

Documented Code For glossaries v4.16

Nicola L.C. Talbot

Dickimaw Books

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

2015-07-08

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 v4.16: \TeX 2e 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	59
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	92
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	153
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	Mfirstuc Documented Code	257
4	Mfirstuc-english Documented Code	260
5	Glossary Styles	261
5.1	Glossary hyper-navigation definitions (glossary-hypernav pack- age)	261
5.2	In-line Style (glossary-inline.sty)	263
5.3	List Style (glossary-list.sty)	266
5.4	Glossary Styles using longtable (the glossary-long package) . . .	269
5.5	Glossary Styles using longtable (the glossary-longragged package)	275
5.6	Glossary Styles using multicol (glossary-mcols.sty)	280
5.7	Glossary Styles using supertabular environment (glossary-super package)	284

5.8	Glossary Styles using supertabular environment (glossary-superragged package)	291
5.9	Tree Styles (glossary-tree.sty)	297
6	glossaries-compatible-207	305
7	Accessibility Support (glossaries-accsupp Code)	325
7.1	Defining Replacement Text	326
7.2	Accessing Replacement Text	329
7.3	Displaying the Glossary	344
7.4	Acronyms	346
7.5	Debugging Commands	360
8	Multi-Lingual Support	362
8.1	Polyglossia Captions	362
	Glossary	363
	Change History	363
	Index	388

1 Main Package Code

1.1 Package Definition

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

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries}[2015/07/08 v4.16 (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
```

```
\glsorg@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 redefined \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](#).)

`\@glossary@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”:

`\glossaryentrynumbers`

```

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 ignore 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*{\@gls@seeautonumberlist{}
```

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

```
77 \@gls@declareoption{seeautonumberlist}{%  
78   \renewcommand*{\@gls@seeautonumberlist}{%  
79     \def\@gls@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 `\gls glossarymark`

```

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
```

`lo@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 argu-
  ment 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 }
```

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 }

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

norededefwarn Suppress warning on redefinition of \printglossary
317 \@gls@declareoption{norededefwarn}{%
318   \renewcommand*{\@gls@warnonglossdefined}{}%
319   \renewcommand*{\@gls@warnontheGLOSSdefined}{}%
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 \glssetnoexpandfield{name}

```

@gls@sanitizesymbol

```

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

```

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

```

339 \glssetnoexpandfield{symbol}

```

@symbolplural@field

```

340 \glssetnoexpandfield{symbolplural}

```

Sanitizing stuff:

\@gls@sanitizesort

```

341 \newcommand*{\@gls@sanitizesort}{%
342   \ifglssanitizesort
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 }

```

oidx@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 }

```

lsnoidxstripaccents

```

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      \glsssetexpandfield{sortvalue}%
448    }%
449    \fi
450 }

```

Default setting:

```

451 \glssanitizesorttrue
452 \glsssetnoexpandfield{sortvalue}%

```

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

```

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

457 \define@choicekey{gls}{sanitimize}{sort}{true,false}[true]{%
458   \setbool{glssanitizesort}{#1}%
459   \ifglssanitizesort
460     \glsssetnoexpandfield{sortvalue}%
461   \else
462     \glsssetexpandfield{sortvalue}%
463   \fi
464   \GlossariesWarning{sanitimize={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       \glsssetexpandfield{name}%
472       \glsssetexpandfield{symbol}%
473       \glsssetexpandfield{symbolplural}%
474       \glsssetexpandfield{desc}%
475       \glsssetexpandfield{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 }
```

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

```
505 \@gls@declareoption{nottranslate}{%
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   }
```

```

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@glstrue}
```

`\@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   \glsxindytrue
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   \ifglssautomake
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 \glssautomakefalse
```

\@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][]{%
684     \newglossaryentry{##2}%
685     {type={index},name={##2},description={\nopostdesc},##1}}
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}
```

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

```
757 \newcommand*{\glssettocitle}[1]{%
758   \def\glossarytocitle{\csname @gls@#1@title\endcsname}}
```

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

`\entryname`
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:

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

`glsnumbersgroupname`
764 `\providecommand*{\glsnumbersgroupname}{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.ldf}{\input{glossaries-#1.ldf}}{}`
772 `}`

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

dglossarytocaptions 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*{\glstttitle}[1]{%
828     \ifcsdef{gls@tr@set@#1@ttitle}%
829     {%
830       \csuse{gls@tr@set@#1@ttitle}%
831     }%
832     {%
833       \def\glossaryttitle{\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*{\gl@istfilebase}{#1}%
  Just in case \istfilename has been modified.
860   \ifglsxindy
861     \def\istfilename{\gl@istfilebase.xdy}
862   \else
863     \def\istfilename{\gl@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{\gl@istfilebase.xdy}
869 \else
870   \def\istfilename{\gl@istfilebase.ist}
871 \fi
```

`\gl@istfilebase`

```
872 \newcommand*{\gl@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}
```

`\glsSetAlphaCompositor` Sets the alpha compositor.

```
879 \ifglsxindy
880   \newcommand*\glsSetAlphaCompositor[1]{%
881     \renewcommand*\@glsAlphacompositor{#1}}
882 \else
883   \newcommand*\glsSetAlphaCompositor[1]{%
884     \glsnoxywarning\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][\glsdefaulttype]{%
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   \gls@clearpage
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 \ifglxsindy
1023   \edef\@xdyattributes{\string"default\string"}%
1024 \fi

```

\@xdyattributelist Comma-separated list of attributes.

```

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

```

\@xdylocref Define list of markup location references.

```

1028 \ifglxsindy
1029   \def\@xdylocref{}
1030 \fi

```

\@gls@ifinlist

```

1031 \newcommand*\@gls@ifinlist[4]{%
1032   \def\@do@ifinlist##1,#1,##2\end@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@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 \ifglxsindy
1043   \newcommand*\@xdycounters{\glscounter}
1044   \newcommand*\GlsAddXdyCounters[1]{%
1045     \@for\@gls@ctr:=#1\do{%

```

Check if already in list before adding.

```

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@glssaddxdycounters` Counters must all be identified before adding attributes.

```

1062 \newcommand*\@disabled@glssaddxdycounters{%
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 \ifglsxindy

```

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

```

1069 \newcommand*\@glssaddxdyattribute[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 \glsnoxindywarning\GlsAddXdyAttribute}
1102 \fi

```

redefinedattributes Add known attributes for all defined counters

```

1103 \ifglsxindy
1104 \newcommand*\@gls@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{hyper rm}
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{glsignore}
1128 }
1129 \else
1130 \let\@gls@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       \glsnnoxindywarning\GlsAddXdyAlphabet}
1141   \fi
```

This code is only required for xindy:

```
1142 \ifglsxindy
```

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

```
1143   \newcommand*{\@glx@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 `\@glx@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\@glx@roman{\@roman{0}\string"
1153     \string"roman-numbers-lowercase\string" :sep \string"}%
1154   \@onelevel@sanitize\@glx@roman
1155   \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
1156     :sep \string"}%
1157   \@onelevel@sanitize\@tmp
1158   \ifx\@tmp\@glx@roman
1159     \expandafter
1160       \edef\csname @glx@xdy@Lclass@roman-page-numbers\endcsname{%
1161         \string"roman-numbers-lowercase\string"%
1162       }%
1163   \else
1164     \expandafter
```

```

1165     \edef\csname @gls@xdy@Lclass@roman-page-numbers\endcsname{
1166         :sep \string"@gls@roman\string"%
1167     }%
1168 \fi

@Roman-page-numbers Upper case Roman numerals (I, II, ...).
1169 \expandafter\def\csname @gls@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 }%

pendix-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 }

bic-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{}

dyuserlocationnames List of additional user location names
1192 \def\@xdyuserlocationnames{}

```

End of xindy-only block:

1193 \fi

\GlsAddXdyLocation \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 \ifglxindy
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"{}\glssopenbrace\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 "\glssopenbrace"
1209           #1
1210           :sep "\glsclosebrace\glssopenbrace" #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     \glsnxindywarning\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     \glsnnoxindywarning\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     \glsnnoxindywarning\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     \glsnnoxindywarning\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     \glsnxindywarning\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 \ifglxindy
1276   \newcommand*\GlsSetXdyLanguage[2][\glstypetype]{%
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     \glsnxindywarning\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 \ifglxindy
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 \glsnoxywarning\GlsSetXdyCodePage}
1300 \fi

```

`\@xdylettergroups` Store letter group definitions.

```

1301 \ifglxindy
1302 \ifglx@xindy@glxnumbers
1303 \def\@xdylettergroups{(define-letter-group
1304 \string\glxnumbers\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"@glxfirstletter\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:=\@glsacronymlists\do{\ifx#1\@empty\else#2\fi}%
1324 }
```

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

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

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

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

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

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

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

```
1332 \newcommand*\forallglsentries[3][\@glo@types]{%
1333   \expandafter\forallglossaries\expandafter[#1]{\@this@glo@}%
1334   {%
1335     \forallglsentries[\@this@glo@]{#2}{#3}%
1336   }%
1337 }
```

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

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

where *<type>* 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 `\glsentrytext` will no longer be usable in sectioning, caption etc commands. If the user really wants to be able to construct a label with UTF8 characters, allow them the means to do so (but on their own head be it, if they then use entries in `\section` etc). This can be done via:

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

(Note, don't use `\detokenize` or it will cause commands like `\glsaddall` to fail.) Since 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}
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```

1354 \newcommand{\glsdoifnoexists}[2]{%
1355   \ifglstryexists{#1}{%
1356     \PackageError{glossaries}{Glossary entry ‘\glsdetoklabel{#1}’ has already
1357     been defined}{-}{#2}%
1358 }

```

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

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

```

1359 \newcommand{\glsdoifexistsorwarn}[2]{%
1360   \ifglstryexists{#1}{#2}{%
1361     \GlossariesWarning{Glossary entry ‘\glsdetoklabel{#1}’
1362     has not been defined}%
1363   }%
1364 }

```

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

```

1365 \newcommand{\ifglshaschildren}[3]{%
1366   \glsdoifexists{#1}%
1367   {%
1368     \def\do@glshaschildren{#3}%
1369     \edef\@gls@thislabel{\glsdetoklabel{#1}}%
1370     \expandafter\forglstryentries\expandafter
1371     [\csname glo@\@gls@thislabel @type\endcsname]
1372     {\glo@label}%
1373     {%
1374       \letcs\glo@parent{glo@\glo@label @parent}%
1375       \ifdefequal\@gls@thislabel\glo@parent
1376       {%
1377         \def\do@glshaschildren{#2}%
1378         \@endfortrue
1379       }%
1380     }%
1381   }%
1382   \do@glshaschildren
1383 }%
1384 }

```

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

```

1385 \newcommand{\ifglshasparent}[3]{%
1386   \glsdoifexists{#1}%
1387   {%
1388     \ifcsempy{glo@\glsdetoklabel{#1}@parent}{#3}{#2}%

```

```

1389 }%
1390 }

\ifglshasdesc \ifglshasdesc{<label>}{<true part>}{<false part>}
1391 \newcommand*{\ifglshasdesc}[3]{%
1392 \ifcsequal{glo@\glsdetoklabel{#1}@desc}%
1393 {#3}%
1394 {#2}%
1395 }

ifglsdessuppressed \ifglsdessuppressed{<label>}{<true part>}{<false part>} Does <true part>
if the description is just \nopostdesc otherwise does <false part>.
1396 \newcommand*{\ifglsdessuppressed}[3]{%
1397 \ifcsequal{glo@\glsdetoklabel{#1}@desc}{@no@post@desc}%
1398 {#2}%
1399 {#3}%
1400 }

\ifglshassymbol \ifglshassymbol{<label>}{<true part>}{<false part>}
1401 \newcommand*{\ifglshassymbol}[3]{%
1402 \letcs{\@glo@symbol}{glo@\glsdetoklabel{#1}@symbol}%
1403 \ifdefempty\@glo@symbol
1404 {#3}%
1405 {%
1406 \ifdefequal\@glo@symbol\@gls@default@value
1407 {#3}%
1408 {#2}%
1409 }%
1410 }

\ifglshaslong \ifglshaslong{<label>}{<true part>}{<false part>}
1411 \newcommand*{\ifglshaslong}[3]{%
1412 \letcs{\@glo@long}{glo@\glsdetoklabel{#1}@long}%
1413 \ifdefempty\@glo@long
1414 {#3}%
1415 {%
1416 \ifdefequal\@glo@long\@gls@default@value
1417 {#3}%
1418 {#2}%
1419 }%
1420 }

\ifglshasshort \ifglshasshort{<label>}{<true part>}{<false part>}
1421 \newcommand*{\ifglshasshort}[3]{%
1422 \letcs{\@glo@short}{glo@\glsdetoklabel{#1}@short}%
1423 \ifdefempty\@glo@short
1424 {#3}%
1425 {%

```

```

1426 \ifdefequal\@glo@short\@gls@default@value
1427 {#3}%
1428 {#2}%
1429 }%
1430 }

```

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

```

1431 \newcommand*{\ifglshasfield}[4]{%
1432 \glsdoifexists{#2}%
1433 {%
1434 \letcs{\@glo@thisvalue}{gls@detoklabel{#2}@#1}%

```

First check supplied field label is defined.

```

1435 \ifdef\@glo@thisvalue
1436 {%

```

Is defined, so now check if empty.

```

1437 \ifdefempty\@glo@thisvalue
1438 {%

```

Is empty, so doesn't have field set.

```

1439 #4%
1440 }%
1441 {%

```

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

```

1442 \ifdefequal\@glo@thisvalue\@gls@default@value{#4}{#3}%
1443 }%
1444 }%
1445 {%

```

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

```

1446 \@gls@fetchfield{\@gls@thisfield}{#1}%

```

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

```

1447 \ifdef\@gls@thisfield
1448 {%

```

Is defined, so now check if empty.

```

1449 \letcs{\@glo@thisvalue}{gls@detoklabel{#2}@\@gls@thisfield}%
1450 \ifdefempty\@glo@thisvalue
1451 {%

```

Is empty so field hasn't been set.

```

1452 #4%
1453 }%
1454 {%

```

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

```

1455         \ifdefequal\@glo@thisvalue\@gls@default@value{#4}{#3}%
1456     }%
1457 }%
1458 {%

```

Not defined.

```

1459     \GlossariesWarning{Unknown entry field '#1'}%
1460     #4%
1461 }%
1462 }%
1463 }%
1464 }

```

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

```

1465 \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.

```

1466 \newcommand*{\@gls@provide@newglossary{%
1467     \protected@write\@auxout{}\string\providecommand\string\@newglossary[4]{}%

```

Only need to do this once.

```

1468     \let\@gls@provide@newglossary\relax
1469 }

```

\defglsentryfmt Allow different glossaries to have different display styles.

```

1470 \newcommand*{\defglsentryfmt}[2][\glsdefaulttype]{%
1471     \csgdef{gls@#1@entryfmt}{#2}%
1472 }

```

\gls@doentryfmt

```

1473 \newcommand*{\gls@doentryfmt}[1]{\csuse{gls@#1@entryfmt}}

```

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

```

1474 \newcommand*{\@gls@forbidtexext}[1]{%
1475     \ifboolexpr{test {\ifdefstring{#1}{tex}}}

```

```

1476         or test {\ifdefstring{#1}{TEX}}
1477 {%
1478   \def#1{nottex}%
1479   \PackageError{glossaries}%
1480     {Forbidden '.tex' extension replaced with '.nottex'}%
1481     {I'm sorry, I can't allow you to do something so reckless.\MessageBreak
1482       Don't use '.tex' as an extension for a temporary file.}%
1483 }%
1484 {%
1485 }%
1486 }

```

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`

```

1487 \newcommand*{\newglossary}{\@ifstar\s@newglossary\@ns@newglossary}

```

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

```

1488 \newcommand*{\s@newglossary}[2]{%
1489   \ns@newglossary[#1-glg]{#1}{#1-gls}{#1-glo}{#2}%
1490 }

```

`\ns@newglossary` Define the unstarred version.

```

1491 \newcommand*{\ns@newglossary}[5][glg]{%
1492   \ifglossaryexists{#2}%
1493   {%
1494     \PackageError{glossaries}{Glossary type '#2' already exists}{%
1495       You can't define a new glossary called '#2' because it already
1496       exists}%
1497   }%
1498   {%

```

Check if default has been set

```

1499   \ifundef\glsdefaulttype
1500   {%
1501     \gdef\glsdefaulttype{#2}%
1502   }{}%

```


Add this to the list of glossary types:

```
1503 \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.

```
1504 \expandafter\gdef\csname glolist@#2\endcsname{,}%
```

Store the file extensions:

```
1505 \expandafter\edef\csname @glo@type@#2@log\endcsname{#1}%
1506 \expandafter\edef\csname @glo@type@#2@in\endcsname{#3}%
1507 \expandafter\edef\csname @glo@type@#2@out\endcsname{#4}%
1508 \expandafter\@gls@forbidtexext\csname @glo@type@#2@log\endcsname
1509 \expandafter\@gls@forbidtexext\csname @glo@type@#2@in\endcsname
1510 \expandafter\@gls@forbidtexext\csname @glo@type@#2@out\endcsname
```

Store the title:

```
1511 \expandafter\def\csname @glo@type@#2@title\endcsname{#5}%
```

```
1512 \@gls@provide@newglossary
```

```
1513 \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.

```
1514 \ifcsundef{gls@#2@entryfmt}%
1515 {%
1516   \defglsentryfmt[#2]{\glsentryfmt}%
1517 }%
1518 {}%
```

Define sort counter if required:

```
1519 \@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.)

```
1520 \ifnextchar[{\@gls@setcounter{#2}}%
1521   {\@gls@setcounter{#2}[\glscounter]}%
1522 }
```

\altnewglossary

```
1523 \newcommand*\altnewglossary}[3]{%
1524   \newglossary[#2-glg]{#1}{#2-gls}{#2-glo}{#3}%
1525 }
```

Only define new glossaries in the preamble:

```
1526 \@onlypreamble{\newglossary}
```

Only define new glossaries before \makeglossaries

```
1527 \@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`

```
1528 \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`

```
1529 \def\@gls@setcounter#1[#2]{%
1530   \expandafter\def\csname @gls@#1@counter\endcsname{#2}%

```

Add counter to xindy list, if not already added:

```
1531   \ifglxindy
1532     \GlsAddXdyCounters{#2}%
1533   \fi
1534 }

```

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

`\@gls@getcounter`

```
1535 \newcommand*{\@gls@getcounter}[1]{%
1536   \csname @gls@#1@counter\endcsname
1537 }

```

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

```
1538 \glsdefmain

```

Define the “acronym” glossaries if required.

```
1539 \@gls@do@acronymsdef

```

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

```
1540 \@gls@do@symbolsdef
1541 \@gls@do@numbersdef
1542 \@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.

```
1543 \newcommand*{\newignoredglossary}[1]{%
1544   \ifdefempty\@ignored@glossaries
1545     {%
1546       \edef\@ignored@glossaries{#1}%
1547     }%
1548     {%
1549       \eappto\@ignored@glossaries{,#1}%

```

```

1550 }%
1551 \csgdef{glolist@#1}{,}%
1552 \ifcsundef{gls@#1@entryfmt}%
1553 {%
1554   \defglentryfmt[#1]{\glentryfmt}%
1555 }%
1556 }%
1557 \ifdefempty\@gls@nohyperlist
1558 {%
1559   \renewcommand*{\@gls@nohyperlist}{#1}%
1560 }%
1561 {%
1562   \eappto\@gls@nohyperlist{, #1}%
1563 }%
1564 }

```

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

```

1565 \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.

```

1566 \newcommand*{\ifignoredglossary}[3]{%
1567   \edef\@gls@igtype{#1}%
1568   \expandafter\DTLifinlist\expandafter
1569     {\@gls@igtype}{\@ignored@glossaries}{#2}{#3}%
1570 }

```

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.

```

1571 \define@key{glossentry}{name}{%
1572   \def\@glo@name{#1}%
1573 }

```

description The description key is usually only used in the glossary, but can be made to appear in the text by redefining `\glentryfmt` or using `\defglentryfmt`. The

description key is required when defining a new glossary entry. If a long description is required, use `\longnewglossaryentry` instead of `\newglossaryentry`.

```
1574 \define@key{glossentry}{description}{%
1575 \def\@glo@desc{#1}%
1576 }
```

descriptionplural

```
1577 \define@key{glossentry}{descriptionplural}{%
1578 \def\@glo@descplural{#1}%
1579 }
```

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>*.

```
1580 \define@key{glossentry}{sort}{%
1581 \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.

```
1582 \define@key{glossentry}{text}{%
1583 \def\@glo@text{#1}%
1584 }
```

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.

```
1585 \define@key{glossentry}{plural}{%
1586 \def\@glo@plural{#1}%
1587 }
```

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.

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

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.

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

`\@gls@default@value`

```
1594 \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).

```
1595 \define@key{glossentry}{symbol}{%
1596 \def\@glo@symbol{#1}%
1597 }
```

symbolplural

```
1598 \define@key{glossentry}{symbolplural}{%
1599 \def\@glo@symbolplural{#1}%
1600 }
```

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

```
1601 \define@key{glossentry}{type}{%
1602 \def\@glo@type{#1}}
```

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

```
1603 \define@key{glossentry}{counter}{%
1604   \ifcsundef{c@#1}%
1605   {%
1606     \PackageError{glossaries}%
1607     {There is no counter called ‘#1’}%
1608     {%
1609       The counter key should have the name of a valid counter
1610       as its value%
1611     }%
1612   }%
1613   {%
1614     \def\@glo@counter{#1}%
1615   }%
1616 }
```

see The see key specifies a list of cross-references

```
1617 \define@key{glossentry}{see}{%
1618   \gls@checkseeallowed
1619   \def\@glo@see{#1}%
1620   \@glo@seeautonumberlist
1621 }
```

\gls@checkseeallowed

```
1622 \newcommand*{\gls@checkseeallowed}{%
1623   \PackageError{glossaries}%
```

```

1624  {'see' key may only be used after \string\makeglossaries\space
1625    or \string\makenoidxglossaries}%
1626  {You must use \string\makeglossaries\space
1627    or \string\makenoidxglossaries\space before defining
1628    any entries that have a 'see' key}%
1629 }

```

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

```

1630 \define@key{glossentry}{parent}{%
1631 \def\@glo@parent{#1}}

```

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

```

1632 \define@choicekey{glossentry}{nonumberlist}[\val\nr]{true,false}[true]{%
1633 \ifcase\nr\relax
1634 \def\@glo@prefix{\glsnonextpages}%
1635 \else
1636 \def\@glo@prefix{\glsnextpages}%
1637 \fi
1638 }

```

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

user1

```

1639 \define@key{glossentry}{user1}{%
1640 \def\@glo@useri{#1}%
1641 }

```

user2

```

1642 \define@key{glossentry}{user2}{%
1643 \def\@glo@userii{#1}%
1644 }

```

user3

```

1645 \define@key{glossentry}{user3}{%
1646 \def\@glo@useriii{#1}%
1647 }

```

user4

```

1648 \define@key{glossentry}{user4}{%
1649 \def\@glo@useriv{#1}%
1650 }

```

user5

```

1651 \define@key{glossentry}{user5}{%
1652 \def\@glo@userv{#1}%
1653 }

```

user6

```
1654 \define@key{glossentry}{user6}{%  
1655   \def\@glo@uservi{#1}%  
1656 }
```

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.

```
1657 \define@key{glossentry}{short}{%  
1658   \def\@glo@short{#1}%  
1659 }
```

shortplural This key is provided for use by \newacronym.

```
1660 \define@key{glossentry}{shortplural}{%  
1661   \def\@glo@shortpl{#1}%  
1662 }
```

long This key is provided for use by \newacronym.

```
1663 \define@key{glossentry}{long}{%  
1664   \def\@glo@long{#1}%  
1665 }
```

longplural This key is provided for use by \newacronym.

```
1666 \define@key{glossentry}{longplural}{%  
1667   \def\@glo@longpl{#1}%  
1668 }
```

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

```
1669 \newcommand*{\@glsnname}{%  
1670   \PackageError{glossaries}{name key required in  
1671   \string\newglossaryentry\space for entry '\@glo@label'}{You  
1672   haven't specified the entry name}}
```

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

```
1673 \newcommand*{\@glsnodelsc}{%  
1674   \PackageError{glossaries}  
1675   {%  
1676     description key required in \string\newglossaryentry\space  
1677     for entry '\@glo@label'%  
1678   }%  
1679   {%  
1680     You haven't specified the entry description%  
1681   }%  
1682 }%
```

\@glsdefaultplural Now obsolete. Don't use.

```
1683 \newcommand*{\@glsdefaultplural}{{}}
```

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

```
1684 \newcommand*{\@gls@missingnumberlist}[1]{%
1685   ??%
1686   \ifglssavenumberlist
1687     \GlossariesWarning{Missing number list for entry ‘#1’.
1688       Maybe makeglossaries + rerun required.}%
1689   \else
1690     \PackageError{glossaries}%
1691       {Package option ‘savenumberlist=true’ required.}%
1692     {%
1693       You must use the ‘savenumberlist’ package option
1694       to reference location lists.%
1695     }%
1696   \fi
1697 }
```

\@glsdefaultsort Define command to set default sort.

```
1698 \newcommand*{\@glsdefaultsort}{\@glo@name}
```

\gls@level Register to increment entry levels.

```
1699 \newcount\gls@level
```

@gls@noexpand@field

```
1700 \newcommand{\@gls@noexpand@field}[3]{%
1701   \expandafter\global\expandafter
1702     \let\csname glo@#1@#2\endcsname#3%
1703 }
```

gls@noexpand@fields

```
1704 \newcommand{\@gls@noexpand@fields}[4]{%
1705   \ifcsdef{gls@assign@#3@field}
1706     {%
1707       \ifdefequal{#4}{\@gls@default@value}%
1708       {%
1709         \edef\@gls@value{\expandonce{#1}}%
1710         \csuse{gls@assign@#3@field}{#2}{\@gls@value}%
1711       }%
1712     }%
1713     \csuse{gls@assign@#3@field}{#2}{#4}%
1714   }%
1715 }%
1716 {%
1717   \ifdefequal{#4}{\@gls@default@value}%
1718   {%
1719     \edef\@gls@value{\expandonce{#1}}%
1720     \@gls@noexpand@field{#2}{#3}{\@gls@value}%
1721   }%
1722   {%
```



```

1723     \@@gls@noexpand@field{#2}{#3}{#4}%
1724   }%
1725 }%
1726 }

```

\@@gls@expand@field

```

1727 \newcommand{\@@gls@expand@field}[3]{%
1728   \expandafter
1729     \protected@xdef\csname glo@#1@#2\endcsname{#3}%
1730 }

```

@gls@expand@fields

```

1731 \newcommand{\@gls@expand@fields}[4]{%
1732   \ifcsdef{gls@assign@#3@field}
1733   {%
1734     \ifdefequal{#4}{\@gls@default@value}%
1735     {%
1736       \edef\@gls@value{\expandonce{#1}}%
1737       \csuse{gls@assign@#3@field}{#2}{\@gls@value}%
1738     }%
1739     {%
1740       \expandafter\@gls@startswitexpandonce#4\relax\relax\gls@endcheck
1741       {%
1742         \@@gls@expand@field{#2}{#3}{#4}%
1743       }%
1744       {%
1745         \csuse{gls@assign@#3@field}{#2}{#4}%
1746       }%
1747     }%
1748   }%
1749   {%
1750     \ifdefequal{#4}{\@gls@default@value}%
1751     {%
1752       \@@gls@expand@field{#2}{#3}{#1}%
1753     }%
1754     {%
1755       \@@gls@expand@field{#2}{#3}{#4}%
1756     }%
1757   }%
1758 }

```

startswitexpandonce

```

1759 \def\@gls@expandonce{\expandonce}
1760 \def\@gls@startswitexpandonce#1#2\gls@endcheck#3#4{%
1761   \def\@gls@tmp{#1}%
1762   \ifdefequal{\@gls@expandonce}{\@gls@tmp}{#3}{#4}%
1763 }

```

<code>\gls@assign@field</code>	<div style="border: 1px solid black; background-color: #ffffcc; padding: 5px; margin-bottom: 10px;"> <code>\gls@assign@field{<def value>}{<glossary type>}{<field>}{<tmp cs>}</code> </div> <p>Assigns an entry field. Expansion performed by default (except for name, symbol and description where backward compatibility required). If <code><tmp cs></code> is <code><@gls@default@value></code>, <code><def value></code> is used instead.</p> <pre>1764 \let\gls@assign@field\@gls@expand@fields</pre>
<code>\glsexpandfields</code>	<p>Fully expand values when assigning fields (except for specific fields that are overridden by <code>\glssetnoexpandfield</code>).</p> <pre>1765 \newcommand*\glsexpandfields{% 1766 \let\gls@assign@field\@gls@expand@fields 1767 }</pre>
<code>\glsnoexpandfields</code>	<p>Don't expand values when assigning fields (except for specific fields that are overridden by <code>\glssetexpandfield</code>).</p> <pre>1768 \newcommand*\glsnoexpandfields{% 1769 \let\gls@assign@field\@gls@noexpand@fields 1770 }</pre>
<code>\newglossaryentry</code>	<p>Define <code>\newglossaryentry {<label>} {<key-val list>}</code>. There are two required fields in <code><key-val list></code>: name (or parent) and description. (See above.)</p> <pre>1771 \newrobustcmd{\newglossaryentry}[2]{% Check to see if this glossary entry has already been defined: 1772 \glsdoifnoexists{#1}% 1773 {% 1774 \gls@defglossaryentry{#1}{#2}% 1775 }% 1776 }</pre>
<code>\docnewglossaryentry</code>	<p>The definition of <code>\newglossaryentry</code> is changed at the start of the document environment.</p> <pre>1777 \newcommand*\gls@defdocnewglossaryentry{% 1778 \let\newglossaryentry\new@glossaryentry 1779 }</pre>
<code>\provideglossaryentry</code>	<p>Like <code>\newglossaryentry</code> but does nothing if the entry has already been defined.</p> <pre>1780 \newrobustcmd{\provideglossaryentry}[2]{% 1781 \ifglstryexists{#1}% 1782 {% 1783 {% 1784 \gls@defglossaryentry{#1}{#2}% 1785 }% 1786 } 1787 \@onlypreamble{\provideglossaryentry}</pre>

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

```
1788 \newrobustcmd{\new@glossaryentry}[2]{%
1789   \ifundef\@gls@deffile
1790   {%
1791     \global\newwrite\@gls@deffile
1792     \immediate\openout\@gls@deffile=\jobname.glsdefs
1793   }%
1794   }%
1795   \ifglentryexists{#1}{}%
1796   {%
1797     \gls@defglossaryentry{#1}{#2}%
1798   }%
1799   \@gls@writedef{#1}%
1800 }
1801 \AtBeginDocument
1802 {
1803   \makeatletter
1804   \InputIfFileExists{\jobname.glsdefs}{\}{}%
1805   \makeatother
1806   \gls@defdocnewglossaryentry
1807 }
1808 \AtEndDocument{\ifdef\@gls@deffile{\closeout\@gls@deffile}{}}
```

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

```
1809 \newcommand*{\@gls@writedef}[1]{%
1810   \immediate\write\@gls@deffile
1811   {%
1812     \string\ifglentryexists{#1}{}\glspercentchar^^J%
1813     \expandafter\@gobble\string\{\glspercentchar^^J%
1814     \string\gls@defglossaryentry{\glsdetoklabel{#1}}\glspercentchar^^J%
1815     \expandafter\@gobble\string\{\glspercentchar%
1816   }%
1817 }
```

Write key value information:

```
1817 \@for\@gls@map:=\@gls@keymap\do
1818 {%
1819   \edef\glo@value{\expandafter\expandonce
1820     \csname glo@\glsdetoklabel{#1}\@expandafter
1821     \@secondoftwo\@gls@map\endcsname}%
1822   \@onelevel@sanitize\glo@value
1823   \immediate\write\@gls@deffile
1824   {%
1825     \expandafter\@firstoftwo\@gls@map
1826     =\expandafter\@gobble\string\{\glo@value\expandafter\@gobble\string\},%
1827     \glspercentchar%
1828   }%
1829 }
```

Provide hook:

```
1830 \glswritedefhook
```

```

1831 \immediate\write\@gls@deffile
1832 {%
1833     \glspercentchar^^J%
1834     \expandafter\@gobble\string\}\glspercentchar^^J%
1835     \expandafter\@gobble\string\}\glspercentchar%
1836 }%
1837 }

```

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

```

1838 \newcommand*{\@gls@keymap}{%
1839 {name}{name},%
1840 {sort}{sortvalue},% unescaped sort value
1841 {type}{type},%
1842 {first}{first},%
1843 {firstplural}{firstpl},%
1844 {text}{text},%
1845 {plural}{plural},%
1846 {description}{desc},%
1847 {descriptionplural}{descplural},%
1848 {symbol}{symbol},%
1849 {symbolplural}{symbolplural},%
1850 {user1}{useri},%
1851 {user2}{userii},%
1852 {user3}{useriii},%
1853 {user4}{useriv},%
1854 {user5}{userv},%
1855 {user6}{uservi},%
1856 {long}{long},%
1857 {longplural}{longpl},%
1858 {short}{short},%
1859 {shortplural}{shortpl},%
1860 {counter}{counter},%
1861 {parent}{parent}}%
1862 }

```

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

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

```
1863 \newcommand*{\@gls@fetchfield}[2]{%
```

Ensure user field name is fully expanded

```
1864 \edef\@gls@thisval{#2}%
```

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

```

1865 \@for\@gls@map:=\@gls@keymap\do{%
1866 \edef\@this@key{\expandafter\@firstoftwo\@gls@map}%
1867 \ifdefequal{\@this@key}{\@gls@thisval}%
1868 {%

```

Found it.

```
1869 \edef#1{\expandafter\@secondoftwo\@gls@map}%
```

Break out of loop.

```
1870 \@endfortrue
1871 }%
1872 {}%
1873 }%
1874 }
```

`\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.

```
1875 \newcommand*{\glsaddstoragekey}{\@ifstar\@sglsaddstoragekey\@glsaddstoragekey}
```

Starred version switches on expansion for this key.

```
1876 \newcommand*{\@sglsaddstoragekey}[1]{%
1877 \key@ifundefined{glossentry}{#1}%
1878 {%
1879 \expandafter\newcommand\expandafter*\expandafter
1880 {\csname gls@assign@#1@field\endcsname}[2]{%
1881 \@@gls@expand@field{##1}{#1}{##2}%
1882 }%
1883 }%
1884 {}%
1885 \@glsaddstoragekey{#1}%
1886 }
```

Unstarred version doesn't override default expansion.

```
1887 \newcommand*{\@glsaddstoragekey}[3]{%
```

Check the specified key doesn't already exist.

```
1888 \key@ifundefined{glossentry}{#1}%
1889 {%
```

Set up the key.

```
1890 \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
1891 \appto\@gls@keymap{, {#1}{#1}}%
```

Set the default value.

```
1892 \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
```

Assignment code.

```
1893 \appto\@newglossaryentryposthook{%
1894 \letcs{\@glo@tmp}{@glo@#1}%
1895 \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
1896 }%
```

Define the no-link commands.

```

1897 \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
1898 }%
1899 {%
1900 \PackageError{glossaries}{Key ‘#1’ already exists}{}%
1901 }%
1902 }

```

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

Allow user to add their own custom keys.

```

1903 \newcommand*\@glsaddkey{\@ifstar\@sglsaddkey\@glsaddkey}

```

Starred version switches on expansion for this key.

```

1904 \newcommand*\@sglsaddkey[1]{%
1905 \key@ifundefined{glossentry}{#1}%
1906 {%
1907 \expandafter\newcommand\expandafter*\expandafter
1908 {\csname gls@assign@#1@field\endcsname}[2]{%
1909 \@gls@expand@field{##1}{#1}{##2}%
1910 }%
1911 }%
1912 }%
1913 \@glsaddkey{#1}%
1914 }

```

Unstarred version doesn't override default expansion.

```

1915 \newcommand*\@glsaddkey[7]{%

```

Check the specified key doesn't already exist.

```

1916 \key@ifundefined{glossentry}{#1}%
1917 {%

```

Set up the key.

```

1918 \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
1919 \appto\@gls@keymap{, {#1}{#1}}%

```

Set the default value.

```

1920 \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%

```

Assignment code.

```

1921 \appto\@newglossaryentryposthook{%
1922 \letcs{\@glo@tmp}{@glo@#1}%
1923 \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
1924 }%

```

Define the no-link commands.

```

1925 \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
1926 \newcommand*{#4}[1]{\@Gls@entry@field{##1}{#1}}%

```

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

```

1927 \ifcsdef{@gls@user@#1@}%
1928 {%
1929     \PackageError{glossaries}%
1930     {Can't define '\string#5' as helper command
1931     '\expandafter\string\csname @gls@user@#1@\endcsname' already exists}%
1932     }%
1933 }%
1934 {%
1935     \expandafter\newcommand\expandafter*\expandafter
1936     {\csname @gls@user@#1\endcsname}[2][ ]{%
1937         \new@ifnextchar[%
1938             {\csuse{@gls@user@#1@}{##1}{##2}}%
1939             {\csuse{@gls@user@#1@}{##1}{##2}[ ]}}%
1940     \csdef{@gls@user@#1@}##1##2[##3]{%
1941         \@gls@field@link{##1}{##2}{#3{##2}##3}%
1942     }%
1943     \newrobustcmd*{#5}{%
1944         \expandafter\@gls@hyp@opt\csname @gls@user@#1\endcsname}%
1945     }%

```

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

```

1946 \ifcsdef{@Gls@user@#1@}%
1947 {%
1948     \PackageError{glossaries}%
1949     {Can't define '\string#6' as helper command
1950     '\expandafter\string\csname @Gls@user@#1@\endcsname' already exists}%
1951     }%
1952 }%
1953 {%
1954     \expandafter\newcommand\expandafter*\expandafter
1955     {\csname @Gls@user@#1\endcsname}[2][ ]{%
1956         \new@ifnextchar[%
1957             {\csuse{@Gls@user@#1@}{##1}{##2}}%
1958             {\csuse{@Gls@user@#1@}{##1}{##2}[ ]}}%
1959     \csdef{@Gls@user@#1@}##1##2[##3]{%
1960         \@gls@field@link{##1}{##2}{#4{##2}##3}%
1961     }%
1962     \newrobustcmd*{#6}{%
1963         \expandafter\@gls@hyp@opt\csname @Gls@user@#1\endcsname}%
1964     }%

```

Finally the all caps version:

```

1965 \ifcsdef{@GLS@user@#1@}%
1966 {%
1967     \PackageError{glossaries}%
1968     {Can't define '\string#7' as helper command
1969     '\expandafter\string\csname @GLS@user@#1@\endcsname' already exists}%

```

```

1970     {}%
1971   }%
1972   {%

1973   \expandafter\newcommand\expandafter*\expandafter
1974     {\csname @GLS@user@#1\endcsname}[2][\{%
1975     \new@ifnextchar[%
1976       {\csuse{@GLS@user@#1@}{##1}{##2}}}%
1977       {\csuse{@GLS@user@#1@}{##1}{##2}[]}}}%
1978   \csdef{@GLS@user@#1@}##1##2[##3]{%
1979     \@gls@field@link{##1}{##2}{\mfirstucMakeUppercase{#3{##2}##3}}}%
1980   }%
1981   \newrobustcmd*{#7}{%
1982     \expandafter\@gls@hyp@opt\csname @GLS@user@#1\endcsname}%
1983   }%
1984   }%
1985   {%
1986   \PackageError{glossaries}{Key ‘#1’ already exists}{}%
1987   }%
1988 }

```

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

```

1989 \newcommand{\glsfieldxdef}[3]{%
1990   \glsdoifexists{#1}%
1991   {%
1992     \edef\@glo@label{\glsdetoklabel{#1}}%
1993     \ifcsdef{glo@\@glo@label @#2}%
1994     {%
1995       \expandafter\xdef\csname glo@\@glo@label @#2\endcsname{#3}%
1996     }%
1997     {%
1998       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
1999     }%
2000   }%
2001 }

```

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

```

2002 \newcommand{\glsfielddedef}[3]{%
2003   \glsdoifexists{#1}%
2004   {%
2005     \edef\@glo@label{\glsdetoklabel{#1}}%
2006     \ifcsdef{glo@\@glo@label @#2}%
2007     {%

```



```

2008     \expandafter\edef\csname glo@\glo@label @#2\endcsname{#3}%
2009 }%
2010 {%
2011     \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2012 }%
2013 }%
2014 }

```

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

```

2015 \newcommand{\glsfieldgdef}[3]{%
2016   \glsdoifexists{#1}%
2017   {%
2018     \edef\glo@label{\glsdetoklabel{#1}}%
2019     \ifcsdef{glo@\glo@label @#2}%
2020     {%
2021       \expandafter\gdef\csname glo@\glo@label @#2\endcsname{#3}%
2022     }%
2023     {%
2024       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2025     }%
2026   }%
2027 }

```

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

```

2028 \newcommand{\glsfielddef}[3]{%
2029   \glsdoifexists{#1}%
2030   {%
2031     \edef\glo@label{\glsdetoklabel{#1}}%
2032     \ifcsdef{glo@\glo@label @#2}%
2033     {%
2034       \expandafter\def\csname glo@\glo@label @#2\endcsname{#3}%
2035     }%
2036     {%
2037       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2038     }%
2039   }%
2040 }

```

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

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

```

2041 \newcommand{\glsfieldfetch}[3]{%
2042   \glsdoifexists{#1}%
2043   {%
2044     \edef\@glo@label{\glsdetoklabel{#1}}%
2045     \ifcsdef{glo@\@glo@label @#2}%
2046     {%
2047       \letcs#3{glo@\@glo@label @#2}%
2048     }%
2049     {%
2050       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2051     }%
2052   }%
2053 }

```

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

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

```

2054 \newcommand{\ifglsfieldeq}[5]{%
2055   \glsdoifexists{#1}%
2056   {%
2057     \edef\@glo@label{\glsdetoklabel{#1}}%
2058     \ifcsdef{glo@\@glo@label @#2}%
2059     {%
2060       \ifcsstring{glo@\@glo@label @#2}{#3}{#4}{#5}%
2061     }%
2062     {%
2063       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2064     }%
2065   }%
2066 }

```

`\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.

```

2067 \newcommand{\ifglsfielddefeq}[5]{%
2068   \glsdoifexists{#1}%
2069   {%
2070     \edef\@glo@label{\glsdetoklabel{#1}}%
2071     \ifcsdef{glo@\@glo@label @#2}%
2072     {%
2073       \expandafter\ifdefstrequal
2074       \csname glo@\@glo@label @#2\endcsname{#3}{#4}{#5}%
2075     }%
2076     {%
2077       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%

```

```

2078 }%
2079 }%
2080 }

```

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

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

```

2081 \newcommand{\ifglsfieldcseq}[5]{%
2082   \glsdoifexists{#1}%
2083   {%
2084     \edef\@glo@label{\glsdetoklabel{#1}}%
2085     \ifcsdef{glo@\@glo@label @#2}%
2086     {%
2087       \ifcsstrequal{glo@\@glo@label @#2}{#3}{#4}{#5}%
2088       }%
2089     {%
2090       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2091     }%
2092   }%
2093 }

```

`\glswritedefhook`

```

2094 \newcommand*{\glswritedefhook}{}

```

`\gls@assign@desc`

```

2095 \newcommand*{\gls@assign@desc}[1]{%
2096   \gls@assign@field{#1}{desc}{\@glo@desc}%
2097   \gls@assign@field{\@glo@desc}{#1}{descplural}{\@glo@descplural}%
2098 }

```

`\longnewglossaryentry`

```

2099 \newcommand{\longnewglossaryentry}[3]{%
2100   \glsdoifnoexists{#1}%
2101   {%
2102     \bgroup
2103     \let\@org@newglossaryentryprehook\@newglossaryentryprehook
2104     \long\def\@newglossaryentryprehook{%
2105       \long\def\@glo@desc{#3\leavevmode\unskip\nopostdesc}%
2106       \@org@newglossaryentryprehook
2107     }%
2108     \renewcommand*{\gls@assign@desc}[1]{%
2109       \global\cslet{glo@\glsdetoklabel{#1}@desc}{\@glo@desc}%
2110       \global\cslet{glo@\glsdetoklabel{#1}@descplural}{\@glo@desc}%
2111     }
2112     \gls@defglossaryentry{#1}{#2}%
2113   \egroup
2114 }
2115 }

```

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

```
2116 \onlypreamble{\longnewglossaryentry}
```

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

```
2117 \newcommand{\longprovideglossaryentry}[3]{%
2118   \ifglentryexists{#1}{}%
2119   {\longnewglossaryentry{#1}{#2}{#3}}%
2120 }
2121 \onlypreamble{\longprovideglossaryentry}
```

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

Defines a new entry without checking if it already exists.

```
2122 \newcommand{\gls@defglossaryentry}[2]{%
```

Store label

```
2123   \edef\@glo@label{\glstoklabel{#1}}%
```

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

```
2124   \let\glslabel\@glo@label
```

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

```
2125   \let\@glo@name\@gls@name
```

```
2126   \let\@glo@desc\@gls@desc
```

```
2127   \let\@glo@descplural\@gls@default@value
```

```
2128   \let\@glo@type\@gls@default@value
```

```
2129   \let\@glo@symbol\@gls@default@value
```

```
2130   \let\@glo@symbolplural\@gls@default@value
```

```
2131   \let\@glo@text\@gls@default@value
```

```
2132   \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.)

```
2133   \let\@glo@first\@gls@default@value
```

```
2134   \let\@glo@firstplural\@gls@default@value
```

Set the default sort:

```
2135   \let\@glo@sort\@gls@default@value
```

Set the default counter:

```
2136   \let\@glo@counter\@gls@default@value
```

```
2137   \def\@glo@see{}
```

2138 \def\@glo@parent{}%

2139 \def\@glo@prefix{}%

2140 \def\@glo@useri{}%

2141 \def\@glo@userii{}%

2142 \def\@glo@useriii{}%

2143 \def\@glo@useriv{}%

2144 \def\@glo@userv{}%

2145 \def\@glo@uservi{}%

2146 \def\@glo@short{}%

2147 \def\@glo@shortpl{}%

2148 \def\@glo@long{}%

2149 \def\@glo@longpl{}%

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

2150 \@newglossaryentryprehook

Extract key-val information from third parameter:

2151 \setkeys{glossentry}{#2}%

Check there is a default glossary.

2152 \ifundef\glsdefaulttype

2153 {%

2154 \PackageError{glossaries}%

2155 {No default glossary type (have you used ‘nomain’?)}%

2156 {If you use package option ‘nomain’ you must define

2157 a new glossary before you can define entries}%

2158 }%

2159 {%

Assign type. This must be fully expandable

2160 \gls@assign@field{\glsdefaulttype}{\@glo@label}{type}{\@glo@type}%

2161 \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.

2162 \ifcsundef{glolist@\@glo@type}%

2163 {%

2164 \PackageError{glossaries}%

2165 {Glossary type ‘\@glo@type’ has not been defined}%

2166 {You need to define a new glossary type, before making entries

2167 in it}%

2168 }%

2169 {%

Check if it's an ignored glossary

2170 \ifignoredglossary\@glo@type

2171 {%

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

```

2172      \ifx\@glo@desc\@glsnodels
2173      \let\@glo@desc\@empty
2174      \fi
2175      }%
2176      {%
2177      }%
2178      \protected@edef\@glolist@{\csname glo@list@\@glo@type\endcsname}%
2179      \expandafter\edef\csname glo@list@\@glo@type\endcsname{%
2180      \@glolist@{\@glo@label},}%
2181      }%

```

Initialise level to 0.

```

2182      \gls@level=0\relax

```

Has this entry been assigned a parent?

```

2183      \ifx\@glo@parent\@empty

```

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

```

2184      \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%
2185      \else

```

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

```

2186      \ifdefequal\@glo@label\@glo@parent%
2187      {%
2188      \PackageError{glossaries}{Entry '@glo@label' can't be its own parent}{}%
2189      \def\@glo@parent{}%
2190      \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%
2191      }%
2192      {%

```

Check the parent exists:

```

2193      \ifglentryexists{\@glo@parent}%
2194      {%

```

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

```

2195      \expandafter\edef\csname glo@\@glo@label @parent\endcsname{%
2196      \@glo@parent}%

```

Determine level.

```

2197      \gls@level=\csname glo@\@glo@parent @level\endcsname\relax
2198      \advance\gls@level by 1\relax

```

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

```

2199      \ifx\@glo@name\@glsnoname
2200      \expandafter\let\expandafter\@glo@name
2201      \csname glo@\@glo@parent @name\endcsname

```

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

```

2202      \ifx\@glo@plural\@gls@default@value
2203      \expandafter\let\expandafter\@glo@plural
2204      \csname glo@\@glo@parent @plural\endcsname
2205      \fi

```

```

2206         \fi
2207     }%
2208     {%

```

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

```

2209         \PackageError{glossaries}%
2210     {%
2211         Invalid parent '\@glo@parent'
2212         for entry '\@glo@label' - parent doesn't exist%
2213     }%
2214     {%
2215         Parent entries must be defined before their children%
2216     }%
2217     \def\@glo@parent{%
2218         \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{%
2219     }%
2220     }%
2221     \fi

```

Set the level for this entry

```

2222     \expandafter\xdef\csname glo@\@glo@label @level\endcsname{\number\gls@level}%

```

Define commands associated with this entry:

```

2223     \gls@assign@field{\@glo@name}{\@glo@label}{sortvalue}{\@glo@sort}%
2224     \letcs\@glo@sort{glo@\@glo@label @sortvalue}%
2225     \gls@assign@field{\@glo@name}{\@glo@label}{text}{\@glo@text}%
2226     \expandafter\gls@assign@field\expandafter
2227         {\csname glo@\@glo@label @text\endcsname\glspluralsuffix}%
2228         {\@glo@label}{plural}{\@glo@plural}%
2229     \expandafter\gls@assign@field\expandafter
2230         {\csname glo@\@glo@label @text\endcsname}%
2231         {\@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.

```

2232     \ifx\@glo@first\@gls@default@value
2233         \expandafter\gls@assign@field\expandafter
2234             {\csname glo@\@glo@label @plural\endcsname}%
2235             {\@glo@label}{firstpl}{\@glo@firstplural}%
2236     \else
2237         \expandafter\gls@assign@field\expandafter
2238             {\csname glo@\@glo@label @first\endcsname\glspluralsuffix}%
2239             {\@glo@label}{firstpl}{\@glo@firstplural}%
2240     \fi

2241     \ifcsundef{@glotype@\@glo@type @counter}%
2242     {%
2243         \def\@glo@defaultcounter{\glscounter}%
2244     }%
2245     {%

```

```

2246 \letcs\@glo@defaultcounter{@glotype@\@glo@type @counter}%
2247 }%
2248 \gls@assign@field{\@glo@defaultcounter}{\@glo@label}{counter}{\@glo@counter}%
2249 \gls@assign@field{}{\@glo@label}{useri}{\@glo@useri}%
2250 \gls@assign@field{}{\@glo@label}{userii}{\@glo@userii}%
2251 \gls@assign@field{}{\@glo@label}{useriii}{\@glo@useriii}%
2252 \gls@assign@field{}{\@glo@label}{useriv}{\@glo@useriv}%
2253 \gls@assign@field{}{\@glo@label}{userv}{\@glo@userv}%
2254 \gls@assign@field{}{\@glo@label}{uservi}{\@glo@uservi}%
2255 \gls@assign@field{}{\@glo@label}{short}{\@glo@short}%
2256 \gls@assign@field{}{\@glo@label}{shortpl}{\@glo@shortpl}%
2257 \gls@assign@field{}{\@glo@label}{long}{\@glo@long}%
2258 \gls@assign@field{}{\@glo@label}{longpl}{\@glo@longpl}%
2259 \ifx\@glo@name\@gls@name
2260 \@gls@name
2261 \let\@glo@name\@gls@default@value
2262 \fi
2263 \gls@assign@field{}{\@glo@label}{name}{\@glo@name}%

```

Set default numberlist if not defined:

```

2264 \ifcsundef{glo@\@glo@label @numberlist}%
2265 {%
2266 \csxdef{glo@\@glo@label @numberlist}{%
2267 \noexpand\@gls@missingnumberlist{\@glo@label}}%
2268 }%
2269 {}%

```

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.

```

2270 \def\@glo@@desc{\@glo@first}%
2271 \ifx\@glo@desc\@glo@@desc
2272 \let\@glo@desc\@glo@first
2273 \fi
2274 \ifx\@glo@desc\@gls@nodesc
2275 \@gls@nodesc
2276 \let\@glo@desc\@gls@default@value
2277 \fi
2278 \gls@assign@desc{\@glo@label}%

```

Set the sort key for this entry:

```

2279 \@gls@defsort{\@glo@type}{\@glo@label}%
2280 \def\@glo@@symbol{\@glo@text}%
2281 \ifx\@glo@symbol\@glo@@symbol
2282 \let\@glo@symbol\@glo@text
2283 \fi
2284 \gls@assign@field{\relax}{\@glo@label}{symbol}{\@glo@symbol}%
2285 \expandafter
2286 \gls@assign@field\expandafter

```



```

2287     {\csname glo@\@glo@label @symbol\endcsname}
2288     {\@glo@label}{symbolplural}{\@glo@symbolplural}}%

```

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

```

2289     \expandafter\xdef\csname glo@\@glo@label @flagfalse\endcsname{%
2290     \noexpand\global
2291     \noexpand\let\expandafter\noexpand
2292     \csname ifglo@\@glo@label @flag\endcsname\noexpand\iffalse
2293     }%
2294     \expandafter\xdef\csname glo@\@glo@label @flagtrue\endcsname{%
2295     \noexpand\global
2296     \noexpand\let\expandafter\noexpand
2297     \csname ifglo@\@glo@label @flag\endcsname\noexpand\iftrue
2298     }%
2299     \csname glo@\@glo@label @flagfalse\endcsname

```

Sort out any cross-referencing if required.

```

2300     \ifdefined\@glo@see
2301     {}%
2302     {%
2303     \protected@edef\@do@glsee{%
2304     \noexpand\@gl@fixbraces\noexpand\@glo@list\@glo@see
2305     \noexpand\@nil
2306     \noexpand\expandafter\noexpand\@glsee\noexpand\@glo@list{\@glo@label}}%
2307     \@do@glsee
2308     }%

```

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

```

2309     \ifignoredglossary\@glo@type
2310     {%
2311     \csdef{glo@\@glo@label @index}{}%
2312     }
2313     {%
2314     \do@glo@storeentry{\@glo@label}%
2315     }%

```

Define entry counters if enabled:

```

2316     \@newglossaryentry@defcounters

```

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

```

2317     \@newglossaryentryposthook
2318 }

```

`\glossaryentryprehook` Allow extra information to be added to glossary entries:

```

2319 \newcommand*{\@newglossaryentryprehook}{}

```

`\glossaryentryposthook` Allow extra information to be added to glossary entries:

```

2320 \newcommand*{\@newglossaryentryposthook}{}

```

ryentry@defcounters

```
2321 \newcommand*{\@newglossaryentry@defcounters}{}
```

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

```
2322 \newcommand*{\glsmoveentry}[2]{%
2323   \edef\@glo@thislabel{\glsdetoklabel{#1}}%
2324   \edef\glo@type{\csname glo@\@glo@thislabel @type\endcsname}%
2325   \def\glo@list{,}%
2326   \forglsentries[\glo@type]{\glo@label}%
2327   {%
2328     \ifdefequal\@glo@thislabel\glo@label
2329       {}\eappto\glo@list{\glo@label,}%
2330     }%
2331     \cslet\glo@list@\glo@type{\glo@list}%
2332     \csdef\glo@\@glo@thislabel @type{#2}%
2333 }
```

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

```
2334 \ifglxindy
2335   \newcommand*{\@glossaryentryfield}{\string\glossentry}
2336 \else
2337   \newcommand*{\@glossaryentryfield}{\string\glossentry}
2338 \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.)

