef\@glo@thislabel{\glsdetoklabel{#1}}%
2341 \edef\glo@type{\csname glo@\@glo@thislabel @type\endcsname}%
2342 \def\glo@list{,}%
2343 \forglsentries[\glo@type]{\glo@label}%
2344 {%

2345 \ifdefequal\@glo@thislabel\glo@label
2346 {\eappto\glo@list{\glo@label,}}%
2347 }%
2348 \cslet{glolist@\glo@type}{\glo@list}%
2349 \csdef{glo@\@glo@thislabel @type}{#2}%
2350 }

\@glossaryentryfield Indicate what command should be used to display each entry in the glossary.
(This enables the glossaries-accsupp package to use \accsuppglossaryentryfield
instead.)
2351 \ifglxindy
2352 \newcommand*{\@glossaryentryfield}{\string\glossentry}
2353 \else
2354 \newcommand*{\@glossaryentryfield}{\string\glossentry}
2355 \fi

\glossarysubentryfield Indicate what command should be used to display each subentry in the glossary.
(This enables the glossaries-accsupp package to use \accsuppglossarysubentryfield
instead.)
2356 \ifglxindy

```

```

2357 \newcommand*{\@glossarysubentryfield}{%
2358 \string\@subglossentry}
2359 \else
2360 \newcommand*{\@glossarysubentryfield}{%
2361 \string\subglossentry}
2362 \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.)

```
2363 \newcommand{\@glo@storeentry}[1]{%
```

Escape makeindex/xindy special characters in the label:

```

2364 \edef\@glo@esclabel{#1}%
2365 \@gls@checkmkidxchars\@glo@esclabel

```

Get the sort string and escape any special characters

```

2366 \protected@edef\@glo@sort{\csname glo@#1@sort\endcsname}%
2367 \@gls@checkmkidxchars\@glo@sort

```

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

```
2368 \@gls@checkmkidxchars\@glo@prefix
```

Get the parent, if one exists

```
2369 \edef\@glo@parent{\csname glo@#1@parent\endcsname}%
```

Write the information to the glossary file.

```
2370 \ifglsxindy
```

Store using xindy syntax.

```
2371 \ifx\@glo@parent\@empty
```

Entry doesn't have a parent

```

2372 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2373 (\string"\@glo@sort\string" %
2374 \string"\@glo@prefix\@glossaryentryfield{\@glo@esclabel}\string") %
2375 }%
2376 \else

```

Entry has a parent

```

2377 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2378 \csname glo@\@glo@parent @index\endcsname
2379 (\string"\@glo@sort\string" %
2380 \string"\@glo@prefix\@glossarysubentryfield
2381 {\csname glo@#1@level\endcsname}{\@glo@esclabel}\string") %
2382 }%
2383 \fi
2384 \else

```

Store using makeindex syntax.

```

2385   \ifx\@glo@parent\@empty
      Sanitize \@glo@prefix
2386   \@onelevel@sanitize\@glo@prefix

Entry doesn't have a parent
2387   \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2388     \@glo@sort\@gls@actualchar\@glo@prefix
2389     \@glossaryentryfield{\@glo@esclabel}%
2390   }%
2391   \else

Entry has a parent
2392   \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2393     \csname glo@\@glo@parent @index\endcsname\@gls@levelchar
2394     \@glo@sort\@gls@actualchar\@glo@prefix
2395     \@glossarysubentryfield
2396     {\csname glo@#1@level\endcsname}{\@glo@esclabel}%
2397   }%
2398   \fi
2399 \fi
2400 }
```

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.

```

\gls@ifnotmeasuring
2401 \AtBeginDocument{%
2402   \@ifpackageloaded{amsmath}%
2403   {\let\gls@ifnotmeasuring\@gls@ifnotmeasuring}%
2404   }%
2405 }
2406 \newcommand*\@gls@ifnotmeasuring[1]{%
2407   \ifmeasuring@
2408   \else
2409     #1%
2410   \fi
2411 }
2412 \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.

```

2413 \newcommand*\glsreset[1]{%
```

```

2414 \gls@ifnotmeasuring
2415 {%
2416     \glsdoifexists{#1}%
2417     {%
2418         \@glsreset{#1}%
2419     }%
2420 }%
2421 }

```

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

```

2422 \newcommand*{\glslocalreset}[1]{%
2423     \gls@ifnotmeasuring
2424     {%
2425         \glsdoifexists{#1}%
2426         {%
2427             \@glslocalreset{#1}%
2428         }%
2429     }%
2430 }

```

`\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.

```

2431 \newcommand*{\glsunset}[1]{%
2432     \gls@ifnotmeasuring
2433     {%
2434         \glsdoifexists{#1}%
2435         {%
2436             \@glsunset{#1}%
2437         }%
2438     }%
2439 }

```

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

```

2440 \newcommand*{\glslocalunset}[1]{%
2441     \gls@ifnotmeasuring
2442     {%
2443         \glsdoifexists{#1}%
2444         {%
2445             \@glslocalunset{#1}%
2446         }%
2447     }%
2448 }

```

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

```

2449 \newcommand*{\@glslocalunset}{\@glslocalunset}

```

`\@@glslocalunset` Local unset without checks.

```

2450 \newcommand*{\@@glslocalunset}[1]{%
2451   \expandafter\let\csname ifglo@\glsdetoklabel{#1}@flag\endcsname\iftrue
2452 }

```

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

```

2453 \newcommand*{\@glsunset}{\@@glsunset}

```

`\@@glsunset` Global unset without checks.

```

2454 \newcommand*{\@@glsunset}[1]{%
2455   \expandafter\global\csname glo@\glsdetoklabel{#1}@flagtrue\endcsname
2456 }

```

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

```

2457 \newcommand*{\@glslocalreset}{\@@glslocalreset}

```

`\@@glslocalreset` Local reset without checks.

```

2458 \newcommand*{\@@glslocalreset}[1]{%
2459   \expandafter\let\csname ifglo@\glsdetoklabel{#1}@flag\endcsname\iffalse
2460 }

```

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

```

2461 \newcommand*{\@glsreset}{\@@glsreset}

```

`\@@glsreset` Global reset without checks.

```

2462 \newcommand*{\@@glsreset}[1]{%
2463   \expandafter\global\csname glo@\glsdetoklabel{#1}@flagfalse\endcsname
2464 }

```

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

`\glsresetall`

```

2465 \newcommand*{\glsresetall}[1][\@glo@types]{%
2466   \forallglsentries[#1]{\@glsentry}%
2467   {%
2468     \glsreset{\@glsentry}%
2469   }%
2470 }

```

As above, but with only a local effect:

`\glslocalresetall`

```

2471 \newcommand*{\glslocalresetall}[1][\@glo@types]{%
2472   \forallglsentries[#1]{\@glsentry}%
2473   {%

```

```

2474 \glslocalreset{\@glsentry}%
2475 }%
2476 }

```

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

`\glsunsetall`

```

2477 \newcommand*{\glsunsetall}[1][\@glo@types]{%
2478 \forallglsentries[#1]{\@glsentry}%
2479 {%
2480 \glsunset{\@glsentry}%
2481 }%
2482 }

```

As above, but with only a local effect:

`\glslocalunsetall`

```

2483 \newcommand*{\glslocalunsetall}[1][\@glo@types]{%
2484 \forallglsentries[#1]{\@glsentry}%
2485 {%
2486 \glslocalunset{\@glsentry}%
2487 }%
2488 }

```

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 \LaTeX 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.

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

```

2489 \newcommand*{\@newglossaryentry@defcounters}{%
2490 \csdef{glo@\@glo@label @currcount}{0}%
2491 \csdef{glo@\@glo@label @prevcount}{0}%
2492 }

```

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

```

2493 \newcommand*{\glsenableentrycount}{%

```

Enable new entry fields.

```

2494 \let\@newglossaryentry@defcounters\@newglossaryentry@defcounters

```

Disable `\newglossaryentry` in the document environment.

```

2495 \renewcommand*{\gls@defdocnewglossaryentry}{%
2496   \renewcommand*{\newglossaryentry}[2]{%
2497     \PackageError{glossaries}{\string\newglossaryentry\space
2498       may only be used in the preamble when entry counting has
2499       been activated}{If you use \string\glsenableentrycount\space
2500       you must place all entry definitions in the preamble not in
2501       the document environment}%
2502   }%
2503 }%
```

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

```

2504 \newcommand*{\glsentrycurrcount}[1]{%
2505   \ifcsundef{glo@\glsdetoklabel{##1}@currcount}%
2506     {0}{\@gls@entry@field{##1}{currcount}}%
2507 }%
2508 \newcommand*{\glsentryprevcount}[1]{%
2509   \ifcsundef{glo@\glsdetoklabel{##1}@prevcount}%
2510     {0}{\@gls@entry@field{##1}{prevcount}}%
2511 }%
```

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

```

2512 \renewcommand*{\@glsunset}[1]{%
2513   \@@glsunset{##1}%
2514   \@gls@increment@currcount{##1}%
2515 }%
2516 \renewcommand*{\@glslocalunset}[1]{%
2517   \@@glslocalunset{##1}%
2518   \@gls@local@increment@currcount{##1}%
2519 }%
2520 \renewcommand*{\@glsreset}[1]{%
2521   \@@glsreset{##1}%
2522   \csgdef{glo@\glsdetoklabel{##1}@currcount}{0}%
2523 }%
2524 \renewcommand*{\@glslocalreset}[1]{%
2525   \@@glslocalreset{##1}%
2526   \csdef{glo@\glsdetoklabel{##1}@currcount}{0}%
2527 }%
```

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.)

```

2528 \def\@cgl@##1##2[##3]{%
2529   \ifnum\glsentryprevcount{##2}=1\relax
2530     \cglformat{##2}{##3}%
2531     \glsunset{##2}%
2532   \else
2533     \@gls@{##1}{##2}[##3]%
2534   \fi
```


2535 }%

Similarly for the analogous commands. No case change plural:

```
2536 \def\@cglsp1@##1##2[##3]{%
2537   \ifnum\glentryprevcount{##2}=1\relax
2538     \cglsp1format{##2}{##3}%
2539     \glunset{##2}%
2540   \else
2541     \@glsp1@{##1}{##2}[##3]%
2542   \fi
2543 }%
```

First letter uppercase singular:

```
2544 \def\@cGls1@##1##2[##3]{%
2545   \ifnum\glentryprevcount{##2}=1\relax
2546     \cGls1format{##2}{##3}%
2547     \glunset{##2}%
2548   \else
2549     \@Gls1@{##1}{##2}[##3]%
2550   \fi
2551 }%
```

First letter uppercase plural:

```
2552 \def\@cGlspl1@##1##2[##3]{%
2553   \ifnum\glentryprevcount{##2}=1\relax
2554     \cGlspl1format{##2}{##3}%
2555     \glunset{##2}%
2556   \else
2557     \@Glspl1@{##1}{##2}[##3]%
2558   \fi
2559 }%
```

Write information to aux file at the end of the document

```
2560 \AtEndDocument{\@gls@write@entrycounts}%
```

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

```
2561 \renewcommand*{\@gls@entry@count}[2]{%
2562   \csgdef{glo@\glsdetoklabel{##1}@prevcount}{##2}%
2563 }%
```

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

```
2564 \let\glsenableentrycount\relax
2565 }
2566 \@onlypreamble\glsenableentrycount
```

increment@currcount

```
2567 \newcommand*{\@gls@increment@currcount}[1]{%
2568   \csxdef{glo@\glsdetoklabel{##1}@currcount}{%
2569     \number\numexpr\glentrycurrcount{##1}+1}%
2570 }
```

increment@currcount

```
2571 \newcommand*{\@gls@local@increment@currcount}[1]{%
2572   \csedef{glo@glsetoklabel{#1}@currcount}{%
2573     \number\numexpr\glsetentrycurrcount{#1}+1}%
2574 }
```

s@write@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.)

```
2575 \newcommand*{\@gls@write@entrycounts}{%
2576   \immediate\write\@auxout
2577   {\string\providecommand*{\string\@gls@entry@count}[2]{}}%
2578   \forallglsetentries{\@glsetentry}{%
2579     \ifglsetused{\@glsetentry}%
2580     {\immediate\write\@auxout
2581       {\string\@gls@entry@count{\@glsetentry}{\glsetentrycurrcount{\@glsetentry}}}%
2582     }%
2583   }%
2584 }
```

\@gls@entry@count

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

```
2585 \newcommand*{\@gls@entry@count}[2]{}%
```

\cglset

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.)

```
2586 \newrobustcmd*{\cglset}{\@gls@hyp@opt\@cglset}
```

\@cglset

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

```
2587 \newcommand*{\@cglset}[2][]{%
2588   \new@ifnextchar[{\@cglset@{#1}{#2}}{\@cglset@{#1}{#2}[]}%
2589 }
```

\@cglset@

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

```
2590 \def\@cglset@#1#2[#3]{%
2591   \GlossariesWarning{\string\cglset\space is defaulting to
2592     \string\gls\space since you haven't enabled entry counting}%
2593   \@gls@{#1}{#2}[#3]%
2594 }
```

\cglsetformat

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

```
2595 \newcommand*{\cglsetformat}[2]{%
```

```

2596 \ifglshaslong{#1}{\glentrylong{#1}}{\glentryfirst{#1}}#2%
2597 }

```

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

```

2598 \newrobustcmd*{\cGls}{\@gls@hyp@opt\@cGls}

```

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

```

2599 \newcommand*{\@cGls}[2] [] {%
2600   \new@ifnextchar[{\@cGls@{#1}{#2}}{\@cGls@{#1}{#2} []}%
2601 }

```

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

```

2602 \def\@cGls@#1#2[#3] {%
2603   \GlossariesWarning{\string\cGls\space is defaulting to
2604     \string\Gls\space since you haven't enabled entry counting}%
2605   \@Gls@{#1}{#2}[#3]%
2606 }

```

`\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.

```

2607 \newcommand*{\cGlsformat}[2] {%
2608   \ifglshaslong{#1}{\Glentrylong{#1}}{\Glentryfirst{#1}}#2%
2609 }

```

`\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.)

```

2610 \newrobustcmd*{\cglsp1}{\@gls@hyp@opt\@cglsp1}

```

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

```

2611 \newcommand*{\@cglsp1}[2] [] {%
2612   \new@ifnextchar[{\@cglsp1@{#1}{#2}}{\@cglsp1@{#1}{#2} []}%
2613 }

```

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

```

2614 \def\@cglsp1@#1#2[#3] {%
2615   \GlossariesWarning{\string\cglsp1\space is defaulting to
2616     \string\glsp1\space since you haven't enabled entry counting}%
2617   \@glsp1@{#1}{#2}[#3]%
2618 }

```

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

```

2619 \newcommand*{\cglspformat}[2]{%
2620   \ifglshaslong{#1}{\glentrylongpl{#1}}{\glentryfirstplural{#1}}#2%
2621 }

```

`\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.)

```

2622 \newrobustcmd*{\cGlspl}{\@gls@hyp@opt\@cGlspl}

```

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

```

2623 \newcommand*{\@cGlspl}[2][{}]{%
2624   \new@ifnextchar[\@cGlspl@{#1}{#2}}{\@cGlspl@{#1}{#2}[{}]}%
2625 }

```

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

```

2626 \def\@cGlspl@#1#2[#3]{%
2627   \GlossariesWarning{\string\cGlspl\space is defaulting to
2628     \string\Glspl\space since you haven't enabled entry counting}%
2629   \@Glspl@{#1}{#2}[#3]%
2630 }

```

`\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.

```

2631 \newcommand*{\cGlsplformat}[2]{%
2632   \ifglshaslong{#1}{\Glsentrylongpl{#1}}{\Glsentryfirstplural{#1}}#2%
2633 }

```

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).

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

`\loadglsentries`

```
2634 \newcommand*{\loadglsentries}[2][\@gls@default]{%
2635   \let\@gls@default\glsdefaulttype
2636   \def\glsdefaulttype{#1}\input{#2}%
2637   \let\glsdefaulttype\@gls@default
2638 }
```

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

```
2639 \@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`

```
2640 \newcommand*{\glstextformat}[1]{#1}
```

`\glsentryfmt` As from version 3.11a, the way in which an entry is displayed is now governed by `\glsentryfmt`. This doesn’t take any arguments. The required information is set by commands like `\gls`. To ensure backward compatibility, the default use the old `\glsdisplay` and `\glsdisplayfirst` style of commands

```
2641 \newcommand*{\glsentryfmt}{%
2642   \@@gls@default@entryfmt\glsdisplayfirst\glsdisplay
2643 }
```

Format that provides backwards compatibility:

```
2644 \newcommand*{\@@gls@default@entryfmt}[2]{%
2645   \ifdefempty\glscustomtext
2646     {%
2647       \glsifplural
2648       {%
```

Plural form

```
2649       \glscapscase
2650       {%
```

Don’t adjust case

```
2651       \ifglsused\glslabel
2652       {%
```

Subsequent use

```
2653       #2{\glsentryplural{\glslabel}}}%
2654       {\glsentrydescplural{\glslabel}}}%
```

```

2655         {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2656     }%
2657     {%

```

First use

```

2658         #1{\glsentryfirstplural{\glslabel}}%
2659         {\glsentrydescplural{\glslabel}}%
2660         {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2661     }%
2662 }%
2663 {%

```

Make first letter upper case

```

2664     \ifglsused\glslabel
2665     {%

```

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.)

```

2666     \ifbool{glscompatible-3.07}%
2667     {%
2668         \protected@edef\@glo@etext{%
2669             #2{\glsentryplural{\glslabel}}%
2670             {\glsentrydescplural{\glslabel}}%
2671             {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2672         \xmakefirstuc\@glo@etext
2673     }%
2674     {%
2675         #2{\Glsentryplural{\glslabel}}%
2676         {\glsentrydescplural{\glslabel}}%
2677         {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2678     }%
2679 }%
2680 {%

```

First use

```

2681     \ifbool{glscompatible-3.07}%
2682     {%
2683         \protected@edef\@glo@etext{%
2684             #1{\glsentryfirstplural{\glslabel}}%
2685             {\glsentrydescplural{\glslabel}}%
2686             {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2687         \xmakefirstuc\@glo@etext
2688     }%
2689     {%
2690         #1{\Glsentryfirstplural{\glslabel}}%
2691         {\glsentrydescplural{\glslabel}}%
2692         {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2693     }%
2694 }%

```

2695 }%
 2696 {%

Make all upper case

2697 \ifglsused\glslabel
 2698 {%

Subsequent use

2699 \mfirstucMakeUppercase{#2{\glsentryplural{\glslabel}}}%
 2700 {\glsentrydescplural{\glslabel}}}%
 2701 {\glsentrysymbolplural{\glslabel}}{\glsinsert}}}%
 2702 }%
 2703 {%

First use

2704 \mfirstucMakeUppercase{#1{\glsentryfirstplural{\glslabel}}}%
 2705 {\glsentrydescplural{\glslabel}}}%
 2706 {\glsentrysymbolplural{\glslabel}}{\glsinsert}}}%
 2707 }%
 2708 }%
 2709 }%
 2710 {%

Singular form

2711 \glscapscase
 2712 {%

Don't adjust case

2713 \ifglsused\glslabel
 2714 {%

Subsequent use

2715 #2{\glsentrytext{\glslabel}}}%
 2716 {\glsentrydesc{\glslabel}}}%
 2717 {\glsentrysymbol{\glslabel}}{\glsinsert}}}%
 2718 }%
 2719 {%

First use

2720 #1{\glsentryfirst{\glslabel}}}%
 2721 {\glsentrydesc{\glslabel}}}%
 2722 {\glsentrysymbol{\glslabel}}{\glsinsert}}}%
 2723 }%
 2724 }%
 2725 {%

Make first letter upper case

2726 \ifglsused\glslabel
 2727 {%

Subsequent use

2728 \ifbool{glscompatible-3.07}%
 2729 {%

```

2730         \protected@edef\@glo@etext{%
2731             #2{\glsentrytext{\glslabel}}}%
2732             {\glsentrydesc{\glslabel}}}%
2733             {\glsentrysymbol{\glslabel}}{\glsinsert}}}%
2734         \xmakefirstuc\@glo@etext
2735     }%
2736     {%
2737         #2{\Glsentrytext{\glslabel}}}%
2738         {\glsentrydesc{\glslabel}}}%
2739         {\glsentrysymbol{\glslabel}}{\glsinsert}}%
2740     }%
2741 }%
2742 {%

```

First use

```

2743     \ifbool{glscompatible-3.07}%
2744     {%
2745         \protected@edef\@glo@etext{%
2746             #1{\glsentryfirst{\glslabel}}}%
2747             {\glsentrydesc{\glslabel}}}%
2748             {\glsentrysymbol{\glslabel}}{\glsinsert}}}%
2749         \xmakefirstuc\@glo@etext
2750     }%
2751     {%
2752         #1{\Glsentryfirst{\glslabel}}}%
2753         {\glsentrydesc{\glslabel}}}%
2754         {\glsentrysymbol{\glslabel}}{\glsinsert}}%
2755     }%
2756 }%
2757 }%
2758 {%

```

Make all upper case

```

2759     \ifglsused\glslabel
2760     {%

```

Subsequent use

```

2761         \mfirstucMakeUppercase{#2{\glsentrytext{\glslabel}}}%
2762         {\glsentrydesc{\glslabel}}}%
2763         {\glsentrysymbol{\glslabel}}{\glsinsert}}}%
2764     }%
2765     {%

```

First use

```

2766         \mfirstucMakeUppercase{#1{\glsentryfirst{\glslabel}}}%
2767         {\glsentrydesc{\glslabel}}}%
2768         {\glsentrysymbol{\glslabel}}{\glsinsert}}}%
2769     }%
2770 }%
2771 }%
2772 }%

```



```

2773  {%
      Custom text provided in \glsdisp
2774  \ifglsused{\glslabel}%
2775  {%
      Subsequent use
2776      #2{\glscustomtext}%
2777      {\glsentrydesc{\glslabel}}%
2778      {\glsentrysymbol{\glslabel}}{%}%
2779  }%
2780  {%
      First use
2781      #1{\glscustomtext}%
2782      {\glsentrydesc{\glslabel}}%
2783      {\glsentrysymbol{\glslabel}}{%}%
2784  }%
2785  }%
2786 }

```

`\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.

```

2787 \newcommand*{\glsenentryfmt}{%
2788   \ifdefempty\glscustomtext
2789   {%
2790     \glsifplural
2791     {%
      Plural form
2792       \glscapscase
2793       {%
      Don't adjust case
2794       \ifglsused\glslabel
2795       {%
      Subsequent use
2796       \glsentryplural{\glslabel}\glsinsert
2797       }%
2798       {%
      First use
2799       \glsentryfirstplural{\glslabel}\glsinsert
2800       }%
2801       }%
2802       {%
      Make first letter upper case
2803       \ifglsused\glslabel
2804       {%

```

Subsequent use.

```
2805      \Glsentryplural{\glslabel}\glsinsert
2806      }%
2807      {%
```

First use

```
2808      \Glsentryfirstplural{\glslabel}\glsinsert
2809      }%
2810      }%
2811      {%
```

Make all upper case

```
2812      \ifglsused\glslabel
2813      {%
```

Subsequent use

```
2814      \mfirstucMakeUppercase
2815      {\Glsentryplural{\glslabel}\glsinsert}%
2816      }%
2817      {%
```

First use

```
2818      \mfirstucMakeUppercase
2819      {\Glsentryfirstplural{\glslabel}\glsinsert}%
2820      }%
2821      }%
2822      }%
2823      {%
```

Singular form

```
2824      \glscapscase
2825      {%
```

Don't adjust case

```
2826      \ifglsused\glslabel
2827      {%
```

Subsequent use

```
2828      \Glsentrytext{\glslabel}\glsinsert
2829      }%
2830      {%
```

First use

```
2831      \Glsentryfirst{\glslabel}\glsinsert
2832      }%
2833      }%
2834      {%
```

Make first letter upper case

```
2835      \ifglsused\glslabel
2836      {%
```

Subsequent use

```
2837      \Glsentrytext{\glslabel}\glsinsert
2838      }%
2839      {%
```

First use

```
2840      \Glsentryfirst{\glslabel}\glsinsert
2841      }%
2842      }%
2843      {%
```

Make all upper case

```
2844      \ifglsused\glslabel
2845      {%
```

Subsequent use

```
2846      \mfirstucMakeUppercase{\glsentrytext{\glslabel}\glsinsert}%
2847      }%
2848      {%
```

First use

```
2849      \mfirstucMakeUppercase{\glsentryfirst{\glslabel}\glsinsert}%
2850      }%
2851      }%
2852      }%
2853      }%
2854      {%
```

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

```
2855      \glscustomtext\glsinsert
2856      }%
2857 }
```

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

```
2858 \newcommand*{\glsngenacfmt}{%
2859   \ifdefempty\glscustomtext
2860   {%
2861     \ifglsused\glslabel
2862     {%
```

Subsequent use:

```
2863     \glsifplural
2864     {%
```

Subsequent plural form:

```
2865     \glscapscase
2866     {%
```

Subsequent plural form, don't adjust case:

```
2867      \acronymfont{\glsentryshortpl{\glslabel}}\glsinsert
2868      }%
2869      {%
```

Subsequent plural form, make first letter upper case:

```
2870      \acronymfont{\Glsentryshortpl{\glslabel}}\glsinsert
2871      }%
2872      {%
```

Subsequent plural form, all caps:

```
2873      \mfirstucMakeUppercase
2874      {\acronymfont{\glsentryshortpl{\glslabel}}\glsinsert}%
2875      }%
2876      }%
2877      {%
```

Subsequent singular form

```
2878      \glscapscase
2879      {%
```

Subsequent singular form, don't adjust case:

```
2880      \acronymfont{\glsentryshort{\glslabel}}\glsinsert
2881      }%
2882      {%
```

Subsequent singular form, make first letter upper case:

```
2883      \acronymfont{\Glsentryshort{\glslabel}}\glsinsert
2884      }%
2885      {%
```

Subsequent singular form, all caps:

```
2886      \mfirstucMakeUppercase
2887      {\acronymfont{\glsentryshort{\glslabel}}\glsinsert}%
2888      }%
2889      }%
2890      }%
2891      {%
```

First use:

```
2892      \glsifplural
2893      {%
```

First use plural form:

```
2894      \glscapscase
2895      {%
```

First use plural form, don't adjust case:

```
2896      \genplacrfullformat{\glslabel}{\glsinsert}%
2897      }%
2898      {%
```

First use plural form, make first letter upper case:

```
2899      \Genplacrfullformat{\glslabel}{\glsinsert}%
2900      }%
2901      {%
```

First use plural form, all caps:

```
2902      \mfirstucMakeUppercase
2903      {\genplacrfullformat{\glslabel}{\glsinsert}}%
2904      }%
2905      }%
2906      {%
```

First use singular form

```
2907      \glscapscase
2908      {%
```

First use singular form, don't adjust case:

```
2909      \genacrfullformat{\glslabel}{\glsinsert}%
2910      }%
2911      {%
```

First use singular form, make first letter upper case:

```
2912      \Genacrfullformat{\glslabel}{\glsinsert}%
2913      }%
2914      {%
```

First use singular form, all caps:

```
2915      \mfirstucMakeUppercase
2916      {\genacrfullformat{\glslabel}{\glsinsert}}%
2917      }%
2918      }%
2919      }%
2920      }%
2921      {%
```

User supplied text.

```
2922      \glscustomtext
2923      }%
2924 }
```

\genacrfullformat

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

The full format used by \gls`genacfmt` (singular).

```
2925 \newcommand*{\genacrfullformat}[2]{%
2926   \glsentrylong{#1}#2\space
2927   (\protect\firstacronymfont{\glsentryshort{#1}})%
2928 }
```

\Genacrfullformat

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

As above but makes the first letter upper case.

```
2929 \newcommand*{\Genacrfullformat}[2]{%
2930   \protected@edef\gls@text{\genacrfullformat{#1}{#2}}%
2931   \xmakefirstuc\gls@text
2932 }
```

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

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

```
2933 \newcommand*{\genplacrfullformat}[2]{%
2934   \glsentrylongpl{#1}#2\space
2935   (\protect\firstacronymfont{\glsentryshortpl{#1}})%
2936 }
```

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

As above but makes the first letter upper case.

```
2937 \newcommand*{\Genplacrfullformat}[2]{%
2938   \protected@edef\gls@text{\genplacrfullformat{#1}{#2}}%
2939   \xmakefirstuc\gls@text
2940 }
```

`\glsdisplayfirst` Deprecated. Kept for backward compatibility.

```
2941 \newcommand*{\glsdisplayfirst}[4]{#1#4}
```

`\glsdisplay` Deprecated. Kept for backward compatibility.

```
2942 \newcommand*{\glsdisplay}[4]{#1#4}
```

`\defglsdisplay` Deprecated. Kept for backward compatibility.

```
2943 \newcommand*{\defglsdisplay}[2][\glsdefaulttype]{%
2944   \GlossariesWarning{\string\defglsdisplay\space is now obsolete.^^J
2945   Use \string\defglsentryfmt\space instead}%
2946   \expandafter\def\csname gls@#1@display\endcsname##1##2##3##4{#2}%
2947   \edef\@gls@doentrydef{%
2948     \noexpand\defglsentryfmt[#1]{%
2949       \noexpand\ifcsdef{gls@#1@displayfirst}%
2950       {%
2951         \noexpand\@gls@default@entryfmt
2952         {\noexpand\csuse{gls@#1@displayfirst}}%
2953         {\noexpand\csuse{gls@#1@display}}%
2954       }%
2955       {%
2956         \noexpand\@gls@default@entryfmt
2957         {\noexpand\glsdisplayfirst}%
2958         {\noexpand\csuse{gls@#1@display}}%
2959       }%
2960     }%
2961   }
```

```

2960 }%
2961 }%
2962 \@gls@doentrydef
2963 }

```

`\defglsdisplayfirst` Deprecated. Kept for backward compatibility.

```

2964 \newcommand*{\defglsdisplayfirst}[2][\glsdefaultttype]{%
2965   \GlossariesWarning{\string\defglsdisplayfirst\space is now obsolete.^^J
2966   Use \string\defglsentryfmt\space instead}%
2967   \expandafter\def\csname gls@#1@displayfirst\endcsname##1##2##3##4{#2}%
2968   \edef\@gls@doentrydef{%
2969     \noexpand\defglsentryfmt[#1]{%
2970       \noexpand\ifcsdef{gls@#1@display}%
2971       {%
2972         \noexpand\@gls@default@entryfmt
2973         {\noexpand\csuse{gls@#1@displayfirst}}}%
2974         {\noexpand\csuse{gls@#1@display}}}%
2975       }%
2976       {%
2977         \noexpand\@gls@default@entryfmt
2978         {\noexpand\csuse{gls@#1@displayfirst}}}%
2979         {\noexpand\glsdisplay}%
2980       }%
2981     }%
2982   }%
2983   \@gls@doentrydef
2984 }

```

1.11.1 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.

```

2985 \define@key{glslink}{counter}{%
2986   \ifcsundef{c@#1}%
2987   {%
2988     \PackageError{glossaries}%

```

```

2989     {There is no counter called ‘#1’}%
2990     {%
2991         The counter key should have the name of a valid counter
2992         as its value%
2993     }%
2994 }%
2995 {%
2996     \def\@gls@counter{#1}%
2997 }%
2998 }

```

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.

```

2999 \define@key{glslink}{format}{%
3000     \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.

```

3001 \define@boolkey{glslink}{hyper}[true]{}

```

Initialise hyper key.

```

3002 \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.

```

3003 \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.

```

3004 \newcommand*\glslinkvar[3]{#1}

```

`\glsifhyper` Now deprecated.

```

3005 \newcommand*\glsifhyper[2]{%
3006     \glslinkvar{#1}{#2}{#1}%
3007     \GlossariesWarning{\string\glsifhyper\space is deprecated. Did
3008     you mean \string\glsifhyperon\space or \string\glslinkvar?}%
3009 }

```


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

```
3010 \newcommand*{\@gls@hyp@opt}[1]{%
3011 \let\glslinkvar\@firstofthree
3012 \let\@gls@hyp@opt@cs#1\relax
3013 \@ifstar{\s@gls@hyp@opt}%
3014 {\@ifnextchar+{\@firstoftwo{\p@gls@hyp@opt}}{#1}}%
3015 }
```

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

```
3016 \newcommand*{\s@gls@hyp@opt}[1][ ]{%
3017 \let\glslinkvar\@secondofthree
3018 \@gls@hyp@opt@cs[hyper=false,#1]}
```

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

```
3019 \newcommand*{\p@gls@hyp@opt}[1][ ]{%
3020 \let\glslinkvar\@thirdofthree
3021 \@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`

```
3022 \newrobustcmd*{\glslink}{%
3023 \@gls@hyp@opt\@gls@@link
3024 }
```

`\@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.

```
3025 \newcommand*{\@gls@@link}[3][ ]{%
3026 \glsdoifexistsordo{#2}%
3027 {%
3028 \let\do@gls@link@checkfirsthyper\relax
3029 \@gls@link[#1]{#2}{#3}%
3030 }{%
```

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

```

3031 \glstextformat{#3}%
3032 }%

3033 \glspostlinkhook
3034 }

```

\glspostlinkhook

```

3035 \newcommand*{\glspostlinkhook}{}
3036 % \end{macrocode}
3037 \end{macro}
3038 %
3039 %
3040 %\begin{macro}{\@gls@link@checkfirsthyper}
3041 % Check for first use and switch off \gloskey[glslink]{hyper} key
3042 % if hyperlink not wanted. (Should be off if first use and
3043 % hyper=false is on or if first use and both the entry is in an acronym
3044 % list and the acrfootnote setting is on.)
3045 % This assumes the glossary type is stored in \cs{glstype} and the
3046 % label is stored in \cs{glslabel}.
3047 %\changes{4.08}{2014-07-30}{new}
3048 % \begin{macrocode}
3049 \newcommand*{\@gls@link@checkfirsthyper}{%
3050 \ifglsused{\glslabel}%
3051 {%
3052 }%
3053 {%
3054 \gls@checkisacronymlist\glstype
3055 \ifglshyperfirst
3056 \if@glsisacronymlist
3057 \ifglsacrfootnote
3058 \KV@glslink@hyperfalse
3059 \fi
3060 \fi
3061 \else
3062 \KV@glslink@hyperfalse
3063 \fi
3064 }%

```

Allow user to hook into this

```

3065 \glslinkcheckfirsthyperhook
3066 }

```

checkfirsthyperhook Allow used to hook into the \@gls@link@checkfirsthyper macro

```

3067 \newcommand*{\glslinkcheckfirsthyperhook}{}

```

\glslinkpostsetkeys

```

3068 \newcommand*{\glslinkpostsetkeys}{}

```

`\glsifhyperon` Check the value of the hyper key:

```

3069 \newcommand{\glsifhyperon}[2]{\ifKV@glslink@hyper#1\else#2\fi}

```

`\glsdisablehyperinlist` Disable hyperlink if in the “nohyper” list.

```

3070 \newcommand*\do@gl:disablehyperinlist{%
3071   \expandafter\DTLifinlist\expandafter{\glstype}{\@gls@nohyperlist}%
3072   {\KV@glslink@hyperfalse}{}}%
3073 }

```

`\@gls@link`

```

3074 \def\@gls@link[#1]#2#3{%

```

Inserting `\leavevmode` suggested by Donald Arseneau (avoids problem with `tabularx`).

```

3075   \leavevmode
3076   \edef\glslabel{\glsdetoklabel{#2}}%

```

Save options in `\@gls@link@opts` and label in `\@gls@link@label`

```

3077   \def\@gls@link@opts{#1}%
3078   \let\@gls@link@label\glslabel

```

```

3079   \def\@glsnumberformat{\glsnumberformat}%
3080   \edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%

```

If this is in one of the “nohypertypes” glossaries, suppress the hyperlink by default

```

3081   \edef\glstype{\csname glo@\glslabel @type\endcsname}%

```

Save original setting

```

3082   \let\org@ifKV@glslink@hyper\ifKV@glslink@hyper

```

Switch off hyper setting if the glossary type has been identified in `nohyperlist`.

```

3083   \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`.

```

3084   \do@gls@link@checkfirsthyper
3085   \setkeys{glslink}{#1}%

```

Add a hook for the user to customise things after the keys have been set.

```

3086   \glslinkpostsetkeys

```

Store the entry’s counter in `\theglentrycounter`

```

3087   \@gls@saveentrycounter

```

Define sort key if necessary:

```

3088   \@gls@setsort{\glslabel}%

```

(De-tok’ing done by `\@do@wrglossary`)

```

3089   \@do@wrglossary{#2}%
3090   \ifKV@glslink@hyper
3091     \@glslink{\glo@linkprefix\glslabel}{\glstextformat{#3}}%
3092   \else

```

```

3093     \glstextformat{#3}%
3094     \fi

    Restore original setting
3095     \let\ifKV@glslink@hyper\org@ifKV@glslink@hyper
3096 }

\glolinkprefix
3097 \newcommand*{\glolinkprefix}{glo:}

\glsentrycounter    Set default value of entry counter
3098 \def\glsentrycounter{\glscounter}%

\ls@saveentrycounter    Need to check if using equation counter in align environment:
3099 \newcommand*{\@gls@saveentrycounter}{%
3100   \def\@gls@Hcounter{}%

    Are we using equation counter?
3101   \ifthenelse{\equal{\@gls@counter}{equation}}{%
3102     {

        If we're in align environment, \xatlevel@ will be defined. (Can't test for
        \@currentvir as may be inside an inner environment.)
3103     \ifcsundef{xatlevel@}%
3104     {%
3105       \edef\theglsentrycounter{\expandafter\noexpand
3106         \csname the\@gls@counter\endcsname}%
3107     }%
3108     {%
3109       \ifx\xatlevel@\@empty
3110         \edef\theglsentrycounter{\expandafter\noexpand
3111           \csname the\@gls@counter\endcsname}%
3112       \else
3113         \savecounters@
3114         \advance\c@equation by 1\relax
3115         \edef\theglsentrycounter{\csname the\@gls@counter\endcsname}%

        Check if hyperref version of this counter
3116         \ifcsundef{theH\@gls@counter}%
3117         {%
3118           \def\@gls@Hcounter{\theglsentrycounter}%
3119         }%
3120         {%
3121           \def\@gls@Hcounter{\csname theH\@gls@counter\endcsname}%
3122         }%
3123         \protected@edef\theHglentrycounter{\@gls@Hcounter}%
3124         \restorecounters@
3125       \fi
3126     }%
3127   }%
3128   {%

```

Not using equation counter so no special measures:

```

3129 \edef\theglentrycounter{\expandafter\noexpand
3130 \csname the\@gls@counter\endcsname}%
3131 }%
```

Check if hyperref version of this counter

```

3132 \ifx\@gls@Hcounter\@empty
3133 \ifcsundef{theH\@gls@counter}%
3134 {%
3135 \def\theHglentrycounter{\theglentrycounter}%
3136 }%
3137 {%
3138 \protected@edef\theHglentrycounter{\expandafter\noexpand
3139 \csname theH\@gls@counter\endcsname}%
3140 }%
3141 \fi
3142 }
```

`\@set@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.

```

3143 \def\@set@glo@numformat#1#2#3#4{%
3144 \expandafter\@glo@check@mkidxrangear#3\@nil
3145 \protected@edef#1{%
3146 \@glo@prefix setentrycounter[#4]{#2}%
3147 \expandafter\string\csname\@glo@suffix\endcsname
3148 }%
3149 \@gls@checkmkidxchars#1%
3150 }
```

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.

```

3151 \def\@glo@check@mkidxrangear#1#2\@nil{%
3152 \if#1(\relax
3153 \def\@glo@prefix{(%
3154 \if\relax#2\relax
3155 \def\@glo@suffix{glsnumberformat}%
3156 \else
3157 \def\@glo@suffix{#2}%
3158 \fi
3159 \else
3160 \if#1)\relax
3161 \def\@glo@prefix{)%
3162 \if\relax#2\relax
```

```

3163     \def\@glo@suffix{glsnumberformat}%
3164     \else
3165     \def\@glo@suffix{#2}%
3166 \fi
3167 \else
3168     \def\@glo@prefix{}\def\@glo@suffix{#1#2}%
3169 \fi
3170 \fi}

```

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

```

3171 \newcommand*{\@gls@escbsdq}[1]{%
3172   \def\@gls@checkedmkidx{}%
3173   \let\gls@xdystring=#1\relax
3174   \@onelevel@sanitize\gls@xdystring
3175   \edef\do@gls@xdycheckbackslash{%
3176     \noexpand\@gls@xdycheckbackslash\gls@xdystring\noexpand\@nil
3177     \@backslashchar\@backslashchar\noexpand\null}%
3178   \do@gls@xdycheckbackslash
3179   \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdystring}%
3180   \def\@gls@checkedmkidx{}%
3181   \expandafter\@gls@xdycheckquote\gls@xdystring\@nil""\null
3182   \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdystring}%

```

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

```

3183   \@for\@gls@tmp:=\gls@protected@pagefmts\do
3184   {%
3185     \edef\@gls@sanitized@tmp{\expandafter\@gobble\string\\expandonce\@gls@tmp}%
3186     \@onelevel@sanitize\@gls@sanitized@tmp
3187     \edef\gls@dosubst{%
3188       \noexpand\DTLsubstituteall\noexpand\gls@xdystring
3189       {\@gls@sanitized@tmp}{\expandonce\@gls@tmp}%
3190     }%
3191     \gls@dosubst
3192   }%

```

Assign to required control sequence

```

3193   \let#1=\gls@xdystring
3194 }

```

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

`\gls@checkmkidxchars`

```

3195 \newcommand{\@gls@checkmkidxchars}[1]{%
3196   \ifglxsindy
3197     \@gls@escbsdq{#1}%
3198   \else
3199     \def\@gls@checkedmkidx{}%
3200     \expandafter\@gls@checkquote#1\@nil""\null

```

```

3201 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3202 \def\@gls@checkedmkidx{}%
3203 \expandafter\@gls@checkescquote#1\@nil\""\null
3204 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3205 \def\@gls@checkedmkidx{}%
3206 \expandafter\@gls@checkescactual#1\@nil\?\?\null
3207 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3208 \def\@gls@checkedmkidx{}%
3209 \expandafter\@gls@checkactual#1\@nil??\null
3210 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3211 \def\@gls@checkedmkidx{}%
3212 \expandafter\@gls@checkbar#1\@nil||\null
3213 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3214 \def\@gls@checkedmkidx{}%
3215 \expandafter\@gls@checkesbar#1\@nil\\|\null
3216 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3217 \def\@gls@checkedmkidx{}%
3218 \expandafter\@gls@checklevel#1\@nil!!\null
3219 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3220 \fi
3221 }

```

Update the control sequence and strip trailing \@nil:

\@gls@updatechecked

```

3222 \def\@gls@updatechecked#1\@nil#2{\def#2{#1}}

```

\@gls@tmpb Define temporary token

```

3223 \newtoks\@gls@tmpb

```

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

```

3224 \def\@gls@checkquote#1"#2"#3\null{%
3225 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3226 \toks@={#1}%
3227 \ifx\null#2\null
3228 \ifx\null#3\null
3229 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3230 \def\@gls@checkquote{\relax}%
3231 \else
3232 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3233 \@gls@quotechar\@gls@quotechar\@gls@quotechar\@gls@quotechar}%
3234 \def\@gls@checkquote{\@gls@checkquote#3\null}%
3235 \fi
3236 \else
3237 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3238 \@gls@quotechar\@gls@quotechar}%
3239 \ifx\null#3\null
3240 \def\@gls@checkquote{\@gls@checkquote#2""\null}%
3241 \else

```

```

3242     \def\@gls@checkquote{\@gls@checkquote#2"#3\null}%
3243   \fi
3244 \fi
3245 \@gls@checkquote
3246 }

```

\@gls@checkescquote Do the same for \":

```

3247 \def\@gls@checkescquote#1\"#2\"#3\null{%
3248   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3249   \toks@={#1}%
3250   \ifx\null#2\null
3251     \ifx\null#3\null
3252       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3253       \def\@gls@checkescquote{\relax}%
3254     \else
3255       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3256         \@gls@quotechar\string\" \@gls@quotechar
3257         \@gls@quotechar\string\" \@gls@quotechar}%
3258       \def\@gls@checkescquote{\@gls@checkescquote#3\null}%
3259     \fi
3260   \else
3261     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3262       \@gls@quotechar\string\" \@gls@quotechar}%
3263     \ifx\null#3\null
3264       \def\@gls@checkescquote{\@gls@checkescquote#2\"\" \null}%
3265     \else
3266       \def\@gls@checkescquote{\@gls@checkescquote#2\"#3\null}%
3267     \fi
3268   \fi
3269 \@gls@checkescquote
3270 }

```

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

```

3271 \def\@gls@checkescactual#1\?#2\?#3\null{%
3272   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3273   \toks@={#1}%
3274   \ifx\null#2\null
3275     \ifx\null#3\null
3276       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3277       \def\@gls@checkescactual{\relax}%
3278     \else
3279       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3280         \@gls@quotechar\string\" \@gls@actualchar
3281         \@gls@quotechar\string\" \@gls@actualchar}%
3282       \def\@gls@checkescactual{\@gls@checkescactual#3\null}%
3283     \fi
3284   \else
3285     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3286       \@gls@quotechar\string\" \@gls@actualchar}%

```



```

3287 \ifx\null#3\null
3288 \def\@gls@checkescactual{\@gls@checkescactual#2\?\?\null}%
3289 \else
3290 \def\@gls@checkescactual{\@gls@checkescactual#2\?#3\null}%
3291 \fi
3292 \fi
3293 \@gls@checkescactual
3294 }

```

\@gls@checkescbar Similarly for \|:

```

3295 \def\@gls@checkescbar#1\|#2\|#3\null{%
3296 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3297 \toks@={#1}%
3298 \ifx\null#2\null
3299 \ifx\null#3\null
3300 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3301 \def\@gls@checkescbar{\relax}%
3302 \else
3303 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3304 \@gls@quotechar\string\\"\@gls@encapchar
3305 \@gls@quotechar\string\\"\@gls@encapchar}%
3306 \def\@gls@checkescbar{\@gls@checkescbar#3\null}%
3307 \fi
3308 \else
3309 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3310 \@gls@quotechar\string\\"\@gls@encapchar}%
3311 \ifx\null#3\null
3312 \def\@gls@checkescbar{\@gls@checkescbar#2\|\|\null}%
3313 \else
3314 \def\@gls@checkescbar{\@gls@checkescbar#2\|#3\null}%
3315 \fi
3316 \fi
3317 \@gls@checkescbar
3318 }

```

\@gls@checkesclevel Similarly for \!:

```

3319 \def\@gls@checkesclevel#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@checkesclevel{\relax}%
3326 \else
3327 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3328 \@gls@quotechar\string\!\@gls@levelchar
3329 \@gls@quotechar\string\!\@gls@levelchar}%
3330 \def\@gls@checkesclevel{\@gls@checkesclevel#3\null}%
3331 \fi

```

```

3332 \else
3333 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3334 \@gls@quotechar\string\" \@gls@levelchar}%
3335 \ifx\@null#3\@null
3336 \def\@gls@checkesclevel{\@gls@checkesclevel#2\!\!\@null}%
3337 \else
3338 \def\@gls@checkesclevel{\@gls@checkesclevel#2\!#3\@null}%
3339 \fi
3340 \fi
3341 \@gls@checkesclevel
3342 }

```

\@gls@checkbar and for |:

```

3343 \def\@gls@checkbar#1|#2|#3\@null{%
3344 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3345 \toks@={#1}%
3346 \ifx\@null#2\@null
3347 \ifx\@null#3\@null
3348 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3349 \def\@gls@checkbar{\relax}%
3350 \else
3351 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3352 \@gls@quotechar\@gls@encapchar\@gls@quotechar\@gls@encapchar}%
3353 \def\@gls@checkbar{\@gls@checkbar#3\@null}%
3354 \fi
3355 \else
3356 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3357 \@gls@quotechar\@gls@encapchar}%
3358 \ifx\@null#3\@null
3359 \def\@gls@checkbar{\@gls@checkbar#2||\@null}%
3360 \else
3361 \def\@gls@checkbar{\@gls@checkbar#2|#3\@null}%
3362 \fi
3363 \fi
3364 \@gls@checkbar
3365 }

```

\@gls@checklevel and for !:

```

3366 \def\@gls@checklevel#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@checklevel{\relax}%
3373 \else
3374 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3375 \@gls@quotechar\@gls@levelchar\@gls@quotechar\@gls@levelchar}%
3376 \def\@gls@checklevel{\@gls@checklevel#3\@null}%

```

```

3377 \fi
3378 \else
3379 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3380 \@gls@quotechar\@gls@levelchar}%
3381 \ifx\null#3\null
3382 \def\@gls@checklevel{\@gls@checklevel#2!!\null}%
3383 \else
3384 \def\@gls@checklevel{\@gls@checklevel#2!#3\null}%
3385 \fi
3386 \fi
3387 \@gls@checklevel
3388 }

```

\@gls@checkactual and for ?:

```

3389 \def\@gls@checkactual#1?#2?#3\null{%
3390 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3391 \toks@={#1}%
3392 \ifx\null#2\null
3393 \ifx\null#3\null
3394 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3395 \def\@gls@checkactual{\relax}%
3396 \else
3397 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3398 \@gls@quotechar\@gls@actualchar\@gls@quotechar\@gls@actualchar}%
3399 \def\@gls@checkactual{\@gls@checkactual#3\null}%
3400 \fi
3401 \else
3402 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3403 \@gls@quotechar\@gls@actualchar}%
3404 \ifx\null#3\null
3405 \def\@gls@checkactual{\@gls@checkactual#2??\null}%
3406 \else
3407 \def\@gls@checkactual{\@gls@checkactual#2?#3\null}%
3408 \fi
3409 \fi
3410 \@gls@checkactual
3411 }

```

\@gls@xdycheckquote As before but for use with xindy

```

3412 \def\@gls@xdycheckquote#1"#2"#3\null{%
3413 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3414 \toks@={#1}%
3415 \ifx\null#2\null
3416 \ifx\null#3\null
3417 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3418 \def\@gls@xdycheckquote{\relax}%
3419 \else
3420 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3421 \string"\string"}%

```

```

3422     \def\@gls@xdycheckquote{\@gls@xdycheckquote#3\null}%
3423     \fi
3424   \else
3425     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3426       \string\}%
3427     \ifx\null#3\null
3428       \def\@gls@xdycheckquote{\@gls@xdycheckquote#2""\null}%
3429     \else
3430       \def\@gls@xdycheckquote{\@gls@xdycheckquote#2"#3\null}%
3431     \fi
3432   \fi
3433   \@gls@xdycheckquote
3434 }

```

s@xdycheckbackslash Need to escape all backslashes for xindy. Define command that will define

```

\@gls@xdycheckbackslash
3435 \edef\def\@gls@xdycheckbackslash{%
3436   \noexpand\def\noexpand\@gls@xdycheckbackslash##1\@backslashchar
3437     ##2\@backslashchar##3\noexpand\null{%
3438     \noexpand\@gls@tmpb=\noexpand\expandafter
3439       {\noexpand\@gls@checkedmkidx}%
3440     \noexpand\toks@={##1}%
3441     \noexpand\ifx\noexpand\null##2\noexpand\null
3442       \noexpand\ifx\noexpand\null##3\noexpand\null
3443       \noexpand\edef\noexpand\@gls@checkedmkidx{%
3444         \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
3445       \noexpand\def\noexpand\@gls@xdycheckbackslash{\relax}%
3446     \noexpand\else
3447       \noexpand\edef\noexpand\@gls@checkedmkidx{%
3448         \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
3449         \@backslashchar\@backslashchar\@backslashchar\@backslashchar}%
3450     \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3451       \noexpand\@gls@xdycheckbackslash##3\noexpand\null}%
3452     \noexpand\fi
3453   \noexpand\else
3454     \noexpand\edef\noexpand\@gls@checkedmkidx{%
3455       \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
3456       \@backslashchar\@backslashchar}%
3457   \noexpand\ifx\noexpand\null##3\noexpand\null
3458     \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3459       \noexpand\@gls@xdycheckbackslash##2\@backslashchar
3460       \@backslashchar\noexpand\null}%
3461   \noexpand\else
3462     \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3463       \noexpand\@gls@xdycheckbackslash##2\@backslashchar
3464       ##3\noexpand\null}%
3465   \noexpand\fi
3466   \noexpand\fi
3467   \noexpand\@gls@xdycheckbackslash

```

```

3468 }%
3469 }

Now go ahead and define \@gls@xdycheckbackslash
3470 \def@gls@xdycheckbackslash

```

`\glsdohypertarget`

```

3471 \newlength@gls@tmplen
3472 \newcommand*{\glsdohypertarget}[2]{%
3473   \settoheight@gls@tmplen{#2}%
3474   \raisebox@gls@tmplen{\hypertarget{#1}{}}#2%
3475 }

```

`\glsdohyperlink`

```

3476 \newcommand*{\glsdohyperlink}[2]{\hyperlink{#1}{#2}}

```

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

```

3477 \ifcsundef{hyperlink}%
3478 {%
3479   \let@glslink@secondoftwo
3480 }%
3481 {%
3482   \let@glslink@glsdohyperlink
3483 }

```

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

```

3484 \ifcsundef{hypertarget}%
3485 {%
3486   \let@glstarget@secondoftwo
3487 }%
3488 {%
3489   \let@glstarget@glsdohypertarget
3490 }

```

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

`\glsdisablehyper`

```

3491 \newcommand{\glsdisablehyper}{%
3492   \KV@glslink@hyperfalse
3493   \let@glslink@secondoftwo
3494   \let@glstarget@secondoftwo
3495 }

```

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

`\glsenablehyper`

```
3496 \newcommand{\glsenablehyper}{%
3497   \KV@glslink@hypertrue
3498   \let\@glslink\glsdohyperlink
3499   \let\@glstarget\glsdohypertarget
3500 }
```

Provide some convenience commands if not already defined:

```
3501 \providecommand{\@firstofthree}[3]{#1}
3502 \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`

```
3503 \newrobustcmd*{\gls}{\@gls@hyp@opt\gls}
```

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

`\@gls`

```
3504 \newcommand*{\@gls}[2] [] {%
3505   \new@ifnextchar[{\@gls@{#1}{#2}}{\@gls@{#1}{#2} []}%
3506 }
```

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

```
3507 \def\@gls@#1#2[#3]{%
3508   \glsdoifexists{#2}%
3509   {%
3510     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3511     \let\glsifplural\@secondoftwo
3512     \let\glscapscase\@firstofthree
3513     \let\glscustomtext\@empty
3514     \def\glsinsert{#3}%
```

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

```
3515 \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.

```
3516 \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3517 \ifKV@glslink@local
```

```
3518 \glsllocalunset{#2}%
```

```
3519 \else
```

```
3520 \glsunset{#2}%
```

```
3521 \fi
```

```
3522 }%
```

```
3523 \glspostlinkhook
```

```
3524 }
```

\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

```
3525 \newrobustcmd*{\Gls}{\@gls@hyp@opt\@Gls}
```

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

```
3526 \newcommand*{\@Gls}[2][{}]{%
```

```
3527 \new@ifnextchar[{\@Gls@{#1}{#2}}{\@Gls@{#1}{#2}[]}%
```

```
3528 }
```

\@Gls@ Read in the final optional argument:

```
3529 \def\@Gls@#1#2[#3]{%
```

```
3530 \glsoifexists{#2}%
```

```
3531 {%
```

```
3532 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
```

```
3533 \let\glsifplural\@secondoftwo
```

```
3534 \let\glscapscase\@secondofthree
```

```
3535 \let\glscustomtext\@empty
```

```
3536 \def\glsinsert{#3}%
```

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

```
3537 \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.

```
3538 \gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3539 \ifKV@glslink@local
3540 \glslocalunset{#2}%
3541 \else
3542 \glsunset{#2}%
3543 \fi
3544 }%
```

```
3545 \glspostlinkhook
3546 }
```

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

`\GLS`

```
3547 \newrobustcmd*{\GLS}{\@gls@hyp@opt\@GLS}
```

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

```
3548 \newcommand*{\@GLS}[2] [] {%
3549 \new@ifnextchar[{\@GLS@{#1}{#2}}{\@GLS@{#1}{#2} []}%
3550 }
```

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

```
3551 \def\@GLS@#1#2[#3]{%
3552 \glsdoifexists{#2}%
3553 {%
3554 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3555 \let\glsifplural\@secondoftwo
3556 \let\glscapscase\@thirdofthree
3557 \let\glscustomtext\@empty
3558 \def\glsinsert{#3}%
```

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

```
3559 \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.

```
3560 \gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3561 \ifKV@glslink@local
3562 \glslocalunset{#2}%
```



```

3563 \else
3564 \glsunset{#2}%
3565 \fi
3566 }%

3567 \glspostlinkhook
3568 }

```

`\glspl` behaves in the same way as `\gls` except it uses the plural form.

`\glspl`

```

3569 \newrobustcmd*{\glspl}{\@gls@hyp@opt\@glspl}

```

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

```

3570 \newcommand*{\@glspl}[2] [] {%
3571 \new@ifnextchar[{\@glspl@{#1}{#2}}{\@glspl@{#1}{#2} []}%
3572 }

```

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

```

3573 \def\@glspl@#1#2[#3] {%
3574 \glsdoifexists{#2}%
3575 {%
3576 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3577 \let\glsifplural\@firstoftwo
3578 \let\glsupcase\@firstofthree
3579 \let\glsustomtext\@empty
3580 \def\glsinsert{#3}%

```

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

```

3581 \def\@glo@text{\csname gls@\glstyp @entryfmt\endcsname}%

```

Call `\@gls@link`. If footnote package option has been used and the glossary type is `\acronymtype`, suppress hyperlink for first use. Likewise if the `hyperfirst=false` package option is used.

```

3582 \@gls@link[#1]{#2}{\@glo@text}%

```

Indicate that this entry has now been used

```

3583 \ifKV@glslink@local
3584 \glslocalunset{#2}%
3585 \else
3586 \glsunset{#2}%
3587 \fi
3588 }%

3589 \glspostlinkhook
3590 }

```

\Glspl

3594 }

3612 }

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

```
3614 \newcommand*{\@GLSp1}[2][\{%
3615   \new@ifnextchar[{\@GLSp1@{#1}{#2}}{\@GLSp1@{#1}{#2}[]}%
3616 }
```

\@GLSp1 Read in the final optional argument:

```
3617 \def\@GLSp1@#1#2[#3]{%
3618   \glsdoifexists{#2}%
3619   {%
3620     \let\do@glsl@link@checkfirsthyper\@glsl@link@checkfirsthyper

3621     \let\glsifplural\@firstoftwo
3622     \let\glscapscase\@thirdofthree
3623     \let\glscustomtext\@empty
3624     \def\glsinsert{#3}%
```

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

```
3625   \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.

```
3626   \@glsl@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3627   \ifKV@glsl@link@local
3628     \glsllocalunset{#2}%
3629   \else
3630     \glslunset{#2}%
3631   \fi
3632 }
```

```
3633 \glspostlinkhook
3634 }
```

\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:

```
3635 \newrobustcmd*{\glsdisp}{\@glsl@hyp@opt\@glsdisp}
```

Defined the un-starred form.

\@glsdisp

```
3636 \newcommand*{\@glsdisp}[3][\{%
3637   \glsdoifexists{#2}{%

3638   \let\do@glsl@link@checkfirsthyper\@glsl@link@checkfirsthyper
```

```

3639 \let\glsifplural\@secondoftwo
3640 \let\glscapscase\@firstofthree
3641 \def\glscustomtext{#3}%
3642 \def\glsinsert{}%

```

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

```

3643 \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.

```

3644 \@gls@link[#1]{#2}{\@glo@text}%

```

Indicate that this entry has now been used

```

3645 \ifKV@glslink@local
3646 \glslocalunset{#2}%
3647 \else
3648 \glsunset{#2}%
3649 \fi
3650 }%

```

```

3651 \glspostlinkhook
3652 }

```

k@nocheckfirsthyper 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.

```

3653 \newcommand*{\@gls@link@nocheckfirsthyper}{}

```

\@gls@field@link

```

3654 \newcommand{\@gls@field@link}[3]{%
3655 \glsdoifexists{#2}%
3656 {%
3657 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
3658 \@gls@link[#1]{#2}{#3}%
3659 }%
3660 \glspostlinkhook
3661 }

```

\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

```

3662 \newrobustcmd*{\glstext}{\@gls@hyp@opt\@glstext}

```

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

```

3663 \newcommand*{\@glstext}[2][ ]{%
3664 \new@ifnextchar[{\@glstext@{#1}{#2}}{\@glstext@{#1}{#2}[ ]}]

```

Read in the final optional argument:

```
3665 \def\@glstext@#1#2[#3]{%
3666   \@gls@field@link{#1}{#2}{\glsentrytext{#2}#3}%
3667 }
```

\GLStext behaves like \glstext except the text is converted to uppercase.

\GLStext

```
3668 \newrobustcmd*{\GLStext}{\@gls@hyp@opt\@GLStext}
```

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

```
3669 \newcommand*{\@GLStext}[2][]{%
3670   \new@ifnextchar[{\@GLStext@{#1}{#2}}{\@GLStext@{#1}{#2}[]}]}
```

Read in the final optional argument:

```
3671 \def\@GLStext@#1#2[#3]{%
3672   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrytext{#2}#3}}%
3673 }
```

\Glstext behaves like \glstext except that the first letter of the text is converted to uppercase.

\Glstext

```
3674 \newrobustcmd*{\Glstext}{\@gls@hyp@opt\@Glstext}
```

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

```
3675 \newcommand*{\@Glstext}[2][]{%
3676   \new@ifnextchar[{\@Glstext@{#1}{#2}}{\@Glstext@{#1}{#2}[]}]}
```

Read in the final optional argument:

```
3677 \def\@Glstext@#1#2[#3]{%
3678   \@gls@field@link{#1}{#2}{\Glsentrytext{#2}#3}%
3679 }
```

\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

```
3680 \newrobustcmd*{\glsfirst}{\@gls@hyp@opt\@glsfirst}
```

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

```
3681 \newcommand*{\@glsfirst}[2][]{%
3682   \new@ifnextchar[{\@glsfirst@{#1}{#2}}{\@glsfirst@{#1}{#2}[]}]}
```

Read in the final optional argument:

```
3683 \def\@glsfirst@#1#2[#3]{%
3684   \@gls@field@link{#1}{#2}{\glsentryfirst{#2}#3}%
3685 }
```

`\Glsfirst` behaves like `\glsfirst` except it displays the first letter in uppercase.

`\Glsfirst`

```
3686 \newrobustcmd*{\Glsfirst}{\@gls@hyp@opt\@Glsfirst}
```

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

```
3687 \newcommand*{\@Glsfirst}[2] []{%
```

```
3688   \new@ifnextchar[{\@Glsfirst@{#1}{#2}}{\@Glsfirst@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3689 \def\@Glsfirst@#1#2[#3]{%
```

```
3690   \@gls@field@link{#1}{#2}{\Glsentryfirst{#2}#3}%
```

```
3691 }
```

`\GLSfirst` behaves like `\Glsfirst` except it displays the text in uppercase.

`\GLSfirst`

```
3692 \newrobustcmd*{\GLSfirst}{\@gls@hyp@opt\@GLSfirst}
```

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

```
3693 \newcommand*{\@GLSfirst}[2] []{%
```

```
3694   \new@ifnextchar[{\@GLSfirst@{#1}{#2}}{\@GLSfirst@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3695 \def\@GLSfirst@#1#2[#3]{%
```

```
3696   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentryfirst{#2}#3}}%
```

```
3697 }
```

`\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`

```
3698 \newrobustcmd*{\glsplural}{\@gls@hyp@opt\@glsplural}
```

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

```
3699 \newcommand*{\@glsplural}[2] []{%
```

```
3700   \new@ifnextchar[{\@glsplural@{#1}{#2}}{\@glsplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3701 \def\@glsplural@#1#2[#3]{%
```

```
3702   \@gls@field@link{#1}{#2}{\Glsentryplural{#2}#3}%
```

```
3703 }
```

`\Glsplural` behaves like `\glsplural` except that the first letter is converted to uppercase.

`\Glsplural`

```
3704 \newrobustcmd*{\Glsplural}{\@gls@hyp@opt\@Glsplural}
```

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

```
3705 \newcommand*{\@Glsplural}[2] [] {%
3706   \new@ifnextchar[{\@Glsplural@{#1}{#2}}{\@Glsplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3707 \def\@Glsplural@#1#2[#3] {%
3708   \@gls@field@link{#1}{#2}{\Glsentryplural{#2}#3}%
3709 }
```

\Glsplural behaves like \glsplural except that the text is converted to uppercase.

\Glsplural

```
3710 \newrobustcmd*{\Glsplural}{\@gls@hyp@opt\@Glsplural}
```

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

```
3711 \newcommand*{\@Glsplural}[2] [] {%
3712   \new@ifnextchar[{\@Glsplural@{#1}{#2}}{\@Glsplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3713 \def\@Glsplural@#1#2[#3] {%
3714   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryplural{#2}#3}}%
3715 }
```

\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

```
3716 \newrobustcmd*{\glsfirstplural}{\@gls@hyp@opt\@glsfirstplural}
```

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

```
3717 \newcommand*{\@glsfirstplural}[2] [] {%
3718   \new@ifnextchar[{\@glsfirstplural@{#1}{#2}}{\@glsfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3719 \def\@glsfirstplural@#1#2[#3] {%
3720   \@gls@field@link{#1}{#2}{\glsentryfirstplural{#2}#3}%
3721 }
```

\Glsfirstplural behaves like \glsfirstplural except that the first letter is converted to uppercase.

\Glsfirstplural

```
3722 \newrobustcmd*{\Glsfirstplural}{\@gls@hyp@opt\@Glsfirstplural}
```

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

```
3723 \newcommand*{\@Glsfirstplural}[2] [] {%
3724   \new@ifnextchar[{\@Glsfirstplural@{#1}{#2}}{\@Glsfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3725 \def\@Glsfirstplural@#1#2[#3]{%
3726   \@gls@field@link{#1}{#2}{\Glsentryfirstplural{#2}#3}%
3727 }
```

\Glsfirstplural behaves like \glsfirstplural except that the link text is converted to uppercase.

\Glsfirstplural

```
3728 \newrobustcmd*{\Glsfirstplural}{\@gls@hyp@opt\@Glsfirstplural}
```

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

```
3729 \newcommand*{\@Glsfirstplural}[2][ ]{%
3730   \new@ifnextchar[{\@Glsfirstplural@{#1}{#2}}{\@Glsfirstplural@{#1}{#2}[ ]}]
```

Read in the final optional argument:

```
3731 \def\@Glsfirstplural@#1#2[#3]{%
3732   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentryfirstplural{#2}#3}}%
3733 }
```

\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

```
3734 \newrobustcmd*{\glsname}{\@gls@hyp@opt\@glsname}
```

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

```
3735 \newcommand*{\@glsname}[2][ ]{%
3736   \new@ifnextchar[{\@glsname@{#1}{#2}}{\@glsname@{#1}{#2}[ ]}]
```

Read in the final optional argument:

```
3737 \def\@glsname@#1#2[#3]{%
3738   \@gls@field@link{#1}{#2}{\Glsentryname{#2}#3}%
3739 }
```

\Glsname behaves like \glsname except that the first letter is converted to uppercase.

\Glsname

```
3740 \newrobustcmd*{\Glsname}{\@gls@hyp@opt\@Glsname}
```

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

```
3741 \newcommand*{\@Glsname}[2][ ]{%
3742   \new@ifnextchar[{\@Glsname@{#1}{#2}}{\@Glsname@{#1}{#2}[ ]}]
```

Read in the final optional argument:

```
3743 \def\@Glsname@#1#2[#3]{%
3744   \@gls@field@link{#1}{#2}{\Glsentryname{#2}#3}%
3745 }
```


`\GLSname` behaves like `\glsname` except that the link text is converted to uppercase.

`\GLSname`

```
3746 \newrobustcmd*{\GLSname}{\@gls@hyp@opt\@GLSname}
```

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

```
3747 \newcommand*{\@GLSname}[2] [] {%
```

```
3748   \new@ifnextchar[{\@GLSname@{#1}{#2}}{\@GLSname@{#1}{#2} []}]
```

Read in the final optional argument:

```
3749 \def\@GLSname@#1#2[#3] {%
```

```
3750   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryname{#2}#3}}%
```

```
3751 }
```

`\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`

```
3752 \newrobustcmd*{\glsdesc}{\@gls@hyp@opt\@glsdesc}
```

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

```
3753 \newcommand*{\@glsdesc}[2] [] {%
```

```
3754   \new@ifnextchar[{\@glsdesc@{#1}{#2}}{\@glsdesc@{#1}{#2} []}]
```

Read in the final optional argument:

```
3755 \def\@glsdesc@#1#2[#3] {%
```

```
3756   \@gls@field@link{#1}{#2}{\glsentrydesc{#2}#3}%
```

```
3757 }
```

`\Glsdesc` behaves like `\glsdesc` except that the first letter is converted to uppercase.

`\Glsdesc`

```
3758 \newrobustcmd*{\Glsdesc}{\@gls@hyp@opt\@Glsdesc}
```

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

```
3759 \newcommand*{\@Glsdesc}[2] [] {%
```

```
3760   \new@ifnextchar[{\@Glsdesc@{#1}{#2}}{\@Glsdesc@{#1}{#2} []}]
```

Read in the final optional argument:

```
3761 \def\@Glsdesc@#1#2[#3] {%
```

```
3762   \@gls@field@link{#1}{#2}{\Glsentrydesc{#2}#3}%
```

```
3763 }
```

`\GLSdesc` behaves like `\glsdesc` except that the link text is converted to uppercase.

`\GLSdesc`

```
3764 \newrobustcmd*{\GLSdesc}{\@gls@hyp@opt\@GLSdesc}
```

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

```
3765 \newcommand*{\@GLSdesc}[2] [] {%
3766   \new@ifnextchar[{\@GLSdesc@{#1}{#2}}{\@GLSdesc@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3767 \def\@GLSdesc@#1#2[#3] {%
3768   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrydesc{#2}#3}}%
3769 }
```

`\glsdescplural` behaves like `\gls` except it always uses the value given by the descriptionplural key and it doesn't mark the entry as used.

`\glsdescplural`

```
3770 \newrobustcmd*{\glsdescplural}{\@gls@hyp@opt\@glsdescplural}
```

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

```
3771 \newcommand*{\@glsdescplural}[2] [] {%
3772   \new@ifnextchar[{\@glsdescplural@{#1}{#2}}{\@glsdescplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3773 \def\@glsdescplural@#1#2[#3] {%
3774   \@gls@field@link{#1}{#2}{\glsentrydescplural{#2}#3}%
3775 }
```

`\Glsdescplural` behaves like `\glsdescplural` except that the first letter is converted to uppercase.

`\Glsdescplural`

```
3776 \newrobustcmd*{\Glsdescplural}{\@gls@hyp@opt\@Glsdescplural}
```

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

```
3777 \newcommand*{\@Glsdescplural}[2] [] {%
3778   \new@ifnextchar[{\@Glsdescplural@{#1}{#2}}{\@Glsdescplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3779 \def\@Glsdescplural@#1#2[#3] {%
3780   \@gls@field@link{#1}{#2}{\Glsentrydescplural{#2}#3}%
3781 }
```

`\GLSdescplural` behaves like `\glsdescplural` except that the link text is converted to uppercase.

`\GLSdescplural`

```
3782 \newrobustcmd*{\GLSdescplural}{\@gls@hyp@opt\@GLSdescplural}
```

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

```
3783 \newcommand*{\@GLSdescplural}[2] [] {%
3784   \new@ifnextchar[{\@GLSdescplural@{#1}{#2}}{\@GLSdescplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3785 \def\@GLSdescplural@#1#2[#3]{%
3786   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrydescplural{#2}#3}}%
3787 }
```

`\glsymbol` behaves like `\gls` except it always uses the value given by the symbol key and it doesn't mark the entry as used.

`\glsymbol`

```
3788 \newrobustcmd*{\glsymbol}{\@gls@hyp@opt\@glsymbol}
```

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

```
3789 \newcommand*{\@glsymbol}[2] [] {%
3790   \new@ifnextchar[{\@glsymbol@{#1}{#2}}{\@glsymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3791 \def\@glsymbol@#1#2[#3]{%
3792   \@gls@field@link{#1}{#2}{\glsentrysymbol{#2}#3}%
3793 }
```

`\Glsymbol` behaves like `\glsymbol` except that the first letter is converted to uppercase.

`\Glsymbol`

```
3794 \newrobustcmd*{\Glsymbol}{\@gls@hyp@opt\@Glsymbol}
```

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

```
3795 \newcommand*{\@Glsymbol}[2] [] {%
3796   \new@ifnextchar[{\@Glsymbol@{#1}{#2}}{\@Glsymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3797 \def\@Glsymbol@#1#2[#3]{%
3798   \@gls@field@link{#1}{#2}{\Glsentrysymbol{#2}#3}%
3799 }
```

`\GLSsymbol` behaves like `\glsymbol` except that the link text is converted to uppercase.

`\GLSsymbol`

```
3800 \newrobustcmd*{\GLSsymbol}{\@gls@hyp@opt\@GLSsymbol}
```

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

```
3801 \newcommand*{\@GLSsymbol}[2] [] {%
3802   \new@ifnextchar[{\@GLSsymbol@{#1}{#2}}{\@GLSsymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3803 \def\@GLSsymbol@#1#2[#3]{%
3804   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrysymbol{#2}#3}}%
3805 }
```

`\glssymbolplural` behaves like `\gls` except it always uses the value given by the `symbolplural` key and it doesn't mark the entry as used.

`\glssymbolplural`

```
3806 \newrobustcmd*{\glssymbolplural}{\@gls@hyp@opt\glssymbolplural}
```

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

```
3807 \newcommand*{\@glssymbolplural}[2] [] {%
```

```
3808   \new@ifnextchar[{\@glssymbolplural@{#1}{#2}}{\@glssymbolplural@{#1}{#2} []}]
```

Read in the final optional argument:

```
3809 \def\@glssymbolplural@#1#2[#3] {%
```

```
3810   \@gls@field@link{#1}{#2}{\glstentrysymbolplural{#2}#3}%
```

```
3811 }
```

`\Glsymbolplural` behaves like `\glssymbolplural` except that the first letter is converted to uppercase.

`\Glsymbolplural`

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

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

```
3813 \newcommand*{\@Glsymbolplural}[2] [] {%
```

```
3814   \new@ifnextchar[{\@Glsymbolplural@{#1}{#2}}{\@Glsymbolplural@{#1}{#2} []}]
```

Read in the final optional argument:

```
3815 \def\@Glsymbolplural@#1#2[#3] {%
```

```
3816   \@gls@field@link{#1}{#2}{\glstentrysymbolplural{#2}#3}%
```

```
3817 }
```

`\GLSsymbolplural` behaves like `\glssymbolplural` except that the link text is converted to uppercase.

`\GLSsymbolplural`