```
2339 \ifglxindy
2340   \newcommand*{\@glossarysubentryfield}{%
2341     \string\subglossentry}
2342 \else
2343   \newcommand*{\@glossarysubentryfield}{%
2344     \string\subglossentry}
2345 \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.)

```
2346 \newcommand{\@glo@storeentry}[1]{%
```

Escape makeindex/xindy special characters in the label:

```
2347 \edef\@glo@esclabel{#1}%  
2348 \@gls@checkmkidxchars\@glo@esclabel
```

Get the sort string and escape any special characters

```
2349 \protected@edef\@glo@sort{\csname glo@#1@sort\endcsname}%  
2350 \@gls@checkmkidxchars\@glo@sort
```

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

```
2351 \@gls@checkmkidxchars\@glo@prefix
```

Get the parent, if one exists

```
2352 \edef\@glo@parent{\csname glo@#1@parent\endcsname}%
```

Write the information to the glossary file.

```
2353 \ifglxindy
```

Store using xindy syntax.

```
2354 \ifx\@glo@parent\@empty
```

Entry doesn't have a parent

```
2355 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%  
2356 (\string"\@glo@sort\string" %  
2357 \string"\@glo@prefix\@glossaryentryfield{\@glo@esclabel}\string") %  
2358 }%  
2359 \else
```

Entry has a parent

```
2360 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%  
2361 \csname glo@\@glo@parent @index\endcsname  
2362 (\string"\@glo@sort\string" %  
2363 \string"\@glo@prefix\@glossarysubentryfield  
2364 {\csname glo@#1@level\endcsname}{\@glo@esclabel}\string") %  
2365 }%  
2366 \fi  
2367 \else
```

Store using makeindex syntax.

```
2368 \ifx\@glo@parent\@empty
```

Sanitize \@glo@prefix

```
2369 \@onelevel@sanitize\@glo@prefix
```

Entry doesn't have a parent

```
2370 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%  
2371 \@glo@sort\@gls@actualchar\@glo@prefix  
2372 \@glossaryentryfield{\@glo@esclabel}%  
2373 }%  
2374 \else
```

Entry has a parent

```
2375 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%  
2376 \csname glo@\@glo@parent @index\endcsname\@gls@levelchar
```

```

2377      \@glo@sort\@gls@actualchar\@glo@prefix
2378      \@glossarysubentryfield
2379      {\csname glo@#1@level\endcsname}{\@glo@esclabel}%
2380    }%
2381    \fi
2382    \fi
2383  }

```

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`

```

2384 \AtBeginDocument{%
2385   \@ifpackageloaded{amsmath}%
2386   {\let\gls@ifnotmeasuring\@gls@ifnotmeasuring}%
2387   }%
2388 }
2389 \newcommand*{\@gls@ifnotmeasuring}[1]{%
2390   \ifmeasuring@
2391   \else
2392     #1%
2393   \fi
2394 }
2395 \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.

```

2396 \newcommand*{\glsreset}[1]{%
2397   \gls@ifnotmeasuring
2398   {%
2399     \glsdoifexists{#1}%
2400     {%
2401       \@glsreset{#1}%
2402     }%
2403   }%
2404 }

```

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

```

2405 \newcommand*{\glslocalreset}[1]{%
2406   \gls@ifnotmeasuring
2407   {%
2408     \glsdoifexists{#1}%
2409     {%

```

```

2410      \@glslocalreset{#1}%
2411    }%
2412  }%
2413 }

```

`\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.

```

2414 \newcommand*{\glsunset}[1]{%
2415   \gls@ifnotmeasuring
2416   {%
2417     \glsdoifexists{#1}%
2418     {%
2419       \@glsunset{#1}%
2420     }%
2421   }%
2422 }

```

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

```

2423 \newcommand*{\glslocalunset}[1]{%
2424   \gls@ifnotmeasuring
2425   {%
2426     \glsdoifexists{#1}%
2427     {%
2428       \@glslocalunset{#1}%
2429     }%
2430   }%
2431 }

```

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

```

2432 \newcommand*{\@glslocalunset}{\@glslocalunset}

```

`\@@glslocalunset` Local unset without checks.

```

2433 \newcommand*{\@@glslocalunset}[1]{%
2434   \expandafter\let\csname ifglo@glsdetoklabel{#1}@flag\endcsname\iftrue
2435 }

```

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

```

2436 \newcommand*{\@glsunset}{\@glsunset}

```

`\@@glsunset` Global unset without checks.

```

2437 \newcommand*{\@@glsunset}[1]{%
2438   \expandafter\global\csname glo@glsdetoklabel{#1}@flagtrue\endcsname
2439 }

```

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

```

2440 \newcommand*{\@glslocalreset}{\@glslocalreset}

```

`\@@glslocalreset` Local reset without checks.

```
2441 \newcommand*{\@@glslocalreset}[1]{%
2442   \expandafter\let\csname ifglo@glsdetoklabel{#1}@flag\endcsname\iffalse
2443 }
```

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

```
2444 \newcommand*{\glsreset}{\@@glsreset}
```

`\@@glsreset` Global reset without checks.

```
2445 \newcommand*{\@@glsreset}[1]{%
2446   \expandafter\global\csname glo@glsdetoklabel{#1}@flagfalse\endcsname
2447 }
```

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

`\glsresetall`

```
2448 \newcommand*{\glsresetall}[1][\@glo@types]{%
2449   \forallglsentries[#1]{\@glsentry}%
2450   {%
2451     \glsreset{\@glsentry}%
2452   }%
2453 }
```

As above, but with only a local effect:

`\glslocalresetall`

```
2454 \newcommand*{\glslocalresetall}[1][\@glo@types]{%
2455   \forallglsentries[#1]{\@glsentry}%
2456   {%
2457     \glslocalreset{\@glsentry}%
2458   }%
2459 }
```

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

`\glsunsetall`

```
2460 \newcommand*{\glsunsetall}[1][\@glo@types]{%
2461   \forallglsentries[#1]{\@glsentry}%
2462   {%
2463     \glsunset{\@glsentry}%
2464   }%
2465 }
```

As above, but with only a local effect:

`\glslocalunsetall`

```
2466 \newcommand*{\glslocalunsetall}[1][\@glo@types]{%
2467   \forallglsentries[#1]{\@glsentry}%
2468   {%
2469     \glslocalunset{\@glsentry}%
2470   }%
2471 }
```

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

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

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

```
2472 \newcommand*{\@newglossaryentry@defcounters}{%
2473   \csdef{glo@\@glo@label @currcount}{0}%
2474   \csdef{glo@\@glo@label @prevcount}{0}%
2475 }
```

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

```
2476 \newcommand*{\glsenableentrycount}{%
```

Enable new entry fields.

```
2477   \let\@newglossaryentry@defcounters\@newglossaryentry@defcounters
```

Disable `\newglossaryentry` in the document environment.

```
2478   \renewcommand*{\gls@defdocnewglossaryentry}{%
2479     \renewcommand*{\newglossaryentry}[2]{%
2480       \PackageError{glossaries}{\string\newglossaryentry\space
2481         may only be used in the preamble when entry counting has
2482         been activated}{If you use \string\glsenableentrycount\space
2483         you must place all entry definitions in the preamble not in
2484         the document environment}%
2485     }%
2486   }%
```

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

```
2487   \newcommand*{\glsentrycurrcount}[1]{%
2488     \ifcsundef{glo@\glsdetoklabel{##1}@currcount}%
2489     {0}{\@gls@entry@field{##1}{currcount}}%
2490   }%
```

```

2491 \newcommand*\glsentryprevcount}[1]{%
2492   \ifcsundef{glo@glsdetoklabel{##1}@prevcount}%
2493     {0}{\@gls@entry@field{##1}{prevcount}}%
2494 }%

```

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

```

2495 \renewcommand*\@glsunset}[1]{%
2496   \@glsunset{##1}%
2497   \@gls@increment@currcount{##1}%
2498 }%
2499 \renewcommand*\@glslocalunset}[1]{%
2500   \@glslocalunset{##1}%
2501   \@gls@local@increment@currcount{##1}%
2502 }%
2503 \renewcommand*\@glsreset}[1]{%
2504   \@glsreset{##1}%
2505   \csgdef{glo@glsdetoklabel{##1}@currcount}{0}%
2506 }%
2507 \renewcommand*\@glslocalreset}[1]{%
2508   \@glslocalreset{##1}%
2509   \csdef{glo@glsdetoklabel{##1}@currcount}{0}%
2510 }%

```

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.)

```

2511 \def\@cgl's@##1##2[##3]{%
2512   \ifnum\glsentryprevcount{##2}=1\relax
2513     \cgl'sformat{##2}{##3}%
2514     \glsunset{##2}%
2515   \else
2516     \@gls@{##1}{##2}[##3]%
2517   \fi
2518 }%

```

Similarly for the analogous commands. No case change plural:

```

2519 \def\@cgl'spl@##1##2[##3]{%
2520   \ifnum\glsentryprevcount{##2}=1\relax
2521     \cgl'splformat{##2}{##3}%
2522     \glsunset{##2}%
2523   \else
2524     \@glspl@{##1}{##2}[##3]%
2525   \fi
2526 }%

```

First letter uppercase singular:

```

2527 \def\@cGls@##1##2[##3]{%
2528   \ifnum\glsentryprevcount{##2}=1\relax
2529     \cGlsformat{##2}{##3}%
2530     \glsunset{##2}%
2531   \else

```



```

2532 \Gls@{##1}-{##2}[##3]%
2533 \fi
2534 }%

```

First letter uppercase plural:

```

2535 \def\cGlspl@##1##2[##3]{%
2536 \ifnum\glentryprevcount{##2}=1\relax
2537 \cGlsplformat{##2}{##3}%
2538 \glunset{##2}%
2539 \else
2540 \Glspl@{##1}-{##2}[##3]%
2541 \fi
2542 }%

```

Write information to aux file at the end of the document

```

2543 \AtEndDocument{\@gls@write@entrycounts}%

```

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

```

2544 \renewcommand*{\@gls@entry@count}[2]{%
2545 \csgdef{glo@\glsetoklabel{##1}@prevcount}{##2}%
2546 }%

```

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

```

2547 \let\glsenableentrycount\relax
2548 }
2549 \@onlypreamble\glsenableentrycount

```

increment@currcount

```

2550 \newcommand*{\@gls@increment@currcount}[1]{%
2551 \csxdef{glo@\glsetoklabel{#1}@currcount}{%
2552 \number\numexpr\glentrycurrcount{#1}+1}%
2553 }

```

increment@currcount

```

2554 \newcommand*{\@gls@local@increment@currcount}[1]{%
2555 \csedef{glo@\glsetoklabel{#1}@currcount}{%
2556 \number\numexpr\glentrycurrcount{#1}+1}%
2557 }

```

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.)

```

2558 \newcommand*{\@gls@write@entrycounts}{%
2559 \immediate\write\@auxout
2560 {\string\providecommand*\string\@gls@entry@count}[2]{}%
2561 \forallglentries{\@glentry}{%
2562 \ifglused{\@glentry}%

```

```

2563     {\immediate\write\@auxout
2564       {\string\@gls@entry@count{\@glsentry}{\glsentrycurrcount{\@glsentry}}}%
2565     }%
2566   }%
2567 }

```

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

```

2568 \newcommand*{\@gls@entry@count}[2]{ }

```

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

```

2569 \newrobustcmd*{\cgl}{\@gls@hyp@opt\@cgl}

```

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

```

2570 \newcommand*{\@cgl}[2][ ]{%
2571   \new@ifnextchar[{\@cgl@{#1}{#2}}{\@cgl@{#1}{#2}[ ]}%
2572 }

```

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

```

2573 \def\@cgl@#1#2[#3]{%
2574   \GlossariesWarning{\string\cgl\space is defaulting to
2575     \string\gls\space since you haven't enabled entry counting}%
2576   \@gls@{#1}{#2}[#3]%
2577 }

```

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

```

2578 \newcommand*{\cglformat}[2]{%
2579   \ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}#2%
2580 }

```

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

```

2581 \newrobustcmd*{\cGl}{\@gls@hyp@opt\@cGl}

```

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

```

2582 \newcommand*{\@cGl}[2][ ]{%
2583   \new@ifnextchar[{\@cGl@{#1}{#2}}{\@cGl@{#1}{#2}[ ]}%
2584 }

```

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

```

2585 \def\cGls@#1#2[#3]{%
2586 \GlossariesWarning{\string\cGls\space is defaulting to
2587 \string\Gls\space since you haven't enabled entry counting}%
2588 \cGls@{#1}{#2}[#3]%
2589 }

```

`\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.

```

2590 \newcommand*\cGlsformat}[2]{%
2591 \ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}#2%
2592 }

```

`\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.)

```

2593 \newrobustcmd*\cglsp1{\cGls@hyp@opt\cglsp1}

```

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

```

2594 \newcommand*\@cglsp1}[2][{}]{%
2595 \new@ifnextchar[\cglsp1@{#1}{#2}]{\cglsp1@{#1}{#2}[]}%
2596 }

```

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

```

2597 \def\@cglsp1@#1#2[#3]{%
2598 \GlossariesWarning{\string\cglsp1\space is defaulting to
2599 \string\glsp1\space since you haven't enabled entry counting}%
2600 \cglsp1@{#1}{#2}[#3]%
2601 }

```

`\cglsp1format` Format used by `\cglsp1` if entry only used once on previous run. The first argument is the label, the second argument is the insert text.

```

2602 \newcommand*\cglsp1format}[2]{%
2603 \ifglshaslong{#1}{\glsp1entrylong{#1}}{\glsp1entryfirstplural{#1}}#2%
2604 }

```

`\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.)

```

2605 \newrobustcmd*\cGlspl{\cGls@hyp@opt\cGlspl}

```

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

```

2606 \newcommand*\@cGlspl}[2][{}]{%
2607 \new@ifnextchar[\cGlspl@{#1}{#2}]{\cGlspl@{#1}{#2}[]}%
2608 }

```

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

```
2609 \def\cGlspl@#1#2[#3]{%
2610 \GlossariesWarning{\string\cGlspl\space is defaulting to
2611 \string\Glspl\space since you haven't enabled entry counting}%
2612 \cGlspl@{#1}{#2}[#3]%
2613 }
```

`\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.

```
2614 \newcommand*\cGlsplformat}[2]{%
2615 \ifglshaslong{#1}{\Glsentrylongpl{#1}}{\Glsentryfirstplural{#1}}#2%
2616 }
```

1.10 Loading files containing glossary entries

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

`\loadglsentries` [*type*] {*filename*}

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

`\loadglsentries`

```
2617 \newcommand*\loadglsentries}[2][\@gls@default]{%
2618 \let\@gls@default\glsdefaulttype
2619 \def\glsdefaulttype{#1}\input{#2}%
2620 \let\glsdefaulttype\@gls@default
2621 }
```

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

```
2622 \@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

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

commands (such as `\textbf` is governed by `\glstextformat`. By default this just displays the link text “as is”.

`\glstextformat`

```
2623 \newcommand*{\glstextformat}[1]{#1}
```

`\glstentryfmt` As from version 3.11a, the way in which an entry is displayed is now governed by `\glstentryfmt`. 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

```
2624 \newcommand*{\glstentryfmt}{%
2625   \@@gls@default@entryfmt\glsdisplayfirst\glsdisplay
2626 }
```

Format that provides backwards compatibility:

```
2627 \newcommand*{\@@gls@default@entryfmt}[2]{%
2628   \ifdefempty\glscustomtext
2629     {%
2630       \glsifplural
2631       {%
```

Plural form

```
2632       \glscapscase
2633       {%
```

Don't adjust case

```
2634       \ifglsused\glslabel
2635       {%
```

Subsequent use

```
2636       #2{\glstentryplural{\glslabel}}%
2637       {\glstentrydescplural{\glslabel}}%
2638       {\glstentrysymbolplural{\glslabel}}{\glsinsert}%
2639     }%
2640     {%
```

First use

```
2641       #1{\glstentryfirstplural{\glslabel}}%
2642       {\glstentrydescplural{\glslabel}}%
2643       {\glstentrysymbolplural{\glslabel}}{\glsinsert}%
2644     }%
2645   }%
2646   {%
```

Make first letter upper case

```
2647       \ifglsused\glslabel
2648       {%
```

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.)

```

2649     \ifbool{glscompatible-3.07}%
2650     {%
2651         \protected@edef\@glo@etext{%
2652             #2{\glsentryplural{\glslabel}}%
2653             {\glsentrydescplural{\glslabel}}%
2654             {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2655         \xmakefirstuc\@glo@etext
2656     }%
2657     {%
2658         #2{\Glsentryplural{\glslabel}}%
2659         {\glsentrydescplural{\glslabel}}%
2660         {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2661     }%
2662 }%
2663 {%

```

First use

```

2664     \ifbool{glscompatible-3.07}%
2665     {%
2666         \protected@edef\@glo@etext{%
2667             #1{\glsentryfirstplural{\glslabel}}%
2668             {\glsentrydescplural{\glslabel}}%
2669             {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2670         \xmakefirstuc\@glo@etext
2671     }%
2672     {%
2673         #1{\Glsentryfirstplural{\glslabel}}%
2674         {\glsentrydescplural{\glslabel}}%
2675         {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2676     }%
2677 }%
2678 }%
2679 {%

```

Make all upper case

```

2680     \ifglsused\glslabel
2681     {%

```

Subsequent use

```

2682         \mfirstucMakeUppercase{#2{\glsentryplural{\glslabel}}%
2683         {\glsentrydescplural{\glslabel}}%
2684         {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2685     }%
2686     {%

```

First use

```

2687         \mfirstucMakeUppercase{#1{\glsentryfirstplural{\glslabel}}%
2688         {\glsentrydescplural{\glslabel}}%
2689         {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%

```

```

2690      }%
2691      }%
2692      }%
2693      {%

```

Singular form

```

2694      \glscapscase
2695      {%

```

Don't adjust case

```

2696      \ifglused\glslabel
2697      {%

```

Subsequent use

```

2698      #2{\glentrytext{\glslabel}}%
2699      {\glentrydesc{\glslabel}}%
2700      {\glentrysymbol{\glslabel}}{\glinsert}%
2701      }%
2702      {%

```

First use

```

2703      #1{\glentryfirst{\glslabel}}%
2704      {\glentrydesc{\glslabel}}%
2705      {\glentrysymbol{\glslabel}}{\glinsert}%
2706      }%
2707      }%
2708      {%

```

Make first letter upper case

```

2709      \ifglused\glslabel
2710      {%

```

Subsequent use

```

2711      \ifbool{glcompatible-3.07}%
2712      {%
2713          \protected@edef\@glo@etext{%
2714              #2{\glentrytext{\glslabel}}%
2715              {\glentrydesc{\glslabel}}%
2716              {\glentrysymbol{\glslabel}}{\glinsert}}%
2717          \xmakefirstuc\@glo@etext
2718      }%
2719      {%
2720      #2{\Glsentrytext{\glslabel}}%
2721      {\glentrydesc{\glslabel}}%
2722      {\glentrysymbol{\glslabel}}{\glinsert}%
2723      }%
2724      }%
2725      {%

```

First use

```

2726      \ifbool{glcompatible-3.07}%
2727      {%

```

```

2728         \protected@edef\@glo@etext{%
2729             #1{\glsentryfirst{\glslabel}}}%
2730             {\glsentrydesc{\glslabel}}}%
2731             {\glsentrysymbol{\glslabel}}{\glsinsert}}}%
2732         \xmakefirstuc\@glo@etext
2733     }%
2734     {%
2735         #1{\Glsentryfirst{\glslabel}}}%
2736         {\glsentrydesc{\glslabel}}}%
2737         {\glsentrysymbol{\glslabel}}{\glsinsert}}}%
2738     }%
2739 }%
2740 }%
2741 {%

```

Make all upper case

```

2742     \ifglsused\glslabel
2743     {%

```

Subsequent use

```

2744         \mfirstucMakeUppercase{#2{\glsentrytext{\glslabel}}}%
2745         {\glsentrydesc{\glslabel}}}%
2746         {\glsentrysymbol{\glslabel}}{\glsinsert}}}%
2747     }%
2748     {%

```

First use

```

2749         \mfirstucMakeUppercase{#1{\glsentryfirst{\glslabel}}}%
2750         {\glsentrydesc{\glslabel}}}%
2751         {\glsentrysymbol{\glslabel}}{\glsinsert}}}%
2752     }%
2753 }%
2754 }%
2755 }%
2756 {%

```

Custom text provided in \glsdisp

```

2757     \ifglsused{\glslabel}%
2758     {%

```

Subsequent use

```

2759         #2{\glscustomtext}%
2760         {\glsentrydesc{\glslabel}}}%
2761         {\glsentrysymbol{\glslabel}}{\}%
2762     }%
2763     {%

```

First use

```

2764         #1{\glscustomtext}%
2765         {\glsentrydesc{\glslabel}}}%
2766         {\glsentrysymbol{\glslabel}}{\}%
2767     }%

```



```

2768 }%
2769 }

```

`\glsgenentryfmt` Define a generic format that just uses the first, text, plural or first plural keys (or the custom text) with the insert text appended.

```

2770 \newcommand*{\glsgenentryfmt}{%
2771   \ifdefempty\glscustomtext
2772   {%
2773     \glusifplural
2774     {%

```

Plural form

```

2775     \glscapscase
2776     {%

```

Don't adjust case

```

2777     \ifglused\glslabel
2778     {%

```

Subsequent use

```

2779     \gl Sentryplural{\glslabel}\glinsert
2780     }%
2781     {%

```

First use

```

2782     \gl Sentryfirstplural{\glslabel}\glinsert
2783     }%
2784     }%
2785     {%

```

Make first letter upper case

```

2786     \ifglused\glslabel
2787     {%

```

Subsequent use.

```

2788     \Glsentryplural{\glslabel}\glinsert
2789     }%
2790     {%

```

First use

```

2791     \Glsentryfirstplural{\glslabel}\glinsert
2792     }%
2793     }%
2794     {%

```

Make all upper case

```

2795     \ifglused\glslabel
2796     {%

```

Subsequent use

```

2797     \mfirstucMakeUppercase
2798     {\gl Sentryplural{\glslabel}\glinsert}%
2799     }%
2800     {%

```

First use

```
2801      \mfirstucMakeUppercase
2802      {\glsentryfirstplural{\glslabel}\glsinsert}%
2803      }%
2804      }%
2805      }%
2806      {%
```

Singular form

```
2807      \glscapscase
2808      {%
```

Don't adjust case

```
2809      \ifglsused\glslabel
2810      {%
```

Subsequent use

```
2811      \glsentrytext{\glslabel}\glsinsert
2812      }%
2813      {%
```

First use

```
2814      \glsentryfirst{\glslabel}\glsinsert
2815      }%
2816      }%
2817      {%
```

Make first letter upper case

```
2818      \ifglsused\glslabel
2819      {%
```

Subsequent use

```
2820      \Glsentrytext{\glslabel}\glsinsert
2821      }%
2822      {%
```

First use

```
2823      \Glsentryfirst{\glslabel}\glsinsert
2824      }%
2825      }%
2826      {%
```

Make all upper case

```
2827      \ifglsused\glslabel
2828      {%
```

Subsequent use

```
2829      \mfirstucMakeUppercase{\glsentrytext{\glslabel}\glsinsert}%
2830      }%
2831      {%
```

First use

```
2832      \mfirstucMakeUppercase{\glsentryfirst{\glslabel}\glsinsert}%
```

```

2833     }%
2834     }%
2835     }%
2836     }%
2837     {%

```

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

```

2838     \glscustomtext\glsinsert
2839     }%
2840 }

```

`\glsngenacfmt` Define a generic acronym format that uses the long and short keys (or their plurals) and `\acrfullformat`, `\firstacronymfont` and `\acronymfont`.

```

2841 \newcommand*{\glsngenacfmt}{%
2842   \ifdefempty\glscustomtext
2843     {%
2844       \ifglsused\glslabel
2845         {%

```

Subsequent use:

```

2846         \glsifplural
2847         {%

```

Subsequent plural form:

```

2848         \glscapscase
2849         {%

```

Subsequent plural form, don't adjust case:

```

2850         \acronymfont{\glsentryshortpl{\glslabel}}\glsinsert
2851         }%
2852         {%

```

Subsequent plural form, make first letter upper case:

```

2853         \acronymfont{\Glsentryshortpl{\glslabel}}\glsinsert
2854         }%
2855         {%

```

Subsequent plural form, all caps:

```

2856         \mfirstucMakeUppercase
2857         {\acronymfont{\glsentryshortpl{\glslabel}}\glsinsert}%
2858         }%
2859         }%
2860         {%

```

Subsequent singular form

```

2861         \glscapscase
2862         {%

```

Subsequent singular form, don't adjust case:

```

2863         \acronymfont{\glsentryshort{\glslabel}}\glsinsert
2864         }%
2865         {%

```

Subsequent singular form, make first letter upper case:

```
2866      \acronymfont{\Glsentryshort{\glslabel}}\glsinsert
2867      }%
2868      {%
```

Subsequent singular form, all caps:

```
2869      \mfirstucMakeUppercase
2870      {\acronymfont{\Glsentryshort{\glslabel}}\glsinsert}%
2871      }%
2872      }%
2873      }%
2874      {%
```

First use:

```
2875      \glsifplural
2876      {%
```

First use plural form:

```
2877      \glscapscase
2878      {%
```

First use plural form, don't adjust case:

```
2879      \genplacrfullformat{\glslabel}{\glsinsert}%
2880      }%
2881      {%
```

First use plural form, make first letter upper case:

```
2882      \Genplacrfullformat{\glslabel}{\glsinsert}%
2883      }%
2884      {%
```

First use plural form, all caps:

```
2885      \mfirstucMakeUppercase
2886      {\genplacrfullformat{\glslabel}{\glsinsert}}%
2887      }%
2888      }%
2889      {%
```

First use singular form

```
2890      \glscapscase
2891      {%
```

First use singular form, don't adjust case:

```
2892      \genacrfullformat{\glslabel}{\glsinsert}%
2893      }%
2894      {%
```

First use singular form, make first letter upper case:

```
2895      \Genacrfullformat{\glslabel}{\glsinsert}%
2896      }%
2897      {%
```

First use singular form, all caps:

```

2898      \mfirstucMakeUppercase
2899      {\genacrfullformat{\glslabel}{\glsinsert}}%
2900      }%
2901      }%
2902      }%
2903      }%
2904      {%

```

User supplied text.

```

2905      \glscustomtext
2906      }%
2907 }

```

`\genacrfullformat` `\genacrfullformat{<label>}{<insert>}`

The full format used by `\glsngenacfmt` (singular).

```

2908 \newcommand*{\genacrfullformat}[2]{%
2909   \glsentrylong{#1}#2\space
2910   (\protect\firstacronymfont{\glsentryshort{#1}})%
2911 }

```

`\Genacrfullformat` `\Genacrfullformat{<label>}{<insert>}`

As above but makes the first letter upper case.

```

2912 \newcommand*{\Genacrfullformat}[2]{%
2913   \protected@edef\gls@text{\genacrfullformat{#1}{#2}}%
2914   \xmakefirstuc\gls@text
2915 }

```

`\genplacrfullformat` `\genplacrfullformat{<label>}{<insert>}`

The full format used by `\glsngenacfmt` (plural).

```

2916 \newcommand*{\genplacrfullformat}[2]{%
2917   \glsentrylongpl{#1}#2\space
2918   (\protect\firstacronymfont{\glsentryshortpl{#1}})%
2919 }

```

`\Genplacrfullformat` `\Genplacrfullformat{<label>}{<insert>}`

As above but makes the first letter upper case.

```

2920 \newcommand*{\Genplacrfullformat}[2]{%
2921   \protected@edef\gls@text{\genplacrfullformat{#1}{#2}}%

```

```

2922 \xmakefirstuc\gls@text
2923 }

```

`\glsdisplayfirst` Deprecated. Kept for backward compatibility.

```

2924 \newcommand*{\glsdisplayfirst}[4]{#1#4}

```

`\glsdisplay` Deprecated. Kept for backward compatibility.

```

2925 \newcommand*{\glsdisplay}[4]{#1#4}

```

`\defglsdisplay` Deprecated. Kept for backward compatibility.

```

2926 \newcommand*{\defglsdisplay}[2][\glsdefaulttype]{%
2927   \GlossariesWarning{\string\defglsdisplay\space is now obsolete.^^J
2928   Use \string\defglsentryfmt\space instead}%
2929   \expandafter\def\csname gls@#1@display\endcsname##1##2##3##4{#2}%
2930   \edef\@gls@doentrydef{%
2931     \noexpand\defglsentryfmt[#1]{%
2932       \noexpand\ifcsdef{gls@#1@displayfirst}%
2933       {%
2934         \noexpand\@gls@default@entryfmt
2935         {\noexpand\csuse{gls@#1@displayfirst}}}%
2936         {\noexpand\csuse{gls@#1@display}}}%
2937       }%
2938       {%
2939         \noexpand\@gls@default@entryfmt
2940         {\noexpand\glsdisplayfirst}%
2941         {\noexpand\csuse{gls@#1@display}}}%
2942       }%
2943     }%
2944   }%
2945   \@gls@doentrydef
2946 }

```

`\defglsdisplayfirst` Deprecated. Kept for backward compatibility.

```

2947 \newcommand*{\defglsdisplayfirst}[2][\glsdefaulttype]{%
2948   \GlossariesWarning{\string\defglsdisplayfirst\space is now obsolete.^^J
2949   Use \string\defglsentryfmt\space instead}%
2950   \expandafter\def\csname gls@#1@displayfirst\endcsname##1##2##3##4{#2}%
2951   \edef\@gls@doentrydef{%
2952     \noexpand\defglsentryfmt[#1]{%
2953       \noexpand\ifcsdef{gls@#1@display}%
2954       {%
2955         \noexpand\@gls@default@entryfmt
2956         {\noexpand\csuse{gls@#1@displayfirst}}}%
2957         {\noexpand\csuse{gls@#1@display}}}%
2958       }%
2959       {%
2960         \noexpand\@gls@default@entryfmt
2961         {\noexpand\csuse{gls@#1@displayfirst}}}%

```

```

2962         {\noexpand\glsdisplay}%
2963     }%
2964 }%
2965 }%
2966 \@gls@doentrydef
2967 }

```

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.

```

2968 \define@key{glslink}{counter}{%
2969     \ifcsundef{c@#1}%
2970     {%
2971         \PackageError{glossaries}%
2972         {There is no counter called ‘#1’}%
2973         {%
2974             The counter key should have the name of a valid counter
2975             as its value%
2976         }%
2977     }%
2978 }%
2979 \def\@gls@counter{#1}%
2980 }%
2981 }

```

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.

```

2982 \define@key{glslink}{format}{%
2983     \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.

```

2984 \define@boolkey{glslink}{hyper}[true]{}

```

Initialise hyper key.

```
2985 \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.

```
2986 \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.

```
2987 \newcommand*{\glslinkvar}[3]{#1}
```

`\glsifhyper` Now deprecated.

```
2988 \newcommand*{\glsifhyper}[2]{%
2989 \glslinkvar{#1}{#2}{#1}%
2990 \GlossariesWarning{\string\glsifhyper\space is deprecated. Did
2991 you mean \string\glsifhyperon\space or \string\glslinkvar?}%
2992 }
```

`\@gls@hyp@opt` Used by the commands such as `\glslink` to determine whether to modify the hyper option.

```
2993 \newcommand*{\@gls@hyp@opt}[1]{%
2994 \let\glslinkvar\@firstofthree
2995 \let\@gls@hyp@opt@cs#1\relax
2996 \@ifstar{\s@gls@hyp@opt}%
2997 {\@ifnextchar+{\@firstoftwo{\p@gls@hyp@opt}}{#1}}%
2998 }
```

`\s@gls@hyp@opt` Starred version

```
2999 \newcommand*{\s@gls@hyp@opt}[1][ ]{%
3000 \let\glslinkvar\@secondofthree
3001 \@gls@hyp@opt@cs[hyper=false,#1]}
```

`\p@gls@hyp@opt` Plus version

```
3002 \newcommand*{\p@gls@hyp@opt}[1][ ]{%
3003 \let\glslinkvar\@thirdofthree
3004 \@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`

```
3005 \newrobustcmd*{\glslink}{%
3006 \@gls@hyp@opt\@gls@@link
3007 }
```

`\@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.

```
3008 \newcommand*{\@gls@@link}[3] [] {%
3009 \ifglsentryexists{#2}%
3010 {%
3011 \let\do@gls@link@checkfirsthyper\relax
3012 \@gls@link[#1]{#2}{#3}%
3013 }{%
3014 \PackageError{glossaries}{Glossary entry ‘#2’ has not been
3015 defined}{You need to define a glossary entry before you
3016 can use it.}%

```

Display the specified text. (The entry doesn't exist so there's nothing to link it to.)

```
3017 \glstextformat{#3}%
3018 }%

3019 \glspostlinkhook
3020 }
```

`\glspostlinkhook`

```
3021 \newcommand*{\glspostlinkhook}{}
3022 % \end{macrocode}
3023 %\end{macro}
3024 %
3025 %
3026 %\begin{macro}{\@gls@link@checkfirsthyper}
3027 % Check for first use and switch off \gloskey[glslink]{hyper} key
3028 % if hyperlink not wanted. (Should be off if first use and

```

```

3029% hyper=false is on or if first use and both the entry is in an acronym
3030% list and the acrfootnote setting is on.)
3031% This assumes the glossary type is stored in \cs{glstype} and the
3032% label is stored in \cs{glslabel}.
3033%\changes{4.08}{2014-07-30}{new}
3034% \begin{macrocode}
3035\newcommand*{\@gls@link@checkfirsthyper}{%
3036 \ifglsused{\glslabel}%
3037 {%
3038 }%
3039 {%
3040 \gls@checkisacronymlist\glstype
3041 \ifglshyperfirst
3042 \ifglsisacronymlist
3043 \ifglsacrfootnote
3044 \KV@glslink@hyperfalse
3045 \fi
3046 \fi
3047 \else
3048 \KV@glslink@hyperfalse
3049 \fi
3050 }%

```

Allow user to hook into this

```

3051 \glslinkcheckfirsthyperhook
3052 }

```

checkfirsthyperhook Allow used to hook into the \@gls@link@checkfirsthyper macro

```

3053 \newcommand*{\glslinkcheckfirsthyperhook}{}

```

\glslinkpostsetkeys

```

3054 \newcommand*{\glslinkpostsetkeys}{}

```

\glsifhyperon Check the value of the hyper key:

```

3055 \newcommand{\glsifhyperon}[2]{\ifKV@glslink@hyper#1\else#2\fi}

```

\@gls@link

```

3056 \def\@gls@link[#1]#2#3{%

```

Inserting \leavevmode suggested by Donald Arseneau (avoids problem with tabularx).

```

3057 \leavevmode
3058 \edef\glslabel{\glsdetoklabel{#2}}%

```

Save options in \@gls@link@opts and label in \@gls@link@label

```

3059 \def\@gls@link@opts{#1}%
3060 \let\@gls@link@label\glslabel

3061 \def\@glsnumberformat{\glsnumberformat}%
3062 \edef\@gls@counter{\csname glo@glslabel @counter\endcsname}%

```

If this is in one of the “nohypertypes” glossaries, suppress the hyperlink by default

```
3063 \edef\glstype{\csname glo@\glslabel @type\endcsname}%
```

Save original setting

```
3064 \let\org@ifKV@glslink@hyper\ifKV@glslink@hyper
```

Switch off hyper setting if the glossary type has been identified in nohyperlist.

```
3065 \expandafter\DTLifinlist\expandafter
```

```
3066 {\glstype}{\@gls@nohyperlist}%
```

```
3067 {%
```

```
3068 \KV@glslink@hyperfalse
```

```
3069 }%
```

```
3070 {%
```

```
3071 }%
```

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.

```
3072 \do@gls@link@checkfirsthyper
```

```
3073 \setkeys{glslink}{#1}%
```

Add a hook for the user to customise things after the keys have been set.

```
3074 \glslinkpostsetkeys
```

Store the entry's counter in \theglscopycounter

```
3075 \@gls@saveentrycounter
```

Define sort key if necessary:

```
3076 \@gls@setsort{\glslabel}%
```

(De-tok'ing done by \@do@wrglossary)

```
3077 \@do@wrglossary{#2}%
```

```
3078 \ifKV@glslink@hyper
```

```
3079 \@glslink{\glolinkprefix\glslabel}{\glstextformat{#3}}%
```

```
3080 \else
```

```
3081 \glstextformat{#3}%
```

```
3082 \fi
```

Restore original setting

```
3083 \let\ifKV@glslink@hyper\org@ifKV@glslink@hyper
```

```
3084 }
```

\glolinkprefix

```
3085 \newcommand*{\glolinkprefix}{glo:}
```

\glscopycounter Set default value of entry counter

```
3086 \def\glscopycounter{\glscounter}%
```

ls@saveentrycounter Need to check if using equation counter in align environment:

```
3087 \newcommand*{\@gls@saveentrycounter}{%
3088   \def\@gls@Hcounter{}}%
```

Are we using equation counter?

```
3089   \ifthenelse{\equal{\@gls@counter}{equation}}{%
3090   {
```

If we're in align environment, \xatlevel@ will be defined. (Can't test for \currentenvir as may be inside an inner environment.)

```
3091   \ifcsundef{xatlevel@}%
3092   {%
3093     \edef\theglsentrycounter{\expandafter\noexpand
3094       \csname the\@gls@counter\endcsname}%
3095   }%
3096   {%
3097     \ifx\xatlevel@\@empty
3098       \edef\theglsentrycounter{\expandafter\noexpand
3099         \csname the\@gls@counter\endcsname}%
3100     \else
3101       \savecounters@
3102       \advance\c@equation by 1\relax
3103       \edef\theglsentrycounter{\csname the\@gls@counter\endcsname}%
3104
```

Check if hyperref version of this counter

```
3104     \ifcsundef{theH\@gls@counter}%
3105     {%
3106       \def\@gls@Hcounter{\theglsentrycounter}%
3107     }%
3108     {%
3109       \def\@gls@Hcounter{\csname theH\@gls@counter\endcsname}%
3110     }%
3111     \protected@edef\theHglentrycounter{\@gls@Hcounter}%
3112     \restorecounters@
3113   \fi
3114 }%
3115 }%
3116 {%
```

Not using equation counter so no special measures:

```
3117   \edef\theglsentrycounter{\expandafter\noexpand
3118     \csname the\@gls@counter\endcsname}%
3119   }%
```

Check if hyperref version of this counter

```
3120   \ifx\@gls@Hcounter\@empty
3121     \ifcsundef{theH\@gls@counter}%
3122     {%
3123       \def\theHglentrycounter{\theglsentrycounter}%
3124     }%
3125     {%
```

```

3126     \protected@edef\theHglentrycounter{\expandafter\noexpand
3127     \csname theH\@gls@counter\endcsname}%
3128 }%
3129 \fi
3130 }

```

`\@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.

```

3131 \def\@set@glo@numformat#1#2#3#4{%
3132   \expandafter\@glo@check@mkidxrangechar#3\@nil
3133   \protected@edef#1{%
3134     \@glo@prefix setentrycounter[#4]{#2}%
3135     \expandafter\string\csname\@glo@suffix\endcsname
3136   }%
3137   \@gls@checkmkidxchars#1%
3138 }

```

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.

```

3139 \def\@glo@check@mkidxrangechar#1#2\@nil{%
3140 \if#1(\relax
3141   \def\@glo@prefix{(%}
3142   \if\relax#2\relax
3143     \def\@glo@suffix{glsnumberformat}%
3144   \else
3145     \def\@glo@suffix{#2}%
3146   \fi
3147 \else
3148   \if#1)\relax
3149     \def\@glo@prefix{)%}
3150     \if\relax#2\relax
3151       \def\@glo@suffix{glsnumberformat}%
3152     \else
3153       \def\@glo@suffix{#2}%
3154     \fi
3155   \else
3156     \def\@glo@prefix{}\def\@glo@suffix{#1#2}%
3157   \fi
3158 \fi}

```

`\@gls@escbsdq` Escape backslashes and double quote marks. The argument must be a control sequence.

```

3159 \newcommand*{\@gls@escbsdq}[1]{%
3160   \def\@gls@checkedmkidx{}%
3161   \let\gls@xdystring=#1\relax
3162   \@onelevel@sanitize\gls@xdystring
3163   \edef\do@gls@xdycheckbackslash{%
3164     \noexpand\@gls@xdycheckbackslash\gls@xdystring\noexpand\@nil
3165     \@backslashchar\@backslashchar\noexpand\null}%
3166   \do@gls@xdycheckbackslash
3167   \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdystring}%
3168   \def\@gls@checkedmkidx{}%
3169   \expandafter\@gls@xdycheckquote\gls@xdystring\@nil""\null
3170   \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdystring}%

```

Unsanitize \gls@numberpage, \gls@alphpage, \gls@Alphpage and \gls@romanpage
(thanks to David Carlisle for the suggestion.)

```

3171   \@for\@gls@tmp:=\gls@protected@pagefmts\do
3172   {%
3173     \edef\@gls@sanitized@tmp{\expandafter\@gobble\string\\expandonce\@gls@tmp}%
3174     \@onelevel@sanitize\@gls@sanitized@tmp
3175     \edef\gls@dosubst{%
3176       \noexpand\DTLsubstituteall\noexpand\gls@xdystring
3177       {\@gls@sanitized@tmp}{\expandonce\@gls@tmp}%
3178     }%
3179     \gls@dosubst
3180   }%

```

Assign to required control sequence

```

3181   \let#1=\gls@xdystring
3182 }

```

Catch special characters (argument must be a control sequence):

`\gls@checkmkidxchars`

```

3183 \newcommand{\@gls@checkmkidxchars}[1]{%
3184   \ifglxindy
3185     \@gls@escbsdq{#1}%
3186   \else
3187     \def\@gls@checkedmkidx{}%
3188     \expandafter\@gls@checkquote#1\@nil""\null
3189     \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3190     \def\@gls@checkedmkidx{}%
3191     \expandafter\@gls@checkescquote#1\@nil""\null
3192     \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3193     \def\@gls@checkedmkidx{}%
3194     \expandafter\@gls@checkescactual#1\@nil"??"\null
3195     \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3196     \def\@gls@checkedmkidx{}%
3197     \expandafter\@gls@checkactual#1\@nil??\null
3198     \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3199     \def\@gls@checkedmkidx{}%

```

```

3200 \expandafter\@gls@checkbar#1\@nil||\null
3201 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3202 \def\@gls@checkedmkidx{}%
3203 \expandafter\@gls@checkescbar#1\@nil|||\null
3204 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3205 \def\@gls@checkedmkidx{}%
3206 \expandafter\@gls@checklevel#1\@nil!!\null
3207 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3208 \fi
3209 }

```

Update the control sequence and strip trailing \@nil:

\@gls@updatechecked

```

3210 \def\@gls@updatechecked#1\@nil#2{\def#2{#1}}

```

\@gls@tmpb Define temporary token

```

3211 \newtoks\@gls@tmpb

```

\@gls@checkquote Replace " with "" since " is a makeindex special character.

```

3212 \def\@gls@checkquote#1"#2"#3\null{%
3213 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3214 \toks@={#1}%
3215 \ifx\null#2\null
3216 \ifx\null#3\null
3217 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3218 \def\@gls@checkquote{\relax}%
3219 \else
3220 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3221 \@gls@quotechar\@gls@quotechar\@gls@quotechar\@gls@quotechar}%
3222 \def\@gls@checkquote{\@gls@checkquote#3\null}%
3223 \fi
3224 \else
3225 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3226 \@gls@quotechar\@gls@quotechar}%
3227 \ifx\null#3\null
3228 \def\@gls@checkquote{\@gls@checkquote#2""\null}%
3229 \else
3230 \def\@gls@checkquote{\@gls@checkquote#2"#3\null}%
3231 \fi
3232 \fi
3233 \@gls@checkquote
3234 }

```

\@gls@checkescquote Do the same for \":

```

3235 \def\@gls@checkescquote#1\"#2\"#3\null{%
3236 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3237 \toks@={#1}%
3238 \ifx\null#2\null

```

```

3239 \ifx\null#3\null
3240 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3241 \def\@@gls@checkescquote{\relax}%
3242 \else
3243 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3244 \@gls@quotearchar\string"\@gls@quotearchar
3245 \@gls@quotearchar\string"\@gls@quotearchar}%
3246 \def\@@gls@checkescquote{\@gls@checkescquote#3\null}%
3247 \fi
3248 \else
3249 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3250 \@gls@quotearchar\string"\@gls@quotearchar}%
3251 \ifx\null#3\null
3252 \def\@@gls@checkescquote{\@gls@checkescquote#2"\null}%
3253 \else
3254 \def\@@gls@checkescquote{\@gls@checkescquote#2"#3\null}%
3255 \fi
3256 \fi
3257 \@@gls@checkescquote
3258 }

```

\@gls@checkescactual Similarly for \? (which is replaces @ as makeindex's special character):

```

3259 \def\@gls@checkescactual#1\?#2\?#3\null{%
3260 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3261 \toks@={#1}%
3262 \ifx\null#2\null
3263 \ifx\null#3\null
3264 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3265 \def\@@gls@checkescactual{\relax}%
3266 \else
3267 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3268 \@gls@quotearchar\string"\@gls@actualchar
3269 \@gls@quotearchar\string"\@gls@actualchar}%
3270 \def\@@gls@checkescactual{\@gls@checkescactual#3\null}%
3271 \fi
3272 \else
3273 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3274 \@gls@quotearchar\string"\@gls@actualchar}%
3275 \ifx\null#3\null
3276 \def\@@gls@checkescactual{\@gls@checkescactual#2\?\?\null}%
3277 \else
3278 \def\@@gls@checkescactual{\@gls@checkescactual#2\?#3\null}%
3279 \fi
3280 \fi
3281 \@@gls@checkescactual
3282 }

```

\@gls@checkescbar Similarly for \|:

```

3283 \def\@gls@checkescbar#1\|#2\|#3\null{%

```



```

3284 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3285 \toks@={#1}%
3286 \ifx\null#2\null
3287 \ifx\null#3\null
3288 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3289 \def\@gls@checkescbar{\relax}%
3290 \else
3291 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3292 \@gls@quotechar\string"\@gls@encapchar
3293 \@gls@quotechar\string"\@gls@encapchar}%
3294 \def\@gls@checkescbar{\@gls@checkescbar#3\null}%
3295 \fi
3296 \else
3297 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3298 \@gls@quotechar\string"\@gls@encapchar}%
3299 \ifx\null#3\null
3300 \def\@gls@checkescbar{\@gls@checkescbar#2\|\|\null}%
3301 \else
3302 \def\@gls@checkescbar{\@gls@checkescbar#2\|#3\null}%
3303 \fi
3304 \fi
3305 \@gls@checkescbar
3306 }

```

\@gls@checkesclevel Similarly for \!:

```

3307 \def\@gls@checkesclevel#1\!#2\!#3\null{%
3308 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3309 \toks@={#1}%
3310 \ifx\null#2\null
3311 \ifx\null#3\null
3312 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3313 \def\@gls@checkesclevel{\relax}%
3314 \else
3315 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3316 \@gls@quotechar\string"\@gls@levelchar
3317 \@gls@quotechar\string"\@gls@levelchar}%
3318 \def\@gls@checkesclevel{\@gls@checkesclevel#3\null}%
3319 \fi
3320 \else
3321 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3322 \@gls@quotechar\string"\@gls@levelchar}%
3323 \ifx\null#3\null
3324 \def\@gls@checkesclevel{\@gls@checkesclevel#2\!\!\null}%
3325 \else
3326 \def\@gls@checkesclevel{\@gls@checkesclevel#2\!#3\null}%
3327 \fi
3328 \fi
3329 \@gls@checkesclevel
3330 }

```

\@gls@checkbar and for |:

```
3331 \def\@gls@checkbar#1|#2|#3\null{%
3332   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3333   \toks@={#1}%
3334   \ifx\null#2\null
3335     \ifx\null#3\null
3336       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3337       \def\@gls@checkbar{\relax}%
3338     \else
3339       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3340         \@gls@quotechar\@gls@encapchar\@gls@quotechar\@gls@encapchar}%
3341       \def\@gls@checkbar{\@gls@checkbar#3\null}%
3342     \fi
3343   \else
3344     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3345       \@gls@quotechar\@gls@encapchar}%
3346     \ifx\null#3\null
3347       \def\@gls@checkbar{\@gls@checkbar#2||\null}%
3348     \else
3349       \def\@gls@checkbar{\@gls@checkbar#2|#3\null}%
3350     \fi
3351   \fi
3352   \@gls@checkbar
3353 }
```

\@gls@checklevel and for !:

```
3354 \def\@gls@checklevel#1!#2!#3\null{%
3355   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3356   \toks@={#1}%
3357   \ifx\null#2\null
3358     \ifx\null#3\null
3359       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3360       \def\@gls@checklevel{\relax}%
3361     \else
3362       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3363         \@gls@quotechar\@gls@levelchar\@gls@quotechar\@gls@levelchar}%
3364       \def\@gls@checklevel{\@gls@checklevel#3\null}%
3365     \fi
3366   \else
3367     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3368       \@gls@quotechar\@gls@levelchar}%
3369     \ifx\null#3\null
3370       \def\@gls@checklevel{\@gls@checklevel#2!!\null}%
3371     \else
3372       \def\@gls@checklevel{\@gls@checklevel#2!#3\null}%
3373     \fi
3374   \fi
3375   \@gls@checklevel
3376 }
```

\@gls@checkactual and for ?:

```
3377 \def\@gls@checkactual#1?#2?#3\null{%
3378   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3379   \toks@={#1}%
3380   \ifx\null#2\null
3381     \ifx\null#3\null
3382       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3383       \def\@gls@checkactual{\relax}%
3384     \else
3385       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3386         \@gls@quotechar\@gls@actualchar\@gls@quotechar\@gls@actualchar}%
3387       \def\@gls@checkactual{\@gls@checkactual#3\null}%
3388     \fi
3389   \else
3390     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3391       \@gls@quotechar\@gls@actualchar}%
3392     \ifx\null#3\null
3393       \def\@gls@checkactual{\@gls@checkactual#2??\null}%
3394     \else
3395       \def\@gls@checkactual{\@gls@checkactual#2?#3\null}%
3396     \fi
3397   \fi
3398   \@gls@checkactual
3399 }
```

\@gls@xdycheckquote As before but for use with xindy

```
3400 \def\@gls@xdycheckquote#1"#2"#3\null{%
3401   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3402   \toks@={#1}%
3403   \ifx\null#2\null
3404     \ifx\null#3\null
3405       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3406       \def\@gls@xdycheckquote{\relax}%
3407     \else
3408       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3409         \string"\string"}%
3410       \def\@gls@xdycheckquote{\@gls@xdycheckquote#3\null}%
3411     \fi
3412   \else
3413     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3414       \string"}%
3415     \ifx\null#3\null
3416       \def\@gls@xdycheckquote{\@gls@xdycheckquote#2""\null}%
3417     \else
3418       \def\@gls@xdycheckquote{\@gls@xdycheckquote#2"#3\null}%
3419     \fi
3420   \fi
3421   \@gls@xdycheckquote
3422 }
```

s@xdycheckbackslash Need to escape all backslashes for xindy. Define command that will define

```

\@gls@xdycheckbackslash
3423 \edef\def@gls@xdycheckbackslash{%
3424 \noexpand\def\noexpand\@gls@xdycheckbackslash##1\@backslashchar
3425 ##2\@backslashchar##3\noexpand\null{%
3426 \noexpand\@gls@tmpb=\noexpand\expandafter
3427 {\noexpand\@gls@checkedmkidx}%
3428 \noexpand\toks@={##1}%
3429 \noexpand\ifx\noexpand\null##2\noexpand\null
3430 \noexpand\ifx\noexpand\null##3\noexpand\null
3431 \noexpand\edef\noexpand\@gls@checkedmkidx{%
3432 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
3433 \noexpand\def\noexpand\@gls@xdycheckbackslash{\relax}%
3434 \noexpand\else
3435 \noexpand\edef\noexpand\@gls@checkedmkidx{%
3436 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
3437 \@backslashchar\@backslashchar\@backslashchar\@backslashchar}%
3438 \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3439 \noexpand\@gls@xdycheckbackslash##3\noexpand\null}%
3440 \noexpand\fi
3441 \noexpand\else
3442 \noexpand\edef\noexpand\@gls@checkedmkidx{%
3443 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
3444 \@backslashchar\@backslashchar}%
3445 \noexpand\ifx\noexpand\null##3\noexpand\null
3446 \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3447 \noexpand\@gls@xdycheckbackslash##2\@backslashchar
3448 \@backslashchar\noexpand\null}%
3449 \noexpand\else
3450 \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3451 \noexpand\@gls@xdycheckbackslash##2\@backslashchar
3452 ##3\noexpand\null}%
3453 \noexpand\fi
3454 \noexpand\fi
3455 \noexpand\@gls@xdycheckbackslash
3456 }%
3457 }

```

Now go ahead and define \@gls@xdycheckbackslash

```
3458 \def@gls@xdycheckbackslash
```

\glsdohypertarget

```

3459 \newlength@gls@tmplen
3460 \newcommand*\glsdohypertarget}[2]{%
3461 \settoheight{\gls@tmplen}{#2}%
3462 \raisebox{\gls@tmplen}{\hypertarget{#1}}{#2}%
3463 }

```

\glsdohyperlink

```
3464 \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`.

```
3465 \ifcsundef{hyperlink}%  
3466 {%  
3467   \let\@glslink\@secondoftwo  
3468 }%  
3469 {%  
3470   \let\@glslink\glsdohyperlink  
3471 }
```

`\@glstarget` If `\hypertarget` is not defined, `\@glstarget` ignores its first argument and just does the second argument, otherwise it is equivalent to `\hypertarget`.

```
3472 \ifcsundef{hypertarget}%  
3473 {%  
3474   \let\@glstarget\@secondoftwo  
3475 }%  
3476 {%  
3477   \let\@glstarget\glsdohypertarget  
3478 }
```

Glossary hyperlinks can be disabled using `\glsdisablehyper` (effect can be localised):

`\glsdisablehyper`

```
3479 \newcommand{\glsdisablehyper}{%  
3480   \KV@glslink@hyperfalse  
3481   \let\@glslink\@secondoftwo  
3482   \let\@glstarget\@secondoftwo  
3483 }
```

Glossary hyperlinks can be enabled using `\glsenablehyper` (effect can be localised):

`\glsenablehyper`

```
3484 \newcommand{\glsenablehyper}{%  
3485   \KV@glslink@hypertrue  
3486   \let\@glslink\glsdohyperlink  
3487   \let\@glstarget\glsdohypertarget  
3488 }
```

Provide some convenience commands if not already defined:

```
3489 \providecommand{\@firstofthree}[3]{#1}  
3490 \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`

```
3491 \newrobustcmd*{\gls}{\@gls@hyp@opt\@gls}
```

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

`\@gls`

```
3492 \newcommand*{\@gls}[2][]{%
3493   \new@ifnextchar[{\@gls@{#1}{#2}}{\@gls@{#1}{#2}[]}%
3494 }
```

`\@gls@` Read in the final optional argument:

```
3495 \def\@gls@#1#2[#3]{%
3496   \glsdoifexists{#2}%
3497   {%
3498     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
3499     \let\glsifplural\@secondoftwo
3500     \let\glsupcase\@firstofthree
3501     \let\glscustomtext\@empty
3502     \def\glsinsert{#3}%
```

Determine what the link text should be (this is stored in `\@glo@text`) Note that `\@gls@link` sets `\glstyp`.

```
3503   \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.

```
3504   \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3505   \ifKV@glslink@local
3506     \glslocalunset{#2}%
3507   \else
3508     \glsunset{#2}%
3509   \fi
3510 }
```

`\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:

```

3514 \newcommand*{\@Gls}[2][\%
3515   \new@ifnextchar[{\@Gls@{#1}{#2}}{\@Gls@{#1}{#2}[]}%
3516 }

```

```

3517 \def\@Gls@#1#2[#3]{%
3518   \glsdoifexists{#2}%
3519   {%
3520     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
3521     \let\glsifplural\@secondoftwo
3522     \let\glscapscase\@secondofthree
3523     \let\glscustomtext\@empty
3524     \def\glsinsert{#3}%

```

```
3525 \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%
```

```
3526 \@gls@link[#1]{#2}{\@glo@text}%
```

```

3527     \ifKV@glslink@local
3528         \glsllocalunset{#2}%
3529     \else
3530         \glslunset{#2}%
3531     \fi
3532 }%

```

`\GLS` behaves like `\gls`, but the link text is converted to uppercase:

`\GLS`

```
3535 \newrobustcmd*{\GLS}{\@gls@hyp@opt\@GLS}
```

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

```
3536 \newcommand*{\@GLS}[2][\@GLS@hyp@opt\@GLS]{%
3537   \new@ifnextchar[{\@GLS@{#1}{#2}}{\@GLS@{#1}{#2}[]}%
3538 }
```

`\@GLS@` Read in the final optional argument:

```
3539 \def\@GLS@#1#2[#3]{%
3540   \glsdoifexists{#2}%
3541   {%
3542     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
3543     \let\glsifplural\@secondoftwo
3544     \let\glscapscase\@thirdofthree
3545     \let\glscustomtext\@empty
3546     \def\glsinsert{#3}%
3547     \def\@glo@text{\csname gls@\gls@type @entryfmt\endcsname}%
3548     \def\@gls@link[#1]{#2}{\@glo@text}%
3549     \ifKV@glslink@local
3550       \glslocalunset{#2}%
3551     \else
3552       \glsunset{#2}%
3553     \fi
3554   }%
3555   \gls@postlinkhook
3556 }
```

Determine what the link text should be (this is stored in `\@glo@text`). Note that `\@gls@link` sets `\gls@type`.

```
3547 \def\@glo@text{\csname gls@\gls@type @entryfmt\endcsname}%
```

Call `\@gls@link` If footnote package option has been used and the glossary type is `\acronym@type`, suppress hyperlink for first use. Likewise if the `hyperfirst=false` package option is used.

```
3548 \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3549 \ifKV@glslink@local
3550   \glslocalunset{#2}%
3551 \else
3552   \glsunset{#2}%
3553 \fi
3554 }%
3555 \gls@postlinkhook
3556 }
```

`\glspl` behaves in the same way as `\gls` except it uses the plural form.

`\glspl`

```
3557 \newrobustcmd*{\glspl}{\@gls@hyp@opt\@glspl}
```

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

```
3558 \newcommand*{\@glspl}[2][\@glspl@hyp@opt\@glspl]{%
3559   \new@ifnextchar[{\@glspl@{#1}{#2}}{\@glspl@{#1}{#2}[]}%
3560 }
```


`\@glsp1@` Read in the final optional argument:

```
3561 \def\@glsp1@#1#2[#3]{%
3562   \glsdoifexists{#2}%
3563   {%
3564     \let\do@gl@link@checkfirsthyper\@gl@link@checkfirsthyper

3565     \let\glsifplural\@firstoftwo
3566     \let\glscapscase\@firstofthree
3567     \let\glscustomtext\@empty
3568     \def\glsinsert{#3}%
```

Determine what the link text should be (this is stored in `\@glo@text`) Note that `\@gl@link` sets `\glstype`.

```
3569   \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%
```

Call `\@gl@link`. If footnote package option has been used and the glossary type is `\acronymtype`, suppress hyperlink for first use. Likewise if the `hyperfirst=false` package option is used.

```
3570   \@gl@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3571   \ifKV@gl@link@local
3572   \glslocalunset{#2}%
3573   \else
3574   \glsunset{#2}%
3575   \fi
3576 }%
```

```
3577 \glspostlinkhook
3578 }
```

`\Glspl` behaves in the same way as `\glsp1`, except that the first letter of the link text is converted to uppercase (as with `\Gls`, if the first letter has an accent, it will need to be grouped).

`\Glspl`

```
3579 \newrobustcmd*{\Glspl}{\@gl@hyp@opt\@Glspl}
```

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

```
3580 \newcommand*{\@Glspl}[2][ ]{%
3581   \new@ifnextchar[{\@Glspl@{#1}{#2}}{\@Glspl@{#1}{#2}[]}%
3582 }
```

`\@Glspl@` Read in the final optional argument:

```
3583 \def\@Glspl@#1#2[#3]{%
3584   \glsdoifexists{#2}%
3585   {%
3586     \let\do@gl@link@checkfirsthyper\@gl@link@checkfirsthyper
```

```

3587 \let\glsifplural\@firstoftwo
3588 \let\glscapscase\@secondofthree
3589 \let\glscustomtext\@empty
3590 \def\glsinsert{#3}%

```

Determine what the link text should be (this is stored in `\@glo@text`). This needs to be expanded so that the `\@glo@text` can be passed to `\xmakefirstuc`. Note that `\@gls@link` sets `\glstype`.

```

3591 \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.

```

3592 \@gls@link[#1]{#2}{\@glo@text}%

```

Indicate that this entry has now been used

```

3593 \ifKV@glslink@local
3594 \glsllocalunset{#2}%
3595 \else
3596 \glsunset{#2}%
3597 \fi
3598 }%

```

```

3599 \glspostlinkhook
3600 }

```

`\GLSp1` behaves like `\glspl` except that all the link text is converted to uppercase.

`\GLSp1`

```

3601 \newrobustcmd*{\GLSp1}{\@gls@hyp@opt\@GLSp1}

```

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

```

3602 \newcommand*{\@GLSp1}[2][{}]{%
3603 \new@ifnextchar[{\@GLSp1@{#1}{#2}}{\@GLSp1@{#1}{#2}[]}%
3604 }

```

`\@GLSp1` Read in the final optional argument:

```

3605 \def\@GLSp1@#1#2[#3]{%
3606 \glsoifexists{#2}%
3607 {%
3608 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3609 \let\glsifplural\@firstoftwo
3610 \let\glscapscase\@thirdofthree
3611 \let\glscustomtext\@empty
3612 \def\glsinsert{#3}%

```

Determine what the link text should be (this is stored in `\@glo@text`) Note that `\@gls@link` sets `\glstype`.

```
3613 \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.

```
3614 \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3615 \ifKV@glslink@local
```

```
3616 \glslocalunset{#2}%
```

```
3617 \else
```

```
3618 \glsunset{#2}%
```

```
3619 \fi
```

```
3620 }%
```

```
3621 \glspostlinkhook
```

```
3622 }
```

`\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:

```
3623 \newrobustcmd*{\glsdisp}{\@gls@hyp@opt\@glsdisp}
```

Defined the un-starred form.

`\@glsdisp`

```
3624 \newcommand*{\@glsdisp}[3][]{%
```

```
3625 \glsdoifexists{#2}{%
```

```
3626 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
```

```
3627 \let\glsifplural\@secondoftwo
```

```
3628 \let\glscapscase\@firstofthree
```

```
3629 \def\glscustomtext{#3}%
```

```
3630 \def\glsinsert{}%
```

Determine what the link text should be (this is stored in `\@glo@text`) Note that `\@gls@link` sets `\glstype`.

```
3631 \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.

```
3632 \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3633 \ifKV@glslink@local
3634 \glsllocalunset{#2}%
3635 \else
3636 \glunset{#2}%
3637 \fi
3638 }%

3639 \glspostlinkhook
3640 }
```

\@gls@field@link

```
3641 \newcommand{\@gls@field@link}[3]{%
3642 \glstoifexists{#2}%
3643 {%
3644 \let\do@gls@link@checkfirsthyper\relax
3645 \@gls@link[#1]{#2}{#3}%
3646 }%

3647 \glspostlinkhook
3648 }
```

\gls{text} behaves like \gls except it always uses the value given by the text key and it doesn't mark the entry as used.

\gls{text}

```
3649 \newrobustcmd*{\gls{text}}{\@gls@hyp@opt\@gls{text}}
```

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

```
3650 \newcommand*{\@gls{text}}[2][ ]{%
3651 \new@ifnextchar[{\@gls{text@{#1}{#2}}]{\@gls{text@{#1}{#2}}[ ]}}
```

Read in the final optional argument:

```
3652 \def\@gls{text@#1#2[#3]}{%
3653 \@gls@field@link{#1}{#2}{\glstentrytext{#2}#3}%
3654 }
```

\GLStext behaves like \gls{text} except the text is converted to uppercase.

\GLStext

```
3655 \newrobustcmd*{\GLStext}{\@gls@hyp@opt\@GLStext}
```

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

```
3656 \newcommand*{\@GLStext}[2][ ]{%
3657 \new@ifnextchar[{\@GLStext@{#1}{#2}}{\@GLStext@{#1}{#2}}[ ]}}
```

Read in the final optional argument:

```
3658 \def\@GLStext@#1#2[#3]{%
3659 \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glstentrytext{#2}#3}}%
3660 }
```

`\Glstext` behaves like `\glstext` except that the first letter of the text is converted to uppercase.

`\Glstext`

```
3661 \newrobustcmd*{\Glstext}{\@gls@hyp@opt\@Glstext}
```

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

```
3662 \newcommand*{\@Glstext}[2][]{%
```

```
3663   \new@ifnextchar[{\@Glstext@{#1}{#2}}{\@Glstext@{#1}{#2}[]}]}
```

Read in the final optional argument:

```
3664 \def\@Glstext@#1#2[#3]{%
```

```
3665   \@gls@field@link{#1}{#2}{\Glsentrytext{#2}#3}%
```

```
3666 }
```

`\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`

```
3667 \newrobustcmd*{\glsfirst}{\@gls@hyp@opt\@glsfirst}
```

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

```
3668 \newcommand*{\@glsfirst}[2][]{%
```

```
3669   \new@ifnextchar[{\@glsfirst@{#1}{#2}}{\@glsfirst@{#1}{#2}[]}]}
```

Read in the final optional argument:

```
3670 \def\@glsfirst@#1#2[#3]{%
```

```
3671   \@gls@field@link{#1}{#2}{\Glsentryfirst{#2}#3}%
```

```
3672 }
```

`\Glsfirst` behaves like `\glsfirst` except it displays the first letter in uppercase.

`\Glsfirst`

```
3673 \newrobustcmd*{\Glsfirst}{\@gls@hyp@opt\@Glsfirst}
```

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

```
3674 \newcommand*{\@Glsfirst}[2][]{%
```

```
3675   \new@ifnextchar[{\@Glsfirst@{#1}{#2}}{\@Glsfirst@{#1}{#2}[]}]}
```

Read in the final optional argument:

```
3676 \def\@Glsfirst@#1#2[#3]{%
```

```
3677   \@gls@field@link{#1}{#2}{\Glsentryfirst{#2}#3}%
```

```
3678 }
```

`\GLSfirst` behaves like `\Glsfirst` except it displays the text in uppercase.

`\GLSfirst`

```
3679 \newrobustcmd*{\GLSfirst}{\@gls@hyp@opt\@GLSfirst}
```

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

```
3680 \newcommand*{\@GLSfirst}[2] [] {%
3681   \new@ifnextchar[{\@GLSfirst@{#1}{#2}}{\@GLSfirst@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3682 \def\@GLSfirst@#1#2[#3] {%
3683   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryfirst{#2}#3}}%
3684 }
```

`\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`

```
3685 \newrobustcmd*{\glsplural}{\@gls@hyp@opt\@glsplural}
```

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

```
3686 \newcommand*{\@glsplural}[2] [] {%
3687   \new@ifnextchar[{\@glsplural@{#1}{#2}}{\@glsplural@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3688 \def\@glsplural@#1#2[#3] {%
3689   \@gls@field@link{#1}{#2}{\glsentryplural{#2}#3}%
3690 }
```

`\Glsplural` behaves like `\glsplural` except that the first letter is converted to uppercase.

`\Glsplural`

```
3691 \newrobustcmd*{\Glsplural}{\@gls@hyp@opt\@Glsplural}
```

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

```
3692 \newcommand*{\@Glsplural}[2] [] {%
3693   \new@ifnextchar[{\@Glsplural@{#1}{#2}}{\@Glsplural@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3694 \def\@Glsplural@#1#2[#3] {%
3695   \@gls@field@link{#1}{#2}{\Glsentryplural{#2}#3}%
3696 }
```

`\GLSplural` behaves like `\glsplural` except that the text is converted to uppercase.

`\GLSplural`

```
3697 \newrobustcmd*{\GLSplural}{\@gls@hyp@opt\@GLSplural}
```

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

```
3698 \newcommand*{\@GLSplural}[2] [] {%
3699   \new@ifnextchar[{\@GLSplural@{#1}{#2}}{\@GLSplural@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3700 \def\@GLSplural@#1#2[#3]{%
3701   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryplural{#2}#3}}%
3702 }
```

`\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`

```
3703 \newrobustcmd*{\glsfirstplural}{\@gls@hyp@opt\@glsfirstplural}
```

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

```
3704 \newcommand*{\@glsfirstplural}[2][{}]{%
3705   \new@ifnextchar[{\@glsfirstplural@{#1}{#2}}{\@glsfirstplural@{#1}{#2}[{}]}}
```

Read in the final optional argument:

```
3706 \def\@glsfirstplural@#1#2[#3]{%
3707   \@gls@field@link{#1}{#2}{\glsentryfirstplural{#2}#3}%
3708 }
```

`\Glsfirstplural` behaves like `\glsfirstplural` except that the first letter is converted to uppercase.

`\Glsfirstplural`

```
3709 \newrobustcmd*{\Glsfirstplural}{\@gls@hyp@opt\@Glsfirstplural}
```

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

```
3710 \newcommand*{\@Glsfirstplural}[2][{}]{%
3711   \new@ifnextchar[{\@Glsfirstplural@{#1}{#2}}{\@Glsfirstplural@{#1}{#2}[{}]}}
```

Read in the final optional argument:

```
3712 \def\@Glsfirstplural@#1#2[#3]{%
3713   \@gls@field@link{#1}{#2}{\Glsentryfirstplural{#2}#3}%
3714 }
```

`\GLSfirstplural` behaves like `\glsfirstplural` except that the link text is converted to uppercase.

`\GLSfirstplural`

```
3715 \newrobustcmd*{\GLSfirstplural}{\@gls@hyp@opt\@GLSfirstplural}
```

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

```
3716 \newcommand*{\@GLSfirstplural}[2][{}]{%
3717   \new@ifnextchar[{\@GLSfirstplural@{#1}{#2}}{\@GLSfirstplural@{#1}{#2}[{}]}}
```

Read in the final optional argument:

```
3718 \def\@GLSfirstplural@#1#2[#3]{%
3719   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryfirstplural{#2}#3}}%
3720 }
```

`\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`

```
3721 \newrobustcmd*{\glsname}{\@gls@hyp@opt\@glsname}
```

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

```
3722 \newcommand*{\@glsname}[2][\@gls@hyp@opt\@glsname]
```

```
3723 \new@ifnextchar[\@glsname]{\@glsname@{#1}{#2}}{\@glsname@{#1}{#2}[]}
```

Read in the final optional argument:

```
3724 \def\@glsname@#1#2[#3]{%
```

```
3725 \@gls@field@link{#1}{#2}{\glsentryname{#2}#3}%
```

```
3726 }
```

`\Glsname` behaves like `\glsname` except that the first letter is converted to uppercase.

`\Glsname`

```
3727 \newrobustcmd*{\Glsname}{\@gls@hyp@opt\@Glsname}
```

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

```
3728 \newcommand*{\@Glsname}[2][\@Glsname]
```

```
3729 \new@ifnextchar[\@Glsname]{\@Glsname@{#1}{#2}}{\@Glsname@{#1}{#2}[]}
```

Read in the final optional argument:

```
3730 \def\@Glsname@#1#2[#3]{%
```

```
3731 \@gls@field@link{#1}{#2}{\Glsentryname{#2}#3}%
```

```
3732 }
```

`\GLSname` behaves like `\glsname` except that the link text is converted to uppercase.

`\GLSname`

```
3733 \newrobustcmd*{\GLSname}{\@gls@hyp@opt\@GLSname}
```

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

```
3734 \newcommand*{\@GLSname}[2][\@GLSname]
```

```
3735 \new@ifnextchar[\@GLSname]{\@GLSname@{#1}{#2}}{\@GLSname@{#1}{#2}[]}
```

Read in the final optional argument:

```
3736 \def\@GLSname@#1#2[#3]{%
```

```
3737 \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryname{#2}#3}}%
```

```
3738 }
```

`\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`

```
3739 \newrobustcmd*{\glsdesc}{\@gls@hyp@opt\@glsdesc}
```


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

```
3740 \newcommand*{\@glsdesc}[2] [] {%  
3741   \new@ifnextchar[{\@glsdesc@{#1}{#2}}{\@glsdesc@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3742 \def\@glsdesc@#1#2[#3] {%  
3743   \@gls@field@link{#1}{#2}{\@glsentrydesc{#2}#3}%  
3744 }
```

\Glsdesc behaves like \glsdesc except that the first letter is converted to uppercase.

\Glsdesc

```
3745 \newrobustcmd*{\Glsdesc}{\@gls@hyp@opt\@Glsdesc}
```

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

```
3746 \newcommand*{\@Glsdesc}[2] [] {%  
3747   \new@ifnextchar[{\@Glsdesc@{#1}{#2}}{\@Glsdesc@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3748 \def\@Glsdesc@#1#2[#3] {%  
3749   \@gls@field@link{#1}{#2}{\@Glsentrydesc{#2}#3}%  
3750 }
```

\GLSdesc behaves like \glsdesc except that the link text is converted to uppercase.

\GLSdesc

```
3751 \newrobustcmd*{\GLSdesc}{\@gls@hyp@opt\@GLSdesc}
```

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

```
3752 \newcommand*{\@GLSdesc}[2] [] {%  
3753   \new@ifnextchar[{\@GLSdesc@{#1}{#2}}{\@GLSdesc@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3754 \def\@GLSdesc@#1#2[#3] {%  
3755   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\@glsentrydesc{#2}#3}}%  
3756 }
```

\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

```
3757 \newrobustcmd*{\glsdescplural}{\@gls@hyp@opt\@glsdescplural}
```

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

```
3758 \newcommand*{\@glsdescplural}[2] [] {%  
3759   \new@ifnextchar[{\@glsdescplural@{#1}{#2}}{\@glsdescplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3760 \def\@glsdescplural@#1#2[#3]{%
3761   \@gls@field@link{#1}{#2}{\glsentrydescplural{#2}#3}%
3762 }
```

`\Glsdescplural` behaves like `\glsdescplural` except that the first letter is converted to uppercase.

`\Glsdescplural`

```
3763 \newrobustcmd*{\Glsdescplural}{\@gls@hyp@opt\@Glsdescplural}
```

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

```
3764 \newcommand*{\@Glsdescplural}[2][ ]{%
3765   \new@ifnextchar[{\@Glsdescplural@{#1}{#2}}{\@Glsdescplural@{#1}{#2}[ ]}]}
```

Read in the final optional argument:

```
3766 \def\@Glsdescplural@#1#2[#3]{%
3767   \@gls@field@link{#1}{#2}{\glsentrydescplural{#2}#3}%
3768 }
```

`\GLSdescplural` behaves like `\glsdescplural` except that the link text is converted to uppercase.

`\GLSdescplural`

```
3769 \newrobustcmd*{\GLSdescplural}{\@gls@hyp@opt\@GLSdescplural}
```

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

```
3770 \newcommand*{\@GLSdescplural}[2][ ]{%
3771   \new@ifnextchar[{\@GLSdescplural@{#1}{#2}}{\@GLSdescplural@{#1}{#2}[ ]}]}
```

Read in the final optional argument:

```
3772 \def\@GLSdescplural@#1#2[#3]{%
3773   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrydescplural{#2}#3}}%
3774 }
```

`\glssymbol` behaves like `\gls` except it always uses the value given by the symbol key and it doesn't mark the entry as used.

`\glssymbol`

```
3775 \newrobustcmd*{\glssymbol}{\@gls@hyp@opt\@glssymbol}
```

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

```
3776 \newcommand*{\@glssymbol}[2][ ]{%
3777   \new@ifnextchar[{\@glssymbol@{#1}{#2}}{\@glssymbol@{#1}{#2}[ ]}]}
```

Read in the final optional argument:

```
3778 \def\@glssymbol@#1#2[#3]{%
3779   \@gls@field@link{#1}{#2}{\glsentrysymbol{#2}#3}%
3780 }
```

`\Glssymbol` behaves like `\glssymbol` except that the first letter is converted to uppercase.

`\Glssymbol`

```
3781 \newrobustcmd*{\Glssymbol}{\@gls@hyp@opt\@Glssymbol}
```

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

```
3782 \newcommand*{\@Glssymbol}[2] [] {%
```

```
3783   \new@ifnextchar[{\@Glssymbol@{#1}{#2}}{\@Glssymbol@{#1}{#2} []}]
```

Read in the final optional argument:

```
3784 \def\@Glssymbol@#1#2[#3] {%
```

```
3785   \@gls@field@link{#1}{#2}{\glstentrysymbol{#2}#3}%
```

```
3786 }
```

`\GLSsymbol` behaves like `\glssymbol` except that the link text is converted to uppercase.

`\GLSsymbol`

```
3787 \newrobustcmd*{\GLSsymbol}{\@gls@hyp@opt\@GLSsymbol}
```

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

```
3788 \newcommand*{\@GLSsymbol}[2] [] {%
```

```
3789   \new@ifnextchar[{\@GLSsymbol@{#1}{#2}}{\@GLSsymbol@{#1}{#2} []}]
```

Read in the final optional argument:

```
3790 \def\@GLSsymbol@#1#2[#3] {%
```

```
3791   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glstentrysymbol{#2}#3}}%
```

```
3792 }
```

`\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`

```
3793 \newrobustcmd*{\glssymbolplural}{\@gls@hyp@opt\@glssymbolplural}
```

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

```
3794 \newcommand*{\@glssymbolplural}[2] [] {%
```

```
3795   \new@ifnextchar[{\@glssymbolplural@{#1}{#2}}{\@glssymbolplural@{#1}{#2} []}]
```

Read in the final optional argument:

```
3796 \def\@glssymbolplural@#1#2[#3] {%
```

```
3797   \@gls@field@link{#1}{#2}{\glstentrysymbolplural{#2}#3}%
```

```
3798 }
```

`\Glssymbolplural` behaves like `\glssymbolplural` except that the first letter is converted to uppercase.

`\Glssymbolplural`

```
3799 \newrobustcmd*{\Glssymbolplural}{\@gls@hyp@opt\@Glssymbolplural}
```

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

```
3800 \newcommand*{\@Glssymbolplural}[2] [] {%
3801   \new@ifnextchar[{\@Glssymbolplural@{#1}{#2}}{\@Glssymbolplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3802 \def\@Glssymbolplural@#1#2[#3] {%
3803   \@gls@field@link{#1}{#2}{\Glsentrysymbolplural{#2}#3}%
3804 }
```

\GLSsymbolplural behaves like \glsymbolplural except that the link text is converted to uppercase.

\GLSsymbolplural

```
3805 \newrobustcmd*{\GLSsymbolplural}{\@gls@hyp@opt\@GLSsymbolplural}
```

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

```
3806 \newcommand*{\@GLSsymbolplural}[2] [] {%
3807   \new@ifnextchar[{\@GLSsymbolplural@{#1}{#2}}{\@GLSsymbolplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3808 \def\@GLSsymbolplural@#1#2[#3] {%
3809   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentrysymbolplural{#2}#3}}%
3810 }
```

\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

```
3811 \newrobustcmd*{\glsuseri}{\@gls@hyp@opt\@glsuseri}
```

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

```
3812 \newcommand*{\@glsuseri}[2] [] {%
3813   \new@ifnextchar[{\@glsuseri@{#1}{#2}}{\@glsuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3814 \def\@glsuseri@#1#2[#3] {%
3815   \@gls@field@link{#1}{#2}{\Glsentryuseri{#2}#3}%
3816 }
```

\Glsuseri behaves like \glsuseri except that the first letter is converted to uppercase.

\Glsuseri

```
3817 \newrobustcmd*{\Glsuseri}{\@gls@hyp@opt\@Glsuseri}
```

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

```
3818 \newcommand*{\@Glsuseri}[2] [] {%
3819   \new@ifnextchar[{\@Glsuseri@{#1}{#2}}{\@Glsuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3820 \def\@Glsuseri@#1#2[#3]{%
3821   \@gls@field@link{#1}{#2}{\Glsentryuseri{#2}#3}%
3822 }
```

\Glsuseri behaves like \glsuseri except that the link text is converted to uppercase.

\Glsuseri

```
3823 \newrobustcmd*{\Glsuseri}{\@gls@hyp@opt\@Glsuseri}
```

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

```
3824 \newcommand*{\@Glsuseri}[2][]{%
3825   \new@ifnextchar[{\@Glsuseri@{#1}{#2}}{\@Glsuseri@{#1}{#2}[]}]}
```

Read in the final optional argument:

```
3826 \def\@Glsuseri@#1#2[#3]{%
3827   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentryuseri{#2}#3}}%
3828 }
```

\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

```
3829 \newrobustcmd*{\glsuserii}{\@gls@hyp@opt\@glsuserii}
```

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

```
3830 \newcommand*{\@glsuserii}[2][]{%
3831   \new@ifnextchar[{\@glsuserii@{#1}{#2}}{\@glsuserii@{#1}{#2}[]}]}
```

Read in the final optional argument:

```
3832 \def\@glsuserii@#1#2[#3]{%
3833   \@gls@field@link{#1}{#2}{\Glsentryuserii{#2}#3}%
3834 }
```

\Glsuserii behaves like \glsuserii except that the first letter is converted to uppercase.

\Glsuserii

```
3835 \newrobustcmd*{\Glsuserii}{\@gls@hyp@opt\@Glsuserii}
```

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

```
3836 \newcommand*{\@Glsuserii}[2][]{%
3837   \new@ifnextchar[{\@Glsuserii@{#1}{#2}}{\@Glsuserii@{#1}{#2}[]}]}
```

Read in the final optional argument:

```
3838 \def\@Glsuserii@#1#2[#3]{%
3839   \@gls@field@link{#1}{#2}{\Glsentryuserii{#2}#3}%
3840 }
```

`\GLSuserii` behaves like `\glsuserii` except that the link text is converted to uppercase.

`\GLSuserii`

```
3841 \newrobustcmd*{\GLSuserii}{\@gls@hyp@opt\@GLSuserii}
```

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

```
3842 \newcommand*{\@GLSuserii}[2] [] {%
```

```
3843   \new@ifnextchar[{\@GLSuserii@{#1}{#2}}{\@GLSuserii@{#1}{#2} []}]
```

Read in the final optional argument:

```
3844 \def\@GLSuserii@#1#2[#3] {%
```

```
3845   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuserii{#2}#3}}%
```

```
3846 }
```

`\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`

```
3847 \newrobustcmd*{\glsuseriii}{\@gls@hyp@opt\@glsuseriii}
```

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

```
3848 \newcommand*{\@glsuseriii}[2] [] {%
```

```
3849   \new@ifnextchar[{\@glsuseriii@{#1}{#2}}{\@glsuseriii@{#1}{#2} []}]
```

Read in the final optional argument:

```
3850 \def\@glsuseriii@#1#2[#3] {%
```

```
3851   \@gls@field@link{#1}{#2}{\glsentryuseriii{#2}#3}}%
```

```
3852 }
```

`\Glsuseriii` behaves like `\glsuseriii` except that the first letter is converted to uppercase.

`\Glsuseriii`

```
3853 \newrobustcmd*{\Glsuseriii}{\@gls@hyp@opt\@Glsuseriii}
```

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

```
3854 \newcommand*{\@Glsuseriii}[2] [] {%
```

```
3855   \new@ifnextchar[{\@Glsuseriii@{#1}{#2}}{\@Glsuseriii@{#1}{#2} []}]
```

Read in the final optional argument:

```
3856 \def\@Glsuseriii@#1#2[#3] {%
```

```
3857   \@gls@field@link{#1}{#2}{\Glsentryuseriii{#2}#3}}%
```

```
3858 }
```

`\GLSuseriii` behaves like `\glsuseriii` except that the link text is converted to uppercase.

`\GLSuseriii`

```
3859 \newrobustcmd*{\GLSuseriii}{\@gls@hyp@opt\@GLSuseriii}
```

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

```
3860 \newcommand*{\@GLSuseriii}[2] [] {%
3861   \new@ifnextchar[{\@GLSuseriii@{#1}{#2}}{\@GLSuseriii@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3862 \def\@GLSuseriii@#1#2[#3] {%
3863   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseriii{#2}#3}}%
3864 }
```

\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

```
3865 \newrobustcmd*{\glsuseriv}{\@gls@hyp@opt\@glsuseriv}
```

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

```
3866 \newcommand*{\@glsuseriv}[2] [] {%
3867   \new@ifnextchar[{\@glsuseriv@{#1}{#2}}{\@glsuseriv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3868 \def\@glsuseriv@#1#2[#3] {%
3869   \@gls@field@link{#1}{#2}{\glsentryuseriv{#2}#3}%
3870 }
```

\Glsuseriv behaves like \glsuseriv except that the first letter is converted to uppercase.

\Glsuseriv

```
3871 \newrobustcmd*{\Glsuseriv}{\@gls@hyp@opt\@Glsuseriv}
```

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

```
3872 \newcommand*{\@Glsuseriv}[2] [] {%
3873   \new@ifnextchar[{\@Glsuseriv@{#1}{#2}}{\@Glsuseriv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3874 \def\@Glsuseriv@#1#2[#3] {%
3875   \@gls@field@link{#1}{#2}{\Glsentryuseriv{#2}#3}%
3876 }
```

\GLSuseriv behaves like \glsuseriv except that the link text is converted to uppercase.

\GLSuseriv

```
3877 \newrobustcmd*{\GLSuseriv}{\@gls@hyp@opt\@GLSuseriv}
```

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

```
3878 \newcommand*{\@GLSuseriv}[2] [] {%
3879   \new@ifnextchar[{\@GLSuseriv@{#1}{#2}}{\@GLSuseriv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3880 \def\@GLSuseriv@#1#2[#3]{%
3881   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseriv{#2}#3}}%
3882 }
```

`\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`

```
3883 \newrobustcmd*{\glsuserv}{\@gls@hyp@opt\@glsuserv}
```

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

```
3884 \newcommand*{\@glsuserv}[2][{}]{%
3885   \new@ifnextchar[{\@glsuserv@{#1}{#2}}{\@glsuserv@{#1}{#2}[]}}
```

Read in the final optional argument:

```
3886 \def\@glsuserv@#1#2[#3]{%
3887   \@gls@field@link{#1}{#2}{\glsentryuserv{#2}#3}%
3888 }
```

`\Glsuserv` behaves like `\glsuserv` except that the first letter is converted to uppercase.

`\Glsuserv`

```
3889 \newrobustcmd*{\Glsuserv}{\@gls@hyp@opt\@Glsuserv}
```

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

```
3890 \newcommand*{\@Glsuserv}[2][{}]{%
3891   \new@ifnextchar[{\@Glsuserv@{#1}{#2}}{\@Glsuserv@{#1}{#2}[]}}
```

Read in the final optional argument:

```
3892 \def\@Glsuserv@#1#2[#3]{%
3893   \@gls@field@link{#1}{#2}{\Glsentryuserv{#2}#3}%
3894 }
```

`\GLSuserv` behaves like `\glsuserv` except that the link text is converted to uppercase.

`\GLSuserv`

```
3895 \newrobustcmd*{\GLSuserv}{\@gls@hyp@opt\@GLSuserv}
```

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

```
3896 \newcommand*{\@GLSuserv}[2][{}]{%
3897   \new@ifnextchar[{\@GLSuserv@{#1}{#2}}{\@GLSuserv@{#1}{#2}[]}}
```

Read in the final optional argument:

```
3898 \def\@GLSuserv@#1#2[#3]{%
3899   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuserv{#2}#3}}%
3900 }
```


\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

```
3901 \newrobustcmd*{\glsuservi}{\@gls@hyp@opt\@glsuservi}
```

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

```
3902 \newcommand*{\@glsuservi}[2][]{\%
```

```
3903   \new@ifnextchar[{\@glsuservi@{#1}{#2}}{\@glsuservi@{#1}{#2}[]}}
```

Read in the final optional argument:

```
3904 \def\@glsuservi@#1#2[#3]{\%
```

```
3905   \@gls@field@link{#1}{#2}{\glsentryuservi{#2}#3}%
```

```
3906 }
```

\Glsuservi behaves like \glsuservi except that the first letter is converted to uppercase.

\Glsuservi

```
3907 \newrobustcmd*{\Glsuservi}{\@gls@hyp@opt\@Glsuservi}
```

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

```
3908 \newcommand*{\@Glsuservi}[2][]{\%
```

```
3909   \new@ifnextchar[{\@Glsuservi@{#1}{#2}}{\@Glsuservi@{#1}{#2}[]}}
```

Read in the final optional argument:

```
3910 \def\@Glsuservi@#1#2[#3]{\%
```

```
3911   \@gls@field@link{#1}{#2}{\Glsentryuservi{#2}#3}%
```

```
3912 }
```

\GLSuservi behaves like \glsuservi except that the link text is converted to uppercase.

\GLSuservi

```
3913 \newrobustcmd*{\GLSuservi}{\@gls@hyp@opt\@GLSuservi}
```

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

```
3914 \newcommand*{\@GLSuservi}[2][]{\%
```

```
3915   \new@ifnextchar[{\@GLSuservi@{#1}{#2}}{\@GLSuservi@{#1}{#2}[]}}
```

Read in the final optional argument:

```
3916 \def\@GLSuservi@#1#2[#3]{\%
```

```
3917   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuservi{#2}#3}}%
```

```
3918 }
```

Now deal with acronym related keys. First the short form:

\acrshort

```
3919 \newrobustcmd*{\acrshort}{\@gls@hyp@opt\@ns@acrshort}
```

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

```
3920 \newcommand*{\ns@acrshort}[2] [] {%
3921   \new@ifnextchar[{\@acrshort{#1}{#2}}{\@acrshort{#1}{#2} []}%
3922 }
```

Read in the final optional argument:

```
3923 \def\@acrshort#1#2[#3] {%
3924   \glsdoifexists{#2}%
3925   {%
3926     \let\do@gl@link@checkfirsthyper\relax
3927     \let\glsifplural\@secondoftwo
3928     \let\glscapscase\@firstofthree
3929     \let\glsinsert\@empty
3930     \def\glscustomtext{%
3931       \acronymfont{\glsentryshort{#2}}#3%
3932     }%
```

Call \@gl@link Note that \@gl@link sets \glstype.

```
3933   \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
3934   }%
3935   \glspostlinkhook
3936 }
```

\Acrshort

```
3937 \newrobustcmd*{\Acrshort}{\@gl@hyp@opt\ns@Acrshort}
```

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

```
3938 \newcommand*{\ns@Acrshort}[2] [] {%
3939   \new@ifnextchar[{\@Acrshort{#1}{#2}}{\@Acrshort{#1}{#2} []}%
3940 }
```

Read in the final optional argument:

```
3941 \def\@Acrshort#1#2[#3] {%
3942   \glsdoifexists{#2}%
3943   {%
3944     \let\do@gl@link@checkfirsthyper\relax
3945     \def\glslabel{#2}%
3946     \let\glsifplural\@secondoftwo
3947     \let\glscapscase\@secondofthree
3948     \let\glsinsert\@empty
3949     \def\glscustomtext{%
3950       \acronymfont{\Glsentryshort{#2}}#3%
3951     }%
```

Call \@gl@link Note that \@gl@link sets \glstype.

```
3952   \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
3953   }%
```

```

3954 \glspostlinkhook
3955 }

```

\ACRshort

```

3956 \newrobustcmd*{\ACRshort}{\@gls@hyp@opt\ns@ACRshort}

```

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

```

3957 \newcommand*{\ns@ACRshort}[2] [] {%
3958   \new@ifnextchar[{\@ACRshort{#1}{#2}}{\@ACRshort{#1}{#2} []}%
3959 }

```

Read in the final optional argument:

```

3960 \def\@ACRshort#1#2[#3] {%
3961   \glsdoidexists{#2}%
3962   {%
3963     \let\do@gls@link@checkfirsthyper\relax

3964     \def\glslabel{#2}%
3965     \let\glsifplural\@secondoftwo
3966     \let\glscapscase\@thirdofthree
3967     \let\glsinsert\@empty
3968     \def\glscustomtext{%
3969       \mfirstucMakeUppercase{\acronymfont{\glsentryshort{#2}}#3}%
3970     }%

```

Call \@gls@link Note that \@gls@link sets \glstype.

```

3971   \@gls@link[#1]{#2}{\cename gls@\glstype @entryfmt\endcsname}%
3972   }%

3973   \glspostlinkhook
3974 }

```

Short plural:

\acrshortpl

```

3975 \newrobustcmd*{\acrshortpl}{\@gls@hyp@opt\ns@acrshortpl}

```

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

```

3976 \newcommand*{\ns@acrshortpl}[2] [] {%
3977   \new@ifnextchar[{\@acrshortpl{#1}{#2}}{\@acrshortpl{#1}{#2} []}%
3978 }

```

Read in the final optional argument:

```

3979 \def\@acrshortpl#1#2[#3] {%
3980   \glsdoidexists{#2}%
3981   {%
3982     \let\do@gls@link@checkfirsthyper\relax

```

```

3983 \def\glslabel{#2}%
3984 \let\glsifplural\@firstoftwo
3985 \let\glscapscase\@firstofthree
3986 \let\glsinsert\@empty
3987 \def\glscustomtext{%
3988 \acronymfont{\glsentryshortpl{#2}}#3%
3989 }%

Call \@gls@link Note that \@gls@link sets \glstype.
3990 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3991 }%

3992 \glspostlinkhook
3993 }

```

\Acrshortpl

```

3994 \newrobustcmd*{\Acrshortpl}{\@gls@hyp@opt\ns@Acrshortpl}

```

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

```

3995 \newcommand*{\ns@Acrshortpl}[2] [] {%
3996 \new@ifnextchar[{\@Acrshortpl{#1}{#2}}{\@Acrshortpl{#1}{#2} []}%
3997 }

```

Read in the final optional argument:

```

3998 \def\@Acrshortpl#1#2[#3] {%
3999 \glsdoifexists{#2}%
4000 {%
4001 \let\do@gls@link@checkfirsthyper\relax

4002 \def\glslabel{#2}%
4003 \let\glsifplural\@firstoftwo
4004 \let\glscapscase\@secondofthree
4005 \let\glsinsert\@empty
4006 \def\glscustomtext{%
4007 \acronymfont{\Glsentryshortpl{#2}}#3%
4008 }%

```

Call \@gls@link Note that \@gls@link sets \glstype.

```

4009 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4010 }%

4011 \glspostlinkhook
4012 }

```

\ACRshortpl

```

4013 \newrobustcmd*{\ACRshortpl}{\@gls@hyp@opt\ns@ACRshortpl}

```

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

```

4014 \newcommand*{\ns@ACRshortpl}[2][{}]{%
4015   \new@ifnextchar[{\@ACRshortpl{#1}{#2}}{\@ACRshortpl{#1}{#2}[]}%
4016 }

  Read in the final optional argument:
4017 \def\@ACRshortpl#1#2[#3]{%
4018   \glsdoifexists{#2}%
4019   {%
4020     \let\do@gl@link@checkfirsthyper\relax

4021     \def\glslabel{#2}%
4022     \let\glsifplural\@firstoftwo
4023     \let\glscapscase\@thirdofthree
4024     \let\glsinsert\@empty
4025     \def\glscustomtext{%
4026       \mfirstucMakeUppercase{\acronymfont{\glentryshortpl{#2}}#3}%
4027     }%

    Call \@gl@link Note that \@gl@link sets \glstype.
4028     \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
4029   }%

4030   \glspostlinkhook
4031 }

```

\acrlong

```

4032 \newrobustcmd*{\acrlong}{\@gl@hyp@opt\ns@acrlong}

```

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

```

4033 \newcommand*{\ns@acrlong}[2][{}]{%
4034   \new@ifnextchar[{\@acrlong{#1}{#2}}{\@acrlong{#1}{#2}[]}%
4035 }

```

Read in the final optional argument:

```

4036 \def\@acrlong#1#2[#3]{%
4037   \glsdoifexists{#2}%
4038   {%
4039     \let\do@gl@link@checkfirsthyper\relax

4040     \def\glslabel{#2}%
4041     \let\glsifplural\@secondoftwo
4042     \let\glscapscase\@firstofthree
4043     \let\glsinsert\@empty

```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```

4044     \def\glscustomtext{%
4045       \glentrylong{#2}#3%
4046     }%

```

Call \@gls@link Note that \@gls@link sets \glstype.

```
4047   \@gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%  
4048   }%  
  
4049   \glspostlinkhook  
4050 }
```

\Acrlong

```
4051 \newrobustcmd*{\Acrlong}{\@gls@hyp@opt\ns@Acrlong}
```

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

```
4052 \newcommand*{\ns@Acrlong}[2][]{%  
4053   \new@ifnextchar[{\@Acrlong{#1}{#2}}{\@Acrlong{#1}{#2} []}%  
4054 }
```

Read in the final optional argument:

```
4055 \def\@Acrlong#1#2[#3]{%  
4056   \glsdoifexists{#2}%  
4057   {%  
4058     \let\do@gls@link@checkfirsthyper\relax  
  
4059     \def\glslabel{#2}%  
4060     \let\glsifplural\@secondoftwo  
4061     \let\glscapscase\@secondofthree  
4062     \let\glsinsert\@empty
```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```
4063   \def\glscustomtext{%  
4064     \Glsentrylong{#2}#3%  
4065   }%
```

Call \@gls@link. Note that \@gls@link sets \glstype.

```
4066   \@gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%  
4067   }%  
  
4068   \glspostlinkhook  
4069 }
```

\ACRlong

```
4070 \newrobustcmd*{\ACRlong}{\@gls@hyp@opt\ns@ACRlong}
```

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

```
4071 \newcommand*{\ns@ACRlong}[2][]{%  
4072   \new@ifnextchar[{\@ACRlong{#1}{#2}}{\@ACRlong{#1}{#2} []}%  
4073 }
```

Read in the final optional argument:

```
4074 \def\@ACRlong#1#2[#3]{%
4075   \glsdoifexists{#2}%
4076   {%
4077     \let\do@gl@link@checkfirsthyper\relax

4078   \def\glslabel{#2}%
4079   \let\glsifplural\@secondoftwo
4080   \let\glscapscase\@thirdofthree
4081   \let\glsinsert\@empty
```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```
4082   \def\glscustomtext{%
4083     \mfirstucMakeUppercase{\glsentrylong{#2}#3}%
4084   }%
```

Call \@gl@link. Note that \@gl@link sets \glstype.

```
4085   \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
4086   }%

4087   \glspostlinkhook
4088 }
```

Short plural:

\acrlongpl

```
4089 \newrobustcmd*{\acrlongpl}{\@gl@hyp@opt\@ns@acrlongpl}
```

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

```
4090 \newcommand*{\ns@acrlongpl}[2][ ]{%
4091   \new@ifnextchar[{\@acrlongpl{#1}{#2}}{\@acrlongpl{#1}{#2}[ ]}%
4092 }
```

Read in the final optional argument:

```
4093 \def\@acrlongpl#1#2[#3]{%
4094   \glsdoifexists{#2}%
4095   {%
4096     \let\do@gl@link@checkfirsthyper\relax

4097   \def\glslabel{#2}%
4098   \let\glsifplural\@firstoftwo
4099   \let\glscapscase\@firstofthree
4100   \let\glsinsert\@empty
```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```
4101   \def\glscustomtext{%
4102     \glsentrylongpl{#2}#3%
4103   }%
```

Call `\@gls@link`. Note that `\@gls@link` sets `\glstype`.

```
4104 \gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
4105 }%

4106 \glspostlinkhook
4107 }
```

`\Acrlongpl`

```
4108 \newrobustcmd*{\Acrlongpl}{\@gls@hyp@opt\ns@Acrlongpl}
```

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

```
4109 \newcommand*{\ns@Acrlongpl}[2] [] {%
4110 \new@ifnextchar[{\@Acrlongpl{#1}{#2}}{\@Acrlongpl{#1}{#2} []}%
4111 }
```

Read in the final optional argument:

```
4112 \def\@Acrlongpl#1#2[#3] {%
4113 \glsdoifexists{#2}%
4114 {%
4115 \let\do@gls@link@checkfirsthyper\relax

4116 \def\glslabel{#2}%
4117 \let\glsifplural\@firstoftwo
4118 \let\glscapscase\@secondofthree
4119 \let\glsinsert\@empty
```

Bug fix v4.02 removed `\acronymfont` from `\glscustomtext` (`\acronymfont` only designed for short form).

```
4120 \def\glscustomtext{%
4121 \Glsentrylongpl{#2}#3%
4122 }%
```

Call `\@gls@link`. Note that `\@gls@link` sets `\glstype`.

```
4123 \gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
4124 }%

4125 \glspostlinkhook
4126 }
```

`\ACRlongpl`

```
4127 \newrobustcmd*{\ACRlongpl}{\@gls@hyp@opt\ns@ACRlongpl}
```

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

```
4128 \newcommand*{\ns@ACRlongpl}[2] [] {%
4129 \new@ifnextchar[{\@ACRlongpl{#1}{#2}}{\@ACRlongpl{#1}{#2} []}%
4130 }
```


Read in the final optional argument:

```

4131 \def\@ACRlongpl#1#2[#3]{%
4132   \glsdoifexists{#2}%
4133   {%
4134     \let\do@gl@link@checkfirsthyper\relax

4135   \def\glslabel{#2}%
4136   \let\glsifplural\@firstoftwo
4137   \let\glscapscase\@thirdofthree
4138   \let\glsinsert\@empty

```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```

4139   \def\glscustomtext{%
4140     \mfirstucMakeUppercase{\glsentrylongpl{#2}#3}%
4141   }%

```

Call \@gl@link. Note that \@gl@link sets \glstype.

```

4142   \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
4143   }%

4144   \glspostlinkhook
4145 }

```

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.

\@gl@entry@field Generic version.

```
\@gl@entry@field{<label>}{<field>}
```

```

4146 \newcommand*{\@gl@entry@field}[2]{%
4147   \csname glo@\glsdetoklabel{#1}@#2\endcsname
4148 }

```

\glsletentryfield \glsletentryfield{<cs>}{<label>}{<field>}

```

4149 \newcommand*{\glsletentryfield}[3]{%
4150   \letcs{#1}{glo@\glsdetoklabel{#2}@#3}%
4151 }

```

\@Gls@entry@field Generic first letter uppercase version.

```
\@Gls@entry@field{<label>}{<field>}
```

```

4152 \newcommand*{\@Gls@entry@field}[2]{%
4153   \letcs\@glo@text{glo@\glsdetoklabel{#1}@#2}%
4154   \ifdef\@glo@text
4155   {%
4156     \xmakefirstuc{\@glo@text}%
4157   }%
4158   {%
4159     \PackageError{glossaries}{Either glossary entry
4160       '\glsdetoklabel{#1}' doesn't exist or the field '#2'
4161       doesn't exist}{Check you have correctly spelt the entry
4162       label and the field name}%
4163   }%
4164 }

```

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`

```

4165 \newcommand*{\glsentryname}[1]{\@Gls@entry@field{#1}{name}}

```

`\Glsentryname`

```

4166 \newrobustcmd*{\Glsentryname}[1]{%
4167   \@Gls@entryname{#1}%
4168 }

```

`\@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:

```

4169 \newcommand*{\@Gls@entryname}[1]{%
4170   \@Gls@entry@field{#1}{name}%
4171 }

```

`\@Gls@acrentryname` Now the behaviour when `\setacronymstyle` is used:

```

4172 \newcommand*{\@Gls@acrentryname}[1]{%
4173   \ifglshaslong{#1}%
4174   {%
4175     \letcs\@glo@text{glo@\glsdetoklabel{#1}@name}%
4176     \expandafter\@gls@getbody\@glo@text{}\@nil
4177     \expandafter\ifx\@gls@body\glsentrylong\relax
4178     \expandafter\Glsentrylong\@gls@rest
4179   }%
4180   \else
4181     \expandafter\ifx\@gls@body\glsentryshort\relax
4182     \expandafter\Glsentryshort\@gls@rest
4183   }%
4184   \else
4185     \expandafter\ifx\@gls@body\acronymfont\relax

```

Temporarily make `\glsentryshort` behave like `\Glsentryshort`. (This is on the assumption that the argument of `\acronymfont` is `\glsentryshort{<label>}`, as that's the behaviour of the predefined acronym styles.) This is scoped to localise the effect of the assignment.

```

4184      {%
4185          \let\glsentryshort\Glsentryshort
4186          \@glo@text
4187      }%
4188      \else
4189          \xmakefirstuc{\@glo@text}%
4190      \fi
4191  \fi
4192  \fi
4193  }%
4194  {%

```

Not an acronym

```

4195      \@Gls@entry@field{#1}{name}%
4196  }%
4197 }

```

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`

```

4198 \newcommand*{\glsentrydesc}[1]{\@gls@entry@field{#1}{desc}}

```

`\Glsentrydesc`

```

4199 \newrobustcmd*{\Glsentrydesc}[1]{%
4200   \@Gls@entry@field{#1}{desc}%
4201 }

```

Plural form:

`\glsentrydescplural`

```

4202 \newcommand*{\glsentrydescplural}[1]{%
4203   \@gls@entry@field{#1}{descplural}%
4204 }

```

`\Glsentrydescplural`

```

4205 \newrobustcmd*{\Glsentrydescplural}[1]{%
4206   \@Gls@entry@field{#1}{descplural}%
4207 }

```

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
4208 \newcommand*{\glsentrytext}[1]{\@gls@entry@field{#1}{text}}
```

```
\Glsentrytext
4209 \newrobustcmd*{\Glsentrytext}[1]{%
4210   \@Gls@entry@field{#1}{text}%
4211 }
```

Get the plural form:

```
\glsentryplural
4212 \newcommand*{\glsentryplural}[1]{%
4213   \@gls@entry@field{#1}{plural}%
4214 }
```

```
\Glsentryplural
4215 \newrobustcmd*{\Glsentryplural}[1]{%
4216   \@Gls@entry@field{#1}{plural}%
4217 }
```

Get the symbol associated with this entry. The argument is the label associated with the entry.

```
\glsentrysymbol
4218 \newcommand*{\glsentrysymbol}[1]{%
4219   \@gls@entry@field{#1}{symbol}%
4220 }
```

```
\Glsentrysymbol
4221 \newrobustcmd*{\Glsentrysymbol}[1]{%
4222   \@Gls@entry@field{#1}{symbol}%
4223 }
```

Plural form:

```
\glsentrysymbolplural
4224 \newcommand*{\glsentrysymbolplural}[1]{%
4225   \@gls@entry@field{#1}{symbolplural}%
4226 }
```

```
\Glsentrysymbolplural
4227 \newrobustcmd*{\Glsentrysymbolplural}[1]{%
4228   \@Gls@entry@field{#1}{symbolplural}%
4229 }
```

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`

```
4230 \newcommand*{\glsentryfirst}[1]{%
4231   \@gls@entry@field{#1}{first}%
4232 }
```

`\Glsentryfirst`

```
4233 \newrobustcmd*{\Glsentryfirst}[1]{%
4234   \@Gls@entry@field{#1}{first}%
4235 }
```

Get the plural form (as specified by the `firstplural` key when the entry was defined).

`\glsentryfirstplural`

```
4236 \newcommand*{\glsentryfirstplural}[1]{%
4237   \@gls@entry@field{#1}{firstpl}%
4238 }
```

`\Glsentryfirstplural`

```
4239 \newrobustcmd*{\Glsentryfirstplural}[1]{%
4240   \@Gls@entry@field{#1}{firstpl}%
4241 }
```

Display the glossary type with which this entry is associated (as specified by the `type` key used when the entry was defined)

`\glsentrytype`

```
4242 \newcommand*{\glsentrytype}[1]{\@gls@entry@field{#1}{type}}
```

Display the sort text used for this entry. Note that the sort key is sanitized, so unexpected results may occur if the sort key contained commands.