```
3818 \newrobustcmd*{\GLSsymbolplural}{\@gls@hyp@opt\GLSsymbolplural}
```

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

```
3819 \newcommand*{\@GLSsymbolplural}[2] [] {%
```

```
3820   \new@ifnextchar[{\@GLSsymbolplural@{#1}{#2}}{\@GLSsymbolplural@{#1}{#2} []}]
```

Read in the final optional argument:

```
3821 \def\@GLSsymbolplural@#1#2[#3] {%
```

```
3822   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glstentrysymbolplural{#2}#3}}%
```

```
3823 }
```

`\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`

```
3824 \newrobustcmd*{\glsuseri}{\@gls@hyp@opt\glsuseri}
```

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

```
3825 \newcommand*{\@glsuseri}[2] [] {%
3826   \new@ifnextchar[{\@glsuseri@{#1}{#2}}{\@glsuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3827 \def\@glsuseri@#1#2[#3] {%
3828   \@gls@field@link{#1}{#2}{\glsentryuseri{#2}#3}%
3829 }
```

\Glsuseri behaves like \glsuseri except that the first letter is converted to uppercase.

\Glsuseri

```
3830 \newrobustcmd*{\Glsuseri}{\@gls@hyp@opt\@Glsuseri}
```

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

```
3831 \newcommand*{\@Glsuseri}[2] [] {%
3832   \new@ifnextchar[{\@Glsuseri@{#1}{#2}}{\@Glsuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3833 \def\@Glsuseri@#1#2[#3] {%
3834   \@gls@field@link{#1}{#2}{\Glsentryuseri{#2}#3}%
3835 }
```

\GLSuseri behaves like \glsuseri except that the link text is converted to uppercase.

\GLSuseri

```
3836 \newrobustcmd*{\GLSuseri}{\@gls@hyp@opt\@GLSuseri}
```

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

```
3837 \newcommand*{\@GLSuseri}[2] [] {%
3838   \new@ifnextchar[{\@GLSuseri@{#1}{#2}}{\@GLSuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3839 \def\@GLSuseri@#1#2[#3] {%
3840   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseri{#2}#3}}%
3841 }
```

\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

```
3842 \newrobustcmd*{\glsuserii}{\@gls@hyp@opt\@glsuserii}
```

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

```
3843 \newcommand*{\@glsuserii}[2] [] {%
3844   \new@ifnextchar[{\@glsuserii@{#1}{#2}}{\@glsuserii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3845 \def\@glsuserii@#1#2[#3]{%
3846   \@gls@field@link{#1}{#2}{\glstryuserii{#2}#3}%
3847 }
```

\Glsuserii behaves like \glsuserii except that the first letter is converted to uppercase.

\Glsuserii

```
3848 \newrobustcmd*{\Glsuserii}{\@gls@hyp@opt\@Glsuserii}
```

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

```
3849 \newcommand*{\@Glsuserii}[2][]{%
3850   \new@ifnextchar[{\@Glsuserii@{#1}{#2}}{\@Glsuserii@{#1}{#2}[]}]
```

Read in the final optional argument:

```
3851 \def\@Glsuserii@#1#2[#3]{%
3852   \@gls@field@link{#1}{#2}{\glstryuserii{#2}#3}%
3853 }
```

\GLSuserii behaves like \glsuserii except that the link text is converted to uppercase.

\GLSuserii

```
3854 \newrobustcmd*{\GLSuserii}{\@gls@hyp@opt\@GLSuserii}
```

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

```
3855 \newcommand*{\@GLSuserii}[2][]{%
3856   \new@ifnextchar[{\@GLSuserii@{#1}{#2}}{\@GLSuserii@{#1}{#2}[]}]
```

Read in the final optional argument:

```
3857 \def\@GLSuserii@#1#2[#3]{%
3858   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glstryuserii{#2}#3}}%
3859 }
```

\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

```
3860 \newrobustcmd*{\glsuseriii}{\@gls@hyp@opt\@glsuseriii}
```

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

```
3861 \newcommand*{\@glsuseriii}[2][]{%
3862   \new@ifnextchar[{\@glsuseriii@{#1}{#2}}{\@glsuseriii@{#1}{#2}[]}]
```

Read in the final optional argument:

```
3863 \def\@glsuseriii@#1#2[#3]{%
3864   \@gls@field@link{#1}{#2}{\glstryuseriii{#2}#3}%
3865 }
```

`\Glsuseriii` behaves like `\glsuseriii` except that the first letter is converted to uppercase.

`\Glsuseriii`

```
3866 \newrobustcmd*{\Glsuseriii}{\@gls@hyp@opt\@Glsuseriii}
```

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

```
3867 \newcommand*{\@Glsuseriii}[2] [] {%
```

```
3868   \new@ifnextchar[{\@Glsuseriii@{#1}{#2}}{\@Glsuseriii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3869 \def\@Glsuseriii@#1#2[#3] {%
```

```
3870   \@gls@field@link{#1}{#2}{\Glsentryuseriii{#2}#3}%
```

```
3871 }
```

`\GLSuseriii` behaves like `\glsuseriii` except that the link text is converted to uppercase.

`\GLSuseriii`

```
3872 \newrobustcmd*{\GLSuseriii}{\@gls@hyp@opt\@GLSuseriii}
```

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

```
3873 \newcommand*{\@GLSuseriii}[2] [] {%
```

```
3874   \new@ifnextchar[{\@GLSuseriii@{#1}{#2}}{\@GLSuseriii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3875 \def\@GLSuseriii@#1#2[#3] {%
```

```
3876   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentryuseriii{#2}#3}}%
```

```
3877 }
```

`\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`

```
3878 \newrobustcmd*{\glsuseriv}{\@gls@hyp@opt\@glsuseriv}
```

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

```
3879 \newcommand*{\@glsuseriv}[2] [] {%
```

```
3880   \new@ifnextchar[{\@glsuseriv@{#1}{#2}}{\@glsuseriv@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3881 \def\@glsuseriv@#1#2[#3] {%
```

```
3882   \@gls@field@link{#1}{#2}{\Glsentryuseriv{#2}#3}%
```

```
3883 }
```

`\Glsuseriv` behaves like `\glsuseriv` except that the first letter is converted to uppercase.

`\Glsuseriv`

```
3884 \newrobustcmd*{\Glsuseriv}{\@gls@hyp@opt\@Glsuseriv}
```

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

```
3885 \newcommand*{\@Glsuseriv}[2] [] {%
3886   \new@ifnextchar[{\@Glsuseriv@{#1}{#2}}{\@Glsuseriv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3887 \def\@Glsuseriv@#1#2[#3] {%
3888   \@gls@field@link{#1}{#2}{\Glsentryuseriv{#2}#3}%
3889 }
```

\Glsuseriv behaves like \glsuseriv except that the link text is converted to uppercase.

\Glsuseriv

```
3890 \newrobustcmd*{\Glsuseriv}{\@gls@hyp@opt\@Glsuseriv}
```

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

```
3891 \newcommand*{\@Glsuseriv}[2] [] {%
3892   \new@ifnextchar[{\@Glsuseriv@{#1}{#2}}{\@Glsuseriv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3893 \def\@Glsuseriv@#1#2[#3] {%
3894   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentryuseriv{#2}#3}}%
3895 }
```

\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

```
3896 \newrobustcmd*{\glsuserv}{\@gls@hyp@opt\@glsuserv}
```

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

```
3897 \newcommand*{\@glsuserv}[2] [] {%
3898   \new@ifnextchar[{\@glsuserv@{#1}{#2}}{\@glsuserv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3899 \def\@glsuserv@#1#2[#3] {%
3900   \@gls@field@link{#1}{#2}{\Glsentryuserv{#2}#3}%
3901 }
```

\Glsuserv behaves like \glsuserv except that the first letter is converted to uppercase.

\Glsuserv

```
3902 \newrobustcmd*{\Glsuserv}{\@gls@hyp@opt\@Glsuserv}
```

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

```
3903 \newcommand*{\@Glsuserv}[2] [] {%
3904   \new@ifnextchar[{\@Glsuserv@{#1}{#2}}{\@Glsuserv@{#1}{#2} [] }}
```


Read in the final optional argument:

```
3905 \def\@Glsuserv@#1#2[#3]{%  
3906   \@gls@field@link{#1}{#2}{\Glsentryuserv{#2}#3}%  
3907 }
```

\Glsuserv behaves like \glsuserv except that the link text is converted to uppercase.

\Glsuserv

```
3908 \newrobustcmd*{\Glsuserv}{\@gls@hyp@opt\@Glsuserv}
```

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

```
3909 \newcommand*{\@Glsuserv}[2][ ]{%  
3910   \new@ifnextchar[{\@Glsuserv@{#1}{#2}}{\@Glsuserv@{#1}{#2}[ ]}]}
```

Read in the final optional argument:

```
3911 \def\@Glsuserv@#1#2[#3]{%  
3912   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentryuserv{#2}#3}}%  
3913 }
```

\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

```
3914 \newrobustcmd*{\glsuservi}{\@gls@hyp@opt\@glsuservi}
```

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

```
3915 \newcommand*{\@glsuservi}[2][ ]{%  
3916   \new@ifnextchar[{\@glsuservi@{#1}{#2}}{\@glsuservi@{#1}{#2}[ ]}]}
```

Read in the final optional argument:

```
3917 \def\@glsuservi@#1#2[#3]{%  
3918   \@gls@field@link{#1}{#2}{\Glsentryuservi{#2}#3}%  
3919 }
```

\Glsuservi behaves like \glsuservi except that the first letter is converted to uppercase.

\Glsuservi

```
3920 \newrobustcmd*{\Glsuservi}{\@gls@hyp@opt\@Glsuservi}
```

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

```
3921 \newcommand*{\@Glsuservi}[2][ ]{%  
3922   \new@ifnextchar[{\@Glsuservi@{#1}{#2}}{\@Glsuservi@{#1}{#2}[ ]}]}
```

Read in the final optional argument:

```
3923 \def\@Glsuservi@#1#2[#3]{%  
3924   \@gls@field@link{#1}{#2}{\Glsentryuservi{#2}#3}%  
3925 }
```

`\GLSuservi` behaves like `\glsuservi` except that the link text is converted to uppercase.

`\GLSuservi`

```
3926 \newrobustcmd*{\GLSuservi}{\@gls@hyp@opt\@GLSuservi}
```

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

```
3927 \newcommand*{\@GLSuservi}[2] [] {%
```

```
3928   \new@ifnextchar[{\@GLSuservi@{#1}{#2}}{\@GLSuservi@{#1}{#2} []}]%
```

Read in the final optional argument:

```
3929 \def\@GLSuservi@#1#2[#3] {%
```

```
3930   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuservi{#2}#3}}%
```

```
3931 }
```

Now deal with acronym related keys. First the short form:

`\acrshort`

```
3932 \newrobustcmd*{\acrshort}{\@gls@hyp@opt\@ns@acrshort}
```

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

```
3933 \newcommand*{\@ns@acrshort}[2] [] {%
```

```
3934   \new@ifnextchar[{\@acrshort{#1}{#2}}{\@acrshort{#1}{#2} []}]%
```

```
3935 }
```

Read in the final optional argument:

```
3936 \def\@acrshort#1#2[#3] {%
```

```
3937   \glsdoifexists{#2}%
```

```
3938   {%
```

```
3939     \let\do@gls@link@checkfirsthyper\relax
```

```
3940     \let\glsifplural\@secondoftwo
```

```
3941     \let\glsupcase\@firstofthree
```

```
3942     \let\glsinsert\@empty
```

```
3943     \def\glsfont{\@font}
```

```
3944     \acronymfont{\glsentryshort{#2}}#3%
```

```
3945   }%
```

Call `\@gls@link` Note that `\@gls@link` sets `\glstype`.

```
3946   \@gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
```

```
3947 }
```

```
3948   \glspostlinkhook
```

```
3949 }
```

`\Acrshort`

```
3950 \newrobustcmd*{\Acrshort}{\@gls@hyp@opt\@ns@Acrshort}
```

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

```
3951 \newcommand*{\ns@Acrshort}[2] [] {%
3952   \new@ifnextchar[{\@Acrshort{#1}{#2}}{\@Acrshort{#1}{#2} []}%
3953 }
```

Read in the final optional argument:

```
3954 \def\@Acrshort#1#2[#3] {%
3955   \glsdoifexists{#2}%
3956   {%
3957     \let\do@gl@link@checkfirsthyper\relax

3958     \def\glslabel{#2}%
3959     \let\glsifplural\@secondoftwo
3960     \let\glscapscase\@secondofthree
3961     \let\glsinsert\@empty
3962     \def\glscustomtext{%
3963       \acronymfont{\Glsentryshort{#2}}#3%
3964     }%
```

Call \@gl@link Note that \@gl@link sets \glstype.

```
3965   \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
3966   }%

3967   \glspostlinkhook
3968 }
```

\ACRshort

```
3969 \newrobustcmd*{\ACRshort}{\@gl@hyp@opt\ns@ACRshort}
```

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

```
3970 \newcommand*{\ns@ACRshort}[2] [] {%
3971   \new@ifnextchar[{\@ACRshort{#1}{#2}}{\@ACRshort{#1}{#2} []}%
3972 }
```

Read in the final optional argument:

```
3973 \def\@ACRshort#1#2[#3] {%
3974   \glsdoifexists{#2}%
3975   {%
3976     \let\do@gl@link@checkfirsthyper\relax

3977     \def\glslabel{#2}%
3978     \let\glsifplural\@secondoftwo
3979     \let\glscapscase\@thirdofthree
3980     \let\glsinsert\@empty
3981     \def\glscustomtext{%
3982       \mfirstucMakeUppercase{\acronymfont{\Glsentryshort{#2}}#3}%
3983     }%
```

Call \@gls@link Note that \@gls@link sets \glstype.

```
3984 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3985 }%

3986 \glspostlinkhook
3987 }
```

Short plural:

\acrshortpl

```
3988 \newrobustcmd*{\acrshortpl}{\@gls@hyp@opt\@ns@acrshortpl}
```

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

```
3989 \newcommand*{\ns@acrshortpl}[2] [] {%
3990 \new@ifnextchar[{\@acrshortpl{#1}{#2}}{\@acrshortpl{#1}{#2} []}%
3991 }
```

Read in the final optional argument:

```
3992 \def\@acrshortpl#1#2[#3] {%
3993 \glsdoifexists{#2}%
3994 {%
3995 \let\do@gls@link@checkfirsthyper\relax

3996 \def\glslabel{#2}%
3997 \let\glsifplural\@firstoftwo
3998 \let\glscapscase\@firstofthree
3999 \let\glsinsert\@empty
4000 \def\glscustomtext{%
4001 \acronymfont{\glsentryshortpl{#2}}#3%
4002 }%
```

Call \@gls@link Note that \@gls@link sets \glstype.

```
4003 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4004 }%

4005 \glspostlinkhook
4006 }
```

\Acrshortpl

```
4007 \newrobustcmd*{\Acrshortpl}{\@gls@hyp@opt\@ns@Acrshortpl}
```

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

```
4008 \newcommand*{\ns@Acrshortpl}[2] [] {%
4009 \new@ifnextchar[{\@Acrshortpl{#1}{#2}}{\@Acrshortpl{#1}{#2} []}%
4010 }
```

Read in the final optional argument:

```

4011 \def\@Acrshortpl#1#2[#3]{%
4012   \glsdoifexists{#2}%
4013   {%
4014     \let\do@gl@link@checkfirsthyper\relax

4015     \def\glslabel{#2}%
4016     \let\glsifplural\@firstoftwo
4017     \let\glscapscase\@secondofthree
4018     \let\glsinsert\@empty
4019     \def\glscustomtext{%
4020       \acronymfont{\Glsentryshortpl{#2}}#3%
4021     }%

```

Call \@gl@link Note that \@gl@link sets \glstype.

```

4022   \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
4023   }%

4024   \glspostlinkhook
4025 }

```

\ACRshortpl

```

4026 \newrobustcmd*{\ACRshortpl}{\@gl@hyp@opt\@ns@ACRshortpl}

```

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

```

4027 \newcommand*{\ns@ACRshortpl}[2][ ]{%
4028   \new@ifnextchar[{\@ACRshortpl{#1}{#2}}{\@ACRshortpl{#1}{#2}[]}%
4029 }

```

Read in the final optional argument:

```

4030 \def\@ACRshortpl#1#2[#3]{%
4031   \glsdoifexists{#2}%
4032   {%
4033     \let\do@gl@link@checkfirsthyper\relax

4034     \def\glslabel{#2}%
4035     \let\glsifplural\@firstoftwo
4036     \let\glscapscase\@thirdofthree
4037     \let\glsinsert\@empty
4038     \def\glscustomtext{%
4039       \mfirstucMakeUppercase{\acronymfont{\Glsentryshortpl{#2}}#3}%
4040     }%

```

Call \@gl@link Note that \@gl@link sets \glstype.

```

4041   \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
4042   }%

4043   \glspostlinkhook
4044 }

```

`\acrlong`

```
4045 \newrobustcmd*{\acrlong}{\@gls@hyp@opt\@ns@acrlong}
```

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

```
4046 \newcommand*{\ns@acrlong}[2] [] {%
4047   \new@ifnextchar[{\@acrlong{#1}{#2}}{\@acrlong{#1}{#2} []}%
4048 }
```

Read in the final optional argument:

```
4049 \def\@acrlong#1#2[#3] {%
4050   \glsdoifexists{#2}%
4051   {%
4052     \let\do@gls@link@checkfirsthyper\relax
4053   }
4054   \def\glslabel{#2}%
4055   \let\glsifplural\@secondoftwo
4056   \let\glsapscase\@firstofthree
4057   \let\glsinsert\@empty
```

Bug fix v4.02 removed `\acronymfont` from `\glscustomtext` (`\acronymfont` only designed for short form).

```
4057   \def\glscustomtext{%
4058     \glsentrylong{#2}#3%
4059   }
```

Call `\@gls@link` Note that `\@gls@link` sets `\glstype`.

```
4060   \@gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
4061   }%
4062   \glspostlinkhook
4063 }
```

`\Acrlong`

```
4064 \newrobustcmd*{\Acrlong}{\@gls@hyp@opt\@ns@Acrlong}
```

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

```
4065 \newcommand*{\ns@Acrlong}[2] [] {%
4066   \new@ifnextchar[{\@Acrlong{#1}{#2}}{\@Acrlong{#1}{#2} []}%
4067 }
```

Read in the final optional argument:

```
4068 \def\@Acrlong#1#2[#3] {%
4069   \glsdoifexists{#2}%
4070   {%
4071     \let\do@gls@link@checkfirsthyper\relax
4072   }
4073   \def\glslabel{#2}%
4074   \let\glsifplural\@secondoftwo
4075   \let\glsapscase\@secondofthree
4076   \let\glsinsert\@empty
```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```
4076 \def\glscustomtext{%
4077 \Glsentrylong{#2}#3%
4078 }%
```

Call \@gls@link. Note that \@gls@link sets \glstype.

```
4079 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4080 }%

4081 \glspostlinkhook
4082 }
```

\ACRlong

```
4083 \newrobustcmd*{\ACRlong}{\@gls@hyp@opt\@ns@ACRlong}
```

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

```
4084 \newcommand*{\ns@ACRlong}[2][ ]{%
4085 \new@ifnextchar[{\@ACRlong{#1}{#2}}{\@ACRlong{#1}{#2}[]}%
4086 }
```

Read in the final optional argument:

```
4087 \def\@ACRlong#1#2[#3]{%
4088 \glsdoifexists{#2}%
4089 {%
4090 \let\do@gls@link@checkfirsthyper\relax

4091 \def\glslabel{#2}%
4092 \let\glsifplural\@secondoftwo
4093 \let\glscapscase\@thirdofthree
4094 \let\glsinsert\@empty
```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```
4095 \def\glscustomtext{%
4096 \mfirstucMakeUppercase{\glsentrylong{#2}#3}%
4097 }%
```

Call \@gls@link. Note that \@gls@link sets \glstype.

```
4098 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4099 }%

4100 \glspostlinkhook
4101 }
```

Short plural:

\acrlongpl

```
4102 \newrobustcmd*{\acrlongpl}{\@gls@hyp@opt\@ns@acrlongpl}
```

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

```
4103 \newcommand*{\ns@acrlongpl}[2] [] {%
4104   \new@ifnextchar[{\@acrlongpl{#1}{#2}}{\@acrlongpl{#1}{#2} []}%
4105 }
```

Read in the final optional argument:

```
4106 \def\@acrlongpl#1#2[#3] {%
4107   \glsdoifexists{#2}%
4108   {%
4109     \let\do@gl@link@checkfirsthyper\relax

4110     \def\glslabel{#2}%
4111     \let\glsifplural\@firstoftwo
4112     \let\glscapscase\@firstofthree
4113     \let\glsinsert\@empty
```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```
4114     \def\glscustomtext{%
4115       \glstrylongpl{#2}#3%
4116     }%
```

Call \@gl@link. Note that \@gl@link sets \glstype.

```
4117     \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
4118   }%
```

```
4119   \glspostlinkhook
4120 }
```

\Acrlongpl

```
4121 \newrobustcmd*{\Acrlongpl}{\@gl@hyp@opt\ns@Acrlongpl}
```

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

```
4122 \newcommand*{\ns@Acrlongpl}[2] [] {%
4123   \new@ifnextchar[{\@Acrlongpl{#1}{#2}}{\@Acrlongpl{#1}{#2} []}%
4124 }
```

Read in the final optional argument:

```
4125 \def\@Acrlongpl#1#2[#3] {%
4126   \glsdoifexists{#2}%
4127   {%
4128     \let\do@gl@link@checkfirsthyper\relax

4129     \def\glslabel{#2}%
4130     \let\glsifplural\@firstoftwo
4131     \let\glscapscase\@secondofthree
4132     \let\glsinsert\@empty
```


Bug fix v4.02 removed `\acronymfont` from `\glscustomtext` (`\acronymfont` only designed for short form).

```
4133 \def\glscustomtext{%
4134 \Glsentrylongpl{#2}#3%
4135 }%
```

Call `\@gls@link`. Note that `\@gls@link` sets `\glstype`.

```
4136 \@gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
4137 }%

4138 \glspostlinkhook
4139 }
```

`\ACRlongpl`

```
4140 \newrobustcmd*{\ACRlongpl}{\@gls@hyp@opt\ns@ACRlongpl}
```

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

```
4141 \newcommand*{\ns@ACRlongpl}[2] [] {%
4142 \new@ifnextchar[{\@ACRlongpl{#1}{#2}}{\@ACRlongpl{#1}{#2} []}%
4143 }
```

Read in the final optional argument:

```
4144 \def\@ACRlongpl#1#2[#3] {%
4145 \glsdoifexists{#2}%
4146 {%
4147 \let\do@gls@link@checkfirsthyper\relax

4148 \def\glslabel{#2}%
4149 \let\glsifplural\@firstoftwo
4150 \let\glscapscase\@thirdofthree
4151 \let\glsinsert\@empty
```

Bug fix v4.02 removed `\acronymfont` from `\glscustomtext` (`\acronymfont` only designed for short form).

```
4152 \def\glscustomtext{%
4153 \mfirstucMakeUppercase{\Glsentrylongpl{#2}#3}%
4154 }%
```

Call `\@gls@link`. Note that `\@gls@link` sets `\glstype`.

```
4155 \@gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
4156 }%

4157 \glspostlinkhook
4158 }
```

1.11.2 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>}
```

```
4159 \newcommand*{\@gls@entry@field}[2]{%
4160   \csname glo@glsdetoklabel{#1}@#2\endcsname
4161 }
```

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

```
4162 \newcommand*{\glsletentryfield}[3]{%
4163   \letcs{#1}{glo@glsdetoklabel{#2}@#3}%
4164 }
```

`\@Gls@entry@field` Generic first letter uppercase version.

```
\@Gls@entry@field{<label>}{<field>}
```

```
4165 \newcommand*{\@Gls@entry@field}[2]{%
4166   \glsdoifexistsordo{#1}%
4167   {%
4168     \letcs\@glo@text{glo@glsdetoklabel{#1}@#2}%
4169     \ifdef\@glo@text
4170     {%
4171       \xmakefirstuc{\@glo@text}%
4172     }%
4173   }%
4174   ??\PackageError{glossaries}{The field ‘#2’ doesn’t exist for glossary
4175     entry ‘\glsdetoklabel{#1}’}{Check you have correctly spelt the entry
4176     label and the field name}%
4177   }%
4178 }%
4179 {%
4180   ??%
4181 }%
4182 }
```

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`

```
4183 \newcommand*{\glsentryname}[1]{\@gls@entry@field{#1}{name}}
```

`\Glsentryname`

```

4184 \newrobustcmd*{\Glsentryname}[1]{%
4185   \@Gls@entryname{#1}%
4186 }

```

\@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:

```

4187 \newcommand*{\@Gls@entryname}[1]{%
4188   \@Gls@entry@field{#1}{name}%
4189 }

```

\@Gls@acentryname Now the behaviour when \setacronymstyle is used:

```

4190 \newcommand*{\@Gls@acentryname}[1]{%
4191   \ifglshaslong{#1}%
4192   {%
4193     \letcs\@glo@text{glo@\glsetoklabel{#1}@name}%
4194     \expandafter\@gls@getbody\@glo@text{}\@nil
4195     \expandafter\ifx\@gls@body\glentrylong\relax
4196     \expandafter\Glsentrylong\@gls@rest
4197   }else
4198     \expandafter\ifx\@gls@body\glentryshort\relax
4199     \expandafter\Glsentryshort\@gls@rest
4200   }else
4201     \expandafter\ifx\@gls@body\acronymfont\relax

```

Temporarily make \glentryshort behave like \Glsentryshort. (This is on the assumption that the argument of \acronymfont is \glentryshort{\label}, as that's the behaviour of the predefined acronym styles.) This is scoped to localise the effect of the assignment.

```

4202     {%
4203       \let\glentryshort\Glsentryshort
4204       \@glo@text
4205     }%
4206   }else
4207     \xmakefirstuc{\@glo@text}%
4208   \fi
4209 \fi
4210 \fi
4211 }%
4212 {%

```

Not an acronym

```

4213   \@Gls@entry@field{#1}{name}%
4214 }%
4215 }

```

Get the entry description (as specified by the description when the entry was defined). The argument is the label associated with the entry. Note that unless you used `description=false` in the `sanitize` package option you may get unexpected results if the description key contained any commands.

`\glsentrydesc`

```
4216 \newcommand*{\glsentrydesc}[1]{\@gls@entry@field{#1}{desc}}
```

`\Glsentrydesc`

```
4217 \newrobustcmd*{\Glsentrydesc}[1]{%
4218   \@Gls@entry@field{#1}{desc}%
4219 }
```

Plural form:

`\glsentrydescplural`

```
4220 \newcommand*{\glsentrydescplural}[1]{%
4221   \@gls@entry@field{#1}{descplural}%
4222 }
```

`\Glsentrydescplural`

```
4223 \newrobustcmd*{\Glsentrydescplural}[1]{%
4224   \@Gls@entry@field{#1}{descplural}%
4225 }
```

Get the entry text, as specified by the text key when the entry was defined.
The argument is the label associated with the entry:

`\glsentrytext`

```
4226 \newcommand*{\glsentrytext}[1]{\@gls@entry@field{#1}{text}}
```

`\Glsentrytext`

```
4227 \newrobustcmd*{\Glsentrytext}[1]{%
4228   \@Gls@entry@field{#1}{text}%
4229 }
```

Get the plural form:

`\glsentryplural`

```
4230 \newcommand*{\glsentryplural}[1]{%
4231   \@gls@entry@field{#1}{plural}%
4232 }
```

`\Glsentryplural`

```
4233 \newrobustcmd*{\Glsentryplural}[1]{%
4234   \@Gls@entry@field{#1}{plural}%
4235 }
```

Get the symbol associated with this entry. The argument is the label associated with the entry.

`\glsentrysymbol`

```
4236 \newcommand*{\glsentrysymbol}[1]{%
4237   \@gls@entry@field{#1}{symbol}%
4238 }
```

`\Glsentrysymbol`

```
4239 \newrobustcmd*{\Glsentrysymbol}[1]{%  
4240   \@Gls@entry@field{#1}{symbol}%  
4241 }
```

Plural form:

`lsentrysymbolplural`

```
4242 \newcommand*{\glsentrysymbolplural}[1]{%  
4243   \@gls@entry@field{#1}{symbolplural}%  
4244 }
```

`lsentrysymbolplural`

```
4245 \newrobustcmd*{\Glsentrysymbolplural}[1]{%  
4246   \@Gls@entry@field{#1}{symbolplural}%  
4247 }
```

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`

```
4248 \newcommand*{\glsentryfirst}[1]{%  
4249   \@gls@entry@field{#1}{first}%  
4250 }
```

`\Glsentryfirst`

```
4251 \newrobustcmd*{\Glsentryfirst}[1]{%  
4252   \@Gls@entry@field{#1}{first}%  
4253 }
```

Get the plural form (as specified by the firstplural key when the entry was defined).

`glsentryfirstplural`

```
4254 \newcommand*{\glsentryfirstplural}[1]{%  
4255   \@gls@entry@field{#1}{firstpl}%  
4256 }
```

`Glsentryfirstplural`

```
4257 \newrobustcmd*{\Glsentryfirstplural}[1]{%  
4258   \@Gls@entry@field{#1}{firstpl}%  
4259 }
```

Display the glossary type with which this entry is associated (as specified by the type key used when the entry was defined)

`\glsentrytype`

```
4260 \newcommand*{\glsentrytype}[1]{\@gls@entry@field{#1}{type}}
```

Display the sort text used for this entry. Note that the sort key is sanitize, so unexpected results may occur if the sort key contained commands.

`\glsentrysort`

```
4261 \newcommand*{\glsentrysort}[1]{%  
4262   \@gls@entry@field{#1}{sort}%  
4263 }
```

`\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.

```
4264 \newcommand*{\glsentryuseri}[1]{%  
4265   \@gls@entry@field{#1}{useri}%  
4266 }
```

`\Glsentryuseri`

```
4267 \newrobustcmd*{\Glsentryuseri}[1]{%  
4268   \@Gls@entry@field{#1}{useri}%  
4269 }
```

`\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.

```
4270 \newcommand*{\glsentryuserii}[1]{%  
4271   \@gls@entry@field{#1}{userii}%  
4272 }
```

`\Glsentryuserii`

```
4273 \newrobustcmd*{\Glsentryuserii}[1]{%  
4274   \@Gls@entry@field{#1}{userii}%  
4275 }
```

`\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.

```
4276 \newcommand*{\glsentryuseriii}[1]{%  
4277   \@gls@entry@field{#1}{useriii}%  
4278 }
```

`\Glsentryuseriii`

```
4279 \newrobustcmd*{\Glsentryuseriii}[1]{%  
4280   \@Gls@entry@field{#1}{useriii}%  
4281 }
```

`\glsentryuseriv` Get the fourth user key (as specified by the user4 when the entry was defined).
The argument is the label associated with the entry.

```
4282 \newcommand*{\glsentryuseriv}[1]{%  
4283   \@gls@entry@field{#1}{useriv}%  
4284 }
```

\Glsentryuseriv

```
4285 \newrobustcmd*{\Glsentryuseriv}[1]{%
4286   \@Gls@entry@field{#1}{useriv}%
4287 }
```

\glsentryuserv Get the fifth user key (as specified by the user5 when the entry was defined).
The argument is the label associated with the entry.

```
4288 \newcommand*{\glsentryuserv}[1]{%
4289   \@Gls@entry@field{#1}{userv}%
4290 }
```

\Glsentryuserv

```
4291 \newrobustcmd*{\Glsentryuserv}[1]{%
4292   \@Gls@entry@field{#1}{userv}%
4293 }
```

\glsentryuservi Get the sixth user key (as specified by the user6 when the entry was defined).
The argument is the label associated with the entry.

```
4294 \newcommand*{\glsentryuservi}[1]{%
4295   \@Gls@entry@field{#1}{uservi}%
4296 }
```

\Glsentryuservi

```
4297 \newrobustcmd*{\Glsentryuservi}[1]{%
4298   \@Gls@entry@field{#1}{uservi}%
4299 }
```

\glsentryshort Get the short key (as specified by the short the entry was defined). The argument is the label associated with the entry.

```
4300 \newcommand*{\glsentryshort}[1]{\@Gls@entry@field{#1}{short}}
```

\Glsentryshort

```
4301 \newrobustcmd*{\Glsentryshort}[1]{%
4302   \@Gls@entry@field{#1}{short}%
4303 }
```

\glsentryshortpl Get the short plural key (as specified by the shortplural the entry was defined).
The argument is the label associated with the entry.

```
4304 \newcommand*{\glsentryshortpl}[1]{\@Gls@entry@field{#1}{shortpl}}
```

\Glsentryshortpl

```
4305 \newrobustcmd*{\Glsentryshortpl}[1]{%
4306   \@Gls@entry@field{#1}{shortpl}%
4307 }
```

\glsentrylong Get the long key (as specified by the long the entry was defined). The argument is the label associated with the entry.

```
4308 \newcommand*{\glsentrylong}[1]{\@Gls@entry@field{#1}{long}}
```

```

\Glsentrylong
4309 \newrobustcmd*{\Glsentrylong}[1]{%
4310   \@Gls@entry@field{#1}{long}%
4311 }

\glsentrylongpl  Get the long plural key (as specified by the longplural the entry was defined).
                  The argument is the label associated with the entry.
4312 \newcommand*{\glsentrylongpl}[1]{\@Gls@entry@field{#1}{longpl}}

\Glsentrylongpl
4313 \newrobustcmd*{\Glsentrylongpl}[1]{%
4314   \@Gls@entry@field{#1}{longpl}%
4315 }

Short cut macros to access full form:

\glsentryfull
4316 \newcommand*{\glsentryfull}[1]{%
4317   \acrfullformat{\glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%
4318 }

\Glsentryfull
4319 \newrobustcmd*{\Glsentryfull}[1]{%
4320   \acrfullformat{\Glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%
4321 }

\glsentryfullpl
4322 \newcommand*{\glsentryfullpl}[1]{%
4323   \acrfullformat{\glsentrylongpl{#1}}{\acronymfont{\glsentryshortpl{#1}}}%
4324 }

\Glsentryfullpl
4325 \newrobustcmd*{\Glsentryfullpl}[1]{%
4326   \acrfullformat{\Glsentrylongpl{#1}}{\acronymfont{\glsentryshortpl{#1}}}%
4327 }

\glsentrynumberlist  Displays the number list as is.
4328 \newcommand*{\glsentrynumberlist}[1]{%
4329   \glsdoifexists{#1}%
4330   {%
4331     \@Gls@entry@field{#1}{numberlist}%
4332   }%
4333 }

\displaynumberlist  Formats the number list for the given entry label. Doesn't work with hyperref.
4334 \@ifpackageloaded{hyperref} {%
4335   \newcommand*{\displaynumberlist}[1]{%
4336     \GlossariesWarning

```



```

4337   {%
4338       \string\glsdisplaynumberlist\space
4339       doesn't work with hyperref.^^JUsing
4340       \string\glsentrynumberlist\space instead%
4341   }%
4342   \glsentrynumberlist{#1}%
4343 }%
4344 }%
4345 {%
4346   \newcommand*{\glsdisplaynumberlist}[1]{%
4347       \glsdoifexists{#1}%
4348       {%
4349           \bgroup
4350
4351           \edef\@glo@label{\glsdetoklabel{#1}}%
4352           \let\@org@glsnnumberformat\glsnnumberformat
4353           \def\glsnnumberformat##1{##1}%
4354           \protected@edef\the@numberlist{%
4355               \csname glo@\@glo@label @numberlist\endcsname}%
4356           \def\@gls@numlist@sep{}%
4357           \def\@gls@numlist@nextsep{}%
4358           \def\@gls@numlist@lastsep{}%
4359           \def\@gls@thislist{}%
4360           \def\@gls@donext@def{}%
4361           \renewcommand\do[1]{%
4362               \protected@edef\@gls@thislist{%
4363                   \@gls@thislist
4364                   \noexpand\@gls@numlist@sep
4365                   ##1%
4366               }%
4367               \let\@gls@numlist@sep\@gls@numlist@nextsep
4368               \def\@gls@numlist@nextsep{\glsnnumlistsep}%
4369               \@gls@donext@def
4370               \def\@gls@donext@def{%
4371                   \def\@gls@numlist@lastsep{\glsnnumlistlastsep}%
4372               }%
4373               \expandafter \glsnnumlistparser \expandafter{\the@numberlist}%
4374               \let\@gls@numlist@sep\@gls@numlist@lastsep
4375               \@gls@thislist
4376           \egroup
4377       }%
4378   }
4379 }

```

\glsnnumlistsep

```
4380 \newcommand*{\glsnnumlistsep}{, }
```

\glsnnumlistlastsep

```
4381 \newcommand*{\glsnnumlistlastsep}{ \& }
```

`\glshyperlink` Provide a hyperlink to a glossary entry without adding information to the glossary file. The entry needs to be added using a command like `\glslink` or `\glsadd` 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.

```
4382 \newcommand*{\glshyperlink}[2][\glsentrytext{\@glo@label}]{%
4383 \def\@glo@label{#2}%
4384 \@glslink{\glo@linkprefix\glsdetoklabel{#2}}{#1}}
```

1.12 Adding an entry to the glossary without generating text

The following keys are provided for `\glsadd` and `\glsaddall`:

```
4385 \define@key{glossadd}{counter}{\def\@gls@counter{#1}}
4386 \define@key{glossadd}{format}{\def\@glsnumberformat{#1}}
```

This key is only used by `\glsaddall`:

```
4387 \define@key{glossadd}{types}{\def\@glo@type{#1}}
```

`\glsadd[<options>]{<label>}`

Add a term to the glossary without generating any link text. The optional argument indicates which counter to use, and how to format it (using a key-value list) the second argument is the entry label. Note that *<options>* only has two keys: counter and format (the types key will be ignored).

`\glsadd`

```
4388 \newrobustcmd*{\glsadd}[2][\@gls@counter]{%
```

Need to move to horizontal mode if not already in it, but only if not in preamble.

```
4389 \@gls@adjustmode
4390 \glsdoifexists{#2}%
4391 {%
4392 \def\@glsnumberformat{\glsnumberformat}%
4393 \edef\@gls@counter{\csname glo@%glsdetoklabel{#2}@counter\endcsname}%
4394 \setkeys{glossadd}{#1}%
```

Store the entry's counter in `\theglsentrycounter`

```
4395 \@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.

```
4396 \@do@wrglossary{#2}%
4397 }%
4398 }
```

`\@gls@adjustmode`

```
4399 \newcommand*{\@gls@adjustmode}{}
4400 \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`

```
4401 \newrobustcmd*{\glsaddall}[1] [] {%
4402   \edef\@glo@type{\@glo@types}%
4403   \setkeys{glossadd}{#1}%
4404   \forallglsentries[\@glo@type]{\@glo@entry}{%
4405     \glsadd[#1]{\@glo@entry}%
4406   }%
4407 }
```

`\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.

```
4408 \newrobustcmd*{\glsaddallunused}[1] [\@glo@types] {%
4409   \forallglsentries[#1]{\@glo@entry}%
4410   {%
4411     \ifglsused{\@glo@entry}{\@glsadd[format=glsignore]{\@glo@entry}}%
4412   }%
4413 }
```

`\glsignore`

```
4414 \newcommand*{\glsignore}[1] {}
```

1.13 Creating associated files

The `\writeist` command creates the associated customized `.ist` makeindex style file. While defining this command, some characters have their catcodes temporarily changed to ensure they get written to the `.ist` file correctly. The `makeindex` actual character (usually `@`) is redefined to be a `?`, to allow internal commands to be written to the glossary file output file.

The special characters are stored in `\@gls@actualchar`, `\@gls@encapchar`, `\@gls@levelchar` and `\@gls@quotechar` to make them easier to use later, but don't change these values, because the characters are encoded in the command definitions that are used to escape the special characters (which means that the user no longer needs to worry about `makeindex` special characters).

The symbols and numbers label for group headings are hardwired into the `.ist` file as `glsymbols` and `glsnumbers`, the group titles can be translated (so that `\glsymbolsgroupname` replaces `glsymbols` and `\glsnumbersgroupname`

replaces `glsnumbers`) using the command `\glsgetgrouptitle` which is defined in `.` This is done to prevent any problem characters in `\glssymbolsgroupname` and `\glsnumbersgroupname` from breaking hyperlinks.

`\glsopenbrace` Define `\glsopenbrace` to make it easier to write an opening brace to a file.

```
4415 \edef\glsopenbrace{\expandafter\@gobble\string\{}
```

`\glsclosebrace` Define `\glsclosebrace` to make it easier to write an opening brace to a file.

```
4416 \edef\glsclosebrace{\expandafter\@gobble\string\}}
```

`\glsbackslash` Define `\glsbackslash` to make it easier to write a backslash to a file.

```
4417 \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.

```
4418 \edef\glsquote#1{\string"#1\string"}
```

`\glspercentchar` Define `\glspercentchar` to make it easier to write a percent character to a file.

```
4419 \edef\glspercentchar{\expandafter\@gobble\string\%}
```

`\glstildechar` Define `\glstildechar` to make it easier to write a tilde character to a file.

```
4420 \edef\glstildechar{\string~}
```

`\@glsfirstletter` Define the first letter to come after the digits 0,...,9. Only required for `xindy`.

```
4421 \ifglsxindy
```

```
4422   \newcommand*{\@glsfirstletter}{A}
```

```
4423 \fi
```

`\stLetterAfterDigits` Sets the first letter to come after the digits 0,...,9.

```
4424 \ifglsxindy
```

```
4425   \newcommand*{\GlsSetXdyFirstLetterAfterDigits}[1]{%
```

```
4426     \renewcommand*{\@glsfirstletter}{#1}}
```

```
4427 \else
```

```
4428   \newcommand*{\GlsSetXdyFirstLetterAfterDigits}[1]{%
```

```
4429     \glsnoxywarning\GlsSetXdyFirstLetterAfterDigits}
```

```
4430 \fi
```

`\@glsminrange` Define the minimum number of successive location references to merge into a range.

```
4431 \newcommand*{\@glsminrange}{2}
```

`\setXdyMinRangeLength` 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`.

```
4432 \ifglsxindy
```

```
4433   \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
```

```
4434     \renewcommand*{\@glsminrange}{#1}}
```

```

4435 \else
4436   \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
4437     \glsnoxindywarning\GlsSetXdyMinRangeLength}
4438 \fi

\writeist

4439 \ifglxindy
  Code to use if xindy is required.
4440   \def\writeist{%
  Define write register if not already defined
4441     \ifundef{\glswrite}{\newwrite\glswrite}{}%
  Update attributes list
4442     \@gls@addpredefinedattributes
  Open the file.
4443     \openout\glswrite=\istfilename
  Write header comment at the start of the file
4444     \write\glswrite{;; xindy style file created by the glossaries
4445       package}%
4446     \write\glswrite{;; for document '\jobname' on
4447       \the\year-\the\month-\the\day}%
  Specify the required styles
4448     \write\glswrite{^^J; required styles^^J}
4449     \@for\@xdystyle:=\@xdyrequiredstyles\do{%
4450       \ifx\@xdystyle\@empty
4451       \else
4452         \protected@write\glswrite{{(require
4453           \string"\@xdystyle.xdy\string")}}%
4454       \fi
4455     }%
  List the allowed attributes (possible values used by the format key)
4456     \write\glswrite{^^J%
4457       ; list of allowed attributes (number formats)^^J}%
4458     \write\glswrite{(define-attributes ((\@xdyattributes)))}%
  Define any additional alphabets
4459     \write\glswrite{^^J; user defined alphabets^^J}%
4460     \write\glswrite{\@xdyuseralphabets}%
  Define location classes.
4461     \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.
4462     \@for\@gls@classI:=\@gls@xdy@locationlist\do{%

```

Case were $\langle Hprefix \rangle$ is empty:

```

4463 \protected@write\glswrite{}\{(define-location-class
4464 \string"\@gls@classI\string"^^J\space\space\space
4465 (
4466 :sep "{\{"
4467 \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
4468 :sep "\}"
4469 )
4470 ^^J\space\space\space
4471 :min-range-length \@glsminrange^^J%
4472 )
4473 }%
```

Nested iteration over all classes:

```

4474 {%
4475 \@for\@gls@classII:=\@gls@xdy@locationlist\do{%
4476 \protected@write\glswrite{}\{(define-location-class
4477 \string"\@gls@classII-\@gls@classI\string"
4478 ^^J\space\space\space
4479 (
4480 :sep "{"
4481 \csname @gls@xdy@Lclass@\@gls@classII\endcsname\space
4482 :sep "\{"
4483 \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
4484 :sep "\}"
4485 )
4486 ^^J\space\space\space
4487 :min-range-length \@glsminrange^^J%
4488 )
4489 }%
4490 }%
4491 }%
4492 }%
```

User defined location classes (needs checking for new location format).

```

4493 \write\glswrite{^^J; user defined location classes}%
4494 \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 $\backslash\text{glsseeformat}$ which xindy won't recognise.)

```

4495 \write\glswrite{^^J; define cross-reference class^^J}%
4496 \write\glswrite{(define-crossref-class \string"see\string"
4497 :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 $\backslash\text{glsseeformat}$ which gets ignored. (When using `makeindex` this final argument contains the location information which is not required.)

```

4498 \write\glswrite{(markup-crossref-list
```

```

4499      :class \string"see\string"^^J\space\space\space
4500      :open \string"\string\glssseeformat\string"
4501      :close \string"{}\string")}%

```

List the order to sort the classes.

```

4502      \write\glswrite{^^J; define the order of the location classes}%
4503      \write\glswrite{(define-location-class-order
4504      (\@xdylocationclassorder))}%

```

Specify what to write to the start and end of the glossary file.

```

4505      \write\glswrite{^^J; define the glossary markup^^J}%

4506      \write\glswrite{(markup-index^^J\space\space\space
4507      :open \string"\string
4508      \glossarysection[\string\glossarytoctitle]{\string
4509      \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)

```

4510      \@for\@this@ctr:=\@xdycounters\do{%
4511      {%
4512      \@for\@this@attr:=\@xdyattributelist\do{%
4513      \protected\write\glswrite{}\{\string\providecommand*%
4514      \expandafter\string
4515      \csname glsX\@this@ctr X\@this@attr\endcsname[2]%
4516      {%
4517      \string\setentrycounter
4518      [\expandafter\@gobble\string\#1]{\@this@ctr}%
4519      \expandafter\string
4520      \csname\@this@attr\endcsname
4521      {\expandafter\@gobble\string\#2}%
4522      }%
4523      }%
4524      }%
4525      }%
4526      }%

```

Add the end part of the open tag and the rest of the markup-index information:

```

4527      \write\glswrite{%
4528      \string\begin
4529      {theglossary}\string\glossaryheader\glstildechar n\string" ^^J\space
4530      \space\space:close \string"\glpercentchar\glstildechar n\string
4531      \end{theglossary}\string\glossarypostamble
4532      \glstildechar n\string" ^^J\space\space\space
4533      :tree)}}%

```

Specify what to put between letter groups

```

4534      \write\glswrite{(markup-letter-group-list
4535      :sep \string"\string\glsgroupskip\glstildechar n\string")}%

```

Specify what to put between entries

```

4536      \write\glswrite{(markup-indexentry

```

```

4537      :open \string"\string\relax \string\glsresetentrylist
4538      \glstildechar n\string"))}%

```

Specify how to format entries

```

4539      \write\glswrite{(markup-locclass-list :open
4540      \string"\glsopenbrace\string\glossaryentrynumbers
4541      \glsopenbrace\string\relax\space \string"^^J\space\space\space
4542      :sep \string", \string"
4543      :close \string"\glsclosebrace\glsclosebrace\string"))}%

```

Specify how to separate location numbers

```

4544      \write\glswrite{(markup-locref-list
4545      :sep \string"\string\delimN\space\string"))}%

```

Specify how to indicate location ranges

```

4546      \write\glswrite{(markup-range
4547      :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.

```

4548      \@onelevel@sanitize\gls@suffixF
4549      \@onelevel@sanitize\gls@suffixFF

4550      \ifx\gls@suffixF\@empty
4551      \else
4552      \write\glswrite{(markup-range
4553      :close "\gls@suffixF" :length 1 :ignore-end)}%
4554      \fi
4555      \ifx\gls@suffixFF\@empty
4556      \else
4557      \write\glswrite{(markup-range
4558      :close "\gls@suffixFF" :length 2 :ignore-end)}%
4559      \fi

```

Specify how to format locations.

```

4560      \write\glswrite{^^J; define format to use for locations^^J}%
4561      \write\glswrite{\@xdylocref}%

```

Specify how to separate letter groups.

```

4562      \write\glswrite{^^J; define letter group list format^^J}%
4563      \write\glswrite{(markup-letter-group-list
4564      :sep \string"\string\glsgroupskip\glstildechar n\string"))}%

```

Define letter group headings.

```

4565      \write\glswrite{^^J; letter group headings^^J}%
4566      \write\glswrite{(markup-letter-group
4567      :open-head \string"\string\glsgroupheading
4568      \glsopenbrace\string"^^J\space\space\space
4569      :close-head \string"\glsclosebrace\string"))}%

```

Define additional letter groups.

```

4570      \write\glswrite{^^J; additional letter groups^^J}%
4571      \write\glswrite{\@xdylettergroups}%

```


Define additional sort rules

```

4572 \write\glswrite{^^J; additional sort rules^^J}
4573 \write\glswrite{\@xdysortrules}%

Close the style file
4574 \closeout\glswrite

Suppress any further calls.
4575 \let\writeist\relax
4576 }
4577 \else

Code to use if makeindex is required.
4578 \edef\@gls@actualchar{\string?}
4579 \edef\@gls@encapchar{\string|}
4580 \edef\@gls@levelchar{\string!}
4581 \edef\@gls@quotechar{\string"}
4582 \def\writeist{\relax
4583 \ifundef\glswrite{\newwrite\glswrite}{\relax
4584 \openout\glswrite=\istfilename
4585 \write\glswrite{\glspersentchar\space makeindex style file
4586 created by the glossaries package}
4587 \write\glswrite{\glspersentchar\space for document
4588 '\jobname' on \the\year-\the\month-\the\day}
4589 \write\glswrite{actual '\@gls@actualchar'}
4590 \write\glswrite{encap '\@gls@encapchar'}
4591 \write\glswrite{level '\@gls@levelchar'}
4592 \write\glswrite{quote '\@gls@quotechar'}
4593 \write\glswrite{keyword \string"\string\glossaryentry\string"}
4594 \write\glswrite{preamble \string"\string\glossarysection[\string
4595 \glossarytoctitle]{\string\glossarytitle}\string
4596 \glossarypreamble\string\n\string\begin{theglossary}\string
4597 \glossaryheader\string\n\string"}
4598 \write\glswrite{postamble \string"\string%\string\n\string
4599 \end{theglossary}\string\glossarypostamble\string\n
4600 \string"}
4601 \write\glswrite{group_skip \string"\string\glsgroupskip\string\n
4602 \string"}
4603 \write\glswrite{item_0 \string"\string%\string\n\string"}
4604 \write\glswrite{item_1 \string"\string%\string\n\string"}
4605 \write\glswrite{item_2 \string"\string%\string\n\string"}
4606 \write\glswrite{item_01 \string"\string%\string\n\string"}
4607 \write\glswrite{item_x1
4608 \string"\string\relax \string\glsresetentrylist\string\n
4609 \string"}
4610 \write\glswrite{item_12 \string"\string%\string\n\string"}
4611 \write\glswrite{item_x2
4612 \string"\string\relax \string\glsresetentrylist\string\n
4613 \string"}

4614 \write\glswrite{delim_0 \string"\string\{ \string

```

```

4615     \glossaryentrynumbers\string\{\string\relax \string"}
4616 \write\glswrite{delim_1 \string"\string\{\string
4617     \glossaryentrynumbers\string\{\string\relax \string"}
4618 \write\glswrite{delim_2 \string"\string\{\string
4619     \glossaryentrynumbers\string\{\string\relax \string"}
4620 \write\glswrite{delim_t \string"\string\}\string\}\string"}
4621 \write\glswrite{delim_n \string"\string\delimN \string"}
4622 \write\glswrite{delim_r \string"\string\delimR \string"}
4623 \write\glswrite{headings_flag 1}
4624 \write\glswrite{heading_prefix
4625     \string"\string\glsgroupheading\string\{\string"}
4626 \write\glswrite{heading_suffix
4627     \string"\string\}\string\relax
4628     \string\glsgroupresetentrylist \string"}
4629 \write\glswrite{symhead_positive \string"glssymbols\string"}
4630 \write\glswrite{numhead_positive \string"glsglnumbers\string"}
4631 \write\glswrite{page_compositor \string"glsglcompositor\string"}
4632 \@glsgl@escbsdq\glsgl@suffixF
4633 \@glsgl@escbsdq\glsgl@suffixFF
4634 \ifx\glsgl@suffixF\@empty
4635 \else
4636     \write\glswrite{suffix_2p \string"\glsgl@suffixF\string"}
4637 \fi
4638 \ifx\glsgl@suffixFF\@empty
4639 \else
4640     \write\glswrite{suffix_3p \string"\glsgl@suffixFF\string"}
4641 \fi
4642 \closeout\glswrite
4643 \let\writeist\relax
4644 }
4645 \fi

```

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`

```

4646 \newcommand{\noist}{%
    Update attributes list
4647 \@glsgl@addpredefinedattributes
4648 \let\writeist\relax
4649 }

```

`\@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`

```

4650 \newcommand*\@makeglossary}[1]{%
4651   \ifglossaryexists{#1}%
4652   {%
      Only create a new write if savewrites=false otherwise create a token to collect
      the information.
4653   \ifglssavewrites
4654     \expandafter\newtoks\csname glo@#1@filetok\endcsname
4655   \else
4656     \expandafter\newwrite\csname glo@#1@file\endcsname
4657     \expandafter\@glsopenfile\csname glo@#1@file\endcsname{#1}%
4658   \fi
4659   \@gls@renewglossary
4660   \writeist
4661 }%
4662 {%
4663   \PackageError{glossaries}%
4664   {Glossary type ‘#1’ not defined}%
4665   {New glossaries must be defined before using \string\makeglossary}%
4666 }%
4667 }
```

`\@glsopenfile` Open write file associated with the given glossary.

```

4668 \newcommand*\@glsopenfile}[2]{%
4669   \immediate\openout#1=\jobname.\csname @glotype@#2@out\endcsname
4670   \PackageInfo{glossaries}{Writing glossary file
4671     \jobname.\csname @glotype@#2@out\endcsname}%
4672 }
```

`\@closegls`

```

4673 \newcommand*\@closegls}[1]{%
4674   \closeout\csname glo@#1@file\endcsname
4675 }
4676 % \end{macrocode}
4677 %\end{macro}
4678 %
4679 %\begin{macro}\@gls@automake}
4680 %\changes{4.08}{2014-07-30}{new}
4681 % \begin{macrocode}
4682 \ifglxsindy
4683 \newcommand*\@gls@automake}[1]{%
4684   \ifglossaryexists{#1}
4685   {%
4686     \@closegls{#1}%
```

```

4687 \ifdefstring{\glsorder}{letter}%
4688 {\def\@gls@order{-M ord/letorder }}%
4689 {\let\@gls@order\@empty}%
4690 \ifcsundef{\xdy@#1@language}%
4691 {\let\@gls@langmod\@xdy@main@language}%
4692 {\letcs\@gls@langmod{\xdy@#1@language}}%
4693 \edef\@gls@dothiswrite{\noexpand\write18{xindy
4694 -I xindy
4695 \@gls@order
4696 -L \@gls@langmod\space
4697 -M \@gls@istfilebase\space
4698 -C \@gls@codepage\space
4699 -t \jobname.\csuse{\@glotype@#1@log}
4700 -o \jobname.\csuse{\@glotype@#1@in}
4701 \jobname.\csuse{\@glotype@#1@out}}}%
4702 }%
4703 \@gls@dothiswrite
4704 }%
4705 {%
4706 \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
4707 }%
4708 }
4709 \else
4710 \newcommand*{\@gls@automake}[1]{%
4711 \ifglossaryexists{#1}
4712 {%
4713 \@closegls{#1}%
4714 \ifdefstring{\glsorder}{letter}%
4715 {\def\@gls@order{-l }}%
4716 {\let\@gls@order\@empty}%
4717 \edef\@gls@dothiswrite{\noexpand\write18{makeindex \@gls@order
4718 -s \istfilename\space
4719 -t \jobname.\csuse{\@glotype@#1@log}
4720 -o \jobname.\csuse{\@glotype@#1@in}
4721 \jobname.\csuse{\@glotype@#1@out}}}%
4722 }%
4723 \@gls@dothiswrite
4724 }%
4725 {%
4726 \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
4727 }%
4728 }
4729 \fi

```

`\warn@nomakeglossaries` Issue warning that `\makeglossaries` hasn't been used.

```
4730 \newcommand*{\@warn@nomakeglossaries}{}%
```

Only use this if warning if `\printglossary` has been used without `\makeglossaries`

```
4731 \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`

```

4732 \newcommand*{\makeglossaries}{%
    Define the write used for style file also used for all other output files if
    savewrites=true.
4733 \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.
4734 \protected@write\@auxout{}{\string\providecommand\string\@glsorder[1]{}%
4735 \protected@write\@auxout{}{\string\providecommand\string\@istfilename[1]{}%
    Write the name of the style file to the aux file (needed by makeglossaries)
4736 \protected@write\@auxout{}{\string\@istfilename{\istfilename}}%
4737 \protected@write\@auxout{}{\string\@glsorder{\glsorder}}%
    Iterate through each glossary type and activate it.
4738 \@for\@glo@type:=\@glo@types\do{%
4739 \ifthenelse{\equal{\@glo@type}{}}{}{%
4740 \@makeglossary{\@glo@type}}%
4741 }%
    New glossaries must be created before \makeglossaries so disable \newglossary.
4742 \renewcommand*\newglossary[4][]{%
4743 \PackageError{glossaries}{New glossaries
4744 must be created before \string\makeglossaries}{You need
4745 to move \string\makeglossaries\space after all your
4746 \string\newglossary\space commands}}%
    Any subsequence instances of this command should have no effect
4747 \let\@makeglossary\relax
4748 \let\makeglossary\relax
4749 \let\makeglossaries\relax
    Disable all commands that have no effect after \makeglossaries
4750 \@disable@onlypremakeg
    Allow see key:
4751 \let\gls@checkseeallowed\relax
    Suppress warning about no \makeglossaries
4752 \let\warn@nomakeglossaries\relax
    Activate warning about missing \printglossary
4753 \def\warn@noprintglossary{%
4754 \GlossariesWarningNoLine{No \string\printglossary\space

```

```

4755     or \string\printglossaries\space
4756     found.^^J(Remove \string\makeglossaries\space if you don't want
4757     any glossaries.)^^JThis document will not have a glossary}%
4758 }%

```

Declare list parser for \glsdisplaynumberlist

```

4759 \ifglssavenumberlist
4760 \edef\@gls@dodolistparser{\noexpand\DeclareListParser
4761     {\noexpand\glsnumlistparser}{\delimN}}}%
4762 \@gls@dodolistparser
4763 \fi

```

Prevent user from also using \makenoidxglossaries

```

4764 \let\makenoidxglossaries\@no@makeglossaries

```

Prohibit sort key in printgloss family:

```

4765 \renewcommand*{\@printgloss@setsort}{%
4766     \let\@glo@assign@sortkey\@glo@no@assign@sortkey
4767 }%

```

Check the automake setting:

```

4768 \ifglsautomake
4769 \renewcommand*{\@gls@doautomake}{%
4770     \@for\@gls@type:=\@glo@types\do{%
4771         \ifdefempty{\@gls@type}{}%
4772         {\@gls@automake{\@gls@type}}%
4773     }%
4774 }%
4775 \fi
4776 }

```

Must occur in the preamble:

```

4777 \@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

```

4778 \let\makeglossary\makeglossaries

```

If \makeglossaries hasn't been used, issue a warning. Also issue a warning if neither \printglossaries nor \printglossary have been used.

```

4779 \AtEndDocument{%
4780     \warn@nomakeglossaries
4781     \warn@noprintglossary
4782 }

```

`\makenoidxglossaries` Analogous to `\makeglossaries` this activates the commands needed for `\printnoidxglossary`

```
4783 \newcommand*\makenoidxglossaries{%
```

Redefine empty glossary warning:

```
4784 \renewcommand{\@gls@noref@warn}[1]{%
4785   \GlossariesWarning{Empty glossary for
4786   \string\printnoidxglossary[type={##1}].
4787   Rerun may be required (or you may have forgotten to use
4788   commands like \string\gls).}%
4789 }
```

Don't escape makeindex/xindy characters

```
4790 \let\@gls@checkmkidxchars\@gobble
```

Write glossary information to aux instead of glossary files

```
4791 \let\@@do@cwrglossary\gls@noidxglossary
```

Switch on group headings that use the character code:

```
4792 \let\@gls@getgrouptitle\@gls@noidx@getgrouptitle
```

Allow see key:

```
4793 \let\gls@checkseeallowed\relax
```

Redefine cross-referencing macro:

```
4794 \renewcommand{\@do@seeglossary}[2]{%
4795   \edef\@gls@label{\glsdetoklabel{##1}}%
4796   \protected@write\@auxout{%
4797     \string\@gls@reference
4798     {\csname glo@\@gls@label @type\endcsname}%
4799     {\@gls@label}%
4800     {%
4801       \string\glsseeformat##2}%
4802     }%
4803   }%
4804 }
```

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.

```
4805 \AtBeginDocument
4806 {%
4807   \write\@auxout{\string\providecommand\string\@gls@reference[3]{}}%
4808 }
```

Change warning about no glossaries

```
4809 \def\warn@noprintglossary{%
4810   \GlossariesWarningNoLine{No \string\printnoidxglossary\space
4811   or \string\printnoidxglossaries ^^J
4812   found. (Remove \string\makenoidxglossaries\space if you
4813   don't want any glossaries.)^^JThis document will not have a glossary}%
4814 }
```

Suppress warning about no \makeglossaries

```

4815 \let\warn@nomakeglossaries\relax

Prevent user from also using \makeglossaries
4816 \let\makeglossaries\@no@makeglossaries

Allow sort key in printgloss family:
4817 \renewcommand*{\@printgloss@setsort}{%
4818 \let\@glo@assign@sortkey\@glo@assign@sortkey

Initialise default sort order:
4819 \def\@glo@sorttype{\@glo@default@sorttype}%
4820 }%

All entries must be defined in the preamble:
4821 \renewcommand*\new@glossaryentry[2]{%
4822 \PackageError{glossaries}{Glossary entries must be
4823 defined in the preamble^^Jwhen you use
4824 \string\makenoidxglossaries}%
4825 {Either move your definitions to the preamble or use
4826 \string\makeglossaries}%
4827 }%

Redefine \glsentrynumberlist
4828 \renewcommand*\glsentrynumberlist[1]{%
4829 \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
4830 \ifdef\@gls@loclist
4831 {%
4832 \glsnoidxloclist{\@gls@loclist}%
4833 }%
4834 {%
4835 ??\glsdoifexists{##1}%
4836 {%
4837 \GlossariesWarning{Missing location list for ‘##1’. Either
4838 a rerun is required or you haven’t referenced the entry.}%
4839 }%
4840 }%
4841 }%

Redefine \glsdisplaynumberlist
4842 \renewcommand*\glsdisplaynumberlist[1]{%
4843 \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
4844 \ifdef\@gls@loclist
4845 {%
4846 \def\@gls@noidxloclist@sep{%
4847 \def\@gls@noidxloclist@sep{%
4848 \def\@gls@noidxloclist@sep{%
4849 \glsnumlistsep
4850 }%
4851 \def\@gls@noidxloclist@finalsep{\glsnumlistlastsep}%
4852 }%
4853 }%

```



```

4854 \def\@gls@noidxloclist@finalsep{%
4855 \def\@gls@noidxloclist@prev{%
4856 \forlistloop{\glsnoidxdisplayloclisthandler}{\@gls@loclist}%
4857 \@gls@noidxloclist@finalsep
4858 \@gls@noidxloclist@prev
4859 }%
4860 {%
4861 ??\glsdoifexists{##1}%
4862 {%
4863 \GlossariesWarning{Missing location list for ‘##1’. Either
4864 a rerun is required or you haven’t referenced the entry.}%
4865 }%
4866 }%
4867 }%

```

Provide a generic way of iterating through the number list:

```

4868 \renewcommand*{\glsnumberlistloop}[3]{%
4869 \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
4870 \let\@gls@org@glsnoidxdisplayloc\glsnoidxdisplayloc
4871 \let\@gls@org@glsseeformat\glsseeformat
4872 \let\glsnoidxdisplayloc##2\relax
4873 \let\glsseeformat##3\relax
4874 \ifdef\@gls@loclist
4875 {%
4876 \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
4877 }%
4878 {%
4879 ??\glsdoifexists{##1}%
4880 {%
4881 \GlossariesWarning{Missing location list for ‘##1’. Either
4882 a rerun is required or you haven’t referenced the entry.}%
4883 }%
4884 }%
4885 \let\glsnoidxdisplayloc\@gls@org@glsnoidxdisplayloc
4886 \let\glsseeformat\@gls@org@glsseeformat
4887 }%

```

Modify sanitize sort function

```

4888 \let\@gls@sanitizesort\@gls@noidx@sanitizesort
4889 \let\@gls@nosanitizesort\@gls@noidx@nosanitizesort
4890 \@gls@noidx@setsanitizesort
4891 }

```

Preamble-only command:

```

4892 \@onlypreamble{\makenoidxglossaries}

```

`\glsnumberlistloop` `\glsnumberlistloop{<label>}{<handler>}`

```

4893 \newcommand*{\glsnumberlistloop}[2]{%

```

```

4894 \PackageError{glossaries}{\string\glsnumberlistloop\space
4895   only works with \string\makenoidxglossaries}{}%
4896 }

mberlistloophandler  Handler macro for \glsnumberlistloop. (The argument should be in the
                      form \glsnoidxdisplayloc{<prefix>}{<counter>}{<format>}{<n>})
4897 \newcommand*{\glsnoidxnumberlistloophandler}[1]{%
4898   #1%
4899 }

\@no@makeglossaries  Can't use both \makeglossaries and \makenoidxglossaries
4900 \newcommand*{\@no@makeglossaries}{%
4901   \PackageError{glossaries}{You can't use both
4902   \string\makeglossaries\space and \string\makenoidxglossaries}%
4903   {Either use one or other (or none) of those commands but not both
4904   together.}%
4905 }

\@gls@noref@warn  Warning when no instances of \@gls@reference found.
4906 \newcommand{\@gls@noref@warn}[1]{%
4907   \GlossariesWarning{\string\makenoidxglossaries\space
4908     is required to make \string\printnoidxglossary[type={#1}] work}%
4909 }

\gls@noidxglossary  Write the glossary information to the aux file:
4910 \newcommand*{\gls@noidxglossary}{%
4911   \protected@write\@auxout{}{%
4912     \string\@gls@reference
4913     {\csname glo@\@gls@label @type\endcsname}%
4914     {\@gls@label}%
4915     {\string\glsnoidxdisplayloc
4916       {\@glo@counterprefix}%
4917       {\@gls@counter}%
4918       {\@glsnumberformat}%
4919       {\@glslocref}%
4920     }%
4921   }%
4922 }

```

1.14 Writing information to associated files

\istfile Deprecated.

```
4923 \def\istfile{\glswrite}
```

At the end of the document, the files should be created if savewrites=true.

```

4924 \AtEndDocument{%
4925   \glswritefiles
4926 }

```

```

\@glswritefiles Only write the files if savewrites=true
4927 \newcommand*{\@glswritefiles}{%
  Iterate through all the glossaries
4928 \foralllglossaries{\@glo@type}{%
  Check for empty glossaries (patch provided by Patrick Häcker)
4929 \ifcsundef{glo@\@glo@type @filetok}%
4930 {%
4931 \def\gls@tmp{%
4932 }%
4933 {%
4934 \edef\gls@tmp{\expandafter\the
4935 \csname glo@\@glo@type @filetok\endcsname}%
4936 }%
4937 \ifx\gls@tmp\@empty
4938 \ifx\@glo@type\glsdefaulttype
4939 \GlossariesWarningNoLine{Glossary ‘\@glo@type’ has no
4940 entries.^^JRemember to use package option ‘nomain’ if
4941 you
4942 don’t want to^^Juse the main glossary}%
4943 \else
4944 \GlossariesWarningNoLine{Glossary ‘\@glo@type’ has no
4945 entries}%
4946 \fi
4947 \else
4948 \@glsopenfile{\glswrite}{\@glo@type}%
4949 \immediate\write\glswrite{%
4950 \expandafter\the
4951 \csname glo@\@glo@type @filetok\endcsname}%
4952 \immediate\closeout\glswrite
4953 \fi
4954 }%
4955 }

```

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
4956 \if@gls@docloaded
4957 \else

```

```

4958 \renewcommand*{\glossary}[1][main]{\gls@glossary{#1}}
4959 \fi

```

The associated number should be stored in `\theglsentrycounter` before using `\gls@glossary`.

`\gls@glossary`

```

4960 \newcommand*{\gls@glossary}[1]{%
4961   \@gls@glossary{#1}%
4962 }

```

`\@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.

```

4963 \newcommand*{\@gls@glossary}[1]{\index}

```

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.

`\@gls@renewglossary`

```

4964 \newcommand{\@gls@renewglossary}{%
4965   \gdef\@gls@glossary##1{\@bsphack\beginingroup\gls@wrglossary{##1}}%
4966   \let\@gls@renewglossary\@empty
4967 }

```

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`

```

4968 \newcommand*{\gls@wrglossary}[2]{%
4969   \ifglssavewrites
4970     \protected@edef\@gls@tmp{\the\csname glo@#1@filetok\endcsname#2}%
4971     \expandafter\global\expandafter\csname glo@#1@filetok\endcsname
4972       \expandafter{\@gls@tmp^^J}%
4973   \else
4974     \ifcsdef{glo@#1@file}%
4975     {%
4976       \expandafter\protected@write\csname glo@#1@file\endcsname{%
4977         \gls@disablepagerefexpansion}{#2}%
4978     }%
4979     {%
4980       \ifignoredglossary{#1}{}%
4981       {%
4982         \GlossariesWarning{No file defined for glossary ‘#1’}%
4983       }%

```

```

4984 }%
4985 \fi
4986 \endgroup\@esphack
4987 }

\@do@wrglossary
4988 \newcommand*\@do@wrglossary}[1]{%
4989 \glswriteentry{#1}{\@do@wrglossary{#1}}%
4990 }

\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.
4991 \newcommand*\glswriteentry}[2]{%
4992 \ifglindexonlyfirst
4993 \ifglused{#1}{#2}%
4994 \else
4995 #2%
4996 \fi
4997 }

@protected@pagefmts List of page formats to be protected against expansion.
4998 \newcommand*\gl@protected@pagefmts{%
4999 \gl@numberpage,\gl@alphpage,\gl@Alphpage,\gl@romanpage,\gl@Romanpage%
5000 }

blepagerefexpansion
5001 \newcommand*\gl@disablepagerefexpansion{%
5002 \@for\gl@this:=\gl@protected@pagefmts\do
5003 {%
5004 \expandafter\let\gl@this\relax
5005 }%
5006 }

\gl@alphpage
5007 \newcommand*\gl@alphpage{\@alph\c@page}

\gl@Alphpage
5008 \newcommand*\gl@Alphpage{\@Alph\c@page}

\gl@numberpage
5009 \newcommand*\gl@numberpage{\number\c@page}

\gl@romanpage
5010 \newcommand*\gl@romanpage{\romannumeral\c@page}

\gl@Romanpage
5011 \newcommand*\gl@Romanpage{\@Roman\c@page}

```

saddprotectedpagefmt `\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 T_EX register as the argument (`\<csname>\c@page` must be valid).

```

5012 \newcommand*{\glsaddprotectedpagefmt}[1]{%
5013   \eappto\gls@protected@pagefmts{\expandonce{\csname gls#1page\endcsname}}}%
5014   \csedef{gls#1page}{\expandonce{\csname#1\endcsname}\noexpand\c@page}%
5015   \eappto\@wrglossarynumberhook{%
5016     \noexpand\let\expandonce{\csname org@gls#1\endcsname}%
5017     \expandonce{\csname#1\endcsname}%
5018     \noexpand\def\expandonce{\csname#1\endcsname}{%
5019       \noexpand\@wrglossary@pageformat
5020       \expandonce{\csname gls#1page\endcsname}%
5021       \expandonce{\csname org@gls#1\endcsname}%
5022     }%
5023   }%
5024 }
```

rglossarynumberhook Hook used by `\@do@wrglossary`

```

5025 \newcommand*{\@wrglossarynumberhook{}
```

glossary@pageformat

```

5026 \newcommand{\@wrglossary@pageformat}[3]{%
5027   \ifx#3\c@page #1\else #2#3\fi
5028 }
```

`\@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.

```

5029 \newcommand*{\@do@wrglossary}[1]{%
5030   \begingroup
```

First a bit of hackery to prevent premature expansion of `\c@page`. Store original definitions:

```

5031   \let\orgthe\the
5032   \let\orgnumber\number
5033   \let\orgromannumeral\romannumeral
5034   \let\orgalph\@alph
5035   \let\orgAlph\@Alph
5036   \let\orgRoman\@Roman
```

Redefine:

```

5037   \def\the##1{%
5038     \ifx##1\c@page \gls@numberpage\else\orgthe##1\fi}%
5039   \def\number##1{%
5040     \ifx##1\c@page \gls@numberpage\else\orgnumber##1\fi}%
5041   \def\romannumeral##1{%
5042     \ifx##1\c@page \gls@romanpage\else\orgromannumeral##1\fi}%

```

```

5043 \def\@Roman##1{%
5044 \ifx##1\c@page \gls@Romanpage\else\orgRoman##1\fi}%
5045 \def\@alph##1{%
5046 \ifx##1\c@page \gls@alphpage\else\orgalph##1\fi}%
5047 \def\@Alph##1{%
5048 \ifx##1\c@page \gls@Alphpage\else\orgAlph##1\fi}%

```

Add hook to allow for other number formats:

```
5049 \@wrglossarynumberhook
```

Prevent expansion:

```
5050 \gls@disablepagerefexpansion
```

Now store location in \@glslocref:

```

5051 \protected@xdef\@glslocref{\theglsentrycounter}%
5052 \endgroup

```

Escape any special characters

```
5053 \@gls@checkmkidxchars\@glslocref
```

Check if the hyper-location is the same as the location and set the hyper prefix.