`\glsentrysort`

```
4243 \newcommand*{\glsentrysort}[1]{%
4244   \@gls@entry@field{#1}{sort}%
4245 }
```

`\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.

```
4246 \newcommand*{\glsentryuseri}[1]{%
4247   \@gls@entry@field{#1}{useri}%
4248 }
```

`\Glsentryuseri`

```
4249 \newrobustcmd*{\Glsentryuseri}[1]{%
4250   \@Gls@entry@field{#1}{useri}%
4251 }
```

`\glentryuserii` Get the second user key (as specified by the user2 when the entry was defined).
The argument is the label associated with the entry.

```

4252 \newcommand*{\glentryuserii}[1]{%
4253   \@gls@entry@field{#1}{userii}%
4254 }

```

`\Glsentryuserii`

```

4255 \newrobustcmd*{\Glsentryuserii}[1]{%
4256   \@Gls@entry@field{#1}{userii}%
4257 }

```

`\glentryuseriii` Get the third user key (as specified by the user3 when the entry was defined).
The argument is the label associated with the entry.

```

4258 \newcommand*{\glentryuseriii}[1]{%
4259   \@gls@entry@field{#1}{useriii}%
4260 }

```

`\Glsentryuseriii`

```

4261 \newrobustcmd*{\Glsentryuseriii}[1]{%
4262   \@Gls@entry@field{#1}{useriii}%
4263 }

```

`\glentryuseriv` Get the fourth user key (as specified by the user4 when the entry was defined).
The argument is the label associated with the entry.

```

4264 \newcommand*{\glentryuseriv}[1]{%
4265   \@gls@entry@field{#1}{useriv}%
4266 }

```

`\Glsentryuseriv`

```

4267 \newrobustcmd*{\Glsentryuseriv}[1]{%
4268   \@Gls@entry@field{#1}{useriv}%
4269 }

```

`\glentryuserv` Get the fifth user key (as specified by the user5 when the entry was defined).
The argument is the label associated with the entry.

```

4270 \newcommand*{\glentryuserv}[1]{%
4271   \@gls@entry@field{#1}{userv}%
4272 }

```

`\Glsentryuserv`

```

4273 \newrobustcmd*{\Glsentryuserv}[1]{%
4274   \@Gls@entry@field{#1}{userv}%
4275 }

```

`\glentryuservi` Get the sixth user key (as specified by the user6 when the entry was defined).
The argument is the label associated with the entry.

```

4276 \newcommand*{\glentryuservi}[1]{%
4277   \@gls@entry@field{#1}{uservi}%
4278 }

```

`\Glsentryuservi`

```
4279 \newrobustcmd*{\Glsentryuservi}[1]{%
4280   \@Gls@entry@field{#1}{uservi}%
4281 }
```

`\glsentryshort` Get the short key (as specified by the short the entry was defined). The argument is the label associated with the entry.

```
4282 \newcommand*{\glsentryshort}[1]{\@Gls@entry@field{#1}{short}}
```

`\Glsentryshort`

```
4283 \newrobustcmd*{\Glsentryshort}[1]{%
4284   \@Gls@entry@field{#1}{short}%
4285 }
```

`\glsentryshortpl` Get the short plural key (as specified by the shortplural the entry was defined). The argument is the label associated with the entry.

```
4286 \newcommand*{\glsentryshortpl}[1]{\@Gls@entry@field{#1}{shortpl}}
```

`\Glsentryshortpl`

```
4287 \newrobustcmd*{\Glsentryshortpl}[1]{%
4288   \@Gls@entry@field{#1}{shortpl}%
4289 }
```

`\glsentrylong` Get the long key (as specified by the long the entry was defined). The argument is the label associated with the entry.

```
4290 \newcommand*{\glsentrylong}[1]{\@Gls@entry@field{#1}{long}}
```

`\Glsentrylong`

```
4291 \newrobustcmd*{\Glsentrylong}[1]{%
4292   \@Gls@entry@field{#1}{long}%
4293 }
```

`\glsentrylongpl` Get the long plural key (as specified by the longplural the entry was defined). The argument is the label associated with the entry.

```
4294 \newcommand*{\glsentrylongpl}[1]{\@Gls@entry@field{#1}{longpl}}
```

`\Glsentrylongpl`

```
4295 \newrobustcmd*{\Glsentrylongpl}[1]{%
4296   \@Gls@entry@field{#1}{longpl}%
4297 }
```

Short cut macros to access full form:

`\glsentryfull`

```
4298 \newcommand*{\glsentryfull}[1]{%
4299   \acrfullformat{\glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%
4300 }
```

```

\Glsentryfull
4301 \newrobustcmd*{\Glsentryfull}[1]{%
4302   \acrfullformat{\Glsentrylong{#1}}{\acronymfont{\glentryshort{#1}}}%
4303 }

\glentryfullpl
4304 \newcommand*{\glentryfullpl}[1]{%
4305   \acrfullformat{\glentrylongpl{#1}}{\acronymfont{\glentryshortpl{#1}}}%
4306 }

\Glsentryfullpl
4307 \newrobustcmd*{\Glsentryfullpl}[1]{%
4308   \acrfullformat{\Glsentrylongpl{#1}}{\acronymfont{\glentryshortpl{#1}}}%
4309 }

\glentrynumberlist  Displays the number list as is.
4310 \newcommand*{\glentrynumberlist}[1]{%
4311   \glsdoifexists{#1}%
4312   {%
4313     \@gls@entry@field{#1}{numberlist}%
4314   }%
4315 }

\glsdisplaynumberlist  Formats the number list for the given entry label. Doesn't work with hyperref.
4316 \@ifpackageloaded{hyperref} {%
4317   \newcommand*{\glsdisplaynumberlist}[1]{%
4318     \GlossariesWarning
4319     {%
4320       \string\glsdisplaynumberlist\space
4321       doesn't work with hyperref.^^JUsing
4322       \string\glentrynumberlist\space instead%
4323     }%
4324     \glentrynumberlist{#1}%
4325   }%
4326 }%
4327 {%
4328   \newcommand*{\glsdisplaynumberlist}[1]{%
4329     \glsdoifexists{#1}%
4330     {%
4331       \bgroup
4332
4333       \edef\@glo@label{\glsdetoklabel{#1}}%
4334       \let\@org@glnumberformat\glnumberformat
4335       \def\glnumberformat##1{##1}%
4336       \protected@edef\the@numberlist{%
4337         \csname glo@\@glo@label @numberlist\endcsname}%
4338       \def\@gls@numlist@sep{}%
4339       \def\@gls@numlist@nextsep{}%
4340       \def\@gls@numlist@lastsep{}%

```



```

4340      \def\@gls@thislist{}%
4341      \def\@gls@donext@def{}%
4342      \renewcommand\do[1]{%
4343        \protected@edef\@gls@thislist{%
4344          \@gls@thislist
4345          \noexpand\@gls@numlist@sep
4346          ##1%
4347        }%
4348        \let\@gls@numlist@sep\@gls@numlist@nextsep
4349        \def\@gls@numlist@nextsep{\glsnumlistsep}%
4350        \@gls@donext@def
4351        \def\@gls@donext@def{%
4352          \def\@gls@numlist@lastsep{\glsnumlistlastsep}%
4353        }%
4354      }%
4355      \expandafter \glsnumlistparser \expandafter{\the@numberlist}%
4356      \let\@gls@numlist@sep\@gls@numlist@lastsep
4357      \@gls@thislist
4358    \egroup
4359  }%
4360 }
4361 }

```

`\glsnumlistsep`

```
4362 \newcommand*{\glsnumlistsep}{, }
```

`\glsnumlistlastsep`

```
4363 \newcommand*{\glsnumlistlastsep}{ \& }
```

`\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.

```

4364 \newcommand*{\glshyperlink}[2][\glsentrytext{\@glo@label}]{%
4365   \def\@glo@label{#2}%
4366   \@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`:

```
4367 \define@key{glossadd}{counter}{\def\@gls@counter{#1}}
```

```
4368 \define@key{glossadd}{format}{\def\@glsnumberformat{#1}}
```

This key is only used by `\glsaddall`:

```
4369 \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`

```
4370 \newrobustcmd*{\glsadd}[2] [] {%
```

Need to move to horizontal mode if not already in it, but only if not in preamble.

```
4371 \@gls@adjustmode
```

```
4372 \glsdoifexists{#2}%
```

```
4373 {%
```

```
4374 \def\@glsnumberformat{\glsnumberformat}%
```

```
4375 \edef\@gls@counter{\csname glo@glsetoklabel{#2}@counter\endcsname}%
```

```
4376 \setkeys{glossadd}{#1}%
```

Store the entry's counter in `\theglentrycounter`

```
4377 \@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.

```
4378 \@@do@wrglossary{#2}%
```

```
4379 }%
```

```
4380 }
```

`\@gls@adjustmode`

```
4381 \newcommand*{\@gls@adjustmode}{}%
```

```
4382 \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`

```
4383 \newrobustcmd*{\glsaddall}[1] [] {%
```

```
4384 \edef\@glo@type{\@glo@types}%
```

```
4385 \setkeys{glossadd}{#1}%
```

```
4386 \forallglsentries[\@glo@type]{\@glo@entry}{%
```

```
4387 \glsadd[#1]{\@glo@entry}%
```

```
4388 }%
```

```
4389 }
```

`\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.

```
4390 \newrobustcmd*{\glsaddallunused}[1][\@glo@types]{%
4391 \forallglsentries[#1]{\@glo@entry}%
4392 {%
4393     \ifglsused{\@glo@entry}{\@glsadd[format=glsignore]{\@glo@entry}}%
4394 }%
4395 }
```

`\glsignore`

```
4396 \newcommand*{\glsignore}[1]{}
```

1.13 Creating associated files

The `\writeist` command creates the associated customized `.ist` `makeindex` style file. While defining this command, some characters have their catcodes temporarily changed to ensure they get written to the `.ist` file correctly. The `makeindex` actual character (usually `@`) is redefined to be a `?`, to allow internal commands to be written to the glossary file output file.

The special characters are stored in `\@gls@actualchar`, `\@gls@encapchar`, `\@gls@levelchar` and `\@gls@quotechar` to make them easier to use later, but don't change these values, because the characters are encoded in the command definitions that are used to escape the special characters (which means that the user no longer needs to worry about `makeindex` special characters).

The symbols and numbers label for group headings are hardwired into the `.ist` file as `glsymbols` and `glsnumbers`, the group titles can be translated (so that `\glsymbolsgroupname` replaces `glsymbols` and `\glsnumbersgroupname` replaces `glsnumbers`) using the command `\glsgetgrouptitle` which is defined in `.` This is done to prevent any problem characters in `\glsymbolsgroupname` and `\glsnumbersgroupname` from breaking hyperlinks.

`\glsopenbrace` Define `\glsopenbrace` to make it easier to write an opening brace to a file.

```
4397 \edef\glsopenbrace{\expandafter\@gobble\string\{}
```

`\glsclosebrace` Define `\glsclosebrace` to make it easier to write an opening brace to a file.

```
4398 \edef\glsclosebrace{\expandafter\@gobble\string\}}
```

`\glsbackslash` Define `\glsbackslash` to make it easier to write a backslash to a file.

```
4399 \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.

```
4400 \edef\glsquote#1{\string"#1\string"}
```

`\glspercentchar` Define `\glspercentchar` to make it easier to write a percent character to a file.

```
4401 \edef\glspercentchar{\expandafter\@gobble\string\%}
```

`\glstildechar` Define `\glstildechar` to make it easier to write a tilde character to a file.

```
4402 \edef\glstildechar{\string~}
```

`\@glsfirstletter` Define the first letter to come after the digits 0,...,9. Only required for xindy.

```
4403 \ifglsxindy
4404   \newcommand*{\@glsfirstletter}{A}
4405 \fi
```

`\GlsSetXdyFirstLetterAfterDigits` Sets the first letter to come after the digits 0,...,9.

```
4406 \ifglsxindy
4407   \newcommand*{\GlsSetXdyFirstLetterAfterDigits}[1]{%
4408     \renewcommand*{\@glsfirstletter}{#1}}
4409 \else
4410   \newcommand*{\GlsSetXdyFirstLetterAfterDigits}[1]{%
4411     \glsnoxywarning\GlsSetXdyFirstLetterAfterDigits}
4412 \fi
```

`\@glsminrange` Define the minimum number of successive location references to merge into a range.

```
4413 \newcommand*{\@glsminrange}{2}
```

`\GlsSetXdyMinRangeLength` 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.

```
4414 \ifglsxindy
4415   \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
4416     \renewcommand*{\@glsminrange}{#1}}
4417 \else
4418   \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
4419     \glsnoxywarning\GlsSetXdyMinRangeLength}
4420 \fi
```

`\writeist`

```
4421 \ifglsxindy
  Code to use if xindy is required.
4422   \def\writeist{%
    Define write register if not already defined
4423     \ifundef{\glswrite}{\newwrite\glswrite}{}%
    Update attributes list
4424     \@gls@addpredefinedattributes
    Open the file.
4425     \openout\glswrite=\istfilename
```

Write header comment at the start of the file

```
4426 \write\glswrite{;; xindy style file created by the glossaries
4427 package}%
4428 \write\glswrite{;; for document '\jobname' on
4429 \the\year-\the\month-\the\day}%
```

Specify the required styles

```
4430 \write\glswrite{^^J; required styles^^J}
4431 \@for\@xdystyle:=\@xdyrequiredstyles\do{%
4432 \ifx\@xdystyle\@empty
4433 \else
4434 \protected@write\glswrite{}{(require
4435 \string"\@xdystyle.xdy\string")}%
4436 \fi
4437 }%
```

List the allowed attributes (possible values used by the format key)

```
4438 \write\glswrite{^^J%
4439 ; list of allowed attributes (number formats)^^J}%
4440 \write\glswrite{(define-attributes ((\@xdyattributes)))}%
```

Define any additional alphabets

```
4441 \write\glswrite{^^J; user defined alphabets^^J}%
4442 \write\glswrite{\@xdyuseralphabets}%
```

Define location classes.

```
4443 \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.

```
4444 \@for\@gls@classI:=\@gls@xdy@locationlist\do{%
```

Case were $\langle Hprefix \rangle$ is empty:

```
4445 \protected@write\glswrite{}{(define-location-class
4446 \string"\@gls@classI\string"^^J\space\space\space
4447 (
4448 :sep "{ }{"
4449 \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
4450 :sep "}"
4451 )
4452 ^^J\space\space\space
4453 :min-range-length \@glsminrange^^J%
4454 )
4455 }%
```

Nested iteration over all classes:

```
4456 {%
4457 \@for\@gls@classII:=\@gls@xdy@locationlist\do{%
4458 \protected@write\glswrite{}{(define-location-class
4459 \string"\@gls@classII-\@gls@classI\string"
4460 ^^J\space\space\space
4461 (
```

```

4462         :sep "{"
4463         \csname @gls@xdy@Lclass@\@gls@classII\endcsname\space
4464         :sep "{}"
4465         \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
4466         :sep "}"
4467     )
4468     ^^J\space\space\space
4469     :min-range-length \@glsminrange^^J%
4470 )
4471 }%
4472 }%
4473 }%
4474 }%

```

User defined location classes (needs checking for new location format).

```

4475 \write\glswrite{^^J; user defined location classes}%
4476 \write\glswrite{\@xdyuserlocationdefs}%

```

Cross-reference class. (The unverified option is used as the cross-references are supplied using the list of labels along with the optional argument for `\glsseeformat` which xindy won't recognise.)

```

4477 \write\glswrite{^^J; define cross-reference class^^J}%
4478 \write\glswrite{(define-crossref-class \string"see\string"
4479     :unverified )}%

```

Define how cross-references should be displayed. This adds an empty set of braces after the cross-referencing information allowing for the final argument of `\glsseeformat` which gets ignored. (When using `makeindex` this final argument contains the location information which is not required.)

```

4480 \write\glswrite{(markup-crossref-list
4481     :class \string"see\string"^^J\space\space\space
4482     :open \string"\string\glsseeformat\string"
4483     :close \string"{}\string")}%

```

List the order to sort the classes.

```

4484 \write\glswrite{^^J; define the order of the location classes}%
4485 \write\glswrite{(define-location-class-order
4486     (\@xdylocationclassorder))}%

```

Specify what to write to the start and end of the glossary file.

```

4487 \write\glswrite{^^J; define the glossary markup^^J}%

4488 \write\glswrite{(markup-index^^J\space\space\space
4489     :open \string"\string
4490     \glossarysection[\string\glossarytoctitle]{\string
4491     \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`)

```

4492 \@for\@this@ctr:=\@xdycounters\do{%
4493     {%

```

```

4494         \@for\@this@attr:=\@xdyattributelist\do{%
4495             \protected@write\glswrite{\{\string\providecommand*%
4496                 \expandafter\string
4497                 \csname glsX\@this@ctr X\@this@attr\endcsname[2]%
4498                 {%
4499                     \string\setentrycounter
4500                     [\expandafter\@gobble\string\#1]{\@this@ctr}%
4501                     \expandafter\string
4502                     \csname\@this@attr\endcsname
4503                     {\expandafter\@gobble\string\#2}%
4504                 }%
4505             }%
4506         }%
4507     }%
4508 }%

```

Add the end part of the open tag and the rest of the markup-index information:

```

4509     \write\glswrite{%
4510         \string\begin
4511         {theglossary}\string\glossaryheader\glstildechar n\string" ^^J\space
4512         \space\space:close \string"glsperscentchar\glstildechar n\string
4513         \end{theglossary}\string\glossarypostamble
4514         \glstildechar n\string" ^^J\space\space\space
4515         :tree)}}%

```

Specify what to put between letter groups

```

4516     \write\glswrite{(markup-letter-group-list
4517         :sep \string"\string\glsgroupskip\glstildechar n\string"))}%

```

Specify what to put between entries

```

4518     \write\glswrite{(markup-indexentry
4519         :open \string"\string\relax \string\glsresetentrylist
4520         \glstildechar n\string"))}%

```

Specify how to format entries

```

4521     \write\glswrite{(markup-locclass-list :open
4522         \string"\glsopenbrace\string\glossaryentrynumbers
4523         \glsopenbrace\string\relax\space \string"^^J\space\space\space
4524         :sep \string", \string"
4525         :close \string"\glsclosebrace\glsclosebrace\string"))}%

```

Specify how to separate location numbers

```

4526     \write\glswrite{(markup-locref-list
4527         :sep \string"\string\delimN\space\string"))}%

```

Specify how to indicate location ranges

```

4528     \write\glswrite{(markup-range
4529         :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.

```

4530     \@onelevel@sanitize\gls@suffixF
4531     \@onelevel@sanitize\gls@suffixFF

```

```

4532 \ifx\gls@suffixF\@empty
4533 \else
4534 \write\glswrite{(markup-range
4535 :close "\gls@suffixF" :length 1 :ignore-end)}}%
4536 \fi
4537 \ifx\gls@suffixFF\@empty
4538 \else
4539 \write\glswrite{(markup-range
4540 :close "\gls@suffixFF" :length 2 :ignore-end)}}%
4541 \fi

```

Specify how to format locations.

```

4542 \write\glswrite{^^J; define format to use for locations^^J}%
4543 \write\glswrite{\@xdylocref}%

```

Specify how to separate letter groups.

```

4544 \write\glswrite{^^J; define letter group list format^^J}%
4545 \write\glswrite{(markup-letter-group-list
4546 :sep \string"\string\glsgroupskip\glstildechar n\string")}%

```

Define letter group headings.

```

4547 \write\glswrite{^^J; letter group headings^^J}%
4548 \write\glswrite{(markup-letter-group
4549 :open-head \string"\string\glsgroupheading
4550 \glsopenbrace\string"^^J\space\space\space
4551 :close-head \string"\glsclosebrace\string")}%

```

Define additional letter groups.

```

4552 \write\glswrite{^^J; additional letter groups^^J}%
4553 \write\glswrite{\@xdylettergroups}%

```

Define additional sort rules

```

4554 \write\glswrite{^^J; additional sort rules^^J}
4555 \write\glswrite{\@xdysortrules}%

```

Close the style file

```

4556 \closeout\glswrite

```

Suppress any further calls.

```

4557 \let\writeist\relax
4558 }
4559 \else

```

Code to use if makeindex is required.

```

4560 \edef\@gls@actualchar{\string?}
4561 \edef\@gls@encapchar{\string|}
4562 \edef\@gls@levelchar{\string!}
4563 \edef\@gls@quotechar{\string"}
4564 \def\writeist{\relax
4565 \ifundef{\glswrite}{\newwrite\glswrite}{\relax
4566 \openout\glswrite=\istfilename
4567 \write\glswrite{\glspercentchar\space makeindex style file

```



```

4568     created by the glossaries package}
4569 \write\glswrite{\glpercentchar\space for document
4570   '\jobname' on \the\year-\the\month-\the\day}
4571 \write\glswrite{actual '@glactualchar'}
4572 \write\glswrite{encap '@glencapchar'}
4573 \write\glswrite{level '@gllevelchar'}
4574 \write\glswrite{quote '@glquotechar'}
4575 \write\glswrite{keyword \string"\string\glossaryentry\string"}
4576 \write\glswrite{preamble \string"\string\glossarysection[\string
4577   \glossarytoctitle]{\string\glossarytitle}\string
4578   \glossarypreamble\string\n\string\begin{theglossary}\string
4579   \glossaryheader\string\n\string"}
4580 \write\glswrite{postamble \string"\string%\string\n\string
4581   \end{theglossary}\string\glossarypostamble\string\n
4582   \string"}
4583 \write\glswrite{group_skip \string"\string\glsgroupskip\string\n
4584   \string"}
4585 \write\glswrite{item_0 \string"\string%\string\n\string"}
4586 \write\glswrite{item_1 \string"\string%\string\n\string"}
4587 \write\glswrite{item_2 \string"\string%\string\n\string"}
4588 \write\glswrite{item_01 \string"\string%\string\n\string"}
4589 \write\glswrite{item_x1
4590   \string"\string\relax \string\glresetentrylist\string\n
4591   \string"}
4592 \write\glswrite{item_12 \string"\string%\string\n\string"}
4593 \write\glswrite{item_x2
4594   \string"\string\relax \string\glresetentrylist\string\n
4595   \string"}

4596 \write\glswrite{delim_0 \string"\string\{\string
4597   \glossaryentrynumbers\string\{\string\relax \string"}
4598 \write\glswrite{delim_1 \string"\string\{\string
4599   \glossaryentrynumbers\string\{\string\relax \string"}
4600 \write\glswrite{delim_2 \string"\string\{\string
4601   \glossaryentrynumbers\string\{\string\relax \string"}
4602 \write\glswrite{delim_t \string"\string\}\string\}\string"}
4603 \write\glswrite{delim_n \string"\string\delimN \string"}
4604 \write\glswrite{delim_r \string"\string\delimR \string"}
4605 \write\glswrite{headings_flag 1}
4606 \write\glswrite{heading_prefix
4607   \string"\string\glsgroupheading\string\{\string"}
4608 \write\glswrite{heading_suffix
4609   \string"\string\}\string\relax
4610   \string\glresetentrylist \string"}
4611 \write\glswrite{symhead_positive \string"glssymbols\string"}
4612 \write\glswrite{numhead_positive \string"glnumbers\string"}
4613 \write\glswrite{page_compositor \string"glscpositor\string"}
4614 \@glscbsdq\glscsuffixF
4615 \@glscbsdq\glscsuffixFF
4616 \ifx\glscsuffixF\empty

```

```

4617 \else
4618   \write\glswrite{suffix_2p \string"\gls@suffixF\string"}
4619 \fi
4620 \ifx\gls@suffixFF\@empty
4621 \else
4622   \write\glswrite{suffix_3p \string"\gls@suffixFF\string"}
4623 \fi
4624 \closeout\glswrite
4625 \let\writeist\relax
4626 }
4627 \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`

```

4628 \newcommand{\noist}{%
  Update attributes list
4629   \@gls@addpredefinedattributes
4630   \let\writeist\relax
4631 }

```

`\@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`

```

4632 \newcommand*{\@makeglossary}[1]{%
4633   \ifglossaryexists{#1}%
4634   {%

```

Only create a new write if `savewrites=false` otherwise create a token to collect the information.

```

4635   \ifglssavewrites
4636     \expandafter\newtoks\csname glo@#1@filetok\endcsname
4637   \else
4638     \expandafter\newwrite\csname glo@#1@file\endcsname
4639     \expandafter\@glsopenfile\csname glo@#1@file\endcsname{#1}%
4640   \fi
4641   \@gls@renewglossary
4642   \writeist

```

```

4643 }%
4644 {%
4645   \PackageError{glossaries}%
4646   {Glossary type ‘#1’ not defined}%
4647   {New glossaries must be defined before using \string\makeglossary}%
4648 }%
4649 }

```

`\@glsopenfile` Open write file associated with the given glossary.

```

4650 \newcommand*{\@glsopenfile}[2]{%
4651   \immediate\openout#1=\jobname.\csname @glotype@#2@out\endcsname
4652   \PackageInfo{glossaries}{Writing glossary file
4653     \jobname.\csname @glotype@#2@out\endcsname}%
4654 }

```

`\@closegls`

```

4655 \newcommand*{\@closegls}[1]{%
4656   \closeout\csname glo@#1@file\endcsname
4657 }
4658 %   \end{macrocode}
4659 %\end{macro}
4660 %
4661 %\begin{macro}{\@gls@automake}
4662 %\changes{4.08}{2014-07-30}{new}
4663 %   \begin{macrocode}
4664 \ifglxindy
4665   \newcommand*{\@gls@automake}[1]{%
4666     \ifglossaryexists{#1}
4667     {%
4668       \@closegls{#1}%
4669       \ifdefstring{\glsorder}{letter}%
4670       {\def\@gls@order{-M ord/letorder }}%
4671       {\let\@gls@order\@empty}%
4672       \ifcsundef{@xdy@#1@language}%
4673       {\let\@gls@langmod\@xdy@main@language}%
4674       {\letcs\@gls@langmod{@xdy@#1@language}}%
4675       \edef\@gls@dothiswrite{\noexpand\write18{xindy
4676         -I xindy
4677         \@gls@order
4678         -L \@gls@langmod\space
4679         -M \gls@istfilebase\space
4680         -C \gls@codepage\space
4681         -t \jobname.\csuse{@glotype@#1@log}
4682         -o \jobname.\csuse{@glotype@#1@in}
4683         \jobname.\csuse{@glotype@#1@out}}}%
4684     }%
4685     \@gls@dothiswrite
4686   }%
4687   {%

```

```

4688     \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
4689   }%
4690 }
4691 \else
4692 \newcommand*{\@gls@automake}[1]{%
4693   \ifglossaryexists{#1}
4694   {%
4695     \@closegls{#1}%
4696     \ifdefstring{\glsorder}{letter}%
4697     {\def\@gls@order{-1 }}%
4698     {\let\@gls@order\@empty}%
4699     \edef\@gls@dothiswrite{\noexpand\write18{makeindex \@gls@order
4700       -s \istfilename\space
4701       -t \jobname.\csuse{\@glotype@#1@log}
4702       -o \jobname.\csuse{\@glotype@#1@in}
4703       \jobname.\csuse{\@glotype@#1@out}}}%
4704   }%
4705   \@gls@dothiswrite
4706 }%
4707 {%
4708   \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
4709 }%
4710 }
4711 \fi

```

`\nomakeglossaries` Issue warning that `\makeglossaries` hasn't been used.

```

4712 \newcommand*{\@warn@nomakeglossaries}{}

```

Only use this if warning if `\printglossary` has been used without `\makeglossaries`

```

4713 \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`

```

4714 \newcommand*{\makeglossaries}{%

```

Define the write used for style file also used for all other output files if `savewrites=true`.

```

4715   \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.

```

4716   \protected@write\@auxout{}{\string\providecommand\string\@glsorder[1]{} }

```

```

4717   \protected@write\@auxout{}{\string\providecommand\string\@istfilename[1]{} }

```

Write the name of the style file to the aux file (needed by makeglossaries)

```
4718 \protected@write\@auxout{}\string\@istfilename{\istfilename}}%
4719 \protected@write\@auxout{}\string\@glsorder{\glsorder}}
```

Iterate through each glossary type and activate it.

```
4720 \@for\@glo@type:=\@glo@types\do{%
4721   \ifthenelse{\equal{\@glo@type}{}}{\}%
4722   \@makeglossary{\@glo@type}}%
4723 }%
```

New glossaries must be created before \makeglossaries so disable \newglossary.

```
4724 \renewcommand*\newglossary[4][]{%
4725   \PackageError{glossaries}{New glossaries
4726   must be created before \string\makeglossaries}{You need
4727   to move \string\makeglossaries\space after all your
4728   \string\newglossary\space commands}}%
```

Any subsequent instances of this command should have no effect

```
4729 \let\@makeglossary\relax
4730 \let\makeglossary\relax
4731 \let\makeglossaries\relax
```

Disable all commands that have no effect after \makeglossaries

```
4732 \@disable@onlypremakeg
```

Allow see key:

```
4733 \let\gls@checkseeallowed\relax
```

Suppress warning about no \makeglossaries

```
4734 \let\warn@nomakeglossaries\relax
```

Activate warning about missing \printglossary

```
4735 \def\warn@noprintglossary{%
4736   \GlossariesWarningNoLine{No \string\printglossary\space
4737   or \string\printglossaries\space
4738   found.^^J(Remove \string\makeglossaries\space if you don't want
4739   any glossaries.)^^JThis document will not have a glossary}%
4740 }%
```

Declare list parser for \glsdisplaynumberlist

```
4741 \ifglssavenumberlist
4742   \edef\@gls@dodeflistparser{\noexpand\DeclareListParser
4743   {\noexpand\glsnumlistparser}{\delimN}}%
4744   \@gls@dodeflistparser
4745 \fi
```

Prevent user from also using \makenoidxglossaries

```
4746 \let\makenoidxglossaries\@no@makeglossaries
```

Prohibit sort key in printgloss family:

```
4747 \renewcommand*\@printgloss@setsort{%
4748   \let\@glo@assign@sortkey\@glo@no@assign@sortkey
4749 }%
```

Check the automake setting:

```
4750 \ifglsautomake
4751 \renewcommand*{\@gls@doautomake}{%
4752 \@for\@gls@type:=\@glo@types\do{%
4753 \ifdefempty{\@gls@type}{}%
4754 {\@gls@automake{\@gls@type}}%
4755 }%
4756 }%
4757 \fi
4758 }
```

Must occur in the preamble:

```
4759 \@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`

```
4760 \let\makeglossary\makeglossaries
```

If `\makeglossaries` hasn't been used, issue a warning. Also issue a warning if neither `\printglossaries` nor `\printglossary` have been used.

```
4761 \AtEndDocument{%
4762 \warn@nomakeglossaries
4763 \warn@noprintglossary
4764 }
```

`\makenoidxglossaries` Analogous to `\makeglossaries` this activates the commands needed for `\printnoidxglossary`

```
4765 \newcommand*{\makenoidxglossaries}{%
```

Redefine empty glossary warning:

```
4766 \renewcommand{\@gls@noref@warn}[1]{%
4767 \GlossariesWarning{Empty glossary for
4768 \string\printnoidxglossary[type={##1}].
4769 Rerun may be required (or you may have forgotten to use
4770 commands like \string\gls).}%
4771 }
```

Don't escape makeindex/xindy characters

```
4772 \let\@gls@checkmkidxchars\@gobble
```

Write glossary information to aux instead of glossary files

```
4773 \let\@@do@@wrglossary\gls@noidxglossary
```

Switch on group headings that use the character code:

```
4774 \let\@gls@getgrouptitle\@gls@noidx@getgrouptitle
```

Allow see key:

```
4775 \let\gls@checkseeallowed\relax
```

Redefine cross-referencing macro:

```
4776 \renewcommand{\@do@seeglossary}[2]{%
4777   \edef\@gls@label{\glsdetoklabel{##1}}%
4778   \protected@write\@auxout{}\{%
4779     \string\@gls@reference
4780     {\csname glo@\@gls@label @type\endcsname}%
4781     {\@gls@label}%
4782     {%
4783       \string\glsseeformat##2}%
4784     }%
4785   }%
4786 }
```

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.

```
4787 \AtBeginDocument
4788 {%
4789   \write\@auxout{\string\providecommand\string\@gls@reference[3]{}}%
4790 }
```

Change warning about no glossaries

```
4791 \def\warn@noprintglossary{%
4792   \GlossariesWarningNoLine{No \string\printnoidxglossary\space
4793     or \string\printnoidxglossaries ^^J
4794     found. (Remove \string\makenoidxglossaries\space if you
4795     don't want any glossaries.)^^JThis document will not have a glossary}%
4796 }
```

Suppress warning about no \makeglossaries

```
4797 \let\warn@nomakeglossaries\relax
```

Prevent user from also using \makeglossaries

```
4798 \let\makeglossaries\@no@makeglossaries
```

Allow sort key in printgloss family:

```
4799 \renewcommand*{\@printgloss@setsort}{%
4800   \let\@glo@assign@sortkey\@glo@assign@sortkey
```

Initialise default sort order:

```
4801 \def\@glo@sorttype{\@glo@default@sorttype}%
4802 }
```

All entries must be defined in the preamble:

```
4803 \renewcommand*\new@glossaryentry[2]{%
4804   \PackageError{glossaries}{Glossary entries must be
4805     defined in the preamble^^Jwhen you use
4806     \string\makenoidxglossaries}%
4807   {Either move your definitions to the preamble or use
```

```

4808     \string\makeglossaries}%
4809 }%

  Redefine \glsentrynumberlist
4810 \renewcommand*{\glsentrynumberlist}[1]{%
4811   \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
4812   \ifdef\@gls@loclist
4813   {%
4814     \glsnoidxloclist{\@gls@loclist}%
4815   }%
4816   {%
4817     \ifglsentryexists{##1}%
4818     {%
4819       \GlossariesWarning{Missing location list for ‘##1’. Either
4820         a rerun is required or you haven’t referenced the entry.}%
4821     }%
4822     {%
4823       \PackageError{glossaries}{Glossary entry ‘##1’ has not been
4824         defined.}{}%
4825     }%
4826   }%
4827 }%

  Redefine \glsdisplaynumberlist
4828 \renewcommand*{\glsdisplaynumberlist}[1]{%
4829   \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
4830   \ifdef\@gls@loclist
4831   {%
4832     \def\@gls@noidxloclist@sep{%
4833       \def\@gls@noidxloclist@sep{%
4834         \def\@gls@noidxloclist@sep{%
4835           \glsnumlistsep
4836         }%
4837       \def\@gls@noidxloclist@finalsep{\glsnumlistlastsep}%
4838     }%
4839   }%
4840   \def\@gls@noidxloclist@finalsep{}%
4841   \def\@gls@noidxloclist@prev{}%
4842   \forlistloop{\glsnoidxdisplayloclisthandler}{\@gls@loclist}%
4843   \@gls@noidxloclist@finalsep
4844   \@gls@noidxloclist@prev
4845 }%
4846 {%
4847   ??\ifglsentryexists{##1}%
4848   {%
4849     \GlossariesWarning{Missing location list for ‘##1’. Either
4850       a rerun is required or you haven’t referenced the entry.}%
4851   }%
4852   {%
4853     \PackageError{glossaries}{Glossary entry ‘##1’ has not been

```



```

4854         defined.}{}%
4855     }%
4856 }%
4857 }%

```

Provide a generic way of iterating through the number list:

```

4858 \renewcommand*\glsnumberlistloop}[3]{%
4859     \letcs{\@gls@loclist}{gls@glstoklabel{##1}@loclist}%
4860     \let\@gls@org@glsnoidxdisplayloc\glsnoidxdisplayloc
4861     \let\@gls@org@glsseeformat\glsseeformat
4862     \let\glsnoidxdisplayloc##2\relax
4863     \let\glsseeformat##3\relax
4864     \ifdef\@gls@loclist
4865     {%
4866         \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
4867     }%
4868     {%
4869         \ifglsentryexists{##1}%
4870         {%
4871             \GlossariesWarning{Missing location list for ‘##1’. Either
4872                 a rerun is required or you haven’t referenced the entry.}%
4873         }%
4874         {%
4875             \PackageError{glossaries}{Glossary entry ‘##1’ has not been
4876                 defined.}{}%
4877         }%
4878     }%
4879     \let\glsnoidxdisplayloc\@gls@org@glsnoidxdisplayloc
4880     \let\glsseeformat\@gls@org@glsseeformat
4881 }%

```

Modify sanitize sort function

```

4882 \let\@gls@sanitizesort\@gls@noidx@sanitizesort
4883 \let\@gls@nosanitizesort\@gls@noidx@nosanitizesort
4884 \@gls@noidx@setsanitizesort
4885 }

```

Preamble-only command:

```

4886 \@onlypreamble{\makenoidxglossaries}

```

`\glsnumberlistloop` `\glsnumberlistloop{<label>}{<handler>}`

```

4887 \newcommand*\glsnumberlistloop}[2]{%
4888     \PackageError{glossaries}{\string\glsnumberlistloop\space
4889         only works with \string\makenoidxglossaries}{}%
4890 }

```

`numberlistloophandler` Handler macro for `\glsnumberlistloop`. (The argument should be in the form `\glsnoidxdisplayloc{<prefix>}{<counter>}{<format>}{<n>}`)

```

4891 \newcommand*{\glsnoidxnumberlistloophandler}[1]{%
4892   #1%
4893 }

```

`\@no@makeglossaries` Can't use both `\makeglossaries` and `\makenoidxglossaries`

```

4894 \newcommand*{\@no@makeglossaries}{%
4895   \PackageError{glossaries}{You can't use both
4896     \string\makeglossaries\space and \string\makenoidxglossaries}%
4897   {Either use one or other (or none) of those commands but not both
4898     together.}%
4899 }

```

`\@gls@noref@warn` Warning when no instances of `\@gls@reference` found.

```

4900 \newcommand{\@gls@noref@warn}[1]{%
4901   \GlossariesWarning{\string\makenoidxglossaries\space
4902     is required to make \string\printnoidxglossary[type={#1}] work}%
4903 }

```

`\gls@noidxglossary` Write the glossary information to the aux file:

```

4904 \newcommand*{\gls@noidxglossary}{%
4905   \protected@write\@auxout{}{%
4906     \string\@gls@reference
4907     {\csname glo@\@gls@label @type\endcsname}%
4908     {\@gls@label}%
4909     {\string\glsnoidxdisplayloc
4910       {\@glo@counterprefix}%
4911       {\@gls@counter}%
4912       {\@glsnumberformat}%
4913       {\@glslocref}%
4914     }%
4915   }%
4916 }

```

1.14 Writing information to associated files

`\istfile` Deprecated.

```

4917 \def\istfile{\glswrite}

```

At the end of the document, the files should be created if `savewrites=true`.

```

4918 \AtEndDocument{%
4919   \glswritefiles
4920 }

```

`\@glswritefiles` Only write the files if `savewrites=true`

```

4921 \newcommand*{\@glswritefiles}{%
  Iterate through all the glossaries
4922   \forallglossaries{\@glo@type}{%

```

Check for empty glossaries (patch provided by Patrick Häcker)

```
4923 \ifcsundef{glo@\@glo@type @filetok}%
4924 {%
4925   \def\gls@tmp{%
4926   }%
4927   {%
4928     \edef\gls@tmp{\expandafter\the
4929       \csname glo@\@glo@type @filetok\endcsname}%
4930   }%
4931   \ifx\gls@tmp\@empty
4932     \ifx\@glo@type\glsdefaulttype
4933       \GlossariesWarningNoLine{Glossary ‘\@glo@type’ has no
4934         entries.^^JRemember to use package option ‘nomain’ if
4935 you
4936       don’t want to^^Juse the main glossary}%
4937     \else
4938       \GlossariesWarningNoLine{Glossary ‘\@glo@type’ has no
4939         entries}%
4940     \fi
4941   \else
4942     \@glsopenfile{\glswrite}{\@glo@type}%
4943     \immediate\write\glswrite{%
4944       \expandafter\the
4945       \csname glo@\@glo@type @filetok\endcsname}%
4946     \immediate\closeout\glswrite
4947   \fi
4948 }%
4949 }
```

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`

```
4950 \if@gls@docloaded
4951 \else
4952   \renewcommand*{\glossary}[1][main]{\gls@glossary{#1}}
4953 \fi
```

The associated number should be stored in `\theglsentrycounter` before using `\gls@glossary`.

`\gls@glossary`

```
4954 \newcommand*{\gls@glossary}[1]{%
4955   \@gls@glossary{#1}%
4956 }
```

`\@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.

```
4957 \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`

```
4958 \newcommand{\@gls@renewglossary}{%
4959   \gdef\@gls@glossary##1{\@bsphack\begingroup\gls@wrglossary{##1}}%
4960   \let\@gls@renewglossary\empty
4961 }
```

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 `\gls@link`).

`\gls@wrglossary`

```
4962 \newcommand*{\gls@wrglossary}[2]{%
4963   \ifglssavewrites
4964     \protected@edef\@gls@tmp{\the\csname glo@#1@filetok\endcsname#2}%
4965     \expandafter\global\expandafter\csname glo@#1@filetok\endcsname
4966       \expandafter{\@gls@tmp^^J}%
4967   \else
4968     \ifcsdef{glo@#1@file}%
4969     {%
4970       \expandafter\protected@write\csname glo@#1@file\endcsname{%
4971         \gls@disablepagerefexpansion}{#2}%
4972     }%
4973     {%
4974       \ifignoredglossary{#1}{}%
4975       {%
4976         \GlossariesWarning{No file defined for glossary '#1'}%
4977       }%
4978     }%
4979   \fi
4980 \endgroup\@esphack
4981 }
```

```

\@do@wrglossary
4982 \newcommand*{\@do@wrglossary}[1]{%
4983   \glswriteentry{#1}{\@do@wrglossary{#1}}%
4984 }

\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.
4985 \newcommand*{\glswriteentry}[2]{%
4986   \ifglsindexonlyfirst
4987     \ifglsused{#1}{#2}%
4988   \else
4989     #2%
4990   \fi
4991 }

@protected@pagefmts  List of page formats to be protected against expansion.
4992 \newcommand{\gls@protected@pagefmts}{%
4993   \gls@numberpage,\gls@alphpage,\gls@Alphpage,\gls@romanpage,\gls@Romanpage%
4994 }

blepagerefexpansion
4995 \newcommand*{\gls@disablepagerefexpansion}{%
4996   \@for\@gls@this:=\gls@protected@pagefmts\do
4997   {%
4998     \expandafter\let\@gls@this\relax
4999   }%
5000 }

\gls@alphpage
5001 \newcommand*{\gls@alphpage}{\@alph\c@page}

\gls@Alphpage
5002 \newcommand*{\gls@Alphpage}{\@Alph\c@page}

\gls@numberpage
5003 \newcommand*{\gls@numberpage}{\number\c@page}

\gls@romanpage
5004 \newcommand*{\gls@romanpage}{\romannumeral\c@page}

\gls@Romanpage
5005 \newcommand*{\gls@Romanpage}{\@Roman\c@page}

\glsaddprotectedpagefmt {\glsaddprotectedpagefmt{<cs name>}}

```

Added a page format to the list of protected page formats. The argument should be the name (without a backslash) of the command that takes a \TeX register as the argument ($\backslash\langle csname \rangle\c@page$ must be valid).

```

5006 \newcommand*\glsaddprotectedpagefmt}[1]{%
5007   \eappto\gls@protected@pagefmts{\expandonce{\csname gls#1page\endcsname}}}%
5008   \csedef{gls#1page}{\expandonce{\csname#1\endcsname}\noexpand\c@page}%
5009   \eappto\@wrglossarynumberhook{%
5010     \noexpand\let\expandonce{\csname org@gls#1\endcsname}%
5011     \expandonce{\csname#1\endcsname}%
5012     \noexpand\def\expandonce{\csname#1\endcsname}{%
5013       \noexpand\@wrglossary@pageformat
5014       \expandonce{\csname gls#1page\endcsname}%
5015       \expandonce{\csname org@gls#1\endcsname}%
5016     }%
5017   }%
5018 }
```

$\text{rglossarynumberhook}$ Hook used by $\backslash\@do@wrglossary$

```

5019 \newcommand*\@wrglossarynumberhook{}
```

$\text{glossary@pageformat}$

```

5020 \newcommand{\@wrglossary@pageformat}[3]{%
5021   \ifx#3\c@page #1\else #2#3\fi
5022 }
```

$\backslash\@do@wrglossary$ Write the glossary entry in the appropriate format. (Need to set $\backslash\@glsnumberformat$ and $\backslash\@gls@counter$ prior to use.) The argument is the entry's label.

```

5023 \newcommand*\@do@wrglossary}[1]{%
5024   \begingroup
```

First a bit of hackery to prevent premature expansion of $\c@page$. Store original definitions:

```

5025   \let\orgthe\the
5026   \let\orgnumber\number
5027   \let\orgromannumeral\romannumeral
5028   \let\orgalph\@alph
5029   \let\orgAlph\@Alph
5030   \let\orgRoman\@Roman
```

Redefine:

```

5031   \def\the##1{%
5032     \ifx##1\c@page \gls@numberpage\else\orgthe##1\fi}%
5033   \def\number##1{%
5034     \ifx##1\c@page \gls@numberpage\else\orgnumber##1\fi}%
5035   \def\romannumeral##1{%
5036     \ifx##1\c@page \gls@romanpage\else\orgromannumeral##1\fi}%
5037   \def\@Roman##1{%
5038     \ifx##1\c@page \gls@Romanpage\else\orgRoman##1\fi}%
5039   \def\@alph##1{%
```

```

5040 \ifx##1\c@page \gls@alphpage\else\orgalph##1\fi}%
5041 \def\@Alph##1{%
5042 \ifx##1\c@page \gls@Alphpage\else\orgAlph##1\fi}%

```

Add hook to allow for other number formats:

```
5043 \@wrglossarynumberhook
```

Prevent expansion:

```
5044 \gls@disablepagerefexpansion
```

Now store location in \@glslocref:

```

5045 \protected@xdef\@glslocref{\theglsentrycounter}%
5046 \endgroup

```

Escape any special characters

```
5047 \@gls@checkmkidxchars\@glslocref
```

Check if the hyper-location is the same as the location and set the hyper prefix.

```

5048 \expandafter\ifx\theglsentrycounter\theglsentrycounter\relax
5049 \def\@glo@counterprefix{%
5050 \else
5051 \protected@edef\@glsHlocref{\theglsentrycounter}%
5052 \@gls@checkmkidxchars\@glsHlocref
5053 \edef\@do@gls@getcounterprefix{\noexpand\@gls@getcounterprefix
5054 {\@glslocref}{\@glsHlocref}%
5055 }%
5056 \@do@gls@getcounterprefix
5057 \fi

```

De-tok label if required

```
5058 \edef\@gls@label{\glsdetoklabel{#1}}%
```

Write the information to file:

```

5059 \@do@wrglossary
5060 }

```

\@do@wrglossary

```
5061 \newcommand*{\@do@wrglossary}{%
```

Determine whether to use xindy or makeindex syntax

```
5062 \ifglsxindy
```

Need to determine if the formatting information starts with a (or) indicating a range.

```

5063 \expandafter\@glo@check@mkidxrangechar\@glsnumberformat\@nil
5064 \def\@glo@range{%
5065 \expandafter\if\@glo@prefix(\relax
5066 \def\@glo@range{:open-range}%
5067 \else
5068 \expandafter\if\@glo@prefix)\relax
5069 \def\@glo@range{:close-range}%
5070 \fi
5071 \fi

```

Write to the glossary file using xindy syntax.

```

5072 \gls@glossary{\csname glo@\@gls@label @type\endcsname}{%
5073 (indexentry :tkey (\csname glo@\@gls@label @index\endcsname)

5074 :locoref \string"\@glo@counterprefix}{\@gls@locoref}\string" %
5075 :attr \string"\@gls@counter\@glo@suffix\string"
5076 \@glo@range
5077 )
5078 }%
5079 \else

```

Convert the format information into the format required for makeindex

```

5080 \@set@glo@numformat{\@glo@numfmt}{\@gls@counter}{\@gls@numberformat}%
5081 {\@glo@counterprefix}%

```

Write to the glossary file using makeindex syntax.

```

5082 \gls@glossary{\csname glo@\@gls@label @type\endcsname}{%
5083 \string\glossaryentry{\csname glo@\@gls@label @index\endcsname
5084 \@gls@encapchar\@glo@numfmt}{\@gls@locoref}}%
5085 \fi
5086 }

```

`\gls@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`.) NB this assumes that the prefix ends with a dot, which is the standard. (Otherwise it makes the xindy location classes more complicated.)

```

5087 \newcommand*\@gls@getcounterprefix[2]{%
5088 \edef\@gls@thisloc{#1}\edef\@gls@thisHloc{#2}%
5089 \ifx\@gls@thisloc\@gls@thisHloc
5090 \def\@glo@counterprefix{%
5091 \else
5092 \def\@gls@get@counterprefix##1.#1##2\end@getprefix{%
5093 \def\@glo@tmp{##2}%
5094 \ifx\@glo@tmp\@empty
5095 \def\@glo@counterprefix{%
5096 \else
5097 \def\@glo@counterprefix{##1}%
5098 \fi
5099 }%
5100 \@gls@get@counterprefix#2.#1\end@getprefix

```

Warn if no prefix can be formed.

```

5101 \ifx\@glo@counterprefix\@empty
5102 \GlossariesWarning{Hyper target ‘#2’ can’t be formed by
5103 prefixing\theH\@gls@counter\@glo@suffix\@glo@range\@glo@range
5104 definition of \string\theH\@gls@counter\@glo@suffix\@glo@range\@glo@range
5105 will get the warning: “name{\@gls@counter.#1}’ has been
5106 referenced but does not exist”}%
5107 \fi

```



```

5108 \fi
5109 }

```

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.

```

5110 \newcommand{\do@seeglossary}[2]{%
5111 \def\@gls@xref{#2}%
5112 \@onelevel@sanitize\@gls@xref
5113 \@gls@checkmkidxchars\@gls@xref
5114 \ifglxsindy
5115 \gls@glossary{\csname glo@#1@type\endcsname}{%
5116 (indexentry
5117 :tkey (\csname glo@#1@index\endcsname)
5118 :xref (\string"\@gls@xref\string")
5119 :attr \string"see\string"
5120 )
5121 }%
5122 \else
5123 \gls@glossary{\csname glo@#1@type\endcsname}{%
5124 \string\glossaryentry{\csname glo@#1@index\endcsname
5125 \@gls@encapchar glsseeformat\@gls@xref}{Z}}%
5126 \fi
5127 }

```

`\@gls@fixbraces` If no optional argument is specified, list needs to be enclosed in a set of braces.

```

5128 \def\@gls@fixbraces#1#2#3\@nil{%
5129 \ifx#2[\relax
5130 \@gls@fixbraces#1#2#3\@end@fixbraces
5131 \else
5132 \def#1{{#2#3}}%
5133 \fi
5134 }

```

`\@@gls@fixbraces`

```

5135 \def\@@gls@fixbraces#1[#2]#3\@end@fixbraces{%
5136 \def#1{[#2]{#3}}%
5137 }

```

`\glssee` `\glssee{\langle label \rangle}{\langle cross-ref list \rangle}`

```

5138 \DeclareRobustCommand*\glssee[3][\seename]{%
5139 \do@seeglossary{#2}{[#1]{#3}}
5140 \newcommand*\@glssee[3][\seename]{%
5141 \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.

```
5142 \DeclareRobustCommand*\glsseeformat}[3][\seename]{%
5143   \emph{#1} \glsseelist{#2}}
```

`\glsseelist` `\glsseelist{<list>}` formats list of entry labels.

```
5144 \DeclareRobustCommand*\glsseelist}[1]{%
```

If there is only one item in the list, set the last separator to do nothing.

```
5145   \let\@gls@dolast\relax
```

Don’t display separator on the first iteration of the loop

```
5146   \let\@gls@donext\relax
```

Iterate through the labels

```
5147   \@for\@gls@thislabel:=#1\do{%
```

Check if on last iteration of loop

```
5148     \ifx\@xfor@nextelement\@nnil
```

```
5149       \@gls@dolast
```

```
5150     \else
```

```
5151       \@gls@donext
```

```
5152     \fi
```

Display the entry for this label. (Expanding label as it’s a temporary control sequence that’s used elsewhere.)

```
5153     \expandafter\glsseeitem\expandafter{\@gls@thislabel}%
```

Update separators

```
5154     \let\@gls@dolast\glsseelastsep
```

```
5155     \let\@gls@donext\glsseesep
```

```
5156   }%
```

```
5157 }
```

`\glsseelastsep` Separator to use between penultimate and ultimate entries in a cross-referencing list.

```
5158 \newcommand*\glsseelastsep{\space\andname\space}
```

`\glsseesep` Separator to use between entires in a cross-referencing list.

```
5159 \newcommand*\glsseesep}{, }
```

`\glsseeitem` `\glsseeitem{<label>}` formats individual entry in a cross-referencing list.

```
5160 \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.)

```
5161 \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.

```
5162 \newcommand*{\gls@save@numberlist}[1]{%
5163   \ifglssavenumberlist
5164     \toks@{#1}%
5165     \edef\@do@writeaux@info{%
5166       \noexpand\csgdef{glo@\glscurrententrylabel @numberlist}{\the\toks@}%
5167     }%
5168     \@onelevel@sanitize\@do@writeaux@info
5169     \protected@write\@auxout{}\{\@do@writeaux@info}%
5170   \fi
5171 }
```

`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.)

```
5172 \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.

```
5173 \ifcsundef{printglossary}{}%
5174 {%
```

If `\printglossary` is already defined, issue a warning and undefine it.

```
5175   \@gls@warnonglossdefined
5176   \undef\printglossary
5177 }
```

`\printglossary` has an optional argument. The default value is to set the glossary type to the main glossary.

```
5178 \newcommand*{\printglossary}[1][type=\glsdefaulttype]{%
5179   \@printglossary{#1}{\@print@glossary}%
5180 }
```

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`

```
5181 \newcommand*{\printglossaries}{%
5182   \forallglossaries{\@glo@type}{\printglossary[type=\@glo@type]}%
5183 }
```

`\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.

```
5184 \newcommand*{\printnoidxglossary}[1][type=\glsdefaulttype]{%
5185   \@printglossary{#1}{\@printnoidxglossary}%
5186 }
```

`\printnoidxglossaries` Analogous to `\printglossaries`

```
5187 \newcommand*{\printnoidxglossaries}{%
5188   \forallglossaries{\@glo@type}{\printnoidxglossary[type=\@glo@type]}%
5189 }
```

`@printgloss@setsort` Initialise to do nothing.

```
5190 \newcommand*{\@printgloss@setsort}{}%
```

`\@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.

```
5191 \newcommand{\@printglossary}[2]{%
  Set up defaults.
5192   \def\@glo@type{\glsdefaulttype}%
5193   \def\glossarytitle{\csname @glo@type @title\endcsname}%

5194   \def\glossarytoctitle{\glossarytitle}%
5195   \let\org@glossarytitle\glossarytitle
5196   \def\@glossarystyle{}%
5197   \def\gls@dotoc@title{\glssettoc@title{\@glo@type}}%
```

Store current value of `\glossaryentrynumbers`. (This may be changed via the optional argument)

```
5198   \let\@org@glossaryentrynumbers\glossaryentrynumbers
```

Localise the effects of the optional argument

```
5199   \bgroup
```

Activate or deactivate sort key:

```
5200     \@printgloss@setsort
```

Determine settings specified in the optional argument.

```
5201     \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)

```
5202 \ifx\glossarytitle\org@glossarytitle
5203 \else
5204 \expandafter\let\csname @glo@type\@glo@type @title\endcsname
5205 \glossarytitle
5206 \fi
```

Allow a high-level user command to indicate the current glossary

```
5207 \let\currentglossary\@glo@type
```

Enable individual number lists to be suppressed.

```
5208 \let\org@glossaryentrynumbers\glossaryentrynumbers
5209 \let\glsnonextpages\glsnonextpages
```

Enable individual number list to be activated:

```
5210 \let\glsnextpages\glsnextpages
```

Enable suppression of description terminators.

```
5211 \let\nopostdesc\nopostdesc
```

Set up the entry for the TOC

```
5212 \gls@dotoc@title
```

Set the glossary style

```
5213 \@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):

```
5214 \let\gls@org@glossaryentryfield\glossentry
5215 \let\gls@org@glossarysubentryfield\subglossentry
5216 \renewcommand{\glossentry}[1]{%
5217 \xdef\glscurrententrylabel{\glsdetoklabel{##1}}%
5218 \gls@org@glossaryentryfield{##1}%
5219 }%
5220 \renewcommand{\subglossentry}[2]{%
5221 \xdef\glscurrententrylabel{\glsdetoklabel{##2}}%
5222 \gls@org@glossarysubentryfield{##1}{##2}%
5223 }%
```

Now do the handler macro that deals with the actual glossary:

```
5224 #2%
```

End the current scope

```
5225 \egroup
```

Reset \glossaryentrynumbers

```
5226 \global\let\glossaryentrynumbers\org@glossaryentrynumbers
```

Suppress warning about no \printglossary

```
5227 \global\let\warn@noprintglossary\relax
5228 }
```

\@print@glossary Internal workings of \printglossary dealing with reading the external file.

```
5229 \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.)

```
5230 \makeatletter
```

Input the glossary file, if it exists.

```
5231 \@input@{\jobname.\csname @glo@type@\@glo@type @in\endcsname}%
```

If the glossary file doesn't exist, do \null. (This ensures that the page is shipped out and all write commands are done.) This might produce an empty page, but at this point the document isn't complete, so it shouldn't matter.

```
5232 \IfFileExists{\jobname.\csname @glo@type@\@glo@type @in\endcsname}%
```

```
5233 {}%
```

```
5234 {\null}%
```

If xindy is being used, need to write the language dependent information to the .aux file for makeglossaries.

```
5235 \ifglxindy
```

```
5236 \ifcsundef{@xdy@\@glo@type @language}%
```

```
5237 {%
```

```
5238 \edef\@do@auxoutstuff{%
```

```
5239 \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.

```
5240 \noexpand\immediate\noexpand\write\@auxout{%
```

```
5241 \string\providecommand\string\@xdylanguage[2]{}%}
```

```
5242 \noexpand\immediate\noexpand\write\@auxout{%
```

```
5243 \string\@xdylanguage{\@glo@type}{\@xdy@main@language}}%
```

```
5244 }%
```

```
5245 }%
```

```
5246 }%
```

```
5247 {%
```

```
5248 \edef\@do@auxoutstuff{%
```

```
5249 \noexpand\AtEndDocument{%
```

```
5250 \noexpand\immediate\noexpand\write\@auxout{%
```

```
5251 \string\providecommand\string\@xdylanguage[2]{}%}
```

```
5252 \noexpand\immediate\noexpand\write\@auxout{%
```

```
5253 \string\@xdylanguage{\@glo@type}{\csname @xdy@\@glo@type
```

```
5254 @language\endcsname}}%
```

```
5255 }%
```

```
5256 }%
```

```
5257 }%
```

```
5258 \@do@auxoutstuff
```

```
5259 \edef\@do@auxoutstuff{%
```

```
5260 \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.

```

5261      \noexpand\immediate\noexpand\write\@auxout{%
5262      \string\providecommand\string\@gls@codepage[2]{}}%
5263      \noexpand\immediate\noexpand\write\@auxout{%
5264      \string\@gls@codepage{\@glo@type}{\@gls@codepage}}%
5265      }%
5266    }%
5267    \@do@auxoutstuff
5268  \fi

```

Activate warning if \makeglossaries hasn't been used.

```

5269  \renewcommand*{\@warn@nomakeglossaries}{%
5270    \GlossariesWarningNoLine{\string\makeglossaries\space
5271    hasn't been used,^^Jthe glossaries will not be updated}%
5272  }%
5273 }

```

The sort macros all have the syntax:

$\backslash@glo@sortmacro@<order>\{<type>\}$

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 $\backslash@glsref@<type>$). The actual sorting is done by $\backslash@glo@sortentries\{<handler>\}\{<type>\}$.