```

5054 \expandafter\ifx\theHglentrycounter\theglsentrycounter\relax
5055 \def\@glo@counterprefix{%
5056 \else
5057 \protected@edef\@glsHlocref{\theHglentrycounter}%
5058 \@gls@checkmkidxchars\@glsHlocref
5059 \edef\@do@gls@getcounterprefix{\noexpand\@gls@getcounterprefix
5060 {\@glslocref}\@glsHlocref}%
5061 }%
5062 \@do@gls@getcounterprefix
5063 \fi

```

De-tok label if required

```
5064 \edef\@gls@label{\glsdetoklabel{#1}}%
```

Write the information to file:

```

5065 \@do@wrglossary
5066 }

```

\@do@wrglossary

```
5067 \newcommand*\@do@wrglossary{%
```

Determine whether to use xindy or makeindex syntax

```
5068 \ifglxindy
```

Need to determine if the formatting information starts with a (or) indicating a range.

```

5069 \expandafter\@glo@check@mkidxrangechar\@glsnumberformat\@nil
5070 \def\@glo@range{%
5071 \expandafter\if\@glo@prefix(\relax
5072 \def\@glo@range{:open-range}%

```

```

5073 \else
5074 \expandafter\if\@glo@prefix)\relax
5075 \def\@glo@range{:close-range}%
5076 \fi
5077 \fi

```

Write to the glossary file using xindy syntax.

```

5078 \gls@glossary{\csname glo@\@gls@label @type\endcsname}{%
5079 (indexentry :tkey (\csname glo@\@gls@label @index\endcsname)

5080 :locref \string"\@glo@counterprefix}{\@gls@locref}\string" %
5081 :attr \string"\@gls@counter\@glo@suffix\string"
5082 \@glo@range
5083 )
5084 }%
5085 \else

```

Convert the format information into the format required for makeindex

```

5086 \@set@glo@numformat{\@glo@numfmt}{\@gls@counter}{\@gls@numberformat}%
5087 {\@glo@counterprefix}%

```

Write to the glossary file using makeindex syntax.

```

5088 \gls@glossary{\csname glo@\@gls@label @type\endcsname}{%
5089 \string\glossaryentry{\csname glo@\@gls@label @index\endcsname
5090 \@gls@encapchar\@glo@numfmt}{\@gls@locref}}%
5091 \fi
5092 }

```

`\@getcounterprefix` 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.)

```

5093 \newcommand*\@gls@getcounterprefix[2]{%
5094 \edef\@gls@thisloc{#1}\edef\@gls@thisHloc{#2}%
5095 \ifx\@gls@thisloc\@gls@thisHloc
5096 \def\@glo@counterprefix{%
5097 \else
5098 \def\@gls@get@counterprefix##1.#1##2\end@getprefix{%
5099 \def\@glo@tmp{##2}%
5100 \ifx\@glo@tmp\@empty
5101 \def\@glo@counterprefix{%
5102 \else
5103 \def\@glo@counterprefix{##1}%
5104 \fi
5105 }%
5106 \@gls@get@counterprefix#2.#1\end@getprefix

```

Warn if no prefix can be formed.

```

5107 \ifx\@glo@counterprefix\@empty

```



```

5108     \GlossariesWarning{Hyper target ‘#2’ can’t be formed by
5109     prefixing~Jlocation ‘#1’. You need to modify the
5110     definition of \string\theH\@gls@counter~Jotherwise you
5111     will get the warning: “‘name{\@gls@counter.#1}’ has been~J
5112     referenced but does not exist”}%
5113   \fi
5114 \fi
5115 }

```

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.

```

5116 \newcommand{\do@seeglossary}[2]{%
5117 \def\@gls@xref{#2}%
5118 \@onelevel@sanitize\@gls@xref
5119 \@gls@checkmkidxchars\@gls@xref
5120 \ifglxsindy
5121   \gls@glossary{\csname glo@#1@type\endcsname}{%
5122     (indexentry
5123       :tkey (\csname glo@#1@index\endcsname)
5124       :xref (\string"\@gls@xref\string")
5125       :attr \string"see\string"
5126     )
5127   }%
5128 \else
5129   \gls@glossary{\csname glo@#1@type\endcsname}{%
5130     \string\glossaryentry{\csname glo@#1@index\endcsname
5131     \@gls@encapchar glsseeformat\@gls@xref}{Z}}%
5132 \fi
5133 }

```

`\@gls@fixbraces` If no optional argument is specified, list needs to be enclosed in a set of braces.

```

5134 \def\@gls@fixbraces#1#2#3\@nil{%
5135   \ifx#2[\relax
5136     \@gls@fixbraces#1#2#3\@end@fixbraces
5137   \else
5138     \def#1{{#2#3}}%
5139   \fi
5140 }

```

`\@@gls@fixbraces`

```

5141 \def\@@gls@fixbraces#1[#2]#3\@end@fixbraces{%
5142   \def#1{[#2]{#3}}%
5143 }

```

`\glssee` `\glssee{\langle label \rangle}{\langle cross-ref list \rangle}`

```

5144 \DeclareRobustCommand*\glssee}[3][\seename]{%
5145   \@do@seeglossary{#2}{[#1]{#3}}}%
5146 \newcommand*\@glssee}[3][\seename]{%
5147   \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.

```

5148 \DeclareRobustCommand*\glsseeformat}[3][\seename]{%
5149   \emph{#1} \glsseelist{#2}}

```

`\glsseelist` `\glsseelist{<list>}` formats list of entry labels.

```

5150 \DeclareRobustCommand*\glsseelist}[1]{%

```

If there is only one item in the list, set the last separator to do nothing.

```

5151   \let\@gls@dolast\relax

```

Don’t display separator on the first iteration of the loop

```

5152   \let\@gls@donext\relax

```

Iterate through the labels

```

5153   \@for\@gls@thislabel:=#1\do{%

```

Check if on last iteration of loop

```

5154     \ifx\@xfor@nextelement\@nnil

```

```

5155       \@gls@dolast

```

```

5156     \else

```

```

5157       \@gls@donext

```

```

5158     \fi

```

Display the entry for this label. (Expanding label as it’s a temporary control sequence that’s used elsewhere.)

```

5159     \expandafter\glsseeitem\expandafter{\@gls@thislabel}%

```

Update separators

```

5160     \let\@gls@dolast\glsseelastsep

```

```

5161     \let\@gls@donext\glsseesep

```

```

5162   }%

```

```

5163 }

```

`\glsseelastsep` Separator to use between penultimate and ultimate entries in a cross-referencing list.

```

5164 \newcommand*\glsseelastsep{\space\andname\space}

```

`\glsseesep` Separator to use between entires in a cross-referencing list.

```

5165 \newcommand*\glsseesep{, }

```

`\glsseeitem` `\glsseeitem{<label>}` formats individual entry in a cross-referencing list.

```

5166 \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.)

```

5167 \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.

`gls@save@numberlist` Provide command to store number list.

```
5168 \newcommand*{\gls@save@numberlist}[1]{%
5169   \ifglssavenumberlist
5170     \toks@{#1}%
5171     \edef\@do@writeaux@info{%
5172       \noexpand\csgdef{glo@\glscurrententrylabel @numberlist}{\the\toks@}%
5173     }%
5174     \@onelevel@sanitize\@do@writeaux@info
5175     \protected@write\@auxout{}\@do@writeaux@info}%
5176   \fi
5177 }
```

`warn@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.)

```
5178 \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.

```
5179 \ifcsundef{printglossary}{}%
5180 {%
```

If `\printglossary` is already defined, issue a warning and undefine it.

```
5181   \@gls@warnonglossdefined
5182   \undef\printglossary
5183 }
```

`\printglossary` has an optional argument. The default value is to set the glossary type to the main glossary.

```
5184 \newcommand*{\printglossary}[1][type=\glsdefaulttype]{%
5185   \@printglossary{#1}{\@print@glossary}%
5186 }
```

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`

```
5187 \newcommand*{\printglossaries}{%
5188   \forallglossaries{\@glo@type}{\printglossary[type=\@glo@type]}%
5189 }
```

`\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.

```
5190 \newcommand*{\printnoidxglossary}[1][type=\glsdefaulttype]{%
5191   \@printglossary{#1}{\@printnoidxglossary}%
5192 }
```

`\printnoidxglossaries` Analogous to `\printglossaries`

```
5193 \newcommand*{\printnoidxglossaries}{%
5194   \forallglossaries{\@glo@type}{\printnoidxglossary[type=\@glo@type]}%
5195 }
```

`@printgloss@setsort` Initialise to do nothing.

```
5196 \newcommand*{\@printgloss@setsort}{}%
```

`gls@preglossaryhook`

```
5197 \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.

```
5198 \newcommand{\@printglossary}[2]{%
```

Set up defaults.

```
5199   \def\@glo@type{\glsdefaulttype}%
5200   \def\glossarytitle{\csname @glo@type\@glo@type @title\endcsname}%

5201   \def\glossarytoctitle{\glossarytitle}%
5202   \let\org@glossarytitle\glossarytitle
5203   \def\@glossarystyle{}%
5204   \def\gls@dotoc@title{\glssettoc@title{\@glo@type}}%
```

Store current value of `\glossaryentrynumbers`. (This may be changed via the optional argument)

```
5205   \let\@org@glossaryentrynumbers\glossaryentrynumbers
```

Localise the effects of the optional argument

```
5206   \bgroup
```

Activate or deactivate sort key:

```
5207   \@printgloss@setsort
```

Determine settings specified in the optional argument.

```

5208   \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)
5209   \ifx\glossarytitle\org@glossarytitle
5210   \else
5211     \expandafter\let\csname @glo@type @title\endcsname
5212         \glossarytitle
5213   \fi

Allow a high-level user command to indicate the current glossary
5214   \let\currentglossary\@glo@type

Enable individual number lists to be suppressed.
5215   \let\org@glossaryentrynumbers\glossaryentrynumbers
5216   \let\glsnonextpages\@glsnonextpages

Enable individual number list to be activated:
5217   \let\glsnextpages\@glsnextpages

Enable suppression of description terminators.
5218   \let\nopostdesc\@nopostdesc

Set up the entry for the TOC
5219   \gls@dotoctitle

Set the glossary style
5220   \@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):

5221   \let\gls@org@glossaryentryfield\glossentry
5222   \let\gls@org@glossarysubentryfield\subglossentry
5223   \renewcommand{\glossentry}[1]{%
5224     \xdef\glscurrententrylabel{\glsdetoklabel{##1}}%
5225     \gls@org@glossaryentryfield{##1}%
5226   }%
5227   \renewcommand{\subglossentry}[2]{%
5228     \xdef\glscurrententrylabel{\glsdetoklabel{##2}}%
5229     \gls@org@glossarysubentryfield{##1}{##2}%
5230   }%

5231   \@gls@preglossaryhook

Now do the handler macro that deals with the actual glossary:
5232   #2%

End the current scope
5233   \egroup

Reset \glossaryentrynumbers
5234   \global\let\glossaryentrynumbers\@org@glossaryentrynumbers

```

Suppress warning about no \printglossary

```
5235 \global\let\warn@noprintglossary\relax
5236 }
```

\@print@glossary Internal workings of \printglossary dealing with reading the external file.

```
5237 \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.)

```
5238 \makeatletter
```

Input the glossary file, if it exists.

```
5239 \@input@{\jobname.\csname @glotype@\@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.

```
5240 \IfFileExists{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
5241 {}%
5242 {\null}%
```

If xindy is being used, need to write the language dependent information to the .aux file for makeglossaries.

```
5243 \ifglxindy
5244 \ifcsundef{@xdy@\@glo@type @language}%
5245 {%
5246 \edef\@do@auxoutstuff{%
5247 \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.

```
5248 \noexpand\immediate\noexpand\write\@auxout{%
5249 \string\providecommand\string\@xdylanguage[2]{}%
5250 \noexpand\immediate\noexpand\write\@auxout{%
5251 \string\@xdylanguage{\@glo@type}{\@xdy@main@language}}%
5252 }%
5253 }%
5254 }%
5255 {%
5256 \edef\@do@auxoutstuff{%
5257 \noexpand\AtEndDocument{%
5258 \noexpand\immediate\noexpand\write\@auxout{%
5259 \string\providecommand\string\@xdylanguage[2]{}%
5260 \noexpand\immediate\noexpand\write\@auxout{%
5261 \string\@xdylanguage{\@glo@type}{\csname @xdy@\@glo@type
5262 @language\endcsname}}%
5263 }%
5264 }%
```

```

5265 }%
5266 \@do@auxoutstuff
5267 \edef\@do@auxoutstuff{%
5268     \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.

```

5269     \noexpand\immediate\noexpand\write\@auxout{%
5270         \string\providecommand\string\@gls@codepage[2]{}%
5271     \noexpand\immediate\noexpand\write\@auxout{%
5272         \string\@gls@codepage{\@glo@type}{\gls@codepage}%
5273     }%
5274 }%
5275 \@do@auxoutstuff
5276 \fi

```

Activate warning if \makeglossaries hasn't been used.

```

5277 \renewcommand*\@warn@nomakeglossaries{%
5278     \GlossariesWarningNoLine{\string\makeglossaries\space
5279     hasn't been used,~^Jthe glossaries will not be updated}%
5280 }%
5281 }

```

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

```

5282 \newcommand*\@glo@sortentries[2]{%
5283     \def\@glo@sortinglist{}%
5284     \def\@glo@sortinghandler{#1}%
5285     \edef\@glo@type{#2}%
5286     \forlistcsloop{\@glo@do@sortentries}{\glsref@#2}%
5287     \csdef{\glsref@#2}{}%
5288     \@for\@this@label:=\@glo@sortinglist\do{%

```

Has this entry already been added?

```

5289     \xifinlistcs{\@this@label}{\glsref@#2}%
5290     {}%
5291     {%
5292         \listcsxadd{\glsref@#2}{\@this@label}%
5293     }%
5294     \ifcsdef{\@glo@sortingchildren@\@this@label}%
5295     {%
5296         \@glo@addchildren{#2}{\@this@label}%

```

```

5297    }%
5298    {}%
5299    }%
5300 }

```

```
\@glo@addchildren \@glo@addchildren{<type>}{<parent>}
```

```
5301 \newcommand*{\@glo@addchildren}[2]{%
```

Scope to allow nesting.

```

5302   \bgroup
5303     \letcs{\@glo@childlist}{@glo@sortingchildren@#2}%
5304     \@for\@this@childlabel:=\@glo@childlist\do
5305     {%

```

Check this label hasn't already been added.

```

5306       \xifinlistcs{\@this@childlabel}{@glsref@#1}%
5307       {}%
5308       {%
5309         \listcsxadd{@glsref@#1}{\@this@childlabel}%
5310       }%

```

Does this child have children?

```

5311       \ifcsdef{@glo@sortingchildren@\@this@childlabel}%
5312       {%
5313         \@glo@addchildren{#1}{\@this@childlabel}%
5314       }%
5315       {%
5316       }%
5317     }%
5318   \egroup
5319 }

```

@glo@do@sortentries

```

5320 \newcommand*{\@glo@do@sortentries}[1]{%
5321   \ifglshasparent{#1}%
5322   {%

```

This entry has a parent, so add it to the child list

```

5323     \edef\@glo@parent{\csuse{glo@glsetoklabel{#1}@parent}}%
5324     \ifcsundef{@glo@sortingchildren@\@glo@parent}%
5325     {%
5326       \csdef{@glo@sortingchildren@\@glo@parent}{}%
5327     }%
5328     {}%
5329     \expandafter\@glo@sortedinsert
5330     \csname @glo@sortingchildren@\@glo@parent\endcsname{#1}%

```

Has the parent been added?

```
5331   \xifinlistcs{\@glo@parent}{@glsref@\@glo@type}%

```



```

5332    {%
      Yes, it has so do nothing.
5333    }%
5334    {%
      No, it hasn't so add it now.
5335      \expandafter\@glo@do@sortentries\expandafter{\@glo@parent}%
5336    }%
5337  }%
5338  {%
5339    \@glo@sortedinsert{\@glo@sortinglist}{#1}%
5340  }%
5341 }

```

\@glo@sortedinsert \@glo@sortedinsert{<list>}{<entry label>}

Insert into list.

```

5342 \newcommand*{\@glo@sortedinsert}[2]{%
5343   \dtl@insertinto{#2}{#1}{\@glo@sortinghandler}%
5344 }%

```

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).

\@glo@sorthandler@word

```

5345 \newcommand*{\@glo@sorthandler@word}[2]{%
5346   \letcs\@gls@sort@A{glo@glstetoklabel{#1}@sort}%
5347   \letcs\@gls@sort@B{glo@glstetoklabel{#2}@sort}%
5348   \edef\glo@do@compare{%
5349     \noexpand\dtlwordindexcompare{\noexpand\dtl@sortresult}%
5350     {\expandonce\@gls@sort@B}%
5351     {\expandonce\@gls@sort@A}%
5352   }%
5353   \glo@do@compare
5354 }

```

\@glo@sorthandler@letter

```

5355 \newcommand*{\@glo@sorthandler@letter}[2]{%
5356   \letcs\@gls@sort@A{glo@glstetoklabel{#1}@sort}%
5357   \letcs\@gls@sort@B{glo@glstetoklabel{#2}@sort}%
5358   \edef\glo@do@compare{%
5359     \noexpand\dtlletterindexcompare{\noexpand\dtl@sortresult}%
5360     {\expandonce\@gls@sort@B}%
5361     {\expandonce\@gls@sort@A}%
5362   }%
5363   \glo@do@compare
5364 }

```

lo@sorthandler@case Case-sensitive sort.

```
5365 \newcommand*{\@glo@sorthandler@case}[2]{%
5366   \letcs\@gls@sort@A{glo\glsdetoklabel{#1}@sort}%
5367   \letcs\@gls@sort@B{glo\glsdetoklabel{#2}@sort}%
5368   \edef\glo@do@compare{%
5369     \noexpand\dtlcompare{\noexpand\dtl@sortresult}%
5370     {\expandonce\@gls@sort@B}%
5371     {\expandonce\@gls@sort@A}%
5372   }%
5373   \glo@do@compare
5374 }
```

@sorthandler@nocase Case-insensitive sort.

```
5375 \newcommand*{\@glo@sorthandler@nocase}[2]{%
5376   \letcs\@gls@sort@A{glo\glsdetoklabel{#1}@sort}%
5377   \letcs\@gls@sort@B{glo\glsdetoklabel{#2}@sort}%
5378   \edef\glo@do@compare{%
5379     \noexpand\dtlcompare{\noexpand\dtl@sortresult}%
5380     {\expandonce\@gls@sort@B}%
5381     {\expandonce\@gls@sort@A}%
5382   }%
5383   \glo@do@compare
5384 }
```

@glo@sortmacro@word Sort macro for ‘word’

```
5385 \newcommand*{\@glo@sortmacro@word}[1]{%
5386   \ifdefstring{\@glo@default@sorttype}{standard}%
5387   {%
5388     \@glo@sortentries{\@glo@sorthandler@word}{#1}%
5389   }%
5390   {%
5391     \PackageError{glossaries}{Conflicting sort options:^^J
5392       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5393       \string\printnoidxglossary[sort=word]}{}%
5394   }%
5395 }
```

lo@sortmacro@letter Sort macro for ‘letter’

```
5396 \newcommand*{\@glo@sortmacro@letter}[1]{%
5397   \ifdefstring{\@glo@default@sorttype}{standard}%
5398   {%
5399     \@glo@sortentries{\@glo@sorthandler@letter}{#1}%
5400   }%
5401   {%
5402     \PackageError{glossaries}{Conflicting sort options:^^J
5403       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5404       \string\printnoidxglossary[sort=letter]}{}%
5405   }%
5406 }
```

@sortmacro@standard Sort macro for ‘standard’. (Use either ‘word’ or ‘letter’ order.)

```
5407 \newcommand*{\@glo@sortmacro@standard}[1]{%
5408   \ifdefstring{\@glo@default@sorttype}{standard}%
5409   {%
5410     \ifcsdef{\@glo@sorthandler@\glsorder}%
5411     {%
5412       \@glo@sortentries{\csuse{\@glo@sorthandler@\glsorder}}{#1}%
5413     }%
5414     {%
5415       \PackageError{glossaries}{Unknown sort handler ‘\glsorder’}{}%
5416     }%
5417   }%
5418   {%
5419     \PackageError{glossaries}{Conflicting sort options:^^J
5420       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5421       \string\printnoidxglossary[sort=standard]}{}%
5422   }%
5423 }
```

@glo@sortmacro@case Sort macro for ‘case’

```
5424 \newcommand*{\@glo@sortmacro@case}[1]{%
5425   \ifdefstring{\@glo@default@sorttype}{standard}%
5426   {%
5427     \@glo@sortentries{\@glo@sorthandler@case}{#1}%
5428   }%
5429   {%
5430     \PackageError{glossaries}{Conflicting sort options:^^J
5431       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5432       \string\printnoidxglossary[sort=case]}{}%
5433   }%
5434 }
```

lo@sortmacro@nocase Sort macro for ‘nocase’

```
5435 \newcommand*{\@glo@sortmacro@nocase}[1]{%
5436   \ifdefstring{\@glo@default@sorttype}{standard}%
5437   {%
5438     \@glo@sortentries{\@glo@sorthandler@nocase}{#1}%
5439   }%
5440   {%
5441     \PackageError{glossaries}{Conflicting sort options:^^J
5442       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5443       \string\printnoidxglossary[sort=nocase]}{}%
5444   }%
5445 }
```

\@glo@sortmacro@def Sort macro for ‘def’. The order of definition is given in \glo@list@<type>.

```
5446 \newcommand*{\@glo@sortmacro@def}[1]{%
5447   \def\@glo@sortinglist{}%
5448   \for\glsentries[#1]{\@gls@thislabel}%

```

```

5449  {%
5450    \xifinlistcs{\@gls@thislabel}{\@glsref@#1}%
5451    {%
5452      \listcsadd{\@glo@sortinglist}{\@gls@thislabel}%
5453    }%
5454    {%
      Hasn't been referenced.
5455    }%
5456  }%
5457  \cslet{\@glsref@#1}{\@glo@sortinglist}%
5458 }

\lo@sortmacro@def@do This won't include parent entries that haven't been referenced.
5459 \newcommand*{\@glo@sortmacro@def@do}[1]{%
5460   \ifinlistcs{#1}{\@glsref@\@glo@type}%
5461   {}%
5462   {%
5463     \listcsadd{\@glsref@\@glo@type}{#1}%
5464   }%
5465   \ifcsdef{\@glo@sortingchildren@#1}%
5466   {%
5467     \@glo@addchildren{\@glo@type}{#1}%
5468   }%
5469   {}%
5470 }

\@glo@sortmacro@use Sort macro for 'use'. (No sorting is required, as the entries are already in order
of use, so do nothing.)
5471 \newcommand*{\@glo@sortmacro@use}[1]{%

\print@noidx@glossary Glossary handler for \printnoidxglossary which doesn't use an indexing ap-
plication. Since \printnoidxglossary may occur at the start of the docu-
ment, 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 appro-
priate glossary.
5472 \newcommand*{\@print@noidx@glossary}{%
5473   \ifcsdef{\@glsref@\@glo@type}%
5474   {%
      Sort the entries:
5475     \ifcsdef{\@glo@sortmacro@\@glo@sorttype}%
5476     {%
5477       \csuse{\@glo@sortmacro@\@glo@sorttype}{\@glo@type}%
5478     }%
5479     {%
5480       \PackageError{glossaries}{Unknown sort handler '\@glo@sorttype'}{}%
5481     }%

```

Do the glossary heading and preamble

```
5482 \glossarysection[\glossarytoctitle]{\glossarytitle}%
5483 \glossarypreamble
5484 \begin{theglossary}%
5485 \glossaryheader
5486 \glsresetentrylist
5487 \def\@gls@currentlettergroup{}
```

Iterate through the entries.

```
5488 \forlistcsloop{\@gls@noidx@do}{\@glsref@{\@glo@type}%
```

Finally end the glossary and do the postamble:

```
5489 \end{theglossary}%
5490 \glossarypostamble
5491 }%
5492 {%
5493 \@gls@noref@warn{\@glo@type}%
5494 }%
5495 }
```

\glo@grabfirst

```
5496 \def\glo@grabfirst#1#2\@nil{%
5497 \def\@gls@firsttok{#1}%
5498 \ifdefempty\@gls@firsttok
5499 {%
5500 \def\@glo@thislettergrp{0}%
5501 }%
5502 {%
```

Sanitize it:

```
5503 \@onelevel@sanitize\@gls@firsttok
```

Fetch the first letter:

```
5504 \expandafter\@glo@grabfirst\@gls@firsttok{}\}\@nil
5505 }%
5506 }
```

\@glo@grabfirst

```
5507 \def\@glo@grabfirst#1#2\@nil{%
5508 \ifdefempty\@glo@thislettergrp
5509 {%
5510 \def\@glo@thislettergrp{glssymbols}%
5511 }%
5512 {%
5513 \count@=\uccode'#1\relax
5514 \ifnum\count@=0\relax
5515 \def\@glo@thislettergrp{glssymbols}%
5516 \else
5517 \ifdefstring\@glo@sorttype{case}%
5518 {%
```

```

5519         \count@=#1\relax
5520     }%
5521     {%
5522     }%
5523     \edef\@glo@thislettergrp{\the\count@}%
5524     \fi
5525 }%
5526 }

```

\@gls@noidx@do Handler for list iteration used by \@print@noidx@glossary. The argument is the entry label. This only allows one sublevel.

```

5527 \newcommand{\@gls@noidx@do}[1]{%

```

 Get this entry's location list

```

5528   \global\letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%

```

 Does this entry have a parent?

```

5529   \ifglshasparent{#1}%
5530   {%

```

 Has a parent.

```

5531       \gls@level=\csuse{glo@\glsdetoklabel{#1}@level}\relax
5532       \ifdefvoid{\@gls@loclist}
5533       {%
5534         \subglossentry{\gls@level}{#1}{}%
5535       }%
5536       {%
5537         \subglossentry{\gls@level}{#1}%
5538         {%
5539           \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
5540         }%
5541       }%
5542     }%
5543     {%

```

 Doesn't have a parent Get this entry's sort key

```

5544       \letcs{\@gls@sort}{glo@\glsdetoklabel{#1}@sort}%

```

 Fetch the first letter:

```

5545       \expandafter\glo@grabfirst\@gls@sort{}{}\@nil
5546       \ifdefequal{\@glo@thislettergrp}{\@gls@currentlettergroup}%
5547       {%
5548       {%

```

 Do the group header:

```

5549         \ifdefempty{\@gls@currentlettergroup}{\@gls@groupskip}%
5550         \gls@groupheading{\@glo@thislettergrp}%
5551       }%
5552       \let\@gls@currentlettergroup\@glo@thislettergrp

```

 Do this entry:

```

5553       \ifdefvoid{\@gls@loclist}

```

```

5554    {%
5555      \glossentry{#1}{}%
5556    }%
5557    {%
5558      \glossentry{#1}%
5559      {%
5560        \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
5561      }%
5562    }%
5563  }%
5564 }

```

`\glsnoidxloclist` `\glsnoidxloclist{<list cs>}`

Display location list.

```

5565 \newcommand*{\glsnoidxloclist}[1]{%
5566   \def\@gls@noidxloclist@sep{}%
5567   \def\@gls@noidxloclist@prev{}%
5568   \forlistloop{\glsnoidxloclisthandler}{#1}%
5569 }

```

`\glsnoidxloclisthandler` Handler for location list iterator.

```

5570 \newcommand*{\glsnoidxloclisthandler}[1]{%
5571   \ifdefstring{\@gls@noidxloclist@prev}{#1}%
5572   {%
5573     }%
5574   {%
5575     \@gls@noidxloclist@sep
5576     #1%
5577     \def\@gls@noidxloclist@sep{\delimN}%
5578     \def\@gls@noidxloclist@prev{#1}%
5579   }%
5580 }

```

`\glsdisplayloclisthandler` Handler for location list iterator when used with `\glsdisplaynumberlist`.

```

5581 \newcommand*{\glsnoidxdisplayloclisthandler}[1]{%
5582   \ifdefstring{\@gls@noidxloclist@prev}{#1}%
5583   {%
5584     }%
5585   {%
5586     \@gls@noidxloclist@sep
5587     \@gls@noidxloclist@prev
5588     \def\@gls@noidxloclist@prev{#1}%
5589   }%
5590 }

```

`\glsnoidxdisplayloc` `\glsnoidxdisplayloc{<prefix>}{<counter>}{<format>}{<location>}`

Display a location in the location list.

```
5591 \newcommand*\glsnoidxdisplayloc[4]{%
5592   \setentrycounter[#1]{#2}%
5593   \csuse{#3}{#4}%
5594 }
```

`\@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.

```
5595 \newcommand*\@gls@reference}[3]{%
```

Add to label list

```
5596   \glsdoifexistsorwarn{#2}%
5597   {%
5598     \ifcsundef{@glsref@#1}{\csgdef{@glsref@#1}{}}{}%
5599     \ifinlistcs{#2}{@glsref@#1}%
5600     {}%
5601     {\listcsgadd{@glsref@#1}{#2}}%
```

Add to location list

```
5602     \ifcsundef{glo@\glsdetoklabel{#2}@loclist}%
5603     {\csgdef{glo@\glsdetoklabel{#2}@loclist}{}}%
5604     {}%
5605     \listcsgadd{glo@\glsdetoklabel{#2}@loclist}{#3}%
5606   }%
5607 }
```

The keys that can be used in the optional argument to `\printglossary` or `\printnoidxglossary` are as follows: The `type` key sets the glossary type.

```
5608 \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`.

```
5609 \define@key{printgloss}{title}{%
5610   \def\glossarytitle{#1}%
5611   \let\gls@dotoc@title\relax
5612 }
```

The `toctitle` sets the text used for the relevant entry in the table of contents.

```
5613 \define@key{printgloss}{toctitle}{%
5614   \def\glossarytoctitle{#1}%
5615   \let\gls@dotoc@title\relax
5616 }
```


The style key sets the glossary style (but only for the given glossary).

```

5617 \define@key{printgloss}{style}{%
5618   \ifcsundef{@glsstyle@#1}%
5619   {%
5620     \PackageError{glossaries}%
5621     {Glossary style ‘#1’ undefined}{}%
5622   }%
5623   {%
5624     \def\@glossarystyle{\setglossentrycompatibility
5625       \csname @glsstyle@#1\endcsname}%
5626   }%
5627 }
```

The numberedsection key determines if this glossary should be in a numbered section.

```

5628 \define@choicekey{printgloss}{numberedsection}[\val\nr]{%
5629 false,nolabel,autolabel,nameref}[nolabel]{%
5630   \ifcase\nr\relax
5631     \renewcommand*{\@glossarysecstar}{*}%
5632     \renewcommand*{\@glossaryseclabel}{}%
5633   \or
5634     \renewcommand*{\@glossarysecstar}{}%
5635     \renewcommand*{\@glossaryseclabel}{}%
5636   \or
5637     \renewcommand*{\@glossarysecstar}{}%
5638     \renewcommand*{\@glossaryseclabel}{\label{\glsautoprefix\@glo@type}}%
5639   \or
5640     \renewcommand*{\@glossarysecstar}{*}%
5641     \renewcommand*{\@glossaryseclabel}{%
5642       \protected@edef\@currentlabelname{\glossarytoctitle}%
5643       \label{\glsautoprefix\@glo@type}}%
5644   \fi
5645 }
```

The nogroupskip key determines whether or not there should be a vertical gap between glossary groups.

```

5646 \define@choicekey{printgloss}{nogroupskip}{true,false}[true]{%
5647   \csuse{glsnogroupskip#1}%
5648 }
```

The nopostdot key has the same effect as the package option of the same name.

```

5649 \define@choicekey{printgloss}{nopostdot}{true,false}[true]{%
5650   \csuse{glsnopostdot#1}%
5651 }
```

The entrycounter key is the same as the package option but localised to the current glossary.

```

5652 \define@choicekey{printgloss}{entrycounter}{true,false}[true]{%
5653   \csuse{glsentrycounter#1}%
5654 }
```

```

5654 \ifglentrycounter
5655   \ifx\@gls@counterwithin\@empty
5656     \newcounter{glossaryentry}%
5657   \else
5658     \newcounter{glossaryentry}[\@gls@counterwithin]%
5659   \fi
5660   \def\theHglossaryentry{\currentglossary.\theglossaryentry}%
5661   \renewcommand*{\glsresetentrycounter}{%
5662     \setcounter{glossaryentry}{0}%
5663   }%
5664   \renewcommand*{\glsstepentry}[1]{%
5665     \refstepcounter{glossaryentry}%
5666     \label{glsentry-\glsdetoklabel{##1}}%
5667   }%
5668   \renewcommand*{\glentrycounterlabel}{\theglossaryentry.\space}%
5669   \renewcommand*{\glentryitem}[1]{%
5670     \glsstepentry{##1}\glentrycounterlabel
5671   }%
5672 \else
5673   \renewcommand*{\glsresetentrycounter}{}%
5674   \renewcommand*{\glsstepentry}[1]{}%
5675   \renewcommand*{\glentrycounterlabel}{}%
5676   \renewcommand*{\glentryitem}[1]{\glsresetsubentrycounter}
5677 \fi
5678 }

```

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.

```

5679 \define@choicekey{printgloss}{subentrycounter}{true,false}[true]{%
5680   \csuse{glssubentrycounter#1}%
5681   \ifglssubentrycounter
5682     \ifundef\c@glossarysubentry
5683     {%
5684       \ifglentrycounter
5685         \newcounter{glossarysubentry}[glossaryentry]%
5686       \else
5687         \newcounter{glossarysubentry}
5688       \fi
5689     }{}%
5690   \renewcommand*{\glsstepsubentry}[1]{%
5691     \edef\currentglssubentry{\glsdetoklabel{##1}}%
5692     \refstepcounter{glossarysubentry}%
5693     \label{glsentry-\currentglssubentry}%
5694   }%
5695   \renewcommand*{\glsresetsubentrycounter}{%
5696     \setcounter{glossarysubentry}{0}%
5697   }%

```

```

5698 \renewcommand*{\glssubentryitem}[1]{%
5699 \glssubentry{##1}\glssubentrycounterlabel
5700 }%
5701 \renewcommand*{\glssubentrycounterlabel}{\theglossarysubentry}\space}%
5702 \def\theHglossarysubentry{\currentglssubentry.\theglossarysubentry}
5703 \else
5704 \renewcommand*{\glssubentryitem}[1]{}%
5705 \renewcommand*{\glssubentry}[1]{}%
5706 \renewcommand*{\glssubentrycounter}{}%
5707 \renewcommand*{\glssubentrycounterlabel}{}%
5708 \fi
5709 }

```

The nonnumberlist key determines if this glossary should have a number list.

```

5710 \define@boolkey{printgloss}[gls]{nonnumberlist}[true]{%
5711 \ifglslnonnumberlist
5712 \def\glossaryentrynumbers##1{}%
5713 \else
5714 \def\glossaryentrynumbers##1{##1}%
5715 \fi}

```

The sort key sets the glossary sort handler (\printnoidxglossary only).

```

5716 \define@key{printgloss}{sort}{\@glo@assign@sortkey{#1}}

```

`\no@assign@sortkey` Issue error if used with \printglossary

```

5717 \newcommand*{\@glo@no@assign@sortkey}[1]{%
5718 \PackageError{glossaries}{‘sort’ key not permitted with
5719 \string\printglossary}%
5720 {The ‘sort’ key may only be used with \string\printnoidxglossary}%
5721 }

```

`\glo@assign@sortkey` For use with \printnoidxglossary

```

5722 \newcommand*{\@glo@assign@sortkey}[1]{%
5723 \def\@glo@sorttype{#1}%
5724 }

```

`\@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 redefined.

```

5725 \newcommand*{\@glslnonextpages}{%
5726 \gdef\glossaryentrynumbers##1{%
5727 \glslresetentrylist
5728 }%
5729 }

```

`\@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

`\glsnextpages` is place in the entry's description and 3 column tabular style glossary is used.) `\org@glossaryentrynumbers` needs to be set at the start of each glossary, in the event that `\glossaryentrynumber` is redefined.

```
5730 \newcommand*{\@glsnextpages}{%
5731   \gdef\glossaryentrynumbers##1{%
5732     ##1\glsresetentrylist}}
```

`\glsresetentrylist` Resets `\glossaryentrynumbers`

```
5733 \newcommand*{\glsresetentrylist}{%
5734   \global\let\glossaryentrynumbers\org@glossaryentrynumbers}
```

`\glsnonextpages` Outside of `\printglossary` this does nothing.

```
5735 \newcommand*{\glsnonextpages}{}%
```

`\glsnextpages` Outside of `\printglossary` this does nothing.

```
5736 \newcommand*{\glsnextpages}{}%
```

`glossaryentry` If the `entrycounter` package option has been used, define a counter to number each level 0 entry.

```
5737 \ifglentrycounter
5738   \ifx\@gls@counterwithin\@empty
5739     \newcounter{glossaryentry}
5740   \else
5741     \newcounter{glossaryentry}[\@gls@counterwithin]
5742   \fi
5743   \def\theHglossaryentry{\currentglossary.\theglossaryentry}
5744 \fi
```

`glossarysubentry` If the `subentrycounter` package option has been used, define a counter to number each level 1 entry.

```
5745 \ifglssubentrycounter
5746   \ifglentrycounter
5747     \newcounter{glossarysubentry}[glossaryentry]
5748   \else
5749     \newcounter{glossarysubentry}
5750   \fi
5751   \def\theHglossarysubentry{\currentglssubentry.\theglossarysubentry}
5752 \fi
```

`esetsubentrycounter` Resets the `glossarysubentry` counter.

```
5753 \ifglssubentrycounter
5754   \newcommand*{\glsresetsubentrycounter}{%
5755     \setcounter{glossarysubentry}{0}%
5756   }
5757 \else
5758   \newcommand*{\glsresetsubentrycounter}{}%
5759 \fi
```

`\resetsubentrycounter` Resets the glossentry counter.

```
5760 \ifglentrycounter
5761   \newcommand*{\glsresetentrycounter}{%
5762     \setcounter{glossaryentry}{0}%
5763   }
5764 \else
5765   \newcommand*{\glsresetentrycounter}{}
5766 \fi
```

`\glsstepentry` Advance the glossaryentry counter if in use. The argument is the label associated with the entry.

```
5767 \ifglentrycounter
5768   \newcommand*{\glsstepentry}[1]{%
5769     \refstepcounter{glossaryentry}%
5770     \label{glentry-\glsdetoklabel{#1}}%
5771   }
5772 \else
5773   \newcommand*{\glsstepentry}[1]{}
5774 \fi
```

`\glsstepsubentry` Advance the glossarysubentry counter if in use. The argument is the label associated with the subentry.

```
5775 \ifglssubentrycounter
5776   \newcommand*{\glsstepsubentry}[1]{%
5777     \edef\currentglssubentry{\glsdetoklabel{#1}}%
5778     \refstepcounter{glossarysubentry}%
5779     \label{glentry-\currentglssubentry}%
5780   }
5781 \else
5782   \newcommand*{\glsstepsubentry}[1]{}
5783 \fi
```

`\glsrefentry` Reference the entry or sub-entry counter if in use, otherwise just do `\gls`.

```
5784 \ifglentrycounter
5785   \newcommand*{\glsrefentry}[1]{\ref{glentry-\glsdetoklabel{#1}}}
5786 \else
5787   \ifglssubentrycounter
5788     \newcommand*{\glsrefentry}[1]{\ref{glentry-\glsdetoklabel{#1}}}
5789   \else
5790     \newcommand*{\glsrefentry}[1]{\gls{#1}}
5791   \fi
5792 \fi
```

`\glsentrycounterlabel` Defines how to display the glossaryentry counter.

```
5793 \ifglentrycounter
5794   \newcommand*{\glsentrycounterlabel}{\theglossaryentry.\space}
5795 \else
5796   \newcommand*{\glsentrycounterlabel}{}
5797 \fi
```

ubentrycounterlabel Defines how to display the glossarysubentry counter.

```
5798 \ifglssubentrycounter
5799   \newcommand*{\glssubentrycounterlabel}{\theglossarysubentry}\space}
5800 \else
5801   \newcommand*{\glssubentrycounterlabel}{}
5802 \fi
```

\glentryitem Step and display glossaryentry counter, if appropriate.

```
5803 \ifglentrycounter
5804   \newcommand*{\glentryitem}[1]{%
5805     \glstepentry{#1}\glentrycounterlabel
5806   }
5807 \else
5808   \newcommand*{\glentryitem}[1]{\glresetsubentrycounter}
5809 \fi
```

\glssubentryitem Step and display glossarysubentry counter, if appropriate.

```
5810 \ifglssubentrycounter
5811   \newcommand*{\glssubentryitem}[1]{%
5812     \glstepsubentry{#1}\glssubentrycounterlabel
5813   }
5814 \else
5815   \newcommand*{\glssubentryitem}[1]{}
5816 \fi
```

theglossary If the theglossary environment has already been defined, a warning will be issued. This environment should be redefined by glossary styles.

```
5817 \ifcsundef{theglossary}%
5818 {%
5819   \newenvironment{theglossary}{}{}}%
5820 }%
5821 {%
5822   \@gls@warnontheglossdefined
5823   \renewenvironment{theglossary}{}{}}%
5824 }
```

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

```
5825 \newcommand*{\glossaryheader}{}%
```

\glstarget \glstarget{<label>}{<name>}

Provide user interface to `\@glstarget` to make it easier to modify the glossary style in the document.

```
5826 \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}
```

```
5827 \providecommand*{\compatibleglossentry}[2]{%
5828   \toks@{#2}%
5829   \protected@edef\@do@glossentry{\noexpand\glossaryentryfield{#1}%
5830     {\noexpand\glsnamefont
5831       {\expandafter\expandonce\csname glo@#1@name\endcsname}}}%
5832     {\expandafter\expandonce\csname glo@#1@desc\endcsname}}}%
5833     {\expandafter\expandonce\csname glo@#1@symbol\endcsname}}}%
5834     {\the\toks@}%
5835   }%
5836   \@do@glossentry
5837 }
```

`\glossentryname`

```
5838 \newcommand*{\glossentryname}[1]{%
5839   \glsdoifexistsorwarn{#1}%
5840   {%
5841     \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
5842     \expandafter\glsnamefont\expandafter{\glo@name}%
5843   }%
5844 }
```

`\Glossentryname`

```
5845 \newcommand*{\Glossentryname}[1]{%
5846   \glsdoifexistsorwarn{#1}%
5847   {%
5848     \glsnamefont{\Glsentryname{#1}}%
5849   }%
5850 }
```

`\glossentrydesc`

```
5851 \newcommand*{\glossentrydesc}[1]{%
5852   \glsdoifexistsorwarn{#1}%
5853   {%
5854     \glsentrydesc{#1}%
5855   }%
5856 }
```

\Glossentrydesc

```
5857 \newcommand*{\Glossentrydesc}[1]{%
5858   \glsdoifexistsorwarn{#1}%
5859   {%
5860     \Glsentrydesc{#1}%
5861   }%
5862 }
```

\glossentrysymbol

```
5863 \newcommand*{\glossentrysymbol}[1]{%
5864   \glsdoifexistsorwarn{#1}%
5865   {%
5866     \glsentrysymbol{#1}%
5867   }%
5868 }
```

\Glossentrysymbol

```
5869 \newcommand*{\Glossentrysymbol}[1]{%
5870   \glsdoifexistsorwarn{#1}%
5871   {%
5872     \Glsentrysymbol{#1}%
5873   }%
5874 }
```

patiblesubglossentry

`\subglossentry{<level>}{<label>}{<page-list>}`

```
5875 \providecommand*{\compatiblesubglossentry}[3]{%
5876   \toks@{#3}%
5877   \protected@edef\@do@subglossentry{\noexpand\glossarysubentryfield{\number#1}%
5878     {#2}%
5879     {\noexpand\glsnamefont
5880       {\expandafter\expandonce\csname glo@#2@name\endcsname}}%
5881     {\expandafter\expandonce\csname glo@#2@desc\endcsname}%
5882     {\expandafter\expandonce\csname glo@#2@symbol\endcsname}%
5883     {\the\toks@}}%
5884   }%
5885   \@do@subglossentry
5886 }
```

sentrycompatibility

```
5887 \newcommand*{\setglossentrycompatibility}{%
5888   \let\glossentry\compatibleglossentry
5889   \let\subglossentry\compatiblesubglossentry
5890 }
5891 \setglossentrycompatibility
```


`\glossaryentryfield`

`\glossaryentryfield{<label>}{<name>}{<description>}{<symbol>}{<page-list>}`

This command formerly governed how each entry row should be formatted in the glossary. Now deprecated.

```
5892 \newcommand{\glossaryentryfield}[5]{%
5893   \GlossariesWarning
5894   {Deprecated use of \string\glossaryentryfield.^^J
5895     I recommend you change to \string\glossentry.^^J
5896     If you've just upgraded, try removing your gls auxiliary
5897     files^^J and recompile}%
5898   \noindent\textbf{\glstarget{#1}{#2}} #4 #3. #5\par}
```

`\glossarysubentryfield`

`\glossarysubentryfield{<level>}{<label>}{<name>}{<description>}{<symbol>}{<page-list>}`

This command governs how each subentry should be formatted in the glossary. Glossary styles need to redefine this command. Most of the predefined styles ignore *<symbol>*. The first argument is a number indicating the level. (The level should be greater than or equal to 1.)

```
5899 \newcommand*{\glossarysubentryfield}[6]{%
5900   \GlossariesWarning
5901   {Deprecated use of \string\glossarysubentryfield.^^J
5902     I recommend you change to \string\subglossentry.^^J
5903     If you've just upgraded, try removing your gls auxiliary
5904     files^^J and recompile}%
5905   \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`

```
5906 \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

redefined this command. (In between groups, `\glsgroupheading` comes immediately after `\glsgroupskip`.)

`\glsgroupheading`

```
5907 \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 `\glssymbolsgroupname`, 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`

```
5908 \newcommand*{\glsgetgrouptitle}[1]{%
5909   \@gls@getgrouptitle{#1}{\@gls@grptitle}%
5910   \@gls@grptitle
5911 }
```

`\@gls@getgrouptitle` Gets the group title specified by the label (first argument) and stores in the second argument, which must be a control sequence.

```
5912 \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.

```
5913 \dtl@ifsingle{#1}%
5914 {%
5915   \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
5916 }%
5917 {%
5918   \ifboolexpr{test{\ifstrequal{#1}{glsymbols}}
5919               or test{\ifstrequal{#1}{glsnumbers}}}%
5920   {%
5921     \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
```

```

5922 }%
5923 {%
5924     \def#2{#1}%
5925 }%
5926 }%
5927 }

```

`@getothergrouptitle` Version for the no-indexing app option:

```

5928 \newcommand*{\@gls@noidx@getgrouptitle}[2]{%
5929     \DTLifint{#1}%
5930     {\edef#2{\char#1\relax}}%
5931     {%
5932         \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
5933     }%
5934 }

```

`\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`

```

5935 \newcommand*{\glsgetgrouplabel}[1]{%
5936 \ifthenelse{\equal{#1}{\glssymbolsgroupname}}{\glssymbols}{%
5937 \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`

```

5938 \newcommand*{\setentrycounter}[2][ ]{%
5939     \def\@glo@counterprefix{#1}%
5940     \ifx\@glo@counterprefix\@empty
5941         \def\@glo@counterprefix{.}%
5942     \else
5943         \def\@glo@counterprefix{.#1.}%
5944     \fi
5945     \def\glsentrycounter{#2}%
5946 }

```

The current glossary style can be set using `\setglossarystyle{<style>}`.

`\setglossarystyle`

```

5947 \newcommand*{\setglossarystyle}[1]{%
5948     \ifcsundef{@glsstyle@#1}%
5949     {%

```

```

5950 \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
5951 }%
5952 {%
5953 \csname @glsstyle@#1\endcsname
5954 }%
5955 }

```

`\glossarystyle`

```

5956 \newcommand*{\glossarystyle}[1]{%
5957 \ifcsundef{@glsstyle@#1}%
5958 {%
5959 \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
5960 }%
5961 {%
5962 \GlossariesWarning
5963 {Deprecated command \string\glossarystyle.^^J
5964 I recommend you switch to \string\setglossarystyle\space unless
5965 you want to maintain backward compatibility}%
5966 \setglossentrycompatibility
5967 \csname @glsstyle@#1\endcsname

5968 \ifcsdef{@glscompstyle@#1}%
5969 {\setglossentrycompatibility\csuse{@glscompstyle@#1}}%
5970 }%
5971 }%
5972 }

```

`\newglossarystyle` New glossary styles can be defined using:

```
\newglossarystyle{<name>}{<definition>}
```

The *<definition>* argument should redefine `\theglossary`, `\glossaryheader`, `\glsgroupheading`, `\glossaryentryfield` and `\glsgroupskip` (see [subsection 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.

```

5973 \newcommand{\newglossarystyle}[2]{%
5974 \ifcsundef{@glsstyle@#1}%
5975 {%
5976 \expandafter\def\csname @glsstyle@#1\endcsname{#2}%
5977 }%
5978 {%
5979 \PackageError{glossaries}{Glossary style ‘#1’ is already defined}{}%
5980 }%
5981 }

```

`\renewglossarystyle` Code for this macro supplied by Marco Daniel.

```

5982 \newcommand{\renewglossarystyle}[2]{%
5983 \ifcsundef{@glsstyle@#1}%

```

```

5984  {%
5985    \PackageError{glossaries}{Glossary style ‘#1’ isn’t already defined}{}%
5986  }%
5987  {%
5988    \csdef{@glsstyle@#1}{#2}%
5989  }%
5990 }

```

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`

```

5991 \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`

```

5992 \ifcsundef{hyperlink}%
5993 {%
5994   \def\glshypernumber#1{#1}%
5995 }%
5996 {%
5997   \def\glshypernumber#1{\@glshypernumber#1\nohyperpage{}}\@nil}
5998 }

```

`\@glshypernumber` This code was provided by Heiko Oberdiek to allow material to be attached to the location.

```

5999 \def\@glshypernumber#1\nohyperpage#2#3\@nil{%
6000   \ifx\#1\%
6001     \else
6002       \@delimR#1\delimR\delimR\%

```

```

6003 \fi
6004 \ifx\|#2\|%
6005 \else
6006 #2%
6007 \fi
6008 \ifx\|#3\|%
6009 \else
6010 \@glshypernumber#3\@nil
6011 \fi
6012 }

```

`\@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`

```

6013 \def\@delimR#1\delimR #2\delimR #3\|%
6014 \ifx\|#2\|%
6015 \@delimN{#1}%
6016 \else
6017 \@gls@numberlink{#1}\delimR\@gls@numberlink{#2}%
6018 \fi}

```

`\@delimN` displays a list of individual numbers, instead of a range:

`\@delimN`

```

6019 \def\@delimN#1{\@delimN#1\delimN \@delimN\}
6020 \def\@delimN#1\delimN #2\delimN#3\|%
6021 \ifx\|#3\|%
6022 \@gls@numberlink{#1}%
6023 \else
6024 \@gls@numberlink{#1}\delimN\@gls@numberlink{#2}%
6025 \fi
6026 }

```

The following code is modified from `hyperref's \HyInd@pagelink` where the name of the counter being used is given by `\@gls@counter`.

```

6027 \def\@gls@numberlink#1{%
6028 \begingroup
6029 \toks@={}%
6030 \@gls@removespaces#1 \@nil
6031 \endgroup}

6032 \def\@gls@removespaces#1 #2\@nil{%
6033 \toks@=\expandafter{\the\toks@#1}%
6034 \ifx\|#2\|%
6035 \edef\x{\the\toks@}%
6036 \ifx\x\empty
6037 \else
6038 \hyperlink{\glstrycounter\@gls@counterprefix\the\toks@}%

```

```

6039             {\the\toks@}%
6040     \fi
6041   \else
6042     \@gls@ReturnAfterFi{%
6043       \@gls@removespaces#2\@nil
6044     }%
6045   \fi
6046 }
6047 \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`

```
6048 \newcommand*\hyperrm[1]{\textrm{\glshypernumber{#1}}}
```

`\hypersf`

```
6049 \newcommand*\hypersf[1]{\textsf{\glshypernumber{#1}}}
```

`\hypertt`

```
6050 \newcommand*\hypertt[1]{\texttt{\glshypernumber{#1}}}
```

`\hyperbf`

```
6051 \newcommand*\hyperbf[1]{\textbf{\glshypernumber{#1}}}
```

`\hypermd`

```
6052 \newcommand*\hypermd[1]{\textmd{\glshypernumber{#1}}}
```

`\hyperit`

```
6053 \newcommand*\hyperit[1]{\textit{\glshypernumber{#1}}}
```

`\hypersl`

```
6054 \newcommand*\hypersl[1]{\textsl{\glshypernumber{#1}}}
```

`\hyperup`

```
6055 \newcommand*\hyperup[1]{\textup{\glshypernumber{#1}}}
```

`\hypersc`

```
6056 \newcommand*\hypersc[1]{\textsc{\glshypernumber{#1}}}
```

`\hyperemph`

```
6057 \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.

```
6058 \newcommand{\oldacronym}[4][\gls@label]{%
6059   \def\gls@label{#2}%
6060   \newacronym[#4]{#1}{#2}{#3}%
6061   \ifcsundef{xspace}%
6062   {%
6063     \expandafter\edef\csname#1\endcsname{%
6064       \noexpand\@ifstar{\noexpand\Gls{#1}}{\noexpand\gls{#1}}}%
6065   }%
6066 }%
6067 {%
6068   \expandafter\edef\csname#1\endcsname{%
6069     \noexpand\@ifstar{\noexpand\Gls{#1}\noexpand\xspace}{%
6070       \noexpand\gls{#1}\noexpand\xspace}%
6071   }%
6072 }%
6073 }
```

`\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`

```
6074 \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.

```
6075 \newcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}
```

If `garamondx` has been loaded, need to use `\textulc` instead of `\textup`.

`\glstextup`

```
6076 \newrobustcmd*{\glstextup}[1]{\ifdef\textulc{\textulc{#1}}{\textup{#1}}}
```

The following are defined for compatibility with version 2.07 and earlier.

`\glsshortkey`

```
6077 \newcommand*{\glsshortkey}{short}
```

`\glsshortpluralkey`

```
6078 \newcommand*{\glsshortpluralkey}{shortplural}
```

`\glslongkey`

```
6079 \newcommand*{\glslongkey}{long}
```

`\glslongpluralkey`

```
6080 \newcommand*{\glslongpluralkey}{longplural}
```

`\acrfull` Full form of the acronym.

```
6081 \newrobustcmd*{\acrfull}{\@gls@hyp@opt\ns@acrfull}
```

```
6082 \newcommand*{\ns@acrfull}[2] [] {%
```

```
6083 \new@ifnextchar[{\@acrfull{#1}{#2}}{%
```

```
6084 \@acrfull{#1}{#2} [] }%
```

```
6085 }
```

`\@acrfull` Low-level macro:

```
6086 \def\@acrfull#1#2[#3] {%
```

Make it easier for acronym styles to change this:

```
6087 \acrfullfmt{#1}{#2}{#3}%
```

```
6088 }
```

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.

```
6089 \newcommand*\acrfullfmt}[3]{%
6090   \acrlinkfullformat{\@acrlong}{\@acrshort}{#1}{#2}{#3}%
6091 }
```

`\acrlinkfullformat` Format for full links like `\acrfull`. Syntax: `\acrlinkfullformat{<long cs>}{<short cs>}{<options>}{<label>}{<insert>}`

```
6092 \newcommand*\acrlinkfullformat}[5]{%
6093   \acrfullformat{#1}{#3}{#4}[#5]{#2}{#3}{#4}[]}%
6094 }
```

`\acrfullformat` Default full form is `<long>` (`<short>`).

```
6095 \newcommand*\acrfullformat}[2]{#1\glsspace(#2)}
```

`\glsspace` Robust space to ensure it's written to the `.glsdefs` file.

```
6096 \newrobustcmd*\glsspace{\space}
```

Default format for full acronym

`\Acrfull`

```
6097 \newrobustcmd*\Acrfull{\@gls@hyp@opt\ns@Acrfull}

6098 \newcommand*\ns@Acrfull[2][]{%
6099   \new@ifnextchar[{\@Acrfull{#1}{#2}}%
6100     {\@Acrfull{#1}{#2}[]}%
6101 }
```

Low-level macro:

```
6102 \def\@Acrfull#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6103   \Acrfullfmt{#1}{#2}{#3}%
6104 }
```

`\Acrfullfmt` First letter upper case full format.

```
6105 \newcommand*\Acrfullfmt}[3]{%
6106   \acrlinkfullformat{\@Acrlong}{\@acrshort}{#1}{#2}{#3}%
6107 }
```

`\ACRfull`

```
6108 \newrobustcmd*\ACRfull{\@gls@hyp@opt\ns@ACRfull}

6109 \newcommand*\ns@ACRfull[2][]{%
6110   \new@ifnextchar[{\@ACRfull{#1}{#2}}%
6111     {\@ACRfull{#1}{#2}[]}%
6112 }
```

Low-level macro:

```
6113 \def\@ACRfull#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6114 \ACRfullfmt{#1}{#2}{#3}%  
6115 }
```

\ACRfullfmt All upper case full format.

```
6116 \newcommand*\ACRfullfmt}[3]{%  
6117 \acrlinkfullformat{\@ACRlong}{\@ACRshort}{#1}{#2}{#3}%  
6118 }
```

Plural:

\acrfullpl

```
6119 \newrobustcmd*\acrfullpl{\@gls@hyp@opt\ns@acrfullpl}  
  
6120 \newcommand*\ns@acrfullpl[2][ ]{%  
6121 \new@ifnextchar[{\@acrfullpl{#1}{#2}}%  
6122 {\@acrfullpl{#1}{#2}[ ]}%  
6123 }
```

Low-level macro:

```
6124 \def\@acrfullpl#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6125 \acrfullplfmt{#1}{#2}{#3}%  
6126 }
```

\acrfullplfmt No case change plural full format.

```
6127 \newcommand*\acrfullplfmt}[3]{%  
6128 \acrlinkfullformat{\@acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%  
6129 }
```

\Acrfullpl

```
6130 \newrobustcmd*\Acrfullpl{\@gls@hyp@opt\ns@Acrfullpl}  
  
6131 \newcommand*\ns@Acrfullpl[2][ ]{%  
6132 \new@ifnextchar[{\@Acrfullpl{#1}{#2}}%  
6133 {\@Acrfullpl{#1}{#2}[ ]}%  
6134 }
```

Low-level macro:

```
6135 \def\@Acrfullpl#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6136 \Acrfullplfmt{#1}{#2}{#3}%  
6137 }
```

`\Acrfullplfmt` First letter upper case plural full format.

```
6138 \newcommand*\Acrfullplfmt}[3]{%
6139   \acrlinkfullformat{\@Acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%
6140 }
```

`\ACRfullpl`

```
6141 \newrobustcmd*\ACRfullpl{\@gls@hyp@opt\ns@ACRfullpl}

6142 \newcommand*\ns@ACRfullpl[2][{}]{%
6143   \new@ifnextchar[{\@ACRfullpl{#1}{#2}}%
6144     {\@ACRfullpl{#1}{#2}[]}%
6145 }
```

Low-level macro:

```
6146 \def\@ACRfullpl#1#2[#3]{%
  Make it easier for acronym styles to change this:
6147   \ACRfullplfmt{#1}{#2}{#3}%
6148 }
```

`\ACRfullplfmt` All upper case plural full format.

```
6149 \newcommand*\ACRfullplfmt}[3]{%
6150   \acrlinkfullformat{\@ACRlongpl}{\@ACRshortpl}{#1}{#2}{#3}%
6151 }
```

1.18 Predefined acronym styles

`\acronymfont` This is only used with the additional acronym styles:

```
6152 \newcommand{\acronymfont}[1]{#1}
```

`\firstacronymfont` This is only used with the additional acronym styles:

```
6153 \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.

```
6154 \newcommand*\acrnameformat[2]{\acronymfont{#1}}
```

Define some tokens used by `\newacronym`:

`\glskeylisttok`

```
6155 \newtoks\glskeylisttok
```

`\glslabeltok`

```
6156 \newtoks\glslabeltok
```

`\glsshorttok`

```
6157 \newtoks\glsshorttok
```

```

\glslongtok
6158 \newtoks\glslongtok

\newacronymhook Provide a hook for \newacronym:
6159 \newcommand*{\newacronymhook}{}

etGenericNewAcronym New improved version of setting the acronym style.
6160 \newcommand*{\SetGenericNewAcronym}{%
    Change the behaviour of \Glsentryname to workaround expansion issues that
    cause a problem for \makefirstuc
6161 \let\@Gls@entryname\@Gls@acentryname
    Change the way acronyms are defined:
6162 \renewcommand{\newacronym}[4][\]{%
6163 \ifdefempty{\@glsacronymlists}%
6164 {%
6165 \def\@glo@type{\acronymtype}%
6166 \setkeys{glossentry}{##1}%
6167 \DeclareAcronymList{\@glo@type}%
6168 }%
6169 }%
6170 \glskeylisttok{##1}%
6171 \glslabeltok{##2}%
6172 \glsshorttok{##3}%
6173 \glslongtok{##4}%
6174 \newacronymhook
6175 \protected@edef\@do@newglossaryentry{%
6176 \noexpand\newglossaryentry{\the\glslabeltok}%
6177 {%
6178 type=\acronymtype,%
6179 name={\expandonce{\acronymentry{##2}}},%
6180 sort={\acronymsort{\the\glsshorttok}{\the\glslongtok}},%
6181 text={\the\glsshorttok},%
6182 short={\the\glsshorttok},%
6183 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6184 long={\the\glslongtok},%
6185 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6186 \GenericAcronymFields,%
6187 \the\glskeylisttok
6188 }%
6189 }%
6190 \@do@newglossaryentry
6191 }%
    Make sure that \acrfull etc reflects the new style:
6192 \renewcommand*{\acrfullfmt}[3]{%
6193 \glslink[##1]{##2}{\genacrfullformat{##2}{##3}}}%
6194 \renewcommand*{\Acrfullfmt}[3]{%
6195 \glslink[##1]{##2}{\Genacrfullformat{##2}{##3}}}%

```

```

6196 \renewcommand*\ACRfullfmt}[3]{%
6197   \glslink[##1]{##2}{%
6198     \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}%
6199 \renewcommand*\acrfullplfmt}[3]{%
6200   \glslink[##1]{##2}{\genplacrfullformat{##2}{##3}}}%
6201 \renewcommand*\Acrfullplfmt}[3]{%
6202   \glslink[##1]{##2}{\Genplacrfullformat{##2}{##3}}}%
6203 \renewcommand*\ACRfullplfmt}[3]{%
6204   \glslink[##1]{##2}{%
6205     \mfirstucMakeUppercase{\genplacrfullformat{##2}{##3}}}%

```

Make sure that `\glsentryfull` etc reflects the new style:

```

6206 \renewcommand*\glsentryfull}[1]{\genacrfullformat{##1}{}}%
6207 \renewcommand*\Glsentryfull}[1]{\Genacrfullformat{##1}{}}%
6208 \renewcommand*\glsentryfullpl}[1]{\genplacrfullformat{##1}{}}%
6209 \renewcommand*\Glsentryfullpl}[1]{\Genplacrfullformat{##1}{}}%
6210 }

```

`\GenericAcronymFields` Fields used by `\SetGenericNewAcronym` that can be changed by the acronym style.

```

6211 \newcommand*\GenericAcronymFields{description={\the\glslongtok}}

```

`\acronymentry` `\acronymentry{<label>}`

Display style for the name field in the list of acronyms.

```

6212 \newcommand*\acronymentry}[1]{\acronymfont{\glsentryshort{#1}}}

```

`\acronymsort` `\acronymsort{<short>}{<long>}`

Default sort format for acronyms.

```

6213 \newcommand*\acronymsort}[2]{#1}

```

`\setacronymstyle` `\setacronymstyle{<style name>}`

```

6214 \newcommand*\setacronymstyle}[1]{%
6215   \ifcsundef{@glsacr@dispstyle@#1}%
6216   {%
6217     \PackageError{glossaries}{Undefined acronym style ‘#1’}{}%
6218   }%
6219   {%
6220     \ifdefempty{\@glsacronymlists}%
6221     {%
6222       \DeclareAcronymList{acronymtype}%
6223     }%

```

```

6224    {}%
6225    \SetGenericNewAcronym
6226    \GlsUseAcrStyleDefs{#1}%
6227    \@for\@gls@type:=\@glsacronymlists\do{%
6228        \defglsentryfmt[\@gls@type]{\GlsUseAcrEntryDispStyle{#1}}%
6229    }%
6230 }%
6231 }

```

`\newacronymstyle` `\newacronymstyle{<style name>}{<entry format definition>}{<display definitions>}`

Defines a new acronym style called *<style name>*.

```

6232 \newcommand*{\newacronymstyle}[3]{%
6233     \ifcsdef{@glsacr@dispstyle@#1}%
6234     {%
6235         \PackageError{glossaries}{Acronym style ‘#1’ already exists}{}%
6236     }%
6237     {%
6238         \csdef{@glsacr@dispstyle@#1}{#2}%
6239         \csdef{@glsacr@styledefs@#1}{#3}%
6240     }%
6241 }

```

`\renewacronymstyle` Redefines the given acronym style.

```

6242 \newcommand*{\renewacronymstyle}[3]{%
6243     \ifcsdef{@glsacr@dispstyle@#1}%
6244     {%
6245         \csdef{@glsacr@dispstyle@#1}{#2}%
6246         \csdef{@glsacr@styledefs@#1}{#3}%
6247     }%
6248     {%
6249         \PackageError{glossaries}{Acronym style ‘#1’ doesn’t exist}{}%
6250     }%
6251 }

```

`\seAcrEntryDispStyle`

```

6252 \newcommand*{\GlsUseAcrEntryDispStyle}[1]{\csuse{@glsacr@dispstyle@#1}}

```

`\GlsUseAcrStyleDefs`

```

6253 \newcommand*{\GlsUseAcrStyleDefs}[1]{\csuse{@glsacr@styledefs@#1}}

```

Predefined acronym styles:

`long-short` *<long>* (*<short>*) acronym style.

```

6254 \newacronymstyle{long-short}%
6255 {%

```

Check for long form in case this is a mixed glossary.

```

6256 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6257 }%
6258 {%
6259 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6260 \renewcommand*{\genacrfullformat}[2]{%
6261   \glsentrylong{##1}##2\space
6262   (\protect\firstacronymfont{\glsentryshort{##1}})%
6263 }%
6264 \renewcommand*{\Genacrfullformat}[2]{%
6265   \Glsentrylong{##1}##2\space
6266   (\protect\firstacronymfont{\glsentryshort{##1}})%
6267 }%
6268 \renewcommand*{\genplacrfullformat}[2]{%
6269   \glsentrylongpl{##1}##2\space
6270   (\protect\firstacronymfont{\glsentryshortpl{##1}})%
6271 }%
6272 \renewcommand*{\Genplacrfullformat}[2]{%
6273   \Glsentrylongpl{##1}##2\space
6274   (\protect\firstacronymfont{\glsentryshortpl{##1}})%
6275 }%
6276 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6277 \renewcommand*{\acronymsort}[2]{##1}%
6278 \renewcommand*{\acronymfont}[1]{##1}%
6279 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
6280 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6281 }

```

long-sp-short Similar to the previous style but allows the space between the long and short form to be customized.

```

6282 \newacronymstyle{long-sp-short}%
6283 {%

```

Check for long form in case this is a mixed glossary.

```

6284 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6285 }%
6286 {%
6287 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6288 \renewcommand*{\genacrfullformat}[2]{%
6289   \glsentrylong{##1}##2\glsacspace{##1}%
6290   (\protect\firstacronymfont{\glsentryshort{##1}})%
6291 }%
6292 \renewcommand*{\Genacrfullformat}[2]{%
6293   \Glsentrylong{##1}##2\glsacspace{##1}%
6294   (\protect\firstacronymfont{\glsentryshort{##1}})%
6295 }%
6296 \renewcommand*{\genplacrfullformat}[2]{%
6297   \glsentrylongpl{##1}##2\glsacspace{##1}%
6298   (\protect\firstacronymfont{\glsentryshortpl{##1}})%

```



```

6299 }%
6300 \renewcommand*\Genplacrfullformat}[2]{%
6301   \Glsentrylongpl{##1}##2\glsacspace{##1}%
6302   (\protect\firstacronymfont{\glsentryshortpl{##1}})%
6303 }%
6304 \renewcommand*\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6305 \renewcommand*\acronymsort}[2]{##1}%
6306 \renewcommand*\acronymfont}[1]{##1}%
6307 \renewcommand*\firstacronymfont}[1]{\acronymfont{##1}}%
6308 \renewcommand*\acrpluralsuffix{\glspluralsuffix}%
6309 }

```

`\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.

```

6310 \newcommand*\glsacspace}[1]{%
6311   \settowidth{\dimen@}{(\firstacronymfont{\glsentryshort{##1}})%}
6312   \ifdim\dimen@<3em~\else\space\fi
6313 }

```

`short-long` *<short>* (*<long>*) acronym style.

```

6314 \newacronymstyle{short-long}%
6315 {%
  Check for long form in case this is a mixed glossary.
6316   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6317 }%
6318 {%
6319   \renewcommand*\GenericAcronymFields{description={\the\glslongtok}}%
6320   \renewcommand*\genacrfullformat}[2]{%
6321     \protect\firstacronymfont{\glsentryshort{##1}}##2\space
6322     (\glsentrylong{##1})%
6323   }%
6324   \renewcommand*\Genacrfullformat}[2]{%
6325     \protect\firstacronymfont{\Glsentryshort{##1}}##2\space
6326     (\glsentrylong{##1})%
6327   }%
6328   \renewcommand*\genplacrfullformat}[2]{%
6329     \protect\firstacronymfont{\glsentryshortpl{##1}}##2\space
6330     (\glsentrylongpl{##1})%
6331   }%
6332   \renewcommand*\Genplacrfullformat}[2]{%
6333     \protect\firstacronymfont{\Glsentryshortpl{##1}}##2\space
6334     (\glsentrylongpl{##1})%
6335   }%
6336   \renewcommand*\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6337   \renewcommand*\acronymsort}[2]{##1}%
6338   \renewcommand*\acronymfont}[1]{##1}%
6339   \renewcommand*\firstacronymfont}[1]{\acronymfont{##1}}%

```

```

6340 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6341 }

```

long-sc-short *<long>* (\textsc{<short>}) acronym style.

```

6342 \newacronymstyle{long-sc-short}%
6343 {%
6344   \GlsUseAcrEntryDisplayStyle{long-short}%
6345 }%
6346 {%
6347   \GlsUseAcrStyleDefs{long-short}%
6348   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6349   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6350 }

```

long-sm-short *<long>* (\textsmaller{<short>}) acronym style.

```

6351 \newacronymstyle{long-sm-short}%
6352 {%
6353   \GlsUseAcrEntryDisplayStyle{long-short}%
6354 }%
6355 {%
6356   \GlsUseAcrStyleDefs{long-short}%
6357   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6358   \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6359 }

```

sc-short-long *<short>* (\textsc{<long>}) acronym style.

```

6360 \newacronymstyle{sc-short-long}%
6361 {%
6362   \GlsUseAcrEntryDisplayStyle{short-long}%
6363 }%
6364 {%
6365   \GlsUseAcrStyleDefs{short-long}%
6366   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6367   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6368 }

```

sm-short-long *<short>* (\textsmaller{<long>}) acronym style.