$\backslash@glo@sortentries$

```

5274 \newcommand*{\@glo@sortentries}[2]{%
5275   \def\@glo@sortinglist{}%
5276   \def\@glo@sortinghandler{#1}%
5277   \edef\@glo@type{#2}%
5278   \forlistcsloop{\@glo@do@sortentries}{\@glsref@#2}%
5279   \csdef{\@glsref@#2}{}%
5280   \@for\@this@label:=\@glo@sortinglist\do{%

```

Has this entry already been added?

```

5281     \xifinlistcs{\@this@label}{\@glsref@#2}%
5282     {}%
5283     {%
5284       \listcsxadd{\@glsref@#2}{\@this@label}%
5285     }%
5286     \ifcsdef{\@glo@sortingchildren@\@this@label}%
5287     {%
5288       \@glo@addchildren{#2}{\@this@label}%
5289     }%
5290     {}%
5291   }%
5292 }

```

```
\@glo@addchildren \@glo@addchildren{<type>}{<parent>}
```

```
5293 \newcommand*{\@glo@addchildren}[2]{%
```

Scope to allow nesting.

```
5294 \bgroup
5295 \letcs{\@glo@childlist}{\@glo@sortingchildren@#2}%
5296 \@for\@this@childlabel:=\@glo@childlist\do
5297 {%
```

Check this label hasn't already been added.

```
5298 \xifinlistcs{\@this@childlabel}{\@glsref@#1}%
5299 {}%
5300 {%
5301 \listcsxadd{\@glsref@#1}{\@this@childlabel}%
5302 }%
```

Does this child have children?

```
5303 \ifcsdef{\@glo@sortingchildren@\@this@childlabel}%
5304 {%
5305 \@glo@addchildren{#1}{\@this@childlabel}%
5306 }%
5307 {%
5308 }%
5309 }%
5310 \egroup
5311 }
```

@glo@do@sortentries

```
5312 \newcommand*{\@glo@do@sortentries}[1]{%
5313 \ifglshasparent{#1}%
5314 {%
```

This entry has a parent, so add it to the child list

```
5315 \edef\@glo@parent{\csuse{glo@\glsdetoklabel{#1}@parent}}%
5316 \ifcsundef{\@glo@sortingchildren@\@glo@parent}%
5317 {%
5318 \csdef{\@glo@sortingchildren@\@glo@parent}{}%
5319 }%
5320 {}%
5321 \expandafter\@glo@sortedinsert
5322 \csname @glo@sortingchildren@\@glo@parent\endcsname{#1}%
```

Has the parent been added?

```
5323 \xifinlistcs{\@glo@parent}{\@glsref@\@glo@type}%
5324 {%
```

Yes, it has so do nothing.

```
5325 }%
5326 {%
```


No, it hasn't so add it now.

```

5327 \expandafter\@glo@do@sortentries\expandafter{\@glo@parent}%
5328 }%
5329 }%
5330 {%
5331 \@glo@sortedinsert{\@glo@sortinglist}{#1}%
5332 }%
5333 }

```

```
\@glo@sortedinsert \@glo@sortedinsert{<list>}{<entry label>}
```

Insert into list.

```

5334 \newcommand*\@glo@sortedinsert}[2]{%
5335 \dtl@insertinto{#2}{#1}{\@glo@sortinghandler}%
5336 }%

```

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`

```

5337 \newcommand*\@glo@sorthandler@word}[2]{%
5338 \letcs\@gls@sort@A{glo@glstetoklabel{#1}@sort}%
5339 \letcs\@gls@sort@B{glo@glstetoklabel{#2}@sort}%
5340 \edef\glo@do@compare{%
5341 \noexpand\dtlwordindexcompare{\noexpand\dtl@sortresult}%
5342 {\expandonce\@gls@sort@B}%
5343 {\expandonce\@gls@sort@A}%
5344 }%
5345 \glo@do@compare
5346 }

```

`\@glo@sorthandler@letter`

```

5347 \newcommand*\@glo@sorthandler@letter}[2]{%
5348 \letcs\@gls@sort@A{glo@glstetoklabel{#1}@sort}%
5349 \letcs\@gls@sort@B{glo@glstetoklabel{#2}@sort}%
5350 \edef\glo@do@compare{%
5351 \noexpand\dtlletterindexcompare{\noexpand\dtl@sortresult}%
5352 {\expandonce\@gls@sort@B}%
5353 {\expandonce\@gls@sort@A}%
5354 }%
5355 \glo@do@compare
5356 }

```

`\@glo@sorthandler@case` Case-sensitive sort.

```

5357 \newcommand*\@glo@sorthandler@case}[2]{%
5358 \letcs\@gls@sort@A{glo@glstetoklabel{#1}@sort}%

```

```

5359 \letcs\@gls@sort@B{glo\glsdetoklabel{#2}@sort}%
5360 \edef\glo@do@compare{%
5361   \noexpand\dtlcompare{\noexpand\dtl@sortresult}%
5362   {\expandonce\@gls@sort@B}%
5363   {\expandonce\@gls@sort@A}%
5364 }%
5365 \glo@do@compare
5366 }

```

@sorthandler@nocase Case-insensitive sort.

```

5367 \newcommand*{\@glo@sorthandler@nocase}[2]{%
5368   \letcs\@gls@sort@A{glo\glsdetoklabel{#1}@sort}%
5369   \letcs\@gls@sort@B{glo\glsdetoklabel{#2}@sort}%
5370   \edef\glo@do@compare{%
5371     \noexpand\dtlcompare{\noexpand\dtl@sortresult}%
5372     {\expandonce\@gls@sort@B}%
5373     {\expandonce\@gls@sort@A}%
5374   }%
5375   \glo@do@compare
5376 }

```

@glo@sortmacro@word Sort macro for ‘word’

```

5377 \newcommand*{\@glo@sortmacro@word}[1]{%
5378   \ifdefstring{\@glo@default@sorttype}{standard}%
5379   {%
5380     \@glo@sortentries{\@glo@sorthandler@word}{#1}%
5381   }%
5382   {%
5383     \PackageError{glossaries}{Conflicting sort options:^^J
5384     \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5385     \string\printnoidxglossary[sort=word]}{}%
5386   }%
5387 }

```

@sortmacro@letter Sort macro for ‘letter’

```

5388 \newcommand*{\@glo@sortmacro@letter}[1]{%
5389   \ifdefstring{\@glo@default@sorttype}{standard}%
5390   {%
5391     \@glo@sortentries{\@glo@sorthandler@letter}{#1}%
5392   }%
5393   {%
5394     \PackageError{glossaries}{Conflicting sort options:^^J
5395     \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5396     \string\printnoidxglossary[sort=letter]}{}%
5397   }%
5398 }

```

@sortmacro@standard Sort macro for ‘standard’. (Use either ‘word’ or ‘letter’ order.)

```

5399 \newcommand*{\@glo@sortmacro@standard}[1]{%

```

```

5400 \ifdefstring{\@glo@default@sorttype}{standard}%
5401 {%
5402   \ifcsdef{@glo@sorthandler@\glsorder}%
5403   {%
5404     \@glo@sortentries{\csuse{@glo@sorthandler@\glsorder}}{#1}%
5405   }%
5406   {%
5407     \PackageError{glossaries}{Unknown sort handler ‘\glsorder’}{}%
5408   }%
5409 }%
5410 {%
5411   \PackageError{glossaries}{Conflicting sort options:^^J
5412     \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5413     \string\printnoidxglossary[sort=standard]}{}%
5414 }%
5415 }

```

`@glo@sortmacro@case` Sort macro for ‘case’

```

5416 \newcommand*{\@glo@sortmacro@case}[1]{%
5417   \ifdefstring{\@glo@default@sorttype}{standard}%
5418   {%
5419     \@glo@sortentries{\@glo@sorthandler@case}{#1}%
5420   }%
5421   {%
5422     \PackageError{glossaries}{Conflicting sort options:^^J
5423       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5424       \string\printnoidxglossary[sort=case]}{}%
5425   }%
5426 }

```

`@glo@sortmacro@nocase` Sort macro for ‘nocase’

```

5427 \newcommand*{\@glo@sortmacro@nocase}[1]{%
5428   \ifdefstring{\@glo@default@sorttype}{standard}%
5429   {%
5430     \@glo@sortentries{\@glo@sorthandler@nocase}{#1}%
5431   }%
5432   {%
5433     \PackageError{glossaries}{Conflicting sort options:^^J
5434       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5435       \string\printnoidxglossary[sort=nocase]}{}%
5436   }%
5437 }

```

`@glo@sortmacro@def` Sort macro for ‘def’. The order of definition is given in `\glo@list@<type>`.

```

5438 \newcommand*{\@glo@sortmacro@def}[1]{%
5439   \def\@glo@sortinglist{}%
5440   \for\glsentries[#1]{\@gls@thislabel}%
5441   {%
5442     \xifinlistcs{\@gls@thislabel}{\@gls@ref@#1}%

```

```

5443    {%
5444        \listeadadd{\@glo@sortinglist}{\@gls@thislabel}%
5445    }%
5446    {%

    Hasn't been referenced.

5447    }%
5448    }%
5449    \cslet{@glsref@#1}{\@glo@sortinglist}%
5450 }

```

`\@glo@sortmacro@def@do` This won't include parent entries that haven't been referenced.

```

5451 \newcommand*{\@glo@sortmacro@def@do}[1]{%
5452     \ifinlistcs{#1}{@glsref@{\@glo@type}%
5453     }%
5454     {%
5455         \listcsadd{@glsref@{\@glo@type}{#1}%
5456     }%
5457     \ifcsdef{@glo@sortingchildren@#1}%
5458     {%
5459         \@glo@addchildren{\@glo@type}{#1}%
5460     }%
5461     }%
5462 }

```

`\@glo@sortmacro@use` Sort macro for 'use'. (No sorting is required, as the entries are already in order of use, so do nothing.)

```

5463 \newcommand*{\@glo@sortmacro@use}[1]{%

```

`\print@noidx@glossary` Glossary handler for `\printnoidxglossary` which doesn't use an indexing application. Since `\printnoidxglossary` may occur at the start of the document, we can't just check if an entry has been used. Instead, the first pass needs to write information to the aux file every time an entry is referenced. This needs to be read in on the second run and stored in a list corresponding to the appropriate glossary.

```

5464 \newcommand*{\@print@noidx@glossary}{%
5465     \ifcsdef{@glsref@{\@glo@type}%
5466     {%

```

Sort the entries:

```

5467     \ifcsdef{@glo@sortmacro@{\@glo@sorttype}%
5468     {%
5469         \csuse{@glo@sortmacro@{\@glo@sorttype}{\@glo@type}%
5470     }%
5471     {%
5472         \PackageError{glossaries}{Unknown sort handler '@@glo@sorttype'}{ }%
5473     }%

```

Do the glossary heading and preamble

```

5474 \glossarysection[\glossarytoctitle]{\glossarytitle}%
5475 \glossarypreamble
5476 \begin{theglossary}%
5477 \glossaryheader
5478 \glsresetentrylist
5479 \def\@gls@currentlettergroup{}%

```

Iterate through the entries.

```

5480 \forlistcsloop{\@gls@noidx@do}{\@glsref@{\@glo@type}%

```

Finally end the glossary and do the postamble:

```

5481 \end{theglossary}%
5482 \glossarypostamble
5483 }%
5484 {%
5485 \@gls@noref@warn{\@glo@type}%
5486 }%
5487 }

```

\glo@grabfirst

```

5488 \def\glo@grabfirst#1#2\@nil{%
5489 \def\@gls@firsttok{#1}%
5490 \ifdefempty\@gls@firsttok
5491 {%
5492 \def\@glo@thislettergrp{0}%
5493 }%
5494 {%

```

Sanitize it:

```

5495 \@onelevel@sanitize\@gls@firsttok

```

Fetch the first letter:

```

5496 \expandafter\@glo@grabfirst\@gls@firsttok{ }{\@nil
5497 }%
5498 }

```

\@glo@grabfirst

```

5499 \def\@glo@grabfirst#1#2\@nil{%
5500 \ifdefempty\@glo@thislettergrp
5501 {%
5502 \def\@glo@thislettergrp{glssymbols}%
5503 }%
5504 {%
5505 \count@=\uccode'#1\relax
5506 \ifnum\count@=0\relax
5507 \def\@glo@thislettergrp{glssymbols}%
5508 \else
5509 \ifdefstring\@glo@sorttype{case}%
5510 {%
5511 \count@='#1\relax
5512 }%

```

```

5513      {%
5514      }%
5515      \edef\@glo@thislettergrp{\the\count@}%
5516      \fi
5517    }%
5518 }

```

\@gls@noidx@do Handler for list iteration used by \@print@noidx@glossary. The argument is the entry label. This only allows one sublevel.

```

5519 \newcommand{\@gls@noidx@do}[1]{%

```

 Get this entry's location list

```

5520   \global\letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%

```

 Does this entry have a parent?

```

5521   \ifglshasparent{#1}%
5522   {%

```

 Has a parent.

```

5523     \gls@level=\csuse{glo@\glsdetoklabel{#1}@level}\relax
5524     \ifdefvoid{\@gls@loclist}
5525     {%
5526       \subglossentry{\gls@level}{#1}{}%
5527     }%
5528     {%
5529       \subglossentry{\gls@level}{#1}%
5530     }%
5531     \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
5532     }%
5533   }%
5534 }%
5535 {%

```

 Doesn't have a parent Get this entry's sort key

```

5536   \letcs{\@gls@sort}{glo@\glsdetoklabel{#1}@sort}%

```

 Fetch the first letter:

```

5537   \expandafter\glo@grabfirst\@gls@sort{}{}\@nil
5538   \ifdefequal{\@glo@thislettergrp}{\@gls@currentlettergroup}%
5539   {%
5540   {%

```

 Do the group header:

```

5541       \ifdefempty{\@gls@currentlettergroup}{\@gls@groupskip}%
5542       \gls@groupheading{\@glo@thislettergrp}%
5543     }%
5544   \let\@gls@currentlettergroup\@glo@thislettergrp

```

 Do this entry:

```

5545       \ifdefvoid{\@gls@loclist}
5546       {%
5547       \glossentry{#1}{}%

```

```

5548 }%
5549 {%
5550   \glossentry{#1}%
5551   {%
5552     \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
5553   }%
5554 }%
5555 }%
5556 }

```

`\glsnoidxloclist` `\glsnoidxloclist{<list cs>}`

Display location list.

```

5557 \newcommand*{\glsnoidxloclist}[1]{%
5558   \def\@gls@noidxloclist@sep{%
5559     \def\@gls@noidxloclist@prev{%
5560       \forlistloop{\glsnoidxloclisthandler}{#1}%
5561     }

```

`noidxloclisthandler` Handler for location list iterator.

```

5562 \newcommand*{\glsnoidxloclisthandler}[1]{%
5563   \ifdefstring{\@gls@noidxloclist@prev}{#1}%
5564   {%

```

Same as previous location so skip.

```

5565 }%
5566 {%
5567   \@gls@noidxloclist@sep
5568   #1%
5569   \def\@gls@noidxloclist@sep{\delimN}%
5570   \def\@gls@noidxloclist@prev{#1}%
5571 }%
5572 }

```

`splayloclisthandler` Handler for location list iterator when used with `\glsdisplaynumberlist`.

```

5573 \newcommand*{\glsnoidxdisplayloclisthandler}[1]{%
5574   \ifdefstring{\@gls@noidxloclist@prev}{#1}%
5575   {%

```

Same as previous location so skip.

```

5576 }%
5577 {%
5578   \@gls@noidxloclist@sep
5579   \@gls@noidxloclist@prev
5580   \def\@gls@noidxloclist@prev{#1}%
5581 }%
5582 }

```

`\glsnoidxdisplayloc` `\glsnoidxdisplayloc{<prefix>}{<counter>}{<format>}{<location>}`

Display a location in the location list.

```
5583 \newcommand*\glsnoidxdisplayloc[4]{%
5584   \setentrycounter[#1]{#2}%
5585   \csuse{#3}{#4}%
5586 }
```

`\@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.

```
5587 \newcommand*\@gls@reference[3]{%
```

Add to label list

```
5588   \glsdoifexistsorwarn{#2}%
5589   {%
5590     \ifcsundef{@glsref@#1}{\csgdef{@glsref@#1}{}}{}%
5591     \ifinlistcs{#2}{@glsref@#1}%
5592     {}%
5593     {\listcsgadd{@glsref@#1}{#2}}%
5594 }
```

Add to location list

```
5594   \ifcsundef{glo@\glsdetoklabel{#2}@loclist}%
5595   {\csgdef{glo@\glsdetoklabel{#2}@loclist}{}}{}%
5596   {}%
5597   \listcsgadd{glo@\glsdetoklabel{#2}@loclist}{#3}%
5598   }%
5599 }
```

The keys that can be used in the optional argument to `\printglossary` or `\printnoidxglossary` are as follows: The `type` key sets the glossary type.

```
5600 \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`.

```
5601 \define@key{printgloss}{title}{%
5602   \def\glossarytitle{#1}%
5603   \let\gls@dotoc@title\relax
5604 }
```

The `toctitle` sets the text used for the relevant entry in the table of contents.

```
5605 \define@key{printgloss}{toctitle}{%
5606   \def\glossarytoctitle{#1}%
5607   \let\gls@dotoc@title\relax
5608 }
```


The style key sets the glossary style (but only for the given glossary).

```

5609 \define@key{printgloss}{style}{%
5610   \ifcsundef{@glsstyle@#1}%
5611   {%
5612     \PackageError{glossaries}%
5613     {Glossary style ‘#1’ undefined}{}%
5614   }%
5615   {%
5616     \def\@glossarystyle{\setglossentrycompatibility
5617       \csname @glsstyle@#1\endcsname}%
5618   }%
5619 }
```

The numberedsection key determines if this glossary should be in a numbered section.

```

5620 \define@choicekey{printgloss}{numberedsection}[\val\nr]{%
5621 false,nolabel,autolabel,nameref}[nolabel]{%
5622   \ifcase\nr\relax
5623     \renewcommand*{\@glossarysecstar}{*}%
5624     \renewcommand*{\@glossaryseclabel}{}%
5625   \or
5626     \renewcommand*{\@glossarysecstar}{}%
5627     \renewcommand*{\@glossaryseclabel}{}%
5628   \or
5629     \renewcommand*{\@glossarysecstar}{}%
5630     \renewcommand*{\@glossaryseclabel}{\label{\glsautoprefix\@glo@type}}%
5631   \or
5632     \renewcommand*{\@glossarysecstar}{*}%
5633     \renewcommand*{\@glossaryseclabel}{}%
5634     \protected@edef\@currentlabelname{\glossarytoctitle}%
5635     \label{\glsautoprefix\@glo@type}}%
5636   \fi
5637 }
```

The nogroupskip key determines whether or not there should be a vertical gap between glossary groups.

```

5638 \define@choicekey{printgloss}{nogroupskip}{true,false}[true]{%
5639   \csuse{glsnogroupskip#1}%
5640 }
```

The nopostdot key has the same effect as the package option of the same name.

```

5641 \define@choicekey{printgloss}{nopostdot}{true,false}[true]{%
5642   \csuse{glsnopostdot#1}%
5643 }
```

The entrycounter key is the same as the package option but localised to the current glossary.

```

5644 \define@choicekey{printgloss}{entrycounter}{true,false}[true]{%
5645   \csuse{glsentrycounter#1}%
5646 }
```

```

5646 \ifglentrycounter
5647   \ifx\@gls@counterwithin\@empty
5648     \newcounter{glossaryentry}%
5649   \else
5650     \newcounter{glossaryentry}[\@gls@counterwithin]%
5651   \fi
5652   \def\theHglossaryentry{\currentglossary.\theglossaryentry}%
5653   \renewcommand*{\glsresetentrycounter}{%
5654     \setcounter{glossaryentry}{0}%
5655   }%
5656   \renewcommand*{\glsstepentry}[1]{%
5657     \refstepcounter{glossaryentry}%
5658     \label{glentry-\glsdetoklabel{##1}}%
5659   }%
5660   \renewcommand*{\glentrycounterlabel}{\theglossaryentry.\space}%
5661   \renewcommand*{\glentryitem}[1]{%
5662     \glsstepentry{##1}\glentrycounterlabel
5663   }%
5664 \else
5665   \renewcommand*{\glsresetentrycounter}{}%
5666   \renewcommand*{\glsstepentry}[1]{}%
5667   \renewcommand*{\glentrycounterlabel}{}%
5668   \renewcommand*{\glentryitem}[1]{\glsresetsubentrycounter}
5669 \fi
5670 }

```

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.

```

5671 \define@choicekey{printgloss}{subentrycounter}{true,false}[true]{%
5672   \csuse{glssubentrycounter#1}%
5673   \ifglssubentrycounter
5674     \ifundef\c@glossarysubentry
5675     {%
5676       \ifglentrycounter
5677         \newcounter{glossarysubentry}[glossaryentry]%
5678       \else
5679         \newcounter{glossarysubentry}
5680       \fi
5681     }{}%
5682   \renewcommand*{\glsstepsubentry}[1]{%
5683     \edef\currentglssubentry{\glsdetoklabel{##1}}%
5684     \refstepcounter{glossarysubentry}%
5685     \label{glentry-\currentglssubentry}%
5686   }%
5687   \renewcommand*{\glsresetsubentrycounter}{%
5688     \setcounter{glossarysubentry}{0}%
5689   }%

```

```

5690 \renewcommand*{\glssubentryitem}[1]{%
5691 \glssubentry{##1}\glssubentrycounterlabel
5692 }%
5693 \renewcommand*{\glssubentrycounterlabel}{\theglossarysubentry}\space}%
5694 \def\theHglossarysubentry{\currentglssubentry.\theglossarysubentry}
5695 \else
5696 \renewcommand*{\glssubentryitem}[1]{}%
5697 \renewcommand*{\glssubentry}[1]{}%
5698 \renewcommand*{\glssubentrycounter}{}%
5699 \renewcommand*{\glssubentrycounterlabel}{}%
5700 \fi
5701 }

```

The nonnumberlist key determines if this glossary should have a number list.

```

5702 \define@boolkey{printgloss}[gls]{nonnumberlist}[true]{%
5703 \ifglslnonnumberlist
5704 \def\glossaryentrynumbers##1{}%
5705 \else
5706 \def\glossaryentrynumbers##1{##1}%
5707 \fi}

```

The sort key sets the glossary sort handler (\printnoidxglossary only).

```

5708 \define@key{printgloss}{sort}{\@glo@assign@sortkey{#1}}

```

`\no@assign@sortkey` Issue error if used with \printglossary

```

5709 \newcommand*{\@glo@no@assign@sortkey}[1]{%
5710 \PackageError{glossaries}{‘sort’ key not permitted with
5711 \string\printglossary}%
5712 {The ‘sort’ key may only be used with \string\printnoidxglossary}%
5713 }

```

`\glo@assign@sortkey` For use with \printnoidxglossary

```

5714 \newcommand*{\@glo@assign@sortkey}[1]{%
5715 \def\@glo@sorttype{#1}%
5716 }

```

`\@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.

```

5717 \newcommand*{\@glslnonextpages}{%
5718 \gdef\glossaryentrynumbers##1{%
5719 \glslresetentrylist
5720 }%
5721 }

```

`\@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.

```
5722 \newcommand*{\@glsnextpages}{%
5723   \gdef\glossaryentrynumbers##1{%
5724     ##1\glsresetentrylist}}
```

`\glsresetentrylist` Resets `\glossaryentrynumbers`

```
5725 \newcommand*{\glsresetentrylist}{%
5726   \global\let\glossaryentrynumbers\org@glossaryentrynumbers}
```

`\glsnonextpages` Outside of `\printglossary` this does nothing.

```
5727 \newcommand*{\glsnonextpages}{}%
```

`\glsnextpages` Outside of `\printglossary` this does nothing.

```
5728 \newcommand*{\glsnextpages}{}%
```

`glossaryentry` If the `entrycounter` package option has been used, define a counter to number each level 0 entry.

```
5729 \ifglentrycounter
5730   \ifx\@gls@counterwithin\@empty
5731     \newcounter{glossaryentry}
5732   \else
5733     \newcounter{glossaryentry}[\@gls@counterwithin]
5734   \fi
5735   \def\theHglossaryentry{\currentglossary.\theglossaryentry}
5736 \fi
```

`glossarysubentry` If the `subentrycounter` package option has been used, define a counter to number each level 1 entry.

```
5737 \ifglssubentrycounter
5738   \ifglentrycounter
5739     \newcounter{glossarysubentry}[glossaryentry]
5740   \else
5741     \newcounter{glossarysubentry}
5742   \fi
5743   \def\theHglossarysubentry{\currentglssubentry.\theglossarysubentry}
5744 \fi
```

`esetsubentrycounter` Resets the `glossarysubentry` counter.

```
5745 \ifglssubentrycounter
5746   \newcommand*{\glsresetsubentrycounter}{%
5747     \setcounter{glossarysubentry}{0}%
5748   }
5749 \else
5750   \newcommand*{\glsresetsubentrycounter}{}%
5751 \fi
```

`\resetentrycounter` Resets the glossentry counter.

```
5752 \ifglentrycounter
5753   \newcommand*{\glsresetentrycounter}{%
5754     \setcounter{glossaryentry}{0}%
5755   }
5756 \else
5757   \newcommand*{\glsresetentrycounter}{}
5758 \fi
```

`\glsstepentry` Advance the glossaryentry counter if in use. The argument is the label associated with the entry.

```
5759 \ifglentrycounter
5760   \newcommand*{\glsstepentry}[1]{%
5761     \refstepcounter{glossaryentry}%
5762     \label{glentry-\glsdetoklabel{#1}}%
5763   }
5764 \else
5765   \newcommand*{\glsstepentry}[1]{}
5766 \fi
```

`\glsstepsubentry` Advance the glossarysubentry counter if in use. The argument is the label associated with the subentry.

```
5767 \ifglssubentrycounter
5768   \newcommand*{\glsstepsubentry}[1]{%
5769     \edef\currentglssubentry{\glsdetoklabel{#1}}%
5770     \refstepcounter{glossarysubentry}%
5771     \label{glentry-\currentglssubentry}%
5772   }
5773 \else
5774   \newcommand*{\glsstepsubentry}[1]{}
5775 \fi
```

`\glsrefentry` Reference the entry or sub-entry counter if in use, otherwise just do `\gls`.

```
5776 \ifglentrycounter
5777   \newcommand*{\glsrefentry}[1]{\ref{glentry-\glsdetoklabel{#1}}}
5778 \else
5779   \ifglssubentrycounter
5780     \newcommand*{\glsrefentry}[1]{\ref{glentry-\glsdetoklabel{#1}}}
5781   \else
5782     \newcommand*{\glsrefentry}[1]{\gls{#1}}
5783   \fi
5784 \fi
```

`\glsentrycounterlabel` Defines how to display the glossaryentry counter.

```
5785 \ifglentrycounter
5786   \newcommand*{\glsentrycounterlabel}{\theglossaryentry.\space}
5787 \else
5788   \newcommand*{\glsentrycounterlabel}{}
5789 \fi
```

`subentrycounterlabel` Defines how to display the `glossarysubentry` counter.

```
5790 \ifglssubentrycounter
5791   \newcommand*{\glssubentrycounterlabel}{\theglossarysubentry}\space}
5792 \else
5793   \newcommand*{\glssubentrycounterlabel}{}
5794 \fi
```

`\glstentryitem` Step and display `glossaryentry` counter, if appropriate.

```
5795 \ifglstentrycounter
5796   \newcommand*{\glstentryitem}[1]{%
5797     \glststepentry{#1}\glstentrycounterlabel
5798   }
5799 \else
5800   \newcommand*{\glstentryitem}[1]{\glstresetsubentrycounter}
5801 \fi
```

`\glssubentryitem` Step and display `glossarysubentry` counter, if appropriate.

```
5802 \ifglssubentrycounter
5803   \newcommand*{\glssubentryitem}[1]{%
5804     \glststepsubentry{#1}\glssubentrycounterlabel
5805   }
5806 \else
5807   \newcommand*{\glssubentryitem}[1]{}
5808 \fi
```

`theglossary` If the `theglossary` environment has already been defined, a warning will be issued. This environment should be redefined by glossary styles.

```
5809 \ifcsundef{theglossary}%
5810 {%
5811   \newenvironment{theglossary}{}{}%
5812 }%
5813 {%
5814   \@gls@warnontheglossdefined
5815   \renewenvironment{theglossary}{}{}%
5816 }
```

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`

```
5817 \newcommand*{\glossaryheader}{}%
```

`\glstarget` `\glstarget{<label>}{<name>}`

Provide user interface to `\@glstarget` to make it easier to modify the glossary style in the document.

```
5818 \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}
```

```
5819 \providecommand*{\compatibleglossentry}[2]{%
5820   \toks@{#2}%
5821   \protected@edef\@do@glossentry{\noexpand\glossaryentryfield{#1}%
5822     {\noexpand\glsnamefont
5823       {\expandafter\expandonce\csname glo@#1@name\endcsname}}}%
5824     {\expandafter\expandonce\csname glo@#1@desc\endcsname}}%
5825     {\expandafter\expandonce\csname glo@#1@symbol\endcsname}}%
5826     {\the\toks@}%
5827   }%
5828   \@do@glossentry
5829 }
```

`\glossentryname`

```
5830 \newcommand*{\glossentryname}[1]{%
5831   \glsdoifexistsorwarn{#1}%
5832   {%
5833     \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
5834     \expandafter\glsnamefont\expandafter{\glo@name}%
5835   }%
5836 }
```

`\Glossentryname`

```
5837 \newcommand*{\Glossentryname}[1]{%
5838   \glsdoifexistsorwarn{#1}%
5839   {%
5840     \glsnamefont{\Glsentryname{#1}}%
5841   }%
5842 }
```

`\glossentrydesc`

```
5843 \newcommand*{\glossentrydesc}[1]{%
5844   \glsdoifexistsorwarn{#1}%
5845   {%
5846     \glsentrydesc{#1}%
5847   }%
5848 }
```

\Glossentrydesc

```
5849 \newcommand*{\Glossentrydesc}[1]{%
5850   \glsdoifexistsorwarn{#1}%
5851   {%
5852     \Glsentrydesc{#1}%
5853   }%
5854 }
```

\glossentrysymbol

```
5855 \newcommand*{\glossentrysymbol}[1]{%
5856   \glsdoifexistsorwarn{#1}%
5857   {%
5858     \glsentrysymbol{#1}%
5859   }%
5860 }
```

\Glossentrysymbol

```
5861 \newcommand*{\Glossentrysymbol}[1]{%
5862   \glsdoifexistsorwarn{#1}%
5863   {%
5864     \Glsentrysymbol{#1}%
5865   }%
5866 }
```

patiblesubglossentry

`\subglossentry{<level>}{<label>}{<page-list>}`

```
5867 \providecommand*{\compatiblesubglossentry}[3]{%
5868   \toks@{#3}%
5869   \protected@edef\@do@subglossentry{\noexpand\glossarysubentryfield{\number#1}%
5870     {#2}%
5871     {\noexpand\glsnamefont
5872       {\expandafter\expandonce\csname glo@#2@name\endcsname}}%
5873     {\expandafter\expandonce\csname glo@#2@desc\endcsname}%
5874     {\expandafter\expandonce\csname glo@#2@symbol\endcsname}%
5875     {\the\toks@}}%
5876   }%
5877   \@do@subglossentry
5878 }
```

sentrycompatibility

```
5879 \newcommand*{\setglossentrycompatibility}{%
5880   \let\glossentry\compatibleglossentry
5881   \let\subglossentry\compatiblesubglossentry
5882 }
5883 \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.

```
5884 \newcommand{\glossaryentryfield}[5]{%
5885   \GlossariesWarning
5886   {Deprecated use of \string\glossaryentryfield.^^J
5887     I recommend you change to \string\glossentry.^^J
5888     If you've just upgraded, try removing your gls auxiliary
5889     files^^J and recompile}%
5890   \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.)

```
5891 \newcommand*{\glossarysubentryfield}[6]{%
5892   \GlossariesWarning
5893   {Deprecated use of \string\glossarysubentryfield.^^J
5894     I recommend you change to \string\subglossentry.^^J
5895     If you've just upgraded, try removing your gls auxiliary
5896     files^^J and recompile}%
5897   \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`

```
5898 \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`

```
5899 \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`

```
5900 \newcommand*{\glsgetgrouptitle}[1]{%
5901   \@gls@getgrouptitle{#1}{\@gls@grptitle}%
5902   \@gls@grptitle
5903 }
```

`\@gls@getgrouptitle` Gets the group title specified by the label (first argument) and stores in the second argument, which must be a control sequence.

```
5904 \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.

```
5905   \dtl@ifsingle{#1}%
5906   {%
5907     \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
5908   }%
5909   {%
5910     \ifboolexpr{test{\ifstrequal{#1}{glsymbols}}
5911                or test{\ifstrequal{#1}{glsnumbers}}}%
5912     {%
5913       \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
```

```

5914 }%
5915 {%
5916 \def#2{#1}%
5917 }%
5918 }%
5919 }

```

`@getothergrouptitle` Version for the no-indexing app option:

```

5920 \newcommand*{\@gls@noidx@getgrouptitle}[2]{%
5921 \DTLifint{#1}%
5922 {\edef#2{\char#1\relax}}%
5923 {%
5924 \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
5925 }%
5926 }

```

`\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`

```

5927 \newcommand*{\glsgetgrouplabel}[1]{%
5928 \ifthenelse{\equal{#1}{\glssymbolsgroupname}}{\glssymbols}{%
5929 \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`

```

5930 \newcommand*{\setentrycounter}[2][ ]{%
5931 \def\@glo@counterprefix{#1}%
5932 \ifx\@glo@counterprefix\@empty
5933 \def\@glo@counterprefix{.}%
5934 \else
5935 \def\@glo@counterprefix{.#1.}%
5936 \fi
5937 \def\glsentrycounter{#2}%
5938 }

```

The current glossary style can be set using `\setglossarystyle{<style>}`.

`\setglossarystyle`

```

5939 \newcommand*{\setglossarystyle}[1]{%
5940 \ifcsundef{\glsstyle@#1}%
5941 {%

```

```

5942 \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
5943 }%
5944 {%
5945 \csname @glsstyle@#1\endcsname
5946 }%
5947 }

```

`\glossarystyle`

```

5948 \newcommand*{\glossarystyle}[1]{%
5949 \ifcsundef{@glsstyle@#1}%
5950 {%
5951 \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
5952 }%
5953 {%
5954 \GlossariesWarning
5955 {Deprecated command \string\glossarystyle.^~J
5956 I recommend you switch to \string\setglossarystyle\space unless
5957 you want to maintain backward compatibility}%
5958 \setglossentrycompatibility
5959 \csname @glsstyle@#1\endcsname

5960 \ifcsdef{@glscompstyle@#1}%
5961 {\setglossentrycompatibility\csuse{@glscompstyle@#1}}%
5962 }%
5963 }%
5964 }

```

`\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.

```

5965 \newcommand{\newglossarystyle}[2]{%
5966 \ifcsundef{@glsstyle@#1}%
5967 {%
5968 \expandafter\def\csname @glsstyle@#1\endcsname{#2}%
5969 }%
5970 {%
5971 \PackageError{glossaries}{Glossary style ‘#1’ is already defined}{}%
5972 }%
5973 }

```

`\renewglossarystyle` Code for this macro supplied by Marco Daniel.

```

5974 \newcommand{\renewglossarystyle}[2]{%
5975 \ifcsundef{@glsstyle@#1}%

```

```

5976  {%
5977    \PackageError{glossaries}{Glossary style ‘#1’ isn’t already defined}{}%
5978  }%
5979  {%
5980    \csdef{@glsstyle@#1}{#2}%
5981  }%
5982 }

```

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`

```

5983 \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`

```

5984 \ifcsundef{hyperlink}%
5985 {%
5986   \def\glshypernumber#1{#1}%
5987 }%
5988 {%
5989   \def\glshypernumber#1{\@glshypernumber#1\nohyperpage{}}\@nil}
5990 }

```

`\@glshypernumber` This code was provided by Heiko Oberdiek to allow material to be attached to the location.

```

5991 \def\@glshypernumber#1\nohyperpage#2#3\@nil{%
5992   \ifx\#1\%
5993     \else
5994       \@delimR#1\delimR\delimR\%

```

```

5995 \fi
5996 \ifx\|#2\|%
5997 \else
5998 #2%
5999 \fi
6000 \ifx\|#3\|%
6001 \else
6002 \@glshypernumber#3\@nil
6003 \fi
6004 }

```

`\@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`

```

6005 \def\@delimR#1\delimR #2\delimR #3\|%
6006 \ifx\|#2\|%
6007 \@delimN{#1}%
6008 \else
6009 \@gls@numberlink{#1}\delimR\@gls@numberlink{#2}%
6010 \fi}

```

`\@delimN` displays a list of individual numbers, instead of a range:

`\@delimN`

```

6011 \def\@delimN#1{\@delimN#1\delimN \delimN\}
6012 \def\@delimN#1\delimN #2\delimN#3\|%
6013 \ifx\|#3\|%
6014 \@gls@numberlink{#1}%
6015 \else
6016 \@gls@numberlink{#1}\delimN\@gls@numberlink{#2}%
6017 \fi
6018 }

```

The following code is modified from `hyperref's \HyInd@pagelink` where the name of the counter being used is given by `\@gls@counter`.

```

6019 \def\@gls@numberlink#1{%
6020 \begingroup
6021 \toks@={}%
6022 \@gls@removespaces#1 \@nil
6023 \endgroup}

6024 \def\@gls@removespaces#1 #2\@nil{%
6025 \toks@=\expandafter{\the\toks@#1}%
6026 \ifx\|#2\|%
6027 \edef\x{\the\toks@}%
6028 \ifx\x\empty
6029 \else
6030 \hyperlink{\glstrycounter\@gls@counterprefix\the\toks@}%

```

```

6031             {\the\toks@}%
6032   \fi
6033 \else
6034   \@gls@ReturnAfterFi{%
6035     \@gls@removespaces#2\@nil
6036   }%
6037 \fi
6038 }
6039 \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`

```
6040 \newcommand*\hyperrm[1]{\textrm{\glshypernumber{#1}}}
```

`\hypersf`

```
6041 \newcommand*\hypersf[1]{\textsf{\glshypernumber{#1}}}
```

`\hypertt`

```
6042 \newcommand*\hypertt[1]{\texttt{\glshypernumber{#1}}}
```

`\hyperbf`

```
6043 \newcommand*\hyperbf[1]{\textbf{\glshypernumber{#1}}}
```

`\hypermd`

```
6044 \newcommand*\hypermd[1]{\textmd{\glshypernumber{#1}}}
```

`\hyperit`

```
6045 \newcommand*\hyperit[1]{\textit{\glshypernumber{#1}}}
```

`\hypersl`

```
6046 \newcommand*\hypersl[1]{\textsl{\glshypernumber{#1}}}
```

`\hyperup`

```
6047 \newcommand*\hyperup[1]{\textup{\glshypernumber{#1}}}
```

`\hypersc`

```
6048 \newcommand*\hypersc[1]{\textsc{\glshypernumber{#1}}}
```

`\hyperemph`

```
6049 \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.

```
6050 \newcommand{\oldacronym}[4][\gls@label]{%
6051   \def\gls@label{#2}%
6052   \newacronym[#4]{#1}{#2}{#3}%
6053   \ifcsundef{xspace}%
6054   {%
6055     \expandafter\edef\csname#1\endcsname{%
6056       \noexpand\@ifstar{\noexpand\Gls{#1}}{\noexpand\gls{#1}}}%
6057   }%
6058 }%
6059 {%
6060   \expandafter\edef\csname#1\endcsname{%
6061     \noexpand\@ifstar{\noexpand\Gls{#1}\noexpand\xspace}{%
6062       \noexpand\gls{#1}\noexpand\xspace}%
6063   }%
6064 }%
6065 }
```

`\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`

```
6066 \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.

```
6067 \newcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}
```

If `garamondx` has been loaded, need to use `\textulc` instead of `\textup`.

`\glstextup`

```
6068 \newrobustcmd*{\glstextup}[1]{\ifdef\textulc{\textulc{#1}}{\textup{#1}}}
```

The following are defined for compatibility with version 2.07 and earlier.

`\glsshortkey`

```
6069 \newcommand*{\glsshortkey}{short}
```

`\glsshortpluralkey`

```
6070 \newcommand*{\glsshortpluralkey}{shortplural}
```

`\glslongkey`

```
6071 \newcommand*{\glslongkey}{long}
```

`\glslongpluralkey`

```
6072 \newcommand*{\glslongpluralkey}{longplural}
```

`\acrfull` Full form of the acronym.

```
6073 \newrobustcmd*{\acrfull}{\@gls@hyp@opt\ns@acrfull}
```

```
6074 \newcommand*{\ns@acrfull}[2] [] {%
```

```
6075 \new@ifnextchar[{\@acrfull{#1}{#2}}{%
```

```
6076 \@acrfull{#1}{#2} []}%
```

```
6077 }
```

`\@acrfull` Low-level macro:

```
6078 \def\@acrfull#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6079 \acrfullfmt{#1}{#2}{#3}%
```

```
6080 }
```

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.

```
6081 \newcommand*\acrfullfmt}[3]{%
6082   \acrlinkfullformat{\@acrlong}{\@acrshort}{#1}{#2}{#3}%
6083 }
```

`\acrlinkfullformat` Format for full links like `\acrfull`. Syntax: `\acrlinkfullformat{<long cs>}{<short cs>}{<options>}{<label>}{<insert>}`

```
6084 \newcommand*\acrlinkfullformat}[5]{%
6085   \acrfullformat{#1}{#3}{#4}[#5]{#2}{#3}{#4}[]}%
6086 }
```

`\acrfullformat` Default full form is `<long>` (`<short>`).

```
6087 \newcommand*\acrfullformat}[2]{#1\glsspace(#2)}
```

`\glsspace` Robust space to ensure it's written to the `.glsdefs` file.

```
6088 \newrobustcmd*\glsspace{\space}
```

Default format for full acronym

`\Acrfull`

```
6089 \newrobustcmd*\Acrfull{\@gls@hyp@opt\ns@Acrfull}

6090 \newcommand*\ns@Acrfull[2][]{%
6091   \new@ifnextchar[{\@Acrfull{#1}{#2}}%
6092     {\@Acrfull{#1}{#2}[]}%
6093 }
```

Low-level macro:

```
6094 \def\@Acrfull#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6095   \Acrfullfmt{#1}{#2}{#3}%
6096 }
```

`\Acrfullfmt` First letter upper case full format.

```
6097 \newcommand*\Acrfullfmt}[3]{%
6098   \acrlinkfullformat{\@Acrlong}{\@acrshort}{#1}{#2}{#3}%
6099 }
```

`\ACRfull`

```
6100 \newrobustcmd*\ACRfull{\@gls@hyp@opt\ns@ACRfull}

6101 \newcommand*\ns@ACRfull[2][]{%
6102   \new@ifnextchar[{\@ACRfull{#1}{#2}}%
6103     {\@ACRfull{#1}{#2}[]}%
6104 }
```

Low-level macro:

```
6105 \def\@ACRfull#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6106 \ACRfullfmt{#1}{#2}{#3}%  
6107 }
```

\ACRfullfmt All upper case full format.

```
6108 \newcommand*\ACRfullfmt}[3]{%  
6109 \acrlinkfullformat{\@ACRlong}{\@ACRshort}{#1}{#2}{#3}%  
6110 }
```

Plural:

\acrfullpl

```
6111 \newrobustcmd*\acrfullpl{\@gls@hyp@opt\ns@acrfullpl}  
  
6112 \newcommand*\ns@acrfullpl[2][ ]{%  
6113 \new@ifnextchar[\@acrfullpl{#1}{#2}}%  
6114 {\@acrfullpl{#1}{#2}[ ]}%  
6115 }
```

Low-level macro:

```
6116 \def\@acrfullpl#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6117 \acrfullplfmt{#1}{#2}{#3}%  
6118 }
```

\acrfullplfmt No case change plural full format.

```
6119 \newcommand*\acrfullplfmt}[3]{%  
6120 \acrlinkfullformat{\@acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%  
6121 }
```

\Acrfullpl

```
6122 \newrobustcmd*\Acrfullpl{\@gls@hyp@opt\ns@Acrfullpl}  
  
6123 \newcommand*\ns@Acrfullpl[2][ ]{%  
6124 \new@ifnextchar[\@Acrfullpl{#1}{#2}}%  
6125 {\@Acrfullpl{#1}{#2}[ ]}%  
6126 }
```

Low-level macro:

```
6127 \def\@Acrfullpl#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6128 \Acrfullplfmt{#1}{#2}{#3}%  
6129 }
```

`\Acrfullplfmt` First letter upper case plural full format.

```
6130 \newcommand*\Acrfullplfmt}[3]{%
6131   \acrlinkfullformat{\@Acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%
6132 }
```

`\ACRfullpl`

```
6133 \newrobustcmd*\ACRfullpl{\@gls@hyp@opt\ns@ACRfullpl}

6134 \newcommand*\ns@ACRfullpl[2][]{%
6135   \new@ifnextchar[{\@ACRfullpl{#1}{#2}}%
6136     {\@ACRfullpl{#1}{#2}[]}%
6137 }
```

Low-level macro:

```
6138 \def\@ACRfullpl#1#2[#3]{%
  Make it easier for acronym styles to change this:
6139   \ACRfullplfmt{#1}{#2}{#3}%
6140 }
```

`\ACRfullplfmt` All upper case plural full format.

```
6141 \newcommand*\ACRfullplfmt}[3]{%
6142   \acrlinkfullformat{\@ACRlongpl}{\@ACRshortpl}{#1}{#2}{#3}%
6143 }
```

1.18 Predefined acronym styles

`\acronymfont` This is only used with the additional acronym styles:

```
6144 \newcommand{\acronymfont}[1]{#1}
```

`\firstacronymfont` This is only used with the additional acronym styles:

```
6145 \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.

```
6146 \newcommand*\acrnameformat}[2]{\acronymfont{#1}}
```

Define some tokens used by `\newacronym`:

`\glskeylisttok`

```
6147 \newtoks\glskeylisttok
```

`\glslabeltok`

```
6148 \newtoks\glslabeltok
```

`\glsshorttok`

```
6149 \newtoks\glsshorttok
```

```

\glslongtok
6150 \newtoks\glslongtok

\newacronymhook Provide a hook for \newacronym:
6151 \newcommand*{\newacronymhook}{}

etGenericNewAcronym New improved version of setting the acronym style.
6152 \newcommand*{\SetGenericNewAcronym}{%
    Change the behaviour of \Glsentryname to workaround expansion issues that
    cause a problem for \makefirstuc
6153 \let\@Gls@entryname\@Gls@acentryname
    Change the way acronyms are defined:
6154 \renewcommand{\newacronym}[4][]{%
6155 \ifdefempty{\@glsacronymlists}%
6156 {%
6157 \def\@glo@type{\acronymtype}%
6158 \setkeys{glossentry}{##1}%
6159 \DeclareAcronymList{\@glo@type}%
6160 }%
6161 }%
6162 \glskeylisttok{##1}%
6163 \glslabeltok{##2}%
6164 \glsshorttok{##3}%
6165 \glslongtok{##4}%
6166 \newacronymhook
6167 \protected@edef\@do@newglossaryentry{%
6168 \noexpand\newglossaryentry{\the\glslabeltok}%
6169 {%
6170 type=\acronymtype,%
6171 name={\expandonce{\acronymentry{##2}}},%
6172 sort={\acronymsort{\the\glsshorttok}{\the\glslongtok}},%
6173 text={\the\glsshorttok},%
6174 short={\the\glsshorttok},%
6175 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6176 long={\the\glslongtok},%
6177 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6178 \GenericAcronymFields,%
6179 \the\glskeylisttok
6180 }%
6181 }%
6182 \@do@newglossaryentry
6183 }%
    Make sure that \acrfull etc reflects the new style:
6184 \renewcommand*{\acrfullfmt}[3]{%
6185 \glslink{##1}{##2}{\genacrfullformat{##2}{##3}}}%
6186 \renewcommand*{\Acrfullfmt}[3]{%
6187 \glslink{##1}{##2}{\Genacrfullformat{##2}{##3}}}%

```

```

6188 \renewcommand*{\ACRfullfmt}[3]{%
6189   \glslink[##1]{##2}{%
6190     \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}%
6191 \renewcommand*{\acrfullplfmt}[3]{%
6192   \glslink[##1]{##2}{\genplacrfullformat{##2}{##3}}}%
6193 \renewcommand*{\Acrfullplfmt}[3]{%
6194   \glslink[##1]{##2}{\Genplacrfullformat{##2}{##3}}}%
6195 \renewcommand*{\ACRfullplfmt}[3]{%
6196   \glslink[##1]{##2}{%
6197     \mfirstucMakeUppercase{\genplacrfullformat{##2}{##3}}}%

```

Make sure that \glsentryfull etc reflects the new style:

```

6198 \renewcommand*{\glsentryfull}[1]{\genacrfullformat{##1}{}}%
6199 \renewcommand*{\Glsentryfull}[1]{\Genacrfullformat{##1}{}}%
6200 \renewcommand*{\glsentryfullpl}[1]{\genplacrfullformat{##1}{}}%
6201 \renewcommand*{\Glsentryfullpl}[1]{\Genplacrfullformat{##1}{}}%
6202 }

```

`\GenericAcronymFields` Fields used by `\SetGenericNewAcronym` that can be changed by the acronym style.

```

6203 \newcommand*{\GenericAcronymFields}{description={\the\glslongtok}}

```

`\acronymentry` `\acronymentry{<label>}`

Display style for the name field in the list of acronyms.

```

6204 \newcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{#1}}}

```

`\acronymsort` `\acronymsort{<short>}{<long>}`

Default sort format for acronyms.

```

6205 \newcommand*{\acronymsort}[2]{#1}

```

`\setacronymstyle` `\setacronymstyle{<style name>}`

```

6206 \newcommand*{\setacronymstyle}[1]{%
6207   \ifcsundef{@glsacr@dispstyle@#1}%
6208   {%
6209     \PackageError{glossaries}{Undefined acronym style ‘#1’}{%
6210     }%
6211   }%
6212   \ifdefempty{\@glsacronymlists}%
6213   {%
6214     \DeclareAcronymList{acronymtype}%
6215   }%

```

```

6216    {}%
6217    \SetGenericNewAcronym
6218    \GlsUseAcrStyleDefs{#1}%
6219    \@for\@gls@type:=\@glsacronymlists\do{%
6220        \defglsentryfmt[\@gls@type]{\GlsUseAcrEntryDispStyle{#1}}%
6221    }%
6222 }%
6223 }

```

`\newacronymstyle` `\newacronymstyle{<style name>}{<entry format definition>}{<display definitions>}`

Defines a new acronym style called *<style name>*.

```

6224 \newcommand*\newacronymstyle}[3]{%
6225     \ifcsdef{@glsacr@dispstyle@#1}%
6226     {%
6227         \PackageError{glossaries}{Acronym style ‘#1’ already exists}{}%
6228     }%
6229     {%
6230         \csdef{@glsacr@dispstyle@#1}{#2}%
6231         \csdef{@glsacr@styledefs@#1}{#3}%
6232     }%
6233 }

```

`\renewacronymstyle` Redefines the given acronym style.

```

6234 \newcommand*\renewacronymstyle}[3]{%
6235     \ifcsdef{@glsacr@dispstyle@#1}%
6236     {%
6237         \csdef{@glsacr@dispstyle@#1}{#2}%
6238         \csdef{@glsacr@styledefs@#1}{#3}%
6239     }%
6240     {%
6241         \PackageError{glossaries}{Acronym style ‘#1’ doesn’t exist}{}%
6242     }%
6243 }

```

`\seAcrEntryDispStyle`

```

6244 \newcommand*\GlsUseAcrEntryDispStyle[1]{\csuse{@glsacr@dispstyle@#1}}

```

`\GlsUseAcrStyleDefs`

```

6245 \newcommand*\GlsUseAcrStyleDefs[1]{\csuse{@glsacr@styledefs@#1}}

```

Predefined acronym styles:

`long-short` *<long>* (*<short>*) acronym style.

```

6246 \newacronymstyle{long-short}%
6247 {%

```

Check for long form in case this is a mixed glossary.

```

6248 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6249 }%
6250 {%
6251 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6252 \renewcommand*{\genacrfullformat}[2]{%
6253 \glentrylong{##1}##2\space
6254 (\protect\firstacronymfont{\glentryshort{##1}})%
6255 }%
6256 \renewcommand*{\Genacrfullformat}[2]{%
6257 \Glentrylong{##1}##2\space
6258 (\protect\firstacronymfont{\glentryshort{##1}})%
6259 }%
6260 \renewcommand*{\genplacrfullformat}[2]{%
6261 \glentrylongpl{##1}##2\space
6262 (\protect\firstacronymfont{\glentryshortpl{##1}})%
6263 }%
6264 \renewcommand*{\Genplacrfullformat}[2]{%
6265 \Glentrylongpl{##1}##2\space
6266 (\protect\firstacronymfont{\glentryshortpl{##1}})%
6267 }%
6268 \renewcommand*{\acronymentry}[1]{\acronymfont{\glentryshort{##1}}}%
6269 \renewcommand*{\acronymsort}[2]{##1}%
6270 \renewcommand*{\acronymfont}[1]{##1}%
6271 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
6272 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6273 }

```

long-sp-short Similar to the previous style but allows the space between the long and short form to be customized.

```

6274 \newacronymstyle{long-sp-short}%
6275 {%

```

Check for long form in case this is a mixed glossary.

```

6276 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6277 }%
6278 {%
6279 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6280 \renewcommand*{\genacrfullformat}[2]{%
6281 \glentrylong{##1}##2\glsacspace{##1}%
6282 (\protect\firstacronymfont{\glentryshort{##1}})%
6283 }%
6284 \renewcommand*{\Genacrfullformat}[2]{%
6285 \Glentrylong{##1}##2\glsacspace{##1}%
6286 (\protect\firstacronymfont{\glentryshort{##1}})%
6287 }%
6288 \renewcommand*{\genplacrfullformat}[2]{%
6289 \glentrylongpl{##1}##2\glsacspace{##1}%
6290 (\protect\firstacronymfont{\glentryshortpl{##1}})%

```



```

6291 }%
6292 \renewcommand*\Genplacrfullformat}[2]{%
6293   \Glsentrylongpl{##1}##2\glsacspace{##1}%
6294   (\protect\firstacronymfont{\glsentryshortpl{##1}})%
6295 }%
6296 \renewcommand*\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6297 \renewcommand*\acronymsort}[2]{##1}%
6298 \renewcommand*\acronymfont}[1]{##1}%
6299 \renewcommand*\firstacronymfont}[1]{\acronymfont{##1}}%
6300 \renewcommand*\acrpluralsuffix{\glspluralsuffix}%
6301 }

```

`\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.

```

6302 \newcommand*\glsacspace}[1]{%
6303   \settowidth{\dimen@}{(\firstacronymfont{\glsentryshort{##1}})}%
6304   \ifdim\dimen@<3em~\else\space\fi
6305 }

```

`short-long` *<short>* (*<long>*) acronym style.