```

6369 \newacronymstyle{sm-short-long}%
6370 {%
6371   \GlsUseAcrEntryDisplayStyle{short-long}%
6372 }%
6373 {%
6374   \GlsUseAcrStyleDefs{short-long}%
6375   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6376   \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6377 }

```

long-short-desc *<long>* ({<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

6378 \newacronymstyle{long-short-desc}%
6379 {%
6380   \GlsUseAcrEntryDispStyle{long-short}%
6381 }%
6382 {%
6383   \GlsUseAcrStyleDefs{long-short}%
6384   \renewcommand*{\GenericAcronymFields}{}%
6385   \renewcommand*{\acronymsort}[2]{##2}%
6386   \renewcommand*{\acronymentry}[1]{%
6387     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6388 }

```

long-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.

```

6389 \newacronymstyle{long-sp-short-desc}%
6390 {%
6391   \GlsUseAcrEntryDispStyle{long-sp-short}%
6392 }%
6393 {%
6394   \GlsUseAcrStyleDefs{long-sp-short}%
6395   \renewcommand*{\GenericAcronymFields}{}%
6396   \renewcommand*{\acronymsort}[2]{##2}%
6397   \renewcommand*{\acronymentry}[1]{%
6398     \glentrylong{##1}\glsacspace{##1}(\acronymfont{\glentryshort{##1}})}%
6399 }

```

long-sc-short-desc *⟨long⟩* (\textsc{⟨short⟩}) acronym style that has an accompanying description (which the user needs to supply).

```

6400 \newacronymstyle{long-sc-short-desc}%
6401 {%
6402   \GlsUseAcrEntryDispStyle{long-sc-short}%
6403 }%
6404 {%
6405   \GlsUseAcrStyleDefs{long-sc-short}%
6406   \renewcommand*{\GenericAcronymFields}{}%
6407   \renewcommand*{\acronymsort}[2]{##2}%
6408   \renewcommand*{\acronymentry}[1]{%
6409     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6410 }

```

long-sm-short-desc *⟨long⟩* (\textsmaller{⟨short⟩}) acronym style that has an accompanying description (which the user needs to supply).

```

6411 \newacronymstyle{long-sm-short-desc}%
6412 {%
6413   \GlsUseAcrEntryDispStyle{long-sm-short}%
6414 }%
6415 {%

```

```

6416 \GlsUseAcrStyleDefs{long-sm-short}%
6417 \renewcommand*{\GenericAcronymFields}{}%
6418 \renewcommand*{\acronymsort}[2]{##2}%
6419 \renewcommand*{\acronymentry}[1]{%
6420     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6421 }

```

short-long-desc *<short>* (*{<long>}*) acronym style that has an accompanying description (which the user needs to supply).

```

6422 \newacronymstyle{short-long-desc}%
6423 {%
6424     \GlsUseAcrEntryDisplayStyle{short-long}%
6425 }%
6426 {%
6427     \GlsUseAcrStyleDefs{short-long}%
6428     \renewcommand*{\GenericAcronymFields}{}%
6429     \renewcommand*{\acronymsort}[2]{##2}%
6430     \renewcommand*{\acronymentry}[1]{%
6431         \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6432 }

```

sc-short-long-desc *<long>* (\textsc{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

6433 \newacronymstyle{sc-short-long-desc}%
6434 {%
6435     \GlsUseAcrEntryDisplayStyle{sc-short-long}%
6436 }%
6437 {%
6438     \GlsUseAcrStyleDefs{sc-short-long}%
6439     \renewcommand*{\GenericAcronymFields}{}%
6440     \renewcommand*{\acronymsort}[2]{##2}%
6441     \renewcommand*{\acronymentry}[1]{%
6442         \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6443 }

```

sm-short-long-desc *<long>* (\textsmaller{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

6444 \newacronymstyle{sm-short-long-desc}%
6445 {%
6446     \GlsUseAcrEntryDisplayStyle{sm-short-long}%
6447 }%
6448 {%
6449     \GlsUseAcrStyleDefs{sm-short-long}%
6450     \renewcommand*{\GenericAcronymFields}{}%
6451     \renewcommand*{\acronymsort}[2]{##2}%
6452     \renewcommand*{\acronymentry}[1]{%
6453         \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6454 }

```

dua *⟨long⟩* only acronym style.

```
6455 \newacronymstyle{dua}%  
6456 {%
```

Check for long form in case this is a mixed glossary.

```
6457 \ifdefempty\glscustomtext  
6458 {%  
6459 \ifglshaslong{\glslabel}%  
6460 {%  
6461 \glsifplural  
6462 {%
```

Plural form:

```
6463 \glscapscase  
6464 {%
```

Plural form, don't adjust case:

```
6465 \glentrylongpl{\glslabel}\glsinsert  
6466 }%  
6467 {%
```

Plural form, make first letter upper case:

```
6468 \Glentrylongpl{\glslabel}\glsinsert  
6469 }%  
6470 {%
```

Plural form, all caps:

```
6471 \mfirstucMakeUppercase  
6472 {\glentrylongpl{\glslabel}\glsinsert}%  
6473 }%  
6474 }%  
6475 {%
```

Singular form

```
6476 \glscapscase  
6477 {%
```

Singular form, don't adjust case:

```
6478 \glentrylong{\glslabel}\glsinsert  
6479 }%  
6480 {%
```

Subsequent singular form, make first letter upper case:

```
6481 \Glentrylong{\glslabel}\glsinsert  
6482 }%  
6483 {%
```

Subsequent singular form, all caps:

```
6484 \mfirstucMakeUppercase  
6485 {\glentrylong{\glslabel}\glsinsert}%  
6486 }%  
6487 }%  
6488 }%  
6489 {%
```

Not an acronym:

```
6490     \glsgenentryfmt
6491     }%
6492     }%
6493     {\glscustomtext\glsinsert}%
6494 }%
6495 {%
6496     \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%

6497     \renewcommand*{\acrfullfmt}[3]{%
6498         \glslink[##1]{##2}{\glsentrylong{##2}##3\space
6499             (\acronymfont{\glsentryshort{##2}})}}%
6500     \renewcommand*{\Acrfullfmt}[3]{%
6501         \glslink[##1]{##2}{\Glsentrylong{##2}##3\space
6502             (\acronymfont{\glsentryshort{##2}})}}%
6503     \renewcommand*{\ACRfullfmt}[3]{%
6504         \glslink[##1]{##2}{%
6505             \mfirstucMakeUppercase{\glsentrylong{##2}##3\space
6506                 (\acronymfont{\glsentryshort{##2}})}}}%

6507     \renewcommand*{\acrfullplfmt}[3]{%
6508         \glslink[##1]{##2}{\glsentrylongpl{##2}##3\space
6509             (\acronymfont{\glsentryshortpl{##2}})}}%

6510     \renewcommand*{\Acrfullplfmt}[3]{%
6511         \glslink[##1]{##2}{\Glsentrylongpl{##2}##3\space
6512             (\acronymfont{\glsentryshortpl{##2}})}}%
6513     \renewcommand*{\ACRfullplfmt}[3]{%
6514         \glslink[##1]{##2}{%
6515             \mfirstucMakeUppercase{\glsentrylongpl{##2}##3\space
6516                 (\acronymfont{\glsentryshortpl{##2}})}}}%
6517     \renewcommand*{\glsentryfull}[1]{%
6518         \glsentrylong{##1}\space(\acronymfont{\glsentryshort{##1}})%
6519     }%
6520     \renewcommand*{\Glsentryfull}[1]{%
6521         \Glsentrylong{##1}\space(\acronymfont{\glsentryshort{##1}})%
6522     }%
6523     \renewcommand*{\glsentryfullpl}[1]{%
6524         \glsentrylongpl{##1}\space(\acronymfont{\glsentryshortpl{##1}})%
6525     }%
6526     \renewcommand*{\Glsentryfullpl}[1]{%
6527         \Glsentrylongpl{##1}\space(\acronymfont{\glsentryshortpl{##1}})%
6528     }%
6529     \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6530     \renewcommand*{\acronymsort}[2]{##1}%
6531     \renewcommand*{\acronymfont}[1]{##1}%
6532     \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6533 }
```

dua-desc *<long>* only acronym style with user-supplied description.

```
6534 \newacronymstyle{dua-desc}%
6535 {%
6536   \GlsUseAcrEntryDispStyle{dua}%
6537 }%
6538 {%
6539   \GlsUseAcrStyleDefs{dua}%
6540   \renewcommand*{\GenericAcronymFields}{}%

6541   \renewcommand*{\acronymentry}[1]{\acronymfont{\glentrylong{##1}}}%
6542   \renewcommand*{\acronymsort}[2]{##2}%
6543 }%
```

footnote *<short>*\footnote{*<long>*} acronym style.

```
6544 \newacronymstyle{footnote}%
6545 {%

  Check for long form in case this is a mixed glossary.

6546   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6547 }%
6548 {%
6549   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
```

Need to ensure hyperlinks are switched off on first use:

```
6550   \glshyperfirstfalse
6551   \renewcommand*{\genacrfullformat}[2]{%
6552     \protect\firstacronymfont{\glentryshort{##1}}##2%
6553     \protect\footnote{\glentrylong{##1}}%
6554   }%
6555   \renewcommand*{\Genacrfullformat}[2]{%
6556     \firstacronymfont{\glentryshort{##1}}##2%
6557     \protect\footnote{\glentrylong{##1}}%
6558   }%
6559   \renewcommand*{\genplacrfullformat}[2]{%
6560     \protect\firstacronymfont{\glentryshortpl{##1}}##2%
6561     \protect\footnote{\glentrylongpl{##1}}%
6562   }%
6563   \renewcommand*{\Genplacrfullformat}[2]{%
6564     \protect\firstacronymfont{\glentryshortpl{##1}}##2%
6565     \protect\footnote{\glentrylongpl{##1}}%
6566   }%
6567   \renewcommand*{\acronymentry}[1]{\acronymfont{\glentryshort{##1}}}%
6568   \renewcommand*{\acronymsort}[2]{##1}%
6569   \renewcommand*{\acronymfont}[1]{##1}%
6570   \renewcommand*{\acrpluralsuffix}{\glacrpluralsuffix}%

```

Don't use footnotes for \acrfull:

```
6571   \renewcommand*{\acrfullfmt}[3]{%
6572     \glslink{##1}{##2}{\acronymfont{\glentryshort{##2}}##3\space
6573       (\glentrylong{##2})}%

```

```

6574 \renewcommand*{\Acrfullfmt}[3]{%
6575   \glslink[##1]{##2}{\acronymfont{\Glsentryshort{##2}}##3\space
6576   (\Glsentrylong{##2})}%
6577 \renewcommand*{\ACRfullfmt}[3]{%
6578   \glslink[##1]{##2}{%
6579     \mfirstucMakeUppercase{\acronymfont{\Glsentryshort{##2}}##3\space
6580     (\Glsentrylong{##2})}%
6581 \renewcommand*{\acrfullplfmt}[3]{%
6582   \glslink[##1]{##2}{\acronymfont{\Glsentryshortpl{##2}}##3\space
6583   (\Glsentrylongpl{##2})}%
6584 \renewcommand*{\Acrfullplfmt}[3]{%
6585   \glslink[##1]{##2}{\acronymfont{\Glsentryshortpl{##2}}##3\space
6586   (\Glsentrylongpl{##2})}%
6587 \renewcommand*{\ACRfullplfmt}[3]{%
6588   \glslink[##1]{##2}{%
6589     \mfirstucMakeUppercase{\acronymfont{\Glsentryshortpl{##2}}##3\space
6590     (\Glsentrylongpl{##2})}%

```

Similarly for \glsentryfull etc:

```

6591 \renewcommand*{\glsentryfull}[1]{%
6592   \acronymfont{\Glsentryshort{##1}}\space(\Glsentrylong{##1})}%
6593 \renewcommand*{\Glsentryfull}[1]{%
6594   \acronymfont{\Glsentryshort{##1}}\space(\Glsentrylong{##1})}%
6595 \renewcommand*{\glsentryfullpl}[1]{%
6596   \acronymfont{\Glsentryshortpl{##1}}\space(\Glsentrylongpl{##1})}%
6597 \renewcommand*{\Glsentryfullpl}[1]{%
6598   \acronymfont{\Glsentryshortpl{##1}}\space(\Glsentrylongpl{##1})}%
6599 }

```

footnote-sc \textsc{<short>}\footnote{<long>} acronym style.

```

6600 \newacronymstyle{footnote-sc}%
6601 {%
6602   \GlsUseAcrEntryDisplayStyle{footnote}%
6603 }%
6604 {%
6605   \GlsUseAcrStyleDefs{footnote}%
6606   \renewcommand{\acronymentry}[1]{\acronymfont{\Glsentryshort{##1}}}
6607   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6608   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6609 }%

```

footnote-sm \textsmaller{<short>}\footnote{<long>} acronym style.

```

6610 \newacronymstyle{footnote-sm}%
6611 {%
6612   \GlsUseAcrEntryDisplayStyle{footnote}%
6613 }%
6614 {%
6615   \GlsUseAcrStyleDefs{footnote}%
6616   \renewcommand{\acronymentry}[1]{\acronymfont{\Glsentryshort{##1}}}
6617   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%

```



```

6618 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6619 }%

```

footnote-desc *<short>*\footnote{*<long>*} acronym style that has an accompanying description (which the user needs to supply).

```

6620 \newacronymstyle{footnote-desc}%
6621 {%
6622   \GlsUseAcrEntryDisplayStyle{footnote}%
6623 }%
6624 {%
6625   \GlsUseAcrStyleDefs{footnote}%
6626   \renewcommand*{\GenericAcronymFields}{}%
6627   \renewcommand*{\acronymsort}[2]{##2}%
6628   \renewcommand*{\acronymentry}[1]{%
6629     \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6630 }

```

footnote-sc-desc \textsc{*<short>*}\footnote{*<long>*} acronym style that has an accompanying description (which the user needs to supply).

```

6631 \newacronymstyle{footnote-sc-desc}%
6632 {%
6633   \GlsUseAcrEntryDisplayStyle{footnote-sc}%
6634 }%
6635 {%
6636   \GlsUseAcrStyleDefs{footnote-sc}%
6637   \renewcommand*{\GenericAcronymFields}{}%
6638   \renewcommand*{\acronymsort}[2]{##2}%
6639   \renewcommand*{\acronymentry}[1]{%
6640     \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6641 }

```

footnote-sm-desc \textsmaller{*<short>*}\footnote{*<long>*} acronym style that has an accompanying description (which the user needs to supply).

```

6642 \newacronymstyle{footnote-sm-desc}%
6643 {%
6644   \GlsUseAcrEntryDisplayStyle{footnote-sm}%
6645 }%
6646 {%
6647   \GlsUseAcrStyleDefs{footnote-sm}%
6648   \renewcommand*{\GenericAcronymFields}{}%
6649   \renewcommand*{\acronymsort}[2]{##2}%
6650   \renewcommand*{\acronymentry}[1]{%
6651     \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6652 }

```

`\fineAcronymSynonyms`

```

6653 \newcommand*{\DefineAcronymSynonyms}{%

```

Short form

\acs

6654 \let\acs\acrshort

First letter uppercase short form

\Acs

6655 \let\Acs\Acrshort

Plural short form

\acsp

6656 \let\acsp\acrshortpl

First letter uppercase plural short form

\Acsp

6657 \let\Acsp\Acrshortpl

Long form

\acl

6658 \let\acl\acrlong

Plural long form

\aclp

6659 \let\aclp\acrlongpl

First letter upper case long form

\Acl

6660 \let\Acl\Acrlong

First letter upper case plural long form

\Aclp

6661 \let\Aclp\Acrlongpl

Full form

\acf

6662 \let\acf\acrfull

Plural full form

\acfp

6663 \let\acfp\acrfullpl

First letter upper case full form

\Acf

6664 \let\Acf\Acrfull

First letter upper case plural full form

`\Acfp`

```
6665 \let\Acfp\Acrfullpl
```

Standard form

`\ac`

```
6666 \let\ac\gls
```

First upper case standard form

`\Ac`

```
6667 \let\Ac\Gls
```

Standard plural form

`\acp`

```
6668 \let\acp\glspl
```

Standard first letter upper case plural form

`\Acp`

```
6669 \let\Acp\Glspl
```

```
6670 }
```

Define synonyms if required

```
6671 \ifglsacrshortcuts
```

```
6672 \DefineAcronymSynonyms
```

```
6673 \fi
```

These commands for setting the style are now deprecated but are kept for backward compatibility.

`AcronymDisplayStyle` Sets the default acronym display style for given glossary.

```
6674 \newcommand*\SetDefaultAcronymDisplayStyle}[1]{%
```

```
6675 \defglsentryfmt[#1]{\glsentryfmt}%
```

```
6676 }
```

`DefaultNewAcronymDef` Sets up the acronym definition for the default style. The information is provided by the tokens `\glslabeltok`, `\glsshorttok`, `\glslongtok` and `\glskeylisttok`.

```
6677 \newcommand*\DefaultNewAcronymDef{%
```

```
6678 \edef\@do@newglossaryentry{%
```

```
6679 \noexpand\newglossaryentry{\the\glslabeltok}%
```

```
6680 {%
```

```
6681 type=\acronymtype,%
```

```
6682 name={\the\glsshorttok},%
```

```
6683 sort={\the\glsshorttok},%
```

```
6684 text={\the\glsshorttok},%
```

```

6685 first={\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
6686 plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6687 firstplural={\acrfullformat{\noexpand\expandonce\noexpand\@glo@longpl}%
6688             {\noexpand\expandonce\noexpand\@glo@shortpl}},%
6689 short={\the\glsshorttok},%
6690 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6691 long={\the\glslongtok},%
6692 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6693 description={\the\glslongtok},%
6694 descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%

```

Remaining options specified by the user:

```

6695 \the\glskeylisttok
6696 }%
6697 }%
6698 \let\@org@gls@assign@firstpl\gls@assign@firstpl
6699 \let\@org@gls@assign@plural\gls@assign@plural
6700 \let\@org@gls@assign@descplural\gls@assign@descplural
6701 \def\gls@assign@firstpl##1##2{%
6702     \@@gls@expand@field{##1}{firstpl}{##2}%
6703 }%
6704 \def\gls@assign@plural##1##2{%
6705     \@@gls@expand@field{##1}{plural}{##2}%
6706 }%
6707 \def\gls@assign@descplural##1##2{%
6708     \@@gls@expand@field{##1}{descplural}{##2}%
6709 }%
6710 \do@newglossaryentry
6711 \let\gls@assign@firstpl\@org@gls@assign@firstpl
6712 \let\gls@assign@plural\@org@gls@assign@plural
6713 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
6714 }

```

DefaultAcronymStyle Set up the default acronym style:

```

6715 \newcommand*\SetDefaultAcronymStyle{%

```

Set the display style:

```

6716 \@for\@gls@type:=\@glsacronymlists\do{%
6717     \SetDefaultAcronymDisplayStyle{\@gls@type}%
6718 }%

```

Set up the definition of \newacronym:

```

6719 \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).

```

6720 \ifx\@glsacronymlists\@empty
6721     \def\@glo@type{\acronymtype}%
6722     \setkeys{glossentry}{##1}%
6723     \DeclareAcronymList{\@glo@type}%

```

```

6724     \SetDefaultAcronymDisplayStyle{\@glo@type}%
6725     \fi
6726     \glskeylisttok{##1}%
6727     \glslabeltok{##2}%
6728     \glsshorttok{##3}%
6729     \gslongtok{##4}%
6730     \newacronymhook
6731     \DefaultNewAcronymDef
6732 }%
6733 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6734 }

```

`\acrfootnote` Used by the footnote acronym styles.

```

6735 \newcommand*{\acrfootnote}[3]{\acrlinkfootnote{#1}{#2}{#3}}

```

`\acrlinkfootnote`

```

6736 \newcommand*{\acrlinkfootnote}[3]{%
6737   \footnote{\glslink[1]{#2}{#3}}%
6738 }

```

`\acrnoflinkfootnote`

```

6739 \newcommand*{\acrnoflinkfootnote}[3]{%
6740   \footnote{#3}%
6741 }

```

`AcronymDisplayStyle` Sets the acronym display style for given glossary for the description and footnote combination.

```

6742 \newcommand*{\SetDescriptionFootnoteAcronymDisplayStyle}[1]{%
6743   \def\glsentryfmt[1]{%

6744     \ifdefempty\glscustomtext
6745     {%
6746       \ifglsused{\glslabel}%
6747       {%
6748         \acronymfont{\glsentryfmt}%
6749       }%
6750     }%
6751     \firstacronymfont{\glsentryfmt}%
6752     \ifglsymbol{\glslabel}%
6753     {%
6754       \expandafter\protect\expandafter\acrfootnote\expandafter
6755       {\@gls@link@opts}{\@gls@link@label}%
6756     }%
6757     \glsifplural
6758       {\glsentrysymbolplural{\glslabel}}%
6759       {\glsentrysymbol{\glslabel}}%
6760     }%
6761   }%
6762 }%

```

```

6763 }%
6764 {\glscustomtext\glsinsert}%
6765 }%
6766 }

```

otnoteNewAcronymDef

```

6767 \newcommand*{\DescriptionFootnoteNewAcronymDef}{%
6768   \edef\@do@newglossaryentry{%
6769     \noexpand\newglossaryentry{\the\glslabeltok}%
6770     {%
6771       type=\acronymtype,%
6772       name={\noexpand\acronymfont{\the\glsshorttok}},%
6773       sort={\the\glsshorttok},%
6774       first={\the\glsshorttok},%
6775       firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6776       text={\the\glsshorttok},%
6777       plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6778       short={\the\glsshorttok},%
6779       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6780       long={\the\glslongtok},%
6781       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6782       symbol={\the\glslongtok},%
6783       symbolplural={\noexpand\expandonce\noexpand\@glo@longpl},%
6784       \the\glskeylisttok
6785     }%
6786   }%
6787   \let\@org@gls@assign@firstpl\gls@assign@firstpl
6788   \let\@org@gls@assign@plural\gls@assign@plural
6789   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
6790   \def\gls@assign@firstpl##1##2{%
6791     \@@gls@expand@field{##1}{firstpl}{##2}%
6792   }%
6793   \def\gls@assign@plural##1##2{%
6794     \@@gls@expand@field{##1}{plural}{##2}%
6795   }%
6796   \def\gls@assign@symbolplural##1##2{%
6797     \@@gls@expand@field{##1}{symbolplural}{##2}%
6798   }%
6799   \@do@newglossaryentry
6800   \let\gls@assign@plural\@org@gls@assign@plural
6801   \let\gls@assign@firstpl\@org@gls@assign@firstpl
6802   \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
6803 }

```

ootnoteAcronymStyle 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.

```

6804 \newcommand*{\SetDescriptionFootnoteAcronymStyle}{%

```

```

6805 \renewcommand{\newacronym}[4][\]{%
6806   \ifx\@glsacronymlists\@empty
6807     \def\@glo@type{\acronymtype}%
6808     \setkeys{glossentry}{##1}%
6809     \DeclareAcronymList{\@glo@type}%
6810     \SetDescriptionFootnoteAcronymDisplayStyle{\@glo@type}%
6811   \fi
6812   \glskeylisttok{##1}%
6813   \glslabeltok{##2}%
6814   \glsshorttok{##3}%
6815   \glslongtok{##4}%
6816   \newacronymhook
6817   \DescriptionFootnoteNewAcronymDef
6818 }%

```

If footnote package option is specified, set the first use to append the long form (stored in symbol) as a footnote.

```

6819 \@for\@gls@type:=\@glsacronymlists\do{%
6820   \SetDescriptionFootnoteAcronymDisplayStyle{\@gls@type}%
6821 }%

```

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.

```

6822 \ifglsacrsmallcaps
6823   \renewcommand*\acronymfont{[1]{\textsc{##1}}}%
6824   \renewcommand*\acrpluralsuffix{\glsupacrpluralsuffix}%
6825 \else
6826   \ifglsacrsmaller
6827     \renewcommand*\acronymfont{[1]{\textsmaller{##1}}}%
6828   \fi
6829 \fi

```

Check for package option clash

```

6830 \ifglsacrdua
6831   \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’
6832     can’t both be set}{}%
6833 \fi
6834 }%

```

AcronymDisplayStyle Sets the acronym display style for given glossary with description and dua combination.

```

6835 \newcommand*\SetDescriptionDUAAcronymDisplayStyle[1]{%
6836   \def\glsentryfmt[1]{\glsentryfmt}%
6837 }

```

ionDUANewAcronymDef

```

6838 \newcommand*\DescriptionDUANewAcronymDef{}%
6839 \edef\@do@newglossaryentry{%

```

```

6840 \noexpand\newglossaryentry{\the\glslabeltok}%
6841 {%
6842     type=\acronymtype,%
6843     name={\the\glslongtok},%
6844     sort={\the\glslongtok},
6845     text={\the\glslongtok},%
6846     first={\the\glslongtok},%
6847     plural={\noexpand\expandonce\noexpand\@glo@longpl},%
6848     firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
6849     short={\the\glsshorttok},%
6850     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6851     long={\the\glslongtok},%
6852     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6853     symbol={\the\glsshorttok},%
6854     symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6855     \the\glskeylisttok
6856 }%
6857 }%
6858 \let\@org@gls@assign@firstpl\gls@assign@firstpl
6859 \let\@org@gls@assign@plural\gls@assign@plural
6860 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
6861 \def\gls@assign@firstpl##1##2{%
6862     \@@gls@expand@field{##1}{firstpl}{##2}%
6863 }%
6864 \def\gls@assign@plural##1##2{%
6865     \@@gls@expand@field{##1}{plural}{##2}%
6866 }%
6867 \def\gls@assign@symbolplural##1##2{%
6868     \@@gls@expand@field{##1}{symbolplural}{##2}%
6869 }%
6870 \do@newglossaryentry
6871 \let\gls@assign@firstpl\@org@gls@assign@firstpl
6872 \let\gls@assign@plural\@org@gls@assign@plural
6873 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
6874 }

```

tionDUAAcronymStyle 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.

```

6875 \newcommand*{\SetDescriptionDUAAcronymStyle}{%
6876     \ifglscrsmlcaps
6877         \PackageError{glossaries}{Option clash: 'smallcaps' and 'dua'
6878             can't both be set}{}%
6879     \else
6880         \ifglscrsmlaller
6881             \PackageError{glossaries}{Option clash: 'smaller' and 'dua'
6882                 can't both be set}{}%
6883         \fi
6884     \fi

```



```

6885 \renewcommand{\newacronym}[4][]{%
6886 \ifx\@glsacronymlists\@empty
6887 \def\@glo@type{\acronymtype}%
6888 \setkeys{glossentry}{##1}%
6889 \DeclareAcronymList{\@glo@type}%
6890 \SetDescriptionDUAAcronymDisplayStyle{\@glo@type}%
6891 \fi
6892 \glskeylisttok{##1}%
6893 \glslabeltok{##2}%
6894 \glsshorttok{##3}%
6895 \glslongtok{##4}%
6896 \newacronymhook
6897 \DescriptionDUANewAcronymDef
6898 }%

Set display.
6899 \@for\@gls@type:=\@glsacronymlists\do{%
6900 \SetDescriptionDUAAcronymDisplayStyle{\@gls@type}%
6901 }%
6902 }%

```

AcronymDisplayStyle Sets the acronym display style for given glossary using the description setting (but not footnote or dua).

```

6903 \newcommand*{\SetDescriptionAcronymDisplayStyle}[1]{%
6904 \def\glsentryfmt[#1]{%

6905 \ifdefempty\glscustomtext
6906 {%
6907 \ifglsused{\glslabel}%
6908 {%

Move the inserted text outside of \acronymfont
6909 \let\gls@org@insert\glsinsert
6910 \let\glsinsert\@empty
6911 \acronymfont{\glsentryfmt}\gls@org@insert
6912 }%
6913 {%
6914 \glsentryfmt
6915 \ifglshassymbol{\glslabel}%
6916 {%
6917 \glsifplural
6918 {%
6919 \def\@glo@symbol{\glsentrysymbolplural{\glslabel}}%
6920 }%
6921 {%
6922 \def\@glo@symbol{\glsentrysymbol{\glslabel}}%
6923 }%
6924 \space\protect\firstacronymfont
6925 {\glsupcase
6926 {\@glo@symbol}

```

```

6927         {\@glo@symbol}
6928         {\mfirstucMakeUppercase{\@glo@symbol}}})%
6929     }%
6930     {}%
6931 }%
6932 }%
6933 {\glscustomtext\glsinsert}%
6934 }%
6935 }

```

ptionNewAcronymDef

```

6936 \newcommand*{\DescriptionNewAcronymDef}{%
6937   \edef\@do@newglossaryentry{%
6938     \noexpand\newglossaryentry{\the\glslabeltok}%
6939     {%
6940       type=\acronymtype,%
6941       name={\noexpand
6942         \acrnameformat{\the\glsshorttok}{\the\glslongtok}},%
6943       sort={\the\glsshorttok},%
6944       first={\the\glslongtok},%
6945       firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
6946       text={\the\glsshorttok},%
6947       plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6948       short={\the\glsshorttok},%
6949       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6950       long={\the\glslongtok},%
6951       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6952       symbol={\noexpand\@glo@text},%
6953       symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6954       \the\glskeylisttok}%
6955   }%
6956   \let\@org@gls@assign@firstpl\gls@assign@firstpl
6957   \let\@org@gls@assign@plural\gls@assign@plural
6958   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
6959   \def\gls@assign@firstpl##1##2{%
6960     \@@gls@expand@field{##1}{firstpl}{##2}%
6961   }%
6962   \def\gls@assign@plural##1##2{%
6963     \@@gls@expand@field{##1}{plural}{##2}%
6964   }%
6965   \def\gls@assign@symbolplural##1##2{%
6966     \@@gls@expand@field{##1}{symbolplural}{##2}%
6967   }%
6968   \@do@newglossaryentry
6969   \let\gls@assign@firstpl\@org@gls@assign@firstpl
6970   \let\gls@assign@plural\@org@gls@assign@plural
6971   \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
6972 }

```

riptionAcronymStyle Option description is used, but not dua or footnote. Store long form in

first key and short form in text and symbol key. The name is stored using `\acrnameformat` to allow the user to override the way the name is displayed in the list of acronyms.

```

6973 \newcommand*{\SetDescriptionAcronymStyle}{%
6974   \renewcommand{\newacronym}[4][\]{%
6975     \ifx\@glsacronymlists\@empty
6976       \def\@glo@type{\acronymtype}%
6977       \setkeys{glossentry}{##1}%
6978       \DeclareAcronymList{\@glo@type}%
6979       \SetDescriptionAcronymDisplayStyle{\@glo@type}%
6980     \fi
6981     \glskeylisttok{##1}%
6982     \glslabeltok{##2}%
6983     \glsshorttok{##3}%
6984     \gslongtok{##4}%
6985     \newacronymhook
6986     \DescriptionNewAcronymDef
6987   }%

```

Set display.

```

6988 \@for\@gls@type:=\@glsacronymlists\do{%
6989   \SetDescriptionAcronymDisplayStyle{\@gls@type}%
6990 }%

```

Redefine `\acronymfont` if small caps required. The plural suffix is set in an up-right font so that it remains in normal lower case, otherwise it looks as though it's part of the acronym.

```

6991 \ifglsacrsmallcaps
6992   \renewcommand{\acronymfont}[1]{\textsc{##1}}
6993   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6994 \else
6995   \ifglsacrsmaller
6996     \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
6997   \fi
6998 \fi
6999 }%

```

AcronymDisplayStyle Sets the acronym display style for given glossary with footnote setting (but not description or dua).

```

7000 \newcommand*{\SetFootnoteAcronymDisplayStyle}[1]{%
7001   \defglsentryfmt[#1]{%
7002     \ifdefempty\glscustomtext
7003     {%

```

Move the inserted text outside of `\acronymfont`

```

7004     \let\gls@org@insert\glsinsert
7005     \let\glsinsert\@empty
7006     \ifglsused{\glslabel}%
7007     {%

```

```

7008     \acronymfont{\glsgenentryfmt}\gls@org@insert
7009 }%
7010 {%
7011     \firstacronymfont{\glsgenentryfmt}\gls@org@insert
7012     \ifglshaslong{\glslabel}%
7013     {%
7014         \expandafter\protect\expandafter\acrfootnote\expandafter
7015         {\@gls@link@opts}{\@gls@link@label}%
7016         {%
7017             \glsifplural
7018             {\glsentrylongpl{\glslabel}}%
7019             {\glsentrylong{\glslabel}}%
7020         }%
7021     }%
7022 }%
7023 }%
7024 }%
7025 {\glscustomtext\glsinsert}%
7026 }%
7027 }

```

otnoteNewAcronymDef

```

7028 \newcommand*{\FootnoteNewAcronymDef}{%
7029     \edef\@do@newglossaryentry{%
7030         \noexpand\newglossaryentry{\the\glslabeltok}%
7031         {%
7032             type=\acronymtype,%
7033             name={\noexpand\acronymfont{\the\glsshorttok}},%
7034             sort={\the\glsshorttok},%
7035             text={\the\glsshorttok},%
7036             plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7037             first={\the\glsshorttok},%
7038             firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7039             short={\the\glsshorttok},%
7040             shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7041             long={\the\glslongtok},%
7042             longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7043             description={\the\glslongtok},%
7044             descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7045             \the\glskeylisttok
7046         }%
7047     }%
7048     \let\@org@gls@assign@plural\gls@assign@plural
7049     \let\@org@gls@assign@firstpl\gls@assign@firstpl
7050     \let\@org@gls@assign@descplural\gls@assign@descplural
7051     \def\gls@assign@firstpl##1##2{%
7052         \@gls@expand@field{##1}{firstpl}{##2}%
7053     }%
7054     \def\gls@assign@plural##1##2{%

```

```

7055 \@@gls@expand@field{##1}{plural}{##2}%
7056 }%
7057 \def\gls@assign@descplural##1##2{%
7058 \@@gls@expand@field{##1}{descplural}{##2}%
7059 }%
7060 \do@newglossaryentry
7061 \let\gls@assign@plural\@org@gls@assign@plural
7062 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7063 \let\gls@assign@descplural\@org@gls@assign@descplural
7064 }

```

footnoteAcronymStyle 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.

```

7065 \newcommand*\SetFootnoteAcronymStyle{%
7066 \renewcommand{\newacronym}[4][\]{%
7067 \ifx\@glsacronymlists\@empty
7068 \def\@glo@type{\acronymtype}%
7069 \setkeys{glossentry}{##1}%
7070 \DeclareAcronymList{\@glo@type}%
7071 \SetFootnoteAcronymDisplayStyle{\@glo@type}%
7072 \fi
7073 \glskeylisttok{##1}%
7074 \glslabeltok{##2}%
7075 \glsshorttok{##3}%
7076 \glslongtok{##4}%
7077 \newacronymhook
7078 \FootnoteNewAcronymDef
7079 }%

```

Set display

```

7080 \@for\@gls@type:=\@glsacronymlists\do{%
7081 \SetFootnoteAcronymDisplayStyle{\@gls@type}%
7082 }%

```

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.

```

7083 \ifglsacrsmallcaps
7084 \renewcommand*\acronymfont[1]{\textsc{##1}}%
7085 \renewcommand*\acrpluralsuffix{\glsupacrpluralsuffix}%
7086 \else
7087 \ifglsacrsmaller
7088 \renewcommand*\acronymfont[1]{\textsmaller{##1}}%
7089 \fi
7090 \fi

```

Check for option clash

```

7091 \ifglsacrdua
7092 \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’

```

```

7093     can't both be set}{}%
7094   \fi
7095 }%

```

glsdoparenifnotempty 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.

```

7096 \DeclareRobustCommand*\glsdoparenifnotempty}[2]{%
7097   \protected@edef\gls@tmp{#1}%
7098   \ifdefempty\gls@tmp
7099   {%
7100     {%
7101       \ifx\gls@tmp\@gls@default@value
7102       \else
7103         \space (#2{#1})%
7104       \fi
7105     }%
7106 }

```

AcronymDisplayStyle Sets the acronym display style for given glossary where neither footnote nor description is required, but smallcaps or smaller specified.

```

7107 \newcommand*\SetSmallAcronymDisplayStyle}[1]{%
7108   \def\glsentryfmt[#1]{%

7109     \ifdefempty\glscustomtext
7110     {%

```

Move the inserted text outside of `\acronymfont`

```

7111     \let\gls@org@insert\glsinsert
7112     \let\glsinsert\@empty
7113     \ifglsused{\glslabel}%
7114     {%
7115       \acronymfont{\glsentryfmt}\gls@org@insert
7116     }%
7117     {%
7118       \glsentryfmt
7119       \ifglshassymbol{\glslabel}%
7120       {%
7121         \glsifplural
7122         {%
7123           \def\@glo@symbol{\glsentrysymbolplural{\glslabel}}%
7124         }%
7125         {%
7126           \def\@glo@symbol{\glsentrysymbol{\glslabel}}%
7127         }%
7128         \space
7129         (\glsupercase
7130         {\firstacronymfont{\@glo@symbol}}%
7131         {\firstacronymfont{\@glo@symbol}}%

```

```

7132         {\firstacronymfont{\mfirstucMakeUppercase{\@glo@symbol}}})%
7133     }%
7134     {}%
7135     }%
7136     }%
7137     {\glscustomtext\glsinsert}%
7138 }%
7139 }

```

\SmallNewAcronymDef

```

7140 \newcommand*{\SmallNewAcronymDef}{%
7141     \edef\@do@newglossaryentry{%
7142         \noexpand\newglossaryentry{\the\glslabeltok}%
7143         {%
7144             type=\acronymtype,%
7145             name={\noexpand\acronymfont{\the\glsshorttok}},%
7146             sort={\the\glsshorttok},%
7147             text={\the\glsshorttok},%

```

Default to the short plural.

```

7148         plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7149         first={\the\glslongtok},%

```

Default to the long plural.

```

7150         firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7151         short={\the\glsshorttok},%
7152         shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7153         long={\the\glslongtok},%
7154         longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7155         description={\noexpand\@glo@first},%
7156         descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7157         symbol={\the\glsshorttok},%

```

Default to the short plural.

```

7158         symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7159         \the\glskeylisttok
7160     }%
7161 }%
7162 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7163 \let\@org@gls@assign@plural\gls@assign@plural
7164 \let\@org@gls@assign@descplural\gls@assign@descplural
7165 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7166 \def\gls@assign@firstpl##1##2{%
7167     \@@gls@expand@field{##1}{firstpl}{##2}%
7168 }%
7169 \def\gls@assign@plural##1##2{%
7170     \@@gls@expand@field{##1}{plural}{##2}%
7171 }%
7172 \def\gls@assign@descplural##1##2{%
7173     \@@gls@expand@field{##1}{descplural}{##2}%

```

```

7174 }%
7175 \def\gls@assign@symbolplural##1##2{%
7176   \@@gls@expand@field{##1}{symbolplural}{##2}%
7177 }%
7178 \do@newglossaryentry
7179 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7180 \let\gls@assign@plural\@org@gls@assign@plural
7181 \let\gls@assign@descplural\@org@gls@assign@descplural
7182 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7183 }

```

`\setSmallAcronymStyle` 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.

```

7184 \newcommand*\setSmallAcronymStyle{%
7185   \renewcommand{\newacronym}[4][]{%
7186     \ifx\@glsacronymlists\@empty
7187       \def\@glo@type{\acronymtype}%
7188       \setkeys{glossentry}{##1}%
7189       \DeclareAcronymList{\@glo@type}%
7190       \setSmallAcronymDisplayStyle{\@glo@type}%
7191     \fi
7192     \glskeylisttok{##1}%
7193     \glslabeltok{##2}%
7194     \glsshorttok{##3}%
7195     \glslongtok{##4}%
7196     \newacronymhook
7197     \SmallNewAcronymDef
7198   }%

```

Change the display since first only contains long form.

```

7199 \@for\@gls@type:=\@glsacronymlists\do{%
7200   \setSmallAcronymDisplayStyle{\@gls@type}%
7201 }%

```

Redefine `\acronymfont` if small caps required. The plural suffix is set in an up-right font so that it remains in normal lower case, otherwise it looks as though it's part of the acronym.

```

7202 \ifglsacrsmallcaps
7203   \renewcommand*\acronymfont}[1]{\textsc{##1}}
7204   \renewcommand*\acrpluralsuffix{\glsupacrpluralsuffix}%
7205 \else
7206   \renewcommand*\acronymfont}[1]{\textsmaller{##1}}
7207 \fi

```

check for option clash

```

7208 \ifglsacrdua
7209   \ifglsacrsmallcaps
7210     \PackageError{glossaries}{Option clash: ‘smallcaps’ and ‘dua’
7211       can’t both be set}{}%
7212   \else

```



```

7213     \PackageError{glossaries}{Option clash: ‘smaller’ and ‘dua’
7214     can’t both be set}{}%
7215     \fi
7216     \fi
7217 }%
```

`\SetDUADisplayStyle` Sets the acronym display style for given glossary with dua setting.

```

7218 \newcommand*\SetDUADisplayStyle[1]{%
7219   \def\glsentryfmt[#1]{\glsentryfmt}%
7220 }
```

`\DUANewAcronymDef`

```

7221 \newcommand*\DUANewAcronymDef{%
7222   \edef\@do@newglossaryentry{%
7223     \noexpand\newglossaryentry{\the\glslabeltok}%
7224     {%
7225       type=\acronymtype,%
7226       name={\the\glsshorttok},%
7227       text={\the\glslongtok},%
7228       first={\the\glslongtok},%
7229       plural={\noexpand\expandonce\noexpand\@glo@longpl},%
7230       firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7231       short={\the\glsshorttok},%
7232       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7233       long={\the\glslongtok},%
7234       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7235       description={\the\glslongtok},%
7236       descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7237       symbol={\the\glsshorttok},%
7238       symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7239       \the\glskeylisttok
7240     }%
7241   }%
7242   \let\@org@gls@assign@firstpl\gls@assign@firstpl
7243   \let\@org@gls@assign@plural\gls@assign@plural
7244   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7245   \let\@org@gls@assign@descplural\gls@assign@descplural
7246   \def\gls@assign@firstpl##1##2{%
7247     \@@gls@expand@field{##1}{firstpl}{##2}%
7248   }%
7249   \def\gls@assign@plural##1##2{%
7250     \@@gls@expand@field{##1}{plural}{##2}%
7251   }%
7252   \def\gls@assign@symbolplural##1##2{%
7253     \@@gls@expand@field{##1}{symbolplural}{##2}%
7254   }%
7255   \def\gls@assign@descplural##1##2{%
7256     \@@gls@expand@field{##1}{descplural}{##2}%
7257   }%
```

```

7258 \do@newglossaryentry
7259 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7260 \let\gls@assign@plural\@org@gls@assign@plural
7261 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7262 \let\gls@assign@descplural\@org@gls@assign@descplural
7263 }

```

`\SetDUASStyle` Always expand acronyms.

```

7264 \newcommand*{\SetDUASStyle}{%
7265   \renewcommand{\newacronym}[4][]{%
7266     \ifx\@glsacronymlists\@empty
7267       \def\@glo@type{\acronymtype}%
7268       \setkeys{glossentry}{##1}%
7269       \DeclareAcronymList{\@glo@type}%
7270       \SetDUADisplayStyle{\@glo@type}%
7271     \fi
7272     \glskeylisttok{##1}%
7273     \glslabeltok{##2}%
7274     \glsshorttok{##3}%
7275     \glslongtok{##4}%
7276     \newacronymhook
7277     \DUANewAcronymDef
7278   }%
7279   \SetDUADisplayStyle{\@gls@type}%
7280   \SetDUADisplayStyle{\@gls@type}%
7281 }%
7282 }

```

Set the display

`\SetAcronymStyle`

```

7283 \newcommand*{\SetAcronymStyle}{%
7284   \SetDefaultAcronymStyle
7285   \ifglsacrdescription
7286     \ifglsacrfootnote
7287       \SetDescriptionFootnoteAcronymStyle
7288     \else
7289       \ifglsacrdua
7290         \SetDescriptionDUAAcronymStyle
7291       \else
7292         \SetDescriptionAcronymStyle
7293       \fi
7294     \fi
7295   \else
7296     \ifglsacrfootnote
7297       \SetFootnoteAcronymStyle
7298     \else
7299       \ifthenelse{\boolean{glsacrsmalldcaps}}{OR
7300         \boolean{glsacrsmaller}}{}%
7301     }%

```

```

7302         \SetSmallAcronymStyle
7303     }%
7304     {%
7305         \ifglssacrdua
7306         \SetDUASstyle
7307         \fi
7308     }%
7309     \fi
7310     \fi
7311 }

```

Set the acronym style according to the package options

```

7312 \SetAcronymStyle

```

Allow user to define their own custom acronyms. (For compatibility with versions before v3.0, the short form is stored in the user1 key, the plural short form is stored in the user2 key, the long form is stored in the user3 key and the plural long form is stored in the user4 key.) Defaults to displaying only the acronym with the long form as the description.

SetCustomDisplayStyle Sets the acronym display style.

```

7313 \newcommand*{\SetCustomDisplayStyle}[1]{%
7314     \defglssentryfmt[#1]{\glsgenentryfmt}%
7315 }

```

CustomAcronymFields

```

7316 \newcommand*{\CustomAcronymFields}{%
7317     name={\the\glssshorttok},%
7318     description={\the\glslongtok},%
7319     first={\acrfullformat{\the\glslongtok}{\the\glssshorttok}},%
7320     firstplural={\acrfullformat
7321         {\noexpand\glssentrylongpl{\the\glslabeltok}}%
7322         {\noexpand\glssentryshortpl{\the\glslabeltok}}},%
7323     text={\the\glssshorttok},%
7324     plural={\the\glssshorttok\noexpand\acrpluralsuffix}%
7325 }

```

CustomNewAcronymDef

```

7326 \newcommand*{\CustomNewAcronymDef}{%
7327     \protected@edef\@do@newglossaryentry{%
7328         \noexpand\newglossaryentry{\the\glslabeltok}%
7329         {%
7330             type=\acronymtype,%
7331             short={\the\glssshorttok},%
7332             shortplural={\the\glssshorttok\noexpand\acrpluralsuffix},%
7333             long={\the\glslongtok},%
7334             longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7335             user1={\the\glssshorttok},%

```

```

7336     user2={\the\glsshorttok\noexpand\acrpluralsuffix},%
7337     user3={\the\glslongtok},%
7338     user4={\the\glslongtok\noexpand\acrpluralsuffix},%
7339     \CustomAcronymFields,%
7340     \the\glskeylisttok
7341   }%
7342 }%
7343 \do@newglossaryentry
7344 }

```

`\SetCustomStyle`

```

7345 \newcommand*{\SetCustomStyle}{%
7346   \renewcommand{\newacronym}[4][]{%
7347     \ifx\@glsacronymlists\@empty
7348       \def\@glo@type{\acronymtype}%
7349       \setkeys{glossentry}{##1}%
7350       \DeclareAcronymList{\@glo@type}%
7351       \SetCustomDisplayStyle{\@glo@type}%
7352     \fi
7353     \glskeylisttok{##1}%
7354     \glslabeltok{##2}%
7355     \glsshorttok{##3}%
7356     \glslongtok{##4}%
7357     \newacronymhook
7358     \CustomNewAcronymDef
7359   }%
7360   \@for\@gls@type:=\@glsacronymlists\do{%
7361     \SetCustomDisplayStyle{\@gls@type}%
7362   }%
7363 }

```

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.

```
7364 \RequirePackage{glossary-hypernav}
```

The styles that use list-like environments. These are not loaded if the `nolist` option is used:

```
7365 \@gls@loadlist
```

The styles that use the `longtable` environment. These are not loaded if the `no-long` package option is used.

```
7366 \@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.

```
7367 \@gls@loadsuper
```

The tree-like styles. These are not loaded if the `notree` package option is used.

```
7368 \@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.

```
7369 \ifx\@glossary@default@style\relax
```

```
7370 \else
```

```
7371   \setglossarystyle{\@glossary@default@style}
```

```
7372 \fi
```

1.20 Debugging Commands

`\showgloparent` `\showgloparent{<label>}`

```
7373 \newcommand*{\showgloparent}[1]{%
```

```
7374   \expandafter\show\csname glo@\glsdetoklabel{#1}@parent\endcsname
```

```
7375 }
```

`\showglolevel` `\showglolevel{<label>}`

```
7376 \newcommand*{\showglolevel}[1]{%
```

```
7377   \expandafter\show\csname glo@\glsdetoklabel{#1}@level\endcsname
```

```
7378 }
```

`\showglotext` `\showglotext{<label>}`

```
7379 \newcommand*{\showglotext}[1]{%
```

```
7380   \expandafter\show\csname glo@\glsdetoklabel{#1}@text\endcsname
```

```
7381 }
```

`\showgloplural` `\showgloplural{<label>}`

```
7382 \newcommand*{\showgloplural}[1]{%
```

```
7383   \expandafter\show\csname glo@\glsdetoklabel{#1}@plural\endcsname
```

```
7384 }
```

`\showglofirst` `\showglofirst{<label>}`

```
7385 \newcommand*{\showglofirst}[1]{%
7386   \expandafter\show\csname glo@\glsdetoklabel{#1}@first\endcsname
7387 }
```

`\showglofirstpl` `\showglofirstpl{<label>}`

```
7388 \newcommand*{\showglofirstpl}[1]{%
7389   \expandafter\show\csname glo@\glsdetoklabel{#1}@firstpl\endcsname
7390 }
```

`\showglotype` `\showglotype{<label>}`

```
7391 \newcommand*{\showglotype}[1]{%
7392   \expandafter\show\csname glo@\glsdetoklabel{#1}@type\endcsname
7393 }
```

`\showglocounter` `\showglocounter{<label>}`

```
7394 \newcommand*{\showglocounter}[1]{%
7395   \expandafter\show\csname glo@\glsdetoklabel{#1}@counter\endcsname
7396 }
```

`\showglouserii` `\showglouserii{<label>}`

```
7397 \newcommand*{\showglouserii}[1]{%
7398   \expandafter\show\csname glo@\glsdetoklabel{#1}@userii\endcsname
7399 }
```

`\showglouserii` `\showglouserii{<label>}`

```
7400 \newcommand*{\showglouserii}[1]{%
7401   \expandafter\show\csname glo@\glsdetoklabel{#1}@userii\endcsname
7402 }
```

\showglouseriii \showglouseriii{<label>}

```
7403 \newcommand*{\showglouseriii}[1]{%
7404   \expandafter\show\csname glo@\glsdetoklabel{#1}@useriii\endcsname
7405 }
```

\showglouseriv \showglouseriv{<label>}

```
7406 \newcommand*{\showglouseriv}[1]{%
7407   \expandafter\show\csname glo@\glsdetoklabel{#1}@useriv\endcsname
7408 }
```

\showglouserv \showglouserv{<label>}

```
7409 \newcommand*{\showglouserv}[1]{%
7410   \expandafter\show\csname glo@\glsdetoklabel{#1}@userv\endcsname
7411 }
```

\showglouservi \showglouservi{<label>}

```
7412 \newcommand*{\showglouservi}[1]{%
7413   \expandafter\show\csname glo@\glsdetoklabel{#1}@uservi\endcsname
7414 }
```

\showgloname \showgloname{<label>}

```
7415 \newcommand*{\showgloname}[1]{%
7416   \expandafter\show\csname glo@\glsdetoklabel{#1}@name\endcsname
7417 }
```

\showglodesc \showglodesc{<label>}

```
7418 \newcommand*{\showglodesc}[1]{%
7419   \expandafter\show\csname glo@\glsdetoklabel{#1}@desc\endcsname
7420 }
```

`\showglodescplural` `\showglodescplural{\label}`

```
7421 \newcommand*{\showglodescplural}[1]{%
7422   \expandafter\show\csname glo@\glsdetoklabel{#1}@descplural\endcsname
7423 }
```

`\showglosort` `\showglosort{\label}`

```
7424 \newcommand*{\showglosort}[1]{%
7425   \expandafter\show\csname glo@\glsdetoklabel{#1}@sort\endcsname
7426 }
```

`\showglosymbol` `\showglosymbol{\label}`

```
7427 \newcommand*{\showglosymbol}[1]{%
7428   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbol\endcsname
7429 }
```

`\showglosymbolplural` `\showglosymbolplural{\label}`

```
7430 \newcommand*{\showglosymbolplural}[1]{%
7431   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolplural\endcsname
7432 }
```

`\showgloshort` `\showgloshort{\label}`

```
7433 \newcommand*{\showgloshort}[1]{%
7434   \expandafter\show\csname glo@\glsdetoklabel{#1}@short\endcsname
7435 }
```

`\showglolong` `\showglolong{\label}`

```
7436 \newcommand*{\showglolong}[1]{%
7437   \expandafter\show\csname glo@\glsdetoklabel{#1}@long\endcsname
7438 }
```


`\showgloindex` `\showgloindex{<label>}`

```
7439 \newcommand*{\showgloindex}[1]{%
7440   \expandafter\show\csname glo@\glsdetoklabel{#1}@index\endcsname
7441 }
```

`\showgloflag` `\showgloflag{<label>}`

```
7442 \newcommand*{\showgloflag}[1]{%
7443   \expandafter\show\csname ifglo@\glsdetoklabel{#1}@flag\endcsname
7444 }
```

`\showgloloclist` `\showgloloclist{<label>}`

```
7445 \newcommand*{\showgloloclist}[1]{%
7446   \expandafter\show\csname glo@\glsdetoklabel{#1}@loclist\endcsname
7447 }
```

`\showglofield` `\showglofield{<label>}{<field>}`

```
7448 \newcommand*{\showglofield}[2]{%
7449   \csshow{glo@\glsdetoklabel{#1}@#2}%
7450 }
```

`\showacronymlists` `\showacronymlists`

Show list of glossaries that have been flagged as a list of acronyms.

```
7451 \newcommand*{\showacronymlists}{%
7452   \show\@glsacronymlists
7453 }
```

`\showglossaries` `\showglossaries`

Show list of defined glossaries.

```
7454 \newcommand*{\showglossaries}{%
7455   \show\@glo@types
7456 }
```

`\showglossaryin` `\showglossaryin{<glossary-label>}`

Show the ‘in’ extension for the given glossary.

```
7457 \newcommand*{\showglossaryin}[1]{%
7458   \expandafter\show\csname @glotype@#1@in\endcsname
7459 }
```

`\showglossaryout` `\showglossaryout{<glossary-label>}`

Show the ‘out’ extension for the given glossary.

```
7460 \newcommand*{\showglossaryout}[1]{%
7461   \expandafter\show\csname @glotype@#1@out\endcsname
7462 }
```

`\showglossarytitle` `\showglossarytitle{<glossary-label>}`

Show the title for the given glossary.

```
7463 \newcommand*{\showglossarytitle}[1]{%
7464   \expandafter\show\csname @glotype@#1@title\endcsname
7465 }
```

`\showglossarycounter` `\showglossarycounter{<glossary-label>}`

Show the counter for the given glossary.

```
7466 \newcommand*{\showglossarycounter}[1]{%
7467   \expandafter\show\csname @glotype@#1@counter\endcsname
7468 }
```

`\showglossaryentries` `\showglossaryentries{<glossary-label>}`

Show the list of entry labels for the given glossary.

```
7469 \newcommand*{\showglossaryentries}[1]{%
7470   \expandafter\show\csname glolist@#1\endcsname
7471 }
```

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.

```
7472 \csname ifglscompatible-2.07\endcsname
7473 \RequirePackage{glossaries-compatible-207}
7474 \fi
```

2 Prefix Support (glossaries-prefix Code)

This package provides a means of adding prefixes to your glossary entries. For example, you may want to use “a `\gls{<label>}`” on first use but use “an `\gls{<label>}`” on subsequent use.

```
7475 \NeedsTeXFormat{LaTeX2e}
7476 \ProvidesPackage{glossaries-prefix}[2015/11/22 v4.19 (NLCT)]
```

Pass all options to glossaries:

```
7477 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
7478 \ProcessOptions
```

Load glossaries:

```
7479 \RequirePackage{glossaries}
```

Add the new keys:

```
7480 \define@key{glossentry}{prefixfirst}{\def\@glo@entryprefixfirst{#1}}%
7481 \define@key{glossentry}{prefixfirstplural}{\def\@glo@entryprefixfirstplural{#1}}%
7482 \define@key{glossentry}{prefix}{\def\@glo@entryprefix{#1}}%
7483 \define@key{glossentry}{prefixplural}{\def\@glo@entryprefixplural{#1}}%
```

Add them to `\@gls@keymap`:

```
7484 \appto\@gls@keymap{,%
7485   {prefixfirst}{prefixfirst},%
7486   {prefixfirstplural}{prefixfirstplural},%
7487   {prefix}{prefix},%
7488   {prefixplural}{prefixplural}}%
7489 }
```

Set the default values:

```
7490 \appto\@newglossaryentryprehook{%
7491   \def\@glo@entryprefix{}%
7492   \def\@glo@entryprefixplural{}}%
```

```

7493 \let\@glo@entryprefixfirst\@gls@default@value
7494 \let\@glo@entryprefixfirstplural\@gls@default@value
7495 }

```

Set the assignment code:

```

7496 \appto\@newglossaryentryposthook{%
7497   \gls@assign@field{}\@glo@label}{prefix}\@glo@entryprefix}%
7498   \gls@assign@field{}\@glo@label}{prefixplural}\@glo@entryprefixplural}%

```

If prefixfirst has not been supplied, make it the same as prefix.

```

7499 \expandafter\gls@assign@field\expandafter
7500   {\csname glo@\@glo@label @prefix\endcsname}\@glo@label}{prefixfirst}%
7501   {\@glo@entryprefixfirst}%

```

If prefixfirstplural has not been supplied, make it the same as prefixplural.

```

7502 \expandafter\gls@assign@field\expandafter
7503   {\csname glo@\@glo@label @prefixplural\endcsname}\@glo@label}%
7504   {prefixfirstplural}\@glo@entryprefixfirstplural}%
7505 }

```

Define commands to access these fields:

glsentryprefixfirst

```

7506 \newcommand*\{glsentryprefixfirst}[1]{\csuse{glo@#1@prefixfirst}}

```

ryprefixfirstplural

```

7507 \newcommand*\{glsentryprefixfirstplural}[1]{\csuse{glo@#1@prefixfirstplural}}

```

\glsentryprefix

```

7508 \newcommand*\{glsentryprefix}[1]{\csuse{glo@#1@prefix}}

```

lsentryprefixplural

```

7509 \newcommand*\{glsentryprefixplural}[1]{\csuse{glo@#1@prefixplural}}

```

Now for the initial upper case variants:

Glsentryprefixfirst

```

7510 \newrobustcmd*\{Glsentryprefixfirst}[1]{%
7511   \protected@edef\@glo@text{\csname glo@#1@prefixfirst\endcsname}%
7512   \xmakefirstuc\@glo@text
7513 }

```

ryprefixfirstplural

```

7514 \newrobustcmd*\{Glsentryprefixfirstplural}[1]{%
7515   \protected@edef\@glo@text{\csname glo@#1@prefixfirstplural\endcsname}%
7516   \xmakefirstuc\@glo@text
7517 }

```

\Glsentryprefix

```
7518 \newrobustcmd*{\Glsentryprefix}[1]{%
7519   \protected@edef\@glo@text{\csname glo@#1@prefix\endcsname}%
7520   \xmakefirstuc\@glo@text
7521 }
```

\Glsentryprefixplural

```
7522 \newrobustcmd*{\Glsentryprefixplural}[1]{%
7523   \protected@edef\@glo@text{\csname glo@#1@prefixplural\endcsname}%
7524   \xmakefirstuc\@glo@text
7525 }
```

Define commands to determine if the prefix keys have been set:

\ifglshasprefix

```
7526 \newcommand*{\ifglshasprefix}[3]{%
7527   \ifcempty{glo@#1@prefix}%
7528   {#3}%
7529   {#2}%
7530 }
```

\ifglshasprefixplural

```
7531 \newcommand*{\ifglshasprefixplural}[3]{%
7532   \ifcempty{glo@#1@prefixplural}%
7533   {#3}%
7534   {#2}%
7535 }
```

\ifglshasprefixfirst

```
7536 \newcommand*{\ifglshasprefixfirst}[3]{%
7537   \ifcempty{glo@#1@prefixfirst}%
7538   {#3}%
7539   {#2}%
7540 }
```

\ifglshasprefixfirstplural

```
7541 \newcommand*{\ifglshasprefixfirstplural}[3]{%
7542   \ifcempty{glo@#1@prefixfirstplural}%
7543   {#3}%
7544   {#2}%
7545 }
```

Define commands that insert the prefix before commands like \gls:

\pgls

```
7546 \newrobustcmd{\pgls}{\@gls@hyp@opt\@pgls}
```

\@pgls Unstarred version.

```
7547 \newcommand*{\@pgls}[2] [] {%
7548   \new@ifnextchar [%
7549     {\@pgls@{#1}{#2}}%
7550     {\@pgls@{#1}{#2} []}%
7551 }
```

\@pgls@ Read in the final optional argument:

```
7552 \def\@pgls@#1#2[#3] {%
7553   \glsdoifexists{#2}%
7554   {%
7555     \ifglsused{#2}%
7556     {%
7557       \glsentryprefix{#2}%
7558     }%
7559     {%
7560       \glsentryprefixfirst{#2}%
7561     }%
7562     \@gls@{#1}{#2}[#3]%
7563   }%
7564 }
```

Similarly for the plural version:

\pglsp1

```
7565 \newrobustcmd{\pglsp1}{\@gls@hyp@opt\@pglsp1}
```

\@pglsp1 Unstarred version.

```
7566 \newcommand*{\@pglsp1}[2] [] {%
7567   \new@ifnextchar [%
7568     {\@pglsp1@{#1}{#2}}%
7569     {\@pglsp1@{#1}{#2} []}%
7570 }
```

\@pglsp1@ Read in the final optional argument:

```
7571 \def\@pglsp1@#1#2[#3] {%
7572   \glsdoifexists{#2}%
7573   {%
7574     \ifglsused{#2}%
7575     {%
7576       \glsentryprefixplural{#2}%
7577     }%
7578     {%
7579       \glsentryprefixfirstplural{#2}%
7580     }%
7581     \@glspl@{#1}{#2}[#3]%
7582   }%
7583 }
```

Now for the first letter upper case versions:

\Pgl's

```
7584 \newrobustcmd{\Pgl's}{\@gls@hyp@opt\@Pgl's}
```

\@Pgl's Unstarred version.

```
7585 \newcommand*{\@Pgl's}[2] [] {%
7586   \new@ifnextchar[%
7587     {\@Pgl's@{#1}{#2}}}%
7588   {\@Pgl's@{#1}{#2} []}%
7589 }
```

\@Pgl's@ Read in the final optional argument:

```
7590 \def\@Pgl's@#1#2[#3] {%
7591   \glsdoifexists{#2}%
7592   {%
7593     \ifglsused{#2}%
7594     {%
7595       \ifgls hasprefix{#2}%
7596       {%
7597         \Glsentryprefix{#2}%
7598         \@gls@{#1}{#2}[#3]%
7599       }%
7600       {\@Gls@{#1}{#2}[#3]}%
7601     }%
7602     {%
7603       \ifgls hasprefixfirst{#2}%
7604       {%
7605         \Glsentryprefixfirst{#2}%
7606         \@gls@{#1}{#2}[#3]%
7607       }%
7608       {\@Gls@{#1}{#2}[#3]}%
7609     }%
7610   }%
7611 }
```

Similarly for the plural version:

\Pgl'spl

```
7612 \newrobustcmd{\Pgl'spl}{\@gls@hyp@opt\@Pgl'spl}
```

\@Pgl'spl Unstarred version.

```
7613 \newcommand*{\@Pgl'spl}[2] [] {%
7614   \new@ifnextchar[%
7615     {\@Pgl'spl@{#1}{#2}}}%
7616   {\@Pgl'spl@{#1}{#2} []}%
7617 }
```

\@Pglsp1@ Read in the final optional argument:

```
7618 \def\@Pglsp1@#1#2[#3]{%
7619   \glsdoifexists{#2}%
7620   {%
7621     \ifglsused{#2}%
7622     {%
7623       \ifglshasprefixplural{#2}%
7624       {%
7625         \Glsentryprefixplural{#2}%
7626         \@glsp1@{#1}{#2}[#3]%
7627       }%
7628       {\@Glspl@{#1}{#2}[#3]}%
7629     }%
7630     {%
7631       \ifglshasprefixfirstplural{#2}%
7632       {%
7633         \Glsentryprefixfirstplural{#2}%
7634         \@glsp1@{#1}{#2}[#3]%
7635       }%
7636       {\@Glspl@{#1}{#2}[#3]}%
7637     }%
7638   }%
7639 }
```

Finally the all upper case versions:

\PGLS

```
7640 \newrobustcmd{\PGLS}{\@gls@hyp@opt\@PGLS}
```

\@PGLS Unstarred version.

```
7641 \newcommand*{\@PGLS}[2][ ]{%
7642   \new@ifnextchar[%
7643   {\@PGLS@{#1}{#2}}%
7644   {\@PGLS@{#1}{#2}[]}%
7645 }
```

\@PGLS@ Read in the final optional argument:

```
7646 \def\@PGLS@#1#2[#3]{%
7647   \glsdoifexists{#2}%
7648   {%
7649     \ifglsused{#2}%
7650     {%
7651       \mfirstucMakeUppercase{\glsentryprefix{#2}}%
7652     }%
7653     {%
7654       \mfirstucMakeUppercase{\glsentryprefixfirst{#2}}%
7655     }%
7656     \@GLS@{#1}{#2}[#3]%

```



```

7657 }%
7658 }

```

Plural version:

```

\PGLSpl
7659 \newrobustcmd{\PGLSpl}{\@gls@hyp@opt\@PGLSpl}

```

\@PGLSpl Unstarred version.

```

7660 \newcommand*{\@PGLSpl}[2][{}]{%
7661   \new@ifnextchar[%
7662     {\@PGLSpl@{#1}{#2}}}%
7663   {\@PGLSpl@{#1}{#2}[]}%
7664 }

```

\@PGLSpl@ Read in the final optional argument:

```

7665 \def\@PGLSpl@#1#2[#3]{%
7666   \glsdoifexists{#2}%
7667   {%
7668     \ifglsused{#2}%
7669     {%
7670       \mfirstucMakeUppercase{\glsentryprefixplural{#2}}%
7671     }%
7672     {%
7673       \mfirstucMakeUppercase{\glsentryprefixfirstplural{#2}}%
7674     }%
7675     \@GLSp1@{#1}{#2}[#3]%
7676   }%
7677 }

```

3 Glossary Styles

3.1 Glossary hyper-navigation definitions (glossary-hypernav package)

Package Definition:

```

7678 \ProvidesPackage{glossary-hypernav}[2015/11/22 v4.19 (NLCT)]

```

The commands defined in this package are provided to help navigate around the groups within a glossary (see [subsection 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`

```
7679 \newcommand*{\glsnavhyperlink}[3][\@glo@type]{%
7680   \edef\gls@grplabel{#2}\protected@edef\gls@grptitle{#3}%
7681   \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`

```
7682 \newcommand*{\glsnavhypertarget}[3][\@glo@type]{%
  Add this group to the aux file for re-run check.
7683   \protected@write\@auxout{}{\string\@gls@hypergroup{#1}{#2}}%
  Add the target.
7684   \@glstarget{glsn:#1@#2}{#3}%
  Check list of know groups to determine if a re-run is required.
7685   \expandafter\let
7686     \expandafter\@gls@list\csname @gls@hypergroup@list@#1\endcsname
  Iterate through list and terminate loop if this group is found.
7687   \@for\@gls@elem:=\@gls@list\do{%
7688     \ifthenelse{\equal{\@gls@elem}{#2}}{\@endfortrue}{}%
  Check if list terminated prematurely.
7689   \if@endfor
7690   \else
    This group was not included in the list, so issue a warning.
7691     \GlossariesWarningNoLine{Navigation panel
7692       for glossary type ‘#1’^^Jmissing group ‘#2’}%
7693     \gdef\gls@hypergroup@rerun{%
7694       \GlossariesWarningNoLine{Navigation panel
7695         has changed. Rerun LaTeX}}%
7696     \fi
7697 }
```

`\gls@hypergroup@rerun` Give a warning at the end if re-run required

```
7698 \let\gls@hypergroup@rerun\relax
7699 \AtEndDocument{\gls@hypergroup@rerun}
```

`\@gls@hypergroup` This adds to (or creates) the command `\@gls@hypergroup@list@<glossary type>` which lists all groups for a given glossary, so that the navigation bar only contains those groups that are present. However it requires at least 2 runs to ensure the information is up-to-date.

```
7700 \newcommand*{\@gls@hypergroup}[2]{%
```

```

7701 \ifundefined{@gls@hypergrouplist@#1}{%
7702   \expandafter\xdef\csname @gls@hypergrouplist@#1\endcsname{#2}%
7703 }{%
7704   \expandafter\let\expandafter@gls@tmp
7705     \csname @gls@hypergrouplist@#1\endcsname
7706   \expandafter\xdef\csname @gls@hypergrouplist@#1\endcsname{%
7707     \@gls@tmp,#2}%
7708 }%
7709 }

```

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`

```

7710 \newcommand*{\glsnavigation}{%
7711 \def\@gls@between{%
7712 \ifundefined{@gls@hypergrouplist@\@glo@type}{%
7713   \def\@gls@list{%
7714 }{%
7715   \expandafter\let\expandafter@gls@list
7716     \csname @gls@hypergrouplist@\@glo@type\endcsname
7717 }%
7718 \@for\@gls@tmp:=\@gls@list\do{%
7719   \@gls@between

7720   \@gls@getgrouptitle{\@gls@tmp}{\@gls@grptitle}%
7721   \glsnavhyperlink{\@gls@tmp}{\@gls@grptitle}%
7722   \let\@gls@between\glshypernavsep%
7723 }%
7724 }

```

`\glshypernavsep` Separator for the hyper navigation bar.

```

7725 \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`

```

7726 \newcommand*{\glssymbolnav}{%
7727 \glsnavhyperlink{glssymbols}{\glsgetgrouptitle{glssymbols}}%
7728 \glshypernavsep
7729 \glsnavhyperlink{glsnumbers}{\glsgetgrouptitle{glsnumbers}}%
7730 \glshypernavsep
7731 }

```

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.

7732 \ProvidesPackage{glossary-inline}[2015/11/22 v4.19 (NLCT)]

inline Define the inline style.

7733 \newglossarystyle{inline}{%

Start of glossary sets up first empty separator between entries. (This is then changed by \glossentry)

7734 \renewenvironment{theglossary}{%

7735 {%

7736 \def\gls@inlinesep{}%

7737 \def\gls@inlinesubsep{}%

7738 \def\gls@inlinepostchild{}%

7739 }%

7740 {\glspostinline}%

No header:

7741 \renewcommand*{\glossaryheader}{}%

No group headings (if heading is required, add \glsinlinedopostchild to start definition in case heading follows a child entry):

7742 \renewcommand*{\glsgroupheading}[1]{}%

Just display separator followed by name and description:

7743 \renewcommand{\glossentry}[2] {%

7744 \glsinlinedopostchild

7745 \gls@inlinesep

7746 \glsentryitem{##1}%

7747 \glsinlinenameformat{##1}{%

7748 \glossentryname{##1}%

7749 }%

7750 \ifglsdescsuppressed{##1}%

7751 {%

7752 \glsinlineemptydescformat

7753 {%

7754 \glossentrysymbol{##1}%

7755 }%

7756 {%

7757 ##2%

7758 }%

7759 }%

7760 {%

7761 \ifglshasdesc{##1}%

7762 {\glsinlinedescformat{\glossentrydesc{##1}}{\glossentrysymbol{##1}}{##2}}%

7763 {\glsinlineemptydescformat{\glossentrysymbol{##1}}{##2}}%

7764 }%

7765 \ifglshaschildren{##1}%

```

7766    {%
7767        \glsresetsubentrycounter
7768        \glsinlineparentchildseparator
7769        \def\gls@inlinesubsep{%
7770        \def\gls@inlinepostchild{\glsinlinepostchild}%
7771    }%
7772    {%
7773    \def\gls@inlinesep{\glsinlineseparator}%
7774    }%

```

Sub-entries display description:

```

7775    \renewcommand{\subglossentry}[3]{%
7776        \gls@inlinesubsep%
7777        \glsinlinesubnameformat{##2}{%
7778            \glossentryname{##2}}%
7779        \glssubentryitem{##2}%
7780        \glsinlinesubdescformat{\glossentrydesc{##2}}{\glossentrysymbol{##2}}{##3}%
7781        \def\gls@inlinesubsep{\glsinlinesubseparator}%
7782    }%

```

Nothing special between groups:

```

7783    \renewcommand*{\glsgroupskip}{}%
7784 }

```

glsinlinedopostchild

```

7785 \newcommand*{\glsinlinedopostchild}{%
7786     \gls@inlinepostchild
7787     \def\gls@inlinepostchild{%
7788 }

```

glsinlineseparator Separator to use between entries.

```

7789 \newcommand*{\glsinlineseparator}{;\space}

```

glsinlinesubseparator Separator to use between sub-entries.

```

7790 \newcommand*{\glsinlinesubseparator}{,\space}

```

glsinlineparentchildseparator Separator to use between parent and children.

```

7791 \newcommand*{\glsinlineparentchildseparator}{:\space}

```

glsinlinepostchild Hook to use between child and next entry

```

7792 \newcommand*{\glsinlinepostchild}{}

```

glspostinline Terminator for inline glossary.

```

7793 \newcommand*{\glspostinline}{\glspostdescription\space}

```

glsinlinenameformat Formats the name of the entry (first argument label, second argument name):

```

7794 \newcommand*{\glsinlinenameformat}[2]{\glstarget{#1}{#2}}

```

glsinlinedescformat Formats the entry's description, symbol and location list:

```

7795 \newcommand*{\glsinlinedescformat}[3]{\space#1}

```

`\lineemptydescformat` Formats the entry's symbol and location list when the description is empty:

```
7796 \newcommand*{\glsinlineemptydescformat}[2]{}
```

`\inlinesubnameformat` Formats the name of the subentry (first argument label, second argument name):

```
7797 \newcommand*{\glsinlinesubnameformat}[2]{\glstarget{#1}{}}
```

`\inlinesubdescformat` Formats the subentry's description, symbol and location list:

```
7798 \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.

```
7799 \ProvidesPackage{glossary-list}[2015/11/22 v4.19 (NLCT)]
```

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```
7800 \providecommand{\indexspace}{%
7801   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
7802 }
```

`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.

```
7803 \newglossarystyle{list}{%
```

Use description environment:

```
7804   \renewenvironment{theglossary}%
7805     {\begin{description}}{\end{description}}%
```

No header at the start of the environment:

```
7806   \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
7807   \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries start a new item in the list:

```
7808   \renewcommand*{\glossentry}[2]{%
7809     \item[\glsentryitem{##1}]%
7810       \glstarget{##1}{\glossentryname{##1}}]
7811     \glossentrydesc{##1}\glspostdescription\space ##2}%
```

Sub-entries continue on the same line:

```

7812 \renewcommand*{\subglossentry}[3]{%
7813   \glssubentryitem{##2}%
7814   \glstarget{##2}{\strut}%
7815   \glossentrydesc{##2}\glspostdescription\space ##3.}%
7816 %   \end{macrocode}
7817 % Add vertical space between groups:
7818 %\changes{3.03}{2012/09/21}{added check for glsnogroupskip}
7819 %   \begin{macrocode}
7820 \renewcommand*{\glsgroupskip}{\ifglsgroupskip\else\indexspace\fi}%
7821 }
```

listgroup The listgroup style is like the list style, but the glossary groups have headings.

```

7822 \newglossarystyle{listgroup}{%
  Base it on the list style:
7823   \setglossarystyle{list}%
  Each group has a heading:
7824   \renewcommand*{\glsgroupheading}[1]{\item[\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.

```

7825 \newglossarystyle{listhypergroup}{%
  Base it on the list style:
7826   \setglossarystyle{list}%
  Add navigation links at the start of the environment:
7827   \renewcommand*{\glossaryheader}{%
7828     \item[\glsnavigation]}%
  Each group has a heading with a hypertarget:
7829   \renewcommand*{\glsgroupheading}[1]{%
7830     \item[\glsnavigationhypertarget{##1}{\glsgrouptitle{##1}]}}}
```

altlist The altlist glossary style is like the list style, but places the description on a new line. Sub-entries follow in separate paragraphs without the sub-entry name. This style does not use the entry's symbol.

```

7831 \newglossarystyle{altlist}{%
  Base it on the list style:
7832   \setglossarystyle{list}%
  Main (level 0) entries start a new item in the list with a line break after the entry
  name:
7833   \renewcommand*{\glossentry}[2]{%
7834     \item[\glssentryitem{##1}%
7835       \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.

```
7836      \mbox{}\par\nobreak\@afterheading
7837      \glossentrydesc{##1}\glspostdescription\space ##2}%
```

Sub-entries start a new paragraph:

```
7838  \renewcommand{\subglossentry}[3]{%
7839    \par
7840    \glssubentryitem{##2}%
7841    \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space ##3}%
7842 }
```

`altlistgroup` The `altlistgroup` glossary style is like the `altlist` style, but the glossary groups have headings.

```
7843 \newglossarystyle{altlistgroup}{%
  Base it on the altlist style:
7844   \setglossarystyle{altlist}%
  Each group has a heading:
7845   \renewcommand*{\glsgroupheading}[1]{\item[\glsgrouptitle{##1}]}}
```

`altlisthypergroup` The `altlisthypergroup` glossary style is like the `altlistgroup` style, but has a set of links to the groups at the start of the glossary.

```
7846 \newglossarystyle{altlisthypergroup}{%
  Base it on the altlist style:
7847   \setglossarystyle{altlist}%
  Add navigation links at the start of the environment:
7848   \renewcommand*{\glossaryheader}{%
7849     \item[\glsnavigation]}%
  Each group has a heading with a hypertarget:
7850   \renewcommand*{\glsgroupheading}[1]{%
7851     \item[\glsnavigationhypertarget{##1}]{\glsgrouptitle{##1}}}}
```

`listdotted` The `listdotted` glossary style was supplied by Axel Menzel. I've modified it slightly so that the distance from the start of the name to the end of the dotted line is specified by `\glslistdottedwidth`. Note that this style ignores the page numbers as well as the symbol. Sub-entries are displayed in the same way as top-level entries.

```
7852 \newglossarystyle{listdotted}{%
  Base it on the list style:
7853   \setglossarystyle{list}%
```


Each main (level 0) entry starts a new item:

```

7854 \renewcommand*{\glossentry}[2]{%
7855   \item[\makebox[\glslistdottedwidth][l]{%
7856     \glentryitem{##1}%
7857     \glstarget{##1}{\glossentryname{##1}}}%
7858   \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##1}}%

```

Sub entries have the same format as main entries:

```

7859 \renewcommand*{\subglossentry}[3]{%
7860   \item[\makebox[\glslistdottedwidth][l]{%
7861     \glssubentryitem{##2}%
7862     \glstarget{##2}{\glossentryname{##2}}}%
7863   \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##2}}%
7864 }

```

`\glslistdottedwidth`

```

7865 \newlength\glslistdottedwidth
7866 \setlength{\glslistdottedwidth}{.5\hsize}

```

`sublistdotted` This style is similar to the `glostylelistdotted` style, except that the main entries just have the name displayed.

```
7867 \newglossarystyle{sublistdotted}{%
```

Base it on the `listdotted` style:

```
7868 \setglossarystyle{listdotted}%
```

Main (level 0) entries just display the name:

```

7869 \renewcommand*{\glossentry}[2]{%
7870   \item[\glentryitem{##1}\glstarget{##1}{\glossentryname{##1}}}%
7871 }

```

3.4 Glossary Styles using `longtable` (the `glossary-long` package)

The glossary styles defined in the package used the `longtable` environment in the glossary.

```
7872 \ProvidesPackage{glossary-long}[2015/11/22 v4.19 (NLCT)]
```

Requires the package:

```
7873 \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`.)