```

6306 \newacronymstyle{short-long}%
6307 {%
  Check for long form in case this is a mixed glossary.
6308   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6309 }%
6310 {%
6311   \renewcommand*\GenericAcronymFields{description={\the\glslongtok}}%
6312   \renewcommand*\genacrfullformat}[2]{%
6313     \protect\firstacronymfont{\glsentryshort{##1}}##2\space
6314     (\glsentrylong{##1})%
6315   }%
6316   \renewcommand*\Genacrfullformat}[2]{%
6317     \protect\firstacronymfont{\Glsentryshort{##1}}##2\space
6318     (\glsentrylong{##1})%
6319   }%
6320   \renewcommand*\genplacrfullformat}[2]{%
6321     \protect\firstacronymfont{\glsentryshortpl{##1}}##2\space
6322     (\glsentrylongpl{##1})%
6323   }%
6324   \renewcommand*\Genplacrfullformat}[2]{%
6325     \protect\firstacronymfont{\Glsentryshortpl{##1}}##2\space
6326     (\glsentrylongpl{##1})%
6327   }%
6328   \renewcommand*\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6329   \renewcommand*\acronymsort}[2]{##1}%
6330   \renewcommand*\acronymfont}[1]{##1}%
6331   \renewcommand*\firstacronymfont}[1]{\acronymfont{##1}}%

```

```

6332 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6333 }

```

long-sc-short *<long>* (\textsc{<short>}) acronym style.

```

6334 \newacronymstyle{long-sc-short}%
6335 {%
6336   \GlsUseAcrEntryDisplayStyle{long-short}%
6337 }%
6338 {%
6339   \GlsUseAcrStyleDefs{long-short}%
6340   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6341   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6342 }

```

long-sm-short *<long>* (\textsmaller{<short>}) acronym style.

```

6343 \newacronymstyle{long-sm-short}%
6344 {%
6345   \GlsUseAcrEntryDisplayStyle{long-short}%
6346 }%
6347 {%
6348   \GlsUseAcrStyleDefs{long-short}%
6349   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6350   \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6351 }

```

sc-short-long *<short>* (\textsc{<long>}) acronym style.

```

6352 \newacronymstyle{sc-short-long}%
6353 {%
6354   \GlsUseAcrEntryDisplayStyle{short-long}%
6355 }%
6356 {%
6357   \GlsUseAcrStyleDefs{short-long}%
6358   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6359   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6360 }

```

sm-short-long *<short>* (\textsmaller{<long>}) acronym style.

```

6361 \newacronymstyle{sm-short-long}%
6362 {%
6363   \GlsUseAcrEntryDisplayStyle{short-long}%
6364 }%
6365 {%
6366   \GlsUseAcrStyleDefs{short-long}%
6367   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6368   \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6369 }

```

long-short-desc *<long>* ({<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

6370 \newacronymstyle{long-short-desc}%
6371 {%
6372   \GlsUseAcrEntryDispStyle{long-short}%
6373 }%
6374 {%
6375   \GlsUseAcrStyleDefs{long-short}%
6376   \renewcommand*{\GenericAcronymFields}{}%
6377   \renewcommand*{\acronymsort}[2]{##2}%
6378   \renewcommand*{\acronymentry}[1]{%
6379     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6380 }

```

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.

```

6381 \newacronymstyle{long-sp-short-desc}%
6382 {%
6383   \GlsUseAcrEntryDispStyle{long-sp-short}%
6384 }%
6385 {%
6386   \GlsUseAcrStyleDefs{long-sp-short}%
6387   \renewcommand*{\GenericAcronymFields}{}%
6388   \renewcommand*{\acronymsort}[2]{##2}%
6389   \renewcommand*{\acronymentry}[1]{%
6390     \glentrylong{##1}\glsacspace{##1}(\acronymfont{\glentryshort{##1}})}%
6391 }

```

long-sc-short-desc *⟨long⟩* (\textsc{⟨short⟩}) acronym style that has an accompanying description (which the user needs to supply).

```

6392 \newacronymstyle{long-sc-short-desc}%
6393 {%
6394   \GlsUseAcrEntryDispStyle{long-sc-short}%
6395 }%
6396 {%
6397   \GlsUseAcrStyleDefs{long-sc-short}%
6398   \renewcommand*{\GenericAcronymFields}{}%
6399   \renewcommand*{\acronymsort}[2]{##2}%
6400   \renewcommand*{\acronymentry}[1]{%
6401     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6402 }

```

long-sm-short-desc *⟨long⟩* (\textsmaller{⟨short⟩}) acronym style that has an accompanying description (which the user needs to supply).

```

6403 \newacronymstyle{long-sm-short-desc}%
6404 {%
6405   \GlsUseAcrEntryDispStyle{long-sm-short}%
6406 }%
6407 {%

```

```

6408 \GlsUseAcrStyleDefs{long-sm-short}%
6409 \renewcommand*{\GenericAcronymFields}{}%
6410 \renewcommand*{\acronymsort}[2]{##2}%
6411 \renewcommand*{\acronymentry}[1]{%
6412     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6413 }

```

short-long-desc *<short>* (*{<long>}*) acronym style that has an accompanying description (which the user needs to supply).

```

6414 \newacronymstyle{short-long-desc}%
6415 {%
6416     \GlsUseAcrEntryDisplayStyle{short-long}%
6417 }%
6418 {%
6419     \GlsUseAcrStyleDefs{short-long}%
6420     \renewcommand*{\GenericAcronymFields}{}%
6421     \renewcommand*{\acronymsort}[2]{##2}%
6422     \renewcommand*{\acronymentry}[1]{%
6423         \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6424 }

```

sc-short-long-desc *<long>* (`\textsc{<short>}`) acronym style that has an accompanying description (which the user needs to supply).

```

6425 \newacronymstyle{sc-short-long-desc}%
6426 {%
6427     \GlsUseAcrEntryDisplayStyle{sc-short-long}%
6428 }%
6429 {%
6430     \GlsUseAcrStyleDefs{sc-short-long}%
6431     \renewcommand*{\GenericAcronymFields}{}%
6432     \renewcommand*{\acronymsort}[2]{##2}%
6433     \renewcommand*{\acronymentry}[1]{%
6434         \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6435 }

```

sm-short-long-desc *<long>* (`\textsmaller{<short>}`) acronym style that has an accompanying description (which the user needs to supply).

```

6436 \newacronymstyle{sm-short-long-desc}%
6437 {%
6438     \GlsUseAcrEntryDisplayStyle{sm-short-long}%
6439 }%
6440 {%
6441     \GlsUseAcrStyleDefs{sm-short-long}%
6442     \renewcommand*{\GenericAcronymFields}{}%
6443     \renewcommand*{\acronymsort}[2]{##2}%
6444     \renewcommand*{\acronymentry}[1]{%
6445         \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6446 }

```

dua *⟨long⟩* only acronym style.

```
6447 \newacronymstyle{dua}%
6448 {%
```

Check for long form in case this is a mixed glossary.

```
6449 \ifdefempty\glscustomtext
6450 {%
6451   \ifglshaslong{\glslabel}%
6452   {%
6453     \glsifplural
6454     {%
```

Plural form:

```
6455       \glscapscase
6456       {%
```

Plural form, don't adjust case:

```
6457       \glentrylongpl{\glslabel}\glsinsert
6458       }%
6459       {%
```

Plural form, make first letter upper case:

```
6460       \Glentrylongpl{\glslabel}\glsinsert
6461       }%
6462       {%
```

Plural form, all caps:

```
6463       \mfirstucMakeUppercase
6464       {\glentrylongpl{\glslabel}\glsinsert}%
6465       }%
6466       }%
6467       {%
```

Singular form

```
6468       \glscapscase
6469       {%
```

Singular form, don't adjust case:

```
6470       \glentrylong{\glslabel}\glsinsert
6471       }%
6472       {%
```

Subsequent singular form, make first letter upper case:

```
6473       \Glentrylong{\glslabel}\glsinsert
6474       }%
6475       {%
```

Subsequent singular form, all caps:

```
6476       \mfirstucMakeUppercase
6477       {\glentrylong{\glslabel}\glsinsert}%
6478       }%
6479       }%
6480       }%
6481       {%
```

Not an acronym:

```
6482     \glsgenentryfmt
6483     }%
6484     }%
6485     {\glscustomtext\glsinsert}%
6486 }%
6487 {%
6488     \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%

6489     \renewcommand*{\acrfullfmt}[3]{%
6490         \glslink[##1]{##2}{\glsentrylong{##2}##3\space
6491             (\acronymfont{\glsentryshort{##2}})}}%
6492     \renewcommand*{\Acrfullfmt}[3]{%
6493         \glslink[##1]{##2}{\Glsentrylong{##2}##3\space
6494             (\acronymfont{\glsentryshort{##2}})}}%
6495     \renewcommand*{\ACRfullfmt}[3]{%
6496         \glslink[##1]{##2}{%
6497             \mfirstucMakeUppercase{\glsentrylong{##2}##3\space
6498                 (\acronymfont{\glsentryshort{##2}})}}}%

6499     \renewcommand*{\acrfullplfmt}[3]{%
6500         \glslink[##1]{##2}{\glsentrylongpl{##2}##3\space
6501             (\acronymfont{\glsentryshortpl{##2}})}}%

6502     \renewcommand*{\Acrfullplfmt}[3]{%
6503         \glslink[##1]{##2}{\Glsentrylongpl{##2}##3\space
6504             (\acronymfont{\glsentryshortpl{##2}})}}%
6505     \renewcommand*{\ACRfullplfmt}[3]{%
6506         \glslink[##1]{##2}{%
6507             \mfirstucMakeUppercase{\glsentrylongpl{##2}##3\space
6508                 (\acronymfont{\glsentryshortpl{##2}})}}}%
6509     \renewcommand*{\glsentryfull}[1]{%
6510         \glsentrylong{##1}\space(\acronymfont{\glsentryshort{##1}})%
6511     }%
6512     \renewcommand*{\Glsentryfull}[1]{%
6513         \Glsentrylong{##1}\space(\acronymfont{\glsentryshort{##1}})%
6514     }%
6515     \renewcommand*{\glsentryfullpl}[1]{%
6516         \glsentrylongpl{##1}\space(\acronymfont{\glsentryshortpl{##1}})%
6517     }%
6518     \renewcommand*{\Glsentryfullpl}[1]{%
6519         \Glsentrylongpl{##1}\space(\acronymfont{\glsentryshortpl{##1}})%
6520     }%
6521     \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6522     \renewcommand*{\acronymsort}[2]{##1}%
6523     \renewcommand*{\acronymfont}[1]{##1}%
6524     \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6525 }
```

dua-desc *<long>* only acronym style with user-supplied description.

```
6526 \newacronymstyle{dua-desc}%
6527 {%
6528   \GlsUseAcrEntryDispStyle{dua}%
6529 }%
6530 {%
6531   \GlsUseAcrStyleDefs{dua}%
6532   \renewcommand*{\GenericAcronymFields}{}%

6533   \renewcommand*{\acronymentry}[1]{\acronymfont{\glentrylong{##1}}}%
6534   \renewcommand*{\acronymsort}[2]{##2}%
6535 }%
```

footnote *<short>*\footnote{*<long>*} acronym style.

```
6536 \newacronymstyle{footnote}%
6537 {%

  Check for long form in case this is a mixed glossary.

6538   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6539 }%
6540 {%
6541   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
```

Need to ensure hyperlinks are switched off on first use:

```
6542   \glshyperfirstfalse
6543   \renewcommand*{\genacrfullformat}[2]{%
6544     \protect\firstacronymfont{\glentryshort{##1}}##2%
6545     \protect\footnote{\glentrylong{##1}}%
6546   }%
6547   \renewcommand*{\Genacrfullformat}[2]{%
6548     \firstacronymfont{\Glsentryshort{##1}}##2%
6549     \protect\footnote{\glentrylong{##1}}%
6550   }%
6551   \renewcommand*{\genplacrfullformat}[2]{%
6552     \protect\firstacronymfont{\glentryshortpl{##1}}##2%
6553     \protect\footnote{\glentrylongpl{##1}}%
6554   }%
6555   \renewcommand*{\Genplacrfullformat}[2]{%
6556     \protect\firstacronymfont{\Glsentryshortpl{##1}}##2%
6557     \protect\footnote{\glentrylongpl{##1}}%
6558   }%
6559   \renewcommand*{\acronymentry}[1]{\acronymfont{\glentryshort{##1}}}%
6560   \renewcommand*{\acronymsort}[2]{##1}%
6561   \renewcommand*{\acronymfont}[1]{##1}%
6562   \renewcommand*{\acrpluralsuffix}{\glacrpluralsuffix}%
```

Don't use footnotes for \acrfull:

```
6563   \renewcommand*{\acrfullfmt}[3]{%
6564     \glslink{##1}{##2}{\acronymfont{\glentryshort{##2}}##3\space
6565       (\glentrylong{##2})}%
```

```

6566 \renewcommand*{\Acrfullfmt}[3]{%
6567   \glslink[##1]{##2}{\acronymfont{\Glsentryshort{##2}}##3\space
6568   (\Glsentrylong{##2})}%
6569 \renewcommand*{\ACRfullfmt}[3]{%
6570   \glslink[##1]{##2}{%
6571     \mfirstucMakeUppercase{\acronymfont{\Glsentryshort{##2}}##3\space
6572     (\Glsentrylong{##2})}%
6573 \renewcommand*{\acrfullplfmt}[3]{%
6574   \glslink[##1]{##2}{\acronymfont{\Glsentryshortpl{##2}}##3\space
6575   (\Glsentrylongpl{##2})}%
6576 \renewcommand*{\Acrfullplfmt}[3]{%
6577   \glslink[##1]{##2}{\acronymfont{\Glsentryshortpl{##2}}##3\space
6578   (\Glsentrylongpl{##2})}%
6579 \renewcommand*{\ACRfullplfmt}[3]{%
6580   \glslink[##1]{##2}{%
6581     \mfirstucMakeUppercase{\acronymfont{\Glsentryshortpl{##2}}##3\space
6582     (\Glsentrylongpl{##2})}%

```

Similarly for \glsentryfull etc:

```

6583 \renewcommand*{\glsentryfull}[1]{%
6584   \acronymfont{\Glsentryshort{##1}}\space(\Glsentrylong{##1})}%
6585 \renewcommand*{\Glsentryfull}[1]{%
6586   \acronymfont{\Glsentryshort{##1}}\space(\Glsentrylong{##1})}%
6587 \renewcommand*{\glsentryfullpl}[1]{%
6588   \acronymfont{\Glsentryshortpl{##1}}\space(\Glsentrylongpl{##1})}%
6589 \renewcommand*{\Glsentryfullpl}[1]{%
6590   \acronymfont{\Glsentryshortpl{##1}}\space(\Glsentrylongpl{##1})}%
6591 }

```

footnote-sc \textsc{<short>}\footnote{<long>} acronym style.

```

6592 \newacronymstyle{footnote-sc}%
6593 {%
6594   \GlsUseAcrEntryDisplayStyle{footnote}%
6595 }%
6596 {%
6597   \GlsUseAcrStyleDefs{footnote}%
6598 \renewcommand{\acronymentry}[1]{\acronymfont{\Glsentryshort{##1}}}
6599 \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6600 \renewcommand*{\acrpluralsuffix}{\glssupacrpluralsuffix}%
6601 }%

```

footnote-sm \textsmaller{<short>}\footnote{<long>} acronym style.

```

6602 \newacronymstyle{footnote-sm}%
6603 {%
6604   \GlsUseAcrEntryDisplayStyle{footnote}%
6605 }%
6606 {%
6607   \GlsUseAcrStyleDefs{footnote}%
6608 \renewcommand{\acronymentry}[1]{\acronymfont{\Glsentryshort{##1}}}
6609 \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%

```



```

6610 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6611 }%

```

footnote-desc *<short>*\footnote{*<long>*} acronym style that has an accompanying description (which the user needs to supply).

```

6612 \newacronymstyle{footnote-desc}%
6613 {%
6614   \GlsUseAcrEntryDisplayStyle{footnote}%
6615 }%
6616 {%
6617   \GlsUseAcrStyleDefs{footnote}%
6618   \renewcommand*{\GenericAcronymFields}{}%
6619   \renewcommand*{\acronymsort}[2]{##2}%
6620   \renewcommand*{\acronymentry}[1]{%
6621     \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6622 }

```

footnote-sc-desc \textsc{*<short>*}\footnote{*<long>*} acronym style that has an accompanying description (which the user needs to supply).

```

6623 \newacronymstyle{footnote-sc-desc}%
6624 {%
6625   \GlsUseAcrEntryDisplayStyle{footnote-sc}%
6626 }%
6627 {%
6628   \GlsUseAcrStyleDefs{footnote-sc}%
6629   \renewcommand*{\GenericAcronymFields}{}%
6630   \renewcommand*{\acronymsort}[2]{##2}%
6631   \renewcommand*{\acronymentry}[1]{%
6632     \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6633 }

```

footnote-sm-desc \textsmaller{*<short>*}\footnote{*<long>*} acronym style that has an accompanying description (which the user needs to supply).

```

6634 \newacronymstyle{footnote-sm-desc}%
6635 {%
6636   \GlsUseAcrEntryDisplayStyle{footnote-sm}%
6637 }%
6638 {%
6639   \GlsUseAcrStyleDefs{footnote-sm}%
6640   \renewcommand*{\GenericAcronymFields}{}%
6641   \renewcommand*{\acronymsort}[2]{##2}%
6642   \renewcommand*{\acronymentry}[1]{%
6643     \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6644 }

```

fineAcronymSynonyms

```

6645 \newcommand*{\DefineAcronymSynonyms}{%

```

Short form

\acs

6646 \let\acs\acrshort

First letter uppercase short form

\Acs

6647 \let\Acs\Acrshort

Plural short form

\acsp

6648 \let\acsp\acrshortpl

First letter uppercase plural short form

\Acsp

6649 \let\Acsp\Acrshortpl

Long form

\acl

6650 \let\acl\acrlong

Plural long form

\aclp

6651 \let\aclp\acrlongpl

First letter upper case long form

\Acl

6652 \let\Acl\Acrlong

First letter upper case plural long form

\Aclp

6653 \let\Aclp\Acrlongpl

Full form

\acf

6654 \let\acf\acrfull

Plural full form

\acfp

6655 \let\acfp\acrfullpl

First letter upper case full form

\Acf

6656 \let\Acf\Acrfull

First letter upper case plural full form

`\Acfp`

```
6657 \let\Acfp\Acrfullpl
```

Standard form

`\ac`

```
6658 \let\ac\gls
```

First upper case standard form

`\Ac`

```
6659 \let\Ac\Gls
```

Standard plural form

`\acp`

```
6660 \let\acp\glspl
```

Standard first letter upper case plural form

`\Acp`

```
6661 \let\Acp\Glspl
```

```
6662 }
```

Define synonyms if required

```
6663 \ifglsacrshortcuts
```

```
6664 \DefineAcronymSynonyms
```

```
6665 \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.

```
6666 \newcommand*\SetDefaultAcronymDisplayStyle}[1]{%
```

```
6667 \defglsentryfmt[#1]{\glsentryfmt}%
```

```
6668 }
```

`\DefaultNewAcronymDef` Sets up the acronym definition for the default style. The information is provided by the tokens `\glslabeltok`, `\glsshorttok`, `\glslongtok` and `\glskeylisttok`.

```
6669 \newcommand*\DefaultNewAcronymDef{%
```

```
6670 \edef\@do@newglossaryentry{%
```

```
6671 \noexpand\newglossaryentry{\the\glslabeltok}%
```

```
6672 {%
```

```
6673 type=\acronymtype,%
```

```
6674 name={\the\glsshorttok},%
```

```
6675 sort={\the\glsshorttok},%
```

```
6676 text={\the\glsshorttok},%
```

```

6677     first={\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
6678     plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6679     firstplural={\acrfullformat{\noexpand\expandonce\noexpand\@glo@longpl}%
6680                 {\noexpand\expandonce\noexpand\@glo@shortpl}},%
6681     short={\the\glsshorttok},%
6682     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6683     long={\the\glslongtok},%
6684     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6685     description={\the\glslongtok},%
6686     descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%

```

Remaining options specified by the user:

```

6687     \the\glskeylisttok
6688 }%
6689 }%
6690 \let\@org@gls@assign@firstpl\gls@assign@firstpl
6691 \let\@org@gls@assign@plural\gls@assign@plural
6692 \let\@org@gls@assign@descplural\gls@assign@descplural
6693 \def\gls@assign@firstpl##1##2{%
6694     \@@gls@expand@field{##1}{firstpl}{##2}%
6695 }%
6696 \def\gls@assign@plural##1##2{%
6697     \@@gls@expand@field{##1}{plural}{##2}%
6698 }%
6699 \def\gls@assign@descplural##1##2{%
6700     \@@gls@expand@field{##1}{descplural}{##2}%
6701 }%
6702 \do@newglossaryentry
6703 \let\gls@assign@firstpl\@org@gls@assign@firstpl
6704 \let\gls@assign@plural\@org@gls@assign@plural
6705 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
6706 }

```

DefaultAcronymStyle Set up the default acronym style:

```

6707 \newcommand*\SetDefaultAcronymStyle{%
    Set the display style:
6708     \@for\@gls@type:=\@glsacronymlists\do{%
6709         \SetDefaultAcronymDisplayStyle{\@gls@type}%
6710     }%

```

Set up the definition of \newacronym:

```

6711 \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).

```

6712     \ifx\@glsacronymlists\@empty
6713         \def\@glo@type{\acronymtype}%
6714         \setkeys{glossentry}{##1}%
6715         \DeclareAcronymList{\@glo@type}%

```

```

6716     \SetDefaultAcronymDisplayStyle{\@glo@type}%
6717     \fi
6718     \glskeylisttok{##1}%
6719     \glslabeltok{##2}%
6720     \glsshorttok{##3}%
6721     \gslongtok{##4}%
6722     \newacronymhook
6723     \DefaultNewAcronymDef
6724 }%
6725 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6726 }

```

`\acrfootnote` Used by the footnote acronym styles.

```

6727 \newcommand*{\acrfootnote}[3]{\acrlinkfootnote{#1}{#2}{#3}}

```

`\acrlinkfootnote`

```

6728 \newcommand*{\acrlinkfootnote}[3]{%
6729   \footnote{\glslink{#1}{#2}{#3}}%
6730 }

```

`\acrnoflinkfootnote`

```

6731 \newcommand*{\acrnoflinkfootnote}[3]{%
6732   \footnote{#3}%
6733 }

```

`AcronymDisplayStyle` Sets the acronym display style for given glossary for the description and footnote combination.

```

6734 \newcommand*{\SetDescriptionFootnoteAcronymDisplayStyle}[1]{%
6735   \def\glsentryfmt{#1}%

6736   \ifdefempty\glscustomtext
6737   {%
6738     \ifglsused{\glslabel}%
6739     {%
6740       \acronymfont{\glsentryfmt}%
6741     }%
6742     {%
6743       \firstacronymfont{\glsentryfmt}%
6744       \ifglsymbol{\glslabel}%
6745       {%
6746         \expandafter\protect\expandafter\acrfootnote\expandafter
6747         {\@gls@link@opts}{\@gls@link@label}%
6748       }%
6749       \glsifplural
6750       {\glsentrysymbolplural{\glslabel}}%
6751       {\glsentrysymbol{\glslabel}}%
6752     }%
6753   }%
6754 }%

```

```

6755 }%
6756 {\glscustomtext\glsinsert}%
6757 }%
6758 }

```

otnoteNewAcronymDef

```

6759 \newcommand*{\DescriptionFootnoteNewAcronymDef}{%
6760   \edef\@do@newglossaryentry{%
6761     \noexpand\newglossaryentry{\the\glslabeltok}%
6762     {%
6763       type=\acronymtype,%
6764       name={\noexpand\acronymfont{\the\glsshorttok}},%
6765       sort={\the\glsshorttok},%
6766       first={\the\glsshorttok},%
6767       firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6768       text={\the\glsshorttok},%
6769       plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6770       short={\the\glsshorttok},%
6771       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6772       long={\the\glslongtok},%
6773       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6774       symbol={\the\glslongtok},%
6775       symbolplural={\noexpand\expandonce\noexpand\@glo@longpl},%
6776       \the\glskeylisttok
6777     }%
6778   }%
6779   \let\@org@gls@assign@firstpl\gls@assign@firstpl
6780   \let\@org@gls@assign@plural\gls@assign@plural
6781   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
6782   \def\gls@assign@firstpl##1##2{%
6783     \@@gls@expand@field{##1}{firstpl}{##2}%
6784   }%
6785   \def\gls@assign@plural##1##2{%
6786     \@@gls@expand@field{##1}{plural}{##2}%
6787   }%
6788   \def\gls@assign@symbolplural##1##2{%
6789     \@@gls@expand@field{##1}{symbolplural}{##2}%
6790   }%
6791   \@do@newglossaryentry
6792   \let\gls@assign@plural\@org@gls@assign@plural
6793   \let\gls@assign@firstpl\@org@gls@assign@firstpl
6794   \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
6795 }

```

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.

```

6796 \newcommand*{\SetDescriptionFootnoteAcronymStyle}{%

```

```

6797 \renewcommand{\newacronym}[4][\]{%
6798 \ifx\@glsacronymlists\@empty
6799 \def\@glo@type{\acronymtype}%
6800 \setkeys{glossentry}{##1}%
6801 \DeclareAcronymList{\@glo@type}%
6802 \SetDescriptionFootnoteAcronymDisplayStyle{\@glo@type}%
6803 \fi
6804 \glskeylisttok{##1}%
6805 \glslabeltok{##2}%
6806 \glsshorttok{##3}%
6807 \gslongtok{##4}%
6808 \newacronymhook
6809 \DescriptionFootnoteNewAcronymDef
6810 }%

```

If footnote package option is specified, set the first use to append the long form (stored in symbol) as a footnote.

```

6811 \@for\@gls@type:=\@glsacronymlists\do{%
6812 \SetDescriptionFootnoteAcronymDisplayStyle{\@gls@type}%
6813 }%

```

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.

```

6814 \ifglsacrsmallcaps
6815 \renewcommand*\acronymfont{[1]{\textsc{##1}}}%
6816 \renewcommand*\acrpluralsuffix{\glsupacrpluralsuffix}%
6817 \else
6818 \ifglsacrsmaller
6819 \renewcommand*\acronymfont{[1]{\textsmaller{##1}}}%
6820 \fi
6821 \fi

```

Check for package option clash

```

6822 \ifglsacrdua
6823 \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’
6824 can’t both be set}{}%
6825 \fi
6826 }%

```

AcronymDisplayStyle Sets the acronym display style for given glossary with description and dua combination.

```

6827 \newcommand*\SetDescriptionDUAAcronymDisplayStyle[1]{%
6828 \defglsentryfmt[#1]{\glsentryfmt}%
6829 }

```

ionDUANewAcronymDef

```

6830 \newcommand*\DescriptionDUANewAcronymDef{}%
6831 \edef\@do@newglossaryentry{%

```

```

6832 \noexpand\newglossaryentry{\the\glslabeltok}%
6833 {%
6834     type=\acronymtype,%
6835     name={\the\glslongtok},%
6836     sort={\the\glslongtok},
6837     text={\the\glslongtok},%
6838     first={\the\glslongtok},%
6839     plural={\noexpand\expandonce\noexpand\@glo@longpl},%
6840     firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
6841     short={\the\glsshorttok},%
6842     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6843     long={\the\glslongtok},%
6844     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6845     symbol={\the\glsshorttok},%
6846     symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6847     \the\glskeylisttok
6848 }%
6849 }%
6850 \let\@org@gls@assign@firstpl\gls@assign@firstpl
6851 \let\@org@gls@assign@plural\gls@assign@plural
6852 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
6853 \def\gls@assign@firstpl##1##2{%
6854     \@@gls@expand@field{##1}{firstpl}{##2}%
6855 }%
6856 \def\gls@assign@plural##1##2{%
6857     \@@gls@expand@field{##1}{plural}{##2}%
6858 }%
6859 \def\gls@assign@symbolplural##1##2{%
6860     \@@gls@expand@field{##1}{symbolplural}{##2}%
6861 }%
6862 \do@newglossaryentry
6863 \let\gls@assign@firstpl\@org@gls@assign@firstpl
6864 \let\gls@assign@plural\@org@gls@assign@plural
6865 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
6866 }

```

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.

```

6867 \newcommand*{\SetDescriptionDUAAcronymStyle}{%
6868     \ifglscrsmlcaps
6869         \PackageError{glossaries}{Option clash: 'smallcaps' and 'dua'
6870             can't both be set}{}%
6871     \else
6872         \ifglscrsmlaller
6873             \PackageError{glossaries}{Option clash: 'smaller' and 'dua'
6874                 can't both be set}{}%
6875     \fi
6876 \fi

```



```

6877 \renewcommand{\newacronym}[4][\]{%
6878 \ifx\@glsacronymlists\@empty
6879 \def\@glo@type{\acronymtype}%
6880 \setkeys{glossentry}{##1}%
6881 \DeclareAcronymList{\@glo@type}%
6882 \SetDescriptionDUAAcronymDisplayStyle{\@glo@type}%
6883 \fi
6884 \glskeylisttok{##1}%
6885 \glslabeltok{##2}%
6886 \glsshorttok{##3}%
6887 \glslongtok{##4}%
6888 \newacronymhook
6889 \DescriptionDUANewAcronymDef
6890 }%

Set display.
6891 \@for\@gls@type:=\@glsacronymlists\do{%
6892 \SetDescriptionDUAAcronymDisplayStyle{\@gls@type}%
6893 }%
6894 }%

```

AcronymDisplayStyle Sets the acronym display style for given glossary using the description setting (but not footnote or dua).

```

6895 \newcommand*{\SetDescriptionAcronymDisplayStyle}[1]{%
6896 \def\glsentryfmt[#1]{%

6897 \ifdefempty\glscustomtext
6898 {%
6899 \ifglsused{\glslabel}%
6900 {%

Move the inserted text outside of \acronymfont
6901 \let\gls@org@insert\glsinsert
6902 \let\glsinsert\@empty
6903 \acronymfont{\glsentryfmt}\gls@org@insert
6904 }%
6905 {%
6906 \glsentryfmt
6907 \ifglshassymbol{\glslabel}%
6908 {%
6909 \glsifplural
6910 {%
6911 \def\@glo@symbol{\glsentrysymbolplural{\glslabel}}%
6912 }%
6913 {%
6914 \def\@glo@symbol{\glsentrysymbol{\glslabel}}%
6915 }%
6916 \space\protect\firstacronymfont
6917 {\glsupcase
6918 {\@glo@symbol}

```

```

6919          {\@glo@symbol}
6920          {\mfirstucMakeUppercase{\@glo@symbol}}})%
6921      }%
6922      {}%
6923  }%
6924  }%
6925  {\glscustomtext\glsinsert}%
6926  }%
6927 }

```

ptionNewAcronymDef

```

6928 \newcommand*{\DescriptionNewAcronymDef}{%
6929   \edef\@do@newglossaryentry{%
6930     \noexpand\newglossaryentry{\the\glslabeltok}%
6931     {%
6932       type=\acronymtype,%
6933       name={\noexpand
6934         \acrnameformat{\the\glsshorttok}{\the\glslongtok}},%
6935       sort={\the\glsshorttok},%
6936       first={\the\glslongtok},%
6937       firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
6938       text={\the\glsshorttok},%
6939       plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6940       short={\the\glsshorttok},%
6941       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6942       long={\the\glslongtok},%
6943       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6944       symbol={\noexpand\@glo@text},%
6945       symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6946       \the\glskeylisttok}%
6947   }%
6948   \let\@org@gls@assign@firstpl\gls@assign@firstpl
6949   \let\@org@gls@assign@plural\gls@assign@plural
6950   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
6951   \def\gls@assign@firstpl##1##2{%
6952     \@@gls@expand@field{##1}{firstpl}{##2}%
6953   }%
6954   \def\gls@assign@plural##1##2{%
6955     \@@gls@expand@field{##1}{plural}{##2}%
6956   }%
6957   \def\gls@assign@symbolplural##1##2{%
6958     \@@gls@expand@field{##1}{symbolplural}{##2}%
6959   }%
6960   \@do@newglossaryentry
6961   \let\gls@assign@firstpl\@org@gls@assign@firstpl
6962   \let\gls@assign@plural\@org@gls@assign@plural
6963   \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
6964 }

```

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.

```

6965 \newcommand*{\SetDescriptionAcronymStyle}{%
6966   \renewcommand{\newacronym}[4][\]{%
6967     \ifx\@glsacronymlists\@empty
6968       \def\@glo@type{\acronymtype}%
6969       \setkeys{glossentry}{##1}%
6970       \DeclareAcronymList{\@glo@type}%
6971       \SetDescriptionAcronymDisplayStyle{\@glo@type}%
6972     \fi
6973     \glskeylisttok{##1}%
6974     \glslabeltok{##2}%
6975     \glsshorttok{##3}%
6976     \gslongtok{##4}%
6977     \newacronymhook
6978     \DescriptionNewAcronymDef
6979   }%

```

Set display.

```

6980   \@for\@gls@type:=\@glsacronymlists\do{%
6981     \SetDescriptionAcronymDisplayStyle{\@gls@type}%
6982   }%

```

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.

```

6983   \ifglsacrsmallcaps
6984     \renewcommand{\acronymfont}[1]{\textsc{##1}}
6985     \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6986   \else
6987     \ifglsacrsmaller
6988       \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
6989     \fi
6990   \fi
6991 }%

```

AcronymDisplayStyle Sets the acronym display style for given glossary with footnote setting (but not description or dua).

```

6992 \newcommand*{\SetFootnoteAcronymDisplayStyle}[1]{%
6993   \defglsentryfmt[#1]{%
6994     \ifdefempty\glscustomtext
6995     {%

```

Move the inserted text outside of `\acronymfont`

```

6996     \let\gls@org@insert\glsinsert
6997     \let\glsinsert\@empty
6998     \ifglsused{\glslabel}%
6999     {%

```

```

7000     \acronymfont{\glsgenentryfmt}\gls@org@insert
7001 }%
7002 {%
7003     \firstacronymfont{\glsgenentryfmt}\gls@org@insert
7004     \ifglshaslong{\glslabel}%
7005     {%
7006         \expandafter\protect\expandafter\acrfootnote\expandafter
7007         {\@gls@link@opts}{\@gls@link@label}%
7008         {%
7009             \glsifplural
7010             {\glsentrylongpl{\glslabel}}%
7011             {\glsentrylong{\glslabel}}%
7012         }%
7013     }%
7014 }%
7015 }%
7016 }%
7017 {\glscustomtext\glsinsert}%
7018 }%
7019 }

```

otnoteNewAcronymDef

```

7020 \newcommand*{\FootnoteNewAcronymDef}{%
7021     \edef\@do@newglossaryentry{%
7022         \noexpand\newglossaryentry{\the\glslabeltok}%
7023         {%
7024             type=\acronymtype,%
7025             name={\noexpand\acronymfont{\the\glsshorttok}},%
7026             sort={\the\glsshorttok},%
7027             text={\the\glsshorttok},%
7028             plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7029             first={\the\glsshorttok},%
7030             firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7031             short={\the\glsshorttok},%
7032             shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7033             long={\the\glslongtok},%
7034             longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7035             description={\the\glslongtok},%
7036             descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7037             \the\glskeylisttok
7038         }%
7039     }%
7040     \let\@org@gls@assign@plural\gls@assign@plural
7041     \let\@org@gls@assign@firstpl\gls@assign@firstpl
7042     \let\@org@gls@assign@descplural\gls@assign@descplural
7043     \def\gls@assign@firstpl##1##2{%
7044         \@gls@expand@field{##1}{firstpl}{##2}%
7045     }%
7046     \def\gls@assign@plural##1##2{%

```

```

7047 \@@gls@expand@field{##1}{plural}{##2}%
7048 }%
7049 \def\gls@assign@descplural##1##2{%
7050 \@@gls@expand@field{##1}{descplural}{##2}%
7051 }%
7052 \do@newglossaryentry
7053 \let\gls@assign@plural\@org@gls@assign@plural
7054 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7055 \let\gls@assign@descplural\@org@gls@assign@descplural
7056 }

```

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.

```

7057 \newcommand*\SetFootnoteAcronymStyle{%
7058 \renewcommand{\newacronym}[4][\]{%
7059 \ifx\@glsacronymlists\@empty
7060 \def\@glo@type{\acronymtype}%
7061 \setkeys{glossentry}{##1}%
7062 \DeclareAcronymList{\@glo@type}%
7063 \SetFootnoteAcronymDisplayStyle{\@glo@type}%
7064 \fi
7065 \glskeylisttok{##1}%
7066 \glslabeltok{##2}%
7067 \glsshorttok{##3}%
7068 \glslongtok{##4}%
7069 \newacronymhook
7070 \FootnoteNewAcronymDef
7071 }%

```

Set display

```

7072 \@for\@gls@type:=\@glsacronymlists\do{%
7073 \SetFootnoteAcronymDisplayStyle{\@gls@type}%
7074 }%

```

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.

```

7075 \ifglsacrsmallcaps
7076 \renewcommand*\acronymfont[1]{\textsc{##1}}%
7077 \renewcommand*\acrpluralsuffix{\glsupacrpluralsuffix}%
7078 \else
7079 \ifglsacrsmaller
7080 \renewcommand*\acronymfont[1]{\textsmaller{##1}}%
7081 \fi
7082 \fi

```

Check for option clash

```

7083 \ifglsacrdua
7084 \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’

```

```

7085      can't both be set}{}%
7086  \fi
7087 }%

```

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.

```

7088 \DeclareRobustCommand*\glsdoparenifnotempty}[2]{%
7089   \protected@edef\gls@tmp{#1}%
7090   \ifdefempty\gls@tmp
7091   {%
7092   {%
7093     \ifx\gls@tmp\@gls@default@value
7094     \else
7095     \space (#2{#1})%
7096     \fi
7097   }%
7098 }

```

AcronymDisplayStyle Sets the acronym display style for given glossary where neither footnote nor description is required, but smallcaps or smaller specified.

```

7099 \newcommand*\SetSmallAcronymDisplayStyle}[1]{%
7100   \def\glsentryfmt[#1]{%

7101   \ifdefempty\glscustomtext
7102   {%

```

Move the inserted text outside of `\acronymfont`

```

7103     \let\gls@org@insert\glsinsert
7104     \let\glsinsert\@empty
7105     \ifglsused{\glslabel}%
7106     {%
7107       \acronymfont{\glsentryfmt}\gls@org@insert
7108     }%
7109     {%
7110       \glsentryfmt
7111       \ifgls hassymbol{\glslabel}%
7112       {%
7113         \glsifplural
7114         {%
7115           \def\@glo@symbol{\glsentrysymbolplural{\glslabel}}%
7116         }%
7117         {%
7118           \def\@glo@symbol{\glsentrysymbol{\glslabel}}%
7119         }%
7120         \space
7121         (\gls capscase
7122         {\firstacronymfont{\@glo@symbol}}%
7123         {\firstacronymfont{\@glo@symbol}}%

```

```

7124         {\firstacronymfont{\mfirstucMakeUppercase{\@glo@symbol}}})%
7125     }%
7126     {}%
7127     }%
7128     }%
7129     {\glscustomtext\glsinsert}%
7130 }%
7131 }

```

\SmallNewAcronymDef

```

7132 \newcommand*{\SmallNewAcronymDef}{%
7133     \edef\@do@newglossaryentry{%
7134         \noexpand\newglossaryentry{\the\glslabeltok}%
7135         {%
7136             type=\acronymtype,%
7137             name={\noexpand\acronymfont{\the\glsshorttok}},%
7138             sort={\the\glsshorttok},%
7139             text={\the\glsshorttok},%

```

Default to the short plural.

```

7140         plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7141         first={\the\glslongtok},%

```

Default to the long plural.

```

7142         firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7143         short={\the\glsshorttok},%
7144         shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7145         long={\the\glslongtok},%
7146         longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7147         description={\noexpand\@glo@first},%
7148         descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7149         symbol={\the\glsshorttok},%

```

Default to the short plural.

```

7150         symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7151         \the\glskeylisttok
7152     }%
7153 }%
7154 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7155 \let\@org@gls@assign@plural\gls@assign@plural
7156 \let\@org@gls@assign@descplural\gls@assign@descplural
7157 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7158 \def\gls@assign@firstpl##1##2{%
7159     \@@gls@expand@field{##1}{firstpl}{##2}%
7160 }%
7161 \def\gls@assign@plural##1##2{%
7162     \@@gls@expand@field{##1}{plural}{##2}%
7163 }%
7164 \def\gls@assign@descplural##1##2{%
7165     \@@gls@expand@field{##1}{descplural}{##2}%

```

```

7166 }%
7167 \def\gls@assign@symbolplural##1##2{%
7168   \@@gls@expand@field{##1}{symbolplural}{##2}%
7169 }%
7170 \do@newglossaryentry
7171 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7172 \let\gls@assign@plural\@org@gls@assign@plural
7173 \let\gls@assign@descplural\@org@gls@assign@descplural
7174 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7175 }

```

`etSmallAcronymStyle` 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.

```

7176 \newcommand*\SetSmallAcronymStyle{%
7177   \renewcommand{\newacronym}[4][]{%
7178     \ifx\@glsacronymlists\@empty
7179       \def\@glo@type{\acronymtype}%
7180       \setkeys{glossentry}{##1}%
7181       \DeclareAcronymList{\@glo@type}%
7182       \SetSmallAcronymDisplayStyle{\@glo@type}%
7183     \fi
7184     \glskeylisttok{##1}%
7185     \glslabeltok{##2}%
7186     \glsshorttok{##3}%
7187     \glslongtok{##4}%
7188     \newacronymhook
7189     \SmallNewAcronymDef
7190   }%

```

Change the display since first only contains long form.

```

7191   \@for\@gls@type:=\@glsacronymlists\do{%
7192     \SetSmallAcronymDisplayStyle{\@gls@type}%
7193   }%

```

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.

```

7194   \ifglsacrsmallcaps
7195     \renewcommand*\acronymfont}[1]{\textsc{##1}}
7196     \renewcommand*\acrpluralsuffix{\glsupacrpluralsuffix}%
7197   \else
7198     \renewcommand*\acronymfont}[1]{\textsmaller{##1}}
7199   \fi

```

check for option clash

```

7200   \ifglsacrdua
7201     \ifglsacrsmallcaps
7202       \PackageError{glossaries}{Option clash: 'smallcaps' and 'dua'
7203         can't both be set}{}%
7204     \else

```



```

7205     \PackageError{glossaries}{Option clash: ‘smaller’ and ‘dua’
7206     can’t both be set}{}%
7207     \fi
7208     \fi
7209 }%

```

`\SetDUADisplayStyle` Sets the acronym display style for given glossary with dua setting.

```

7210 \newcommand*\SetDUADisplayStyle[1]{%
7211   \def\glsentryfmt[#1]{\glsentryfmt}%
7212 }

```

`\DUANewAcronymDef`

```

7213 \newcommand*\DUANewAcronymDef{%
7214   \edef\@do@newglossaryentry{%
7215     \noexpand\newglossaryentry{\the\glslabeltok}%
7216     {%
7217       type=\acronymtype,%
7218       name={\the\glsshorttok},%
7219       text={\the\glslongtok},%
7220       first={\the\glslongtok},%
7221       plural={\noexpand\expandonce\noexpand\@glo@longpl},%
7222       firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7223       short={\the\glsshorttok},%
7224       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7225       long={\the\glslongtok},%
7226       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7227       description={\the\glslongtok},%
7228       descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7229       symbol={\the\glsshorttok},%
7230       symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7231       \the\glskeylisttok
7232     }%
7233   }%
7234   \let\@org@gls@assign@firstpl\gls@assign@firstpl
7235   \let\@org@gls@assign@plural\gls@assign@plural
7236   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7237   \let\@org@gls@assign@descplural\gls@assign@descplural
7238   \def\gls@assign@firstpl##1##2{%
7239     \@@gls@expand@field{##1}{firstpl}{##2}%
7240   }%
7241   \def\gls@assign@plural##1##2{%
7242     \@@gls@expand@field{##1}{plural}{##2}%
7243   }%
7244   \def\gls@assign@symbolplural##1##2{%
7245     \@@gls@expand@field{##1}{symbolplural}{##2}%
7246   }%
7247   \def\gls@assign@descplural##1##2{%
7248     \@@gls@expand@field{##1}{descplural}{##2}%
7249   }%

```

```

7250 \@do@newglossaryentry
7251 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7252 \let\gls@assign@plural\@org@gls@assign@plural
7253 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7254 \let\gls@assign@descplural\@org@gls@assign@descplural
7255 }

```

\SetDUASStyle Always expand acronyms.

```

7256 \newcommand*{\SetDUASStyle}{%
7257   \renewcommand{\newacronym}[4][]{%
7258     \ifx\@glsacronymlists\@empty
7259       \def\@glo@type{\acronymtype}%
7260       \setkeys{glossentry}{##1}%
7261       \DeclareAcronymList{\@glo@type}%
7262       \SetDUADisplayStyle{\@glo@type}%
7263     \fi
7264     \glskeylisttok{##1}%
7265     \glslabeltok{##2}%
7266     \glsshorttok{##3}%
7267     \glslongtok{##4}%
7268     \newacronymhook
7269     \DUANewAcronymDef
7270   }%

```

Set the display

```

7271 \@for\@gls@type:=\@glsacronymlists\do{%
7272   \SetDUADisplayStyle{\@gls@type}%
7273 }%
7274 }

```

\SetAcronymStyle

```

7275 \newcommand*{\SetAcronymStyle}{%
7276   \SetDefaultAcronymStyle
7277   \ifglsacrdescription
7278     \ifglsacrfootnote
7279       \SetDescriptionFootnoteAcronymStyle
7280     \else
7281       \ifglsacrdua
7282         \SetDescriptionDUAAcronymStyle
7283       \else
7284         \SetDescriptionAcronymStyle
7285     \fi
7286   \fi
7287 \else
7288   \ifglsacrfootnote
7289     \SetFootnoteAcronymStyle
7290   \else
7291     \ifthenelse{\boolean{glsacrsmalldcaps}}{OR
7292       \boolean{glsacrsmaller}}{}%
7293   }%

```

```

7294         \SetSmallAcronymStyle
7295     }%
7296     {%
7297         \ifglssacrdua
7298         \SetDUASstyle
7299         \fi
7300     }%
7301     \fi
7302     \fi
7303 }

```

Set the acronym style according to the package options

```

7304 \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.

CustomDisplayStyle Sets the acronym display style.

```

7305 \newcommand*{\SetCustomDisplayStyle}[1]{%
7306     \defglssentryfmt[#1]{\glsgenentryfmt}%
7307 }

```

CustomAcronymFields

```

7308 \newcommand*{\CustomAcronymFields}{%
7309     name={\the\glsshorttok},%
7310     description={\the\glslongtok},%
7311     first={\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
7312     firstplural={\acrfullformat
7313         {\noexpand\glsentrylongpl{\the\glslabeltok}}}%
7314         {\noexpand\glsentryshortpl{\the\glslabeltok}}},%
7315     text={\the\glsshorttok},%
7316     plural={\the\glsshorttok\noexpand\acrpluralsuffix}%
7317 }

```

CustomNewAcronymDef

```

7318 \newcommand*{\CustomNewAcronymDef}{%
7319     \protected@edef\@do@newglossaryentry{%
7320         \noexpand\newglossaryentry{\the\glslabeltok}%
7321         {%
7322             type=\acronymtype,%
7323             short={\the\glsshorttok},%
7324             shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7325             long={\the\glslongtok},%
7326             longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7327             user1={\the\glsshorttok},%

```

```

7328     user2={\the\glsshorttok\noexpand\acrpluralsuffix},%
7329     user3={\the\glslongtok},%
7330     user4={\the\glslongtok\noexpand\acrpluralsuffix},%
7331     \CustomAcronymFields,%
7332     \the\glskeylisttok
7333 }%
7334 }%
7335 \@do@newglossaryentry
7336 }

```

`\SetCustomStyle`

```

7337 \newcommand*{\SetCustomStyle}{%
7338   \renewcommand{\newacronym}[4][]{%
7339     \ifx\@glsacronymlists\@empty
7340       \def\@glo@type{\acronymtype}%
7341       \setkeys{glossentry}{##1}%
7342       \DeclareAcronymList{\@glo@type}%
7343       \SetCustomDisplayStyle{\@glo@type}%
7344     \fi
7345     \glskeylisttok{##1}%
7346     \glslabeltok{##2}%
7347     \glsshorttok{##3}%
7348     \glslongtok{##4}%
7349     \newacronymhook
7350     \CustomNewAcronymDef
7351   }%

```

Set the display

```

7352   \@for\@gls@type:=\@glsacronymlists\do{%
7353     \SetCustomDisplayStyle{\@gls@type}%
7354   }%
7355 }

```

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.

```
7356 \RequirePackage{glossary-hypernav}
```

The styles that use list-like environments. These are not loaded if the `nolist` option is used:

```
7357 \@gls@loadlist
```

The styles that use the `longtable` environment. These are not loaded if the `no-long package` option is used.

```
7358 \@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.

```
7359 \@gls@loadsuper
```

The tree-like styles. These are not loaded if the `notree` package option is used.

```
7360 \@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.

```
7361 \ifx\@glossary@default@style\relax
```

```
7362 \else
```

```
7363   \setglossarystyle{\@glossary@default@style}
```

```
7364 \fi
```

1.20 Debugging Commands

```
\showgloparent \showgloparent{\label{}}
```

```
7365 \newcommand*{\showgloparent}[1]{%
```

```
7366   \expandafter\show\csname glo@\glsdetoklabel{#1}@parent\endcsname
```

```
7367 }
```

```
\showglolevel \showglolevel{\label{}}
```

```
7368 \newcommand*{\showglolevel}[1]{%
```

```
7369   \expandafter\show\csname glo@\glsdetoklabel{#1}@level\endcsname
```

```
7370 }
```

```
\showglotext \showglotext{\label{}}
```

```
7371 \newcommand*{\showglotext}[1]{%
```

```
7372   \expandafter\show\csname glo@\glsdetoklabel{#1}@text\endcsname
```

```
7373 }
```

```
\showgloplural \showgloplural{\label{}}
```

```
7374 \newcommand*{\showgloplural}[1]{%
```

```
7375   \expandafter\show\csname glo@\glsdetoklabel{#1}@plural\endcsname
```

```
7376 }
```

`\showglofirst` `\showglofirst{<label>}`

```
7377 \newcommand*{\showglofirst}[1]{%  
7378   \expandafter\show\csname glo@\glsdetoklabel{#1}@first\endcsname  
7379 }
```

`\showglofirstpl` `\showglofirstpl{<label>}`

```
7380 \newcommand*{\showglofirstpl}[1]{%  
7381   \expandafter\show\csname glo@\glsdetoklabel{#1}@firstpl\endcsname  
7382 }
```

`\showglotype` `\showglotype{<label>}`

```
7383 \newcommand*{\showglotype}[1]{%  
7384   \expandafter\show\csname glo@\glsdetoklabel{#1}@type\endcsname  
7385 }
```

`\showglocounter` `\showglocounter{<label>}`

```
7386 \newcommand*{\showglocounter}[1]{%  
7387   \expandafter\show\csname glo@\glsdetoklabel{#1}@counter\endcsname  
7388 }
```

`\showglouserii` `\showglouserii{<label>}`

```
7389 \newcommand*{\showglouserii}[1]{%  
7390   \expandafter\show\csname glo@\glsdetoklabel{#1}@userii\endcsname  
7391 }
```

`\showglouserii` `\showglouserii{<label>}`

```
7392 \newcommand*{\showglouserii}[1]{%  
7393   \expandafter\show\csname glo@\glsdetoklabel{#1}@userii\endcsname  
7394 }
```

\showglouseriii \showglouseriii{<label>}

```
7395 \newcommand*{\showglouseriii}[1]{%
7396   \expandafter\show\csname glo@\glsdetoklabel{#1}@useriii\endcsname
7397 }
```

\showglouseriv \showglouseriv{<label>}

```
7398 \newcommand*{\showglouseriv}[1]{%
7399   \expandafter\show\csname glo@\glsdetoklabel{#1}@useriv\endcsname
7400 }
```

\showglouserv \showglouserv{<label>}

```
7401 \newcommand*{\showglouserv}[1]{%
7402   \expandafter\show\csname glo@\glsdetoklabel{#1}@userv\endcsname
7403 }
```

\showglouservi \showglouservi{<label>}

```
7404 \newcommand*{\showglouservi}[1]{%
7405   \expandafter\show\csname glo@\glsdetoklabel{#1}@uservi\endcsname
7406 }
```

\showgloname \showgloname{<label>}

```
7407 \newcommand*{\showgloname}[1]{%
7408   \expandafter\show\csname glo@\glsdetoklabel{#1}@name\endcsname
7409 }
```

\showglodesc \showglodesc{<label>}

```
7410 \newcommand*{\showglodesc}[1]{%
7411   \expandafter\show\csname glo@\glsdetoklabel{#1}@desc\endcsname
7412 }
```

`\showglodescplural` `\showglodescplural{\langle label \rangle}`

```
7413 \newcommand*{\showglodescplural}[1]{%
7414   \expandafter\show\csname glo@\glsdetoklabel{#1}@descplural\endcsname
7415 }
```

`\showglosort` `\showglosort{\langle label \rangle}`

```
7416 \newcommand*{\showglosort}[1]{%
7417   \expandafter\show\csname glo@\glsdetoklabel{#1}@sort\endcsname
7418 }
```

`\showglosymbol` `\showglosymbol{\langle label \rangle}`

```
7419 \newcommand*{\showglosymbol}[1]{%
7420   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbol\endcsname
7421 }
```

`\showglosymbolplural` `\showglosymbolplural{\langle label \rangle}`

```
7422 \newcommand*{\showglosymbolplural}[1]{%
7423   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolplural\endcsname
7424 }
```

`\showgloshort` `\showgloshort{\langle label \rangle}`

```
7425 \newcommand*{\showgloshort}[1]{%
7426   \expandafter\show\csname glo@\glsdetoklabel{#1}@short\endcsname
7427 }
```

`\showglolong` `\showglolong{\langle label \rangle}`

```
7428 \newcommand*{\showglolong}[1]{%
7429   \expandafter\show\csname glo@\glsdetoklabel{#1}@long\endcsname
7430 }
```


`\showgloindex` `\showgloindex{<label>}`

```
7431 \newcommand*{\showgloindex}[1]{%
7432   \expandafter\show\csname glo@\glsdetoklabel{#1}@index\endcsname
7433 }
```

`\showgloflag` `\showgloflag{<label>}`

```
7434 \newcommand*{\showgloflag}[1]{%
7435   \expandafter\show\csname ifglo@\glsdetoklabel{#1}@flag\endcsname
7436 }
```

`\showgloloclist` `\showgloloclist{<label>}`

```
7437 \newcommand*{\showgloloclist}[1]{%
7438   \expandafter\show\csname glo@\glsdetoklabel{#1}@loclist\endcsname
7439 }
```

`\showglofield` `\showglofield{<label>}{<field>}`

```
7440 \newcommand*{\showglofield}[2]{%
7441   \csshow{glo@\glsdetoklabel{#1}@#2}%
7442 }
```

`\showacronymlists` `\showacronymlists`

Show list of glossaries that have been flagged as a list of acronyms.

```
7443 \newcommand*{\showacronymlists}{%
7444   \show\@glsacronymlists
7445 }
```

`\showglossaries` `\showglossaries`

Show list of defined glossaries.

```
7446 \newcommand*{\showglossaries}{%
7447   \show\@glo@types
7448 }
```

`\showglossaryin` `\showglossaryin{<glossary-label>}`

Show the ‘in’ extension for the given glossary.

```
7449 \newcommand*{\showglossaryin}[1]{%
7450   \expandafter\show\csname @glotype@#1@in\endcsname
7451 }
```

`\showglossaryout` `\showglossaryout{<glossary-label>}`

Show the ‘out’ extension for the given glossary.

```
7452 \newcommand*{\showglossaryout}[1]{%
7453   \expandafter\show\csname @glotype@#1@out\endcsname
7454 }
```

`\showglossarytitle` `\showglossarytitle{<glossary-label>}`

Show the title for the given glossary.