```

7874 \@ifundefined{glsdescwidth}{%
7875   \newlength\glsdescwidth
7876   \setlength{\glsdescwidth}{0.6\hsize}
7877 }{}

```

`\glspagelistwidth` This is a length that governs the width of the page list column.

```
7878 \@ifundefined{glspagelistwidth}{%  
7879   \newlength{glspagelistwidth}  
7880   \setlength{glspagelistwidth}{0.1\hsize}  
7881 }{}
```

`long` The long glossary style command which uses the longtable environment:

```
7882 \newglossarystyle{long}{%
```

Use longtable with two columns:

```
7883   \renewenvironment{theglossary}{%  
7884     {\begin{longtable}{lp{\glstdescwidth}}}%  
7885     {\end{longtable}}}%
```

Do nothing at the start of the environment:

```
7886   \renewcommand*{\glossaryheader}{}%
```

No heading between groups:

```
7887   \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries displayed in a row:

```
7888   \renewcommand{\glossentry}[2]{%  
7889     \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &  
7890     \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline  
7891   }%
```

Sub entries displayed on the following row without the name:

```
7892   \renewcommand{\subglossentry}[3]{%  
7893     &  
7894     \glssubentryitem{##2}%  
7895     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space  
7896     ##3\tabularnewline  
7897   }%
```

Blank row between groups:

```
7898   \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else &  
7899   \tabularnewline\fi}%  
7900 }
```

`longborder` The longborder style is like the above, but with horizontal and vertical lines:

```
7901 \newglossarystyle{longborder}{%
```

Base it on the glostylelong style:

```
7902   \setglossarystyle{long}%
```

Use longtable with two columns with vertical lines between each column:

```
7903   \renewenvironment{theglossary}{%  
7904     \begin{longtable}{|l|p{\glstdescwidth}|}{\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
7905   \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%  
7906 }
```

longheader The longheader style is like the long style but with a header:

```
7907 \newglossarystyle{longheader}{%
```

Base it on the `glostylelong` style:

```
7908 \setglossarystyle{long}%
```

Set the table's header:

```
7909 \renewcommand*{\glossaryheader}{%
```

```
7910 \bfseries \entryname & \bfseries \descriptionname\tabularnewline\endhead}%
```

```
7911 }
```

longheaderborder The longheaderborder style is like the long style but with a header and border:

```
7912 \newglossarystyle{longheaderborder}{%
```

Base it on the `glostylelongborder` style:

```
7913 \setglossarystyle{longborder}%
```

Set the table's header and add horizontal line to table's foot:

```
7914 \renewcommand*{\glossaryheader}{%
```

```
7915 \hline\bfseries \entryname & \bfseries
```

```
7916 \descriptionname\tabularnewline\hline
```

```
7917 \endhead
```

```
7918 \hline\endfoot}%
```

```
7919 }
```

long3col The long3col style is like long but with 3 columns

```
7920 \newglossarystyle{long3col}{%
```

Use a longtable with 3 columns:

```
7921 \renewenvironment{theglossary}%
```

```
7922 {\begin{longtable}{lp{\glstdescwidth}p{\glspagelistwidth}}}%
```

```
7923 {\end{longtable}}%
```

No table header:

```
7924 \renewcommand*{\glossaryheader}{}%
```

No headings between groups:

```
7925 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
7926 \renewcommand{\glossentry}[2]{%
```

```
7927 \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
```

```
7928 \glossentrydesc{##1} & ##2\tabularnewline
```

```
7929 }%
```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```
7930 \renewcommand{\subglossentry}[3]{%
```

```
7931 &
```

```
7932 \glssubentryitem{##2}%
```

```
7933 \glstarget{##2}{\strut}\glossentrydesc{##2} &
```

```

7934      ##3\tabularnewline
7935    }%

    Blank row between groups:
7936    \renewcommand*{\glsgroupskip}{%
7937      \ifglsgroupskip\else & &\tabularnewline\fi}%
7938  }

```

long3colborder The long3colborder style is like the long3col style but with a border:

```

7939 \newglossarystyle{long3colborder}{%

    Base it on the glostylelong3col style:
7940   \setglossarystyle{long3col}%

    Use a longtable with 3 columns with vertical lines around them:
7941   \renewenvironment{theglossary}%
7942     {\begin{longtable}{|l|p{\glsdescwidth}|p{\glspagelistwidth}|}%
7943     {\end{longtable}}}%

    Place horizontal lines at the head and foot of the table:
7944   \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
7945 }

```

long3colheader The long3colheader style is like long3col but with a header row:

```

7946 \newglossarystyle{long3colheader}{%

    Base it on the glostylelong3col style:
7947   \setglossarystyle{long3col}%

    Set the table's header:
7948   \renewcommand*{\glossaryheader}{%
7949     \bfseries\entryname&\bfseries\descriptionname&
7950     \bfseries\pagelistname\tabularnewline\endhead}%
7951 }

```

long3colheaderborder The long3colheaderborder style is like the above but with a border

```

7952 \newglossarystyle{long3colheaderborder}{%

    Base it on the glostylelong3colborder style:
7953   \setglossarystyle{long3colborder}%

    Set the table's header and add horizontal line at table's foot:
7954   \renewcommand*{\glossaryheader}{%
7955     \hline
7956     \bfseries\entryname&\bfseries\descriptionname&
7957     \bfseries\pagelistname\tabularnewline\hline\endhead
7958     \hline\endfoot}%
7959 }

```

long4col The long4col style has four columns where the third column contains the value of the associated symbol key.

```

7960 \newglossarystyle{long4col}{%

```

Use a longtable with 4 columns:

```
7961 \renewenvironment{theglossary}%  
7962 {\begin{longtable}{llll}}%  
7963 {\end{longtable}}%
```

No table header:

```
7964 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
7965 \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):

```
7966 \renewcommand{\glossentry}[2]{%  
7967 \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &  
7968 \glossentrydesc{##1} &  
7969 \glossentrysymbol{##1} &  
7970 ##2\tabularnewline  
7971 }%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
7972 \renewcommand{\subglossentry}[3]{%  
7973 &  
7974 \glssubentryitem{##2}%  
7975 \glstarget{##2}{\strut}\glossentrydesc{##2} &  
7976 \glossentrysymbol{##2} & ##3\tabularnewline  
7977 }%
```

Blank row between groups:

```
7978 \renewcommand*{\glsgroupskip}{}%  
7979 \ifglsgnognroupskip\else & & \tabularnewline\fi}%  
7980 }
```

long4colheader The long4colheader style is like long4col but with a header row.

```
7981 \newglossarystyle{long4colheader}{}%
```

Base it on the glostylelong4col style:

```
7982 \setglossarystyle{long4col}{}%
```

Table has a header:

```
7983 \renewcommand*{\glossaryheader}{}%  
7984 \bfseries\entryname&\bfseries\descriptionname&  
7985 \bfseries \symbolname&  
7986 \bfseries\pagelistname\tabularnewline\endhead}%  
7987 }
```

long4colborder The long4colborder style is like long4col but with a border.

```
7988 \newglossarystyle{long4colborder}{}%
```

Base it on the glostylelong4col style:

```
7989 \setglossarystyle{long4col}{}%
```

Use a longtable with 4 columns surrounded by vertical lines:

```
7990 \renewenvironment{theglossary}%  
7991 {\begin{longtable}{|l|l|l|l|}}%  
7992 {\end{longtable}}%
```

Add horizontal lines to the head and foot of the table:

```
7993 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%  
7994 }
```

long4colheaderborder The long4colheaderborder style is like the above but with a border.

```
7995 \newglossarystyle{long4colheaderborder}{%
```

Base it on the glostylelong4col style:

```
7996 \setglossarystyle{long4col}%
```

Use a longtable with 4 columns surrounded by vertical lines:

```
7997 \renewenvironment{theglossary}%  
7998 {\begin{longtable}{|l|l|l|l|}}%  
7999 {\end{longtable}}%
```

Add table header and horizontal line at the table's foot:

```
8000 \renewcommand*{\glossaryheader}{%  
8001 \hline\bfseries\entryname&\bfseries\descriptionname&  
8002 \bfseries \symbolname&  
8003 \bfseries\pagelistname\tabularnewline\hline\endhead  
8004 \hline\endfoot}%  
8005 }
```

altlong4col The altlong4col style is like the long4col style but can have multiline descriptions and page lists.

```
8006 \newglossarystyle{altlong4col}{%
```

Base it on the glostylelong4col style:

```
8007 \setglossarystyle{long4col}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8008 \renewenvironment{theglossary}%  
8009 {\begin{longtable}{lp{\glstdescwidth}lp{\glspagelistwidth}}}%  
8010 {\end{longtable}}%  
8011 }
```

altlong4colheader The altlong4colheader style is like altlong4col but with a header row.

```
8012 \newglossarystyle{altlong4colheader}{%
```

Base it on the glostylelong4colheader style:

```
8013 \setglossarystyle{long4colheader}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8014 \renewenvironment{theglossary}%
```

```

8015    {\begin{longtable}{lp{\glsdescwidth}lp{\glspagelistwidth}}}%
8016    {\end{longtable}}%
8017 }

```

`altlong4colborder` The `altlong4colborder` style is like `altlong4col` but with a border.

```

8018 \newglossarystyle{altlong4colborder}{%
    Base it on the glostylelong4colborder style:
8019    \setglossarystyle{long4colborder}%
    Use a longtable with 4 columns where the second and last columns may have
    multiple lines in each row:
8020    \renewenvironment{theglossary}%
8021        {\begin{longtable}{|lp{\glsdescwidth}|lp{\glspagelistwidth}|}}%
8022        {\end{longtable}}%
8023 }

```

`ong4colheaderborder` The `altlong4colheaderborder` style is like the above but with a header as well as a border.

```

8024 \newglossarystyle{altlong4colheaderborder}{%
    Base it on the glostylelong4colheaderborder style:
8025    \setglossarystyle{long4colheaderborder}%
    Use a longtable with 4 columns where the second and last columns may have
    multiple lines in each row:
8026    \renewenvironment{theglossary}%
8027        {\begin{longtable}{|lp{\glsdescwidth}|lp{\glspagelistwidth}|}}%
8028        {\end{longtable}}%
8029 }

```

3.5 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.

```

8030 \ProvidesPackage{glossary-longragged}[2015/11/22 v4.19 (NLCT)]

```

Requires the package:

```

8031 \RequirePackage{array}

```

Requires the package:

```

8032 \RequirePackage{longtable}

```

`\glsdescwidth` This is a length that governs the width of the description column. This may have already been defined.

```

8033 \@ifundefined{glsdescwidth}{%
8034     \newlength\glsdescwidth
8035     \setlength{\glsdescwidth}{0.6\hsize}
8036 }{}

```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```
8037 \@ifundefined{glspagelistwidth}{%
8038   \newlength{glspagelistwidth
8039   \setlength{glspagelistwidth}{0.1\hsize}
8040 }{}
```

`longragged` The `longragged` glossary style is like the `long` but uses ragged right formatting for the description column.

```
8041 \newglossarystyle{longragged}{%
```

Use `longtable` with two columns:

```
8042   \renewenvironment{theglossary}{%
8043     {\begin{longtable}{l>{\raggedright}p{glstdescwidth}}}%
8044     {\end{longtable}}}%
```

Do nothing at the start of the environment:

```
8045   \renewcommand*{\glossaryheader}{}%
```

No heading between groups:

```
8046   \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries displayed in a row:

```
8047   \renewcommand{\glossentry}[2]{%
8048     \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8049     \glossentrydesc{##1}\glspostdescription\space ##2%
8050     \tabularnewline
8051   }%
```

Sub entries displayed on the following row without the name:

```
8052   \renewcommand{\subglossentry}[3]{%
8053     &
8054     \glssubentryitem{##2}%
8055     \glstarget{##2}{\strut}\glossentrydesc{##2}%
8056     \glspostdescription\space ##3%
8057     \tabularnewline
8058   }%
```

Blank row between groups:

```
8059   \renewcommand*{\glsgroupskip}{\ifglsgnোগroupskip\else & \tabularnewline\fi}%
8060 }
```

`longraggedborder` The `longraggedborder` style is like the above, but with horizontal and vertical lines:

```
8061 \newglossarystyle{longraggedborder}{%
```

Base it on the `glostylelongragged` style:

```
8062   \setglossarystyle{longragged}%
```

Use `longtable` with two columns with vertical lines between each column:

```
8063   \renewenvironment{theglossary}{%
8064     \begin{longtable}{|l|>{\raggedright}p{glstdescwidth}|}%
8065     {\end{longtable}}%
```


Place horizontal lines at the head and foot of the table:

```
8066 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8067 }
```

`longraggedheader` The `longraggedheader` style is like the `longragged` style but with a header:

```
8068 \newglossarystyle{longraggedheader}{%
```

Base it on the `glostylelongragged` style:

```
8069 \setglossarystyle{longragged}%
```

Set the table's header:

```
8070 \renewcommand*{\glossaryheader}{%
8071 \bfseries \entryname & \bfseries \descriptionname
8072 \tabularnewline\endhead}%
8073 }
```

`longraggedheaderborder` The `longraggedheaderborder` style is like the `longragged` style but with a header and border:

```
8074 \newglossarystyle{longraggedheaderborder}{%
```

Base it on the `glostylelongraggedborder` style:

```
8075 \setglossarystyle{longraggedborder}%
```

Set the table's header and add horizontal line to table's foot:

```
8076 \renewcommand*{\glossaryheader}{%
8077 \hline\bfseries \entryname & \bfseries \descriptionname
8078 \tabularnewline\hline
8079 \endhead
8080 \hline\endfoot}%
8081 }
```

`longragged3col` The `longragged3col` style is like `longragged` but with 3 columns

```
8082 \newglossarystyle{longragged3col}{%
```

Use a longtable with 3 columns:

```
8083 \renewenvironment{theglossary}%
8084 {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}%
8085 >{\raggedright}p{\glspagelistwidth}}}%
8086 {\end{longtable}}%
```

No table header:

```
8087 \renewcommand*{\glossaryheader}{}%
```

No headings between groups:

```
8088 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
8089 \renewcommand{\glossentry}[2]{%
8090 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8091 \glossentrydesc{##1} & ##2\tabularnewline
8092 }%
```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```
8093 \renewcommand{\subglossentry}[3]{%
8094     &
8095     \glssubentryitem{##2}%
8096     \glstarget{##2}{\strut}\glossentrydesc{##2} &
8097     ##3\tabularnewline
8098 }
```

Blank row between groups:

```
8099 \renewcommand*{\glsgroupskip}{%
8100     \ifglsgnogroupskip\else & &\tabularnewline\fi}%
8101 }
```

longragged3colborder The longragged3colborder style is like the longragged3col style but with a border:

```
8102 \newglossarystyle{longragged3colborder}{%
```

Base it on the glostylelongragged3col style:

```
8103 \setglossarystyle{longragged3col}{%
```

Use a longtable with 3 columns with vertical lines around them:

```
8104 \renewenvironment{theglossary}{%
8105     {\begin{longtable}{|l|>{\raggedright}p{\glsgdescwidth}|%
8106      >{\raggedright}p{\glspagelistwidth}|}%
8107     {\end{longtable}}}%
```

Place horizontal lines at the head and foot of the table:

```
8108 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8109 }
```

longragged3colheader The longragged3colheader style is like longragged3col but with a header row:

```
8110 \newglossarystyle{longragged3colheader}{%
```

Base it on the glostylelongragged3col style:

```
8111 \setglossarystyle{longragged3col}{%
```

Set the table's header:

```
8112 \renewcommand*{\glossaryheader}{%
8113     \bfseries\entryname&\bfseries\descriptionname&
8114     \bfseries\pagelistname\tabularnewline\endhead}%
8115 }
```

longragged3colheaderborder The longragged3colheaderborder style is like the above but with a border

```
8116 \newglossarystyle{longragged3colheaderborder}{%
```

Base it on the glostylelongragged3colborder style:

```
8117 \setglossarystyle{longragged3colborder}{%
```

Set the table's header and add horizontal line at table's foot:

```

8118 \renewcommand*{\glossaryheader}{%
8119 \hline
8120 \bfseries\entryname&\bfseries\descriptionname&
8121 \bfseries\pagelistname\tabularnewline\hline\endhead
8122 \hline\endfoot}%
8123 }
```

`altlongragged4col` The `altlongragged4col` style is like the `altlong4col` style defined in the package, except that ragged right formatting is used for the description and page list columns.

```

8124 \newglossarystyle{altlongragged4col}{%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```

8125 \renewenvironment{theglossary}%
8126 {\begin{longtable}[1>{\raggedright}p{\glsdescwidth}1%
8127 >{\raggedright}p{\glspagelistwidth}}}%
8128 {\end{longtable}}}%

```

No table header:

```

8129 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```

8130 \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):

```

8131 \renewcommand{\glossentry}[2]{%
8132 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8133 \glossentrydesc{##1} & \glossentrysymbol{##1} &
8134 ##2\tabularnewline
8135 }%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```

8136 \renewcommand{\subglossentry}[3]{%
8137 &
8138 \glssubentryitem{##2}%
8139 \glstarget{##2}{\strut}\glossentrydesc{##2} &
8140 \glossentrysymbol{##2} & ##3\tabularnewline
8141 }%
```

Blank row between groups:

```

8142 \renewcommand*{\glsgroupskip}{%
8143 \ifglsgroupskip\else & & \tabularnewline\fi}%
8144 }
```

`ongragged4colheader` The `altlongragged4colheader` style is like `altlongragged4col` but with a header row.

```

8145 \newglossarystyle{altlongragged4colheader}{%
```

Base it on the `glostylealtlongragged4col` style:

```
8146 \setglossarystyle{altlongragged4col}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8147 \renewenvironment{theglossary}%
8148   {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}l%
8149     >{\raggedright}p{\glspagelistwidth}}}%
8150   {\end{longtable}}%
```

Table has a header:

```
8151 \renewcommand*{\glossaryheader}{%
8152   \bfseries\entryname&\bfseries\descriptionname&
8153   \bfseries \symbolname&
8154   \bfseries\pagelistname\tabularnewline\endhead}%
8155 }
```

`ongragged4colborder` The `altlongragged4colborder` style is like `altlongragged4col` but with a border.

```
8156 \newglossarystyle{altlongragged4colborder}{%
```

Base it on the `glostylealtlongragged4col` style:

```
8157 \setglossarystyle{altlongragged4col}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8158 \renewenvironment{theglossary}%
8159   {\begin{longtable}{|l|>{\raggedright}p{\glsgdescwidth}|l|%
8160     >{\raggedright}p{\glspagelistwidth}|}%
8161   {\end{longtable}}%
```

Add horizontal lines to the head and foot of the table:

```
8162 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8163 }
```

`ged4colheaderborder` The `altlongragged4colheaderborder` style is like the above but with a header as well as a border.

```
8164 \newglossarystyle{altlongragged4colheaderborder}{%
```

Base it on the `glostylealtlongragged4col` style:

```
8165 \setglossarystyle{altlongragged4col}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8166 \renewenvironment{theglossary}%
8167   {\begin{longtable}{|l|>{\raggedright}p{\glsgdescwidth}|l|%
8168     >{\raggedright}p{\glspagelistwidth}|}%
8169   {\end{longtable}}%
```

Add table header and horizontal line at the table's foot:

```
8170 \renewcommand*{\glossaryheader}{%
8171   \hline\bfseries\entryname&\bfseries\descriptionname&
```

```

8172 \bfseries \symbolname&
8173 \bfseries\pagelistname\tabularnewline\hline\endhead
8174 \hline\endfoot}%
8175 }

```

3.6 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.

```

8176 \ProvidesPackage{glossary-mcols}[2015/11/22 v4.19 (NLCT)]

```

Required packages:

```

8177 \RequirePackage{multicol}
8178 \RequirePackage{glossary-tree}

```

`\indexspace` The are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```

8179 \providecommand{\indexspace}{%
8180 \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
8181 }

```

`\glsmcols` Define macro in which to store the number of columns. (Defaults to 2.)

```

8182 \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.)

```

8183 \newglossarystyle{mcolindex}{%
8184 \setglossarystyle{index}%
8185 \renewenvironment{theglossary}%
8186 {%
8187 \begin{multicols}{\glsmcols}
8188 \setlength{\parindent}{0pt}%
8189 \setlength{\parskip}{0pt plus 0.3pt}%
8190 \let\item\@idxitem}%
8191 {\end{multicols}}%
8192 }

```

`mcolindexgroup` As `mcolindex` but has headings:

```

8193 \newglossarystyle{mcolindexgroup}{%
8194 \setglossarystyle{mcolindex}%
8195 \renewcommand*{\glsgroupheading}[1]{%
8196 \item\textbf{\glsgrouptitle{##1}}\indexspace}%
8197 }

```

`mcolindexhypergroup` The `mcolindexhypergroup` style is like the `mcolindexgroup` style but has hyper navigation.

```

8198 \newglossarystyle{mcolindexhypergroup}{%

```

Base it on the `glostylemcolindex` style:

```
8199 \setglossarystyle{mcolindex}%
```

Put navigation links to the groups at the start of the glossary:

```
8200 \renewcommand*{\glossaryheader}{%
8201 \item\textbf{\glsnavigation}\indexspace}%
```

Add a heading for each group (with a target). The group's title is in bold followed by a vertical gap.

```
8202 \renewcommand*{\glsgroupheading}[1]{%
8203 \item\textbf{\glsnavhypertarget{##1}\glsgetgrouptitle{##1}}}%
8204 \indexspace}%
8205 }
```

`mcoltree` Multi-column index style. Same as the tree, but puts the glossary in multiple columns.

```
8206 \newglossarystyle{mcoltree}{%
8207 \setglossarystyle{tree}%
8208 \renewenvironment{theglossary}%
8209 {%
8210 \begin{multicols}{\glsmcols}
8211 \setlength{\parindent}{0pt}%
8212 \setlength{\parskip}{0pt plus 0.3pt}%
8213 }%
8214 {\end{multicols}}}%
8215 }
```

`mcoltreegroup` Like the `mcoltree` style but the glossary groups have headings.

```
8216 \newglossarystyle{mcoltreegroup}{%
Base it on the glostylemcoltree style:
8217 \setglossarystyle{mcoltree}%
Each group has a heading (in bold) followed by a vertical gap):
8218 \renewcommand{\glsgroupheading}[1]{\par
8219 \noindent\textbf{\glsgetgrouptitle{##1}}\par\indexspace}%
8220 }
```

`mcoltreehypergroup` The `mcoltreehypergroup` style is like the `treegroup` style, but has a set of links to the groups at the start of the glossary.

```
8221 \newglossarystyle{mcoltreehypergroup}{%
Base it on the glostylemcoltree style:
8222 \setglossarystyle{mcoltree}%
Put navigation links to the groups at the start of the theglossary environment:
8223 \renewcommand*{\glossaryheader}{%
8224 \par\noindent\textbf{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
8225 \renewcommand*{\glsgroupheading}[1]{%
8226     \par\noindent
8227     \textbf{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8228     \indexspace}%
8229 }
```

mcoltreenoname Multi-column index style. Same as the **treenoname**, but puts the glossary in multiple columns.

```
8230 \newglossarystyle{mcoltreenoname}{%
8231     \setglossarystyle{treenoname}%
8232     \renewenvironment{theglossary}%
8233     {%
8234         \begin{multicols}{\glsmcols}
8235         \setlength{\parindent}{0pt}%
8236         \setlength{\parskip}{0pt plus 0.3pt}%
8237     }%
8238     {\end{multicols}}%
8239 }
```

mcoltreenonamegroup Like the **mcoltreenoname** style but the glossary groups have headings.

```
8240 \newglossarystyle{mcoltreenonamegroup}{%
8241     \setglossarystyle{mcoltreenoname}%
8242     \renewcommand{\glsgroupheading}[1]{\par
8243         \noindent\textbf{\glsgetgrouptitle{##1}}\par\indexspace}%
8244 }
```

treenonamehypergroup The **mcoltreenonamehypergroup** style is like the **mcoltreenonamegroup** style, but has a set of links to the groups at the start of the glossary.

```
8245 \newglossarystyle{mcoltreenonamehypergroup}{%
8246     \setglossarystyle{mcoltreenoname}%
8247     \renewcommand*{\glossaryheader}{%
8248         \par\noindent\textbf{\glsnavigation}\par\indexspace}%
8249     \renewcommand*{\glsgroupheading}[1]{%
8250         \par\noindent
8251         \textbf{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8252         \indexspace}%
8253 }
```

mcolalmtree Multi-column index style. Same as the **almtree**, but puts the glossary in multiple columns.

```

8254 \newglossarystyle{mcolalmtree}{%
8255   \setglossarystyle{almtree}%
8256   \renewenvironment{theglossary}%
8257   {%

8258     \begin{multicols}{\glsmcols}
8259     \def\@gls@prevlevel{-1}%
8260     \mbox{}\par
8261   }%
8262   {\par\end{multicols}}%
8263 }
```

mcolalmtreegroup Like the **mcolalmtree** style but the glossary groups have headings.

```

8264 \newglossarystyle{mcolalmtreegroup}{%

  Base it on the glostylemcolalmtree style:
8265   \setglossarystyle{mcolalmtree}%

  Give each group a heading.
8266   \renewcommand{\glsgroupheading}[1]{\par
8267     \def\@gls@prevlevel{-1}%
8268     \hangindent0pt\relax
8269     \parindent0pt\relax
8270     \textbf{\glsgetgrouptitle{##1}}\par\indexspace}%
8271 }
```

almtreehypergroup The **mcolalmtreehypergroup** style is like the **mcolalmtreegroup** style, but has a set of links to the groups at the start of the glossary.

```

8272 \newglossarystyle{mcolalmtreehypergroup}{%

  Base it on the glostylemcolalmtree style:
8273   \setglossarystyle{mcolalmtree}%

  Put the navigation links in the header
8274   \renewcommand*{\glossaryheader}{%
8275     \par
8276     \def\@gls@prevlevel{-1}%
8277     \hangindent0pt\relax
8278     \parindent0pt\relax
8279     \textbf{\glsnavigation}\par\indexspace}%

  Put a hypertarget at the start of each group
8280   \renewcommand*{\glsgroupheading}[1]{%
8281     \par
8282     \def\@gls@prevlevel{-1}%
8283     \hangindent0pt\relax
8284     \parindent0pt\relax
8285     \textbf{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8286     \indexspace}}
```


3.7 Glossary Styles using supertabular environment (glossary-super package)

The glossary styles defined in the package use the supertabular environment.

8287 \ProvidesPackage{glossary-super}[2015/11/22 v4.19 (NLCT)]

Requires the package:

8288 \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.

```
8289 \@ifundefined{glsdescwidth}{%
8290   \newlength\glsdescwidth
8291   \setlength{\glsdescwidth}{0.6\hsize}
8292 }{}
```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined if has been loaded.

```
8293 \@ifundefined{glspagelistwidth}{%
8294   \newlength\glspagelistwidth
8295   \setlength{\glspagelistwidth}{0.1\hsize}
8296 }{}
```

`super` The super glossary style uses the supertabular environment (it uses lengths defined in the package.)

8297 \newglossarystyle{super}{%

Put the glossary in a supertabular environment with two columns and no head or tail:

```
8298   \renewenvironment{theglossary}{%
8299     {\tablehead{}\tabletail{}}%
8300     \begin{supertabular}{lp{\glsdescwidth}}%
8301     {\end{supertabular}}%
```

Do nothing at the start of the table:

8302 \renewcommand*{\glossaryheader}{}%

No group headings:

8303 \renewcommand*{\glsgroupheading}[1]{}%

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```
8304 \renewcommand{\glossentry}[2]{%
8305   \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8306   \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline
8307 }%
```

Sub entries put in a row (no name, description and page list in second column):

8308 \renewcommand{\subglossentry}[3]{}%

```

8309      &
8310      \glssubentryitem{##2}%
8311      \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
8312      ##3\tabularnewline
8313    }%

```

Blank row between groups:

```

8314  \renewcommand*{\glsgroupskip}{%
8315    \ifglsgnোগroupskip\else & \tabularnewline\fi}%
8316 }

```

superborder The superborder style is like the above, but with horizontal and vertical lines:

```

8317 \newglossarystyle{superborder}{%

```

Base it on the glostylesuper style:

```

8318  \setglossarystyle{super}%

```

Put the glossary in a supertabular environment with two columns and a horizontal line in the head and tail:

```

8319  \renewenvironment{theglossary}%
8320    {\tablehead{\hline}\tabletail{\hline}%
8321     \begin{supertabular}{|l|p{\glsgdescwidth}|}%
8322     {\end{supertabular}}%
8323 }

```

superheader The superheader style is like the super style, but with a header:

```

8324 \newglossarystyle{superheader}{%

```

Base it on the glostylesuper style:

```

8325  \setglossarystyle{super}%

```

Put the glossary in a supertabular environment with two columns, a header and no tail:

```

8326 \renewenvironment{theglossary}%
8327   {\tablehead{\bfseries \entryname &
8328     \bfseries\descriptionname\tabularnewline}%
8329    \tabletail{}}%
8330   \begin{supertabular}{lp{\glsgdescwidth}}%
8331   {\end{supertabular}}%
8332 }

```

superheaderborder The superheaderborder style is like the super style but with a header and border:

```

8333 \newglossarystyle{superheaderborder}{%

```

Base it on the glostylesuper style:

```

8334  \setglossarystyle{super}%

```

Put the glossary in a supertabular environment with two columns, a header and horizontal lines above and below the table:

```

8335  \renewenvironment{theglossary}%
8336    {\tablehead{\hline\bfseries \entryname &

```

```

8337      \bfseries \descriptionname\tabularnewline\hline}%
8338      \tabletail{\hline}
8339      \begin{supertabular}{|l|p{\glstdescwidth}|}%
8340      {\end{supertabular}}%
8341 }

```

super3col The super3col style is like the super style, but with 3 columns:

```

8342 \newglossarystyle{super3col}{%
    Put the glossary in a supertabular environment with three columns and no head
    or tail:
8343     \renewenvironment{theglossary}%
8344     {\tablehead{}\tabletail{}}%
8345     \begin{supertabular}{lp{\glstdescwidth}p{\glspagelistwidth}}}%
8346     {\end{supertabular}}%
    Do nothing at the start of the table:
8347     \renewcommand*{\glossaryheader}{}%
    No group headings:
8348     \renewcommand*{\glsgroupheading}[1]{}%
    Main (level 0) entries on a row (name in first column, description in second
    column, page list in last column):
8349     \renewcommand{\glossentry}[2]{%
8350         \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8351         \glossentrydesc{##1} & ##2\tabularnewline
8352     }%
    Sub entries on a row (no name, description in second column, page list in last
    column):
8353     \renewcommand{\subglossentry}[3]{%
8354         &
8355         \glssubentryitem{##2}%
8356         \glstarget{##2}{\strut}\glossentrydesc{##2} &
8357         ##3\tabularnewline
8358     }%
    Blank row between groups:
8359     \renewcommand*{\glsgroupskip}{%
8360         \ifglsgnোগroupskip\else & &\tabularnewline\fi}%
8361 }

```

super3colborder The super3colborder style is like the super3col style, but with a border:

```

8362 \newglossarystyle{super3colborder}{%
    Base it on the glostylesuper3col style:
8363     \setglossarystyle{super3col}%
    Put the glossary in a supertabular environment with three columns and a hori-
    zontal line in the head and tail:
8364     \renewenvironment{theglossary}%

```

```

8365     {\tablehead{\hline}\tabletail{\hline}}%
8366     \begin{supertabular}{|l|p{\glsgdescwidth}|p{\glspagelistwidth}|}%
8367     {\end{supertabular}}}%
8368 }

```

super3colheader The `super3colheader` style is like the `super3col` style but with a header row:

```
8369 \newglossarystyle{super3colheader}{%
```

Base it on the `glostylesuper3col` style:

```
8370 \setglossarystyle{super3col}%
```

Put the glossary in a `supertabular` environment with three columns, a header and no tail:

```

8371 \renewenvironment{theglossary}%
8372     {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8373       \bfseries\pagelistname\tabularnewline}\tabletail{}}%
8374     \begin{supertabular}{lp{\glsgdescwidth}p{\glspagelistwidth}}}%
8375     {\end{supertabular}}}%
8376 }

```

super3colheaderborder The `super3colheaderborder` style is like the `super3col` style but with a header and border:

```
8377 \newglossarystyle{super3colheaderborder}{%
```

Base it on the `glostylesuper3colborder` style:

```
8378 \setglossarystyle{super3colborder}%
```

Put the glossary in a `supertabular` environment with three columns, a header with horizontal lines and a horizontal line in the tail:

```

8379 \renewenvironment{theglossary}%
8380     {\tablehead{\hline
8381       \bfseries\entryname&\bfseries\descriptionname&
8382       \bfseries\pagelistname\tabularnewline\hline}%
8383     \tabletail{\hline}%
8384     \begin{supertabular}{|l|p{\glsgdescwidth}|p{\glspagelistwidth}|}%
8385     {\end{supertabular}}}%
8386 }

```

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.

```
8387 \newglossarystyle{super4col}{%
```

Put the glossary in a `supertabular` environment with four columns and no head or tail:

```

8388 \renewenvironment{theglossary}%
8389     {\tablehead{}\tabletail{}}%
8390     \begin{supertabular}{llll}}}%
8391     \end{supertabular}}}%

```

Do nothing at the start of the table:

```
8392 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8393 \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:

```
8394 \renewcommand{\glossentry}[2]{%
8395   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8396   \glossentrydesc{##1} &
8397   \glossentrysymbol{##1} & ##3\tabularnewline
8398 }%
```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```
8399 \renewcommand{\subglossentry}[3]{%
8400   &
8401   \glssubentryitem{##2}%
8402   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8403   \glossentrysymbol{##2} & ##3\tabularnewline
8404 }%
```

Blank row between groups:

```
8405 \renewcommand*{\glsgroupskip}{%
8406   \ifglsnogroupskip\else & & \tabularnewline\fi}%
8407 }
```

super4colheader The super4colheader style is like the super4col but with a header row.

```
8408 \newglossarystyle{super4colheader}{%
```

Base it on the glostylesuper4col style:

```
8409 \setglossarystyle{super4col}%
```

Put the glossary in a supertabular environment with four columns, a header and no tail:

```
8410 \renewenvironment{theglossary}%
8411   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8412     \bfseries\symbolname &
8413     \bfseries\pagelistname\tabularnewline}%
8414   \tabletail{}}%
8415   \begin{supertabular}{llll}}%
8416   {\end{supertabular}}%
8417 }
```

super4colborder The super4colborder style is like the super4col but with a border.

```
8418 \newglossarystyle{super4colborder}{%
```

Base it on the glostylesuper4col style:

```
8419 \setglossarystyle{super4col}%
```

Put the glossary in a supertabular environment with four columns and a horizontal line in the head and tail:

```
8420 \renewenvironment{theglossary}%
```

```

8421    {\tablehead{\hline}\tabletail{\hline}}%
8422    \begin{supertabular}{|l|l|l|l|}%
8423    {\end{supertabular}}%
8424 }

```

per4colheaderborder The `super4colheaderborder` style is like the `super4col` but with a header and border.

```
8425 \newglossarystyle{super4colheaderborder}{%
```

Base it on the `glostypesuper4col` style:

```
8426 \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:

```

8427 \renewenvironment{theglossary}%
8428 {\tablehead{\hline\bfseries\entryname&\bfseries\descriptionname&
8429 \bfseries\symbolname &
8430 \bfseries\pagelistname\tabularnewline\hline}}%
8431 \tabletail{\hline}}%
8432 \begin{supertabular}{|l|l|l|l|}%
8433 {\end{supertabular}}%
8434 }

```

altsuper4col The `altsuper4col` glossary style is like `super4col` but has provision for multiline descriptions.

```
8435 \newglossarystyle{altsuper4col}{%
```

Base it on the `glostypesuper4col` style:

```
8436 \setglossarystyle{super4col}%
```

Put the glossary in a `supertabular` environment with four columns and no head or tail:

```

8437 \renewenvironment{theglossary}%
8438 {\tablehead{}\tabletail{}}%
8439 \begin{supertabular}{lp{\glstdescwidth}lp{\glspagelistwidth}}}%
8440 {\end{supertabular}}%
8441 }

```

altsuper4colheader The `altsuper4colheader` style is like the `altsuper4col` but with a header row.

```
8442 \newglossarystyle{altsuper4colheader}{%
```

Base it on the `glostypesuper4colheader` style:

```
8443 \setglossarystyle{super4colheader}%
```

Put the glossary in a `supertabular` environment with four columns, a header and no tail:

```

8444 \renewenvironment{theglossary}%
8445 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8446 \bfseries\symbolname &
8447 \bfseries\pagelistname\tabularnewline}\tabletail{}}%

```

```

8448     \begin{supertabular}{lp{\glstdescwidth}lp{\glspagelistwidth}}}%
8449     {\end{supertabular}}}%
8450 }

```

altsuper4colborder The altsuper4colborder style is like the altsuper4col but with a border.

```

8451 \newglossarystyle{altsuper4colborder}{%
    Base it on the glostylesuper4colborder style:
8452   \setglossarystyle{super4colborder}%
    Put the glossary in a supertabular environment with four columns and a horizontal line in the head and tail:
8453   \renewenvironment{theglossary}%
8454     {\tablehead{\hline}\tabletail{\hline}%
8455     \begin{supertabular}%
8456       {\lllp{\glstdescwidth}\lllp{\glspagelistwidth}}}%
8457     {\end{supertabular}}}%
8458 }

```

per4colheaderborder The altsuper4colheaderborder style is like the altsuper4col but with a header and border.

```

8459 \newglossarystyle{altsuper4colheaderborder}{%
    Base it on the glostylesuper4colheaderborder style:
8460   \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:
8461   \renewenvironment{theglossary}%
8462     {\tablehead{\hline
8463       \bfseries\entryname &
8464       \bfseries\descriptionname &
8465       \bfseries\symbolname &
8466       \bfseries\pagelistname\tabularnewline\hline}%
8467     \tabletail{\hline}%
8468     \begin{supertabular}%
8469       {\lllp{\glstdescwidth}\lllp{\glspagelistwidth}}}%
8470     {\end{supertabular}}}%
8471 }

```

3.8 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.

```

8472 \ProvidesPackage{glossary-superragged}[2015/11/22 v4.19 (NLCT)]

```

Requires the package:

```

8473 \RequirePackage{array}

```

Requires the package:

```
8474 \RequirePackage{supertabular}
```

`\glsdescwidth` This is a length that governs the width of the description column. This may already have been defined.

```
8475 \@ifundefined{glsdescwidth}{%
8476   \newlength{glsdescwidth
8477   \setlength{glsdescwidth}{0.6\hsize}
8478 }{}
```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```
8479 \@ifundefined{glspagelistwidth}{%
8480   \newlength{glspagelistwidth
8481   \setlength{glspagelistwidth}{0.1\hsize}
8482 }{}
```

`superragged` The superragged glossary style uses the supertabular environment.

```
8483 \newglossarystyle{superragged}{%
```

Put the glossary in a supertabular environment with two columns and no head or tail:

```
8484   \renewenvironment{theglossary}%
8485     {\tablehead{}\tabletail}%
8486     \begin{supertabular}{1>{\raggedright}p{glsdescwidth}}%
8487     {\end{supertabular}}%
```

Do nothing at the start of the table:

```
8488   \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8489   \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```
8490   \renewcommand{\glossentry}[2]{%
8491     \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8492     \glossentrydesc{##1}\glspostdescription\space ##2%
8493     \tabularnewline
8494   }%
```

Sub entries put in a row (no name, description and page list in second column):

```
8495   \renewcommand{\subglossentry}[3]{%
8496     &
8497     \glssubentryitem{##2}%
8498     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
8499     ##3%
8500     \tabularnewline
8501   }%
```


Blank row between groups:

```
8502 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else & \tabularnewline\fi}%  
8503 }
```

superraggedborder The superraggedborder style is like the above, but with horizontal and vertical lines:

```
8504 \newglossarystyle{superraggedborder}{%
```

Base it on the glostylesuperragged style:

```
8505 \setglossarystyle{superragged}%
```

Put the glossary in a supertabular environment with two columns and a horizontal line in the head and tail:

```
8506 \renewenvironment{theglossary}%  
8507 {\tablehead{\hline}\tabletail{\hline}%  
8508 \begin{supertabular}{|l|>\raggedright}p{\glsdescwidth}|}%  
8509 {\end{supertabular}}%  
8510 }
```

superraggedheader The superraggedheader style is like the super style, but with a header:

```
8511 \newglossarystyle{superraggedheader}{%
```

Base it on the glostylesuperragged style:

```
8512 \setglossarystyle{superragged}%
```

Put the glossary in a supertabular environment with two columns, a header and no tail:

```
8513 \renewenvironment{theglossary}%  
8514 {\tablehead{\bfseries \entryname & \bfseries \descriptionname  
8515 \tabularnewline}%  
8516 \tabletail{}}%  
8517 \begin{supertabular}{|l>\raggedright}p{\glsdescwidth}}}%  
8518 {\end{supertabular}}%  
8519 }
```

rraggedheaderborder The superraggedheaderborder style is like the superragged style but with a header and border:

```
8520 \newglossarystyle{superraggedheaderborder}{%
```

Base it on the glostylesuper style:

```
8521 \setglossarystyle{superragged}%
```

Put the glossary in a supertabular environment with two columns, a header and horizontal lines above and below the table:

```
8522 \renewenvironment{theglossary}%  
8523 {\tablehead{\hline\bfseries \entryname &  
8524 \bfseries \descriptionname\tabularnewline\hline}%  
8525 \tabletail{\hline}  
8526 \begin{supertabular}{|l|>\raggedright}p{\glsdescwidth}|}%  
8527 {\end{supertabular}}%  
8528 }
```

`superragged3col` The `superragged3col` style is like the `superragged` style, but with 3 columns:

```
8529 \newglossarystyle{superragged3col}{%
```

Put the glossary in a `supertabular` environment with three columns and no head or tail:

```
8530   \renewenvironment{theglossary}{%
8531     {\tablehead{}\tabletail{}}%
8532     \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}%
8533       >{\raggedright}p{\glspagelistwidth}}}%
8534     {\end{supertabular}}}%
```

Do nothing at the start of the table:

```
8535   \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8536   \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
8537   \renewcommand{\glossentry}[2]{%
8538     \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8539     \glossentrydesc{##1} &
8540     ##2\tabularnewline
8541   }%
```

Sub entries on a row (no name, description in second column, page list in last column):

```
8542   \renewcommand{\subglossentry}[3]{%
8543     &
8544     \glssubentryitem{##2}%
8545     \glstarget{##2}{\strut}\glossentrydesc{##2} &
8546     ##3\tabularnewline
8547   }%
```

Blank row between groups:

```
8548   \renewcommand*{\glsgroupskip}{\ifglsgnigroupskip\else & &\tabularnewline\fi}%
8549 }
```

`superragged3colborder` The `superragged3colborder` style is like the `superragged3col` style, but with a border:

```
8550 \newglossarystyle{superragged3colborder}{%
```

Base it on the `glostylessuperragged3col` style:

```
8551   \setglossarystyle{superragged3col}%
```

Put the glossary in a `supertabular` environment with three columns and a horizontal line in the head and tail:

```
8552   \renewenvironment{theglossary}{%
8553     {\tablehead{\hline}\tabletail{\hline}%
8554     \begin{supertabular}{|l|>{\raggedright}p{\glsgdescwidth}|%
8555       >{\raggedright}p{\glspagelistwidth}|}}%
```

```

8556   {\end{supertabular}}}%
8557 }

```

`superragged3colheader` The `superragged3colheader` style is like the `superragged3col` style but with a header row:

```

8558 \newglossarystyle{superragged3colheader}{%

```

Base it on the `glostypesuperragged3col` style:

```

8559   \setglossarystyle{superragged3col}%

```

Put the glossary in a `supertabular` environment with three columns, a header and no tail:

```

8560   \renewenvironment{theglossary}%
8561   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8562     \bfseries\pagelistname\tabularnewline}\tabletail{}}%
8563   \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}%
8564     >{\raggedright}p{\glspagelistwidth}}}%
8565   {\end{supertabular}}}%
8566 }

```

`ght3colheaderborder` The `superragged3colheaderborder` style is like the `superragged3col` style but with a header and border:

```

8567 \newglossarystyle{superragged3colheaderborder}{%

```

Base it on the `glostypesuperragged3colborder` style:

```

8568   \setglossarystyle{superragged3colborder}%

```

Put the glossary in a `supertabular` environment with three columns, a header with horizontal lines and a horizontal line in the tail:

```

8569   \renewenvironment{theglossary}%
8570   {\tablehead{\hline
8571     \bfseries\entryname&\bfseries\descriptionname&
8572     \bfseries\pagelistname\tabularnewline\hline}%
8573   \tabletail{\hline}%
8574   \begin{supertabular}{l|l>{\raggedright}p{\glsgdescwidth}|%
8575     >{\raggedright}p{\glspagelistwidth}|}%
8576   {\end{supertabular}}}%
8577 }

```

`altsuperragged4col` The `altsuperragged4col` glossary style is like `altsuper4col` style in the package but uses ragged right formatting in the description and page list columns.

```

8578 \newglossarystyle{altsuperragged4col}{%

```

Put the glossary in a `supertabular` environment with four columns and no head or tail:

```

8579   \renewenvironment{theglossary}%
8580   {\tablehead{}\tabletail{}}%
8581   \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}l%
8582     >{\raggedright}p{\glspagelistwidth}}}%
8583   {\end{supertabular}}}%

```

Do nothing at the start of the table:

```
8584 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8585 \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:

```
8586 \renewcommand{\glossentry}[2]{%
8587   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8588   \glossentrydesc{##1} &
8589   \glossentrysymbol{##1} & ##2\tabularnewline
8590 }
```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```
8591 \renewcommand{\subglossentry}[3]{%
8592   &
8593   \glssubentryitem{##2}%
8594   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8595   \glossentrysymbol{##2} & ##3\tabularnewline
8596 }
```

Blank row between groups:

```
8597 \renewcommand*{\glsgroupskip}{\ifglsgnোগroupskip\else & & \tabularnewline\fi}%
8598 }
```

perragged4colheader The `altsuperragged4colheader` style is like the `altsuperragged4col` style but with a header row.

```
8599 \newglossarystyle{altsuperragged4colheader}{%
```

Base it on the `glostylealtsuperragged4col` style:

```
8600 \setglossarystyle{altsuperragged4col}%
```

Put the glossary in a supertabular environment with four columns, a header and no tail:

```
8601 \renewenvironment{theglossary}%
8602   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8603     \bfseries\symbolname &
8604     \bfseries\pagelistname\tabularnewline}\tabletail{}}%
8605   \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}l%
8606     >{\raggedright}p{\glspagelistwidth}}}%
8607   {\end{supertabular}}%
8608 }
```

perragged4colborder The `altsuperragged4colborder` style is like the `altsuperragged4col` style but with a border.

```
8609 \newglossarystyle{altsuperragged4colborder}{%
```

Base it on the `glostylealtsuperragged4col` style:

```
8610 \setglossarystyle{altsuper4col}%
```

Put the glossary in a supertabular environment with four columns and a horizontal line in the head and tail:

```

8611 \renewenvironment{theglossary}%
8612   {\tablehead{\hline}\tabletail{\hline}%
8613   \begin{supertabular}%
8614     {\|l|>{\raggedright}p{\glsdescwidth}|l|}%
8615     >{\raggedright}p{\glspagelistwidth}|}}%
8616   {\end{supertabular}}%
8617 }

```

`ged4colheaderborder` The `altsuperragged4colheaderborder` style is like the `altsuperragged4col` style but with a header and border.

```

8618 \newglossarystyle{altsuperragged4colheaderborder}{%

```

Base it on the `glostylealtsuperragged4col` style:

```

8619 \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:

```

8620 \renewenvironment{theglossary}%
8621   {\tablehead{\hline
8622     \bfseries\entryname &
8623     \bfseries\descriptionname &
8624     \bfseries\symbolname &
8625     \bfseries\pagelistname\tabularnewline\hline}%
8626   \tabletail{\hline}%
8627   \begin{supertabular}%
8628     {\|l|>{\raggedright}p{\glsdescwidth}|l|}%
8629     >{\raggedright}p{\glspagelistwidth}|}}%
8630   {\end{supertabular}}%
8631 }

```

3.9 Tree Styles (`glossary-tree.sty`)

The style file defines glossary styles that have a tree-like structure. These are designed for hierarchical glossaries.

```

8632 \ProvidesPackage{glossary-tree}[2015/11/22 v4.19 (NLCT)]

```

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```

8633 \providecommand{\indexspace}{%
8634   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
8635 }

```

`\glstreenamefmt` Format used to display the name in the tree styles. (This may be counteracted by `\glstnamefont`.) This command is also used to format the group headings.

```

8636 \newcommand*{\glstreenamefmt}[1]{\textbf{#1}}

```

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.

```
8637 \newglossarystyle{index}{%
```

Set the paragraph indentation and skip and define `\item` to be the same as that used by theindex:

```
8638 \renewenvironment{theglossary}{%
8639   {\setlength{\parindent}{0pt}%
8640    \setlength{\parskip}{0pt plus 0.3pt}%
8641    \let\item\@idxitem}%
8642   {\par}%
8643   \renewcommand*{\glossaryheader}{}}%
8644   \renewcommand*{\glsgroupheading}[1]{}%
8645   \renewcommand*{\glossentry}[2]{%
8646     \item\glstentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
8647     \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
8648     \space \glossentrydesc{##1}\glspostdescription\space ##2%
8649   }%
8650   \renewcommand{\subglossentry}[3]{%
8651     \ifcase##1\relax
8652       % level 0
8653       \item
8654     \or
8655       % level 1
8656       \subitem
8657     \glssubentryitem{##2}%
8658   \else
8659     % all other levels
8660     \subsubitem
8661   \fi
8662   \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
8663   \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
8664   \space\glossentrydesc{##2}\glspostdescription\space ##3%
8665 }
```

Do nothing at the start of the environment:

```
8643 \renewcommand*{\glossaryheader}{}%
```

No group headers:

```
8644 \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.

```
8645 \renewcommand*{\glossentry}[2]{%
8646   \item\glstentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
8647   \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
8648   \space \glossentrydesc{##1}\glspostdescription\space ##2%
8649 }
```

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.

```
8650 \renewcommand{\subglossentry}[3]{%
8651   \ifcase##1\relax
8652     % level 0
8653     \item
8654   \or
8655     % level 1
8656     \subitem
8657   \glssubentryitem{##2}%
8658 \else
8659   % all other levels
8660   \subsubitem
8661 \fi
8662 \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
8663 \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
8664 \space\glossentrydesc{##2}\glspostdescription\space ##3%
8665 }
```

Vertical gap between groups is the same as that used by indices:

```
8666 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}}
```

indexgroup The `indexgroup` style is like the `index` style but has headings.

```
8667 \newglossarystyle{indexgroup}{%
```

Base it on the `glostyleindex` style:

```
8668 \setglossarystyle{index}%
```

Add a heading for each group. This puts the group's title in bold followed by a vertical gap.

```
8669 \renewcommand*{\glsgroupheading}[1]{%
```

```
8670 \item\glstreenamefmt{\glsgetgrouptitle{##1}}\indexspace}%
```

```
8671 }
```

indexhypergroup The `indexhypergroup` style is like the `indexgroup` style but has hyper navigation.

```
8672 \newglossarystyle{indexhypergroup}{%
```

Base it on the `glostyleindex` style:

```
8673 \setglossarystyle{index}%
```

Put navigation links to the groups at the start of the glossary:

```
8674 \renewcommand*{\glossaryheader}{%
```

```
8675 \item\glstreenamefmt{\glsnavigation}\indexspace}%
```

Add a heading for each group (with a target). The group's title is in bold followed by a vertical gap.

```
8676 \renewcommand*{\glsgroupheading}[1]{%
```

```
8677 \item\glstreenamefmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
```

```
8678 \indexspace}%
```

```
8679 }
```

tree The `tree` glossary style is similar in style to the `index` style, but can have arbitrary levels.

```
8680 \newglossarystyle{tree}{%
```

Set the paragraph indentation and skip:

```
8681 \renewenvironment{theglossary}%
```

```
8682 {\setlength{\parindent}{0pt}%
```

```
8683 \setlength{\parskip}{0pt plus 0.3pt}}%
```

```
8684 {}%
```

Do nothing at the start of the `theglossary` environment:

```
8685 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8686 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries: name in bold, followed by symbol in brackets (if it exists), the description and the page list:

```
8687 \renewcommand{\glossentry}[2]{%
```

```
8688 \hangindent0pt\relax
```

```

8689 \parindent0pt\relax
8690 \glstentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
8691 \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
8692 \space\glossentrydesc{##1}\glspostdescription\space##2\par
8693 }%

```

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.

```

8694 \renewcommand{\subglossentry}[3]{%
8695 \hangindent##1\glstreeindent\relax
8696 \parindent##1\glstreeindent\relax
8697 \ifnum##1=1\relax
8698 \glssubentryitem{##2}%
8699 \fi
8700 \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
8701 \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
8702 \space\glossentrydesc{##2}\glspostdescription\space ##3\par
8703 }%

```

Vertical gap between groups is the same as that used by indices:

```

8704 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}

```

treegroup Like the tree style but the glossary groups have headings.

```

8705 \newglossarystyle{treegroup}{%

```

Base it on the `glostyletree` style:

```

8706 \setglossarystyle{tree}%

```

Each group has a heading (in bold) followed by a vertical gap):

```

8707 \renewcommand{\glsgroupheading}[1]{\par
8708 \noindent\glstreenamefmt{\glsgrouptitle{##1}}\par\indexspace}%
8709 }

```

treehypergroup The `treehypergroup` style is like the `treegroup` style, but has a set of links to the groups at the start of the glossary.

```

8710 \newglossarystyle{treehypergroup}{%

```

Base it on the `glostyletree` style:

```

8711 \setglossarystyle{tree}%

```

Put navigation links to the groups at the start of the `theglossary` environment:

```

8712 \renewcommand*{\glossaryheader}{%
8713 \par\noindent\glstreenamefmt{\glsgroupnavigation}\par\indexspace}%

```

Each group has a heading (in bold with a target) followed by a vertical gap):

```

8714 \renewcommand*{\glsgroupheading}[1]{%
8715 \par\noindent
8716 \glstreenamefmt{\glsgrouphypertarget{##1}{\glsgrouptitle{##1}}}\par
8717 \indexspace}%
8718 }

```


`\glstreeindent` Length governing left indent for each level of the tree style.

```
8719 \newlength\glstreeindent
8720 \setlength{\glstreeindent}{10pt}
```

`treenoname` The `treenoname` glossary style is like the tree style, but doesn't print the name or symbol for sub-levels.

```
8721 \newglossarystyle{treenoname}{%
```

Set the paragraph indentation and skip:

```
8722 \renewenvironment{theglossary}%
8723 {\setlength{\parindent}{0pt}%
8724 \setlength{\parskip}{0pt plus 0.3pt}}%
8725 {}%
```

No header:

```
8726 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8727 \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.

```
8728 \renewcommand{\glossentry}[2]{%
8729 \hangindent0pt\relax
8730 \parindent0pt\relax
8731 \glstryitem{##1}\glstreenamfmt{\glstarget{##1}{\glossentryname{##1}}}%
8732 \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
8733 \space\glossentrydesc{##1}\glspostdescription\space##2\par
8734 }%
```

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.

```
8735 \renewcommand{\subglossentry}[3]{%
8736 \hangindent##1\glstreeindent\relax
8737 \parindent##1\glstreeindent\relax
8738 \ifnum##1=1\relax
8739 \glssubentryitem{##2}%
8740 \fi
8741 \glstarget{##2}{\strut}%
8742 \glossentrydesc{##2}\glspostdescription\space##3\par
8743 }%
```

Vertical gap between groups is the same as that used by indices:

```
8744 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
8745 }
```

`treenonamegroup` Like the `treenoname` style but the glossary groups have headings.

```
8746 \newglossarystyle{treenonamegroup}{%
```

Base it on the `glostytreetreenoname` style:

```
8747 \setglossarystyle{treenoname}%
```

Give each group a heading:

```
8748 \renewcommand{\glsgroupheading}[1]{\par
8749 \noindent\glstreenamefmt{\glsgrouptitle{##1}}\par\indexspace}%
8750 }
```

`treenonamehypergroup` The `treenonamehypergroup` style is like the `treenonamegroup` style, but has a set of links to the groups at the start of the glossary.

```
8751 \newglossarystyle{treenonamehypergroup}{%
```

Base it on the `glostytreetreenoname` style:

```
8752 \setglossarystyle{treenoname}%
```

Put navigation links to the groups at the start of the `theglossary` environment:

```
8753 \renewcommand*{\glossaryheader}{%
8754 \par\noindent\glstreenamefmt{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
8755 \renewcommand*{\glsgroupheading}[1]{%
8756 \par\noindent
8757 \glstreenamefmt{\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}\par
8758 \indexspace}%
8759 }
```

`\glsssetwidest` `\glsssetwidest[level]{text}` sets the widest text for the given level. It is used by the `alttree` glossary styles to determine the indentation of each level.

```
8760 \newcommand*{\glsssetwidest}[2][0]{%
8761 \expandafter\def\csname @glswidestname\romannumeral#1\endcsname{%
8762 #2}%
8763 }
```

`\@glswidestname` Initialise `\@glswidestname`.

```
8764 \newcommand*{\@glswidestname}{}
```

`\glstreenamebox` Used by the `alttree` style to create the box for the name and associated information.

```
8765 \newcommand*{\glstreenamebox}[2]{%
8766 \makebox[#1][1]{#2}%
8767 }
```

`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 `\glsssetwidest`.

```
8768 \newglossarystyle{alttree}{%
```

Redefine the `theglossary` environment.

```
8769 \renewenvironment{theglossary}%
8770 {\def\@glsprevlevel{-1}%
8771 \mbox{}\par}%
8772 {\par}%
```

Set the header and group headers to nothing.

```
8773 \renewcommand*{\glossaryheader}{}%
8774 \renewcommand*{\glsgroupheading}[1]{}%
```

Redefine the way that the level 0 entries are displayed.

```
8775 \renewcommand{\glossentry}[2]{%
8776 \ifnum\@gls@prevlevel=0\relax
8777 \else
```

Find out how big the indentation should be by measuring the widest entry.

```
8778 \settowidth{\glstreeindent}{\glstreenamfmt{\@glswidestname\space}}%
8779 \fi
```

Set the hangindent and paragraph indent.

```
8780 \hangindent\glstreeindent
8781 \parindent\glstreeindent
```

Put the name to the left of the paragraph block.

```
8782 \makebox[0pt][r]{\glstreenambox{\glstreeindent}{%
8783 \glsentryitem{##1}\glstreenamfmt{\glstarget{##1}{\glossentryname{##1}}}}}%
```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
8784 \ifglshassymbol{##1}{(\glossentrysymbol{##1})\space}{}%
```

Do the description followed by the description terminator and location list.

```
8785 \glossentrydesc{##1}\glspostdescription \space ##2\par
```

Set the previous level to 0.

```
8786 \def\@gls@prevlevel{0}%
8787 }%
```

Redefine the way sub-entries are displayed.

```
8788 \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.

```
8789 \ifnum##1=1\relax
8790 \glssubentryitem{##2}%
8791 \fi
```

If the level hasn't changed, keep the same settings, otherwise adjust \glstreeindent accordingly.

```
8792 \ifnum\@gls@prevlevel=##1\relax
8793 \else
```

Compute the widest entry for this level, or for level 0 if not defined for this level.

Store in \gls@tmplen

```
8794 \@ifundefined{@glswidestname\romannumeral##1}{%
8795 \settowidth{\gls@tmplen}{\glstreenamfmt{\@glswidestname\space}}}%
8796 \settowidth{\gls@tmplen}{\glstreenamfmt{%
8797 \csname @glswidestname\romannumeral##1\endcsname\space}}}%
```

Determine if going up or down a level

```
8798 \ifnum\@gls@prevlevel<##1\relax
```

Depth has increased, so add the width of the widest entry to `\glstreeindent`.

```
8799      \setlength\glstreeindent\gls@tmplen
8800      \addtolength\glstreeindent\parindent
8801      \parindent\glstreeindent
8802      \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`.

```
8803      \@ifundefined{@glswidestname\romannumeral\@gls@prevlevel}{%
8804      \settowidth{\glstreeindent}{\glstreenamfmt{%
8805      \@glswidestname\space}}}{%
8806      \settowidth{\glstreeindent}{\glstreenamfmt{%
8807      \csname @glswidestname\romannumeral\@gls@prevlevel
8808      \endcsname\space}}}{%}
```

Subtract this length from the previous level's paragraph indent and set to `\glstreeindent`.

```
8809      \addtolength\parindent{-\glstreeindent}%
8810      \setlength\glstreeindent\parindent
8811      \fi
8812      \fi
```

Set the hanging indentation.

```
8813      \hangindent\glstreeindent
```

Put the name to the left of the paragraph block

```
8814      \makebox[0pt][r]{\glstreenambox{\gls@tmplen}{%
8815      \glstreenamfmt{\glstarget{##2}{\glossentryname{##2}}}}}%}
```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
8816      \ifglshassymbol{##2}{(\glossentrysymbol{##2})\space}{}%}
```

Do the description followed by the description terminator and location list.

```
8817      \glossentrydesc{##2}\glspostdescription\space ##3\par
```

Set the previous level macro to the current level.

```
8818      \def\@gls@prevlevel{##1}%
8819      }%
```

Vertical gap between groups is the same as that used by indices:

```
8820      \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
8821      }
```

almtreegroup Like the `almtree` style but the glossary groups have headings.

```
8822 \newglossarystyle{almtreegroup}{%
```

Base it on the `glostylealmtree` style:

```
8823 \setglossarystyle{almtree}%
```

Give each group a heading.

```

8824 \renewcommand{\glsgroupheading}[1]{\par
8825   \def\@gls@prevlevel{-1}%
8826   \hangindent0pt\relax
8827   \parindent0pt\relax
8828   \glstreenamefmt{\glsgetgrouptitle{##1}}\par\indexspace}%
8829 }
```

alttreehypergroup The alttreehypergroup style is like the alttreegroup style, but has a set of links to the groups at the start of the glossary.

```
8830 \newglossarystyle{alttreehypergroup}{%
```

Base it on the glostylealttree style:

```
8831 \setglossarystyle{alttree}%
```

Put the navigation links in the header

```

8832 \renewcommand*\glossaryheader{%
8833   \par
8834   \def\@gls@prevlevel{-1}%
8835   \hangindent0pt\relax
8836   \parindent0pt\relax
8837   \glstreenamefmt{\glsnavigation}\par\indexspace}%

```

Put a hypertarget at the start of each group

```

8838 \renewcommand*\glsgroupheading[1]{%
8839   \par
8840   \def\@gls@prevlevel{-1}%
8841   \hangindent0pt\relax
8842   \parindent0pt\relax
8843   \glstreenamefmt{\glsnavhypertarget{##1}}{\glsgetgrouptitle{##1}}\par
8844   \indexspace}}

```

4 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.

```

8845 \NeedsTeXFormat{LaTeX2e}
8846 \ProvidesPackage{glossaries-compatible-207}[2015/11/22 v4.19 (NLCT)]

```

\GlsAddXdyAttribute Adds an attribute in old format.

```

8847 \ifglsxindy
8848   \renewcommand*\GlsAddXdyAttribute[1]{%
8849     \edef\@xdyattributes{\@xdyattributes ^~J \string"#1\string"}%
8850     \expandafter\toks@\expandafter{\@xdylocref}%
8851     \edef\@xdylocref{\the\toks@ ^~J%
8852 (markup-locref
8853 :open \string"\string~n\string\setentrycounter

```

```

8854    {\noexpand\glscounter}%
8855    \expandafter\string\csname#1\endcsname
8856    \expandafter\@gobble\string\{\string" ^^J
8857    :close \string"\expandafter\@gobble\string\}\string" ^^J
8858    :attr \string"#1\string")}}

```

Only has an effect before \writeist:

```

8859 \fi

```

\GlsAddXdyCounters

```

8860 \renewcommand*\GlsAddXdyCounters[1]{%
8861   \GlossariesWarning{\string\GlsAddXdyCounters\space not available
8862     in compatibility mode.}%
8863 }

```

Add predefined attributes

```

8864   \GlsAddXdyAttribute{glsnumberformat}
8865   \GlsAddXdyAttribute{textrm}
8866   \GlsAddXdyAttribute{textsf}
8867   \GlsAddXdyAttribute{texttt}
8868   \GlsAddXdyAttribute{textbf}
8869   \GlsAddXdyAttribute{textmd}
8870   \GlsAddXdyAttribute{textit}
8871   \GlsAddXdyAttribute{textup}
8872   \GlsAddXdyAttribute{textsl}
8873   \GlsAddXdyAttribute{textsc}
8874   \GlsAddXdyAttribute{emph}
8875   \GlsAddXdyAttribute{glshypernumber}
8876   \GlsAddXdyAttribute{hyperrrm}
8877   \GlsAddXdyAttribute{hypersf}
8878   \GlsAddXdyAttribute{hypertt}
8879   \GlsAddXdyAttribute{hyperbf}
8880   \GlsAddXdyAttribute{hypermd}
8881   \GlsAddXdyAttribute{hyperit}
8882   \GlsAddXdyAttribute{hyperup}
8883   \GlsAddXdyAttribute{hypersl}
8884   \GlsAddXdyAttribute{hypersc}
8885   \GlsAddXdyAttribute{hyperemph}

```

\GlsAddXdyLocation Restore v2.07 definition:

```

8886 \ifglxsindy
8887   \renewcommand*\GlsAddXdyLocation[2]{%
8888     \edef\@xdyuserlocationdefs{%
8889       \@xdyuserlocationdefs ^^J%
8890       (define-location-class \string"#1\string"^^J\space\space
8891       \space(#2))
8892     }%
8893     \edef\@xdyuserlocationnames{%
8894       \@xdyuserlocationnames^^J\space\space\space

```

```

8895         \string"#1\string"}%
8896     }
8897 \fi

```

\@do@wrglossary

```

8898 \renewcommand{\@do@wrglossary}[1]{%

```

Determine whether to use xindy or makeindex syntax

```

8899 \ifglxindy

```

Need to determine if the formatting information starts with a (or) indicating a range.

```

8900 \expandafter\@glo@check@mkidxrangechar\@glsnumberformat\@nil
8901 \def\@glo@range{%
8902 \expandafter\if\@glo@prefix(\relax
8903 \def\@glo@range{:open-range}%
8904 \else
8905 \expandafter\if\@glo@prefix)\relax
8906 \def\@glo@range{:close-range}%
8907 \fi
8908 \fi

```

Get the location and escape any special characters

```

8909 \protected@edef\@glslocref{\theglentrycounter}%
8910 \@gls@checkmkidxchars\@glslocref

```

Write to the glossary file using xindy syntax.

```

8911 \glossary[\csname glo@#1@type\endcsname]{%
8912 (indexentry :tkey (\csname glo@#1@index\endcsname)
8913 :locref \string"\@glslocref\string" %
8914 :attr \string"\@glo@suffix\string" \@glo@range
8915 )
8916 }%
8917 \else

```

Convert the format information into the format required for makeindex

```

8918 \@set@glo@numformat\@glo@numfmt\@gls@counter\@glsnumberformat

```

Write to the glossary file using makeindex syntax.

```

8919 \glossary[\csname glo@#1@type\endcsname]{%
8920 \string\glossaryentry{\csname glo@#1@index\endcsname
8921 \@gls@encapchar\@glo@numfmt}{\theglentrycounter}}%
8922 \fi
8923 }

```

\@set@glo@numformat Only had 3 arguments in v2.07

```

8924 \def\@set@glo@numformat#1#2#3{%
8925 \expandafter\@glo@check@mkidxrangechar#3\@nil
8926 \protected@edef#1{%
8927 \@glo@prefix setentrycounter[] {#2}%
8928 \expandafter\string\csname\@glo@suffix\endcsname

```

```

8929 }%
8930 \@gls@checkmkidxchars#1%
8931 }

```

\writeist Redefine \writeist back to the way it was in v2.07, but change \istfile to
\glswrite.

```

8932 \ifglxindy
8933   \def\writeist{%
8934     \openout\glswrite=\istfilename
8935     \write\glswrite{;; xindy style file created by the glossaries
8936       package in compatible-2.07 mode}%
8937     \write\glswrite{;; for document '\jobname' on
8938       \the\year-\the\month-\the\day}%
8939     \write\glswrite{^^J; required styles^^J}
8940     \@for\@xdystyle:=\@xdyrequiredstyles\do{%
8941       \ifx\@xdystyle\@empty
8942         \else
8943           \protected@write\glswrite{{(require
8944             \string"\@xdystyle.xdy\string")}}%
8945         \fi
8946       }%
8947     \write\glswrite{^^J%
8948       ; list of allowed attributes (number formats)^^J}%
8949     \write\glswrite{(define-attributes ((\@xdyattributes)))}%
8950     \write\glswrite{^^J; user defined alphabets^^J}%
8951     \write\glswrite{\@xdyuseralphabets}%
8952     \write\glswrite{^^J; location class definitions^^J}%
8953     \protected@edef\@gls@roman{\@roman{0}\string"
8954       \string"roman-numbers-lowercase\string" :sep \string"}}%
8955     \@onelevel@sanitize\@gls@roman
8956     \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
8957       :sep \string"}%
8958     \@onelevel@sanitize\@tmp
8959     \ifx\@tmp\@gls@roman
8960       \write\glswrite{(define-location-class
8961         \string"roman-page-numbers\string"^^J\space\space\space
8962         (\string"roman-numbers-lowercase\string")
8963         :min-range-length \@glsminrange)}}%
8964     \else
8965       \write\glswrite{(define-location-class
8966         \string"roman-page-numbers\string"^^J\space\space\space
8967         (:sep "\@gls@roman")
8968         :min-range-length \@glsminrange)}}%
8969     \fi
8970     \write\glswrite{(define-location-class
8971       \string"Roman-page-numbers\string"^^J\space\space\space
8972       (\string"roman-numbers-uppercase\string")
8973       :min-range-length \@glsminrange)}}%
8974     \write\glswrite{(define-location-class

```



```

8975 \string"arabic-page-numbers\string"^^J\space\space\space
8976 (\string"arabic-numbers\string")
8977 :min-range-length \@glsminrange))}%
8978 \write\glswrite{(define-location-class
8979 \string"alpha-page-numbers\string"^^J\space\space\space
8980 (\string"alpha\string")
8981 :min-range-length \@glsminrange))}%
8982 \write\glswrite{(define-location-class
8983 \string"Alpha-page-numbers\string"^^J\space\space\space
8984 (\string"ALPHA\string")
8985 :min-range-length \@glsminrange))}%
8986 \write\glswrite{(define-location-class
8987 \string"Appendix-page-numbers\string"^^J\space\space\space
8988 (\string"ALPHA\string"
8989 :sep \string"\@glsAlphacompositor\string"
8990 \string"arabic-numbers\string")
8991 :min-range-length \@glsminrange))}%
8992 \write\glswrite{(define-location-class
8993 \string"arabic-section-numbers\string"^^J\space\space\space
8994 (\string"arabic-numbers\string"
8995 :sep \string"\glscompositor\string"
8996 \string"arabic-numbers\string")
8997 :min-range-length \@glsminrange))}%
8998 \write\glswrite{^^J; user defined location classes}%
8999 \write\glswrite{@xdyuserlocationdefs}%
9000 \write\glswrite{^^J; define cross-reference class^^J}%
9001 \write\glswrite{(define-crossref-class \string"see\string"
9002 :unverified )}%
9003 \write\glswrite{(markup-crossref-list
9004 :class \string"see\string"^^J\space\space\space
9005 :open \string"\string\glsseeformat\string"
9006 :close \string"{}\string"))}%
9007 \write\glswrite{^^J; define the order of the location classes}%
9008 \write\glswrite{(define-location-class-order
9009 (\@xdylocationclassorder))}%
9010 \write\glswrite{^^J; define the glossary markup^^J}%
9011 \write\glswrite{(markup-index^^J\space\space\space
9012 :open \string"\string
9013 \glossarysection[\string\glossarytoctitle]{\string
9014 \glossarytitle}\string\glossarypreamble\string~n\string\begin
9015 {theglossary}\string\glossaryheader\string~n\string" ^^J\space
9016 \space\space:close \string"\expandafter\@gobble
9017 \string\%\string~n\string
9018 \end{theglossary}\string\glossarypostamble
9019 \string~n\string" ^^J\space\space\space
9020 :tree))}%
9021 \write\glswrite{(markup-letter-group-list
9022 :sep \string"\string\glsgroupskip\string~n\string"))}%
9023 \write\glswrite{(markup-indexentry

```

```

9024      :open \string"\string\relax \string\glsresetentrylist
9025      \string~n\string"))}%
9026  \write\glswrite{(markup-locclass-list :open
9027  \string"\glsopenbrace\string\glossaryentrynumbers
9028  \glsopenbrace\string\relax\space \string"^^J\space\space\space
9029  :sep \string", \string"
9030  :close \string"\glsclosebrace\glsclosebrace\string"))}%
9031  \write\glswrite{(markup-locref-list
9032  :sep \string"\string\delimN\space\string"))}%
9033  \write\glswrite{(markup-range
9034  :sep \string"\string\delimR\space\string"))}%
9035  \@onelevel@sanitize\gls@suffixF
9036  \@onelevel@sanitize\gls@suffixFF
9037  \ifx\gls@suffixF\@empty
9038  \else
9039    \write\glswrite{(markup-range
9040    :close "\gls@suffixF" :length 1 :ignore-end)}}%
9041  \fi
9042  \ifx\gls@suffixFF\@empty
9043  \else
9044    \write\glswrite{(markup-range
9045    :close "\gls@suffixFF" :length 2 :ignore-end)}}%
9046  \fi
9047  \write\glswrite{^^J; define format to use for locations^^J}%
9048  \write\glswrite{\@xdylocref}%
9049  \write\glswrite{^^J; define letter group list format^^J}%
9050  \write\glswrite{(markup-letter-group-list
9051  :sep \string"\string\glsgroupskip\string~n\string"))}%
9052  \write\glswrite{^^J; letter group headings^^J}%
9053  \write\glswrite{(markup-letter-group
9054  :open-head \string"\string\glsgroupheading
9055  \glsopenbrace\string"^^J\space\space\space
9056  :close-head \string"\glsclosebrace\string"))}%
9057  \write\glswrite{^^J; additional letter groups^^J}%
9058  \write\glswrite{\@xdylettergroups}%
9059  \write\glswrite{^^J; additional sort rules^^J}%
9060  \write\glswrite{\@xdysortrules}%
9061  \noist}
9062 \else
9063   \edef\@gls@actualchar{\string?}
9064   \edef\@gls@encapchar{\string|}
9065   \edef\@gls@levelchar{\string!}
9066   \edef\@gls@quotechar{\string"}
9067   \def\writeist{\relax
9068   \openout\glswrite=\istfilename
9069   \write\glswrite{\expandafter\@gobble\string\% makeindex style file
9070   created by the glossaries package}
9071   \write\glswrite{\expandafter\@gobble\string\% for document
9072   '\jobname' on \the\year-\the\month-\the\day}

```

```

9073 \write\glswrite{actual '\@gls@actualchar'}
9074 \write\glswrite{encap '\@gls@encapchar'}
9075 \write\glswrite{level '\@gls@levelchar'}
9076 \write\glswrite{quote '\@gls@quotechar'}
9077 \write\glswrite{keyword \string"\string\glossaryentry\string"}
9078 \write\glswrite{preamble \string"\string\glossarysection[\string
9079 \glossarytoctitle]{\string\glossarytitle}\string
9080 \glossarypreamble\string\n\string\begin{theglossary}\string
9081 \glossaryheader\string\n\string"}
9082 \write\glswrite{postamble \string"\string%\string\n\string
9083 \end{theglossary}\string\glossarypostamble\string\n
9084 \string"}
9085 \write\glswrite{group_skip \string"\string\glsgroupskip\string\n
9086 \string"}
9087 \write\glswrite{item_0 \string"\string%\string\n\string"}
9088 \write\glswrite{item_1 \string"\string%\string\n\string"}
9089 \write\glswrite{item_2 \string"\string%\string\n\string"}
9090 \write\glswrite{item_01 \string"\string%\string\n\string"}
9091 \write\glswrite{item_x1
9092 \string"\string\relax \string\glresetentrylist\string\n
9093 \string"}
9094 \write\glswrite{item_12 \string"\string%\string\n\string"}
9095 \write\glswrite{item_x2
9096 \string"\string\relax \string\glresetentrylist\string\n
9097 \string"}
9098 \write\glswrite{delim_0 \string"\string\{\string
9099 \glossaryentrynumbers\string\{\string\relax \string"}
9100 \write\glswrite{delim_1 \string"\string\{\string
9101 \glossaryentrynumbers\string\{\string\relax \string"}
9102 \write\glswrite{delim_2 \string"\string\{\string
9103 \glossaryentrynumbers\string\{\string\relax \string"}
9104 \write\glswrite{delim_t \string"\string\}\string\}\string"}
9105 \write\glswrite{delim_n \string"\string\delimN \string"}
9106 \write\glswrite{delim_r \string"\string\delimR \string"}
9107 \write\glswrite{headings_flag 1}
9108 \write\glswrite{heading_prefix
9109 \string"\string\glsgroupheading\string\{\string"}
9110 \write\glswrite{heading_suffix
9111 \string"\string\}\string\relax
9112 \string\glresetentrylist \string"}
9113 \write\glswrite{symhead_positive \string"glssymbols\string"}
9114 \write\glswrite{numhead_positive \string"glslnumbers\string"}
9115 \write\glswrite{page_compositor \string"glscpositor\string"}
9116 \@gls@escbsdq\gls@suffixF
9117 \@gls@escbsdq\gls@suffixFF
9118 \ifx\gls@suffixF\@empty
9119 \else
9120 \write\glswrite{suffix_2p \string"\gls@suffixF\string"}
9121 \fi

```

```

9122 \ifx\gls@suffixFF\@empty
9123 \else
9124 \write\glswrite{suffix_3p \string"\gls@suffixFF\string"}
9125 \fi
9126 \noist
9127 }
9128 \fi

```

\noist

```

9129 \renewcommand*{\noist}{\let\writeist\relax}

```

Compatibility macros.

```

9130 \NeedsTeXFormat{LaTeX2e}
9131 \ProvidesPackage{glossaries-compatible-307}[2015/11/22 v4.19 (NLCT)]

```

Compatibility macros for predefined glossary styles:

`compatglossarystyle` Defines a compatibility glossary style.

```

9132 \newcommand{\compatglossarystyle}[2]{%
9133 \ifcsundef{@glscompstyle@#1}%
9134 {%
9135 \csdef{@glscompstyle@#1}{#2}%
9136 }%
9137 {%
9138 \PackageError{glossaries}{Glossary compatibility style ‘#1’ is already defined}{}%
9139 }%
9140 }

```

Backward compatible inline style.

```

9141 \compatglossarystyle{inline}{%
9142 \renewcommand{\glossaryentryfield}[5]{%
9143 \glsinlinedopostchild
9144 \gls@inlinesep
9145 \def\glo@desc{##3}%
9146 \def\@no@post@desc{\nopostdesc}%
9147 \glsentryitem{##1}\glsinlinenameformat{##1}{##2}%
9148 \ifx\glo@desc\@no@post@desc
9149 \glsinlineemptydescformat{##4}{##5}%
9150 \else
9151 \ifstrempy{##3}%
9152 {\glsinlineemptydescformat{##4}{##5}}%
9153 {\glsinlinedescformat{##3}{##4}{##5}}%
9154 \fi
9155 \ifglshaschildren{##1}%
9156 {%
9157 \glsresetsubentrycounter
9158 \glsinlineparentchildseparator
9159 \def\gls@inlinesubsep{}%
9160 \def\gls@inlinepostchild{\glsinlinepostchild}%
9161 }%

```

```

9162    {}%
9163    \def\gls@inlinesep{\glsinlineseparator}%
9164  }%

  Sub-entries display description:
9165    \renewcommand{\glossarysubentryfield}[6]{%
9166      \gls@inlinesubsep%
9167      \glsinlinesubnameformat{##2}{##3}%
9168      \glssubentryitem{##2}\glsinlinesubdescformat{##4}{##5}{##6}%
9169      \def\gls@inlinesubsep{\glsinlinesubseparator}%
9170    }%
9171 }

```

Backward compatible list style.

```

9172 \compatglossarystyle{list}{%
9173   \renewcommand*{\glossaryentryfield}[5]{%
9174     \item[\glsentryitem{##1}\glstarget{##1}{##2}]
9175       ##3\glspostdescription\space ##5}%

```

Sub-entries continue on the same line:

```

9176   \renewcommand*{\glossarysubentryfield}[6]{%
9177     \glssubentryitem{##2}%
9178     \glstarget{##2}{\strut}##4\glspostdescription\space ##6.}%
9179 }

```

Backward compatible listgroup style.

```

9180 \compatglossarystyle{listgroup}{%
9181   \csuse{@glscompstyle@list}%
9182 }%

```

Backward compatible listhypergroup style.

```

9183 \compatglossarystyle{listhypergroup}{%
9184   \csuse{@glscompstyle@list}%
9185 }%

```

Backward compatible altlist style.

```

9186 \compatglossarystyle{altlist}{%
9187   \renewcommand*{\glossaryentryfield}[5]{%
9188     \item[\glsentryitem{##1}\glstarget{##1}{##2}]%
9189     \mbox{}\par\nobreak\@afterheading
9190     ##3\glspostdescription\space ##5}%
9191   \renewcommand{\glossarysubentryfield}[6]{%
9192     \par
9193     \glssubentryitem{##2}%
9194     \glstarget{##2}{\strut}##4\glspostdescription\space ##6}%
9195 }%

```

Backward compatible altlistgroup style.

```

9196 \compatglossarystyle{altlistgroup}{%
9197   \csuse{@glscompstyle@altlist}%
9198 }%

```

Backward compatible altlisthypergroup style.

```
9199 \compatglossarystyle{altlisthypergroup}{%
9200 \csuse{@glscmpstyle@altlist}%
9201 }%
```

Backward compatible listdotted style.

```
9202 \compatglossarystyle{listdotted}{%
9203 \renewcommand*{\glossaryentryfield}[5]{%
9204 \item[]\makebox[\glslistdottedwidth][l]{%
9205 \glstryitem{##1}\glstarget{##1}{##2}%
9206 \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##3}%
9207 \renewcommand*{\glossarysubentryfield}[6]{%
9208 \item[]\makebox[\glslistdottedwidth][l]{%
9209 \glssubentryitem{##2}%
9210 \glstarget{##2}{##3}%
9211 \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##4}%
9212 }%
```

Backward compatible sublistdotted style.

```
9213 \compatglossarystyle{sublistdotted}{%
9214 \csuse{@glscmpstyle@listdotted}%
9215 \renewcommand*{\glossaryentryfield}[5]{%
9216 \item[\glstryitem{##1}\glstarget{##1}{##2}]}%
9217 }%
```

Backward compatible long style.

```
9218 \compatglossarystyle{long}{%
9219 \renewcommand*{\glossaryentryfield}[5]{%
9220 \glstryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
9221 \renewcommand*{\glossarysubentryfield}[6]{%
9222 &
9223 \glssubentryitem{##2}%
9224 \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
9225 }%
```

Backward compatible longborder style.

```
9226 \compatglossarystyle{longborder}{%
9227 \csuse{@glscmpstyle@long}%
9228 }%
```

Backward compatible longheader style.

```
9229 \compatglossarystyle{longheader}{%
9230 \csuse{@glscmpstyle@long}%
9231 }%
```

Backward compatible longheaderborder style.

```
9232 \compatglossarystyle{longheaderborder}{%
9233 \csuse{@glscmpstyle@long}%
9234 }%
```

Backward compatible long3col style.

```
9235 \compatglossarystyle{long3col}{%
```

```

9236 \renewcommand*{\glossaryentryfield}[5]{%
9237   \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\\}%
9238 \renewcommand*{\glossarysubentryfield}[6]{%
9239   &
9240   \glssubentryitem{##2}%
9241   \glstarget{##2}{\strut}##4 & ##6\\}%
9242 }%

Backward compatible long3colborder style.
9243 \compatglossarystyle{long3colborder}{%
9244 \csuse{@glscmpstyle@long3col}%
9245 }%

Backward compatible long3colheader style.
9246 \compatglossarystyle{long3colheader}{%
9247 \csuse{@glscmpstyle@long3col}%
9248 }%

Backward compatible long3colheaderborder style.
9249 \compatglossarystyle{long3colheaderborder}{%
9250 \csuse{@glscmpstyle@long3col}%
9251 }%

Backward compatible long4col style.
9252 \compatglossarystyle{long4col}{%
9253 \renewcommand*{\glossaryentryfield}[5]{%
9254   \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%
9255 \renewcommand*{\glossarysubentryfield}[6]{%
9256   &
9257   \glssubentryitem{##2}%
9258   \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%
9259 }%

Backward compatible long4colheader style.
9260 \compatglossarystyle{long4colheader}{%
9261 \csuse{@glscmpstyle@long4col}%
9262 }%

Backward compatible long4colborder style.
9263 \compatglossarystyle{long4colborder}{%
9264 \csuse{@glscmpstyle@long4col}%
9265 }%

Backward compatible long4colheaderborder style.
9266 \compatglossarystyle{long4colheaderborder}{%
9267 \csuse{@glscmpstyle@long4col}%
9268 }%

Backward compatible altlong4col style.
9269 \compatglossarystyle{altlong4col}{%
9270 \csuse{@glscmpstyle@long4col}%
9271 }%

```

Backward compatible altlong4colheader style.

```
9272 \compatglossarystyle{altlong4colheader}{%  
9273 \csuse{@glscompstyle@long4col}%  
9274 }%
```

Backward compatible altlong4colborder style.

```
9275 \compatglossarystyle{altlong4colborder}{%  
9276 \csuse{@glscompstyle@long4col}%  
9277 }%
```

Backward compatible altlong4colheaderborder style.

```
9278 \compatglossarystyle{altlong4colheaderborder}{%  
9279 \csuse{@glscompstyle@long4col}%  
9280 }%
```

Backward compatible long style.

```
9281 \compatglossarystyle{longragged}{%  
9282 \renewcommand*{\glossaryentryfield}[5]{%  
9283 \glstarget{##1}{##2} & ##3\glspostdescription\space ##5%  
9284 \tabularnewline}%  
9285 \renewcommand*{\glossarysubentryfield}[6]{%  
9286 &  
9287 \glssubentryitem{##2}%  
9288 \glstarget{##2}{\strut}##4\glspostdescription\space ##6%  
9289 \tabularnewline}%  
9290 }%
```

Backward compatible longraggedborder style.

```
9291 \compatglossarystyle{longraggedborder}{%  
9292 \csuse{@glscompstyle@longragged}%  
9293 }%
```

Backward compatible longraggedheader style.

```
9294 \compatglossarystyle{longraggedheader}{%  
9295 \csuse{@glscompstyle@longragged}%  
9296 }%
```

Backward compatible longraggedheaderborder style.

```
9297 \compatglossarystyle{longraggedheaderborder}{%  
9298 \csuse{@glscompstyle@longragged}%  
9299 }%
```

Backward compatible longragged3col style.

```
9300 \compatglossarystyle{longragged3col}{%  
9301 \renewcommand*{\glossaryentryfield}[5]{%  
9302 \glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%  
9303 \renewcommand*{\glossarysubentryfield}[6]{%  
9304 &  
9305 \glssubentryitem{##2}%  
9306 \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%  
9307 }%
```


Backward compatible longragged3colborder style.

```
9308 \compatglossarystyle{longragged3colborder}{%
9309 \csuse{@glscmpstyle@longragged3col}%
9310 }%
```

Backward compatible longragged3colheader style.

```
9311 \compatglossarystyle{longragged3colheader}{%
9312 \csuse{@glscmpstyle@longragged3col}%
9313 }%
```

Backward compatible longragged3colheaderborder style.

```
9314 \compatglossarystyle{longragged3colheaderborder}{%
9315 \csuse{@glscmpstyle@longragged3col}%
9316 }%
```

Backward compatible altlongragged4col style.

```
9317 \compatglossarystyle{altlongragged4col}{%
9318 \renewcommand*{\glossaryentryfield}[5]{%
9319 \glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%
9320 \renewcommand*{\glossarysubentryfield}[6]{%
9321 &
9322 \glssubentryitem{##2}%
9323 \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%
9324 }%
```

Backward compatible altlongragged4colheader style.

```
9325 \compatglossarystyle{altlongragged4colheader}{%
9326 \csuse{@glscmpstyle@altlong4col}%
9327 }%
```

Backward compatible altlongragged4colborder style.

```
9328 \compatglossarystyle{altlongragged4colborder}{%
9329 \csuse{@glscmpstyle@altlong4col}%
9330 }%
```

Backward compatible altlongragged4colheaderborder style.

```
9331 \compatglossarystyle{altlongragged4colheaderborder}{%
9332 \csuse{@glscmpstyle@altlong4col}%
9333 }%
```

Backward compatible index style.

```
9334 \compatglossarystyle{index}{%
9335 \renewcommand*{\glossaryentryfield}[5]{%
9336 \item\glstarget{##1}{##2}%
9337 \ifx\relax##4\relax
9338 \else
9339 \space{##4}%
9340 \fi
9341 \space ##3\glspostdescription \space ##5}%
9342 \renewcommand*{\glossarysubentryfield}[6]{%
9343 \ifcase##1\relax
9344 % level 0
```

```

9345     \item
9346   \or
9347     % level 1
9348     \subitem
9349     \glssubentryitem{##2}%
9350   \else
9351     % all other levels
9352     \subsubitem
9353   \fi
9354   \textbf{\glstarget{##2}{##3}}%
9355   \ifx\relax##5\relax
9356   \else
9357     \space(##5)%
9358   \fi
9359   \space##4\glspostdescription\space ##6}%
9360 }%

```

Backward compatible indexgroup style.

```

9361 \compatglossarystyle{indexgroup}{%
9362   \csuse{@glscmpstyle@index}%
9363 }%

```

Backward compatible indexhypergroup style.

```

9364 \compatglossarystyle{indexhypergroup}{%
9365   \csuse{@glscmpstyle@index}%
9366 }%

```

Backward compatible tree style.

```

9367 \compatglossarystyle{tree}{%
9368   \renewcommand{\glossaryentryfield}[5]{%
9369     \hangindent0pt\relax
9370     \parindent0pt\relax
9371     \glssentryitem{##1}\textbf{\glstarget{##1}{##2}}%
9372     \ifx\relax##4\relax
9373     \else
9374       \space(##4)%
9375     \fi
9376     \space ##3\glspostdescription \space ##5\par}%
9377   \renewcommand{\glossarysubentryfield}[6]{%
9378     \hangindent##1\glstreeindent\relax
9379     \parindent##1\glstreeindent\relax
9380     \ifnum##1=1\relax
9381       \glssubentryitem{##2}%
9382     \fi
9383     \textbf{\glstarget{##2}{##3}}%
9384     \ifx\relax##5\relax
9385     \else
9386       \space(##5)%
9387     \fi
9388     \space##4\glspostdescription\space ##6\par}%
9389 }%

```

Backward compatible treegroup style.

```
9390 \compatglossarystyle{treegroup}{%
9391 \csuse{@glscompstyle@tree}%
9392 }%
```

Backward compatible treehypergroup style.

```
9393 \compatglossarystyle{treehypergroup}{%
9394 \csuse{@glscompstyle@tree}%
9395 }%
```

Backward compatible treenoname style.

```
9396 \compatglossarystyle{treenoname}{%
9397 \renewcommand{\glossaryentryfield}[5]{%
9398 \hangindent0pt\relax
9399 \parindent0pt\relax
9400 \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
9401 \ifx\relax##4\relax
9402 \else
9403 \space{##4}%
9404 \fi
9405 \space ##3\glspostdescription \space ##5\par}%
9406 \renewcommand{\glossarysubentryfield}[6]{%
9407 \hangindent##1\glstreeindent\relax
9408 \parindent##1\glstreeindent\relax
9409 \ifnum##1=1\relax
9410 \glssubentryitem{##2}%
9411 \fi
9412 \glstarget{##2}{\strut}%
9413 ##4\glspostdescription\space ##6\par}%
9414 }%
```

Backward compatible treenonamegroup style.

```
9415 \compatglossarystyle{treenonamegroup}{%
9416 \csuse{@glscompstyle@treenoname}%
9417 }%
```

Backward compatible treenonamehypergroup style.

```
9418 \compatglossarystyle{treenonamehypergroup}{%
9419 \csuse{@glscompstyle@treenoname}%
9420 }%
```

Backward compatible alttree style.

```
9421 \compatglossarystyle{alttree}{%
9422 \renewcommand{\glossaryentryfield}[5]{%
9423 \ifnum\@gls@prevlevel=0\relax
9424 \else
9425 \settowidth{\glstreeindent}{\textbf{\@glswidestname\space}}%
9426 \hangindent\glstreeindent
9427 \parindent\glstreeindent
9428 \fi
9429 \makebox[0pt][r]{\makebox[\glstreeindent][l]{%

```

```

9430      \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}}%
9431      \ifx\relax##4\relax
9432      \else
9433        (##4)\space
9434      \fi
9435      ##3\glspostdescription \space ##5\par
9436      \def\@gls@prevlevel{0}%
9437    }%
9438    \renewcommand{\glossarysubentryfield}[6]{%
9439      \ifnum##1=1\relax
9440        \glssubentryitem{##2}%
9441      \fi
9442      \ifnum\@gls@prevlevel=##1\relax
9443      \else
9444        \@ifundefined{\@glswidestname\romannumeral##1}{%
9445          \settowidth{\@gls@tmplen}{\textbf{\@glswidestname\space}}{%
9446            \settowidth{\@gls@tmplen}{\textbf{%
9447              \csname \@glswidestname\romannumeral##1\endcsname\space}}}%
9448          \ifnum\@gls@prevlevel<##1\relax
9449            \setlength\glstreeindent\@gls@tmplen
9450            \addtolength\glstreeindent\parindent
9451            \parindent\glstreeindent
9452          \else
9453            \@ifundefined{\@glswidestname\romannumeral\@gls@prevlevel}{%
9454              \settowidth{\glstreeindent}{\textbf{%
9455                \@glswidestname\space}}{%
9456                \settowidth{\glstreeindent}{\textbf{%
9457                  \csname \@glswidestname\romannumeral\@gls@prevlevel
9458                    \endcsname\space}}}%
9459              \addtolength\parindent{-\glstreeindent}%
9460              \setlength\glstreeindent\parindent
9461            \fi
9462          \fi
9463          \hangindent\glstreeindent
9464          \makebox[0pt][r]{\makebox[\@gls@tmplen][l]{%
9465            \textbf{\glstarget{##2}{##3}}}%
9466          \ifx##5\relax\relax
9467          \else
9468            (##5)\space
9469          \fi
9470          ##4\glspostdescription\space ##6\par
9471          \def\@gls@prevlevel{##1}%
9472        }%
9473    }%

```

Backward compatible alttreegroup style.

```

9474 \compatglossarystyle{alttreegroup}{%
9475   \csuse{\@glscompstyle@almtree}%
9476 }%

```

Backward compatible alttreehypergroup style.

```
9477 \compatglossarystyle{alttreehypergroup}{%  
9478 \csuse{@glscompstyle@alttree}%  
9479 }%
```

Backward compatible mcolindex style.

```
9480 \compatglossarystyle{mcolindex}{%  
9481 \csuse{@glscompstyle@index}%  
9482 }%
```

Backward compatible mcolindexgroup style.

```
9483 \compatglossarystyle{mcolindexgroup}{%  
9484 \csuse{@glscompstyle@index}%  
9485 }%
```

Backward compatible mcolindexhypergroup style.

```
9486 \compatglossarystyle{mcolindexhypergroup}{%  
9487 \csuse{@glscompstyle@index}%  
9488 }%
```

Backward compatible mcoltree style.

```
9489 \compatglossarystyle{mcoltree}{%  
9490 \csuse{@glscompstyle@tree}%  
9491 }%
```

Backward compatible mcoltreegroup style.

```
9492 \compatglossarystyle{mcolindextreegroup}{%  
9493 \csuse{@glscompstyle@tree}%  
9494 }%
```

Backward compatible mcoltreehypergroup style.

```
9495 \compatglossarystyle{mcolindextreehypergroup}{%  
9496 \csuse{@glscompstyle@tree}%  
9497 }%
```

Backward compatible mcoltreenoname style.

```
9498 \compatglossarystyle{mcoltreenoname}{%  
9499 \csuse{@glscompstyle@tree}%  
9500 }%
```

Backward compatible mcoltreenonamegroup style.

```
9501 \compatglossarystyle{mcoltreenonamegroup}{%  
9502 \csuse{@glscompstyle@tree}%  
9503 }%
```

Backward compatible mcoltreenonamehypergroup style.

```
9504 \compatglossarystyle{mcoltreenonamehypergroup}{%  
9505 \csuse{@glscompstyle@tree}%  
9506 }%
```

Backward compatible mcolalttree style.

```
9507 \compatglossarystyle{mcolalttree}{%  
9508 \csuse{@glscompstyle@alttree}%  
9509 }%
```

Backward compatible mcolalmtreegroup style.

```
9510 \compatglossarystyle{mcolalmtreegroup}{%
9511 \csuse{@glscmpstyle@almtree}%
9512 }%
```

Backward compatible mcolalmtreehypergroup style.

```
9513 \compatglossarystyle{mcolalmtreehypergroup}{%
9514 \csuse{@glscmpstyle@almtree}%
9515 }%
```

Backward compatible superragged style.

```
9516 \compatglossarystyle{superragged}{%
9517 \renewcommand*{\glossaryentryfield}[5]{%
9518 \glstryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
9519 \tabularnewline}%
9520 \renewcommand*{\glossarysubentryfield}[6]{%
9521 &
9522 \glssubentryitem{##2}%
9523 \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
9524 \tabularnewline}%
9525 }%
```

Backward compatible superraggedborder style.

```
9526 \compatglossarystyle{superraggedborder}{%
9527 \csuse{@glscmpstyle@superragged}%
9528 }%
```

Backward compatible superraggedheader style.

```
9529 \compatglossarystyle{superraggedheader}{%
9530 \csuse{@glscmpstyle@superragged}%
9531 }%
```

Backward compatible superraggedheaderborder style.

```
9532 \compatglossarystyle{superraggedheaderborder}{%
9533 \csuse{@glscmpstyle@superragged}%
9534 }%
```

Backward compatible superragged3col style.

```
9535 \compatglossarystyle{superragged3col}{%
9536 \renewcommand*{\glossaryentryfield}[5]{%
9537 \glstryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%
9538 \renewcommand*{\glossarysubentryfield}[6]{%
9539 &
9540 \glssubentryitem{##2}%
9541 \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%
9542 }%
```

Backward compatible superragged3colborder style.

```
9543 \compatglossarystyle{superragged3colborder}{%
9544 \csuse{@glscmpstyle@superragged3col}%
9545 }%
```

Backward compatible superragged3colheader style.

```
9546 \compatglossarystyle{superragged3colheader}{%
9547 \csuse{@glscmpstyle@superragged3col}%
9548 }%
```

Backward compatible superragged3colheaderborder style.

```
9549 \compatglossarystyle{superragged3colheaderborder}{%
9550 \csuse{@glscmpstyle@superragged3col}%
9551 }%
```

Backward compatible altsuperragged4col style.

```
9552 \compatglossarystyle{altsuperragged4col}{%
9553 \renewcommand*{\glossaryentryfield}[5]{%
9554 \glstryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%
9555 \renewcommand*{\glossarysubentryfield}[6]{%
9556 &
9557 \glssubentryitem{##2}%
9558 \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%
9559 }%
```

Backward compatible altsuperragged4colheader style.

```
9560 \compatglossarystyle{altsuperragged4colheader}{%
9561 \csuse{@glscmpstyle@altsuperragged4col}%
9562 }%
```

Backward compatible altsuperragged4colborder style.

```
9563 \compatglossarystyle{altsuperragged4colborder}{%
9564 \csuse{@glscmpstyle@altsuperragged4col}%
9565 }%
```

Backward compatible altsuperragged4colheaderborder style.

```
9566 \compatglossarystyle{altsuperragged4colheaderborder}{%
9567 \csuse{@glscmpstyle@altsuperragged4col}%
9568 }%
```

Backward compatible super style.

```
9569 \compatglossarystyle{super}{%
9570 \renewcommand*{\glossaryentryfield}[5]{%
9571 \glstryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
9572 \renewcommand*{\glossarysubentryfield}[6]{%
9573 &
9574 \glssubentryitem{##2}%
9575 \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
9576 }%
```

Backward compatible superborder style.

```
9577 \compatglossarystyle{superborder}{%
9578 \csuse{@glscmpstyle@super}%
9579 }%
```

Backward compatible superheader style.

```
9580 \compatglossarystyle{superheader}{%
```

```

9581 \csuse{@glscompstyle@super}%
9582 }%

```

Backward compatible superheaderborder style.

```

9583 \compatglossarystyle{superheaderborder}{%
9584 \csuse{@glscompstyle@super}%
9585 }%

```

Backward compatible super3col style.

```

9586 \compatglossarystyle{super3col}{%
9587 \renewcommand*{\glossaryentryfield}[5]{%
9588 \glstarget{##1}{##2} & ##3 & ##5\\}%
9589 \renewcommand*{\glossarysubentryfield}[6]{%
9590 &
9591 \glssubentryitem{##2}%
9592 \glstarget{##2}{\strut}##4 & ##6\\}%
9593 }%

```

Backward compatible super3colborder style.

```

9594 \compatglossarystyle{super3colborder}{%
9595 \csuse{@glscompstyle@super3col}%
9596 }%

```

Backward compatible super3colheader style.

```

9597 \compatglossarystyle{super3colheader}{%
9598 \csuse{@glscompstyle@super3col}%
9599 }%

```

Backward compatible super3colheaderborder style.

```

9600 \compatglossarystyle{super3colheaderborder}{%
9601 \csuse{@glscompstyle@super3col}%
9602 }%

```

Backward compatible super4col style.

```

9603 \compatglossarystyle{super4col}{%
9604 \renewcommand*{\glossaryentryfield}[5]{%
9605 \glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%
9606 \renewcommand*{\glossarysubentryfield}[6]{%
9607 &
9608 \glssubentryitem{##2}%
9609 \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%
9610 }%

```

Backward compatible super4colheader style.

```

9611 \compatglossarystyle{super4colheader}{%
9612 \csuse{@glscompstyle@super4col}%
9613 }%

```

Backward compatible super4colborder style.

```

9614 \compatglossarystyle{super4colborder}{%
9615 \csuse{@glscompstyle@super4col}%
9616 }%

```


Backward compatible super4colheaderborder style.

```
9617 \compatglossarystyle{super4colheaderborder}{%
9618 \csuse{@glscompstyle@super4col}%
9619 }%
```

Backward compatible altsuper4col style.

```
9620 \compatglossarystyle{altsuper4col}{%
9621 \csuse{@glscompstyle@super4col}%
9622 }%
```

Backward compatible altsuper4colheader style.

```
9623 \compatglossarystyle{altsuper4colheader}{%
9624 \csuse{@glscompstyle@super4col}%
9625 }%
```

Backward compatible altsuper4colborder style.

```
9626 \compatglossarystyle{altsuper4colborder}{%
9627 \csuse{@glscompstyle@super4col}%
9628 }%
```

Backward compatible altsuper4colheaderborder style.

```
9629 \compatglossarystyle{altsuper4colheaderborder}{%
9630 \csuse{@glscompstyle@super4col}%
9631 }%
```

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.

```
9632 \NeedsTeXFormat{LaTeX2e}
```

Package version number now in line with main glossaries package number but will only be updated when glossaries-accsupp.sty is modified.

```
9633 \ProvidesPackage{glossaries-accsupp}[2015/11/22 v4.19 (NLCT)
9634 Experimental glossaries accessibility]
```

Pass all options to glossaries:

```
9635 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
9636 \ProcessOptions
```

`\compatibleglossentry` Override style compatibility macros:

```
9637 \def\compatibleglossentry#1#2{%
9638 \toks@{#2}%
9639 \protected@edef\@do@glossentry{%
9640 \noexpand\accsuppglossaryentryfield{#1}%
9641 {\noexpand\glsnamefont
9642 {\expandafter\expandonce\csname glo@glsdetoklabel{#1}@name\endcsname}}%
```

```

9643     {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@desc\endcsname}%
9644     {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@symbol\endcsname}%
9645     {\the\toks@}%
9646   }%
9647   \@do@glossentry
9648 }

```

atiblesubglossentry

```

9649 \def\compatiblesubglossentry#1#2#3{%
9650   \toks@{#3}%
9651   \protected@edef\@do@subglossentry{%
9652     \noexpand\accsuppglossarysubentryfield{\number#1}%
9653     {#2}%
9654     {\noexpand\glsnamefont
9655       {\expandafter\expandonce\csname glo@\glsdetoklabel{#2}@name\endcsname}}}%
9656     {\expandafter\expandonce\csname glo@\glsdetoklabel{#2}@desc\endcsname}%
9657     {\expandafter\expandonce\csname glo@\glsdetoklabel{#2}@symbol\endcsname}%
9658     {\the\toks@}%
9659   }%
9660   \@do@subglossentry
9661 }

```

Required packages:

```

9662 \RequirePackage{glossaries}
9663 \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:

```

9664 \define@key{glossentry}{access}{%
9665   \def\@glo@access{#1}%
9666 }

```

textaccess The replacement text corresponding to the text key:

```

9667 \define@key{glossentry}{textaccess}{%
9668   \def\@glo@textaccess{#1}%
9669 }

```

firstaccess The replacement text corresponding to the first key:

```

9670 \define@key{glossentry}{firstaccess}{%
9671   \def\@glo@firstaccess{#1}%
9672 }

```

pluralaccess The replacement text corresponding to the plural key:

```
9673 \define@key{glossentry}{pluralaccess}{%
9674   \def\@glo@pluralaccess{#1}%
9675 }
```

firstpluralaccess The replacement text corresponding to the firstplural key:

```
9676 \define@key{glossentry}{firstpluralaccess}{%
9677   \def\@glo@firstpluralaccess{#1}%
9678 }
```

symbolaccess The replacement text corresponding to the symbol key:

```
9679 \define@key{glossentry}{symbolaccess}{%
9680   \def\@glo@symbolaccess{#1}%
9681 }
```

symbolpluralaccess The replacement text corresponding to the symbolplural key:

```
9682 \define@key{glossentry}{symbolpluralaccess}{%
9683   \def\@glo@symbolpluralaccess{#1}%
9684 }
```

descriptionaccess The replacement text corresponding to the description key:

```
9685 \define@key{glossentry}{descriptionaccess}{%
9686   \def\@glo@descaccess{#1}%
9687 }
```

descriptionpluralaccess The replacement text corresponding to the descriptionplural key:

```
9688 \define@key{glossentry}{descriptionpluralaccess}{%
9689   \def\@glo@descpluralaccess{#1}%
9690 }
```

shortaccess The replacement text corresponding to the short key:

```
9691 \define@key{glossentry}{shortaccess}{%
9692   \def\@glo@shortaccess{#1}%
9693 }
```

shortpluralaccess The replacement text corresponding to the shortplural key:

```
9694 \define@key{glossentry}{shortpluralaccess}{%
9695   \def\@glo@shortpluralaccess{#1}%
9696 }
```

longaccess The replacement text corresponding to the long key:

```
9697 \define@key{glossentry}{longaccess}{%
9698   \def\@glo@longaccess{#1}%
9699 }
```

longpluralaccess The replacement text corresponding to the longplural key:

```
9700 \define@key{glossentry}{longpluralaccess}{%
9701   \def\@glo@longpluralaccess{#1}%
9702 }
```

There are no equivalent keys for the user1...user6 keys. The replacement text would have to be explicitly put in the value, e.g., user1={\glsaccsupp{inches}{in}}.

Append these new keys to \@gls@keymap:

```

9703 \appto\@gls@keymap{,%
9704   {access}{access},%
9705   {textaccess}{textaccess},%
9706   {firstaccess}{firstaccess},%
9707   {pluralaccess}{pluralaccess},%
9708   {firstpluralaccess}{firstpluralaccess},%
9709   {symbolaccess}{symbolaccess},%
9710   {symbolpluralaccess}{symbolpluralaccess},%
9711   {descaccess}{descaccess},%
9712   {descpluralaccess}{descpluralaccess},%
9713   {shortaccess}{shortaccess},%
9714   {shortpluralaccess}{shortpluralaccess},%
9715   {longaccess}{longaccess},%
9716   {longpluralaccess}{longpluralaccess}%
9717 }
```

\@gls@noaccess Indicates that no replacement text has been provided.

```

9718 \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):

```

9719 \let\@gls@oldnewglossaryentryprehook\@newglossaryentryprehook
9720 \renewcommand*{\@newglossaryentryprehook}{%
9721   \@gls@oldnewglossaryentryprehook
9722   \def\@glo@access{\@glo@symbol}%}
```

Initialise the other keys:

```

9723 \def\@glo@textaccess{\@glo@access}%
9724 \def\@glo@firstaccess{\@glo@access}%
9725 \def\@glo@pluralaccess{\@glo@textaccess}%
9726 \def\@glo@firstpluralaccess{\@glo@pluralaccess}%
9727 \def\@glo@symbolaccess{\relax}%
9728 \def\@glo@symbolpluralaccess{\@glo@symbolaccess}%
9729 \def\@glo@descaccess{\relax}%
9730 \def\@glo@descpluralaccess{\@glo@descaccess}%
9731 \def\@glo@shortaccess{\relax}%
9732 \def\@glo@shortpluralaccess{\@glo@shortaccess}%
9733 \def\@glo@longaccess{\relax}%
9734 \def\@glo@longpluralaccess{\@glo@longaccess}%
9735 }
```

Add to the end hook:

```

9736 \let\@gls@oldnewglossaryentryposthook\@newglossaryentryposthook
9737 \renewcommand*{\@newglossaryentryposthook}{%
9738   \@gls@oldnewglossaryentryposthook}
```

Store the access information:

```

9739 \expandafter
9740 \protected@xdef\csname glo@\@glo@label @access\endcsname{%
9741 \@glo@access}%
9742 \expandafter
9743 \protected@xdef\csname glo@\@glo@label @textaccess\endcsname{%
9744 \@glo@textaccess}%
9745 \expandafter
9746 \protected@xdef\csname glo@\@glo@label @firstaccess\endcsname{%
9747 \@glo@firstaccess}%
9748 \expandafter
9749 \protected@xdef\csname glo@\@glo@label @pluralaccess\endcsname{%
9750 \@glo@pluralaccess}%
9751 \expandafter
9752 \protected@xdef\csname glo@\@glo@label @firstpluralaccess\endcsname{%
9753 \@glo@firstpluralaccess}%
9754 \expandafter
9755 \protected@xdef\csname glo@\@glo@label @symbolaccess\endcsname{%
9756 \@glo@symbolaccess}%
9757 \expandafter
9758 \protected@xdef\csname glo@\@glo@label @symbolpluralaccess\endcsname{%
9759 \@glo@symbolpluralaccess}%
9760 \expandafter
9761 \protected@xdef\csname glo@\@glo@label @descaccess\endcsname{%
9762 \@glo@descaccess}%
9763 \expandafter
9764 \protected@xdef\csname glo@\@glo@label @descpluralaccess\endcsname{%
9765 \@glo@descpluralaccess}%
9766 \expandafter
9767 \protected@xdef\csname glo@\@glo@label @shortaccess\endcsname{%
9768 \@glo@shortaccess}%
9769 \expandafter
9770 \protected@xdef\csname glo@\@glo@label @shortpluralaccess\endcsname{%
9771 \@glo@shortpluralaccess}%
9772 \expandafter
9773 \protected@xdef\csname glo@\@glo@label @longaccess\endcsname{%
9774 \@glo@longaccess}%
9775 \expandafter
9776 \protected@xdef\csname glo@\@glo@label @longpluralaccess\endcsname{%
9777 \@glo@longpluralaccess}%
9778 }

```

5.2 Accessing Replacement Text

`\glsentryaccess` Get the value of the access key for the entry with the given label:

```

9779 \newcommand*\glsentryaccess}[1]{%
9780 \@gls@entry@field{#1}{access}%
9781 }

```

`\glsentrytextaccess` Get the value of the `textaccess` key for the entry with the given label:

```

9782 \newcommand*{\glsentrytextaccess}[1]{%
9783   \@gls@entry@field{#1}{textaccess}%
9784 }

```

`\glsentryfirstaccess` Get the value of the `firstaccess` key for the entry with the given label:

```

9785 \newcommand*{\glsentryfirstaccess}[1]{%
9786   \@gls@entry@field{#1}{firstaccess}%
9787 }

```

`\glsentrypluralaccess` Get the value of the `pluralaccess` key for the entry with the given label:

```

9788 \newcommand*{\glsentrypluralaccess}[1]{%
9789   \@gls@entry@field{#1}{pluralaccess}%
9790 }

```

`\glsentryfirstpluralaccess` Get the value of the `firstpluralaccess` key for the entry with the given label:

```

9791 \newcommand*{\glsentryfirstpluralaccess}[1]{%
9792   \csname glo@#1@firstpluralaccess\endcsname
9793 }

```

`\glsentrysymbolaccess` Get the value of the `symbolaccess` key for the entry with the given label:

```

9794 \newcommand*{\glsentrysymbolaccess}[1]{%
9795   \@gls@entry@field{#1}{symbolaccess}%
9796 }

```

`\glsentrysymbolpluralaccess` Get the value of the `symbolpluralaccess` key for the entry with the given label:

```

9797 \newcommand*{\glsentrysymbolpluralaccess}[1]{%
9798   \@gls@entry@field{#1}{symbolpluralaccess}%
9799 }

```

`\glsentrydescaccess` Get the value of the `descriptionaccess` key for the entry with the given label:

```

9800 \newcommand*{\glsentrydescaccess}[1]{%
9801   \@gls@entry@field{#1}{descaccess}%
9802 }

```

`\glsentrydescpluralaccess` Get the value of the `descriptionpluralaccess` key for the entry with the given label:

```

9803 \newcommand*{\glsentrydescpluralaccess}[1]{%
9804   \@gls@entry@field{#1}{descaccess}%
9805 }

```

`\glsentryshortaccess` Get the value of the `shortaccess` key for the entry with the given label:

```

9806 \newcommand*{\glsentryshortaccess}[1]{%
9807   \@gls@entry@field{#1}{shortaccess}%
9808 }

```

ryshortpluralaccess Get the value of the shortpluralaccess key for the entry with the given label:

```
9809 \newcommand*{\glsentryshortpluralaccess}[1]{%
9810   \@gls@entry@field{#1}{shortpluralaccess}%
9811 }
```

\glsentrylongaccess Get the value of the longaccess key for the entry with the given label:

```
9812 \newcommand*{\glsentrylongaccess}[1]{%
9813   \@gls@entry@field{#1}{longaccess}%
9814 }
```

trylongpluralaccess Get the value of the longpluralaccess key for the entry with the given label:

```
9815 \newcommand*{\glsentrylongpluralaccess}[1]{%
9816   \@gls@entry@field{#1}{longpluralaccess}%
9817 }
```

\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.)

```
9818 \newcommand*{\glsaccsupp}[2]{%
9819   \BeginAccSupp{ActualText=#1}#2\EndAccSupp{}%
9820 }
```

\xglsaccsupp Fully expands replacement text before calling \glsaccsupp

```
9821 \newcommand*{\xglsaccsupp}[2]{%
9822   \protected@edef\@gls@replacementtext{#1}%
9823   \expandafter\glsaccsupp\expandafter{\@gls@replacementtext}{#2}%
9824 }
```

@gls@access@display

```
9825 \newcommand*{\@gls@access@display}[2]{%
9826   \protected@edef\@glo@access{#2}%
9827   \ifx\@glo@access\@gls@noaccess
9828     #1%
9829   \else
9830     \xglsaccsupp{\@glo@access}{#1}%
9831   \fi
9832 }
```

l\$nameaccessdisplay Displays the first argument with the accessibility text for the entry with the label given by the second argument (if set).

```
9833 \DeclareRobustCommand*{\glsnameaccessdisplay}[2]{%
9834   \@gls@access@display{#1}{\glsentryaccess{#2}}%
9835 }
```

l\$textaccessdisplay As above but for the textaccess replacement text.

```
9836 \DeclareRobustCommand*{\glstextaccessdisplay}[2]{%
9837   \@gls@access@display{#1}{\glsentrytextaccess{#2}}%
9838 }
```

pluralaccessdisplay As above but for the pluralaccess replacement text.

```

9839 \DeclareRobustCommand*\glspluralaccessdisplay}[2]{%
9840   \@gls@access@display{#1}{\glsentrypluralaccess{#2}}%
9841 }

```

firstaccessdisplay As above but for the firstaccess replacement text.

```

9842 \DeclareRobustCommand*\glsfirstaccessdisplay}[2]{%
9843   \@gls@access@display{#1}{\glsentryfirstaccess{#2}}%
9844 }

```

pluralaccessdisplay As above but for the firstpluralaccess replacement text.

```

9845 \DeclareRobustCommand*\glsfirstpluralaccessdisplay}[2]{%
9846   \@gls@access@display{#1}{\glsentryfirstpluralaccess{#2}}%
9847 }

```

symbolaccessdisplay As above but for the symbolaccess replacement text.

```

9848 \DeclareRobustCommand*\glsymbolaccessdisplay}[2]{%
9849   \@gls@access@display{#1}{\glsentrysymbolaccess{#2}}%
9850 }

```

pluralaccessdisplay As above but for the symbolpluralaccess replacement text.

```

9851 \DeclareRobustCommand*\glsymbolpluralaccessdisplay}[2]{%
9852   \@gls@access@display{#1}{\glsentrysymbolpluralaccess{#2}}%
9853 }

```

descriptionaccessdisplay As above but for the descriptionaccess replacement text.

```

9854 \DeclareRobustCommand*\glsdescriptionaccessdisplay}[2]{%
9855   \@gls@access@display{#1}{\glsentrydescaccess{#2}}%
9856 }

```

pluralaccessdisplay As above but for the descriptionpluralaccess replacement text.

```

9857 \DeclareRobustCommand*\glsdescriptionpluralaccessdisplay}[2]{%
9858   \@gls@access@display{#1}{\glsentrydescpluralaccess{#2}}%
9859 }

```

shortaccessdisplay As above but for the shortaccess replacement text.

```

9860 \DeclareRobustCommand*\glsshortaccessdisplay}[2]{%
9861   \@gls@access@display{#1}{\glsentryshortaccess{#2}}%
9862 }

```

pluralaccessdisplay As above but for the shortpluralaccess replacement text.

```

9863 \DeclareRobustCommand*\glsshortpluralaccessdisplay}[2]{%
9864   \@gls@access@display{#1}{\glsentryshortpluralaccess{#2}}%
9865 }

```

longaccessdisplay As above but for the longaccess replacement text.

```

9866 \DeclareRobustCommand*\glslongaccessdisplay}[2]{%
9867   \@gls@access@display{#1}{\glsentrylongaccess{#2}}%
9868 }

```


pluralaccessdisplay As above but for the longpluralaccess replacement text.

```
9869 \DeclareRobustCommand*\glslongpluralaccessdisplay}[2]{%
9870   \@gls@access@display{#1}{\glsentrylongpluralaccess{#2}}%
9871 }
```

\glsaccessdisplay Gets the replacement text corresponding to the named key given by the first argument and calls the appropriate command defined above.

```
9872 \DeclareRobustCommand*\glsaccessdisplay}[3]{%
9873   \@ifundefined{gls#1accessdisplay}%
9874   {%
9875     \PackageError{glossaries-accsupp}{No accessibility support
9876       for key ‘#1’}{}%
9877   }%
9878   {%
9879     \csname gls#1accessdisplay\endcsname{#2}{#3}%
9880   }%
9881 }
```

ls@default@entryfmt Redefine the default entry format to use accessibility information

```
9882 \renewcommand*\@@gls@default@entryfmt}[2]{%
9883   \ifdefempty\glscustomtext
9884   {%
9885     \glsifplural
9886     {%
```

Plural form

```
9887     \glscapscase
9888     {%
```

Don't adjust case

```
9889     \ifglsused\glslabel
9890     {%
```

Subsequent use

```
9891       #2{\glspluralaccessdisplay
9892         {\glsentryplural{\glslabel}}{\glslabel}}%
9893       {\glsdescriptionpluralaccessdisplay
9894         {\glsentrydescplural{\glslabel}}{\glslabel}}%
9895       {\glssymbolpluralaccessdisplay
9896         {\glsentrysymbolplural{\glslabel}}{\glslabel}}
9897       {\glsinsert}%
9898     }%
9899     {%
```

First use

```
9900       #1{\glsfirstpluralaccessdisplay
9901         {\glsentryfirstplural{\glslabel}}{\glslabel}}%
9902       {\glsdescriptionpluralaccessdisplay
9903         {\glsentrydescplural{\glslabel}}{\glslabel}}%
9904       {\glssymbolpluralaccessdisplay
```

```

9905         {\glentrysymbolplural{\glslabel}}{\glslabel}}%
9906     {\glsinsert}%
9907 }%
9908 }%
9909 {%

```

Make first letter upper case

```

9910     \ifglsused\glslabel
9911     {%

```

Subsequent use.

```

9912     #2{\glspluralaccessdisplay
9913         {\Glsentryplural{\glslabel}}{\glslabel}}%
9914     {\glsdescriptionpluralaccessdisplay
9915         {\glentrydescplural{\glslabel}}{\glslabel}}%
9916     {\glssymbolpluralaccessdisplay
9917         {\glentrysymbolplural{\glslabel}}{\glslabel}}%
9918     {\glsinsert}%
9919 }%
9920 {%

```

First use

```

9921     #1{\glsfirstpluralaccessdisplay
9922         {\Glsentryfirstplural{\glslabel}}{\glslabel}}%
9923     {\glsdescriptionpluralaccessdisplay
9924         {\glentrydescplural{\glslabel}}{\glslabel}}%
9925     {\glssymbolpluralaccessdisplay
9926         {\glentrysymbolplural{\glslabel}}{\glslabel}}%
9927     {\glsinsert}%
9928 }%
9929 }%
9930 {%

```

Make all upper case

```

9931     \ifglsused\glslabel
9932     {%

```

Subsequent use

```

9933     \MakeUppercase{%
9934     #2{\glspluralaccessdisplay
9935         {\glentryplural{\glslabel}}{\glslabel}}%
9936     {\glsdescriptionpluralaccessdisplay
9937         {\glentrydescplural{\glslabel}}{\glslabel}}%
9938     {\glssymbolpluralaccessdisplay
9939         {\glentrysymbolplural{\glslabel}}{\glslabel}}%
9940     {\glsinsert}}%
9941 }%
9942 {%

```

First use

```

9943     \MakeUppercase{%
9944     #1{\glsfirstpluralaccessdisplay

```

```

9945         {\glentryfirstplural{\glslabel}}{\glslabel}}%
9946     {\glsdescriptionpluralaccessdisplay
9947         {\glentrydescplural{\glslabel}}{\glslabel}}%
9948     {\glssymbolpluralaccessdisplay
9949         {\glentrysymbolplural{\glslabel}}{\glslabel}}%
9950     {\glinsert}}%
9951 }%
9952 }%
9953 }%
9954 {%

```

Singular form

```

9955     \glscapscase
9956     {%

```

Don't adjust case

```

9957     \ifglused\glslabel
9958     {%

```

Subsequent use

```

9959     #2{\glstextaccessdisplay
9960         {\glentrytext{\glslabel}}{\glslabel}}%
9961     {\glsdescriptionaccessdisplay
9962         {\glentrydesc{\glslabel}}{\glslabel}}%
9963     {\glssymbolaccessdisplay
9964         {\glentrysymbol{\glslabel}}{\glslabel}}%
9965     {\glinsert}}%
9966 }%
9967 {%

```

First use

```

9968     #1{\glsfirstaccessdisplay
9969         {\glentryfirst{\glslabel}}{\glslabel}}%
9970     {\glsdescriptionaccessdisplay
9971         {\glentrydesc{\glslabel}}{\glslabel}}%
9972     {\glssymbolaccessdisplay
9973         {\glentrysymbol{\glslabel}}{\glslabel}}%
9974     {\glinsert}}%
9975 }%
9976 }%
9977 {%

```

Make first letter upper case

```

9978     \ifglused\glslabel
9979     {%

```

Subsequent use

```

9980     #2{\glstextaccessdisplay
9981         {\Glentrytext{\glslabel}}{\glslabel}}%
9982     {\glsdescriptionaccessdisplay
9983         {\glentrydesc{\glslabel}}{\glslabel}}%
9984     {\glssymbolaccessdisplay

```

```

9985         {\glentrysymbol{\glslabel}}{\glslabel}}%
9986     {\glinsert}%
9987 }%
9988 {%

```

First use

```

9989     #1{\glfirstaccessdisplay
9990         {\glentryfirst{\glslabel}}{\glslabel}}%
9991     {\glsdescriptionaccessdisplay
9992         {\glentrydesc{\glslabel}}{\glslabel}}%
9993     {\glssymbolaccessdisplay
9994         {\glentrysymbol{\glslabel}}{\glslabel}}%
9995     {\glinsert}%
9996 }%
9997 }%
9998 {%

```

Make all upper case

```

9999     \ifglused\glslabel
10000     {%

```

Subsequent use

```

10001     \MakeUppercase{%
10002     #2{\glstextaccessdisplay
10003         {\glentrytext{\glslabel}}{\glslabel}}%
10004     {\glsdescriptionaccessdisplay
10005         {\glentrydesc{\glslabel}}{\glslabel}}%
10006     {\glssymbolaccessdisplay
10007         {\glentrysymbol{\glslabel}}{\glslabel}}%
10008     {\glinsert}}%
10009 }%
10010 {%

```

First use

```

10011     \MakeUppercase{%
10012     #1{\glfirstaccessdisplay
10013         {\glentryfirst{\glslabel}}{\glslabel}}%
10014     {\glsdescriptionaccessdisplay
10015         {\glentrydesc{\glslabel}}{\glslabel}}%
10016     {\glssymbolaccessdisplay
10017         {\glentrysymbol{\glslabel}}{\glslabel}}%
10018     {\glinsert}}%
10019 }%
10020 }%
10021 }%
10022 }%
10023 {%

```

Custom text provided in \glsdisp

```

10024     \ifglused{\glslabel}%
10025     {%

```

Subsequent use

```

10026      #2{\glscustomtext}%
10027      {\glsdescriptionaccessdisplay
10028       {\glsentrydesc{\glslabel}}{\glslabel}}%
10029      {\glssymbolaccessdisplay
10030       {\glsentrysymbol{\glslabel}}{\glslabel}}%
10031      {\glsinsert}%
10032    }%
10033    {%

```

First use

```

10034      #1{\glscustomtext}%
10035      {\glsdescriptionaccessdisplay
10036       {\glsentrydesc{\glslabel}}{\glslabel}}%
10037      {\glssymbolaccessdisplay
10038       {\glsentrysymbol{\glslabel}}{\glslabel}}%
10039      {\glsinsert}%
10040    }%
10041  }%
10042 }

```

`\glsгенentryfmt` Redefine to use accessibility information.

```

10043 \renewcommand*{\glsгенentryfmt}{%
10044   \ifdefempty\glscustomtext
10045   {%
10046     \glsifplural
10047     {%

```

Plural form

```

10048     \glscapscase
10049     {%

```

Don't adjust case

```

10050     \ifglsused\glslabel
10051     {%

```

Subsequent use

```

10052       \glspluralaccessdisplay
10053       {\glsentryplural{\glslabel}}{\glslabel}%
10054       \glsinsert
10055     }%
10056     {%

```

First use

```

10057       \glsfirstpluralaccessdisplay
10058       {\glsentryfirstplural{\glslabel}}{\glslabel}%
10059       \glsinsert
10060     }%
10061   }%
10062   {%

```

Make first letter upper case

```
10063      \ifglused\glslabel
10064      {%
```

Subsequent use.

```
10065      \glsppluralaccessdisplay
10066      {\Glentryplural{\glslabel}}{\glslabel}%
10067      \glinsert
10068      }%
10069      {%
```

First use

```
10070      \glfirstpluralaccessdisplay
10071      {\Glentryfirstplural{\glslabel}}{\glslabel}%
10072      \glinsert
10073      }%
10074      }%
10075      {%
```

Make all upper case

```
10076      \ifglused\glslabel
10077      {%
```

Subsequent use

```
10078      \glsppluralaccessdisplay
10079      {\mfirstucMakeUppercase{\glentryplural{\glslabel}}}%
10080      {\glslabel}%
10081      \mfirstucMakeUppercase{\glinsert}%
10082      }%
10083      {%
```

First use

```
10084      \glfirstpluralacessdisplay
10085      {\mfirstucMakeUppercase{\glentryfirstplural{\glslabel}}}%
10086      {\glslabel}%
10087      \mfirstucMakeUppercase{\glinsert}%
10088      }%
10089      }%
10090      }%
10091      {%
```

Singular form

```
10092      \glscapscase
10093      {%
```

Don't adjust case

```
10094      \ifglused\glslabel
10095      {%
```

Subsequent use

```
10096      \glstextaccessdisplay{\glentrytext{\glslabel}}{\glslabel}%
10097      \glinsert
```

10098 }%
 10099 {%

First use

10100 \glsfirstaccessdisplay{\glsentryfirst{\glslabel}}{\glslabel}%
 10101 \glsinsert
 10102 }%
 10103 }%
 10104 {%

Make first letter upper case

10105 \ifglsused\glslabel
 10106 {%

Subsequent use

10107 \glstextaccessdisplay{\Glsentrytext{\glslabel}}{\glslabel}%
 10108 \glsinsert
 10109 }%
 10110 {%

First use

10111 \glsfirstaccessdisplay{\Glsentryfirst{\glslabel}}{\glslabel}%
 10112 \glsinsert
 10113 }%
 10114 }%
 10115 {%

Make all upper case

10116 \ifglsused\glslabel
 10117 {%

Subsequent use

10118 \glstextaccessdisplay
 10119 {\mfirstucMakeUppercase{\glsentrytext{\glslabel}}}{\glslabel}%
 10120 \mfirstucMakeUppercase{\glsinsert}%
 10121 }%
 10122 {%

First use

10123 \glsfirstaccessdisplay
 10124 {\mfirstucMakeUppercase{\glsentryfirst{\glslabel}}}{\glslabel}%
 10125 \mfirstucMakeUppercase{\glsinsert}%
 10126 }%
 10127 }%
 10128 }%
 10129 }%
 10130 {%

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.

10131 \glscustomtext\glsinsert

```
10132 }%
10133 }
```

`\glsgenacfmt` Redefine to include accessibility information.

```
10134 \renewcommand*{\glsgenacfmt}{%
10135   \ifdefempty\glscustomtext
10136   {%
10137     \ifglused\glslabel
10138     {%
```

Subsequent use:

```
10139     \glsifplural
10140     {%
```

Subsequent plural form:

```
10141     \glscapscase
10142     {%
```

Subsequent plural form, don't adjust case:

```
10143     \acronymfont
10144     {\glsshortpluralaccessdisplay
10145      {\glentryshortpl{\glslabel}}{\glslabel}}%
10146     \glsinsert
10147     }%
10148     {%
```

Subsequent plural form, make first letter upper case:

```
10149     \acronymfont
10150     {\glsshortpluralaccessdisplay
10151      {\Glsentryshortpl{\glslabel}}{\glslabel}}%
10152     \glsinsert
10153     }%
10154     {%
```

Subsequent plural form, all caps:

```
10155     \mfirstucMakeUppercase
10156     {\acronymfont
10157     {\glsshortpluralaccessdisplay
10158      {\glentryshortpl{\glslabel}}{\glslabel}}%
10159     \glsinsert}%
10160     }%
10161     }%
10162     {%
```

Subsequent singular form

```
10163     \glscapscase
10164     {%
```

Subsequent singular form, don't adjust case:

```
10165     \acronymfont
10166     {\glsshortaccessdisplay{\glentryshort{\glslabel}}{\glslabel}}%
10167     \glsinsert
```


10168 }%
 10169 {%

Subsequent singular form, make first letter upper case:

10170 \acronymfont
 10171 {\glsshortaccessdisplay{\Glsentryshort{\glslabel}}{\glslabel}}%
 10172 \glsinsert
 10173 }%
 10174 {%

Subsequent singular form, all caps:

10175 \mfirstucMakeUppercase
 10176 {\acronymfont{%
 10177 \glsshortaccessdisplay{\Glsentryshort{\glslabel}}{\glslabel}}%
 10178 \glsinsert}%
 10179 }%
 10180 }%
 10181 }%
 10182 {%

First use:

10183 \glsifplural
 10184 {%

First use plural form:

10185 \glscapscase
 10186 {%

First use plural form, don't adjust case:

10187 \genplacrfullformat{\glslabel}{\glsinsert}%
 10188 }%
 10189 {%

First use plural form, make first letter upper case:

10190 \Genplacrfullformat{\glslabel}{\glsinsert}%
 10191 }%
 10192 {%

First use plural form, all caps:

10193 \mfirstucMakeUppercase
 10194 {\genplacrfullformat{\glslabel}{\glsinsert}}%
 10195 }%
 10196 }%
 10197 {%

First use singular form

10198 \glscapscase
 10199 {%

First use singular form, don't adjust case:

10200 \genacrfullformat{\glslabel}{\glsinsert}%
 10201 }%
 10202 {%

First use singular form, make first letter upper case:

```
10203      \Genacrfullformat{\glslabel}{\glsinsert}%
10204      }%
10205      {%
```

First use singular form, all caps:

```
10206      \mfirstucMakeUppercase
10207      {\genacrfullformat{\glslabel}{\glsinsert}}%
10208      }%
10209      }%
10210      }%
10211      }%
10212      {%
```

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.

```
10213      \glscustomtext
10214      }%
10215 }
```

`\genacrfullformat` Redefine to include accessibility information.

```
10216 \renewcommand*{\genacrfullformat}[2]{%
10217   \glslongaccessdisplay{\glsentrylong{#1}}{#1}#2\space
10218   (\glsshortaccessdisplay{\protect\firstacronymfont{\glsentryshort{#1}}}{#1})%
10219 }
```

`\Genacrfullformat` Redefine to include accessibility information.

```
10220 \renewcommand*{\Genacrfullformat}[2]{%
10221   \glslongaccessdisplay{\Glsentrylong{#1}}{#1}#2\space
10222   (\glsshortaccessdisplay{\protect\firstacronymfont{\Glsentryshort{#1}}}{#1})%
10223 }
```

`\genplacrfullformat` Redefine to include accessibility information.

```
10224 \renewcommand*{\genplacrfullformat}[2]{%
10225   \glslongpluralaccessdisplay{\glsentrylongpl{#1}}{#1}#2\space
10226   (\glsshortpluralaccessdisplay
10227     {\protect\firstacronymfont{\glsentryshortpl{#1}}}{#1})%
10228 }
```

`\Genplacrfullformat` Redefine to include accessibility information.

```
10229 \renewcommand*{\Genplacrfullformat}[2]{%
10230   \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}#2\space
10231   (\glsshortpluralaccessdisplay
10232     {\protect\firstacronymfont{\Glsentryshortpl{#1}}}{#1})%
10233 }
```

`\@acrshort`

```
10234 \def\@acrshort#1#2[#3]{%
10235   \glsdoifexists{#2}%
```

```

10236 {%
10237   \let\do@gl@link@checkfirsthyper\relax

10238   \let\gl@sifplural\@secondoftwo
10239   \let\gl@scapscase\@firstofthree
10240   \let\gl@insert\@empty
10241   \def\glscustomtext{%
10242     \acronymfont{\glsshortaccessdisplay{\glentryshort{#2}}{#2}}#3%
10243   }%

   Call \@gl@link
10244   \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
10245 }%

10246 \glspostlinkhook
10247 }

```

\@Acrshort

```

10248 \def\@Acrshort#1#2[#3]{%
10249   \glsoifexists{#2}%
10250   {%
10251     \let\do@gl@link@checkfirsthyper\relax

10252     \let\gl@sifplural\@secondoftwo
10253     \let\gl@scapscase\@secondofthree
10254     \let\gl@insert\@empty
10255     \def\glscustomtext{%
10256       \acronymfont{\glsshortaccessdisplay{\Glsentryshort{#2}}{#2}}#3%
10257     }%

     Call \@gl@link
10258     \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
10259   }%

10260   \glspostlinkhook
10261 }

```

\@ACRshort

```

10262 \def\@ACRshort#1#2[#3]{%
10263   \glsoifexists{#2}%
10264   {%
10265     \let\do@gl@link@checkfirsthyper\relax

10266     \let\gl@sifplural\@secondoftwo
10267     \let\gl@scapscase\@thirdofthree
10268     \let\gl@insert\@empty
10269     \def\glscustomtext{%
10270       \acronymfont{\glsshortaccessdisplay
10271         {\MakeUppercase{\glentryshort{#2}}}{#2}}#3%
10272     }%

```

```

    Call \@gls@link
10273   \@gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
10274   }%

10275   \glspostlinkhook
10276 }

\@acrlong
10277 \def\@acrlong#1#2[#3]{%
10278   \glsdoifexists{#2}%
10279   {%
10280     \let\do@gls@link@checkfirsthyper\relax

10281     \let\glsifplural\@secondoftwo
10282     \let\glscapscase\@firstofthree
10283     \let\glsinsert\@empty
10284     \def\glscustomtext{%
10285       \acronymfont{\glslongaccessdisplay{\glsentrylong{#2}}{#2}}#3%
10286     }%

    Call \@gls@link
10287   \@gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
10288   }%

10289   \glspostlinkhook
10290 }

\@Acrlong
10291 \def\@Acrlong#1#2[#3]{%
10292   \glsdoifexists{#2}%
10293   {%
10294     \let\do@gls@link@checkfirsthyper\relax

10295     \let\glsifplural\@secondoftwo
10296     \let\glscapscase\@firstofthree
10297     \let\glsinsert\@empty
10298     \def\glscustomtext{%
10299       \acronymfont{\glslongaccessdisplay{\Glsentrylong{#2}}{#2}}#3%
10300     }%

    Call \@gls@link
10301   \@gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
10302   }%

10303   \glspostlinkhook
10304 }

\@ACRlong
10305 \def\@ACRlong#1#2[#3]{%
10306   \glsdoifexists{#2}%
10307   {%
10308     \let\do@gls@link@checkfirsthyper\relax

```

```

10309 \let\glsifplural\@secondoftwo
10310 \let\glscapscase\@firstofthree
10311 \let\glsinsert\@empty
10312 \def\glscustomtext{%
10313     \acronymfont{\glslongaccessdisplay{%
10314         \MakeUppercase{\glsentrylong{#2}}}{#2}#3}%
10315     }%

    Call \@gls@link
10316     \@gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
10317 }%

10318 \glspostlinkhook
10319 }

```

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.

```

10320 \renewcommand*{\glossentryname}[1]{%
10321     \glsdoifexists{#1}%
10322     {%
10323         \glsnamefont{\glsnameaccessdisplay{\glsentryname{#1}}{#1}}%
10324     }%
10325 }

10326 \renewcommand*{\glossentryname}[1]{%
10327     \glsdoifexists{#1}%
10328     {%
10329         \glsnamefont{\glsnameaccessdisplay{\Glsentryname{#1}}{#1}}%
10330     }%
10331 }

10332 \renewcommand*{\glossentrydesc}[1]{%
10333     \glsdoifexists{#1}%
10334     {%
10335         \glsdescriptionaccessdisplay{\glsentrydesc{#1}}{#1}%
10336     }%
10337 }

10338 \renewcommand*{\Glossentrydesc}[1]{%
10339     \glsdoifexists{#1}%
10340     {%
10341         \glsdescriptionaccessdisplay{\Glsentrydesc{#1}}{#1}%
10342     }%
10343 }

```

```

10344 \renewcommand*{\glossentrysymbol}[1]{%
10345   \glsdoifexists{#1}%
10346   {%
10347     \glssymbolaccessdisplay{\glsentrysymbol{#1}}{#1}%
10348   }%
10349 }

10350 \renewcommand*{\Glossentrysymbol}[1]{%
10351   \glsdoifexists{#1}%
10352   {%
10353     \glssymbolaccessdisplay{\Glsentrysymbol{#1}}{#1}%
10354   }%
10355 }

```

pglossaryentryfield

```

10356 \newcommand*{\accsuppglossaryentryfield}[5]{%
10357   \glossaryentryfield{#1}%
10358   {\glsnameaccessdisplay{#2}{#1}}%
10359   {\glsdescriptionaccessdisplay{#3}{#1}}%
10360   {\glssymbolaccessdisplay{#4}{#1}}{#5}%
10361 }

```

glossarysubentryfield

```

10362 \newcommand*{\accsuppglossarysubentryfield}[6]{%
10363   \glossarysubentryfield{#1}{#2}%
10364   {\glsnameaccessdisplay{#3}{#2}}%
10365   {\glsdescriptionaccessdisplay{#4}{#2}}%
10366   {\glssymbolaccessdisplay{#5}{#2}}{#6}%
10367 }

```

5.4 Acronyms

Redefine acronym styles provided by glossaries:

long-short *<long>* (*<short>*) acronym style.

```

10368 \renewacronymstyle{long-short}%
10369 {%

```

Check for long form in case this is a mixed glossary.

```

10370   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
10371 }%
10372 {%
10373   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
10374   \renewcommand*{\genacrfullformat}[2]{%
10375     \glslongaccessdisplay{\glsentrylong{##1}}{##1}##2\space
10376     (\glsshortaccessdisplay
10377       {\protect\firstacronymfont{\glsentryshort{##1}}}{##1})%
10378   }%
10379   \renewcommand*{\Genacrfullformat}[2]{%
10380     \glslongaccessdisplay{\Glsentrylong{##1}}{##1}##2\space

```

```

10381 (\glsshortaccessdisplay
10382   {\protect\firstacronymfont{\glentryshort{##1}}{##1}})%
10383 }%
10384 \renewcommand*{\genplacrfullformat}[2]{%
10385   \glslongpluralaccessdisplay{\glentrylongpl{##1}}{##1}##2\space
10386   (\glsshortpluralaccessdisplay
10387     {\protect\firstacronymfont{\glentryshortpl{##1}}{##1}})%
10388 }%
10389 \renewcommand*{\Genplacrfullformat}[2]{%
10390   \glslongpluralaccessdisplay{\Glentrylongpl{##1}}{##1}##2\space
10391   (\glsshortpluralaccessdisplay
10392     {\protect\firstacronymfont{\glentryshortpl{##1}}{##1}})%
10393 }%
10394 \renewcommand*{\acronymentry}[1]{%
10395   \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}{##1}}
10396   \renewcommand*{\acronymshort}[2]{##1}%
10397   \renewcommand*{\acronymfont}[1]{##1}%
10398   \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
10399   \renewcommand*{\acrpluralsuffix}{\glpluralsuffix}%
10400 }

```

short-long <short> (<long>) acronym style.

```

10401 \renewacronymstyle{short-long}%
10402 {%
10403   Check for long form in case this is a mixed glossary.
10404   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
10405 }%
10406 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
10407 \renewcommand*{\genacrfullformat}[2]{%
10408   \glsshortaccessdisplay
10409     {\protect\firstacronymfont{\glentryshort{##1}}{##1}##2\space
10410     (\glslongaccessdisplay{\glentrylong{##1}}{##1}})%
10411 }%
10412 \renewcommand*{\Genacrfullformat}[2]{%
10413   \glsshortaccessdisplay
10414     {\protect\firstacronymfont{\Glentryshort{##1}}{##1}##2\space
10415     (\glslongaccessdisplay{\glentrylong{##1}}{##1}})%
10416 }%
10417 \renewcommand*{\genplacrfullformat}[2]{%
10418   \glsshortpluralaccessdisplay
10419     {\protect\firstacronymfont{\glentryshortpl{##1}}{##1}##2\space
10420     (\glslongpluralaccessdisplay
10421       {\glentrylongpl{##1}}{##1}})%
10422 }%
10423 \renewcommand*{\Genplacrfullformat}[2]{%
10424   \glsshortpluralaccessdisplay
10425     {\protect\firstacronymfont{\Glentryshortpl{##1}}{##1}##2\space
10426     (\glslongpluralaccessdisplay{\glentrylongpl{##1}}{##1}})%

```

```

10427 }%
10428 \renewcommand*{\acronymentry}[1]{%
10429   \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}{##1}}%
10430 \renewcommand*{\acronymfont}[2]{##1}%
10431 \renewcommand*{\acronymfont}[1]{##1}%
10432 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
10433 \renewcommand*{\acrpluralsuffix}{\glpluralsuffix}%
10434 }

```

long-short-desc *⟨long⟩* (*⟨short⟩*) acronym style that has an accompanying description (which the user needs to supply).

```

10435 \renewacronymstyle{long-short-desc}%
10436 {%
10437   \GlsUseAcrEntryDisplayStyle{long-short}%
10438 }%
10439 {%
10440   \GlsUseAcrStyleDefs{long-short}%
10441   \renewcommand*{\GenericAcronymFields}{}%
10442   \renewcommand*{\acronymfont}[2]{##2}%
10443   \renewcommand*{\acronymentry}[1]{%
10444     \glslongaccessdisplay{\glentrylong{##1}}{##1}\space
10445     (\glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}{##1}})%
10446 }

```

long-sc-short-desc *⟨long⟩* (\textsc{⟨short⟩}) acronym style that has an accompanying description (which the user needs to supply).

```

10447 \renewacronymstyle{long-sc-short-desc}%
10448 {%
10449   \GlsUseAcrEntryDisplayStyle{long-sc-short}%
10450 }%
10451 {%
10452   \GlsUseAcrStyleDefs{long-sc-short}%
10453   \renewcommand*{\GenericAcronymFields}{}%
10454   \renewcommand*{\acronymfont}[2]{##2}%
10455   \renewcommand*{\acronymentry}[1]{%
10456     \glslongaccessdisplay{\glentrylong{##1}}{##1}\space
10457     (\glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}{##1}})%
10458 }

```

long-sm-short-desc *⟨long⟩* (\textsmaller{⟨short⟩}) acronym style that has an accompanying description (which the user needs to supply).

```

10459 \renewacronymstyle{long-sm-short-desc}%
10460 {%
10461   \GlsUseAcrEntryDisplayStyle{long-sm-short}%
10462 }%
10463 {%
10464   \GlsUseAcrStyleDefs{long-sm-short}%
10465   \renewcommand*{\GenericAcronymFields}{}%
10466   \renewcommand*{\acronymfont}[2]{##2}%

```



```

10467 \renewcommand*{\acronymentry}[1]{%
10468     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10469     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10470 }

```

short-long-desc *<short>* (*{<long>}*) acronym style that has an accompanying description (which the user needs to supply).

```

10471 \renewacronymstyle{short-long-desc}%
10472 {%
10473     \GlsUseAcrEntryDispStyle{short-long}%
10474 }%
10475 {%
10476     \GlsUseAcrStyleDefs{short-long}%
10477     \renewcommand*{\GenericAcronymFields}{}%
10478     \renewcommand*{\acronymsort}[2]{##2}%
10479     \renewcommand*{\acronymentry}[1]{%
10480         \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10481         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10482 }

```

sc-short-long-desc *<long>* (\textsc{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

10483 \renewacronymstyle{sc-short-long-desc}%
10484 {%
10485     \GlsUseAcrEntryDispStyle{sc-short-long}%
10486 }%
10487 {%
10488     \GlsUseAcrStyleDefs{sc-short-long}%
10489     \renewcommand*{\GenericAcronymFields}{}%
10490     \renewcommand*{\acronymsort}[2]{##2}%
10491     \renewcommand*{\acronymentry}[1]{%
10492         \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10493         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10494 }

```

sm-short-long-desc *<long>* (\textsmaller{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

10495 \renewacronymstyle{sm-short-long-desc}%
10496 {%
10497     \GlsUseAcrEntryDispStyle{sm-short-long}%
10498 }%
10499 {%
10500     \GlsUseAcrStyleDefs{sm-short-long}%
10501     \renewcommand*{\GenericAcronymFields}{}%
10502     \renewcommand*{\acronymsort}[2]{##2}%
10503     \renewcommand*{\acronymentry}[1]{%
10504         \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10505         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10506 }

```

dua *⟨long⟩* only acronym style.

```
10507 \renewacronymstyle{dua}%
10508 {%
```

Check for long form in case this is a mixed glossary.

```
10509 \ifdefempty\glscustomtext
10510 {%
10511     \ifglshaslong{\glslabel}%
10512     {%
10513         \glsifplural
10514         {%
```