```
7455 \newcommand*{\showglossarytitle}[1]{%
7456   \expandafter\show\csname @glotype@#1@title\endcsname
7457 }
```

`\showglossarycounter` `\showglossarycounter{<glossary-label>}`

Show the counter for the given glossary.

```
7458 \newcommand*{\showglossarycounter}[1]{%
7459   \expandafter\show\csname @glotype@#1@counter\endcsname
7460 }
```

`\showglossaryentries` `\showglossaryentries{<glossary-label>}`

Show the list of entry labels for the given glossary.

```
7461 \newcommand*{\showglossaryentries}[1]{%
7462   \expandafter\show\csname glolist@#1\endcsname
7463 }
```

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.

```
7464 \csname ifglscompatible-2.07\endcsname
7465 \RequirePackage{glossaries-compatible-207}
7466 \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.

```
7467 \NeedsTeXFormat{LaTeX2e}
7468 \ProvidesPackage{glossaries-prefix}[2015/07/08 v4.16 (NLCT)]
```

Pass all options to glossaries:

```
7469 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
7470 \ProcessOptions
```

Load glossaries:

```
7471 \RequirePackage{glossaries}
```

Add the new keys:

```
7472 \define@key{glossentry}{prefixfirst}{\def\@glo@entryprefixfirst{#1}}%
7473 \define@key{glossentry}{prefixfirstplural}{\def\@glo@entryprefixfirstplural{#1}}%
7474 \define@key{glossentry}{prefix}{\def\@glo@entryprefix{#1}}%
7475 \define@key{glossentry}{prefixplural}{\def\@glo@entryprefixplural{#1}}%
```

Add them to `\@gls@keymap`:

```
7476 \appto\@gls@keymap{,%
7477   {prefixfirst}{prefixfirst},%
7478   {prefixfirstplural}{prefixfirstplural},%
7479   {prefix}{prefix},%
7480   {prefixplural}{prefixplural}}%
7481 }
```

Set the default values:

```
7482 \appto\@newglossaryentryprehook{%
7483   \def\@glo@entryprefix{}%
7484   \def\@glo@entryprefixplural{}}%
```

```

7485 \let\@glo@entryprefixfirst\@gls@default@value
7486 \let\@glo@entryprefixfirstplural\@gls@default@value
7487 }

```

Set the assignment code:

```

7488 \appto\@newglossaryentryposthook{%
7489   \gls@assign@field{\@glo@label}{prefix}{\@glo@entryprefix}%
7490   \gls@assign@field{\@glo@label}{prefixplural}{\@glo@entryprefixplural}%

```

If prefixfirst has not been supplied, make it the same as prefix.

```

7491   \expandafter\gls@assign@field\expandafter
7492   {\csname glo@\@glo@label @prefix\endcsname}{\@glo@label}{prefixfirst}%
7493   {\@glo@entryprefixfirst}%

```

If prefixfirstplural has not been supplied, make it the same as prefixplural.

```

7494   \expandafter\gls@assign@field\expandafter
7495   {\csname glo@\@glo@label @prefixplural\endcsname}{\@glo@label}%
7496   {prefixfirstplural}{\@glo@entryprefixfirstplural}%
7497 }

```

Define commands to access these fields:

```

\glsentryprefixfirst
7498 \newcommand*\{glsentryprefixfirst}[1]{\csuse{glo@#1@prefixfirst}}

\glsentryprefixfirstplural
7499 \newcommand*\{glsentryprefixfirstplural}[1]{\csuse{glo@#1@prefixfirstplural}}

\glsentryprefix
7500 \newcommand*\{glsentryprefix}[1]{\csuse{glo@#1@prefix}}

\glsentryprefixplural
7501 \newcommand*\{glsentryprefixplural}[1]{\csuse{glo@#1@prefixplural}}

```

Now for the initial upper case variants:

```

\Glsentryprefixfirst
7502 \newrobustcmd*\{Glsentryprefixfirst}[1]{%
7503   \protected@edef\@glo@text{\csname glo@#1@prefixfirst\endcsname}%
7504   \xmakefirstuc\@glo@text
7505 }

\Glsentryprefixfirstplural
7506 \newrobustcmd*\{Glsentryprefixfirstplural}[1]{%
7507   \protected@edef\@glo@text{\csname glo@#1@prefixfirstplural\endcsname}%
7508   \xmakefirstuc\@glo@text
7509 }

```

\Glsentryprefix

```
7510 \newrobustcmd*{\Glsentryprefix}[1]{%
7511   \protected@edef\@glo@text{\csname glo@#1@prefix\endcsname}%
7512   \xmakefirstuc\@glo@text
7513 }
```

lentryprefixplural

```
7514 \newrobustcmd*{\Glsentryprefixplural}[1]{%
7515   \protected@edef\@glo@text{\csname glo@#1@prefixplural\endcsname}%
7516   \xmakefirstuc\@glo@text
7517 }
```

Define commands to determine if the prefix keys have been set:

\ifglshasprefix

```
7518 \newcommand*{\ifglshasprefix}[3]{%
7519   \ifcempty{glo@#1@prefix}%
7520   {#3}%
7521   {#2}%
7522 }
```

ifglshasprefixplural

```
7523 \newcommand*{\ifglshasprefixplural}[3]{%
7524   \ifcempty{glo@#1@prefixplural}%
7525   {#3}%
7526   {#2}%
7527 }
```

ifglshasprefixfirst

```
7528 \newcommand*{\ifglshasprefixfirst}[3]{%
7529   \ifcempty{glo@#1@prefixfirst}%
7530   {#3}%
7531   {#2}%
7532 }
```

asprefixfirstplural

```
7533 \newcommand*{\ifglshasprefixfirstplural}[3]{%
7534   \ifcempty{glo@#1@prefixfirstplural}%
7535   {#3}%
7536   {#2}%
7537 }
```

Define commands that insert the prefix before commands like \gls:

\pgls

```
7538 \newrobustcmd{\pgls}{\@gls@hyp@opt\@pgls}
```

\@pgls Unstarred version.

```
7539 \newcommand*\@pgls}[2] [] {%
7540   \new@ifnextchar [%
7541     {\@pgls@{#1}{#2}}%
7542     {\@pgls@{#1}{#2} []}%
7543 }
```

\@pgls@ Read in the final optional argument:

```
7544 \def\@pgls@#1#2[#3] {%
7545   \glsdoifexists{#2}%
7546   {%
7547     \ifglsused{#2}%
7548     {%
7549       \glsentryprefix{#2}%
7550     }%
7551     {%
7552       \glsentryprefixfirst{#2}%
7553     }%
7554     \@gls@{#1}{#2}[#3]%
7555   }%
7556 }
```

Similarly for the plural version:

\pglsp1

```
7557 \newrobustcmd{\pglsp1}{\@gls@hyp@opt\@pglsp1}
```

\@pglsp1 Unstarred version.

```
7558 \newcommand*\@pglsp1}[2] [] {%
7559   \new@ifnextchar [%
7560     {\@pglsp1@{#1}{#2}}%
7561     {\@pglsp1@{#1}{#2} []}%
7562 }
```

\@pglsp1@ Read in the final optional argument:

```
7563 \def\@pglsp1@#1#2[#3] {%
7564   \glsdoifexists{#2}%
7565   {%
7566     \ifglsused{#2}%
7567     {%
7568       \glsentryprefixplural{#2}%
7569     }%
7570     {%
7571       \glsentryprefixfirstplural{#2}%
7572     }%
7573     \@glspl@{#1}{#2}[#3]%
7574   }%
7575 }
```

Now for the first letter upper case versions:

\Pgl's

```
7576 \newrobustcmd{\Pgl's}{\@gls@hyp@opt\@Pgl's}
```

\@Pgl's Unstarred version.

```
7577 \newcommand*{\@Pgl's}[2][ ]{%
7578   \new@ifnextchar[%
7579     {\@Pgl's@{#1}{#2}}}%
7580   {\@Pgl's@{#1}{#2}[ ]}%
7581 }
```

\@Pgl's@ Read in the final optional argument:

```
7582 \def\@Pgl's@#1#2[#3]{%
7583   \glsdoifexists{#2}%
7584   {%
7585     \ifglsused{#2}%
7586     {%
7587       \ifgls hasprefix{#2}%
7588       {%
7589         \Glsentryprefix{#2}%
7590         \@gls@{#1}{#2}[#3]%
7591       }%
7592       {\@Gls@{#1}{#2}[#3]}%
7593     }%
7594     {%
7595       \ifgls hasprefixfirst{#2}%
7596       {%
7597         \Glsentryprefixfirst{#2}%
7598         \@gls@{#1}{#2}[#3]%
7599       }%
7600       {\@Gls@{#1}{#2}[#3]}%
7601     }%
7602   }%
7603 }
```

Similarly for the plural version:

\Pgl'spl

```
7604 \newrobustcmd{\Pgl'spl}{\@gls@hyp@opt\@Pgl'spl}
```

\@Pgl'spl Unstarred version.

```
7605 \newcommand*{\@Pgl'spl}[2][ ]{%
7606   \new@ifnextchar[%
7607     {\@Pgl'spl@{#1}{#2}}}%
7608   {\@Pgl'spl@{#1}{#2}[ ]}%
7609 }
```

\@Pglsp1@ Read in the final optional argument:

```
7610 \def\@Pglsp1@#1#2[#3]{%
7611   \glsdoifexists{#2}%
7612   {%
7613     \ifglsused{#2}%
7614     {%
7615       \ifglshasprefixplural{#2}%
7616       {%
7617         \Glsentryprefixplural{#2}%
7618         \@glsp1@{#1}{#2}[#3]%
7619       }%
7620       {\@Glspl@{#1}{#2}[#3]}%
7621     }%
7622     {%
7623       \ifglshasprefixfirstplural{#2}%
7624       {%
7625         \Glsentryprefixfirstplural{#2}%
7626         \@glsp1@{#1}{#2}[#3]%
7627       }%
7628       {\@Glspl@{#1}{#2}[#3]}%
7629     }%
7630   }%
7631 }
```

Finally the all upper case versions:

\PGLS

```
7632 \newrobustcmd{\PGLS}{\@gls@hyp@opt\@PGLS}
```

\@PGLS Unstarred version.

```
7633 \newcommand*{\@PGLS}[2][ ]{%
7634   \new@ifnextchar[%
7635   {\@PGLS@{#1}{#2}}%
7636   {\@PGLS@{#1}{#2}[]}%
7637 }
```

\@PGLS@ Read in the final optional argument:

```
7638 \def\@PGLS@#1#2[#3]{%
7639   \glsdoifexists{#2}%
7640   {%
7641     \ifglsused{#2}%
7642     {%
7643       \mfirstucMakeUppercase{\glsentryprefix{#2}}%
7644     }%
7645     {%
7646       \mfirstucMakeUppercase{\glsentryprefixfirst{#2}}%
7647     }%
7648     \@GLS@{#1}{#2}[#3]%
7649   }
```



```

7649 }%
7650 }

```

Plural version:

`\PGLSp1`

```

7651 \newrobustcmd{\PGLSp1}{\@gls@hyp@opt\PGLSp1}

```

`\@PGLSp1` Unstarred version.

```

7652 \newcommand*{\@PGLSp1}[2][{}]{%
7653   \new@ifnextchar[%
7654     {\@PGLSp1@{#1}{#2}}%
7655     {\@PGLSp1@{#1}{#2}[]}%
7656 }

```

`\@PGLSp1@` Read in the final optional argument:

```

7657 \def\@PGLSp1@#1#2[#3]{%
7658   \glsdoifexists{#2}%
7659   {%
7660     \ifglsused{#2}%
7661     {%
7662       \mfirstucMakeUppercase{\glsentryprefixplural{#2}}%
7663     }%
7664     {%
7665       \mfirstucMakeUppercase{\glsentryprefixfirstplural{#2}}%
7666     }%
7667     \@GLSp1@{#1}{#2}[#3]%
7668   }%
7669 }

```

3 Mfirstuc Documented Code

```

7670 \NeedsTeXFormat{LaTeX2e}
7671 \ProvidesPackage{mfirstuc}[2015/02/03 v1.10 (NLCT)]

```

Requires etoolbox:

```

7672 \RequirePackage{etoolbox}

```

`\makefirstuc` Syntax:

`\makefirstuc{<text>}`

Makes the first letter uppercase, but will skip initial control sequences if they are followed by a group and make the first thing in the group uppercase, unless the group is empty. Thus `\makefirstuc{abc}` will produce: `Abc`, `\makefirstuc{\ae bc}` will produce: `Æbc`, but `\makefirstuc{\emph{abc}}` will produce `Abc`. This is required by `\Gls` and `\Glspl`.

```

7673 \newif\if@glscs

```

```

7674 \newtoks\@glsmfirst
7675 \newtoks\@glsmrest
7676 \newrobustcmd*{\makefirstuc}[1]{%
7677   \def\@gls@argi{#1}%
7678   \ifx\@gls@argi\@empty

       If the argument is empty, do nothing.
7679   \else

7680     \def\@gls@tmp{\ #1}%
7681     \@onelevel@sanitize\@gls@tmp
7682     \expandafter\@gls@checkcs\@gls@tmp\relax\relax
7683     \if@glscs
7684       \@gls@getbody #1{}\@nil
7685       \ifx\@gls@rest\@empty
7686         \glsmakefirstuc{#1}%
7687       \else
7688         \expandafter\@gls@split\@gls@rest\@nil
7689         \ifx\@gls@first\@empty
7690           \glsmakefirstuc{#1}%
7691         \else
7692           \expandafter\@glsmfirst\expandafter{\@gls@first}%
7693           \expandafter\@glsmrest\expandafter{\@gls@rest}%
7694           \edef\@gls@domfirstuc{\noexpand\@gls@body
7695             {\noexpand\glsmakefirstuc\the\@glsmfirst}%
7696             \the\@glsmrest}%
7697           \@gls@domfirstuc
7698         \fi
7699       \fi
7700     \else
7701       \glsmakefirstuc{#1}%
7702     \fi
7703   \fi
7704 }

       Put first argument in \@gls@first and second argument in \@gls@rest:
7705 \def\@gls@split#1#2\@nil{%
7706   \def\@gls@first{#1}\def\@gls@rest{#2}%
7707 }

7708 \def\@gls@checkcs#1 #2#3\relax{%
7709   \def\@gls@argi{#1}\def\@gls@argii{#2}%
7710   \ifx\@gls@argi\@gls@argii
7711     \@glscstrue
7712   \else
7713     \@glscsfalse
7714   \fi
7715 }

```

\@gls@makefirstuc Make first thing upper case:

```

7716 \def\@gls@makefirstuc#1{\mfirstucMakeUppercase #1}

```

irstucMakeUppercase Allow user to replace \MakeUppercase with another case changing command.

```
7717 \newcommand*{\mfirstucMakeUppercase}{\MakeUppercase}
```

\glsmakefirstuc Provide a user command to make it easier to customise.

```
7718 \newcommand*{\glsmakefirstuc}[1]{\@gls@makefirstuc{#1}}
```

Get the first grouped argument and store in \@gls@body.

```
7719 \def\@gls@getbody#1#\def\@gls@body{#1}\@gls@gobbletonil}
```

Scoup up everything to \@nil and store in \@gls@rest:

```
7720 \def\@gls@gobbletonil#1\@nil{\def\@gls@rest{#1}}
```

\xmakefirstuc Expand argument once before applying \makefirstuc (added v1.01).

```
7721 \newcommand*{\xmakefirstuc}[1]{%
```

```
7722 \expandafter\makefirstuc\expandafter{#1}}
```

\emakefirstuc Fully expand argument before applying \makefirstuc

```
7723 \DeclareRobustCommand*{\emakefirstuc}[1]{%
```

```
7724 \protected@edef\@MFU@caparg{#1}%
```

```
7725 \expandafter\makefirstuc\expandafter{\@MFU@caparg}%
```

```
7726 }
```

\capitalisewords Capitalise each word in the argument. Words are considered to be separated by plain spaces (i.e. non-breakable spaces won't be considered a word break).

```
7727 \newrobustcmd*{\capitalisewords}[1]{%
```

```
7728 \def\gls@add@space{}%
```

```
7729 \let\@mfu@domakefirstuc\makefirstuc
```

```
7730 \let\@mfu@checkword\@gobble
```

```
7731 \mfu@capitalisewords#1 \@nil\mfu@endcap
```

```
7732 }
```

```
7733 \def\mfu@capitalisewords#1 #2\mfu@endcap{%
```

```
7734 \def\mfu@cap@first{#1}%
```

```
7735 \def\mfu@cap@second{#2}%
```

```
7736 \gls@add@space
```

```
7737 \@mfu@checkword{#1}%
```

```
7738 \@mfu@domakefirstuc{#1}%
```

```
7739 \def\gls@add@space{ }%
```

```
7740 \ifx\mfu@cap@second\@nnil
```

```
7741 \let\next\mfu@cap\mfu@noop
```

```
7742 \else
```

```
7743 \let\next\mfu@cap\mfu@capitalisewords
```

```
7744 \let\@mfu@checkword\mfu@checkword
```

```
7745 \fi
```

```
7746 \next\mfu@cap#2\mfu@endcap
```

```
7747 }
```

```
7748 \def\mfu@noop#1\mfu@endcap{}
```

`\mfu@checkword` Check if word should be capitalised.

```
7749 \newcommand*\mfu@checkword[1]{%
7750   \ifinlist{#1}{\@mfu@nocaplist}%
7751   {%
7752     \let\@mfu@domakefirstuc\@firstofone
7753   }%
7754   {%
7755     \let\@mfu@domakefirstuc\makefirstuc
7756   }%
7757 }
```

`\@mfu@nocaplist` List of words that shouldn't be capitalised.

```
7758 \newcommand*\@mfu@nocaplist{}
```

`\MFUnocap` Provide the user with a means to add a word to the list.

```
7759 \newcommand*\MFUnocap[1]{\listadd{\@mfu@nocaplist}{#1}}
```

`\gMFUnocap` Global version.

```
7760 \newcommand*\gMFUnocap[1]{\listgadd{\@mfu@nocaplist}{#1}}
```

`\MFUclear` Clear the list

```
7761 \newcommand*\MFUclear{\renewcommand*\@mfu@nocaplist{}}
```

`\xcapitalisewords` Short-cut command:

```
7762 \newcommand*\xcapitalisewords[1]{%
7763   \expandafter\capitalisewords\expandafter{#1}%
7764 }
```

`\ecapitalisewords` Fully expand argument before applying `\capitalisewords`

```
7765 \DeclareRobustCommand*\ecapitalisewords[1]{%
7766   \protected@edef\@MFU@caparg{#1}%
7767   \expandafter\capitalisewords\expandafter{\@MFU@caparg}%
7768 }
```

4 Mfirstuc-english Documented Code

```
7769 \NeedsTeXFormat{LaTeX2e}
7770 \ProvidesPackage{mfirstuc-english}[2014/07/30 v1.0 (NLCT)]
```

Load mfirstuc if not already loaded:

```
7771 \RequirePackage{mfirstuc}
```

Add no-cap words. (List isn't a complete list.)

```
7772 \MFUnocap{a}
7773 \MFUnocap{an}
7774 \MFUnocap{and}
7775 \MFUnocap{but}
7776 \MFUnocap{for}
```

```

7777 \MFUnocap{in}
7778 \MFUnocap{of}
7779 \MFUnocap{or}
7780 \MFUnocap{no}
7781 \MFUnocap{nor}
7782 \MFUnocap{so}
7783 \MFUnocap{some}
7784 \MFUnocap{the}
7785 \MFUnocap{with}
7786 \MFUnocap{yet}

```

5 Glossary Styles

5.1 Glossary hyper-navigation definitions (glossary-hypernav package)

Package Definition:

```

7787 \ProvidesPackage{glossary-hypernav}[2015/07/08 v4.16 (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
```

```

7788 \newcommand*{\glsnavhyperlink}[3][\@glo@type]{%
7789   \edef\gls@grplabel{#2}\protected@edef\gls@grptitle{#3}%
7790   \@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
```

```

7791 \newcommand*{\glsnavhypertarget}[3][\@glo@type]{%
  Add this group to the aux file for re-run check.
7792   \protected@write\@auxout{}{\string\@gls@hypergroup{#1}{#2}}%
  Add the target.
7793   \@glstarget{glsn:#1@#2}{#3}%
  Check list of know groups to determine if a re-run is required.

```

```

7794 \expandafter\let
7795 \expandafter\@gls@list\csname @gls@hypergrouplist@#1\endcsname

Iterate through list and terminate loop if this group is found.
7796 \@for\@gls@elem:=\@gls@list\do{%
7797 \ifthenelse{\equal{\@gls@elem}{#2}}{\@endfortrue}{}}%

Check if list terminated prematurely.
7798 \if@endfor
7799 \else

This group was not included in the list, so issue a warning.
7800 \GlossariesWarningNoLine{Navigation panel
7801 for glossary type ‘#1’^^Jmissing group ‘#2’}%
7802 \gdef\gls@hypergroupprerun{%
7803 \GlossariesWarningNoLine{Navigation panel
7804 has changed. Rerun LaTeX}}%
7805 \fi
7806 }

```

`\gls@hypergroupprerun` Give a warning at the end if re-run required

```

7807 \let\gls@hypergroupprerun\relax
7808 \AtEndDocument{\gls@hypergroupprerun}

```

`\@gls@hypergroup` This adds to (or creates) the command `\@gls@hypergrouplist@{<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.

```

7809 \newcommand*{\@gls@hypergroup}[2]{%
7810 \@ifundefined{\@gls@hypergrouplist@#1}{%
7811 \expandafter\xdef\csname @gls@hypergrouplist@#1\endcsname{#2}%
7812 }{%
7813 \expandafter\let\expandafter\@gls@tmp
7814 \csname @gls@hypergrouplist@#1\endcsname
7815 \expandafter\xdef\csname @gls@hypergrouplist@#1\endcsname{%
7816 \@gls@tmp,#2}%
7817 }%
7818 }

```

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`

```

7819 \newcommand*{\glsnavigation}{%
7820 \def\@gls@between{}%

```

```

7821 \ifundefined{@gls@hypergrouplist@{@glo@type}}{%
7822   \def@gls@list{%
7823 }{%
7824   \expandafter\let\expandafter@gls@list
7825     \csname @gls@hypergrouplist@{@glo@type}\endcsname
7826 }%
7827 \@for@gls@tmp:=\@gls@list\do{%
7828   \gls@between

7829   \@gls@getgrouptitle{\@gls@tmp}{\@gls@grptitle}%
7830   \glsnavhyperlink{\@gls@tmp}{\@gls@grptitle}%
7831   \let@gls@between\glshypernavsep%
7832 }%
7833 }

```

`\glshypernavsep` Separator for the hyper navigation bar.

```

7834 \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`

```

7835 \newcommand*{\glssymbolnav}{%
7836 \glsnavhyperlink{glssymbols}{\glsgetgrouptitle{glssymbols}}%
7837 \glshypernavsep
7838 \glsnavhyperlink{glsnumbers}{\glsgetgrouptitle{glsnumbers}}%
7839 \glshypernavsep
7840 }

```

5.2 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.

```

7841 \ProvidesPackage{glossary-inline}[2015/07/08 v4.16 (NLCT)]

```

`inline` Define the inline style.

```

7842 \newglossarystyle{inline}{%

```

Start of glossary sets up first empty separator between entries. (This is then changed by `\glossentry`)

```

7843   \renewenvironment{theglossary}%
7844     {%
7845       \def@gls@inlinesep{}%
7846       \def@gls@inlinesubsep{}%
7847       \def@gls@inlinepostchild{}%
7848     }%
7849     {\glspostinline}%

```

No header:

```
7850 \renewcommand*{\glossaryheader}{}%
```

No group headings (if heading is required, add `\glsinlinedopostchild` to start definition in case heading follows a child entry):

```
7851 \renewcommand*{\glsgroupheading}[1]{}%
```

Just display separator followed by name and description:

```
7852 \renewcommand{\glossentry}[2]{%
7853   \glsinlinedopostchild
7854   \gls@inlinesep
7855   \glsentryitem{##1}%
7856   \glsinlinenameformat{##1}{%
7857     \glossentryname{##1}%
7858   }%
7859   \ifglshasdescsuppressed{##1}%
7860   {%
7861     \glsinlineemptydescformat
7862     {%
7863       \glossentrysymbol{##1}%
7864     }%
7865     {%
7866       ##2%
7867     }%
7868   }%
7869   {%
7870     \ifglshasdesc{##1}%
7871     {\glsinlinedescformat{\glossentrydesc{##1}}{\glossentrysymbol{##1}}{##2}}%
7872     {\glsinlineemptydescformat{\glossentrysymbol{##1}}{##2}}%
7873   }%
7874   \ifglshaschildren{##1}%
7875   {%
7876     \glsresetsubentrycounter
7877     \glsinlineparentchildseparator
7878     \def\gls@inlinesubsep{}%
7879     \def\gls@inlinepostchild{\glsinlinepostchild}%
7880   }%
7881   {%
7882     \def\gls@inlinesep{\glsinlineseparator}%
7883   }%
```

Sub-entries display description:

```
7884 \renewcommand{\subglossentry}[3]{%
7885   \gls@inlinesubsep%
7886   \glsinlinesubnameformat{##2}{%
7887     \glossentryname{##2}}%
7888   \glssubentryitem{##2}%
7889   \glsinlinesubdescformat{\glossentrydesc{##2}}{\glossentrysymbol{##2}}{##3}%
7890   \def\gls@inlinesubsep{\glsinlinesubseparator}%
7891   }%
```


Nothing special between groups:

```
7892 \renewcommand*{\glsgroupskip}{}%  
7893 }
```

`\glsinlinedopostchild`

```
7894 \newcommand*{\glsinlinedopostchild}{%  
7895     \gls@inlinepostchild  
7896     \def\gls@inlinepostchild{}%  
7897 }
```

`\glsinlineseparator` Separator to use between entries.

```
7898 \newcommand*{\glsinlineseparator}{;\space}
```

`\glsinlinesubseparator` Separator to use between sub-entries.

```
7899 \newcommand*{\glsinlinesubseparator}{,\space}
```

`\glsinlineparentchildseparator` Separator to use between parent and children.

```
7900 \newcommand*{\glsinlineparentchildseparator}{:\space}
```

`\glsinlinepostchild` Hook to use between child and next entry

```
7901 \newcommand*{\glsinlinepostchild}{}
```

`\glspostinline` Terminator for inline glossary.

```
7902 \newcommand*{\glspostinline}{\glspostdescription\space}
```

`\glsinlinenameformat` Formats the name of the entry (first argument label, second argument name):

```
7903 \newcommand*{\glsinlinenameformat}[2]{\glstarget{#1}{#2}}
```

`\glsinlinedescformat` Formats the entry's description, symbol and location list:

```
7904 \newcommand*{\glsinlinedescformat}[3]{\space#1}
```

`\glsinlineemptydescformat` Formats the entry's symbol and location list when the description is empty:

```
7905 \newcommand*{\glsinlineemptydescformat}[2]{}
```

`\glsinlinesubnameformat` Formats the name of the subentry (first argument label, second argument name):

```
7906 \newcommand*{\glsinlinesubnameformat}[2]{\glstarget{#1}{}}
```

`\glsinlinesubdescformat` Formats the subentry's description, symbol and location list:

```
7907 \newcommand*{\glsinlinesubdescformat}[3]{#1}
```

5.3 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.

```
7908 \ProvidesPackage{glossary-list}[2015/07/08 v4.16 (NLCT)]
```

`\indexspace` The are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```
7909 \providecommand{\indexspace}{%
7910   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
7911 }
```

`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.

```
7912 \newglossarystyle{list}{%
```

Use description environment:

```
7913   \renewenvironment{theglossary}%
7914     {\begin{description}}{\end{description}}%
```

No header at the start of the environment:

```
7915   \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
7916   \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries start a new item in the list:

```
7917   \renewcommand*{\glossentry}[2]{%
7918     \item[\glssentryitem{##1}]%
7919       \glstarget{##1}{\glossentryname{##1}}]
7920     \glossentrydesc{##1}\glspostdescription\space ##2}%
```

Sub-entries continue on the same line:

```
7921   \renewcommand*{\subglossentry}[3]{%
7922     \glssubentryitem{##2}%
7923     \glstarget{##2}{\strut}%
7924     \glossentrydesc{##2}\glspostdescription\space ##3.}%
7925 %   \end{macrocode}
7926 % Add vertical space between groups:
7927 %\changes{3.03}{2012/09/21}{added check for glsnogroupskip}
7928 %   \begin{macrocode}
7929   \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
7930 }
```

`listgroup` The listgroup style is like the list style, but the glossary groups have headings.

```
7931 \newglossarystyle{listgroup}{%
```

Base it on the list style:

```
7932 \setglossarystyle{list}%
```

Each group has a heading:

```
7933 \renewcommand*{\glsgroupheading}[1]{\item[\glsgetgrouptitle{##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.

```
7934 \newglossarystyle{listhypergroup}{%
```

Base it on the list style:

```
7935 \setglossarystyle{list}%
```

Add navigation links at the start of the environment:

```
7936 \renewcommand*{\glossaryheader}{%
```

```
7937 \item[\glsnavigation]]%
```

Each group has a heading with a hypertarget:

```
7938 \renewcommand*{\glsgroupheading}[1]{%
```

```
7939 \item[\glsnavhypertarget{##1}{\glsgetgrouptitle{##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.

```
7940 \newglossarystyle{altlist}{%
```

Base it on the list style:

```
7941 \setglossarystyle{list}%
```

Main (level 0) entries start a new item in the list with a line break after the entry name:

```
7942 \renewcommand*{\glossentry}[2]{%
```

```
7943 \item[\glsentryitem{##1}%
```

```
7944 \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.

```
7945 \mbox{}\par\nobreak\@afterheading
```

```
7946 \glossentrydesc{##1}\glspostdescription\space ##2}%
```

Sub-entries start a new paragraph:

```
7947 \renewcommand{\subglossentry}[3]{%
```

```
7948 \par
```

```
7949 \glssubentryitem{##2}%
```

```
7950 \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space ##3}%
```

```
7951 }
```

altlistgroup The altlistgroup glossary style is like the altlist style, but the glossary groups have headings.

```
7952 \newglossarystyle{altlistgroup}{%
```

Base it on the altlist style:

```
7953 \setglossarystyle{altlist}%
```

Each group has a heading:

```
7954 \renewcommand*{\glsgroupheading}[1]{\item[\glsgetgrouptitle{##1}]}
```

altlisthypergroup The altlisthypergroup glossary style is like the altlistgroup style, but has a set of links to the groups at the start of the glossary.

```
7955 \newglossarystyle{altlisthypergroup}{%
```

Base it on the altlist style:

```
7956 \setglossarystyle{altlist}%
```

Add navigation links at the start of the environment:

```
7957 \renewcommand*{\glossaryheader}{%
```

```
7958 \item[\glsnavigation]}%
```

Each group has a heading with a hypertarget:

```
7959 \renewcommand*{\glsgroupheading}[1]{%
```

```
7960 \item[\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}]}
```

listdotted The listdotted glossary style was supplied by Axel Menzel. I've modified it slightly so that the distance from the start of the name to the end of the dotted line is specified by `\glslistdottedwidth`. Note that this style ignores the page numbers as well as the symbol. Sub-entries are displayed in the same way as top-level entries.

```
7961 \newglossarystyle{listdotted}{%
```

Base it on the list style:

```
7962 \setglossarystyle{list}%
```

Each main (level 0) entry starts a new item:

```
7963 \renewcommand*{\glossentry}[2]{%
```

```
7964 \item[]\makebox[\glslistdottedwidth][l]{%
```

```
7965 \glsentryitem{##1}%
```

```
7966 \glstarget{##1}{\glossentryname{##1}}%
```

```
7967 \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##1}}%
```

Sub entries have the same format as main entries:

```
7968 \renewcommand*{\subglossentry}[3]{%
```

```
7969 \item[]\makebox[\glslistdottedwidth][l]{%
```

```
7970 \glssubentryitem{##2}%
```

```
7971 \glstarget{##2}{\glossentryname{##2}}%
```

```
7972 \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##2}}%
```

```
7973 }
```

`\glslistdottedwidth`

```
7974 \newlength\glslistdottedwidth
```

```
7975 \setlength{\glslistdottedwidth}{.5\hsize}
```

`sublistdotted` This style is similar to the `glostylelistdotted` style, except that the main entries just have the name displayed.

```

7976 \newglossarystyle{sublistdotted}{%
    Base it on the listdotted style:
7977   \setglossarystyle{listdotted}%
    Main (level 0) entries just display the name:
7978   \renewcommand*{\glossentry}[2]{%
7979     \item[\glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}}]}%
7980 }
```

5.4 Glossary Styles using longtable (the glossary-long package)

The glossary styles defined in the package used the `longtable` environment in the glossary.

```

7981 \ProvidesPackage{glossary-long}[2015/07/08 v4.16 (NLCT)]
```

Requires the package:

```

7982 \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`.)

```

7983 \@ifundefined{glsdescwidth}{%
7984   \newlength{glsdescwidth}
7985   \setlength{glsdescwidth}{0.6\hsize}
7986 }{}
```

`\glspagelistwidth` This is a length that governs the width of the page list column.

```

7987 \@ifundefined{glspagelistwidth}{%
7988   \newlength{glspagelistwidth}
7989   \setlength{glspagelistwidth}{0.1\hsize}
7990 }{}
```

`long` The long glossary style command which uses the `longtable` environment:

```

7991 \newglossarystyle{long}{%
    Use longtable with two columns:
7992   \renewenvironment{theglossary}%
7993     {\begin{longtable}\lp{glsdescwidth}}%
7994     {\end{longtable}}%
    Do nothing at the start of the environment:
7995   \renewcommand*{\glossaryheader}{}%
    No heading between groups:
7996   \renewcommand*{\glsgroupheading}[1]{}
```

Main (level 0) entries displayed in a row:

```
7997 \renewcommand{\glossentry}[2]{%
7998   \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
7999   \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline
8000 }%
```

Sub entries displayed on the following row without the name:

```
8001 \renewcommand{\subglossentry}[3]{%
8002   &
8003   \glssubentryitem{##2}%
8004   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
8005   ##3\tabularnewline
8006 }%
```

Blank row between groups:

```
8007 \renewcommand*{\glsgroupskip}{\ifglsgnোগroupskip\else &
8008 \tabularnewline\fi}%
8009 }
```

longborder The longborder style is like the above, but with horizontal and vertical lines:

```
8010 \newglossarystyle{longborder}{%
```

Base it on the glostylelong style:

```
8011 \setglossarystyle{long}%
```

Use longtable with two columns with vertical lines between each column:

```
8012 \renewenvironment{theglossary}{%
8013   \begin{longtable}{|l|p{\glsgdescwidth}|}{\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
8014 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8015 }
```

longheader The longheader style is like the long style but with a header:

```
8016 \newglossarystyle{longheader}{%
```

Base it on the glostylelong style:

```
8017 \setglossarystyle{long}%
```

Set the table's header:

```
8018 \renewcommand*{\glossaryheader}{%
8019   \bfseries \entryname & \bfseries \descriptionname\tabularnewline\endhead}%
8020 }
```

longheaderborder The longheaderborder style is like the long style but with a header and border:

```
8021 \newglossarystyle{longheaderborder}{%
```

Base it on the glostylelongborder style:

```
8022 \setglossarystyle{longborder}%
```

Set the table's header and add horizontal line to table's foot:

```
8023 \renewcommand*{\glossaryheader}{%
8024 \hline\bfseries \entryname & \bfseries
8025 \descriptionname\tabularnewline\hline
8026 \endhead
8027 \hline\endfoot}%
8028 }
```

long3col The long3col style is like long but with 3 columns

```
8029 \newglossarystyle{long3col}{%
```

Use a longtable with 3 columns:

```
8030 \renewenvironment{theglossary}%
8031 {\begin{longtable}{lp{\glstdescwidth}p{\glspagelistwidth}}}%
8032 {\end{longtable}}%
```

No table header:

```
8033 \renewcommand*{\glossaryheader}{}%
```

No headings between groups:

```
8034 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
8035 \renewcommand{\glossentry}[2]{%
8036 \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8037 \glossentrydesc{##1} & ##2\tabularnewline
8038 }%
```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```
8039 \renewcommand{\subglossentry}[3]{%
8040 &
8041 \glssubentryitem{##2}%
8042 \glstarget{##2}{\strut}\glossentrydesc{##2} &
8043 ##3\tabularnewline
8044 }%
```

Blank row between groups:

```
8045 \renewcommand*{\glsgroupskip}{%
8046 \ifglsgnোগroupskip\else & &\tabularnewline\fi}%
8047 }
```

long3colborder The long3colborder style is like the long3col style but with a border:

```
8048 \newglossarystyle{long3colborder}{%
```

Base it on the glostylelong3col style:

```
8049 \setglossarystyle{long3col}%
```

Use a longtable with 3 columns with vertical lines around them:

```
8050 \renewenvironment{theglossary}%
8051 {\begin{longtable}{|l|p{\glstdescwidth}|p{\glspagelistwidth}|}%
8052 {\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
8053 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8054 }
```

long3colheader The long3colheader style is like long3col but with a header row:

```
8055 \newglossarystyle{long3colheader}{%
    Base it on the glostylelong3col style:
8056 \setglossarystyle{long3col}%
    Set the table's header:
8057 \renewcommand*{\glossaryheader}{%
8058 \bfseries\entryname&\bfseries\descriptionname&
8059 \bfseries\pagelistname\tabularnewline\endhead}%
8060 }
```

long3colheaderborder The long3colheaderborder style is like the above but with a border

```
8061 \newglossarystyle{long3colheaderborder}{%
    Base it on the glostylelong3colborder style:
8062 \setglossarystyle{long3colborder}%
    Set the table's header and add horizontal line at table's foot:
8063 \renewcommand*{\glossaryheader}{%
8064 \hline
8065 \bfseries\entryname&\bfseries\descriptionname&
8066 \bfseries\pagelistname\tabularnewline\hline\endhead
8067 \hline\endfoot}%
8068 }
```

long4col The long4col style has four columns where the third column contains the value of the associated symbol key.

```
8069 \newglossarystyle{long4col}{%
    Use a longtable with 4 columns:
8070 \renewenvironment{theglossary}%
8071 {\begin{longtable}{llll}}%
8072 {\end{longtable}}%
    No table header:
8073 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8074 \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):

```
8075 \renewcommand{\glossentry}[2]{%
8076 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8077 \glossentrydesc{##1} &
8078 \glossentrysymbol{##1} &
8079 ##2\tabularnewline
8080 }
```


Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
8081 \renewcommand{\subglossentry}[3]{%
8082     &
8083     \glssubentryitem{##2}%
8084     \glstarget{##2}{\strut}\glossentrydesc{##2} &
8085     \glossentrysymbol{##2} & ##3\tabularnewline
8086 }
```

Blank row between groups:

```
8087 \renewcommand*{\glsgroupskip}{%
8088     \ifglsgnোগroupskip\else & & \tabularnewline\fi}%
8089 }
```

long4colheader The long4colheader style is like long4col but with a header row.

```
8090 \newglossarystyle{long4colheader}{%
```

Base it on the glostylelong4col style:

```
8091 \setglossarystyle{long4col}%
```

Table has a header:

```
8092 \renewcommand*{\glossaryheader}{%
8093     \bfseries\entryname&\bfseries\descriptionname&
8094     \bfseries \symbolname&
8095     \bfseries\pagelistname\tabularnewline\endhead}%
8096 }
```

long4colborder The long4colborder style is like long4col but with a border.

```
8097 \newglossarystyle{long4colborder}{%
```

Base it on the glostylelong4col style:

```
8098 \setglossarystyle{long4col}%
```

Use a longtable with 4 columns surrounded by vertical lines:

```
8099 \renewenvironment{theglossary}%
8100     {\begin{longtable}{|l|l|l|l|}}%
8101     {\end{longtable}}%
```

Add horizontal lines to the head and foot of the table:

```
8102 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8103 }
```

long4colheaderborder The long4colheaderborder style is like the above but with a border.

```
8104 \newglossarystyle{long4colheaderborder}{%
```

Base it on the glostylelong4col style:

```
8105 \setglossarystyle{long4col}%
```

Use a longtable with 4 columns surrounded by vertical lines:

```
8106 \renewenvironment{theglossary}%
8107     {\begin{longtable}{|l|l|l|l|}}%
8108     {\end{longtable}}%
```

Add table header and horizontal line at the table's foot:

```
8109 \renewcommand*{\glossaryheader}{%
8110 \hline\bfseries\entryname&\bfseries\descriptionname&
8111 \bfseries \symbolname&
8112 \bfseries\pagelistname\tabularnewline\hline\endhead
8113 \hline\endfoot}%
8114 }
```

altlong4col The altlong4col style is like the long4col style but can have multiline descriptions and page lists.

```
8115 \newglossarystyle{altlong4col}{%
Base it on the glostylelong4col style:
8116 \setglossarystyle{long4col}%
Use a longtable with 4 columns where the second and last columns may have
multiple lines in each row:
8117 \renewenvironment{theglossary}%
8118 {\begin{longtable}{lp{\glsdescwidth}lp{\glspagelistwidth}}}%
8119 {\end{longtable}}%
8120 }
```

altlong4colheader The altlong4colheader style is like altlong4col but with a header row.

```
8121 \newglossarystyle{altlong4colheader}{%
Base it on the glostylelong4colheader style:
8122 \setglossarystyle{long4colheader}%
Use a longtable with 4 columns where the second and last columns may have
multiple lines in each row:
8123 \renewenvironment{theglossary}%
8124 {\begin{longtable}{lp{\glsdescwidth}lp{\glspagelistwidth}}}%
8125 {\end{longtable}}%
8126 }
```

altlong4colborder The altlong4colborder style is like altlong4col but with a border.

```
8127 \newglossarystyle{altlong4colborder}{%
Base it on the glostylelong4colborder style:
8128 \setglossarystyle{long4colborder}%
Use a longtable with 4 columns where the second and last columns may have
multiple lines in each row:
8129 \renewenvironment{theglossary}%
8130 {\begin{longtable}{|lp{\glsdescwidth}|lp{\glspagelistwidth}|}%
8131 {\end{longtable}}%
8132 }
```

altlong4colheaderborder The altlong4colheaderborder style is like the above but with a header as well as a border.

```
8133 \newglossarystyle{altlong4colheaderborder}{%
```

Base it on the `glostylelong4colheaderborder` style:

```
8134 \setglossarystyle{long4colheaderborder}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8135 \renewenvironment{theglossary}%  
8136   {\begin{longtable}{|l|p{\glsdescwidth}|l|p{\glspagelistwidth}|}%  
8137   {\end{longtable}}%  
8138 }
```

5.5 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.

```
8139 \ProvidesPackage{glossary-longragged}[2015/07/08 v4.16 (NLCT)]
```

Requires the package:

```
8140 \RequirePackage{array}
```

Requires the package:

```
8141 \RequirePackage{longtable}
```

`\glsdescwidth` This is a length that governs the width of the description column. This may have already been defined.

```
8142 \@ifundefined{glsdescwidth}{%  
8143   \newlength\glsdescwidth  
8144   \setlength{\glsdescwidth}{0.6\hsize}  
8145 }{}
```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```
8146 \@ifundefined{glspagelistwidth}{%  
8147   \newlength\glspagelistwidth  
8148   \setlength{\glspagelistwidth}{0.1\hsize}  
8149 }{}
```

`longragged` The `longragged` glossary style is like the `long` but uses ragged right formatting for the description column.

```
8150 \newglossarystyle{longragged}{%
```

Use `longtable` with two columns:

```
8151 \renewenvironment{theglossary}%  
8152   {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}}}%  
8153   {\end{longtable}}%
```

Do nothing at the start of the environment:

```
8154 \renewcommand*{\glossaryheader}{}%
```

No heading between groups:

```
8155 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries displayed in a row:

```
8156 \renewcommand{\glossentry}[2]{%
8157   \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8158   \glossentrydesc{##1}\glspostdescription\space ##2%
8159   \tabularnewline
8160 }%
```

Sub entries displayed on the following row without the name:

```
8161 \renewcommand{\subglossentry}[3]{%
8162   &
8163   \glssubentryitem{##2}%
8164   \glstarget{##2}{\strut}\glossentrydesc{##2}%
8165   \glspostdescription\space ##3%
8166   \tabularnewline
8167 }%
```

Blank row between groups:

```
8168 \renewcommand*{\glsgroupskip}{\ifglsgnোগroupskip\else & \tabularnewline\fi}%
8169 }
```

longraggedborder The longraggedborder style is like the above, but with horizontal and vertical lines:

```
8170 \newglossarystyle{longraggedborder}{%
```

Base it on the glostylelongragged style:

```
8171 \setglossarystyle{longragged}{%
```

Use longtable with two columns with vertical lines between each column:

```
8172 \renewenvironment{theglossary}{%
8173   \begin{longtable}{|l|>{\raggedright}p{\glsgdescwidth}|}%
8174   {\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
8175 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8176 }
```

longraggedheader The longraggedheader style is like the longragged style but with a header:

```
8177 \newglossarystyle{longraggedheader}{%
```

Base it on the glostylelongragged style:

```
8178 \setglossarystyle{longragged}{%
```

Set the table's header:

```
8179 \renewcommand*{\glossaryheader}{%
8180   \bfseries \entryname & \bfseries \descriptionname
8181   \tabularnewline\endhead}%
8182 }
```

raggedheaderborder The longraggedheaderborder style is like the longragged style but with a header and border:

```
8183 \newglossarystyle{longraggedheaderborder}{%
```

Base it on the glostylelongraggedborder style:

```
8184 \setglossarystyle{longraggedborder}{%
```

Set the table's header and add horizontal line to table's foot:

```
8185 \renewcommand*{\glossaryheader}{%
8186 \hline\bfseries \entryname & \bfseries \descriptionname
8187 \tabularnewline\hline
8188 \endhead
8189 \hline\endfoot}%
8190 }
```

longragged3col The longragged3col style is like longragged but with 3 columns

```
8191 \newglossarystyle{longragged3col}{%
```

Use a longtable with 3 columns:

```
8192 \renewenvironment{theglossary}%
8193 {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}%
8194 >{\raggedright}p{\glspagelistwidth}}}%
8195 {\end{longtable}}%
```

No table header:

```
8196 \renewcommand*{\glossaryheader}{}%
```

No headings between groups:

```
8197 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
8198 \renewcommand{\glossentry}[2]{%
8199 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8200 \glossentrydesc{##1} & ##2\tabularnewline
8201 }%
```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```
8202 \renewcommand{\subglossentry}[3]{%
8203 &
8204 \glssubentryitem{##2}%
8205 \glstarget{##2}{\strut}\glossentrydesc{##2} &
8206 ##3\tabularnewline
8207 }%
```

Blank row between groups:

```
8208 \renewcommand*{\glsgroupskip}{%
8209 \ifglsnogroupskip\else & &\tabularnewline\fi}%
8210 }
```

`longragged3colborder` The `longragged3colborder` style is like the `longragged3col` style but with a border:

```
8211 \newglossarystyle{longragged3colborder}{%
```

Base it on the `glostylelongragged3col` style:

```
8212 \setglossarystyle{longragged3col}{%
```

Use a `longtable` with 3 columns with vertical lines around them:

```
8213 \renewenvironment{theglossary}{%
```

```
8214     {\begin{longtable}{|l|>{\raggedright}p{\glsdescwidth}}|%
```

```
8215         >{\raggedright}p{\glspagelistwidth}}|}%
```

```
8216     {\end{longtable}}}%
```

Place horizontal lines at the head and foot of the table:

```
8217 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
```

```
8218 }
```

`longragged3colheader` The `longragged3colheader` style is like `longragged3col` but with a header row:

```
8219 \newglossarystyle{longragged3colheader}{%
```

Base it on the `glostylelongragged3col` style:

```
8220 \setglossarystyle{longragged3col}{%
```

Set the table's header:

```
8221 \renewcommand*{\glossaryheader}{%
```

```
8222     \bfseries\entryname&\bfseries\descriptionname&
```

```
8223     \bfseries\pagelistname\tabularnewline\endhead}%
```

```
8224 }
```

`longragged3colheaderborder` The `longragged3colheaderborder` style is like the above but with a border

```
8225 \newglossarystyle{longragged3colheaderborder}{%
```

Base it on the `glostylelongragged3colborder` style:

```
8226 \setglossarystyle{longragged3colborder}{%
```

Set the table's header and add horizontal line at table's foot:

```
8227 \renewcommand*{\glossaryheader}{%
```

```
8228     \hline
```

```
8229     \bfseries\entryname&\bfseries\descriptionname&
```

```
8230     \bfseries\pagelistname\tabularnewline\hline\endhead
```

```
8231     \hline\endfoot}%
```

```
8232 }
```

`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.

```
8233 \newglossarystyle{altlongragged4col}{%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8234 \renewenvironment{theglossary}{%
```

```

8235     {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}l%
8236         >{\raggedright}p{\glspagelistwidth}}}%
8237     {\end{longtable}}}%

```

No table header:

```

8238     \renewcommand*{\glossaryheader}{}%

```

No group headings:

```

8239     \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):

```

8240     \renewcommand{\glossentry}[2]{%
8241         \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8242         \glossentrydesc{##1} & \glossentrysymbol{##1} &
8243         ##2\tabularnewline
8244     }%

```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```

8245     \renewcommand{\subglossentry}[3]{%
8246         &
8247         \glssubentryitem{##2}%
8248         \glstarget{##2}{\strut}\glossentrydesc{##2} &
8249         \glossentrysymbol{##2} & ##3\tabularnewline
8250     }%

```

Blank row between groups:

```

8251     \renewcommand*{\glsgroupskip}{%
8252         \ifglsgroupskip\else & & \tabularnewline\fi}%
8253 }

```

`ongragged4colheader` The `altlongragged4colheader` style is like `altlongragged4col` but with a header row.

```

8254 \newglossarystyle{altlongragged4colheader}{%

```

Base it on the `glostylealtlongragged4col` style:

```

8255     \setglossarystyle{altlongragged4col}%

```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```

8256     \renewenvironment{theglossary}%
8257     {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}l%
8258         >{\raggedright}p{\glspagelistwidth}}}%
8259     {\end{longtable}}%

```

Table has a header:

```

8260     \renewcommand*{\glossaryheader}{%
8261         \bfseries\entryname&\bfseries\descriptionname&
8262         \bfseries \symbolname&
8263         \bfseries\pagelistname\tabularnewline\endhead}%
8264 }

```

`longragged4colborder` The `altlongragged4colborder` style is like `altlongragged4col` but with a border.

```

8265 \newglossarystyle{altlongragged4colborder}{%
      Base it on the glostylealtlongragged4col style:
8266   \setglossarystyle{altlongragged4col}%

      Use a longtable with 4 columns where the second and last columns may have
      multiple lines in each row:
8267   \renewenvironment{theglossary}%
8268     {\begin{longtable}{|l|>{\raggedright}p{\glsdescwidth}|l|}%
8269      >{\raggedright}p{\glspagelistwidth}|}%
8270     {\end{longtable}}%

      Add horizontal lines to the head and foot of the table:
8271   \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8272 }
```

`longragged4colheaderborder` The `altlongragged4colheaderborder` style is like the above but with a header as well as a border.

```

8273 \newglossarystyle{altlongragged4colheaderborder}{%
      Base it on the glostylealtlongragged4col style:
8274   \setglossarystyle{altlongragged4col}%

      Use a longtable with 4 columns where the second and last columns may have
      multiple lines in each row:
8275   \renewenvironment{theglossary}%
8276     {\begin{longtable}{|l|>{\raggedright}p{\glsdescwidth}|l|}%
8277      >{\raggedright}p{\glspagelistwidth}|}%
8278     {\end{longtable}}%

      Add table header and horizontal line at the table's foot:
8279   \renewcommand*{\glossaryheader}{%
8280     \hline\bfseries\entryname&\bfseries\descriptionname&
8281     \bfseries \symbolname&
8282     \bfseries\pagelistname\tabularnewline\hline\endhead
8283     \hline\endfoot}%
8284 }
```

5.6 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.

```
8285 \ProvidesPackage{glossary-mcols}[2015/07/08 v4.16 (NLCT)]
```

Required packages:

```

8286 \RequirePackage{multicol}
8287 \RequirePackage{glossary-tree}
```


`\indexspace` The are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```
8288 \providecommand{\indexspace}{%
8289   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
8290 }
```

`\glsmcols` Define macro in which to store the number of columns. (Defaults to 2.)

```
8291 \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.)

```
8292 \newglossarystyle{mcolindex}{%
8293   \setglossarystyle{index}%
8294   \renewenvironment{theglossary}%
8295     {%
8296       \begin{multicols}{\glsmcols}
8297       \setlength{\parindent}{0pt}%
8298       \setlength{\parskip}{0pt plus 0.3pt}%
8299       \let\item\@idxitem}%
8300     {\end{multicols}}%
8301 }
```

`mcolindexgroup` As `mcolindex` but has headings:

```
8302 \newglossarystyle{mcolindexgroup}{%
8303   \setglossarystyle{mcolindex}%
8304   \renewcommand*\glsgroupheading[1]{%
8305     \item\textbf{\glsgroupheading{##1}}\indexspace}%
8306 }
```

`mcolindexhypergroup` The `mcolindexhypergroup` style is like the `mcolindexgroup` style but has hyper navigation.

```
8307 \newglossarystyle{mcolindexhypergroup}{%
```

Base it on the `glostylemcolindex` style:

```
8308   \setglossarystyle{mcolindex}%
```

Put navigation links to the groups at the start of the glossary:

```
8309   \renewcommand*\glossaryheader{%
8310     \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.

```
8311   \renewcommand*\glsgroupheading[1]{%
8312     \item\textbf{\glsnavhypertarget{##1}}{\glsgroupheading{##1}}}%
8313     \indexspace}%
8314 }
```

mcoltree Multi-column index style. Same as the tree, but puts the glossary in multiple columns.

```

8315 \newglossarystyle{mcoltree}{%
8316   \setglossarystyle{tree}%
8317   \renewenvironment{theglossary}%
8318   {%
8319     \begin{multicols}{\glsmcols}
8320     \setlength{\parindent}{0pt}%
8321     \setlength{\parskip}{0pt plus 0.3pt}%
8322   }%
8323   {\end{multicols}}}%
8324 }
```

mcoltreegroup Like the mcoltree style but the glossary groups have headings.

```

8325 \newglossarystyle{mcoltreegroup}{%
      Base it on the glostylemcoltree style:
8326   \setglossarystyle{mcoltree}%
      Each group has a heading (in bold) followed by a vertical gap):
8327   \renewcommand{\glsgroupheading}[1]{\par
8328     \noindent\textbf{\glsgrouptitle{##1}}\par\indexspace}%
8329 }
```

mcoltreehypergroup The mcoltreehypergroup style is like the treegroup style, but has a set of links to the groups at the start of the glossary.

```

8330 \newglossarystyle{mcoltreehypergroup}{%
      Base it on the glostylemcoltree style:
8331   \setglossarystyle{mcoltree}%
      Put navigation links to the groups at the start of the theglossary environment:
8332   \renewcommand*\{\glossaryheader}{%
8333     \par\noindent\textbf{\glsnavigation}\par\indexspace}%
      Each group has a heading (in bold with a target) followed by a vertical gap):
8334   \renewcommand*\{\glsgroupheading}[1]{%
8335     \par\noindent
8336     \textbf{\glsnavigationhypertarget{##1}{\glsgrouptitle{##1}}}\par
8337     \indexspace}%
8338 }
```

mcoltreenoname Multi-column index style. Same as the treenoname, but puts the glossary in multiple columns.

```

8339 \newglossarystyle{mcoltreenoname}{%
8340   \setglossarystyle{treenoname}%
8341   \renewenvironment{theglossary}%
8342   {%
```

```

8343     \begin{multicols}{\glsmcols}
8344     \setlength{\parindent}{0pt}%
8345     \setlength{\parskip}{0pt plus 0.3pt}%
8346   }%
8347   {\end{multicols}}}%
8348 }

```

mcoltreenonamegroup Like the mcoltreenoname style but the glossary groups have headings.