Plural form:

```
10515         \glscapscase
10516         {%
```

Plural form, don't adjust case:

```
10517         \glslongpluralaccessdisplay{\glsentrylongpl{\glslabel}}{\glslabel}%
10518         \glsinsert
10519     }%
10520     {%
```

Plural form, make first letter upper case:

```
10521         \glslongpluralaccessdisplay{\Glsentrylongpl{\glslabel}}{\glslabel}%
10522         \glsinsert
10523     }%
10524     {%
```

Plural form, all caps:

```
10525         \glslongpluralaccessdisplay
10526         {\mfirstucMakeUppercase{\glsentrylongpl{\glslabel}}}{\glslabel}%
10527         \mfirstucMakeUppercase{\glsinsert}%
10528     }%
10529     }%
10530     {%
```

Singular form

```
10531         \glscapscase
10532         {%
```

Singular form, don't adjust case:

```
10533         \glslongaccessdisplay{\glsentrylong{\glslabel}}{\glslabel}\glsinsert
10534     }%
10535     {%
```

Subsequent singular form, make first letter upper case:

```
10536         \glslongaccessdisplay{\Glsentrylong{\glslabel}}{\glslabel}\glsinsert
10537     }%
10538     {%
```

Subsequent singular form, all caps:

```
10539         \glslongaccessdisplay
10540         {\mfirstucMakeUppercase
```

```

10541         {\glsentrylong{\glslabel}\glsinsert}}{\glslabel}%
10542     \mfirstucMakeUppercase{\glsinsert}%
10543 }%
10544 }%
10545 }%
10546 {%

    Not an acronym:

10547     \glsgenentryfmt
10548 }%
10549 }%
10550 {\glscustomtext\glsinsert}%
10551 }%
10552 {%
10553 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
10554 \renewcommand*{\acrfullfmt}[3]{%
10555     \glslink[##1]{##2}{%
10556         \glslongaccessdisplay{\glsentrylong{##2}}{##2}##3\space
10557         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}}}%
10558 \renewcommand*{\Acrfullfmt}[3]{%
10559     \glslink[##1]{##2}{%
10560         \glslongaccessdisplay{\Glsentrylong{##2}}{##2}##3\space
10561         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}}}%
10562 \renewcommand*{\ACRfullfmt}[3]{%
10563     \glslink[##1]{##2}{%
10564         \glslongaccessdisplay
10565         {\mfirstucMakeUppercase{\glsentrylong{##2}}{##2}##3\space
10566         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}})}}%
10567 \renewcommand*{\acrfullplfmt}[3]{%
10568     \glslink[##1]{##2}{%
10569         \glslongpluralaccessdisplay
10570         {\glsentrylongpl{##2}}{##2}##3\space
10571         (\glsshortpluralaccessdisplay
10572         {\acronymfont{\glsentryshortpl{##2}}}{##2}}}%
10573 \renewcommand*{\Acrfullplfmt}[3]{%
10574     \glslink[##1]{##2}{%
10575         \glslongpluralaccessdisplay
10576         {\Glsentrylongpl{##2}}{##2}##3\space
10577         (\glsshortpluralaccessdisplay
10578         {\acronymfont{\glsentryshortpl{##2}}}{##2}}}%
10579 \renewcommand*{\ACRfullplfmt}[3]{%
10580     \glslink[##1]{##2}{%
10581         \glslongpluralaccessdisplay
10582         {\mfirstucMakeUppercase{\glsentrylongpl{##2}}{##2}##3\space
10583         (\glsshortpluralaccessdisplay
10584         {\acronymfont{\glsentryshortpl{##2}}}{##2}})}}%
10585 \renewcommand*{\glsentryfull}[1]{%
10586     \glslongaccessdisplay{\glsentrylong{##1}}\space
10587     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})%
10588 }%

```

```

10589 \renewcommand*{\Glsentryfull}[1]{%
10590   \glslongaccessdisplay{\Glsentrylong{##1}}{##1}\space
10591   (\glsshortaccessdisplay{\acronymfont{\Glsentryshort{##1}}}{##1})%
10592 }%
10593 \renewcommand*{\glsentryfullpl}[1]{%
10594   \glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}\space
10595   (\glsshortpluralaccessdisplay{\acronymfont{\glsentryshortpl{##1}}}{##1})%
10596 }%
10597 \renewcommand*{\Glsentryfullpl}[1]{%
10598   \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}\space
10599   (\glsshortpluralaccessdisplay{\acronymfont{\glsentryshortpl{##1}}}{##1})%
10600 }%
10601 \renewcommand*{\acronymentry}[1]{%
10602   \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##1}}}{##1}%
10603 \renewcommand*{\acronymsort}[2]{##1}%
10604 \renewcommand*{\acronymfont}[1]{##1}%
10605 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
10606 }

```

dua-desc *<long>* only acronym style with user-supplied description.

```

10607 \renewacronymstyle{dua-desc}%
10608 {%
10609   \GlsUseAcrEntryDispStyle{dua}%
10610 }%
10611 {%
10612   \GlsUseAcrStyleDefs{dua}%
10613   \renewcommand*{\GenericAcronymFields}{}%
10614   \renewcommand*{\acronymentry}[1]{%
10615     \glslongaccessdisplay{\acronymfont{\Glsentrylong{##1}}}{##1}}%
10616   \renewcommand*{\acronymsort}[2]{##2}%
10617 }%

```

footnote *<short>*\footnote{*<long>*} acronym style.

```

10618 \renewacronymstyle{footnote}%
10619 {%

```

Check for long form in case this is a mixed glossary.

```

10620   \ifglshaslong{\glslabel}{\glsngenacfmt}{\glsngenentryfmt}%
10621 }%
10622 {%
10623   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%

```

Need to ensure hyperlinks are switched off on first use:

```

10624   \glshyperfirstfalse
10625   \renewcommand*{\genacrfullformat}[2]{%
10626     \glsshortaccessdisplay
10627     {\protect\firstacronymfont{\Glsentryshort{##1}}}{##1}##2%
10628     \protect\footnote{\glslongaccessdisplay{\Glsentrylong{##1}}{##1}}%
10629   }%
10630   \renewcommand*{\Genacrfullformat}[2]{%

```

```

10631 \glsshortaccessdisplay
10632 {\firstacronymfont{\Glsentryshort{##1}}}{##1}##2%
10633 \protect\footnote{\glslongaccessdisplay{\glssentrylong{##1}}{##1}}%
10634 }%
10635 \renewcommand*{\genplacrfullformat}[2]{%
10636 \glsshortpluralaccessdisplay
10637 {\protect\firstacronymfont{\glssentryshortpl{##1}}}{##1}##2%
10638 \protect\footnote{\glslongpluralaccessdisplay{\glssentrylongpl{##1}}{##1}}%
10639 }%
10640 \renewcommand*{\Genplacrfullformat}[2]{%
10641 \glsshortpluralaccessdisplay
10642 {\protect\firstacronymfont{\Glsentryshortpl{##1}}}{##1}##2%
10643 \protect\footnote{\glslongpluralaccessdisplay{\glssentrylongpl{##1}}{##1}}%
10644 }%
10645 \renewcommand*{\acronymentry}[1]{%
10646 \glsshortaccessdisplay{\acronymfont{\glssentryshort{##1}}}{##1}}%
10647 \renewcommand*{\acronymsort}[2]{##1}%
10648 \renewcommand*{\acronymfont}[1]{##1}%
10649 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%

```

Don't use footnotes for \acrfull:

```

10650 \renewcommand*{\acrfullfmt}[3]{%
10651 \glslink[##1]{##2}{%
10652 \glsshortaccessdisplay{\acronymfont{\glssentryshort{##2}}}{##2}##3\space
10653 (\glslongaccessdisplay{\glssentrylong{##2}}{##2})}%
10654 \renewcommand*{\Acrfullfmt}[3]{%
10655 \glslink[##1]{##2}{%
10656 \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##2}}}{##2}##3\space
10657 (\glslongaccessdisplay{\glssentrylong{##2}}{##2})}%
10658 \renewcommand*{\ACRfullfmt}[3]{%
10659 \glslink[##1]{##2}{%
10660 \glsshortaccessdisplay
10661 {\mfirstucMakeUppercase
10662 {\acronymfont{\glssentryshort{##2}}}{##2}##3\space
10663 (\glslongaccessdisplay{\glssentrylong{##2}}{##2})}}}%
10664 \renewcommand*{\acrfullplfmt}[3]{%
10665 \glslink[##1]{##2}{%
10666 \glsshortpluralaccessdisplay
10667 {\acronymfont{\glssentryshortpl{##2}}}{##2}##3\space
10668 (\glslongpluralaccessdisplay{\glssentrylongpl{##2}}{##2})}%
10669 \renewcommand*{\Acrfullplfmt}[3]{%
10670 \glslink[##1]{##2}{%
10671 \glsshortpluralaccessdisplay
10672 {\acronymfont{\Glsentryshortpl{##2}}}{##2}##3\space
10673 (\glslongpluralaccessdisplay{\glssentrylongpl{##2}}{##2})}}}%
10674 \renewcommand*{\ACRfullplfmt}[3]{%
10675 \glslink[##1]{##2}{%
10676 \glsshortpluralaccessdisplay
10677 {\mfirstucMakeUppercase
10678 {\acronymfont{\glssentryshortpl{##2}}}{##2}##3\space

```

10679 (\glslongpluralaccessdisplay{\glsentrylongpl{##2}}{##2}})}%

Similarly for \glsentryfull etc:

```

10680 \renewcommand*{\glsentryfull}[1]{%
10681   \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}\space
10682   (\glslongaccessdisplay{\glsentrylong{##1}}{##1}})%
10683 \renewcommand*{\Glsentryfull}[1]{%
10684   \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##1}}}{##1}\space
10685   (\glslongaccessdisplay{\glsentrylong{##1}}{##1}})%
10686 \renewcommand*{\glsentryfullpl}[1]{%
10687   \glsshortpluralaccessdisplay
10688   {\acronymfont{\glsentryshortpl{##1}}}{##1}\space
10689   (\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}})%
10690 \renewcommand*{\Glsentryfullpl}[1]{%
10691   \glsshortpluralaccessdisplay
10692   {\acronymfont{\Glsentryshortpl{##1}}}{##1}\space
10693   (\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}})%
10694 }
```

footnote-sc \textsc{<short>}\footnote{<long>} acronym style.

```

10695 \renewacronymstyle{footnote-sc}%
10696 {%
10697   \GlsUseAcrEntryDispStyle{footnote}%
10698 }%
10699 {%
10700   \GlsUseAcrStyleDefs{footnote}%
10701   \renewcommand{\acronymentry}[1]{%
10702     \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}
10703   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
10704   \renewcommand*{\acrpluralsuffix}{\glstextup{\glspluralsuffix}}%
10705 }%
```

footnote-sm \textsmaller{<short>}\footnote{<long>} acronym style.

```

10706 \renewacronymstyle{footnote-sm}%
10707 {%
10708   \GlsUseAcrEntryDispStyle{footnote}%
10709 }%
10710 {%
10711   \GlsUseAcrStyleDefs{footnote}%
10712   \renewcommand{\acronymentry}[1]{%
10713     \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}
10714   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
10715   \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
10716 }%
```

footnote-desc <short>\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

10717 \renewacronymstyle{footnote-desc}%
10718 {%
```

```

10719 \GlsUseAcrEntryDispStyle{footnote}%
10720 }%
10721 {%
10722 \GlsUseAcrStyleDefs{footnote}%
10723 \renewcommand*{\GenericAcronymFields}{}%
10724 \renewcommand*{\acronymsort}[2]{##2}%
10725 \renewcommand*{\acronymentry}[1]{%
10726 \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10727 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10728 }

```

footnote-sc-desc \textsc{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

10729 \renewacronymstyle{footnote-sc-desc}%
10730 {%
10731 \GlsUseAcrEntryDispStyle{footnote-sc}%
10732 }%
10733 {%
10734 \GlsUseAcrStyleDefs{footnote-sc}%
10735 \renewcommand*{\GenericAcronymFields}{}%
10736 \renewcommand*{\acronymsort}[2]{##2}%
10737 \renewcommand*{\acronymentry}[1]{%
10738 \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10739 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10740 }

```

footnote-sm-desc \textsmaller{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

10741 \renewacronymstyle{footnote-sm-desc}%
10742 {%
10743 \GlsUseAcrEntryDispStyle{footnote-sm}%
10744 }%
10745 {%
10746 \GlsUseAcrStyleDefs{footnote-sm}%
10747 \renewcommand*{\GenericAcronymFields}{}%
10748 \renewcommand*{\acronymsort}[2]{##2}%
10749 \renewcommand*{\acronymentry}[1]{%
10750 \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10751 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10752 }

```

Use \newacronymhook to modify the key list to set the access text to the long version by default.

```

10753 \renewcommand*{\newacronymhook}{%
10754 \edef\@gls@keylist{shortaccess=\the\glslongtok,%
10755 \the\glskeylisttok}%
10756 \expandafter\glskeylisttok\expandafter{\@gls@keylist}%
10757 }

```

efaultNewAcronymDef Modify default style to use access text:

```

10758 \renewcommand*{\DefaultNewAcronymDef}{%
10759   \edef\@do@newglossaryentry{%
10760     \noexpand\newglossaryentry{\the\glslabeltok}%
10761     {%
10762       type=\acronymtype,%
10763       name={\the\glsshorttok},%
10764       description={\the\glslongtok},%
10765       descriptionaccess=\relax,
10766       text={\the\glsshorttok},%
10767       access={\noexpand\@glo@textaccess},%
10768       sort={\the\glsshorttok},%
10769       short={\the\glsshorttok},%
10770       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10771       shortaccess={\the\glslongtok},%
10772       long={\the\glslongtok},%
10773       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10774       descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10775       first={\noexpand\glslongaccessdisplay
10776         {\the\glslongtok}{\the\glslabeltok}\space
10777         (\noexpand\glsshortaccessdisplay
10778           {\the\glsshorttok}{\the\glslabeltok})},%
10779       plural={\the\glsshorttok\acrpluralsuffix},%
10780       firstplural={\noexpand\glslongpluralaccessdisplay
10781         {\noexpand\@glo@longpl}{\the\glslabeltok}\space
10782         (\noexpand\glsshortpluralaccessdisplay
10783           {\noexpand\@glo@shortpl}{\the\glslabeltok})},%
10784       firstaccess=\relax,
10785       firstpluralaccess=\relax,
10786       textaccess={\noexpand\@glo@shortaccess},%
10787       \the\glskeylisttok
10788     }%
10789   }%
10790   \let\@org@gls@assign@firstpl\gls@assign@firstpl
10791   \let\@org@gls@assign@plural\gls@assign@plural
10792   \let\@org@gls@assign@descplural\gls@assign@descplural
10793   \def\gls@assign@firstpl##1##2{%
10794     \@@gls@expand@field{##1}{firstpl}{##2}%
10795   }%
10796   \def\gls@assign@plural##1##2{%
10797     \@@gls@expand@field{##1}{plural}{##2}%
10798   }%
10799   \def\gls@assign@descplural##1##2{%
10800     \@@gls@expand@field{##1}{descplural}{##2}%
10801   }%
10802   \@do@newglossaryentry
10803   \let\gls@assign@firstpl\@org@gls@assign@firstpl
10804   \let\gls@assign@plural\@org@gls@assign@plural
10805   \let\gls@assign@symbolplural\@org@gls@assign@symbolplural

```


10806 }

otnoteNewAcronymDef

```
10807 \renewcommand*{\DescriptionFootnoteNewAcronymDef}{%
10808   \edef\@do@newglossaryentry{%
10809     \noexpand\newglossaryentry{\the\glslabeltok}%
10810     {%
10811       type=\acronymtype,%
10812       name={\noexpand\acronymfont{\the\glsshorttok}},%
10813       sort={\the\glsshorttok},%
10814       text={\the\glsshorttok},%
10815       short={\the\glsshorttok},%
10816       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10817       shortaccess={\the\glslongtok},%
10818       long={\the\glslongtok},%
10819       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10820       access={\noexpand\@glo@textaccess},%
10821       plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10822       symbol={\the\glslongtok},%
10823       symbolplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10824       firstpluralaccess=\relax,
10825       textaccess={\noexpand\@glo@shortaccess},%
10826       \the\glskeylisttok
10827     }%
10828   }%
10829   \let\@org@gls@assign@firstpl\gls@assign@firstpl
10830   \let\@org@gls@assign@plural\gls@assign@plural
10831   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
10832   \def\gls@assign@firstpl##1##2{%
10833     \@@gls@expand@field{##1}{firstpl}{##2}%
10834   }%
10835   \def\gls@assign@plural##1##2{%
10836     \@@gls@expand@field{##1}{plural}{##2}%
10837   }%
10838   \def\gls@assign@symbolplural##1##2{%
10839     \@@gls@expand@field{##1}{symbolplural}{##2}%
10840   }%
10841   \@do@newglossaryentry
10842   \let\gls@assign@plural\@org@gls@assign@plural
10843   \let\gls@assign@firstpl\@org@gls@assign@firstpl
10844   \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
10845 }
```

ipitionNewAcronymDef

```
10846 \renewcommand*{\DescriptionNewAcronymDef}{%
10847   \edef\@do@newglossaryentry{%
10848     \noexpand\newglossaryentry{\the\glslabeltok}%
10849     {%
10850       type=\acronymtype,%
```

```

10851     name={\noexpand
10852         \acrnameformat{\the\glsshorttok}{\the\glslongtok}},%
10853     access={\noexpand\@glo@textaccess},%
10854     sort={\the\glsshorttok},%
10855     short={\the\glsshorttok},%
10856     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10857     shortaccess={\the\glslongtok},%
10858     long={\the\glslongtok},%
10859     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10860     first={\the\glslongtok},%
10861     firstaccess=\relax,
10862     firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10863     text={\the\glsshorttok},%
10864     textaccess={\the\glslongtok},%
10865     plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10866     symbol={\noexpand\@glo@text},%
10867     symbolaccess={\noexpand\@glo@textaccess},%
10868     symbolplural={\noexpand\@glo@plural},%
10869     firstpluralaccess=\relax,
10870     textaccess={\noexpand\@glo@shortaccess},%
10871     \the\glskeylisttok}%
10872 }%
10873 \let\@org@gls@assign@firstpl\gls@assign@firstpl
10874 \let\@org@gls@assign@plural\gls@assign@plural
10875 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
10876 \def\gls@assign@firstpl##1##2{%
10877     \@@gls@expand@field{##1}{firstpl}{##2}%
10878 }%
10879 \def\gls@assign@plural##1##2{%
10880     \@@gls@expand@field{##1}{plural}{##2}%
10881 }%
10882 \def\gls@assign@symbolplural##1##2{%
10883     \@@gls@expand@field{##1}{symbolplural}{##2}%
10884 }%
10885 \do@newglossaryentry
10886 \let\gls@assign@firstpl\@org@gls@assign@firstpl
10887 \let\gls@assign@plural\@org@gls@assign@plural
10888 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
10889 }

```

otnoteNewAcronymDef

```

10890 \renewcommand*{\FootnoteNewAcronymDef}{%
10891     \edef\@do@newglossaryentry{%
10892         \noexpand\newglossaryentry{\the\glslabeltok}%
10893         {%
10894             type=\acronymtype,%
10895             name={\noexpand\acronymfont{\the\glsshorttok}},%
10896             sort={\the\glsshorttok},%
10897             text={\the\glsshorttok},%

```

```

10898     textaccess={\the\glslongtok},%
10899     access={\noexpand\@glo@textaccess},%
10900     plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10901     short={\the\glsshorttok},%
10902     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10903     long={\the\glslongtok},%
10904     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10905     description={\the\glslongtok},%
10906     descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10907     \the\glskeylisttok
10908   }%
10909 }%
10910 \let\@org@gls@assign@plural\gls@assign@plural
10911 \let\@org@gls@assign@firstpl\gls@assign@firstpl
10912 \let\@org@gls@assign@descplural\gls@assign@descplural
10913 \def\gls@assign@firstpl##1##2{%
10914   \@@gls@expand@field{##1}{firstpl}{##2}%
10915 }%
10916 \def\gls@assign@plural##1##2{%
10917   \@@gls@expand@field{##1}{plural}{##2}%
10918 }%
10919 \def\gls@assign@descplural##1##2{%
10920   \@@gls@expand@field{##1}{descplural}{##2}%
10921 }%
10922 \do@newglossaryentry
10923 \let\gls@assign@plural\@org@gls@assign@plural
10924 \let\gls@assign@firstpl\@org@gls@assign@firstpl
10925 \let\gls@assign@descplural\@org@gls@assign@descplural
10926 }

```

\SmallNewAcronymDef

```

10927 \renewcommand*{\SmallNewAcronymDef}{%
10928   \edef\@do@newglossaryentry{%
10929     \noexpand\newglossaryentry{\the\glslabeltok}%
10930     {%
10931       type=\acronymtype,%
10932       name={\noexpand\acronymfont{\the\glsshorttok}},%
10933       access={\noexpand\@glo@symbolaccess},%
10934       sort={\the\glsshorttok},%
10935       short={\the\glsshorttok},%
10936       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10937       shortaccess={\the\glslongtok},%
10938       long={\the\glslongtok},%
10939       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10940       text={\noexpand\@glo@short},%
10941       textaccess={\noexpand\@glo@shortaccess},%
10942       plural={\noexpand\@glo@shortpl},%
10943       first={\the\glslongtok},%
10944       firstaccess=\relax,

```

```

10945     firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10946     description={\noexpand\@glo@first},%
10947     descriptionplural={\noexpand\@glo@firstplural},%
10948     symbol={\the\glsshorttok},%
10949     symbolaccess={\the\glslongtok},%
10950     symbolplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10951     \the\glskeylisttok
10952 }%
10953 }%
10954 \let\@org@gls@assign@firstpl\gls@assign@firstpl
10955 \let\@org@gls@assign@plural\gls@assign@plural
10956 \let\@org@gls@assign@descplural\gls@assign@descplural
10957 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
10958 \def\gls@assign@firstpl##1##2{%
10959   \@@gls@expand@field{##1}{firstpl}{##2}%
10960 }%
10961 \def\gls@assign@plural##1##2{%
10962   \@@gls@expand@field{##1}{plural}{##2}%
10963 }%
10964 \def\gls@assign@descplural##1##2{%
10965   \@@gls@expand@field{##1}{descplural}{##2}%
10966 }%
10967 \def\gls@assign@symbolplural##1##2{%
10968   \@@gls@expand@field{##1}{symbolplural}{##2}%
10969 }%
10970 \do@newglossaryentry
10971 \let\gls@assign@firstpl\@org@gls@assign@firstpl
10972 \let\gls@assign@plural\@org@gls@assign@plural
10973 \let\gls@assign@descplural\@org@gls@assign@descplural
10974 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
10975 }

```

The following are kept for compatibility with versions before 3.0:

```

\glsshortaccesskey
10976   \newcommand*{\glsshortaccesskey}{\glsshortkey access}%

hortpluralaccesskey
10977   \newcommand*{\glsshortpluralaccesskey}{\glsshortpluralkey access}%

\glslongaccesskey
10978   \newcommand*{\glslongaccesskey}{\glslongkey access}%

longpluralaccesskey
10979   \newcommand*{\glslongpluralaccesskey}{\glslongpluralkey access}%

```

5.5 Debugging Commands

```

\showglongnameaccess

```

```

10980 \newcommand*{\showglonameaccess}[1]{%
10981   \expandafter\show\csname glo@\glsdetoklabel{#1}@textaccess\endcsname
10982 }

\showglotextaccess

10983 \newcommand*{\showglotextaccess}[1]{%
10984   \expandafter\show\csname glo@\glsdetoklabel{#1}@textaccess\endcsname
10985 }

showglopluralaccess

10986 \newcommand*{\showglopluralaccess}[1]{%
10987   \expandafter\show\csname glo@\glsdetoklabel{#1}@pluralaccess\endcsname
10988 }

\showglofirstaccess

10989 \newcommand*{\showglofirstaccess}[1]{%
10990   \expandafter\show\csname glo@\glsdetoklabel{#1}@firstaccess\endcsname
10991 }

lofirstpluralaccess

10992 \newcommand*{\showglofirstpluralaccess}[1]{%
10993   \expandafter\show\csname glo@\glsdetoklabel{#1}@firstpluralaccess\endcsname
10994 }

showglosymbolaccess

10995 \newcommand*{\showglosymbolaccess}[1]{%
10996   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolaccess\endcsname
10997 }

osymbolpluralaccess

10998 \newcommand*{\showglosymbolpluralaccess}[1]{%
10999   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolpluralaccess\endcsname
11000 }

\showglodescaccess

11001 \newcommand*{\showglodescaccess}[1]{%
11002   \expandafter\show\csname glo@\glsdetoklabel{#1}@descaccess\endcsname
11003 }

glodescpluralaccess

11004 \newcommand*{\showglodescpluralaccess}[1]{%
11005   \expandafter\show\csname glo@\glsdetoklabel{#1}@descpluralaccess\endcsname
11006 }

\showgloshortaccess

11007 \newcommand*{\showgloshortaccess}[1]{%
11008   \expandafter\show\csname glo@\glsdetoklabel{#1}@shortaccess\endcsname
11009 }

```

loshortpluralaccess

```
11010 \newcommand*{\showgloshortpluralaccess}[1]{%
11011   \expandafter\show\csname glo@\glsdetoklabel{#1}@shortpluralaccess\endcsname
11012 }
```

\showglolongaccess

```
11013 \newcommand*{\showglolongaccess}[1]{%
11014   \expandafter\show\csname glo@\glsdetoklabel{#1}@longaccess\endcsname
11015 }
```

glolongpluralaccess

```
11016 \newcommand*{\showglolongpluralaccess}[1]{%
11017   \expandafter\show\csname glo@\glsdetoklabel{#1}@longpluralaccess\endcsname
11018 }
```

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.

```
11019 \NeedsTeXFormat{LaTeX2e}
11020 \ProvidesPackage{glossaries-babel}[2015/11/22 v4.19 (NLCT)]
```

Load tracklang to obtain language settings.

```
11021 \RequirePackage{tracklang}
11022 \let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
11023 \AnyTrackedLanguages
11024 {%
11025   \ForEachTrackedDialect{\this@dialect}{%
11026     \IfTrackedLanguageFileExists{\this@dialect}%
11027     {glossaries-}% prefix
11028     {.ldf}%
11029     {%
11030       \RequireGlossariesLang{\CurrentTrackedTag}%
11031     }%
11032   }%
11033   \PackageWarningNoLine{glossaries}%
11034   {No language module detected for ‘\this@dialect’.\MessageBreak
11035   Language modules need to be installed separately.\MessageBreak
11036   Please check on CTAN for a bundle called\MessageBreak
11037   ‘glossaries-\CurrentTrackedLanguage’ or similar}%
11038 }%
11039 }%
11040 }%
11041 }
```

6.1 Polyglossia Captions

Language support has now been split off into independent language modules.

```
11042 \NeedsTeXFormat{LaTeX2e}
```

```
11043 \ProvidesPackage{glossaries-polyglossia}[2015/11/22 v4.19 (NLCT)]
```

Load tracklang to obtain language settings.

```
11044 \RequirePackage{tracklang}
```

```
11045 \let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
11046 \AnyTrackedLanguages
```

```
11047 {%
```

```
11048 \ForEachTrackedDialect{\this@dialect}{%
```

```
11049 \IfTrackedLanguageFileExists{\this@dialect}%
```

```
11050 {glossaries-}% prefix
```

```
11051 {.ldf}%
```

```
11052 {%
```

```
11053 \RequireGlossariesLang{\CurrentTrackedTag}%
```

```
11054 }%
```

```
11055 {%
```

```
11056 \PackageWarningNoLine{glossaries}%
```

```
11057 {No language module detected for ‘\this@dialect’.\MessageBreak
```

```
11058 Language modules need to be installed separately.\MessageBreak
```

```
11059 Please check on CTAN for a bundle called\MessageBreak
```

```
11060 ‘glossaries-\CurrentTrackedLanguage’ or similar}%
```

```
11061 }%
```

```
11062 }%
```

```
11063 }%
```

```
11064 {}%
```

Glossary

makeindex An indexing application. [10](#), [25](#), [26](#), [171](#)

xindy An flexible indexing application with multilingual support written in Perl. [10](#), [25](#), [26](#), [171](#)

Change History

??	format key	109
super: fixed typo in \subglossentry		
(\glossentrydesc)	281	
1.01	\writeist: Added spaces after	
	\delimN and \delimR in ist	
General: Added range facility in	file	157

- 1.04
General: Added `\glstextformat` 93
- 1.05
`\glossarysection:` added
`\@mkboth to \glossarysection`
..... 37
`\gls@defglossaryentry:`
Changed the default value of
the sort key to just the value of
the name key 77
- 1.07
`\@gls@link:` fixed bug caused by
`\theglsentrycounter` set-
ting the page number too soon
..... 107
`\glsadd:` fixed bug caused by
`\theglsentrycounter` set-
ting the page number too soon
..... 154
- 1.08
General: Added babel support ... 31
`listgroup:` changed `listgroup`
style to use `\glsgetgrouptitle`
..... 263
`altlistgroup:` changed `al-`
`tlistgroup` style to use
`\glsgetgrouptitle` 264
- 1.1
`\@glossarysection:` numbered
sections and auto label added 38
`\@gls@tmpb:` changed `\toksdef`
to `\newtoks` 111
`\@gls@toc:` numberline added .. 40
`\@p@glossarysection:` num-
bered sections and auto label
added 39
General: `amsgen` now loaded
(`\new@ifnextchar` needed) .. 4
`translate:` `translate` option
added 22
`\setglossarysection:` new ... 38
`numberedsection:` numbered-
section package option added . 6
`numberline:` `numberline` option
added 5
- 1.12
`\@GLSp1@:` now uses `\glentrydescplural`
and `\glentrysymbolplural`
instead of `\glentrydesc`
and `\glentrysymbol` 123
`\@GLSp1@:` now uses `\glentrydescplural`
and `\glentrysymbolplural`
instead of `\glentrydesc`
and `\glentrysymbol` 122
`\@GLSp1@:` now uses `\glentrydescplural`
and `\glentrysymbolplural`
instead of `\glentrydesc`
and `\glentrysymbol` 121
General: added check for
`\hypertarget` separate to
`\hyperlink` (memoir de-
fines `\hyperlink` but not
`\hypertarget`) 117
`descriptionplural:` new 60
`\gls@defglossaryentry:`
Changed default first plural to
be first key with s appended
(was text key with s appended) 77
`descriptionplural` support
added 76
`symbolplural` support added .. 77
`\Glsentrydescplural:` New .. 148
`\glentrydescplural:` New .. 148
`\Glsentrysymbolplural:` New 149
`\glentrysymbolplural:` New 149
`\SetDescriptionFootnoteAcronymStyle:`
Added `\protect` before
`\footnote` and `\glslink` . 231
`\SetFootnoteAcronymStyle:`
Added `\protect` before
`\footnote` and `\glslink` . 237
`symbolplural:` new 61
- 1.13
General: fixed bug that ignored 3rd
parameter 125–132
`\ACRfullpl:` new 212
`\Acrfullpl:` new 211
`\acrfullpl:` new 211
`\acrpluralsuffix:` New 209
`\gls@defglossaryentry:`
Changed default first value .. 77
Changed default firstplural
value 77
Removed restriction on only
using `\newglossaryentry` in
the preamble 82

\newacronym: Removed restriction on only using \newacronym in the preamble	209	\@gls@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	119
1.14		\@glsdisp: Test glossary type is \acronymtype in addition to checking if footnote option has been used	124
\@gls@hypergroup: new	258	\@glspl@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	121
General: added nonnumberlist key to \printglossary	195	\@gls@defglossaryentry: Changed def to let	77
added numberedsection key to \printglossary	193	1.17	
\firstacronymfont: new	212	\@@do@wrglossary: new	174
\glsautoprefix: new	6	\@do@seeglossary: new	177
\glsnavhyperlink: changed 'edef to 'protected@edef	258	\@glo@storeentry: new	83
\glsnavhypertarget: added write to aux file	258	\@gls@glossary: changed definition to use \index instead of \@index	172
\glsnavigation: changed to only use labels for groups that are present	259	\@glsdefaultplural: new	64
1.15		\@glsdefaultsort: new	64
\@gls@link: added \glslabel	107	\@gls@hypernumber: new	205
\gls@defglossaryentry: check for \@glo@first in description	80	\@gls@noname: new	64
check for \@glo@text in symbol	81	\@gls@nonextpages: new	195
\gls@hypergroup: new	258	General: added xindy support	25
\glsnavhypertarget: added check if rerun required	258	parent: new	62
\glssettoctitle: new	30	see: new	62
\printglossary: changed the way the TOC title is set	179	\gls@defglossaryentry: added nonnumberlist key	77
1.16		added parent key	77
\@GLS@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	120	added see key	77
\@GLSpl: Test glossary type is \acronymtype in addition to checking if footnote option has been used	123	Stored main part of entry format when entry is defined	81
\@Gls@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	120	\gls@suffixF: new	35
\@Glspl@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	122	\gls@suffixFF: new	35
		\gls@wrglossary: modified to allow for xindy support	172
		\gls@hyperlink: new	154
		\gls@hypernumber: modified to allow material to be attached to location	205
		\glsnavhyperlink: replaced 'hyperlink to '@glslink	258

\glsnavhypertarget: replaced hypertarget to '@gls@target' . 258	use \textsmaller instead of \smaller 231
\glssee: new 177	\SetFootnoteAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller 237
\glsseeformat: new 178	\SetSmallAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller 240
\glsSetSuffixF: new 35	
\glsSetSuffixFF: new 35	
\ifglxindy: new 25	
\istfilename: added xindy sup- port 34	
\newglossarystyle: made \newglossarystyle long . 204	2.01
\nopostdesc: new 33	\@gls@link: moved \@do@wrglossary before term is displayed to pre- vent unwanted whatsit 107
nonumberlist: new 62	\foralllglossaries: replaced \ifthenelse with \ifx 49
\printglossary: added check to determine if \printglossary is already defined 179	\forallglsentries: replaced \ifthenelse with \ifx 50
added print language to aux file 179	\glsdefmain: new 12
order: order package option added 24	\glsdescwidth: changed \linewidth to \hsize . 265, 281
\writeist: added xindy support 157	\glslistdottedwidth: changed \linewidth to \hsize 265
1.18	\glslistwidth: changed \linewidth to \hsize . 266, 281
\@gls@loadlist: new 8	nomain: added nomain package option 13
\@gls@loadlong: new 8	\writeist: removed item_02 - no such makeindex key 161
\@gls@loadsuper: new 8	2.02
\@gls@loadtree: new 8	\@printglossary: suppressed warning globally rather than locally 182
\gls@defglossaryentry: Changed default value of sort to \glsdefaultsort 77	\glossarysection: changed \@mkboth to \glossarymark 37
moved sort sanitization to \newglossaryentry 81	\gls@defglossaryentry: New 37
\gls@target: new 198	2.03
\oldacronym: new 208	\@GLS@: Added check for hyper- first 120
nolist: new 8	\@GLS@p1: Added check for hyper- first 123
nolong: new 8	\@GLS@: Added check for hyper- first 120
sort: moved sanitization to \newglossaryentry 60	\@GLS@p1@: Added check for hyper- first 122
nostyles: new 8	\@gls@: Added check for hyper- first 119
nosuper: new 8	\@gls@@link: new 105
notree: new 8	
1.19	
\gls@clearpage: new 40	
\glsdisp: new 123	
\SetDescriptionAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller 235	
\SetDescriptionFootnoteAcronymStyle: changed \acronymfont to	

\@gls@link: added \leavevmode	107	\glsentryuseri: new	150
Moved entry existence check to avoid duplicate code	107	\Glsentryuserii: new	150
\@glsdisp: Added check for hyperfirst	124	\glsentryuserii: new	150
\@glspl@: Added check for hyperfirst	121	\Glsentryuseriii: new	150
\gls glossarymark: Added check to see if it's already defined ..	37	\glsentryuseriii: new	150
hyperfirst: new	23	\Glsentryuseriv: new	151
2.04		\glsentryuseriv: new	150
\@GLS@: Changed test to check if glossary type has been identified as a list of acronyms ...	120	\Glsentryuserv: new	151
\@GLSpl: Changed test to check if glossary type has been identified as a list of acronyms ...	123	\glsentryuserv: new	151
\@GLs@: Changed test to check if glossary type has been identified as a list of acronyms ...	120	\Glsentryuservi: new	151
\@GLspl@: Changed test to check if glossary type has been identified as a list of acronyms ..	122	\glsentryuservi: new	151
\@glossaryentryfield: new ..	82	\ns@newglossary: added check to determine if \gls@<type>@display and \gls@<type>@displayfirst have been defined.	57
\@glossarysubentryfield: new	82	\SetAcronymLists: new	15
\@gls@: Changed test to check if glossary type has been identified as a list of acronyms ...	119	\SetDefaultAcronymDisplayStyle: new	227
\@glsacronymlists: new	14	\SetDefaultAcronymStyle: new	228
\@glsdisp: Changed test to check if glossary type has been identified as a list of acronyms ..	124	\SetDescriptionAcronymDisplayStyle: new	233
\@glspl@: Changed test to check if glossary type has been identified as a list of acronyms ..	121	\SetDescriptionDUAAcronymDisplayStyle: new	231
\@newglossaryentryposthook: new	82	\SetDescriptionFootnoteAcronymDisplayStyle: new	229
\@newglossaryentryprehook: new	82	\SetDUADisplayStyle: new ..	241
acronymlists: new	15	\SetFootnoteAcronymDisplayStyle: new	235
\DeclareAcronymList: new ...	14	\SetSmallAcronymDisplayStyle: new	238
\DefineAcronymSynonyms: new	225	2.05	
\gls@defglossaryentry: added user1-6 keys	77	\@glsdisp: Added closing brace. Patch provided by Sergiu Dotenco	123
\glsadd: fixed bug that ignored counter	154	Removed spurious brace. Patch provided by Sergiu Dotenco	124
\Glsentryuseri: new	150	\writeist: Added \string before opening and closing braces. Patch provided by Segiu Dotenco	161
		2.06	
		\altnewglossary: new	58
		\CustomAcronymFields: new ..	243
		\CustomNewAcronymDef: new ..	243
		\SetCustomDisplayStyle: new	243
		\SetCustomStyle: new	244

2.07		\acrfull: added starred version	209
General: glssadd format key stored in \@glsnumberformat (was mistakenly stored in \@glo@format)	154	\ACRfullpl: added starred ver- sion	212
3.0		\Acrfullpl: added starred ver- sion	211
\@do@wrglossary: added check for hyper location prefix ...	175	\acrfullpl: added starred ver- sion	211
modified to use new format ..	174	\acrlinkfootnote: new	229
\@glossarysec: replaced \@ifundefined with \ifcsundef	5	\acrnolinkfootnote: new ...	229
\@do@seeglossary: Sanitize and escape cross-referencing in- formation	177	savewrites: new	26
\@gls@counterwithin: new	9	see: added \@glo@seeautonumberlist	62
\@gls@ifinlist: new	41	seeautonumberlist: new	8
\@gls@link: added \@gls@saveentrycounter	107	\glossarysection: replaced \@ifundefined with \ifcsundef	37
added \@gls@setsort	107	\glossarystyle: replaced \@ifundefined with \ifcsundef	204
\@gls@saveentrycounter: new	108	\gls@codepage: replaced \@ifundefined with \ifcsundef	25
\@gls@setupsort@def: new ...	11	\gls@defglossaryentry: added \@gls@defsort	81
\@gls@setupsort@standard: new	10	added short and long keys	77
\@gls@setupsort@use: new ...	11	replaced \@ifundefined with \ifcsundef	78
\@gls@xdy@locationlist: new	44	\gls@doclearpage: replaced \@ifundefined with \ifcsundef	39
\@glslink: replaced \@ifundefined with \ifcsundef	117	\gls@wrglossary: modified to take into account savewrites	172
\@glsnextpages: new	195	\glsadd: added \@gls@saveentrycounter	154
\@makeglossary: Added check for savewrites	163	\GlsAddXdyCounters: new	41
\@print@glossary: replaced \@ifundefined with \ifcsundef	182	\glsentrycounterlabel: new	197
\@printglossary: added \currentglossary	181	\glsentryitem: new	198
added \@glsnextpages	181	\Glsentrylong: new	152
make toctitle default to title ..	181	\glsentrylong: new	151
\@set@glo@numformat: added 4th argument	109	\Glsentrylongpl: new	152
\@xdy@attributelist: new	40	\glsentrylongpl: new	152
General: added prefix to hyperlink	206	\Glsentryshort: new	151
etoolbox now loaded	4	\glsentryshort: new	151
replaced \@ifundefined with \ifcsundef ...	29, 32, 103, 193	\Glsentryshortpl: new	151
\acrfootnote: new	229	\glsentryshortpl: new	151
\ACRfull: added starred version	210	\glsgetgrouptitle: re- placed \@ifundefined with \ifcsundef	202
\Acrfull: added starred version	210		

<code>\glsglossarymark:</code>	replaced	<code>entrycounterwithin:new</code> 9
<code>\@ifundefined</code>	with	<code>\oldacronym:replaced\@ifundefined</code>	
<code>\ifcsundef</code> 37	<code>with\ifcsundef</code> 208
<code>\glshyperlink:</code>	changed default from <code>\glsentryname</code> to	<code>compatible-2.07: compatible-</code>	
<code>\glsentrytext</code> 154	<code>2.07 option added</code> 26
<code>\glshypernumber:</code>	replaced	<code>long:new</code> 63
<code>\@ifundefined</code>	with	<code>longplural:new</code> 63
<code>\ifcsundef</code> 205	<code>nonumberlist: now boolean</code>	... 62
<code>\glsnumberformat:</code>	replaced	<code>sort:new</code> 10
<code>\@ifundefined</code>	with	<code>counter:replaced\@ifundefined</code>	
<code>\ifcsundef</code> 36	<code>with\ifcsundef</code> 61
<code>\glsrefentry:new</code> 197	<code>\printglossary:</code>	replaced
<code>\glsresetsubentrycounter:</code>		<code>\@ifundefined</code>	with
<code>new</code> 196	<code>\ifcsundef</code> 179
<code>\glsseeitem:</code>	hyperlink uses	<code>\SetDescriptionFootnoteAcronymDisplayStyle:</code>	
<code>\glsseeitemformat</code>	instead	<code>expanded options link op-</code>	
<code>of\glsentryname</code> 178	<code>tions</code> 229
<code>\glsseeitemformat:new</code> 178	<code>\setentrycounter:</code>	added op-
<code>\gls sortnumberfmt:new</code> 11	<code>tional argument</code> 203
<code>\glsstepentry:new</code> 197	<code>\showacronymlists:new</code> 249
<code>\glsstepsubentry:new</code> 197	<code>\showglocounter:new</code> 246
<code>\gls subentrycounterlabel:</code>		<code>\showglodesc:new</code> 247
<code>new</code> 198	<code>\showglodescplural:new</code>	... 248
<code>\gls subentryitem:new</code> 198	<code>\showglofirst:new</code> 246
<code>theglossary:replaced\@ifundefined</code>		<code>\showglofirstpl:new</code> 246
<code>with\ifcsundef</code> 198	<code>\showgloflag:new</code> 249
<code>short:new</code> 63	<code>\showgloindex:new</code> 249
<code>shortplural:new</code> 63	<code>\showglolevel:new</code> 245
<code>\ifglossaryexists:</code>	re-	<code>\showglolevel:new</code> 245
<code>placed \@ifundefined with</code>		<code>\showgloparent:new</code> 247
<code>\ifcsundef</code> 50	<code>\showgloparent:new</code> 245
<code>\ifgl sentryexists:</code>	re-	<code>\showgloplural:new</code> 245
<code>placed \@ifundefined with</code>		<code>\showglosort:new</code> 248
<code>\ifcsundef</code> 51	<code>\showglossaries:new</code> 249
<code>\istfile: deprecated</code> 170	<code>\showglossarycounter:new</code>	. 250
<code>glossaryentry:new</code> 196	<code>\showglossaryentries:new</code>	. 250
<code>glossarysubentry:new</code> 196	<code>\showglossaryin:new</code> 250
<code>\newglossaryentry:</code>	replaced	<code>\showglossaryout:new</code> 250
<code>\DeclareRobustCommand</code>		<code>\showglossarytitle:new</code>	... 250
<code>with\newrobustcmd</code> 66	<code>\showglosymbol:new</code> 248
<code>\newglossarystyle:</code>	re-	<code>\showglosymbolplural:new</code>	. 248
<code>placed \@ifundefined with</code>		<code>\showglo text:new</code> 245
<code>\ifcsundef</code> 204	<code>\showglo type:new</code> 246
<code>\ns@newglossary:</code>	added	<code>\showglouser i:new</code> 246
<code>\@gls@defsortcount</code> 57	<code>\showglouser ii:new</code> 247
<code>replaced \@ifundefined with</code>		<code>\showglouser iii:new</code> 247
<code>\ifcsundef</code> 57	<code>\showglouser iv:new</code> 247
<code>entrycounter:new</code> 9	<code>\showglouser v:new</code> 247
		<code>subentrycounter:new</code> 9

\writeist: added xindy-only macro definitions to glossary open tag	159	\Glspl: made robust	122
modified to support new for- mat	157	\glspl: made robust	121
3.01		\GLSplural: made robust	127
\@glswritefiles: added check for empty glossaries	171	\GLSsymbol: made robust	131
General: made robust	120	\Glsymbol: made robust	131
\ACRfull: made robust	210	\glssymbol: made robust	131
\Acrfull: made robust	210	\GLSsymbolplural: made robust	132
\acrfull: made robust	209	\Glsymbolplural: made robust	132
\acrfullformat: removed \acronymfont as it should al- ready be set in the second ar- gument.	210	\glssymbolplural: made robust	132
\ACRfullpl: made robust	212	\Glstext: made robust	125
\Acrfullpl: made robust	211	\glstext: made robust	124
\acrfullpl: made robust	211	\GLSuseri: made robust	133
\ACRlong: made robust	143	\Glsuseri: made robust	133
\Acrlong: made robust	142	\glsuseri: made robust	132
\acrlong: made robust	142	\GLSuserii: made robust	134
\ACRlongpl: made robust	145	\Glsuserii: made robust	134
\Acrlongpl: made robust	144	\glsuserii: made robust	133
\acrlongpl: made robust	143	\GLSuseriii: made robust	135
\ACRshort: made robust	139	\Glsuseriii: made robust	135
\Acrshort: made robust	138	\glsuseriii: made robust	134
\acrshort: made robust	138	\GLSuseriv: made robust	136
\ACRshortpl: made robust	141	\Glsuseriv: made robust	135
\Acrshortpl: made robust	140	\glsuseriv: made robust	135
\acrshortpl: made robust	140	\GLSuserv: made robust	137
\Gls: made robust	119	\Glsuserv: made robust	136
\glsadd: made robust	154	\glsuserv: made robust	136
\glsaddall: made robust	155	\GLSuservi: made robust	138
\GLSdesc: made robust	129	\Glsuservi: made robust	137
\Glsdesc: made robust	129	\glsuservi: made robust	137
\glsdesc: made robust	129	3.02	
\GLSdescplural: made robust .	130	\@do@wrglossary: changed \@glslocref to \@theglscopycounter	176
\Glsdescplural: made robust .	130	\@do@wrglossary: changed \@do@wr@glossary to test for indexonlyfirst option; put old \@do@wr@glossary code into \@do@wrglossary	173
\glsdescplural: made robust .	130	\@gls@missingnumberlist: new	64
\glsfirst: made robust	125	\@glswritefiles: added check for existence of token in case \makeglossaries has been omitted	171
\GLSfirstplural: made robust	128	\@printglossary: add a way to fetch current entry label ...	181
\Glsfirstplural: made robust	127		
\glsfirstplural: made robust	127		
\glslink: made robust	105		
\GLSname: made robust	129		
\Glsname: made robust	128		
\glsname: made robust	128		
\GLSpl: made robust	122		

savenumberlist: new	7	\glspostinline: replaced “.”	
ucmark: new	9	with \glspostdescription	261
\gls@defglossaryentry: added		altlongragged4col: added	
numberlist element	80	check for glsnogroupskip ..	275
\gls@save@numberlist: new ..	179	altsuperragged4col: added	
\gls@wrglossary: added check		check for glsnogroupskip ..	292
for glossary file defined	172	alttree: added check for	
\glsdisplaynumberlist: new	152	glsnogroupskip	300
\glstentrycounter: set default		index: added check for	
value	108	glsnogroupskip	295
\Glsentryfull: fixed bug (re-		nogroupskip: new	9
placed \glstentryshortpl		long: added check for	
with \glstentryshort)	152	glsnogroupskip	266
\glstentryfullpl: fixed bug (re-		long3col: added check for	
placed \glstentryshort with		glsnogroupskip	268
\glstentryshortpl)	152	long4col: added check for	
\glstentrynumberlist: new ..	152	glsnogroupskip	269
\glsmoveentry: new	82	longragged: added check for	
\glsnumlistlastsep: new ...	153	glsnogroupskip	272
\glsnumlistsep: new	153	longragged3col: added check	
\glsresetsubentrycounter:		for glsnogroupskip	274
new	197	nopostdot: new	9
\ifglshaschildren: new	52	tree: added check for	
\ifglshasparent: new	53	glsnogroupskip	296
\makeglossaries: added list		treenoname: added check for	
parser	166	glsnogroupskip	297
indexonlyfirst: new	23	super: added check for	
\renewglossarystyle: new ..	204	glsnogroupskip	282
\showglossaryentries: fixed		super3col: added check for	
misspelt command	250	glsnogroupskip	283
\SmallNewAcronymDef: fixed		super4col: added check for	
broken short and long plural	239	glsnogroupskip	285
3.03		superragged: added check for	
\@gls@sanitizesort: new	18	glsnogroupskip	289
\@gls@setupsort@standard:		superragged3col: added check	
used \@gls@sanitizesort ..	10	for glsnogroupskip	290
\@printglossary: allow title to		3.04	
override default toctitle	180	\@do@wrglossary: changed	
General: allow title to set toctitle	192	\theglsentrycounter back	
\glsinlinedescformat: new ..	261	to \@glslocref	176
\glsinlineemptydescformat:		\@do@wrglossary: modified to	
new	262	compensate for possible incor-	
\glsinlinenameformat: new ..	261	rect page number	174
\glsinlinepostchild: new ..	261	\@gls@escbsdq: unsani-	
\glsinlinesubdescformat:		tize \gls@numberpage,	
new	262	\gls@alphpage, \gls@Alphpage	
\glsinlinesubnameformat:		and \gls@romanpage	110
new	262	\@print@glossary: Moved aux	
		write to end of document to	

prevent unwanted whatsit occurring here.	182		
General: Added check for doc package	4		
added datatool-base as a required package	4		
added local key	104		
\gls@Alphpage: new	173		
\gls@alphpage: new	173		
\gls@disablepagerefexpansion: new	173		
\gls@numberpage: new	173		
\gls@protected@pagefmts: new	173		
\gls@romanpage: new	173		
\glsdefmain: added check for doc package	12		
\glsorg@endtheglossary: new .	5		
\glsorg@theglossary: new	5		
altlist: replaced \newline with paragraph break	264		
\PrintChanges: new	5		
3.05			
\@do@wrglossary: add Roman case. Fixed bugs in the else statements	174		
\@gls@link: added check for “no-hypertypes”	107		
\@gls@nohyperlist: new	16		
mcolalttree: replaced ‘2’ with \glsmcols	280		
mcolindex: replaced ‘2’ with \glsmcols	277		
mcoltree: replaced ‘2’ with \glsmcols	278		
mcoltreename: replaced ‘2’ with \glsmcols	279		
\gls@protected@pagefmts: added Roman to list	173		
\gls@Romanpage: new	173		
\GlsDeclareNoHyperList: new	16		
\glsgetgrouplabel: fixed bug (typo in \equal)	203		
\nopostdesc: made robust	33		
nohypertypes: new	16		
3.06			
\@xdy@main@language: Changed back to using \language name	25		
\findrootlanguage: Obsoleted	47		
		3.07	
		\@gls@link: fixed bug that failed to find entry in list	107
		\glossarypreamble: modified to work with \setglossarypreamble	36
		\gls@docclearpage: added check for openright	39
		\glspostdescription: Added spacefactor code	9
		\GlsSetXdyCodePage: Added check for fontspec	48
		\SetDescriptionAcronymDisplayStyle: now using \glsdoparenifnotempty	233
		\setglossarypreamble: new ..	36
		3.08a	
		\@glo@storeentry: no longer need to check for special characters in any of the fields other than sort	83
		updated for \glossentry	83
		\@glossaryentryfield: switched to \glossentry	82
		\@glossarysubentryfield: switched to \subglossentry	82
		General: added nogroupskip key to \printglossary	193
		removed definition of \@glossaryentryfield ..	341
		removed definition of \@glossarysubentryfield	341
		\compatibleglossentry: new	199
		\compatiblesubglossentry: new	200
		\glossaryentryfield: deprecated	201
		\Glossentrydesc: new	200
		\glossentrydesc: new	199
		\Glossentryname: new	199
		\glossentryname: new	199
		\Glossentrysymbol: new	200
		\glossentrysymbol: new	200
		\gls@assign@desc@field: new	17
		\gls@assign@descplural@field: new	17
		\gls@assign@field: new	66
		\gls@ifnotmeasuring: new ...	84
		\glsaddallunused: new	155

\glsexpandfields: new	66	3.09a	
\glsnoexpandfields: new	66		\@gls@assign@symbolplural@field:
\glssee: made robust	177		new
\glsseeformat: made robust ..	178		18
\glsseeitem: made robust	178		\@gls@default@value: new ...
\glsseelist: made robust	178		61
\ifglsdescsuppressed: new ..	53		\Glsentrydesc: made robust ..
\ifglshasdesc: new	53		148
\ifglshassymbol: new	53		\Glsentrydescplural: made ro-
list: updated list style to			bust
use \glossentry and			148
\subglossentry	262		\Glsentryfirst: made robust .
listdotted: updated listdotted			149
style to use \glossentry and			\Glsentryfirstplural: made
\subglossentry	265		robust
altlist: updated altlist style			149
to use \glossentry and			\Glsentryfull: made robust ..
\subglossentry	263		152
altlongragged4col: updated			\Glsentryfullpl: made robust
to use \glossentry and			152
\subglossentry	275		\Glsentrylong: made robust ..
alttree: updated to use			152
\glossentry and \subglossentry			\Glsentrylongpl: made robust
.....	299		152
index: added paragraph break at			\Glsentryname: made robust ..
end of environment	294		146
updated to use \glossentry			\Glsentryplural: made robust
and \subglossentry	294		148
inline: updated inline style			\Glsentryshort: made robust .
to use \glossentry and			151
\subglossentry	260		\Glsentryshortpl: made robust
long: updated to use \glossentry		
and \subglossentry	266		151
longragged: updated to			\Glsentrysymbol: made robust
use \glossentry and			149
\subglossentry	272		\Glsentrysymbolplural: made
longragged3col: updated			robust
to use \glossentry and			149
\subglossentry	273		\Glsentrytext: made robust ..
tree: updated to use \glossentry			148
and \subglossentry	295		\Glsentryuseri: made robust .
\setglossarystyle: new	203		150
\setglossentrycompatibility:			\Glsentryuserii: made robust
new	200		150
superragged: updated to			\Glsentryuseriii: made robust
use \glossentry and		
\subglossentry	288		150
			\Glsentryuseriv: made robust
			151
			\Glsentryuserv: made robust .
			151
			\Glsentryuservi: made robust
			151
			\glstextup: new
			209
			\ifglshassymbol: changed test
			to check for \@gls@default@symbol
		
			53
		3.10a	
			\@gls@keymap: new
			68
			\@gls@provide@newglossary:
			new
			55
			\@gls@writedef: new
			67
			\@gls@defaultplural: Obsolete .
			64
			\@gls@nodesc: new
			64
			\@print@glossary: Added
			providecommand code to aux
			file
			182, 183
			\gls@assign@type@field: new
			17
			\gls@defglossaryentry:
			Changed to using \@gls@default@value
		
			77
			new
			76

<code>\glswritedefhook:new</code>	75	removed <code>\makefirstuc</code> (now dealt with in <code>\glentryfmt</code>)	120
<code>\makeglossaries:</code> Added providecommand code to aux file	165	<code>\@Glspl@:</code> add <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glscustomtext</code> and <code>\glsinsert</code>	122
<code>\new@glossaryentry:new</code>	67	change to using <code>\glentryfmt</code> style commands	122
<code>\ns@newglossary:</code> added <code>\@gls@provide@newglossary</code>	57	removed <code>\makefirstuc</code> (now dealt with in <code>\glentryfmt</code>)	122
3.11a		<code>\@acrlong:</code> added <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glsinsert</code> and <code>\glscustomtext</code>	340
<code>\@ACRlong:</code> added <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glsinsert</code> and <code>\glscustomtext</code>	341	<code>\@acrshort:</code> added <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glsinsert</code> and <code>\glscustomtext</code>	339
<code>\@ACRshort:</code> added <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glsinsert</code> and <code>\glscustomtext</code>	339	<code>\@gls@:</code> add <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glscustomtext</code> and <code>\glsinsert</code>	118
<code>\@Acrlong:</code> added <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glsinsert</code> and <code>\glscustomtext</code>	340	change to using <code>\glentryfmt</code> style commands	119
<code>\@Acrshort:</code> added <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glsinsert</code> and <code>\glscustomtext</code>	339	<code>\@gls@noexpand@fields:</code> Fixed bug expand replaced with noexpand	65
<code>\@GLS@:</code> add <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glscustomtext</code> and <code>\glsinsert</code>	120	<code>\@glsdisp:</code> add <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glscustomtext</code> and <code>\glsinsert</code>	124
change to using <code>\glentryfmt</code> style commands	120	change to using <code>\glentryfmt</code> style commands	124
removed <code>\MakeUppercase</code> (now moved to <code>\glentryfmt</code>)	120	<code>\@Glspl@:</code> add <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glscustomtext</code> and <code>\glsinsert</code>	121
<code>\@GLSpl:</code> add <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glscustomtext</code> and <code>\glsinsert</code>	123	change to using <code>\glentryfmt</code> style commands	121
change to using <code>\glentryfmt</code> style commands	123	General: added <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glsinsert</code> and <code>\glscustomtext</code>	138–145
removed <code>\MakeUppercase</code> as now dealt with in <code>\glentryfmt</code>	123	changed to just use <code>\Glsentrydescplural</code>	130
<code>\@Gls@:</code> add <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glscustomtext</code> and <code>\glsinsert</code>	119	changed to just use <code>\glentrydescplural</code>	130, 131
change to using <code>\glentryfmt</code> style commands	119	changed to just use <code>\Glsentrydesc</code>	129

changed to just use <code>\glentrydesc</code>	changed to just use <code>\glentryuservi</code>
..... 129, 130 137, 138
changed to just use <code>\Glsentryfirstplural</code>	changed to just use <code>\Glsentryuserv</code>
..... 128 137
changed to just use <code>\glentryfirstplural</code>	changed to just use <code>\glentryuserv</code>
..... 127, 128 136, 137
changed to just use <code>\Glsentryfirst</code>	Now requires textcase 4
..... 126	acronymlists: replaced
changed to just use <code>\glentryfirst</code>	<code>\@addtoacronymlists</code> with
..... 125, 126	<code>\DeclareAcronymList</code> 15
changed to just use <code>\Glsentryname</code>	<code>\defglssdisplay: obsolete</code> ... 102
..... 128	<code>\defglssdisplayfirst: obso-</code>
changed to just use <code>\glentryname</code>	leted 103
..... 128, 129	<code>\defglssentryfmt: new</code> 56
changed to just use <code>\Glsentryplural</code>	<code>\forglssentries: replaced \ifx</code>
..... 127	with <code>\ifdefempty</code> 50
changed to just use <code>\glssentryplural</code>	<code>\glss@assign@desc: new</code> 75
..... 126, 127	<code>\glss@defglossaryentry: Fixed</code>
changed to just use <code>\Glsentrysymbolplural</code>	default counter if none sup-
..... 132	plied 80
changed to just use <code>\glssentrysymbolplural</code>	<code>\glss@doentryfmt: new</code> 56
..... 132	<code>\glssdisplay: obsolete</code> 102
changed to just use <code>\Glsentrysymbol</code>	<code>\glssdisplayfirst: obsolete</code> . 102
..... 131	<code>\glssgenentryfmt: new</code> 97
changed to just use <code>\glssentrysymbol</code>	<code>\glssgetgrouptitle: Added</code>
..... 131	check in case non-Latin alpha-
Changed to just use	bet in use 202
<code>\Glsentrytext</code> 125	<code>\glssglossarymark: replaced</code>
changed to just use <code>\glssentrytext</code>	<code>\MakeUppercase</code> with
..... 125	<code>\mfirstucMakeUppercase</code> . 37
changed to just use <code>\Glsentryuseriii</code>	<code>\glssnavigation: switched to us-</code>
..... 135	ing <code>\@glss@getgrouptitle</code> 259
changed to just use <code>\glssentryuseriii</code>	<code>\ifglsshasdesc: replaced</code>
..... 134, 135	<code>\ifdefempty</code> with <code>\ifcsempy</code>
changed to just use <code>\Glsentryuserii</code> 53
..... 134	<code>\ifglsshaslong: new</code> 54
changed to just use <code>\glssentryuserii</code>	<code>\ifglsshasshort: new</code> 54
..... 134	<code>\ifglsshasymbol: replaced</code>
changed to just use <code>\Glsentryuseriv</code>	<code>\ifdefempty</code> with <code>\ifcsempy</code>
..... 136 53
changed to just use <code>\glssentryuseriv</code>	<code>\ifglssused: replaced \ifthenelse</code>
..... 135, 136	with <code>\ifbool</code> 51
changed to just use <code>\Glsentryuseri</code>	<code>\longnewglossaryentry: new</code> . 76
..... 133	<code>\ns@newglossary: replaced</code>
changed to just use <code>\glssentryuseri</code>	<code>\glssdisplay</code> and <code>\glssdisplayfirst</code>
..... 133	with <code>\glssentryfmt</code> 57
changed to just use <code>\Glsentryuservi</code>	compatible-3.07: cnew 26
..... 137	<code>\SetCustomDisplayStyle: up-</code>
	dated to use <code>\defglssentryfmt</code>

..... 243	\glossarystyle: fixed bug
\SetDefaultAcronymDisplayStyle:	caused by using \ifdef in-
changed to use \defglentryfmt	stead of \ifcsdef 204
..... 227	\gls@assign@desc@field:
\SetDescriptionAcronymDisplayStyle:	changed to use \glsetnoexpandfield
updated to use \defglentryfmt 17
..... 233	\gls@assign@descplural@field:
\SetDescriptionDUAAcronymDisplayStyle:	changed to use \glsetnoexpandfield
updated to use \defglentryfmt 17
..... 231	\gls@assign@name@field:
\SetDescriptionFootnoteAcronymDisplayStyle:	changed to use \glsetnoexpandfield
updated to use \defglentryfmt 18
..... 229	\gls@assign@type@field:
\SetDUADisplayStyle: updated	changed to use \glsetexpandfield
to use \defglentryfmt .. 241 17
\SetFootnoteAcronymDisplayStyle:	\gls@checkseeallowed: new .. 62
updated to use \defglentryfmt	\glsaddallunused: set default to
..... 235	\@glo@types 155
\SetSmallAcronymDisplayStyle:	\Glsentryfull: changed to use
updated to use \defglentryfmt	\acrfullformat 152
..... 238	\glentryfull: changed to use
\setupglossaries: new 28	\acrfullformat 152
\showglolong: new 248	\Glsentryfullpl: changed to
\showgloshort: new 248	use \acrfullformat 152
numbers: new 27	\glentryfullpl: changed to
symbols: new 27	use \acrfullformat 152
3.12a	\glsglossarymark: renamed
\gls@defglossaryentry: added	\glossarymark to \glsglossarymark
\glslabel 76	to avoid conflict with memoir 37
\glsaddkey: new 70	\glsprestandardssort: new ... 10
3.13a	\glsetexpandfield: new 17
\@gls@assign@symbol@field:	\glsetnoexpandfield: new .. 17
changed to use \glsetnoexpandfield	altsuper4colheader: switched
..... 18	to \tabularnewline 286
\@gls@assign@symbolplural@field:	altsuper4colheaderborder:
changed to use \glsetnoexpandfield	switched to \tabularnewline
..... 18 287
\@gls@link: removed \relax . 108	long: switched to \tabularnewline
\@gls@notranslatorhook: new 21 266
\@gls@setupsort@standard:	long3col: switched to \tabularnewline
moved \@gls@santizesort 267
to \glsprestandardssort .. 10	long3colheader: switched to
ucmark: added check for memoir . 9	\tabularnewline 268
see: added \gls@checkseeallowed	long3colheaderborder: switched
..... 62	to \tabularnewline 268
\glossarysection: changed	long4col: switched to \tabularnewline
\glossarymark to \glsglossarymark 269
..... 37	long4colheader: switched to
	\tabularnewline 269

longheader: switched to \tabularnewline 267	document class 7
longheaderborder: switched to \tabularnewline 267	\CustomAcronymFields: in- serted missing comma 243
\SetFootnoteAcronymDisplayStyle: fixed missing argument bug 236	4.02
super: switched to \tabularnewline 281	\@acrfull: now using \acrfullfmt 209
super3col: switched to \tabularnewline 283	\@gls@indexdef: new 28
super3colheader: switched to \tabularnewline 284	\@gls@numbersdef: new 27
super4col: switched to \tabularnewline 285	\@gls@symbolsdef: new 27
super4colheader: switched to \tabularnewline 285	General: Removed \acronymfont 142–145
super4colheaderborder: switched to \tabularnewline 286	\ACRfullfmt: new 211
superheader: switched to \tabularnewline 282	\Acrfullfmt: new 210
superheaderborder: switched to \tabularnewline 282	\acrfullfmt: new 210
3.14a	\ACRfullplfmt: new 212
\@glswritefiles: renamed \glswritefiles to \@glswritefiles and used “savewrites” option to set \glswritefiles 171	\Acrfullplfmt: new 212
General: new 251	\acrfullplfmt: new 211
acronyms: new 14	\acronymentry: new 214
\gls@defglossaryentry: added check for existence of default glossary 77	sanitize: fixed bug that caused an error here 21
set the default for firstplural to be the value of plural 80	sc-short-long: new 218
xindygloss: new 25	sc-short-long-desc: new ... 220
\longprovideglossaryentry: new 76	\Genacrfullformat: new 101
compatible-2.07: added check for 2.07 before setting 3.07 compatibility 27	\genacrfullformat: new 101
notranslate: new 22	\GenericAcronymFields: new 214
\provideglossaryentry: new . 67	\Genplacrfullformat: new .. 102
4.0	\genplacrfullformat: new .. 102
\gls@defglossaryentry: added check for first key 80	\Glsentryfull: bug fix: added missing \acronymfont 152
4.01	\glentryfull: bug fix: added missing \acronymfont 152
General: fixed non-value options so that they can be passed to	\Glsentryfullpl: bug fix: added missing \acronymfont 152
	\glentryfullpl: bug fix: added missing \acronymfont 152
	\glsgenacfmt: new 99
	\GlsUseAcrEntryDisplayStyle: new 215
	\GlsUseAcrStyleDefs: new .. 215
	short-long: new 217
	short-long-desc: new 220
	xindynoglsnumbers: new 26
	sm-short-long: new 218
	sm-short-long-desc: new ... 220
	\makeglossaries: made pream- ble only 166
	index: new 27
	\newacronymstyle: new 215
	long-sc-short: new 218

long-sc-short-desc: new ...	219	\@Gls@: removed \glslabel (defined in \@gls@link)	119
long-short: new	215	\@Gls@entry@field: new	146
long-short-desc: new	218	\@Glspl@: removed \glslabel (defined in \@gls@link) ...	122
long-sm-short: new	218	\@acrlong: removed \glslabel (defined in \@gls@link) ...	340
long-sm-short-desc: new ...	219	\@acrshort: removed \glslabel (defined in \@gls@link) ...	339
long-sp-short-desc: new ...	219	\@gls@: removed \glslabel (defined in \@gls@link)	118
footnote: new	223	\@gls@access@display: new .	327
footnote-desc: new	225	\@gls@entry@field: new	146
footnote-sc: new	224	\@gls@fetchfield: new	69
footnote-sc-desc: new	225	\@gls@field@link: new	124
footnote-sm: new	224	\@gls@link: added \glsdetoklabel	107
footnote-sm-desc: new	225	\@gls@link@opts and \@gls@link@label to \@gls@link	107
\setacronymstyle: new	214	\@gls@writedef: added \glsdetoklabel	67
\SetDescriptionAcronymDisplayStyle: Moved check for empty custom text to prevent unwanted parenthetical material	233	\@glsdisp: removed \glslabel (defined in \@gls@link) ...	124
\SetDescriptionFootnoteAcronymDisplayStyle: Moved check for empty custom text to prevent unwanted parenthetical material	229	\@glspl@: removed \glslabel (defined in \@gls@link) ...	121
\SetFootnoteAcronymDisplayStyle: Moved check for empty custom text to prevent unwanted parenthetical material	235	\@printglossary: added \glsdetoklabel	181
\SetGenericNewAcronym: new	213	General: changed default to \@empty instead of \relax ..	26
\SetSmallAcronymDisplayStyle: Moved check for empty custom text to prevent unwanted parenthetical material	238	removed \glslabel (defined in \@gls@link)	138
dua: new	221	sc-short-long-desc: redefined to use accessibility information	345
dua-desc: new	222	\compatibleglossentry: added \glsdetoklabel	321
numberedsection: added		\compatiblesubglossentry: added \glsdetoklabel ...	322
nameref option	6	\Genacrformat: redefined to use accessibility information	338
4.03		\genacrformat: redefined to use accessibility information	338
\@do@wrglossary: added		\Genplacrformat: redefined to use accessibility information	338
\glsdetoklabel	175		
\@ACRlong: removed \glslabel (defined in \@gls@link) ...	341		
\@ACRshort: removed \glslabel (defined in \@gls@link) ...	339		
\@Acrlong: removed \glslabel (defined in \@gls@link) ...	340		
\@Acrshort: removed \glslabel (defined in \@gls@link) ...	339		
\@GLS@: removed \glslabel (defined in \@gls@link)	120		
\@GLSpl: removed \glslabel (defined in \@gls@link) ...	123		

<code>\genplacrfullformat:</code> redefined to use accessibility information 338	<code>\glstentrytextaccess:</code> switched to using <code>\@gls@entry@field</code> 326
<code>\glossentryname:</code> added	<code>\glsgenacfmt:</code> redefined to use accessibility information ... 336
<code>\glsdetoklabel</code> 199	<code>\glsgenentryfmt:</code> redefined to use accessibility information 333
<code>\gls@defglossaryentry:</code> added	<code>\glshyperlink:</code> added <code>\glsdetoklabel</code> 154
<code>\glsdetoklabel</code> 76	<code>\glslocalreset:</code> added
replaced <code>#1</code> with <code>\@gls@label</code> 78	<code>\glsdetoklabel</code> 85
replaced <code>\ifthenelse</code> with <code>\ifdefequal</code> 78	<code>\glslocalunset:</code> added
<code>\glsadd:</code> added <code>\glsdetoklabel</code> 154	<code>\glsdetoklabel</code> 85
<code>\glsaddkey:</code> switched to using <code>\@gls@field@link</code> 71	<code>\glsmoveentry:</code> added <code>\glsdetoklabel</code> 82
<code>\glsdetoklabel:</code> new 50	replaced <code>\ifthenelse</code> with <code>\ifdefequal</code> 82
<code>\glsdisplaynumberlist:</code> added	<code>\glsrefentry:</code> added <code>\glsdetoklabel</code> 197
<code>\glsdetoklabel</code> 153	<code>\glsreset:</code> added <code>\glsdetoklabel</code> 84
<code>\glsdoifexistsorwarn:</code> new .. 52	<code>\glsseelist:</code> added <code>\expandafter</code> commands 178
<code>\glstentryaccess:</code> switched to using <code>\@gls@entry@field</code> . 325	<code>\glsstepentry:</code> added <code>\glsdetoklabel</code> 197
<code>\glstentrydescaccess:</code> switched to using <code>\@gls@entry@field</code> 326	<code>\glsstepsubentry:</code> added
<code>\glstentrydescpluralaccess:</code> switched to using <code>\@gls@entry@field</code> 326	<code>\glsdetoklabel</code> 197
<code>\glstentryfirstaccess:</code> switched to using <code>\@gls@entry@field</code> 326	<code>\glsunset:</code> added <code>\glsdetoklabel</code> 85
<code>\glstentryfirstplural:</code> added	short-long: commented spurious EOL 217
<code>\glsdetoklabel</code> 149	redefined to use accessibility information 343
<code>\glstentrylongaccess:</code> switched to using <code>\@gls@entry@field</code> 327	short-long-desc: redefined to use accessibility information 345
<code>\glstentrylongpluralaccess:</code> switched to using <code>\@gls@entry@field</code> 327	<code>\ifglsdscsuppressed:</code> added
<code>\glstentrypluralaccess:</code> switched to using <code>\@gls@entry@field</code> 326	<code>\glsdetoklabel</code> 53
<code>\glstentryshortaccess:</code> switched to using <code>\@gls@entry@field</code> 326	fixed typo 53
<code>\glstentryshortpluralaccess:</code> switched to using <code>\@gls@entry@field</code> 327	<code>\ifglstentryexists:</code> added
<code>\glstentrysymbolaccess:</code> switched to using <code>\@gls@entry@field</code> 326	<code>\glsdetoklabel</code> 51
<code>\glstentrysymbolpluralaccess:</code> switched to using <code>\@gls@entry@field</code> 326	<code>\ifglshaschildren:</code> added
	<code>\glsdetoklabel</code> 52
	<code>\ifglshasdesc:</code> added <code>\glsdetoklabel</code> 53
	<code>\ifglshasfield:</code> new 54
	<code>\ifglshaslong:</code> added <code>\glsdetoklabel</code> 54
	<code>\ifglshasparent:</code> added
	<code>\glsdetoklabel</code> 53

<code>\ifglshasshort:</code>	added	<code>\showglofirstpl:</code>	added
<code>\glsdetoklabel</code>	54	<code>\glsdetoklabel</code>	246
<code>\ifglshassymbol:</code>	added	<code>\showglofirstpluralaccess:</code>	
<code>\glsdetoklabel</code>	53	added <code>\glsdetoklabel</code> ...	357
replaced <code>\ifcempty</code> with		<code>\showgloflag:added\glsdetoklabel</code>	
<code>\ifdefempty</code> and replaced		249
<code>\ifx</code> with <code>\ifdefequal</code>	53	<code>\showgloindex:added\glsdetoklabel</code>	
<code>\ifglused:added\glsdetoklabel</code>		249
.....	51	<code>\showglolevel:added\glsdetoklabel</code>	
<code>sm-short-long-desc:</code> redefined		245
to use accessibility informa-		<code>\showglolong:added\glsdetoklabel</code>	
tion	345	248
<code>long-sc-short-desc:</code> redefined		<code>\showglolongaccess:</code> added	
to use accessibility informa-		<code>\glsdetoklabel</code>	358
tion	344	<code>\showglolongpluralaccess:</code>	
<code>long-short:</code> redefined to use ac-		added <code>\glsdetoklabel</code> ...	358
cessibility information	342	<code>\showglongname:added\glsdetoklabel</code>	
<code>long-short-desc:</code> redefined to		247
use accessibility information	344	<code>\showglongnameaccess:</code> added	
<code>long-sm-short-desc:</code> redefined		<code>\glsdetoklabel</code>	356
to use accessibility informa-		<code>\showgloparent:</code> added	
tion	344	<code>\glsdetoklabel</code>	245
<code>footnote:</code> redefined to use acces-		<code>\showgloplural:</code> added	
sibility information	348	<code>\glsdetoklabel</code>	245
<code>footnote-desc:</code> redefined to use		<code>\showglopluralaccess:</code> added	
accessibility information ...	350	<code>\glsdetoklabel</code>	357
<code>footnote-sc:</code> redefined to use ac-		<code>\showgloshort:added\glsdetoklabel</code>	
cessibility information	350	248
<code>footnote-sc-desc:</code> redefined to		<code>\showgloshortaccess:</code> added	
use accessibility information	351	<code>\glsdetoklabel</code>	357
<code>footnote-sm:</code> redefined to use ac-		<code>\showgloshortpluralaccess:</code>	
cessibility information	350	added <code>\glsdetoklabel</code> ...	358
<code>footnote-sm-desc:</code> redefined to		<code>\showglosort:added\glsdetoklabel</code>	
use accessibility information	351	248
<code>\renewacronymstyle:</code> new ...	215	<code>\showglosymbol:</code> added	
<code>\showglocounter:</code> added		<code>\glsdetoklabel</code>	248
<code>\glsdetoklabel</code>	246	<code>\showglosymbolaccess:</code> added	
<code>\showglodesc:added\glsdetoklabel</code>		<code>\glsdetoklabel</code>	357
.....	247	<code>\showglosymbolplural:</code> added	
<code>\showglodescaccess:</code> added		<code>\glsdetoklabel</code>	248
<code>\glsdetoklabel</code>	357	<code>\showglosymbolpluralaccess:</code>	
<code>\showglodescplural:</code> added		added <code>\glsdetoklabel</code> ...	357
<code>\glsdetoklabel</code>	248	<code>\showglotext:added\glsdetoklabel</code>	
<code>\showglodescpluralaccess:</code>		245
added <code>\glsdetoklabel</code> ...	357	<code>\showglotextaccess:</code> added	
<code>\showglofirst:added\glsdetoklabel</code>		<code>\glsdetoklabel</code>	357
.....	246	<code>\showglotype:added\glsdetoklabel</code>	
<code>\showglofirstaccess:</code> added		246
<code>\glsdetoklabel</code>	357		

<code>\showglouserii:added\glsdetoklabel</code> 246	<code>\@gls@getcounterprefix:</code> added warning if no prefix can be formed 176
<code>\showglouserii:</code> added <code>\glsdetoklabel</code> 246	<code>\@gls@getothergrouptitle:</code> new 203
<code>\showglouseriii:</code> added <code>\glsdetoklabel</code> 247	<code>\@gls@noidx@do:new</code> 190
<code>\showglouseriv:</code> added <code>\glsdetoklabel</code> 247	<code>\@gls@noref@warn:new</code> 170
<code>\showglouserv:added\glsdetoklabel</code> 247	<code>\@gls@reference:new</code> 192
<code>\showglouservi:</code> added <code>\glsdetoklabel</code> 247	<code>\@gls@warnonglossdefined:</code> new 16
<code>dua: fixed bug in \acrfullfmt</code> . 222	<code>\@gls@warnontheGLOSSdefined:</code> new 16
fixed bug in <code>\Acrfullplfmt</code> . 222	<code>\@no@makeGLOSSaries:new</code> .. 170
fixed bug in <code>\acrfullplfmt</code> . 222	<code>\@print@GLOSSary:new</code> 182
redefined to use accessibility in- formation 346	<code>\@print@noidx@GLOSSary:new</code> 188
<code>dua-desc: commented spurious</code> EOL 223	<code>\@printGLOSS@setsort:new</code> . 180
redefined to use accessibility in- formation 348	<code>\@printGLOSSary:new</code> 180
4.04	General: added sort key to print- GLOSS group 195
<code>\@@gls@noidx@nosanitizesort:</code> new 18	<code>\compatIBLEGLOSSentry:</code> changed <code>\newcommand</code> to <code>\def</code> as is may or may not be defined 321
<code>\@@gls@noidx@sanitizesort:</code> new 18	<code>\compatIBLESUBGLOSSentry:</code> changed <code>\newcommand</code> to <code>\def</code> as is may or may not be defined 322
<code>\@@gls@nosanitizesort:new</code> . 18	<code>\defGLSdisplayfirst: fixed un-</code> wanted space 103
<code>\@@gls@sanitizesort:new</code> ... 18	<code>\glo@grabfirst:new</code> 189
<code>\@glo@addchildren:new</code> 184	<code>\gls@defGLOSSaryentry:</code> re-
<code>\@glo@do@sortentries:new</code> . 184	placed <code>\ifx</code> with <code>\ifdefvoid</code> 81
<code>\@glo@grabfirst:new</code> 189	<code>\glsnoidxdisplayloc:new</code> .. 192
<code>\@glo@sortedinsert:new</code> ... 185	<code>\glsnoidxdisplaylocLISThandler:</code> new 191
<code>\@glo@sortentries:new</code> 183	<code>\glsnoidxlocLIST:new</code> 191
<code>\@glo@sorthandler@case:new</code> 186	<code>\glsnoidxlocLISThandler:</code> new 191
<code>\@glo@sorthandler@letter:</code> new 185	<code>\glsnoidxstripaccents:new</code> . 19
<code>\@glo@sorthandler@nocase:</code> new 186	<code>alttree: moved hangindent and</code> <code>parindent</code> assignments out-
<code>\@glo@sorthandler@word:new</code> 185	side level test 299
<code>\@glo@sortmacro@case:new</code> . 187	<code>\makeGLOSSaries: Moved def-</code> inition of <code>\glswrite</code> to
<code>\@glo@sortmacro@def:new</code> .. 187	<code>\makeGLOSSaries</code> 165
<code>\@glo@sortmacro@def@do:new</code> 188	<code>\makenoidxGLOSSaries:new</code> . 167
<code>\@glo@sortmacro@letter:new</code> 186	<code>\printGLOSSary: changed to use</code> new <code>\@printGLOSSary</code> ... 179
<code>\@glo@sortmacro@nocase:new</code> 187	<code>\printnoidxGLOSSaries:new</code> 180
<code>\@glo@sortmacro@standard:</code> new 187	
<code>\@glo@sortmacro@use:new</code> .. 188	
<code>\@glo@sortmacro@word:new</code> . 186	

<code>\printnoidxglossary:new</code> .. 180	<code>\@GLSpl:moved\glsifhyper</code> . 123
<code>\showgloclist:new</code> 249	moved check for first use to
<code>\warn@noprintglossary: Activate warning in \makeglossaries</code> 179	<code>\@gls@link</code> 123
<code>\writeist: checked for definition</code> of <code>\glswrite</code> 157, 161	<code>\@Gls@:moved\glsifhyper</code> .. 119
4.06	moved check for first use to
<code>\@GLS@:added\glsifhyper</code> .. 120	<code>\@gls@link</code> 119
<code>\@GLSpl:added\glsifhyper</code> . 123	<code>\@GLspl@:moved\glsifhyper</code> 122
<code>\@Gls@:added\glsifhyper</code> .. 119	moved check for first use to
<code>\@GLspl@:added\glsifhyper</code> 122	<code>\@gls@link</code> 122
<code>\@gls@:added\glsifhyper</code> .. 119	<code>\@acrlong:added\do@gls@link@checkfirsthyper</code> 340
<code>\@gls@numbersdef: added hook</code> to set toc title 27	<code>\@acrshort:added\do@gls@link@checkfirsthyper</code> 338
<code>\@gls@symbolsdef: added hook</code> to set toc title 27	<code>\@closegls:new</code> 163
<code>\@glsdisp:added\glsifhyper</code> 124	<code>\@gls@:moved\glsifhyper</code> .. 119
<code>\@glspl@:added\glsifhyper</code> 121	moved check for first use to
General: added <code>\glsifhyper</code> 138–145	<code>\@gls@link</code> 119
acronym: added hook to set toc ti- tle 13	<code>\@gls@doautomake:new</code> 26
acronyms: added hook to set toc title 14	<code>\@gls@field@link: added as-</code> segment of <code>\do@gls@link@checkfirsthyper</code> 124
<code>\glsdefmain: added hook to set</code> toc title 12	<code>\@gls@forbidtexext:new</code> 56
4.07	<code>\@gls@hyp@opt:new</code> 105
<code>\@glossarysection: added op-</code> tional argument when using unstarred version 38	<code>\@gls@link: removed redun-</code> dancy 107
<code>\@gls@noidx@do:added\global</code> in case it's used in a tabular- like style 190	renamed <code>\gls@type</code> to <code>\glstype</code> 107
<code>\Acrfullplfmt: fixed no case</code> change bug 212	<code>\@glsdisp:moved\glsifhyper</code> 124
<code>\glsletentryfield:new</code> 146	moved check for first use to
4.08	<code>\@gls@link</code> 124
<code>\@ACRlong:added\do@gls@link@checkfirsthyper</code> 340	<code>\@glspl@:moved\glsifhyper</code> 121
<code>\@ACRshort:added\do@gls@link@checkfirsthyper</code> 339	moved check for first use to
<code>\@Acrlong:added\do@gls@link@checkfirsthyper</code> 340	<code>\@gls@link</code> 121
<code>\@Acrshort:added\do@gls@link@checkfirsthyper</code> 339	<code>\@ignored@glossaries:new</code> .. 59
<code>\@GLS@:moved\glsifhyper</code> .. 120	General: added entrycounter op- tion to <code>printgloss</code> family . 193
moved check for first use to	added <code>nopostdot</code> option to
<code>\@gls@link</code> 120	<code>printgloss</code> family 193
	added subentrycounter option to <code>printgloss</code> family 194
	explicitly initialise hyper key . 104
	moved <code>\glsifhyper</code> ... 138–145
	removed <code>\@sACRlongpl</code> 145
	removed <code>\@sAcrlongpl</code> 144
	removed <code>\@sacrlongpl</code> 144
	removed <code>\@sACRlong</code> 143
	removed <code>\@sAcrlong</code> 142
	removed <code>\@sacrlong</code> 142
	removed <code>\@sACRshortpl</code> ... 141

removed \@sAcrshortpl ...	140	removed \@sGLSuseri	133
removed \@sacrshortpl ...	140	removed \@sGlsuseri	133
removed \@sACRshort	139	removed \@sglsuseri	133
removed \@sAcrshort	139	removed \@sGLSuservi	138
removed \@sacrshort	138	removed \@sGlsuservi	137
removed \@sgls@link	105	removed \@sglsuservi	137
removed \@sGLSdescplural	130	removed \@sGLSuserv	137
removed \@sGlsdescplural	130	removed \@sGlsuserv	136
removed \@sglsdescplural	130	removed \@sglsuserv	136
removed \@sGLSdesc	130	removed \@sGLS	120
removed \@sGlsdesc	129	removed \@sGls	119
removed \@sglsdesc	129	removed \@sgls	118
removed \@sglsdisp	123	removed \@thirdofthree (de-	
removed \@sGLSfirstplural	128	fined in kernel)	118
removed \@sGlsfirstplural	127	removed sPGLS	256
removed \@sglsfirstplural	127	removed sPgls	255
removed \@sGLSfirst	126	removed spgls	253
removed \@sGlsfirst	126	removed sPGLSpl	257
removed \@sglsfirst	125	removed sPglspl	255
removed \@sGLSname	129	removed spglspl	254
removed \@sGlsname	128	\ACRfull: removed \s@ACRfull	210
removed \@sglsname	128	switched to using \@gls@hyp@opt	
removed \@sGLSplural	127	210
removed \@sGlsplural	127	\Acrfull: removed \@sAcrfull	210
removed \@sglsplural	126	switched to using \@gls@hyp@opt	
removed \@sGLSpl	123	210
removed \@sGlspl	122	\acrfull: removed \@sacrfull	209
removed \@sglspl	121	switched to using \@gls@hyp@opt	
removed \@sGLSsymbolplural		209
.....	132	\ACRfullpl: removed \s@ACRfullpl	
removed \@sGLssymbolplural		212
.....	132	switched to using \@gls@hyp@opt	
removed \@sGLSsymbol	131	212
removed \@sGLssymbol	131	\Acrfullpl: removed \s@Acrfullpl	
removed \@sglssymbol	131	211
removed \@sGLStext	125	switched to using \@gls@hyp@opt	
removed \@sGlstext	125	211
removed \@sglstext	124	\acrfullpl: removed \s@acrfullpl	
removed \@sGLSuseriii ...	135	211
removed \@sGlsuseriii ...	135	switched to using	
removed \@sglsuseriii ...	134	@gls@hyp@opt	143
removed \@sGLSuserii	134	\Acrlong: switched to using	
removed \@sGlsuserii	134	@gls@hyp@opt	142
removed \@sglsuserii	133	\acrlong: switched to using	
removed \@sGLSuseriv	136	@gls@hyp@opt	142
removed \@sGlsuseriv	136	\ACRlongpl: switched to using	
removed \@sglsuseriv	135	@gls@hyp@opt	145

\Acrlongpl: switched to using \@gls@hyp@opt	144	\glsdohyperlink:new	117
\acrlongpl: switched to using \@gls@hyp@opt	143	\glsdohypertarget:new	117
\ACRshort: switched to using \@gls@hyp@opt	139	\glsenablehyper: added \KV@glslink@hypertrue to definition	118
\Acrshort: switched to using \@gls@hyp@opt	138	\GLSfirst: switched to using \@gls@hyp@opt	126
\acrshort: switched to using \@gls@hyp@opt	138	\Glsfirst: switched to using \@gls@hyp@opt	126
\ACRshorttpl: switched to using \@gls@hyp@opt	141	\glsfirst: switched to using \@gls@hyp@opt	125
\Acrshorttpl: switched to using \@gls@hyp@opt	140	\GLSfirstplural: switched to using \@gls@hyp@opt	128
\acrshorttpl: switched to using \@gls@hyp@opt	140	\Glsfirstplural: switched to using \@gls@hyp@opt	127
\forallacronyms:new	49	\glsfirstplural: switched to using \@gls@hyp@opt	127
\GLS: switched to using \@gls@hyp@opt	120	\glsifhyper: deprecated	104
\Gls: switched to using \@gls@hyp@opt	119	\glslink: switched to using \@gls@hyp@opt	105
\gls: switched to using \@gls@hyp@opt	118	\glslinkcheckfirsthyperhook: new	106
\gls@defglossaryentry: added check for ignored glossary ...	78	\glslinkvar:new	104
\gls@istfilebase:new	34	\GLSname: switched to using \@gls@hyp@opt	129
\glsaddkey: removed \@sGLS@user@<key>	72	\Glsname: switched to using \@gls@hyp@opt	128
removed \@sGls@user@<key> .	72	\glsname: switched to using \@gls@hyp@opt	128
removed \@sgls@user@<key> .	71	\GLSpl: switched to using \@gls@hyp@opt	122
switched to using \@gls@hyp@opt	71, 72	\Glspl: switched to using \@gls@hyp@opt	122
\GLSdesc: switched to using \@gls@hyp@opt	129	\glspl: switched to using \@gls@hyp@opt	121
\Glsdesc: switched to using \@gls@hyp@opt	129	\GLSplural: switched to using \@gls@hyp@opt	127
\glsdesc: switched to using \@gls@hyp@opt	129	\Glsplural: switched to using \@gls@hyp@opt	126
\GLSdescplural: switched to us- ing \@gls@hyp@opt	130	\glsplural: switched to using \@gls@hyp@opt	126
\Glsdescplural: switched to us- ing \@gls@hyp@opt	130	\glsspace:new	210
\glsdescplural: switched to us- ing \@gls@hyp@opt	130	\GLSsymbol: switched to using \@gls@hyp@opt	131
\glsdisablehyper: added \KV@glslink@hyperfalse to definition	117	\Glsymbol: switched to using \@gls@hyp@opt	131
\glsdisp: switched to using \@gls@hyp@opt	123	\glssymbol: switched to using \@gls@hyp@opt	131

\GLSsymbolplural: switched to using \@gls@hyp@opt	132	\ifignoredglossary:new	59
\Glsymbolplural: switched to using \@gls@hyp@opt	132	altlongragged4col: fixed bug that displayed description instead of symbol	275
\glssymbolplural: switched to using \@gls@hyp@opt	132	\newglossary: added starred version	56
\GLStext: switched to using \@gls@hyp@opt	125	\newignoredglossary:new ...	59
\Glstext: switched to using \@gls@hyp@opt	125	\ns@newglossary: added \@glotype@<name>@log ...	57
\glstext: switched to using \@gls@hyp@opt	124	new	57
\glstreenamfmt:new	293	\p@gls@hyp@opt:new	105
\GLSuseri: switched to using \@gls@hyp@opt	133	\PGLS: changed to use \@gls@hyp@opt	256
\Glsuseri: switched to using \@gls@hyp@opt	133	\Pgls: changed to use \@gls@hyp@opt	255
\gluseri: switched to using \@gls@hyp@opt	132	\pgls: changed to use \@gls@hyp@opt	253
\GLSuserii: switched to using \@gls@hyp@opt	134	\PGLSpl: changed to use \@gls@hyp@opt	257
\Glsuserii: switched to using \@gls@hyp@opt	134	\Pglspl: changed to use \@gls@hyp@opt	255
\gluserii: switched to using \@gls@hyp@opt	133	\pglspl: changed to use \@gls@hyp@opt	254
\GLSuseriii: switched to using \@gls@hyp@opt	135	\s@gls@hyp@opt:new	105
\Glsuseriii: switched to using \@gls@hyp@opt	135	\s@newglossary:new	57
\gluseriii: switched to using \@gls@hyp@opt	134	automake:new	26
\GLSuseriv: switched to using \@gls@hyp@opt	136	4.09	
\Glsuseriv: switched to using \@gls@hyp@opt	135	\glsaddkey: fixed bug in user commands	71
\gluseriv: switched to using \@gls@hyp@opt	135	4.10	
\GLSuseriv: switched to using \@gls@hyp@opt	136	\@Gls@acrentryname:new ...	147
\Glsuseriv: switched to using \@gls@hyp@opt	135	\@Gls@entryname:new	147
\gluseriv: switched to using \@gls@hyp@opt	135	\@gls@glossary: Renamed \@glossary to \@gls@glossary	172
\GLSuserv: switched to using \@gls@hyp@opt	137	\glspercentchar:new	156
\Glsuserv: switched to using \@gls@hyp@opt	136	\glstildechar:new	156
\gluserv: switched to using \@gls@hyp@opt	136	alttree: moved space after symbol	299, 300
\GLSuservi: switched to using \@gls@hyp@opt	138	4.11	
\Glsuservi: switched to using \@gls@hyp@opt	137	\@do@wrglossary: added hook	175
\gluservi: switched to using \@gls@hyp@opt	137	sanitize: none option	21
		\gls@wrglossary: renamed from \@wrglossary to \gls@wrglossary	172
		\glsaddprotectedpagefmt:new	174
		\glsbackslash:new	156

4.12			\@newglossaryentry@defcounters:
	\@gls@addpredefinedattributes:	new	82
	Added glsignore attribute ...	43	
	\@gls@adjustmode: new	154	
	\@gls@notranslatorhook: re-		
	moved	21	
	\@gls@toc: added \protect to		
	\numberline	40	
	\@gls@usetranslator: new ...	21	
	\glsacrpluralsuffix: new ...	31	
	\glsadd: added check for vertical		
	mode	154	
	\glsaddallunused: replaced		
	@gobble with glsignore	155	
	\glsifusedtranslatordict:		
	new	22	
	\glsignore: new	155	
	\glsupacrpluralsuffix: new .	31	
	\ProvidesGlossariesLang:		
	new	31	
	\RequireGlossariesLang: new	31	
4.13	\indexspace: new ...	262, 277, 293	
4.14	\@@glslocalreset: new	86	
	\@@glslocalunset: new	85	
	\@@glsreset: new	86	
	\@@glsunset: new	86	
	\@@newglossaryentry@defcounters:		
	new	87	
	\@cGls: new	91	
	\@cGls@: new	91	
	\@cGlspl@: new	92	
	\@cgls: new	90	
	\@cgls@: new	90	
	\@cglspl: new	91, 92	
	\@cglspl@: new	91	
	\@gls@entry@count: new	90	
	\@gls@increment@currcount:		
	new	89	
	\@gls@local@increment@currcount:		
	new	90	
	\@gls@write@entrycounts:		
	new	90	
	\@glslocalreset: new	86	
	\@glslocalunset: new	85	
	\@glsreset: new	86	
	\@glsunset: new	86	
			\@newglossaryentry@defcounters:
			new
			82
			\cGls: new
			91
			\cgls: new
			90
			\cGlsformat: new
			91
			\cglsformat: new
			90
			\cGlspl: new
			92
			\cglspl: new
			91
			\cGlsplformat: new
			92
			\cglsplformat: new
			92
			\gls@defdocnewglossaryentry:
			new
			67
			\glsenableentrycount: new ..
			87
			\glslocalreset: switched to
			\@glslocalreset
			85
			\glslocalunset: switched to
			\@glslocalunset
			85
			\glsreset: switched to
			\@glsreset
			84
			\glsunset: switched to
			\@glsunset
			85
4.15	General: bug fix replaced		
	\@glo@type with \glstype	145	
4.16	\@ACRlong: added \glspostlinkhook		
	341	
	\@ACRshort: added \glspostlinkhook		
	340	
	\@Acrlong: added \glspostlinkhook		
	340	
	\@Acrshort: added \glspostlinkhook		
	339	
	\@GLS@: added \glspostlinkhook		
	121	
	\@GLSpl: added \glspostlinkhook		
	123	
	\@Gls@: added \glspostlinkhook		
	120	
	\@Glspl@: added \glspostlinkhook		
	122	
	\@acrlong: added \glspostlinkhook		
	340	
	\@acrshort: added \glspostlinkhook		
	339	
	\@gls@: added \glspostlinkhook		
	119	
	\@gls@@link: added \glspostlinkhook		
	106	

\@gls@field@link: added	\glspostlinkhook:new	106
\glspostlinkhook	\glswriteentry:new	173
\@gls@link: moved definition	\ifglsfieldcseq:new	75
of \glsifhyperon outside of	\ifglsfielddefeq:new	75
this macro	\ifglsfieldeq:new	74
\@glsdisp:added\glspostlinkhook	long-sp-short:new	216
.....	\showglofield:new	249
\@glspl@:added\glspostlinkhook	4.18	
.....	General: split mfirstuc into sepa-	
General:added\glspostlinkhook	rate bundle	4
.....	4.19	
..... 138–145	\@gls@link@nocheckfirsthyper:	
\glsacspace:new	new	124
\glsadd:changed\@do@wrglossary	\@gls@preglossaryhook:new	180
to\@do@wrglossary	\@printglossary: added	
\glsaddstoragekey:new	\@gls@preglossaryhook .	181
\glsfielddef:new	\do@gl:disablehyperinlist:	
\glsfieldedef:new	new	107
\glsfieldfetch:new	\doifglossarynoexistsordo:	
\glsfieldgdef:new	new	52
\glsfieldxdef:new	\gls@gobbleopt:new	56
\glsifhyperon: moved defini-	\glsdoifexistsordo:new	52
tion of\glsifhyperon	\glstreenamebox:new	298
\glslinkpostsetkeys:new ..		106

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	
\@@do@@wrglossary	175
\@@do@wrglossary	174
\@@glo@assign@sortkey	195
\@@glossarysec	5
\@@glossaryseclabel	6
\@@glossarysecstar	6
\@@gls@default@entryfmt	329
\@@gls@expand@field	65
\@@gls@fixbraces	177
\@@gls@noidx@nosanitizesort .	18
\@@gls@noidx@sanitizesort ...	18
\@@gls@nosanitizesort	18
\@@gls@sanitizesort	18
\@@glslocalreset	86
\@@glslocalunset	85
\@@glsreset	86
\@@glsunset	86
\@@newglossaryentry@defcounters	
.....	87
\@ACRlong	340
\@ACRshort	339
\@Acrlong	340
\@Acrshort	339
\@GLS@	120
\@GLSpl	123
\@Gls@	119
\@Gls@acrentryname	147
\@Gls@entry@field	146
\@Gls@entryname	147
\@Glspl@	122
\@PGLS	256
\@PGLS@	256
\@PGLSpl	257
\@PGLSpl@	257
\@Pgls	255
\@Pgls@	255
\@Pglspl	255
\@Pglspl@	256
\@acrfull	209
\@acrlong	340
\@acrshort	338
\@addtoacronymlists	14
\@cGls	91
\@cGls@	91
\@cGlspl@	92
\@cgls	90
\@cgls@	90
\@cglspl	91, 92
\@cglspl@	91
\@closegls	163
\@delimN	206
\@delimR	206
\@disable@onlypremakeg	30
\@disable@premakecs	30
\@disabled@glsaddxdycounters	41
\@do@seeglossary	177
\@do@wrglossary	173, 303
\@glo@addchildren	184
\@glo@default@sorttype	10
\@glo@do@sortentries	184
\@glo@grabfirst	189
\@glo@no@assign@sortkey	195
\@glo@seeautonumberlist	7
\@glo@sortedinsert	185
\@glo@sortentries	183
\@glo@sorthandler@case	186
\@glo@sorthandler@letter ...	185
\@glo@sorthandler@nocase ...	186
\@glo@sorthandler@word	185
\@glo@sortmacro@case	187
\@glo@sortmacro@def	187
\@glo@sortmacro@def@do	188
\@glo@sortmacro@letter	186
\@glo@sortmacro@nocase	187
\@glo@sortmacro@standard ...	187
\@glo@sortmacro@use	188
\@glo@sortmacro@word	186
\@glo@storeentry	83
\@glo@types	55
\@glossary@default@style	7
\@glossaryentryfield	82
\@glossarysection	38
\@glossarysubentryfield	82
\@gls	118
\@gls@	118

<code>\@gls@link</code>	105	<code>\@gls@local@increment@currcount</code>	
<code>\@gls@access@display</code>	327	90
<code>\@gls@addpredefinedattributes</code>		<code>\@gls@missingnumberlist</code>	64
.....	43	<code>\@gls@noaccess</code>	324
<code>\@gls@adjustmode</code>	154	<code>\@gls@noexpand@field</code>	64
<code>\@gls@assign@symbol@field</code> ...	18	<code>\@gls@noexpand@fields</code>	65
<code>\@gls@assign@symbolplural@field</code>		<code>\@gls@nohyperlist</code>	16
.....	18	<code>\@gls@noidx@do</code>	190
<code>\@gls@checkactual</code>	115	<code>\@gls@noidx@setsanitizesort</code> .	21
<code>\@gls@checkbar</code>	114	<code>\@gls@noref@warn</code>	170
<code>\@gls@checkescactual</code>	112	<code>\@gls@notranslatorhook</code>	21
<code>\@gls@checkescbar</code>	113	<code>\@gls@numbersdef</code>	27
<code>\@gls@checkesclevel</code>	113	<code>\@gls@onlypremakeg</code>	29
<code>\@gls@checkescquote</code>	112	<code>\@gls@preglossaryhook</code>	180
<code>\@gls@checklevel</code>	114	<code>\@gls@provide@newglossary</code> ...	55
<code>\@gls@checkmkidxchars</code>	110	<code>\@gls@reference</code>	192
<code>\@gls@checkquote</code>	111	<code>\@gls@renewglossary</code>	172
<code>\@gls@codepage</code>	48	<code>\@gls@sanitizedesc</code>	17
<code>\@gls@counterwithin</code>	9	<code>\@gls@sanitizename</code>	17
<code>\@gls@declareoption</code>	7	<code>\@gls@sanitizesort</code>	18
<code>\@gls@default@value</code>	61	<code>\@gls@sanitizesymbol</code>	18
<code>\@gls@do@acronymsdef</code>	14	<code>\@gls@saveentrycounter</code>	108
<code>\@gls@doautomake</code>	26	<code>\@gls@setacrstyle</code>	23
<code>\@gls@entry@count</code>	90	<code>\@gls@setcounter</code>	58
<code>\@gls@entry@field</code>	146	<code>\@gls@setupsort@def</code>	11
<code>\@gls@escbsdq</code>	110	<code>\@gls@setupsort@standard</code>	10
<code>\@gls@expand@fields</code>	65	<code>\@gls@setupsort@use</code>	11
<code>\@gls@fetchfield</code>	69	<code>\@gls@startswithexpandonce</code> ..	66
<code>\@gls@field@link</code>	124	<code>\@gls@symbolsdef</code>	27
<code>\@gls@fixbraces</code>	177	<code>\@gls@tmpb</code>	111
<code>\@gls@forbidtexext</code>	56	<code>\@gls@toc</code>	40
<code>\@gls@getcounter</code>	58	<code>\@gls@updatechecked</code>	111
<code>\@gls@getcounterprefix</code>	176	<code>\@gls@usetranslator</code>	21
<code>\@gls@getgrouptitle</code>	202	<code>\@gls@warnonglossdefined</code>	16
<code>\@gls@getothergrouptitle</code> ...	203	<code>\@gls@warnonthe glossdefined</code> .	16
<code>\@gls@glossary</code>	172	<code>\@gls@write@entrycounts</code>	90
<code>\@gls@hyp@opt</code>	105	<code>\@gls@writedef</code>	67
<code>\@gls@hypergroup</code>	258	<code>\@gls@xdy@Lclass@Alpha-page-numbers</code>	
<code>\@gls@ifinlist</code>	41	45
<code>\@gls@increment@currcount</code> ...	89	<code>\@gls@xdy@Lclass@Appendix-page-numbers</code>	
<code>\@gls@indexdef</code>	28	45
<code>\@gls@keymap</code>	68, 251	<code>\@gls@xdy@Lclass@Roman-page-numbers</code>	
<code>\@gls@link</code>	107	44
<code>\@gls@link@nocheckfirsthyper</code>	124	<code>\@gls@xdy@Lclass@alpha-page-numbers</code>	
<code>\@gls@loadlist</code>	8	45
<code>\@gls@loadlong</code>	8	<code>\@gls@xdy@Lclass@arabic-page-numbers</code>	
<code>\@gls@loadsuper</code>	8	45
<code>\@gls@loadtree</code>	8	<code>\@gls@xdy@Lclass@arabic-section-numbers</code>	
		45

<code>\acrlinkfullformat</code>	210	<code>\acs</code>	226
<code>\ACRlong</code>	143	<code>\Acsp</code>	226
<code>\Acrlong</code>	142	<code>\acsp</code>	226
<code>\acrlong</code>	142	<code>\addglossarytocaptions</code>	31
<code>\ACRlongpl</code>	145	<code>\addto</code>	32
<code>\Acrlongpl</code>	144	<code>align (environment)</code>	84, 108
<code>\acrlongpl</code>	143	<code>altlist (style)</code>	263
<code>\acrnameformat</code>	212, 235	<code>altlistgroup (style)</code>	264
<code>\acrnolinkfootnote</code>	229	<code>altlisthypergroup (style)</code>	264
<code>acronym (option)</code>	13	<code>altlong4col (style)</code>	270
acronym styles:		<code>altlong4colborder (style)</code>	271
<code>dua</code>	221, 346	<code>altlong4colheader (style)</code>	270
<code>dua-desc</code>	222, 348	<code>altlong4colheaderborder (style)</code>	271
<code>footnote</code>	223, 348	<code>altlongragged4col (style)</code>	275
<code>footnote-desc</code>	225, 350	<code>altlongragged4colborder (style)</code>	276
<code>footnote-sc</code>	224, 350	<code>altlongragged4colheader (style)</code>	275
<code>footnote-sc-desc</code>	225, 351	<code>altlongragged4colheaderborder</code>	
<code>footnote-sm</code>	224, 350	<code>(style)</code>	276
<code>footnote-sm-desc</code>	225, 351	<code>\altnewglossary</code>	58
<code>long-sc-short</code>	218	<code>altsuper4col (style)</code>	286
<code>long-sc-short-desc</code>	219, 344	<code>altsuper4colborder (style)</code>	287
<code>long-short</code>	215, 342	<code>altsuper4colheader (style)</code>	286
<code>long-short-desc</code>	218, 344	<code>altsuper4colheaderborder</code>	
<code>long-sm-short</code>	218	<code>(style)</code>	287
<code>long-sm-short-desc</code>	219, 344	<code>altsuperragged4col (style)</code>	291
<code>long-sp-short</code>	216	<code>altsuperragged4colborder</code>	
<code>long-sp-short-desc</code>	219	<code>(style)</code>	292
<code>sc-short-long</code>	218	<code>altsuperragged4colheader</code>	
<code>sc-short-long-desc</code>	220, 345	<code>(style)</code>	292
<code>short-long</code>	217, 343	<code>altsuperragged4colheaderborder</code>	
<code>short-long-desc</code>	220, 345	<code>(style)</code>	293
<code>sm-short-long</code>	218	<code>almtree (style)</code>	298
<code>sm-short-long-desc</code>	220, 345	<code>almtreegroup (style)</code>	300
<code>\acronymentry</code>	214	<code>almtreehypergroup (style)</code>	301
<code>\acronymfont</code>		<code>amsgen package</code>	4, 103
.....	99, 212, 231, 235, 237, 240	<code>amsmath package</code>	84
<code>acronymlists (option)</code>	15	<code>\andname</code>	31
<code>\acronymname</code>	30	<code>array package</code>	271, 287
<code>acronyms (option)</code>	14	<code>article class</code>	176
<code>\acronymsort</code>	214	<code>automake (option)</code>	26
<code>\acronymtype</code>	13, 208		
<code>\acrpluralsuffix</code>	209	B	
<code>\ACRshort</code>	139	<code>babel package</code>	21, 30, 32, 48
<code>\Acrshort</code>	138		
<code>\acrshort</code>	138	C	
<code>\ACRshortpl</code>	141	<code>\cGls</code>	91
<code>\Acrshortpl</code>	140	<code>\cgls</code>	90
<code>\acrshortpl</code>	140	<code>\cGlsformat</code>	91
<code>\acrshortpl</code>	140	<code>\cglsformat</code>	90
<code>\Acs</code>	226	<code>\cGlspl</code>	92

`\cglspl` 91
`\cGlsplformat` 92
`\cglsplformat` 92
`\compatglossarystyle` 308
`compatible-2.07 (option)` 26
`compatible-3.07 (option)` 26
`\compatibleglossentry` .. 199, 321
`\compatiblesubglossentry` 200, 322
`counter (key)` 61
`counter (option)` 16
`\CustomAcronymFields` 243
`\CustomNewAcronymDef` 243

D

`datatool package` 185
`\DeclareAcronymList` 14
`\DefaultNewAcronymDef` .. 227, 352
`\defentryfmt` 103
`\defglstdisplay` 102
`\defglstdisplayfirst` 103
`\defglsentry` 57
`\defglsentryfmt` 56, 60, 61, 94
`\DefineAcronymSynonyms` 225
`\delimN` 36, 205
`\delimR` 36, 205
`description (environment)` 262
`description (key)` 60
`description (option)` 24
`descriptionaccess (key)` 323
`\DescriptionDUANewAcronymDef` 231
`\DescriptionFootnoteNewAcronymDef`
..... 230, 353
`\descriptionname` 30
`\DescriptionNewAcronymDef` ..
..... 234, 353
`descriptionplural (key)` 60
`descriptionpluralaccess (key)` 323
`\detokenize` 50
`\do@glstdisablehyperinlist` .. 107
`doc package` 4, 5, 12
`document (environment)` 67, 88
`\doifglossarynoexistsordo` ... 52
`dua (acrstyle)` 221, 346
`dua (option)` 24
`dua-desc (acrstyle)` 222, 348
`\DUANewAcronymDef` 241

E

`entrycounter (option)` 9
`entrycounterwithin (option)` 9

`\entryname` 30
environments:
 `align` 84, 108
 `description` 262
 `document` 67, 88
 `longtable` 8, 244, 265–276
 `multicols` 277
 `supertabular` ... 8, 244, 281–293
 `theglossary` ... 5, 16, 36, 37,
 198, 204, 278, 279, 295, 296, 298
 `theindex` 294
`equation (counter)` 108, 109
`etoolbox package` 4

F

file types
 `.aux` 182
 `.glo` 83
 `.ist` 155, 162
 `.toc` 40
 `.xdy` 34
 `glo` 250
`\findrootlanguage` 47
`first (key)` 61
`firstaccess (key)` 322
`\firstacronymfont` 99, 212
`firstplural (key)` 61
`firstpluralaccess (key)` 323
`footnote (acrstyle)` 223, 348
`footnote (option)` 23
`footnote-desc (acrstyle)` .. 225, 350
`footnote-sc (acrstyle)` 224, 350
`footnote-sc-desc (acrstyle)` 225, 351
`footnote-sm (acrstyle)` 224, 350
`footnote-sm-desc (acrstyle)` 225, 351
`\FootnoteNewAcronymDef` . 236, 354
`\forallacronyms` 49
`\forallglossaries` 49
`\forallglsentries` 50
`\forglsentries` 49

G

`garamondx package` 209
`\Genacrfullformat` 101, 338
`\genacrfullformat` 101, 338
`\GenericAcronymFields` 214
`\Genplacrfullformat` 102, 338
`\genplacrfullformat` 102, 338
`\glo@grabfirst` 189
`\glolinkprefix` 108

glossareentry (counter)	197	glossary package	1, 208
glossaries package	28,	glossary styles:	
48, 156, 244, 251, 262, 301, 321		altlist	263, 264, 309
glossaries-accsupp package ...	82, 321	altlist	263
\GlossariesWarning	16	altlistgroup	264, 309
\GlossariesWarningNoLine	16	altlistgroup	264
\glossary	56, 162, 171, 203	altlisthypergroup ...	264, 310
glossary counters:		altlisthypergroup	264
glossaryentry	196	altlong4col ..	270, 271, 275, 311
glossarysubentry	196	altlong4col	270
glossary keys:		altlong4colborder ...	271, 312
access	322	altlong4colborder	271
counter	61	altlong4colheader ...	270, 312
description	60	altlong4colheader	270
descriptionaccess	323	altlong4colheaderborder .	
descriptionplural	60	271, 312
descriptionpluralaccess .	323	altlong4colheaderborder .	271
first	61	altlongragged4col	275, 276, 313
firstaccess	322	altlongragged4col	275
firstplural	61	altlongragged4colborder .	
firstpluralaccess	323	276, 313
long	63	altlongragged4colborder .	276
longaccess	323	altlongragged4colheader .	
longplural	63	275, 313
longpluralaccess	323	altlongragged4colheader .	275
name	60	altlongragged4colheaderborder	
nonumberlist	62	276, 313
parent	62	altlongragged4colheaderborder	
plural	60	276
pluralaccess	323	altsuper4col .	286, 287, 291, 321
see	62	altsuper4col	286
short	63	altsuper4colborder ..	287, 321
shortaccess	323	altsuper4colborder	287
shortplural	63	altsuper4colheader ..	286, 321
shortpluralaccess	323	altsuper4colheader	286
sort	60	altsuper4colheaderborder	
symbol	61	287, 321
symbolaccess	323	altsuper4colheaderborder	287
symbolplural	61	altsuperragged4col	291–293, 319
symbolpluralaccess	323	altsuperragged4col	291
text	60	altsuperragged4colborder	
textaccess	322	292, 319
type	61	altsuperragged4colborder	292
user1	62	altsuperragged4colheader	
user2	63	292, 319
user3	63	altsuperragged4colheader	292
user4	63	altsuperragged4colheaderborder	
user5	63	293, 319
user6	63		

altsuperragged4colheaderborder	293	longragged	272
alttree	280, 298, 300, 315	longragged3col	273, 274, 312
alttree	298	longragged3col	273
alttreegroup	301, 316	longragged3colborder	274, 313
alttreegroup	300	longragged3colborder	274
alttreehypergroup	301, 317	longragged3colheader	274, 313
alttreehypergroup	301	longragged3colheader	274
index	277, 294, 295, 313	longragged3colheaderborder	274, 313
index	294	longragged3colheaderborder	274
indexgroup	295, 314	longraggedborder	272, 312
indexgroup	295	longraggedborder	272
indexhypergroup	295, 314	longraggedheader	273, 312
indexhypergroup	295	longraggedheader	273
inline	308	longraggedheaderborder	273, 312
inline	260	longraggedheaderborder	273
list	6, 262–264, 309	mcolalttree	280, 317
list	262	mcolalttree	280
listdotted	264, 265, 310	mcolalttreegroup	280, 318
listdotted	264	mcolalttreegroup	280
listgroup	263, 309	mcolalttreehypergroup	280, 318
listgroup	263	mcolalttreehypergroup	280
listhypergroup	263, 309	mcolindex	277, 317
listhypergroup	263	mcolindex	277
long	266, 267, 272, 310, 312	mcolindexgroup	277, 317
long	266	mcolindexgroup	277
long3col	267, 268, 310	mcolindexhypergroup	277, 317
long3col	267	mcolindexhypergroup	277
long3colborder	268, 311	mcoltree	278, 317
long3colborder	268	mcoltree	278
long3colheader	268, 311	mcoltreegroup	317
long3colheader	268	mcoltreegroup	278
long3colheaderborder	268, 311	mcoltreehypergroup	278, 317
long3colheaderborder	268	mcoltreehypergroup	278
long4col	268–270, 311	mcoltreenoname	279, 317
long4col	268	mcoltreenoname	279
long4colborder	269, 311	mcoltreenonamegroup	279, 317
long4colborder	269	mcoltreenonamegroup	279
long4colheader	269, 311	mcoltreenonamehypergroup	279, 317
long4colheader	269	mcoltreenonamehypergroup	279
long4colheaderborder	270, 311	sublistdotted	310
long4colheaderborder	270	sublistdotted	265
longborder	266, 310	super	281–283, 289, 319
longborder	266	super	281
longheader	267, 310	super3col	283, 284, 320
longheader	267	super3col	283
longheaderborder	267, 310	super3colborder	283, 320
longheaderborder	267		
longragged	272, 273		

super3colborder	283	treenonamehypergroup	298
super3colheader	284, 320	glossary-hypernav package	156
super3colheader	284	glossary-list package	6, 8, 262
super3colheaderborder	284, 320	glossary-long package .	8, 265, 275, 281
super3colheaderborder ...	284	glossary-longragged package	271
super4col	284–286, 320	glossary-mcols package	277
super4col	284	glossary-super package	
super4colborder	285, 320	8, 265, 281, 287, 291
super4colborder	285	glossary-superragged package	287
super4colheader	285, 320	glossary-tree package	8, 293
super4colheader	285	glossaryentry (counter) .	9, 197, 198
super4colheaderborder	286, 321	glossaryentry (counter)	196
super4colheaderborder ...	286	\glossaryentryfield .	201, 204, 205
superborder	282, 319	\glossaryentrynumber ...	195, 196
superborder	282	\glossaryentrynumbers	
superheader	282, 319	7, 36, 180, 181
superheader	282	\glossaryheader	198, 204
superheaderborder ...	282, 320	\glossarymark	38
superheaderborder	282	\glossaryname	30, 32
superragged	288–290, 318	\glossarypostamble	37, 204
superragged	288	\glossarypreamble	36, 204
superragged3col ..	290, 291, 318	\glossarysection	6, 37, 56
superragged3col	290	\glossarystyle	204, 245
superragged3colborder	290, 318	glossarysubentry (counter) ...	
superragged3colborder ...	290	9, 196–198
superragged3colheader	291, 319	glossarysubentry (counter) ...	196
superragged3colheader ...	291	\glossarysubentryfield	201
superragged3colheaderborder	291, 319	\glossentry	61, 199
.....	291, 319	\Glossentrydesc	200
superraggedborder ...	289, 318	\glossentrydesc	199, 341
superraggedborder	289	\Glossentryname	199
superraggedheader ...	289, 318	\glossentryname	199, 341
superraggedheader	289	\Glossentrysymbol	200
superraggedheaderborder .		\glossentrysymbol	200, 341
.....	289, 318	\GLS	120
superraggedheaderborder .	289	\Gls	119, 122
superraggedright3colheaderborder	291	\gls	4, 61, 92, 104, 118, 120, 121, 124–137, 197, 253
tree	278, 295–298, 314	\gls@Alphpage	173
tree	295	\gls@alphpage	173
treegroup	278, 296, 315	\gls@assign@desc	75
treegroup	296	\gls@assign@desc@field	17
treehypergroup	296, 315	\gls@assign@descplural@field	17
treehypergroup	296	\gls@assign@field	66
treenoname	279, 297, 315	\gls@assign@name@field	18
treenoname	297	\gls@assign@type@field	17
treenonamegroup	298, 315	\gls@checkisacronymlist	15
treenonamegroup	297	\gls@checkseeallowed	62
treenonamehypergroup	298, 315	\gls@codepage	25

<code>\gls@defdocnewglossaryentry</code>	67	<code>\glsdefaultttype</code>	13
<code>\gls@defglossaryentry</code>	76	<code>\glsdefmain</code>	12
<code>\gls@disablepagerefexpansion</code>	173	<code>\GLSdesc</code>	129
<code>\gls@doclearpage</code>	39	<code>\Glsdesc</code>	129
<code>\gls@doentryfmt</code>	56	<code>\glsdesc</code>	129
<code>\gls@glossary</code>	172	<code>\GLSdescplural</code>	130
<code>\gls@gobbleopt</code>	56	<code>\Glsdescplural</code>	130
<code>\gls@hypergroup prerun</code>	258	<code>\glsdescplural</code>	130
<code>\gls@ifnotmeasuring</code>	84	<code>\glsdescriptionaccessdisplay</code>	328
<code>\gls@istfilebase</code>	34	<code>\glsdescriptionpluralaccessdisplay</code>	328
<code>\gls@level</code>	64	<code>\glsdescwidth</code>	265, 271, 281, 288
<code>\gls@noidxglossary</code>	170	<code>\glsdetoklabel</code>	50
<code>\gls@numberpage</code>	173	<code>\glsdisablehyper</code>	117
<code>\gls@protected@pagefmts</code>	173	<code>\glsdisp</code>	123
<code>\gls@Romanpage</code>	173	<code>\glsdisplay</code>	93, 102, 118
<code>\gls@romanpage</code>	173	<code>\glsdisplayfirst</code>	93, 102, 118
<code>\gls@save@numberlist</code>	179	<code>\glsdisplaynumberlist</code>	152, 168, 191
<code>\gls@suffixF</code>	35	<code>\glsdohyperlink</code>	117
<code>\gls@suffixFF</code>	35	<code>\glsdohypertarget</code>	117
<code>\gls@wrglossary</code>	172	<code>\glsdoifexists</code>	51
<code>\glsaccessdisplay</code>	329	<code>\glsdoifexistsordo</code>	52
<code>\glsaccsupp</code>	327	<code>\glsdoifexistsorwarn</code>	51
<code>\glsacrpluralsuffix</code>	31	<code>\glsdoifnoexists</code>	51
<code>\glsacspace</code>	217	<code>\glsdoparenifnotempty</code>	238
<code>\glsadd</code>	92, 154, 203	<code>\glsenableentrycount</code>	87
<code>\glsadd options</code>		<code>\glsenablehyper</code>	118
<code>counter</code>	154	<code>\glsentryaccess</code>	325
<code>format</code>	154, 205	<code>\glsentrycounter</code>	108
<code>\glsaddall</code>	50, 92, 155	<code>\glsentrycounterlabel</code>	197
<code>\glsaddall options</code>		<code>\glsentrycurrcount</code>	88
<code>types</code>	154, 155	<code>\Glsentrydesc</code>	148
<code>\glsaddallunused</code>	155	<code>\glsentrydesc</code>	148
<code>\glsaddkey</code>	70	<code>\glsentrydescaccess</code>	326
<code>\GlsAddLetterGroup</code>	49	<code>\Glsentrydescplural</code>	148
<code>\glsaddprotectedpagefmt</code>	174	<code>\glsentrydescplural</code>	148
<code>\GlsAddSortRule</code>	47	<code>\glsentrydescpluralaccess</code>	326
<code>\glsaddstoragekey</code>	69	<code>\Glsentryfirst</code>	149
<code>\GlsAddXdyAlphabet</code>	43	<code>\glsentryfirst</code>	149
<code>\GlsAddXdyAttribute</code>	42, 301	<code>\glsentryfirstaccess</code>	326
<code>\GlsAddXdyCounters</code>	41, 302	<code>\Glsentryfirstplural</code>	149
<code>\GlsAddXdyLocation</code>	45, 302	<code>\glsentryfirstplural</code>	149
<code>\GlsAddXdyStyle</code>	47	<code>\glsentryfirstpluralaccess</code>	326
<code>\glsautoprefix</code>	6	<code>\glsentryfmt</code>	60, 61, 93
<code>\glsbackslash</code>	156	<code>\Glsentryfull</code>	152
<code>\glsclearpage</code>	40	<code>\glsentryfull</code>	152, 214, 224, 350
<code>\glsclosebrace</code>	156	<code>\Glsentryfullpl</code>	152
<code>\glscompositor</code>	34, 45	<code>\glsentryfullpl</code>	152
<code>\glscounter</code>	15, 58	<code>\glsentryitem</code>	198
<code>\GlsDeclareNoHyperList</code>	16		

<code>\Glsentrylong</code>	152	<code>\glsentryuservi</code>	151
<code>\glsentrylong</code>	151	<code>\glsexpandfields</code>	66
<code>\glsentrylongaccess</code>	327	<code>\glsfielddef</code>	73
<code>\Glsentrylongpl</code>	152	<code>\glsfieldedef</code>	73
<code>\glsentrylongpl</code>	152	<code>\glsfieldfetch</code>	74
<code>\glsentrylongpluralaccess</code>	327	<code>\glsfieldgdef</code>	73
<code>\Glsentryname</code>	146	<code>\glsfieldxdef</code>	72
<code>\glsentryname</code>	146, 178	<code>\GLSfirst</code>	126
<code>\glsentrynumberlist</code>	152, 168	<code>\Glsfirst</code>	126
<code>\Glsentryplural</code>	148	<code>\glsfirst</code>	125, 126
<code>\glsentryplural</code>	148	<code>\glsfirstaccessdisplay</code>	328
<code>\glsentrypluralaccess</code>	326	<code>\GLSfirstplural</code>	128
<code>\Glsentryprefix</code>	253	<code>\Glsfirstplural</code>	127
<code>\glsentryprefix</code>	252	<code>\glsfirstplural</code>	127, 128
<code>\Glsentryprefixfirst</code>	252	<code>\glsfirstpluralaccessdisplay</code>	328
<code>\glsentryprefixfirst</code>	252	<code>\glsгенacfmt</code>	99, 336
<code>\Glsentryprefixfirstplural</code>	252	<code>\glsгенentryfmt</code>	97, 333
<code>\glsentryprefixfirstplural</code>	252	<code>\glsgetgrouplabel</code>	203
<code>\Glsentryprefixplural</code>	253	<code>\glsgetgrouptitle</code>	156, 202
<code>\glsentryprefixplural</code>	252	<code>\gls glossarymark</code>	9, 37
<code>\glsentryprevcount</code>	88	<code>\glsgroupheading</code>	202, 204
<code>\Glsentryshort</code>	151	<code>\glsgroupskip</code>	201, 204, 262
<code>\glsentryshort</code>	151	<code>\gls hyperlink</code>	154
<code>\glsentryshortaccess</code>	326	<code>\gls hypernavsep</code>	259
<code>\Glsentryshortpl</code>	151	<code>\gls hypernumber</code>	36, 205
<code>\glsentryshortpl</code>	151	<code>\glsifhyper</code>	104
<code>\glsentryshortpluralaccess</code>	327	<code>\glsifhyperon</code>	104, 107
<code>\glsentrysort</code>	150	<code>\glsIfListOfAcronyms</code>	14
<code>\Glsentrysymbol</code>	149	<code>\glsifusedtranslatordict</code>	22
<code>\glsentrysymbol</code>	148	<code>\glsignore</code>	155
<code>\glsentrysymbolaccess</code>	326	<code>\glsinlinedescformat</code>	261
<code>\Glsentrysymbolplural</code>	149	<code>\glsinlinedopostchild</code>	260, 261
<code>\glsentrysymbolplural</code>	149	<code>\glsinlineemptydescformat</code>	262
<code>\glsentrysymbolpluralaccess</code>	326	<code>\glsinlinenameformat</code>	261
<code>\Glsentrytext</code>	148	<code>\glsinlineparentchildseparator</code>	261
<code>\glsentrytext</code>	50, 148, 178		
<code>\glsentrytextaccess</code>	326	<code>\glsinlinepostchild</code>	261
<code>\glsentrytype</code>	149	<code>\glsinlineseparator</code>	261
<code>\Glsentryuseri</code>	150	<code>\glsinlinesubdescformat</code>	262
<code>\glsentryuseri</code>	150	<code>\glsinlinesubnameformat</code>	262
<code>\Glsentryuserii</code>	150	<code>\glsinlinesubseparator</code>	261
<code>\glsentryuserii</code>	150	<code>\glskeylisttok</code>	212
<code>\Glsentryuseriii</code>	150	<code>\glslabeltok</code>	212
<code>\glsentryuseriii</code>	150	<code>\glsletentryfield</code>	146
<code>\Glsentryuseriv</code>	151	<code>\glslink</code>	92, 93, 105, 118, 154, 203, 205
<code>\glsentryuseriv</code>	150	<code>\glslink options</code>	
<code>\Glsentryuserv</code>	151	<code>counter</code>	103, 118, 251
<code>\glsentryuserv</code>	151	<code>format</code>	104, 118, 205
<code>\Glsentryuservi</code>	151	<code>hyper</code>	104, 107, 118

local	104	\glspar	34
\glslinkcheckfirsthyperhook	106	\glsppercentchar	156
\glslinkpostsetkeys	106	\GLSpl	122
\glslinkvar	104	\Glspl	122
\glslistdottedwidth	265	\glspl	92, 121, 122
\glsllocalreset	85	\GLSplural	127
\glsllocalresetall	86	\Glsplural	126
\glsllocalunset	85	\glsplural	126, 127
\glsllocalunsetall	87	\glspluralaccessdisplay	328
\glslongaccessdisplay	328	\glspluralsuffix	31, 60, 61
\glslongaccesskey	356	\glspostdescription	9
\glslongkey	209	\glspostinline	261
\glslongpluralaccessdisplay	329	\glspostlinkhook	106
\glslongpluralaccesskey	356	\glsprestandardsort	10
\glslongpluralkey	209	\glsglquote	156
\glslongtok	213	\glsglrefentry	197
\glsmcols	277	\glsglreset	84
\glsmoveentry	82	\glsglresetall	86
\GLSname	129	\glsglresetentrylist	196
\Glsname	128	\glsglresetsubentrycounter	196, 197
\glsglname	128, 129	\glsglsee	177
\glsglnameaccessdisplay	327	\glsglseeformat	158, 178
\glsglnamefont	205, 293	\glsglseeitem	178
\glsglnavhyperlink	258	\glsglseeitemformat	178
\glsglnavhypertarget	258	\glsglseelastsep	178
\glsglnavigation	259	\glsglseelist	178
\glsglnextpages	196	\glsglseesep	178
\glsglnoexpandfields	66	\glsglSetAlphaCompositor	35
\glsglnoidxdisplayloc	192	\glsglSetCompositor	34, 35
\glsglnoidxdisplayloclisthandler	191	\glsglsetexpandfield	17
\glsglnoidxloclist	191	\glsglsetnoexpandfield	17
\glsglnoidxloclisthandler	191	\glsglSetSuffixF	35
\glsglnoidxnumberlistloophandler	170	\glsglSetSuffixFF	35
\glsglnoidxstripaccents	19	\glsglsettoctitle	30
\glsglnoexpandpages	196	\glsglsetwidest	298
\glsglnoexpandwarning	40	\GlsSetXdyCodePage	48
\glsglnumberformat	36	\GlsSetXdyFirstLetterAfterDigits	156
\glsglnumberlistloop	169	\GlsSetXdyLanguage	48
\glsglnumbersgroupname	31, 155, 156, 202	\GlsSetXdyLocationClassOrder	46
\glsglnumlistlastsep	153	\GlsSetXdyMinRangeLength ...	156
\glsglnumlistsep	153	\GlsSetXdyStyles	47
\glsglopenbrace	156	\glsglshortaccessdisplay	328
\glsglorder	24	\glsglshortaccesskey	356
\glsglorg@endtheglossary	5	\glsglshortkey	209
\glsglorg@theglossary	5	\glsglshortpluralaccessdisplay	328
\glsglspagelistwidth	266, 272, 281, 288	\glsglshortpluralaccesskey ...	356
		\glsglshortpluralkey	209
		\glsglshorttok	212

<code>\glssortnumberfmt</code>	11
<code>\glsspace</code>	210
<code>\glsstepeentry</code>	197
<code>\glssubentry</code>	197
<code>\glssubentrycounterlabel</code> ...	198
<code>\glssubentryitem</code>	198
<code>\GLSsymbol</code>	131
<code>\Glsymbol</code>	131
<code>\glssymbol</code>	131
<code>\glssymbolaccessdisplay</code>	328
<code>\glssymbolnav</code>	259
<code>\GLSsymbolplural</code>	132
<code>\Glsymbolplural</code>	132
<code>\glssymbolplural</code>	132
<code>\glssymbolpluralaccessdisplay</code>	328
<code>\glssymbolsgroupname</code>	
.....	31, 155, 156, 202
<code>\glstarget</code>	198
<code>\GLStext</code>	125
<code>\Glstext</code>	125
<code>\glstext</code>	124
<code>\glstextaccessdisplay</code>	327
<code>\glstextformat</code>	93
<code>\glstextup</code>	209
<code>\glstildechar</code>	156
<code>\glstreeindent</code>	296, 297
<code>\glstreenamebox</code>	298
<code>\glstreenamefmt</code>	293
<code>\glunset</code>	85
<code>\glunsetall</code>	87
<code>\glsupacrpluralsuffix</code>	31
<code>\GlsUseAcrEntryDispStyle</code> ...	215
<code>\GlsUseAcrStyleDefs</code>	215
<code>\GLSuseri</code>	133
<code>\Glsuseri</code>	133
<code>\gluseri</code>	132, 133
<code>\GLSuserii</code>	134
<code>\Glsuserii</code>	134
<code>\gluserii</code>	133, 134
<code>\GLSuseriii</code>	135
<code>\Glsuseriii</code>	135
<code>\gluseriii</code>	134, 135
<code>\GLSuseriv</code>	136
<code>\Glsuseriv</code>	135
<code>\gluseriv</code>	135, 136
<code>\GLSuserv</code>	137
<code>\Glsuserv</code>	136
<code>\gluserv</code>	136, 137

<code>\GLSuservi</code>	138
<code>\Glsuservi</code>	137
<code>\gluservi</code>	137, 138
<code>\glswrite</code>	166
<code>\glswritedefhook</code>	75
<code>\glswriteentry</code>	173

H

<code>\hyperbf</code>	207
<code>\hyperemph</code>	207
<code>hyperfirst (option)</code>	23
<code>\hyperit</code>	207
<code>\hyperlink</code>	117
<code>\hypermd</code>	207
<code>\hyperpage</code>	205
<code>hyperref package</code> ...	176, 179, 205, 251
<code>\hyperrm</code>	207
<code>\hypersc</code>	207
<code>\hypersf</code>	207
<code>\hypersl</code>	207
<code>\hypertarget</code>	117
<code>\hypertt</code>	207
<code>\hyperup</code>	207

I

<code>\if@gl@docloaded</code>	4
<code>\if@gl@isacronymlist</code>	15
<code>\ifglossaryexists</code>	50
<code>\ifgl@descsuppressed</code>	53
<code>\ifgl@entryexists</code>	51
<code>\ifgl@fieldcseq</code>	75
<code>\ifgl@fielddefeq</code>	75
<code>\ifgl@fieldeq</code>	74
<code>\ifgl@haschildren</code>	52
<code>\ifgl@hasdesc</code>	53
<code>\ifgl@hasfield</code>	54
<code>\ifgl@haslong</code>	54
<code>\ifgl@hasparent</code>	53
<code>\ifgl@hasprefix</code>	253
<code>\ifgl@hasprefixfirst</code>	253
<code>\ifgl@hasprefixfirstplural</code> .	253
<code>\ifgl@hasprefixplural</code>	253
<code>\ifgl@hasshort</code>	54
<code>\ifgl@hasymbol</code>	53
<code>\ifgl@stranslate</code>	21
<code>\ifgl@sused</code>	51, 84
<code>\ifgl@xindy</code>	25
<code>\ifignoredglossary</code>	59
<code>index (option)</code>	27
<code>index (style)</code>	294

indexgroup (style) 295
 indexhypergroup (style) 295
 indexonlyfirst (option) 23
 \indexspace 262, 277, 293
 inline (style) 260
 \inputencodingname 25
 \istfile 170
 \istfilename 34
 \item 205, 262, 294

L

link text 93
 list (style) 262
 listdotted (style) 264
 listgroup (style) 263
 listhypergroup (style) 263
 \loadglsentries 13, 93
 long (key) 63
 long (style) 266
 long-sc-short (acrstyle) 218
 long-sc-short-desc (acrstyle) ..
 219, 344
 long-short (acrstyle) 215, 342
 long-short-desc (acrstyle) . 218, 344
 long-sm-short (acrstyle) 218
 long-sm-short-desc (acrstyle) ..
 219, 344
 long-sp-short (acrstyle) 216
 long-sp-short-desc (acrstyle) .. 219
 long3col (style) 267
 long3colborder (style) 268
 long3colheader (style) 268
 long3colheaderborder (style) .. 268
 long4col (style) 268
 long4colborder (style) 269
 long4colheader (style) 269
 long4colheaderborder (style) .. 270
 longaccess (key) 323
 longborder (style) 266
 longheader (style) 267
 longheaderborder (style) 267
 \longnewglossaryentry 60, 76
 longplural (key) 63
 longpluralaccess (key) 323
 \longprovideglossaryentry ... 76
 longragged (style) 272
 longragged3col (style) 273
 longragged3colborder (style) .. 274
 longragged3colheader (style) .. 274

longragged3colheaderborder
 (style) 274
 longraggedborder (style) 272
 longraggedheader (style) 273
 longraggedheaderborder (style) 273
 longtable (environment)
 8, 244, 265–276
 longtable package 265, 271

M

makeglossaries 24, 34, 48, 56, 165, 182
 \makeglossaries
 ... 26, 29, 34, 35, 55, 58, 165, 166
 \makeglossary 166
 makeindex 359
 makeindex
 . 10, 25, 26, 31, 34–36, 56, 58,
 60, 84, 109, 112, 155, 158, 161,
 162, 171, 175, 176, 201, 202, 303
 delim_n 36
 delim_r 36
 page_compositor 34
 special characters ... 110, 111, 155
 makeindex (option) 25
 \makenoidxglossaries 21, 167
 mcolalttree (style) 280
 mcolalttreegroup (style) 280
 mcolalttreehypergroup (style) . 280
 mcolindex (style) 277
 mcolindexgroup (style) 277
 mcolindexhypergroup (style) ... 277
 mcoltree (style) 278
 mcoltreegroup (style) 278
 mcoltreehypergroup (style) 278
 mcoltreenoname (style) 279
 mcoltreenonamegroup (style) ... 279
 mcoltreenonamehypergroup
 (style) 279
 memoir class 172
 mfirstuc package 1
 multicol package 277
 multicol (environment) 277

N

name (key) 60
 \new@glossaryentry 67
 \newacronym .. 24, 63, 92, 93, 208, 209
 \newacronymhook 213
 \newacronymstyle 215
 \newglossary 15, 56, 58, 162, 165, 192

`\newglossaryentry` 60, 66, 92, 93, 208
`\newglossaryentry options`
 `access` 324, 325
 `counter` 61
 `description` 24, 60,
 64, 66, 76, 129, 147, 209, 237, 323
 `descriptionaccess` 326, 328
 `descriptionplural` 130, 323
 `descriptionpluralaccess` .. 326, 328
 `first` 61,
 80, 118, 125, 149, 235, 240, 322
 `firstaccess` 326, 328
 `firstplural` 61, 127, 149, 323
 `firstpluralaccess` 326, 328
 `format` 157
 `long` 99, 151, 323
 `longaccess` 327, 328
 `longplural` 152, 323
 `longpluralaccess` 327, 329
 `name` 60, 64, 66, 76, 128, 146, 178, 322
 `nonumberlist` 62
 `parent` 62, 66
 `plural` 60, 80, 126, 323
 `pluralaccess` 326, 328
 `prefix` 252
 `prefixfirst` 252
 `prefixfirstplural` 252
 `prefixplural` 252
 `see` 8, 62, 165, 167
 `short` 99, 151, 323
 `shortaccess` 326, 328
 `shortplural` 151, 323
 `shortpluralaccess` 327, 328
 `sort` 60, 150, 201, 202
 `symbol` 60, 61, 131, 230–
 232, 235, 240, 268, 284, 322–324
 `symbolaccess` 326, 328
 `symbolplural` 132, 323
 `symbolpluralaccess` 326, 328
 `text` 60,
 61, 118, 124, 148, 230, 235, 322
 `textaccess` 326, 327
 `type` 12, 61, 92, 149
 `user1` 132, 150, 324
 `user2` 133, 150
 `user3` 134, 150
 `user4` 135, 150
 `user5` 136, 151
 `user6` 137, 151, 324

`\newglossarystyle` 204
`\newignoredglossary` 59
`nogroupskip (option)` 9
`nohypertypes (option)` 16
`\noist` 162, 251, 308
`nolist (option)` 8
`nolong (option)` 8
`nomain (option)` 13
`nonumberlist (key)` 62
`nonumberlist (option)` 7
`\nopostdesc` 33
`nopostdot (option)` 9
`noredefwarn (option)` 17
`nostyles (option)` 8
`nosuper (option)` 8
`notranslate (option)` 22
`notree (option)` 8
`nowarn (option)` 16
`\ns@newglossary` 57
`numberedsection (option)` 6
`numberline (option)` 5
`numbers (option)` 27

O

`\oldacronym` 208
`order (option)` 24

P

`\p@gl@s@hyp@opt` 105
package options:
 `acronym` 13, 30, 179, 208
 `true` 14
 `acronym` 13
 `acronymlists` 15
 `acronyms` 14
 `automake` 26
 `compatible-2.07` 26
 `compatible-3.07` 26
 `counter` 16
 `counter` 16
 `description` 234, 235
 `description` 24
 `dua` 233–235
 `dua` 24
 `entrycounter` 194, 196
 `true` 9
 `entrycounter` 9
 `entrycounterwithin` 9
 `footnote` 119–124, 231, 233, 234, 237
 `footnote` 23

hyperfirst		
false	119–124	
hyperfirst	23	
index	27	
indexonlyfirst	366	
indexonlyfirst	23	
makeindex	159, 251	
makeindex	25	
nogroupskip	9	
nohypertypes	16	
nolist	244	
nolist	8	
nolong	244, 265	
nolong	8	
nomain	12, 13	
nomain	13	
nonumberlist	7	
nonumberlist	7	
nopostdot	9	
noredefwarn	17	
nostyles	8	
nosuper	244	
nosuper	8	
notranslate	22	
notree	245	
notree	8	
nowarn	16	
numberedsection	6	
numberline	5	
numberline	5	
numbers	27	
order	24	
sanitize	19, 60, 146, 147	
sanitize	21	
sanitizesort	17	
sanitizesort	20	
savenumberlist	7	
savewrites	26, 364	
false	163	
true	165, 170, 171	
savewrites	26	
section	6, 38	
section	6	
seeautonumberlist	8	
shotcuts	24	
smallcaps	24	
smaller	24	
sort		
def	10, 11	
standard	10	
use	10, 11	
sort	10	
style	7, 244, 245	
style	7	
subentrycounter	194, 196	
subentrycounter	9	
symbols	27	
toc	5	
true	5	
toc	5	
translate	22	
false	22	
translate	22	
translator	21	
ucmark	9	
xindy	25, 159, 251	
xindy	25	
xindygloss	25	
xindynoglsnumbers	26	
\pagelistname	30	
parent (key)	62	
\PGLS	256	
\Pgls	255	
\pgls	253	
\PGLSpl	257	
\Pglspl	255	
\pglspl	254	
\phantomsection	37–39	
plural (key)	60	
pluralaccess (key)	323	
polyglossia package	21, 32	
\printacronyms	13	
\PrintChanges	5	
\printglossaries		
.	12, 36, 55, 59, 166, 179, 180, 257	
\printglossary	36, 37,	
	56, 59, 166, 179, 182, 192, 257, 258	
\printglossary options		
entrycounter	193	
nogroupskip	193	
nonumberlist	195	
nopostdot	193	
numberedsection	193	
style	193	
subentrycounter	194	
title	192	
toctitle	192	
type	12, 179, 192	

\printnoidxglossaries	180	\setentrycounter	203
\printnoidxglossary	180, 192	\SetFootnoteAcronymDisplayStyle	235
\printnoidxglossary options		\SetFootnoteAcronymStyle ...	237
sort	195	\SetGenericNewAcronym	213
\provideglossaryentry	67	\setglossarypreamble	36
\ProvidesGlossariesLang	31	\setglossarysection	38
R			
\renewacronymstyle	215	\setglossarystyle	203
\renewglossarystyle	204	\setglossentrycompatibility	200
\RequireGlossariesLang	31	\SetSmallAcronymDisplayStyle	238
\roman	44	\SetSmallAcronymStyle	240
S			
\s@glshyp@opt	105	\setStyleFile	34
\s@newglossary	57	\setupglossaries	28
sanitize (option)	21	short (key)	63
sanitizesort (option)	20	short-long (acrstyle)	217, 343
savenumberlist (option)	7	short-long-desc (acrstyle) .	220, 345
savewrites (option)	26	shortaccess (key)	323
sc-short-long (acrstyle)	218	shortplural (key)	63
sc-short-long-desc (acrstyle) .		shortpluralaccess (key)	323
.....	220, 345	shotcuts (option)	24
\scantokens	50	\showacronymlists	249
\section	50	\showglocounter	246
section (option)	6	\showglodesc	247
see (key)	62	\showglodescaccess	357
seeautonumberlist (option)	8	\showglodescplural	248
\seename	31	\showglodescpluralaccess ...	357
\SetAcronymLists	15	\showglofield	249
\SetAcronymStyle	14, 242	\showglofirst	246
\setacronymstyle	214	\showglofirstaccess	357
\SetCustomDisplayStyle	243	\showglofirsttpl	246
\SetCustomStyle	244	\showglofirstpluralaccess ..	357
\SetDefaultAcronymDisplayStyle	227	\showgloflag	249
\SetDefaultAcronymStyle	228	\showgloindex	249
\SetDescriptionAcronymDisplayStyle	233	\showglolevel	245
\SetDescriptionAcronymStyle	234	\showgloloclist	249
\SetDescriptionDUAAcronymDisplayStyle	231	\showglolong	248
\SetDescriptionDUAAcronymStyle	232	\showglolongaccess	358
\SetDescriptionFootnoteAcronymDisplayStyle	229	\showglolongpluralaccess ...	358
\SetDescriptionFootnoteAcronymStyle	230	\showgloname	247
\SetDUADisplayStyle	241	\showglonameaccess	356
\SetDUASyle	242	\showgloparent	245
		\showgloplural	245
		\showglopluralaccess	357
		\showgloshort	248
		\showgloshortaccess	357
		\showgloshortpluralaccess ..	358
		\showglosort	248
		\showglossaries	249
		\showglossarycounter	250

`\showglossaryentries` 250
`\showglossaryin` 250
`\showglossaryout` 250
`\showglossarytitle` 250
`\showglosymbol` 248
`\showglosymbolaccess` 357
`\showglosymbolplural` 248
`\showglosymbolpluralaccess` . 357
`\showglotext` 245
`\showglotextaccess` 357
`\showglotype` 246
`\showglouserii` 246
`\showglouseriii` 246
`\showglouseriv` 247
`\showglouserv` 247
`\showglouservi` 247
`sm-short-long` (acrstyle) 218
`sm-short-long-desc` (acrstyle) ..
..... 220, 345
`smallcaps` (option) 24
`smaller` (option) 24
`\SmallNewAcronymDef` 239, 355
`sort` (key) 60
`sort` (option) 10
`style` (option) 7
`subentrycounter` (option) 9
`\subglossentry` 199
`\subitem` 294
`sublistdotted` (style) 265
`\subsubitem` 294
`super` (style) 281
`super3col` (style) 283
`super3colborder` (style) 283
`super3colheader` (style) 284
`super3colheaderborder` (style) . 284
`super4col` (style) 284
`super4colborder` (style) 285
`super4colheader` (style) 285
`super4colheaderborder` (style) . 286
`superborder` (style) 282
`superheader` (style) 282
`superheaderborder` (style) 282
`superragged` (style) 288
`superragged3col` (style) 290
`superragged3colborder` (style) . 290
`superragged3colheader` (style) . 291
`superraggedborder` (style) 289
`superraggedheader` (style) 289

`superraggedheaderborder` (style) 289
`superraggedright3colheaderborder`
(style) 291
`supertabular` (environment) ...
..... 8, 244, 281–293
`supertabular` package .. 8, 244, 281, 288
`symbol` (key) 61
`symbolaccess` (key) 323
`\symbolname` 30
`symbolplural` (key) 61
`symbolpluralaccess` (key) 323
`symbols` (option) 27

T

`text` (key) 60
`textaccess` (key) 322
`textcase` package 4
`\theequation` 176
`theglossary` (environment)
..... 5, 16, 36, 37,
198, 204, 278, 279, 295, 296, 298
`\theHequation` 176
`theindex` (environment) 294
`toc` (option) 5
`tracklang` package 31, 358, 359
`\translate` 32
`translate` (option) 22
`translator` package
..... 12–14, 22, 27, 31–33, 179
`tree` (style) 295
`treegroup` (style) 296
`treehypergroup` (style) 296
`treenoname` (style) 297
`treenonamegroup` (style) 297
`treenonamehypergroup` (style) .. 298
`type` (key) 61

U

`ucmark` (option) 9
`user1` (key) 62
`user2` (key) 63
`user3` (key) 63
`user4` (key) 63
`user5` (key) 63
`user6` (key) 63

W

`\warn@nomakeglossaries` 164
`\warn@noprintglossary` 179
`\writeist` 34, 41, 43, 46, 157, 302, 304

X

\xglsacccsupp	327	171, 175, 176, 182, 201, 251, 303
xindy	359	xindy (option) 25
xindy . 10, 25, 26, 34, 35, 40, 43, 45,		xindygloss (option) 25
47–49, 83, 115, 116, 156–158,		xindynoglsnumbers (option) 26
		xspace package 4, 208