```

8349 \newglossarystyle{mcoltreenonamegroup}{%
    Base it on the glostylemcoltreenoname style:
8350   \setglossarystyle{mcoltreenoname}%
    Give each group a heading:
8351   \renewcommand{\glsgroupheading}[1]{\par
8352     \noindent\textbf{\glsgrouptitle{##1}}\par\indexspace}%
8353 }

```

treenonamehypergroup The mcoltreenonamehypergroup style is like the mcoltreenonamegroup style, but has a set of links to the groups at the start of the glossary.

```

8354 \newglossarystyle{mcoltreenonamehypergroup}{%
    Base it on the glostylemcoltreenoname style:
8355   \setglossarystyle{mcoltreenoname}%
    Put navigation links to the groups at the start of the theglossary environment:
8356   \renewcommand*{\glossaryheader}{%
8357     \par\noindent\textbf{\glsnavigation}\par\indexspace}%
    Each group has a heading (in bold with a target) followed by a vertical gap):
8358   \renewcommand*{\glsgroupheading}[1]{%
8359     \par\noindent
8360     \textbf{\glsnavigationhypertarget{##1}{\glsgrouptitle{##1}}}\par
8361     \indexspace}%
8362 }

```

mcolalmtree Multi-column index style. Same as the almtree, but puts the glossary in multiple columns.

```

8363 \newglossarystyle{mcolalmtree}{%
8364   \setglossarystyle{almtree}%
8365   \renewenvironment{theglossary}%
8366   {%
8367     \begin{multicols}{\glsmcols}
8368     \def\@gls@prevlevel{-1}%
8369     \mbox{}\par
8370   }%
8371   {\par\end{multicols}}%
8372 }

```

`mcolalattoreegroup` Like the `mcolalattoree` style but the glossary groups have headings.

```
8373 \newglossarystyle{mcolalattoreegroup}{%  
    Base it on the glostylemcolalattoree style:  
8374 \setglossarystyle{mcolalattoree}%  
    Give each group a heading.  
8375 \renewcommand{\glsgroupheading}[1]{\par  
8376 \def\@gls@prevlevel{-1}%  
8377 \hangindent0pt\relax  
8378 \parindent0pt\relax  
8379 \textbf{\glsgrouptitle{##1}}\par\indexspace}%  
8380 }
```

`colalattoreehypergroup` The `mcolalattoreehypergroup` style is like the `mcolalattoreegroup` style, but has a set of links to the groups at the start of the glossary.

```
8381 \newglossarystyle{mcolalattoreehypergroup}{%  
    Base it on the glostylemcolalattoree style:  
8382 \setglossarystyle{mcolalattoree}%  
    Put the navigation links in the header  
8383 \renewcommand*{\glossaryheader}{%  
8384 \par  
8385 \def\@gls@prevlevel{-1}%  
8386 \hangindent0pt\relax  
8387 \parindent0pt\relax  
8388 \textbf{\glsnavigation}\par\indexspace}%  
    Put a hypertarget at the start of each group  
8389 \renewcommand*{\glsgroupheading}[1]{%  
8390 \par  
8391 \def\@gls@prevlevel{-1}%  
8392 \hangindent0pt\relax  
8393 \parindent0pt\relax  
8394 \textbf{\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}\par  
8395 \indexspace}}
```

5.7 Glossary Styles using supertabular environment (`glossary-super` package)

The glossary styles defined in the package use the `supertabular` environment.

```
8396 \ProvidesPackage{glossary-super}[2015/07/08 v4.16 (NLCT)]
```

Requires the package:

```
8397 \RequirePackage{supertabular}
```

`\glsdescwidth` This is a length that governs the width of the description column. This may already have been defined if `has` has been loaded.

```
8398 \ifundefined{glsdescwidth}{%
```

```

8399 \newlength\glsdescwidth
8400 \setlength{\glsdescwidth}{0.6\hsize}
8401 }{}

```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined if has been loaded.

```

8402 \@ifundefined{glspagelistwidth}{%
8403 \newlength\glspagelistwidth
8404 \setlength{\glspagelistwidth}{0.1\hsize}
8405 }{}

```

`super` The super glossary style uses the supertabular environment (it uses lengths defined in the package.)

```
8406 \newglossarystyle{super}{%
```

Put the glossary in a supertabular environment with two columns and no head or tail:

```

8407 \renewenvironment{theglossary}%
8408 {\tablehead{}\tabletail{}}%
8409 \begin{supertabular}{lp{\glsdescwidth}}}%
8410 {\end{supertabular}}%

```

Do nothing at the start of the table:

```
8411 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8412 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```

8413 \renewcommand{\glossentry}[2]{%
8414 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8415 \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline
8416 }%

```

Sub entries put in a row (no name, description and page list in second column):

```

8417 \renewcommand{\subglossentry}[3]{%
8418 &
8419 \glssubentryitem{##2}%
8420 \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
8421 ##3\tabularnewline
8422 }%

```

Blank row between groups:

```

8423 \renewcommand*{\glsgroupskip}{%
8424 \ifglsnogroupskip\else & \tabularnewline\fi}%
8425 }

```

`superborder` The superborder style is like the above, but with horizontal and vertical lines:

```
8426 \newglossarystyle{superborder}{%
```

Base it on the `glostylesuper` style:

```
8427 \setglossarystyle{super}%
```

Put the glossary in a `supertabular` environment with two columns and a horizontal line in the head and tail:

```
8428 \renewenvironment{theglossary}%  
8429 {\tablehead{\hline}\tabletail{\hline}%  
8430 \begin{supertabular}{|l|p{\glsgdescwidth}|}%  
8431 {\end{supertabular}}%  
8432 }
```

`superheader` The `superheader` style is like the `super` style, but with a header:

```
8433 \newglossarystyle{superheader}{%
```

Base it on the `glostylesuper` style:

```
8434 \setglossarystyle{super}%
```

Put the glossary in a `supertabular` environment with two columns, a header and no tail:

```
8435 \renewenvironment{theglossary}%  
8436 {\tablehead{\bfseries \entryname &  
8437 \bfseries \descriptionname \tabularnewline}%  
8438 \tabletail{}}%  
8439 \begin{supertabular}{lp{\glsgdescwidth}}%  
8440 {\end{supertabular}}%  
8441 }
```

`superheaderborder` The `superheaderborder` style is like the `super` style but with a header and border:

```
8442 \newglossarystyle{superheaderborder}{%
```

Base it on the `glostylesuper` style:

```
8443 \setglossarystyle{super}%
```

Put the glossary in a `supertabular` environment with two columns, a header and horizontal lines above and below the table:

```
8444 \renewenvironment{theglossary}%  
8445 {\tablehead{\hline\bfseries \entryname &  
8446 \bfseries \descriptionname \tabularnewline\hline}%  
8447 \tabletail{\hline}  
8448 \begin{supertabular}{|l|p{\glsgdescwidth}|}%  
8449 {\end{supertabular}}%  
8450 }
```

`super3col` The `super3col` style is like the `super` style, but with 3 columns:

```
8451 \newglossarystyle{super3col}{%
```

Put the glossary in a `supertabular` environment with three columns and no head or tail:

```
8452 \renewenvironment{theglossary}%  
8453 {\tablehead{} \tabletail{}}%  
8454 \begin{supertabular}{lp{\glsgdescwidth}p{\glspagelistwidth}}%  
8455 {\end{supertabular}}%
```

Do nothing at the start of the table:

```
8456 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8457 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
8458 \renewcommand{\glossentry}[2]{%
8459   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8460   \glossentrydesc{##1} & ##2\tabularnewline
8461 }%
```

Sub entries on a row (no name, description in second column, page list in last column):

```
8462 \renewcommand{\subglossentry}[3]{%
8463   &
8464   \glssubentryitem{##2}%
8465   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8466   ##3\tabularnewline
8467 }%
```

Blank row between groups:

```
8468 \renewcommand*{\glsgroupskip}{}%
8469 \ifglsnogroupskip\else & &\tabularnewline\fi}%
8470 }
```

super3colborder The `super3colborder` style is like the `super3col` style, but with a border:

```
8471 \newglossarystyle{super3colborder}{}%
```

Base it on the `glostylesuper3col` style:

```
8472 \setglossarystyle{super3col}%
```

Put the glossary in a `supertabular` environment with three columns and a horizontal line in the head and tail:

```
8473 \renewenvironment{theglossary}%
8474   {\tablehead{\hline}\tabletail{\hline}}%
8475   \begin{supertabular}{|l|p{\glsdescwidth}|p{\glspagelistwidth}|}%
8476   {\end{supertabular}}%
8477 }
```

super3colheader The `super3colheader` style is like the `super3col` style but with a header row:

```
8478 \newglossarystyle{super3colheader}{}%
```

Base it on the `glostylesuper3col` style:

```
8479 \setglossarystyle{super3col}%
```

Put the glossary in a `supertabular` environment with three columns, a header and no tail:

```
8480 \renewenvironment{theglossary}%
8481   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
```

```

8482      \bfseries\pagelistname\tabularnewline\tabletail{}}%
8483      \begin{supertabular}{lp{\glstdescwidth}p{\glspagelistwidth}}}%
8484      {\end{supertabular}}}%
8485 }

```

super3colheaderborder The `super3colheaderborder` style is like the `super3col` style but with a header and border:

```

8486 \newglossarystyle{super3colheaderborder}{%
      Base it on the glostylesuper3colborder style:
8487   \setglossarystyle{super3colborder}%
      Put the glossary in a supertabular environment with three columns, a header
      with horizontal lines and a horizontal line in the tail:
8488   \renewenvironment{theglossary}%
8489     {\tablehead{\hline
8490       \bfseries\entryname&\bfseries\descriptionname&
8491       \bfseries\pagelistname\tabularnewline\hline}%
8492     \tabletail{\hline}%
8493     \begin{supertabular}{|l|p{\glstdescwidth}|p{\glspagelistwidth}|}%
8494     {\end{supertabular}}}%
8495 }

```

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.

```

8496 \newglossarystyle{super4col}{%
      Put the glossary in a supertabular environment with four columns and no head
      or tail:
8497   \renewenvironment{theglossary}%
8498     {\tablehead{}\tabletail{}}%
8499     \begin{supertabular}{|l|l|l|l|}%
8500     \end{supertabular}}%

```

Do nothing at the start of the table:

```

8501 \renewcommand*{\glossaryheader}{}%

```

No group headings:

```

8502 \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:

```

8503 \renewcommand{\glossentry}[2]{%
8504   \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8505   \glossentrydesc{##1} &
8506   \glossentrysymbol{##1} & ##3\tabularnewline
8507 }%

```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```

8508 \renewcommand{\subglossentry}[3]{%

```



```

8509      &
8510      \glssubentryitem{##2}%
8511      \glstarget{##2}{\strut}\glossentrydesc{##2} &
8512      \glossentrysymbol{##2} & ##3\tabularnewline
8513    }%

```

Blank row between groups:

```

8514  \renewcommand*{\glsgroupskip}{%
8515    \ifglsgnোগroupskip\else & & \tabularnewline\fi}%
8516 }

```

super4colheader The super4colheader style is like the super4col but with a header row.

```

8517 \newglossarystyle{super4colheader}{%
      Base it on the glostylesuper4col style:
8518  \setglossarystyle{super4col}%
      Put the glossary in a supertabular environment with four columns, a header and
      no tail:
8519  \renewenvironment{theglossary}%
8520    {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8521      \bfseries\symbolname &
8522      \bfseries\pagelistname\tabularnewline}%
8523    \tabletail{}}%
8524    \begin{supertabular}{|l|l|l|l|}%
8525    {\end{supertabular}}%
8526 }

```

super4colborder The super4colborder style is like the super4col but with a border.

```

8527 \newglossarystyle{super4colborder}{%
      Base it on the glostylesuper4col style:
8528  \setglossarystyle{super4col}%
      Put the glossary in a supertabular environment with four columns and a hori-
      zontal line in the head and tail:
8529  \renewenvironment{theglossary}%
8530    {\tablehead{\hline}\tabletail{\hline}%
8531    \begin{supertabular}{|l|l|l|l|}%
8532    {\end{supertabular}}%
8533 }

```

super4colheaderborder The super4colheaderborder style is like the super4col but with a header and border.

```

8534 \newglossarystyle{super4colheaderborder}{%
      Base it on the glostylesuper4col style:
8535  \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:

```

8536 \renewenvironment{theglossary}%
8537   {\tablehead{\hline\bfseries\entryname&\bfseries\descriptionname&
8538     \bfseries\symbolname &
8539     \bfseries\pagelistname\tabularnewline\hline}%
8540   \tabletail{\hline}%
8541   \begin{supertabular}{|l|l|l|l|}%
8542   {\end{supertabular}}}%
8543 }
```

altsuper4col The altsuper4col glossary style is like super4col but has provision for multiline descriptions.

```

8544 \newglossarystyle{altsuper4col}{%
      Base it on the glostylesuper4col style:
8545   \setglossarystyle{super4col}%
      Put the glossary in a supertabular environment with four columns and no head
      or tail:
8546   \renewenvironment{theglossary}%
8547   {\tablehead{}\tabletail{}}%
8548   \begin{supertabular}{lp{\glsgdescwidth}lp{\glspagelistwidth}}}%
8549   {\end{supertabular}}}%
8550 }
```

altsuper4colheader The altsuper4colheader style is like the altsuper4col but with a header row.

```

8551 \newglossarystyle{altsuper4colheader}{%
      Base it on the glostylesuper4colheader style:
8552   \setglossarystyle{super4colheader}%
      Put the glossary in a supertabular environment with four columns, a header and
      no tail:
8553   \renewenvironment{theglossary}%
8554   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8555     \bfseries\symbolname &
8556     \bfseries\pagelistname\tabularnewline}\tabletail{}}%
8557   \begin{supertabular}{lp{\glsgdescwidth}lp{\glspagelistwidth}}}%
8558   {\end{supertabular}}}%
8559 }
```

altsuper4colborder The altsuper4colborder style is like the altsuper4col but with a border.

```

8560 \newglossarystyle{altsuper4colborder}{%
      Base it on the glostylesuper4colborder style:
8561   \setglossarystyle{super4colborder}%
      Put the glossary in a supertabular environment with four columns and a hori-
      zontal line in the head and tail:
```

```

8562 \renewenvironment{theglossary}%
8563   {\tablehead{\hline}\tabletail{\hline}%
8564     \begin{supertabular}%
8565       {\llp{\glstdescwidth}\llp{\glspagelistwidth}}}%
8566   {\end{supertabular}}%
8567 }

```

`per4colheaderborder` The `altsuper4colheaderborder` style is like the `altsuper4col` but with a header and border.

```

8568 \newglossarystyle{altsuper4colheaderborder}{%

```

Base it on the `glostylesuper4colheaderborder` style:

```

8569 \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:

```

8570 \renewenvironment{theglossary}%
8571   {\tablehead{\hline
8572     \bfseries\entryname &
8573     \bfseries\descriptionname &
8574     \bfseries\symbolname &
8575     \bfseries\pagelistname\tabularnewline\hline}%
8576   \tabletail{\hline}%
8577   \begin{supertabular}%
8578     {\llp{\glstdescwidth}\llp{\glspagelistwidth}}}%
8579   {\end{supertabular}}%
8580 }

```

5.8 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.

```

8581 \ProvidesPackage{glossary-superragged}[2015/07/08 v4.16 (NLCT)]

```

Requires the package:

```

8582 \RequirePackage{array}

```

Requires the package:

```

8583 \RequirePackage{supertabular}

```

`\glstdescwidth` This is a length that governs the width of the description column. This may already have been defined.

```

8584 \@ifundefined{glstdescwidth}{%
8585   \newlength\glstdescwidth
8586   \setlength{\glstdescwidth}{0.6\hsize}
8587 }{}

```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```
8588 \@ifundefined{glspagelistwidth}{%
8589   \newlength{glspagelistwidth}
8590   \setlength{glspagelistwidth}{0.1\hsize}
8591 }{}
```

`superragged` The `superragged` glossary style uses the `supertabular` environment.

```
8592 \newglossarystyle{superragged}{%
```

Put the glossary in a `supertabular` environment with two columns and no head or tail:

```
8593   \renewenvironment{theglossary}%
8594     {\tablehead{}\tabletail{}}%
8595     \begin{supertabular}{1>{\raggedright}p{\glstdescwidth}}}%
8596     {\end{supertabular}}%
```

Do nothing at the start of the table:

```
8597   \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8598   \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```
8599   \renewcommand{\glossentry}[2]{%
8600     \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8601     \glossentrydesc{##1}\glspostdescription\space ##2%
8602     \tabularnewline
8603   }%
```

Sub entries put in a row (no name, description and page list in second column):

```
8604   \renewcommand{\subglossentry}[3]{%
8605     &
8606     \glssubentryitem{##2}%
8607     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
8608     ##3%
8609     \tabularnewline
8610   }%
```

Blank row between groups:

```
8611   \renewcommand*{\glsgroupskip}{\ifglsgnোগroupskip\else & \tabularnewline\fi}%
8612 }
```

`superraggedborder` The `superraggedborder` style is like the above, but with horizontal and vertical lines:

```
8613 \newglossarystyle{superraggedborder}{%
```

Base it on the `glostylessuperragged` style:

```
8614   \setglossarystyle{superragged}%
```

Put the glossary in a supertabular environment with two columns and a horizontal line in the head and tail:

```
8615 \renewenvironment{theglossary}%
8616   {\tablehead{\hline}\tabletail{\hline}%
8617    \begin{supertabular}{|l|>{\raggedright}p{\glsgdescwidth}|}}%
8618   {\end{supertabular}}%
8619 }
```

superraggedheader The superraggedheader style is like the super style, but with a header:

```
8620 \newglossarystyle{superraggedheader}{%
```

Base it on the glostylessuperragged style:

```
8621 \setglossarystyle{superragged}%
```

Put the glossary in a supertabular environment with two columns, a header and no tail:

```
8622 \renewenvironment{theglossary}%
8623   {\tablehead{\bfseries \entryname & \bfseries \descriptionname
8624    \tabularnewline}%
8625    \tabletail{}}%
8626    \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}}}%
8627   {\end{supertabular}}%
8628 }
```

superraggedheaderborder The superraggedheaderborder style is like the superragged style but with a header and border:

```
8629 \newglossarystyle{superraggedheaderborder}{%
```

Base it on the glostylessuper style:

```
8630 \setglossarystyle{superragged}%
```

Put the glossary in a supertabular environment with two columns, a header and horizontal lines above and below the table:

```
8631 \renewenvironment{theglossary}%
8632   {\tablehead{\hline\bfseries \entryname &
8633    \bfseries \descriptionname\tabularnewline\hline}%
8634    \tabletail{\hline}
8635    \begin{supertabular}{|l|>{\raggedright}p{\glsgdescwidth}|}}%
8636   {\end{supertabular}}%
8637 }
```

superragged3col The superragged3col style is like the superragged style, but with 3 columns:

```
8638 \newglossarystyle{superragged3col}{%
```

Put the glossary in a supertabular environment with three columns and no head or tail:

```
8639 \renewenvironment{theglossary}%
8640   {\tablehead{} \tabletail{}}%
8641   \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}%
8642    >{\raggedright}p{\glspagelistwidth}}}%
8643   {\end{supertabular}}%
```

Do nothing at the start of the table:

```
8644 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8645 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
8646 \renewcommand{\glossentry}[2]{%
8647   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8648   \glossentrydesc{##1} &
8649   ##2\tabularnewline
8650 }%
```

Sub entries on a row (no name, description in second column, page list in last column):

```
8651 \renewcommand{\subglossentry}[3]{%
8652   &
8653   \glssubentryitem{##2}%
8654   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8655   ##3\tabularnewline
8656 }%
```

Blank row between groups:

```
8657 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else & &\tabularnewline\fi}%
8658 }
```

superragged3colborder The `superragged3colborder` style is like the `superragged3col` style, but with a border:

```
8659 \newglossarystyle{superragged3colborder}{}%
```

Base it on the `glostylesuperragged3col` style:

```
8660 \setglossarystyle{superragged3col}%
```

Put the glossary in a `supertabular` environment with three columns and a horizontal line in the head and tail:

```
8661 \renewenvironment{theglossary}%
8662   {\tablehead{\hline}\tabletail{\hline}}%
8663   \begin{supertabular}{|l|>{\raggedright}p{\glsdescwidth}|%
8664     >{\raggedright}p{\glspagerlistwidth}|}%
8665   {\end{supertabular}}%
8666 }
```

superragged3colheader The `superragged3colheader` style is like the `superragged3col` style but with a header row:

```
8667 \newglossarystyle{superragged3colheader}{}%
```

Base it on the `glostylesuperragged3col` style:

```
8668 \setglossarystyle{superragged3col}%
```

Put the glossary in a supertabular environment with three columns, a header and no tail:

```

8669 \renewenvironment{theglossary}%
8670   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8671     \bfseries\pagelistname\tabularnewline}\tabletail{}}%
8672   \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}%
8673     >{\raggedright}p{\glspagelistwidth}}}%
8674   {\end{supertabular}}%
8675 }

```

ght3colheaderborder The `superragged3colheaderborder` style is like the `superragged3col` style but with a header and border:

```

8676 \newglossarystyle{superragged3colheaderborder}{%

```

Base it on the `glostylesuperragged3colborder` style:

```

8677 \setglossarystyle{superragged3colborder}%

```

Put the glossary in a supertabular environment with three columns, a header with horizontal lines and a horizontal line in the tail:

```

8678 \renewenvironment{theglossary}%
8679   {\tablehead{\hline
8680     \bfseries\entryname&\bfseries\descriptionname&
8681     \bfseries\pagelistname\tabularnewline\hline}%
8682   \tabletail{\hline}%
8683   \begin{supertabular}{l|l>{\raggedright}p{\glsgdescwidth}|%
8684     >{\raggedright}p{\glspagelistwidth}}}%
8685   {\end{supertabular}}%
8686 }

```

altsuperragged4col The `altsuperragged4col` glossary style is like `altsuper4col` style in the package but uses ragged right formatting in the description and page list columns.

```

8687 \newglossarystyle{altsuperragged4col}{%

```

Put the glossary in a supertabular environment with four columns and no head or tail:

```

8688 \renewenvironment{theglossary}%
8689   {\tablehead{}\tabletail{}}%
8690   \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}l%
8691     >{\raggedright}p{\glspagelistwidth}}}%
8692   {\end{supertabular}}%

```

Do nothing at the start of the table:

```

8693 \renewcommand*{\glossaryheader}{}%

```

No group headings:

```

8694 \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:

```

8695 \renewcommand{\glossentry}[2]{}%

```

```

8696 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8697 \glossentrydesc{##1} &
8698 \glossentrysymbol{##1} & ##2\tabularnewline
8699 }%

```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```

8700 \renewcommand{\subglossentry}[3]{%
8701   &
8702   \glssubentryitem{##2}%
8703   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8704   \glossentrysymbol{##2} & ##3\tabularnewline
8705 }%

```

Blank row between groups:

```

8706 \renewcommand*{\glsgroupskip}{\ifglsgnogroupskip\else & & \tabularnewline\fi}%
8707 }

```

altperragged4colheader The `altperragged4colheader` style is like the `altperragged4col` style but with a header row.

```

8708 \newglossarystyle{altperragged4colheader}{%

```

Base it on the `glostylealtperragged4col` style:

```

8709 \setglossarystyle{altperragged4col}%

```

Put the glossary in a supertabular environment with four columns, a header and no tail:

```

8710 \renewenvironment{theglossary}%
8711   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8712     \bfseries\symbolname &
8713     \bfseries\pagelistname\tabularnewline}\tabletail{}}%
8714   \begin{supertabular}{1>{\raggedright}p{\glsdescwidth}1%
8715     >{\raggedright}p{\glspagelistwidth}}}%
8716   {\end{supertabular}}%
8717 }

```

altperragged4colborder The `altperragged4colborder` style is like the `altperragged4col` style but with a border.

```

8718 \newglossarystyle{altperragged4colborder}{%

```

Base it on the `glostylealtperragged4col` style:

```

8719 \setglossarystyle{altperragged4col}%

```

Put the glossary in a supertabular environment with four columns and a horizontal line in the head and tail:

```

8720 \renewenvironment{theglossary}%
8721   {\tablehead{\hline}\tabletail{\hline}%
8722   \begin{supertabular}%
8723     {|1|>{\raggedright}p{\glsdescwidth}|1|%
8724     >{\raggedright}p{\glspagelistwidth}|}%
8725   {\end{supertabular}}%
8726 }

```


`ged4colheaderborder` The `altsuperragged4colheaderborder` style is like the `altsuperragged4col` style but with a header and border.

```
8727 \newglossarystyle{altsuperragged4colheaderborder}{%
```

Base it on the `glostylealtsuperragged4col` style:

```
8728 \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:

```
8729 \renewenvironment{theglossary}{%
8730   {\tablehead{\hline
8731     \bfseries\entryname &
8732     \bfseries\descriptionname &
8733     \bfseries\symbolname &
8734     \bfseries\pagelistname\tabularnewline\hline}%
8735   \tabletail{\hline}%
8736   \begin{supertabular}%
8737     {\l|>{\raggedright}p{\glstdescwidth}|l|}%
8738     >{\raggedright}p{\glspagelistwidth}|}%
8739   {\end{supertabular}}}%
8740 }
```

5.9 Tree Styles (`glossary-tree.sty`)

The style file defines glossary styles that have a tree-like structure. These are designed for hierarchical glossaries.

```
8741 \ProvidesPackage{glossary-tree}[2015/07/08 v4.16 (NLCT)]
```

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```
8742 \providecommand{\indexspace}{%
8743   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
8744 }
```

`\glstreenamefmt` Format used to display the name in the tree styles. (This may be counteracted by `\glslnamefont`.) This command is also used to format the group headings.

```
8745 \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.

```
8746 \newglossarystyle{index}{%
```

Set the paragraph indentation and skip and define `\item` to be the same as that used by the index:

```
8747 \renewenvironment{theglossary}{%
8748   {\setlength{\parindent}{0pt}}%
```

```

8749 \setlength{\parskip}{0pt plus 0.3pt}%
8750 \let\item\@idxitem}%

8751 {\par}%

```

Do nothing at the start of the environment:

```
8752 \renewcommand*{\glossaryheader}{}%
```

No group headers:

```
8753 \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.

```

8754 \renewcommand*{\glossentry}[2]{%
8755 \item\glstentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
8756 \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
8757 \space \glossentrydesc{##1}\glspostdescription\space ##2%
8758 }%

```

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.

```

8759 \renewcommand{\subglossentry}[3]{%
8760 \ifcase##1\relax
8761 % level 0
8762 \item
8763 \or
8764 % level 1
8765 \subitem
8766 \glssubentryitem{##2}%
8767 \else
8768 % all other levels
8769 \subsubitem
8770 \fi
8771 \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
8772 \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
8773 \space\glossentrydesc{##2}\glspostdescription\space ##3%
8774 }%

```

Vertical gap between groups is the same as that used by indices:

```
8775 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}}
```

`indexgroup` The `indexgroup` style is like the `index` style but has headings.

```
8776 \newglossarystyle{indexgroup}{%
```

Base it on the `glostyleindex` style:

```
8777 \setglossarystyle{index}%
```

Add a heading for each group. This puts the group's title in bold followed by a vertical gap.

```

8778 \renewcommand*{\glsgroupheading}[1]{%
8779 \item\glstreenamefmt{\glsgrouptitle{##1}}\indexspace}%
8780 }

```

indexhypergroup The indexhypergroup style is like the indexgroup style but has hyper navigation.

```

8781 \newglossarystyle{indexhypergroup}{%

```

Base it on the glostyleindex style:

```

8782 \setglossarystyle{index}%

```

Put navigation links to the groups at the start of the glossary:

```

8783 \renewcommand*{\glossaryheader}{%
8784 \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.

```

8785 \renewcommand*{\glsgroupheading}[1]{%
8786 \item\glstreenamefmt{\glshypertarget{##1}}{\glsgrouptitle{##1}}}%
8787 \indexspace}%
8788 }

```

tree The tree glossary style is similar in style to the index style, but can have arbitrary levels.

```

8789 \newglossarystyle{tree}{%

```

Set the paragraph indentation and skip:

```

8790 \renewenvironment{theglossary}%
8791 {\setlength{\parindent}{0pt}%
8792 \setlength{\parskip}{0pt plus 0.3pt}}%
8793 {}%

```

Do nothing at the start of the theglossary environment:

```

8794 \renewcommand*{\glossaryheader}{}%

```

No group headings:

```

8795 \renewcommand*{\glsgroupheading}[1]{}%

```

Main (level 0) entries: name in bold, followed by symbol in brackets (if it exists), the description and the page list:

```

8796 \renewcommand{\glossentry}[2]{%
8797 \hangindent0pt\relax
8798 \parindent0pt\relax
8799 \glstryitem{##1}\glstreenamefmt{\glstarget{##1}}{\glossentryname{##1}}}%
8800 \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
8801 \space\glossentrydesc{##1}\glspostdescription\space##2\par
8802 }%

```

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.

```

8803 \renewcommand{\subglossentry}[3]{%

```

```

8804 \hangindent##1\glstreeindent\relax
8805 \parindent##1\glstreeindent\relax
8806 \ifnum##1=1\relax
8807 \glssubentryitem{##2}%
8808 \fi
8809 \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
8810 \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
8811 \space\glossentrydesc{##2}\glspostdescription\space ##3\par
8812 }%

```

Vertical gap between groups is the same as that used by indices:

```

8813 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}

```

treegroup Like the tree style but the glossary groups have headings.

```

8814 \newglossarystyle{treegroup}{%

```

Base it on the glostyletree style:

```

8815 \setglossarystyle{tree}%

```

Each group has a heading (in bold) followed by a vertical gap):

```

8816 \renewcommand{\glsgroupheading}[1]{\par
8817 \noindent\glstreenamefmt{\glsgrouptitle{##1}}\par\indexspace}%
8818 }

```

treehypergroup The treehypergroup style is like the treegroup style, but has a set of links to the groups at the start of the glossary.

```

8819 \newglossarystyle{treehypergroup}{%

```

Base it on the glostyletree style:

```

8820 \setglossarystyle{tree}%

```

Put navigation links to the groups at the start of the theglossary environment:

```

8821 \renewcommand*{\glossaryheader}{%
8822 \par\noindent\glstreenamefmt{\glsgroupnavigation}\par\indexspace}%

```

Each group has a heading (in bold with a target) followed by a vertical gap):

```

8823 \renewcommand*{\glsgroupheading}[1]{%
8824 \par\noindent
8825 \glstreenamefmt{\glsgrouphypertarget{##1}{\glsgrouptitle{##1}}}\par
8826 \indexspace}%
8827 }

```

\glstreeindent Length governing left indent for each level of the tree style.

```

8828 \newlength\glstreeindent
8829 \setlength{\glstreeindent}{10pt}

```

treenoname The treenoname glossary style is like the tree style, but doesn't print the name or symbol for sub-levels.

```

8830 \newglossarystyle{treenoname}{%

```

Set the paragraph indentation and skip:

```
8831 \renewenvironment{theglossary}%
8832   {\setlength{\parindent}{0pt}%
8833    \setlength{\parskip}{0pt plus 0.3pt}}%
8834   {}%
```

No header:

```
8835 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8836 \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.

```
8837 \renewcommand{\glossentry}[2]{%
8838   \hangindent0pt\relax
8839   \parindent0pt\relax
8840   \glstryitem{##1}\glstreenamfmt{\glstarget{##1}{\glossentryname{##1}}}%
8841   \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
8842   \space\glossentrydesc{##1}\glspostdescription\space##2\par
8843 }%
```

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.

```
8844 \renewcommand{\subglossentry}[3]{%
8845   \hangindent##1\glstreeindent\relax
8846   \parindent##1\glstreeindent\relax
8847   \ifnum##1=1\relax
8848     \glssubentryitem{##2}%
8849   \fi
8850   \glstarget{##2}{\strut}%
8851   \glossentrydesc{##2}\glspostdescription\space##3\par
8852 }%
```

Vertical gap between groups is the same as that used by indices:

```
8853 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
8854 }
```

`treenonamegroup` Like the `treenoname` style but the glossary groups have headings.

```
8855 \newglossarystyle{treenonamegroup}{%
```

Base it on the `glostyletreenoname` style:

```
8856 \setglossarystyle{treenoname}%
```

Give each group a heading:

```
8857 \renewcommand{\glsgroupheading}[1]{\par
8858   \noindent\glstreenamfmt{\glsgtgroup title{##1}}\par\indexspace}%
8859 }
```

`treenonamehypergroup` The `treenonamehypergroup` style is like the `treenonamegroup` style, but has a set of links to the groups at the start of the glossary.

```
8860 \newglossarystyle{treenonamehypergroup}{%
```

Base it on the `glostytreeoname` style:

```
8861 \setglossarystyle{treeoname}%
```

Put navigation links to the groups at the start of the `theglossary` environment:

```
8862 \renewcommand*{\glossaryheader}{%
8863   \par\noindent\glstreenamefmt{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
8864 \renewcommand*{\glsgroupheading}[1]{%
8865   \par\noindent
8866   \glstreenamefmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8867   \indexspace}%
8868 }
```

`\glssetwidest` `\glssetwidest[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.

```
8869 \newcommand*{\glssetwidest}[2][0]{%
8870   \expandafter\def\csname @glswidestname\romannumeral#1\endcsname{%
8871     #2}%
8872 }
```

`\@glswidestname` Initialise `\@glswidestname`.

```
8873 \newcommand*{\@glswidestname}{}%
```

`alttree` The `alttree` glossary style is similar in style to the `tree` style, but the indentation is obtained from the width of `\@glswidestname` which is set using `\glssetwidest`.

```
8874 \newglossarystyle{alttree}{%
```

Redefine the `theglossary` environment.

```
8875 \renewenvironment{theglossary}%
8876   {\def\@gls@prevlevel{-1}%
8877    \mbox{}\par}%
8878   {\par}%
```

Set the header and group headers to nothing.

```
8879 \renewcommand*{\glossaryheader}{}%
8880 \renewcommand*{\glsgroupheading}[1]{}%
```

Redefine the way that the level 0 entries are displayed.

```
8881 \renewcommand{\glosseentry}[2]{%
8882   \ifnum\@gls@prevlevel=0\relax
8883   \else
```

Find out how big the indentation should be by measuring the widest entry.

```
8884     \settowidth{\glstreeindent}{\glstreenamefmt{\@glswidestname\space}}%
8885     \fi
```

Set the hangindent and paragraph indent.

```
8886     \hangindent\glstreeindent
8887     \parindent\glstreeindent
```

Put the name to the left of the paragraph block.

```
8888 \makebox[0pt][r]{\makebox[\glstreeindent][l]{%
8889 \glstryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}}}%
```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
8890 \ifglshassymbol{##1}{(\glossentrysymbol{##1})\space}{}%
```

Do the description followed by the description terminator and location list.

```
8891 \glossentrydesc{##1}\glspostdescription \space ##2\par
```

Set the previous level to 0.

```
8892 \def\@gls@prevlevel{0}%
8893 }%
```

Redefine the way sub-entries are displayed.

```
8894 \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.

```
8895 \ifnum##1=1\relax
8896 \glssubentryitem{##2}%
8897 \fi
```

If the level hasn't changed, keep the same settings, otherwise adjust \glstreeindent accordingly.

```
8898 \ifnum\@gls@prevlevel=##1\relax
8899 \else
```

Compute the widest entry for this level, or for level 0 if not defined for this level.

Store in \gls@tmplen

```
8900 \@ifundefined{@glswidestname\romannumeral##1}{%
8901 \settowidth{\gls@tmplen}{\glstreenamefmt{\@glswidestname\space}}}%
8902 \settowidth{\gls@tmplen}{\glstreenamefmt{%
8903 \csname @glswidestname\romannumeral##1\endcsname\space}}}%
```

Determine if going up or down a level

```
8904 \ifnum\@gls@prevlevel<##1\relax
```

Depth has increased, so add the width of the widest entry to \glstreeindent.

```
8905 \setlength\glstreeindent\gls@tmplen
8906 \addtolength\glstreeindent\parindent
8907 \parindent\glstreeindent
8908 \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.

```
8909 \@ifundefined{@glswidestname\romannumeral\@gls@prevlevel}{%
8910 \settowidth{\glstreeindent}{\glstreenamefmt{%
8911 \@glswidestname\space}}}%
8912 \settowidth{\glstreeindent}{\glstreenamefmt{%
8913 \csname @glswidestname\romannumeral\@gls@prevlevel
8914 \endcsname\space}}}%
```

Subtract this length from the previous level's paragraph indent and set to
`\glstreeindent`.

```
8915      \addtolength\parindent{-\glstreeindent}%
8916      \setlength\glstreeindent\parindent
8917      \fi
8918      \fi
```

Set the hanging indentation.

```
8919      \hangindent\glstreeindent
```

Put the name to the left of the paragraph block

```
8920      \makebox[0pt][r]{\makebox[\glstemplen][l]{%
8921      \glstreenamfmt{\glstarget{##2}{\glossentryname{##2}}}}}%
```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
8922      \ifglshassymbol{##2}{(\glossentrysymbol{##2})\space}{}%
```

Do the description followed by the description terminator and location list.

```
8923      \glossentrydesc{##2}\glspostdescription\space ##3\par
```

Set the previous level macro to the current level.

```
8924      \def\@gls@prevlevel{##1}%
8925      }%
```

Vertical gap between groups is the same as that used by indices:

```
8926      \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
8927 }
```

almtreegroup Like the almtree style but the glossary groups have headings.

```
8928 \newglossarystyle{almtreegroup}{%
```

Base it on the glostylealmtree style:

```
8929 \setglossarystyle{almtree}%
```

Give each group a heading.

```
8930 \renewcommand{\glsgroupheading}[1]{\par
8931 \def\@gls@prevlevel{-1}%
8932 \hangindent0pt\relax
8933 \parindent0pt\relax
8934 \glstreenamfmt{\glsgetgrouptitle{##1}}\par\indexspace}%
8935 }
```

almtreehypergroup The almtreehypergroup style is like the almtreegroup style, but has a set of links to the groups at the start of the glossary.

```
8936 \newglossarystyle{almtreehypergroup}{%
```

Base it on the glostylealmtree style:

```
8937 \setglossarystyle{almtree}%
```

Put the navigation links in the header

```
8938 \renewcommand*{\glossaryheader}{%
8939 \par
```



```

8940 \def\@gls@prevlevel{-1}%
8941 \hangindent0pt\relax
8942 \parindent0pt\relax
8943 \glstreenamefmt{\glsnavigation}\par\indexspace}%

Put a hypertarget at the start of each group
8944 \renewcommand*\@gls@groupheading[1]{%
8945 \par
8946 \def\@gls@prevlevel{-1}%
8947 \hangindent0pt\relax
8948 \parindent0pt\relax
8949 \glstreenamefmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8950 \indexspace}}

```

6 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.

```

8951 \NeedsTeXFormat{LaTeX2e}
8952 \ProvidesPackage{glossaries-compatible-207}[2015/07/08 v4.16 (NLCT)]

```

`\GlsAddXdyAttribute` Adds an attribute in old format.

```

8953 \ifglsxindy
8954 \renewcommand*\GlsAddXdyAttribute[1]{%
8955 \edef\@xdyattributes{\@xdyattributes ^^J \string"#1\string"}%
8956 \expandafter\toks@\expandafter{\@xdylocref}%
8957 \edef\@xdylocref{\the\toks@ ^^J%
8958 (markup-locref
8959 :open \string"\string~n\string\setentrycounter
8960 {\noexpand\glscounter}%
8961 \expandafter\string\csname#1\endcsname
8962 \expandafter\@gobble\string\{\string" ^^J
8963 :close \string"\expandafter\@gobble\string\}\string" ^^J
8964 :attr \string"#1\string"))}

```

Only has an effect before `\writeist`:

```

8965 \fi

```

`\GlsAddXdyCounters`

```

8966 \renewcommand*\GlsAddXdyCounters[1]{%
8967 \GlossariesWarning{\string\GlsAddXdyCounters\space not available
8968 in compatibility mode.}%
8969 }

```

Add predefined attributes

```

8970 \GlsAddXdyAttribute{glsnumberformat}
8971 \GlsAddXdyAttribute{textrm}

```

```

8972 \GlsAddXdyAttribute{textsf}
8973 \GlsAddXdyAttribute{texttt}
8974 \GlsAddXdyAttribute{textbf}
8975 \GlsAddXdyAttribute{textmd}
8976 \GlsAddXdyAttribute{textit}
8977 \GlsAddXdyAttribute{textup}
8978 \GlsAddXdyAttribute{textsl}
8979 \GlsAddXdyAttribute{textsc}
8980 \GlsAddXdyAttribute{emph}
8981 \GlsAddXdyAttribute{glshypernumber}
8982 \GlsAddXdyAttribute{hyperrrm}
8983 \GlsAddXdyAttribute{hypersf}
8984 \GlsAddXdyAttribute{hypertt}
8985 \GlsAddXdyAttribute{hyperbf}
8986 \GlsAddXdyAttribute{hypermd}
8987 \GlsAddXdyAttribute{hyperit}
8988 \GlsAddXdyAttribute{hyperup}
8989 \GlsAddXdyAttribute{hypersl}
8990 \GlsAddXdyAttribute{hypersc}
8991 \GlsAddXdyAttribute{hyperemph}

```

\GlsAddXdyLocation Restore v2.07 definition:

```

8992 \ifglxsindy
8993 \renewcommand*{\GlsAddXdyLocation}[2]{%
8994 \edef\@xdyuserlocationdefs{%
8995 \@xdyuserlocationdefs ^^J%
8996 (define-location-class \string"#1\string"^^J\space\space
8997 \space(#2))
8998 }%
8999 \edef\@xdyuserlocationnames{%
9000 \@xdyuserlocationnames^^J\space\space\space
9001 \string"#1\string"}%
9002 }
9003 \fi

```

\@do@wrglossary

```

9004 \renewcommand{\@do@wrglossary}[1]{%
  Determine whether to use xindy or makeindex syntax
9005 \ifglxsindy
  Need to determine if the formatting information starts with a ( or ) indicating a
  range.
9006 \expandafter\@glo@check@mkidxrangechar\@glsnumberformat\@nil
9007 \def\@glo@range{}%
9008 \expandafter\if\@glo@prefix(\relax
9009 \def\@glo@range{:open-range}%
9010 \else
9011 \expandafter\if\@glo@prefix)\relax
9012 \def\@glo@range{:close-range}%

```

9013 \fi

9014 \fi

Get the location and escape any special characters

9015 \protected@edef\@glslocref{\theglsentrycounter}%

9016 \@gls@checkmkidxchars\@glslocref

Write to the glossary file using xindy syntax.

9017 \glossary[\csname glo@#1@type\endcsname]{%

9018 (indexentry :tkey (\csname glo@#1@index\endcsname)

9019 :locref \string"\@glslocref\string" %

9020 :attr \string"\@glo@suffix\string" \@glo@range

9021)

9022 }%

9023 \else

Convert the format information into the format required for makeindex

9024 \@set@glo@numformat\@glo@numfmt\@gls@counter\@glsnumberformat

Write to the glossary file using makeindex syntax.

9025 \glossary[\csname glo@#1@type\endcsname]{%

9026 \string\glossaryentry{\csname glo@#1@index\endcsname

9027 \@gls@encapchar\@glo@numfmt}{\theglsentrycounter}}%

9028 \fi

9029 }

\@set@glo@numformat Only had 3 arguments in v2.07

9030 \def\@set@glo@numformat#1#2#3{%

9031 \expandafter\@glo@check@mkidxrangechar#3\@nil

9032 \protected@edef#1{%

9033 \@glo@prefix setentrycounter[] {#2}%

9034 \expandafter\string\csname\@glo@suffix\endcsname

9035 }%

9036 \@gls@checkmkidxchars#1%

9037 }

\writeist Redefine \writeist back to the way it was in v2.07, but change \istfile to
 \glswrite.

9038 \ifglsxindy

9039 \def\writeist{%

9040 \openout\glswrite=\istfilename

9041 \write\glswrite{;; xindy style file created by the glossaries

9042 package in compatible-2.07 mode}%

9043 \write\glswrite{;; for document '\jobname' on

9044 \the\year-\the\month-\the\day}%

9045 \write\glswrite{^^J; required styles^^J}

9046 \@for\@xdystyle:=\@xdyrequiredstyles\do{%

9047 \ifx\@xdystyle\@empty

9048 \else

9049 \protected@write\glswrite{{(require

9050 \string"\@xdystyle.xdy\string")}}%

```

9051     \fi
9052 }%
9053 \write\glswrite{^^J%
9054     ; list of allowed attributes (number formats)^^J}%
9055 \write\glswrite{(define-attributes ((\@xdyattributes)))}%
9056 \write\glswrite{^^J; user defined alphabets^^J}%
9057 \write\glswrite{\@xdyuseralphabets}%
9058 \write\glswrite{^^J; location class definitions^^J}%
9059 \protected@edef\@gls@roman{\@roman{0}\string"
9060     \string"roman-numbers-lowercase\string" :sep \string"}}%
9061 \@onelevel@sanitize\@gls@roman
9062 \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
9063     :sep \string"}%
9064 \@onelevel@sanitize\@tmp
9065 \ifx\@tmp\@gls@roman
9066     \write\glswrite{(define-location-class
9067         \string"roman-page-numbers\string"^^J\space\space\space
9068         (\string"roman-numbers-lowercase\string")
9069         :min-range-length \@glsminrange)}}%
9070 \else
9071     \write\glswrite{(define-location-class
9072         \string"roman-page-numbers\string"^^J\space\space\space
9073         (:sep "\@gls@roman")
9074         :min-range-length \@glsminrange)}}%
9075 \fi
9076 \write\glswrite{(define-location-class
9077     \string"Roman-page-numbers\string"^^J\space\space\space
9078     (\string"roman-numbers-uppercase\string")
9079     :min-range-length \@glsminrange)}}%
9080 \write\glswrite{(define-location-class
9081     \string"arabic-page-numbers\string"^^J\space\space\space
9082     (\string"arabic-numbers\string")
9083     :min-range-length \@glsminrange)}}%
9084 \write\glswrite{(define-location-class
9085     \string"alpha-page-numbers\string"^^J\space\space\space
9086     (\string"alpha\string")
9087     :min-range-length \@glsminrange)}}%
9088 \write\glswrite{(define-location-class
9089     \string"Alpha-page-numbers\string"^^J\space\space\space
9090     (\string"ALPHA\string")
9091     :min-range-length \@glsminrange)}}%
9092 \write\glswrite{(define-location-class
9093     \string"Appendix-page-numbers\string"^^J\space\space\space
9094     (\string"ALPHA\string"
9095     :sep \string"\@glsAlphacompositor\string"
9096     \string"arabic-numbers\string")
9097     :min-range-length \@glsminrange)}}%
9098 \write\glswrite{(define-location-class
9099     \string"arabic-section-numbers\string"^^J\space\space\space

```

```

9100      (\string"arabic-numbers\string"
9101      :sep \string"\glscompositor\string"
9102      \string"arabic-numbers\string")
9103      :min-range-length \@glsminrange))}%
9104 \write\glswrite{^^J; user defined location classes}%
9105 \write\glswrite{\@xdyuserlocationdefs}%
9106 \write\glswrite{^^J; define cross-reference class^^J}%
9107 \write\glswrite{(define-crossref-class \string"see\string"
9108 :unverified )}%
9109 \write\glswrite{(markup-crossref-list
9110 :class \string"see\string"^^J\space\space\space
9111 :open \string"\string\glsseeformat\string"
9112 :close \string"{}\string"))}%
9113 \write\glswrite{^^J; define the order of the location classes}%
9114 \write\glswrite{(define-location-class-order
9115 (\@xdylocationclassorder))}%
9116 \write\glswrite{^^J; define the glossary markup^^J}%
9117 \write\glswrite{(markup-index^^J\space\space\space
9118 :open \string"\string
9119 \glossarysection[\string\glossarytoctitle]{\string
9120 \glossarytitle}\string\glossarypreamble\string~n\string\begin
9121 {theglossary}\string\glossaryheader\string~n\string" ^^J\space
9122 \space\space:close \string"\expandafter\@gobble
9123 \string\%\string~n\string
9124 \end{theglossary}\string\glossarypostamble
9125 \string~n\string" ^^J\space\space\space
9126 :tree)}}%
9127 \write\glswrite{(markup-letter-group-list
9128 :sep \string"\string\glsgroupskip\string~n\string"))}%
9129 \write\glswrite{(markup-indexentry
9130 :open \string"\string\relax \string\glsresetentrylist
9131 \string~n\string"))}%
9132 \write\glswrite{(markup-locclass-list :open
9133 \string"\glsopenbrace\string\glossaryentrynumbers
9134 \glsopenbrace\string\relax\space \string"^^J\space\space\space
9135 :sep \string", \string"
9136 :close \string"\glsclosebrace\glsclosebrace\string"))}%
9137 \write\glswrite{(markup-locref-list
9138 :sep \string"\string\delimN\space\string"))}%
9139 \write\glswrite{(markup-range
9140 :sep \string"\string\delimR\space\string"))}%
9141 \@onelevel@sanitize\gls@suffixF
9142 \@onelevel@sanitize\gls@suffixFF
9143 \ifx\gls@suffixF\@empty
9144 \else
9145 \write\glswrite{(markup-range
9146 :close "\gls@suffixF" :length 1 :ignore-end)}}%
9147 \fi
9148 \ifx\gls@suffixFF\@empty

```

```

9149 \else
9150 \write\glswrite{(markup-range
9151 :close "\gls@suffixFF" :length 2 :ignore-end)}}%
9152 \fi
9153 \write\glswrite{^^J; define format to use for locations^^J}%
9154 \write\glswrite{\@xdylocref}%
9155 \write\glswrite{^^J; define letter group list format^^J}%
9156 \write\glswrite{(markup-letter-group-list
9157 :sep \string\string\glsgroupskip\string~n\string)}}%
9158 \write\glswrite{^^J; letter group headings^^J}%
9159 \write\glswrite{(markup-letter-group
9160 :open-head \string\string\glsgroupheading
9161 \glsopenbrace\string^^J\space\space\space
9162 :close-head \string\glsclosebrace\string)}}%
9163 \write\glswrite{^^J; additional letter groups^^J}%
9164 \write\glswrite{\@xdylettergroups}%
9165 \write\glswrite{^^J; additional sort rules^^J}%
9166 \write\glswrite{\@xdysortrules}%
9167 \noist}
9168 \else
9169 \edef\@gls@actualchar{\string?}
9170 \edef\@gls@encapchar{\string|}
9171 \edef\@gls@levelchar{\string!}
9172 \edef\@gls@quotechar{\string"}
9173 \def\writeist{\relax
9174 \openout\glswrite=\istfilename
9175 \write\glswrite{\expandafter\@gobble\string\% makeindex style file
9176 created by the glossaries package}
9177 \write\glswrite{\expandafter\@gobble\string\% for document
9178 '\jobname' on \the\year-\the\month-\the\day}
9179 \write\glswrite{actual '\@gls@actualchar'}
9180 \write\glswrite{encap '\@gls@encapchar'}
9181 \write\glswrite{level '\@gls@levelchar'}
9182 \write\glswrite{quote '\@gls@quotechar'}
9183 \write\glswrite{keyword \string\string\glossaryentry\string"}
9184 \write\glswrite{preamble \string\string\glossarysection[\string
9185 \glossarytoctitle]{\string\glossarytitle}\string
9186 \glossarypreamble\string\n\string\begin{theglossary}\string
9187 \glossaryheader\string\n\string"}
9188 \write\glswrite{postamble \string\string\%\string\n\string
9189 \end{theglossary}\string\glossarypostamble\string\n
9190 \string"}
9191 \write\glswrite{group_skip \string\string\glsgroupskip\string\n
9192 \string"}
9193 \write\glswrite{item_0 \string\string\%\string\n\string"}
9194 \write\glswrite{item_1 \string\string\%\string\n\string"}
9195 \write\glswrite{item_2 \string\string\%\string\n\string"}
9196 \write\glswrite{item_01 \string\string\%\string\n\string"}
9197 \write\glswrite{item_x1

```

```

9198     \string"\string\relax \string\glsresetentrylist\string\n
9199     \string"}
9200     \write\glswrite{item_12 \string"\string%\string\n\string"}
9201     \write\glswrite{item_x2
9202     \string"\string\relax \string\glsresetentrylist\string\n
9203     \string"}
9204     \write\glswrite{delim_0 \string"\string\{\string
9205     \glossaryentrynumbers\string\{\string\relax \string"}
9206     \write\glswrite{delim_1 \string"\string\{\string
9207     \glossaryentrynumbers\string\{\string\relax \string"}
9208     \write\glswrite{delim_2 \string"\string\{\string
9209     \glossaryentrynumbers\string\{\string\relax \string"}
9210     \write\glswrite{delim_t \string"\string\}\string\}\string"}
9211     \write\glswrite{delim_n \string"\string\delimN \string"}
9212     \write\glswrite{delim_r \string"\string\delimR \string"}
9213     \write\glswrite{headings_flag 1}
9214     \write\glswrite{heading_prefix
9215     \string"\string\glsgruppeheading\string\{\string"}
9216     \write\glswrite{heading_suffix
9217     \string"\string\}\string\relax
9218     \string\glsresetentrylist \string"}
9219     \write\glswrite{symhead_positive \string"glssymbols\string"}
9220     \write\glswrite{numhead_positive \string"glssymbols\string"}
9221     \write\glswrite{page_compositor \string"glscpositor\string"}
9222     \@gls@escbsdq\gls@suffixF
9223     \@gls@escbsdq\gls@suffixFF
9224     \ifx\gls@suffixF\@empty
9225     \else
9226     \write\glswrite{suffix_2p \string"\gls@suffixF\string"}
9227     \fi
9228     \ifx\gls@suffixFF\@empty
9229     \else
9230     \write\glswrite{suffix_3p \string"\gls@suffixFF\string"}
9231     \fi
9232     \noist
9233   }
9234 \fi

```

\noist

```
9235 \renewcommand*{\noist}{\let\writeist\relax}
```

Compatibility macros.

```

9236 \NeedsTeXFormat{LaTeX2e}
9237 \ProvidesPackage{glossaries-compatible-307}[2015/07/08 v4.16 (NLCT)]

```

Compatibility macros for predefined glossary styles:

compatglossarystyle Defines a compatibility glossary style.

```

9238 \newcommand{\compatglossarystyle}[2]{%
9239   \ifcsundef{@glscompstyle#1}%

```

```

9240  {%
9241    \csdef{@glscompstyle@#1}{#2}%
9242  }%
9243  {%
9244    \PackageError{glossaries}{Glossary compatibility style ‘#1’ is already defined}{}%
9245  }%
9246 }

```

Backward compatible inline style.

```

9247 \compatglossarystyle{inline}{%
9248   \renewcommand{\glossaryentryfield}[5]{%
9249     \glsinlinedopostchild
9250     \gls@inlinesep
9251     \def\glo@desc{##3}%
9252     \def\@no@post@desc{\nopostdesc}%
9253     \glsentryitem{##1}\glsinlinenameformat{##1}{##2}%
9254     \ifx\glo@desc\@no@post@desc
9255       \glsinlineemptydescformat{##4}{##5}%
9256     \else
9257       \ifstrempy{##3}%
9258         {\glsinlineemptydescformat{##4}{##5}}%
9259         {\glsinlinedescformat{##3}{##4}{##5}}%
9260     \fi
9261     \ifglshaschildren{##1}%
9262     {%
9263       \glsresetsubentrycounter
9264       \glsinlineparentchildseparator
9265       \def\gls@inlinesubsep{}%
9266       \def\gls@inlinepostchild{\glsinlinepostchild}%
9267     }%
9268   }%
9269   \def\gls@inlinesep{\glsinlineseparator}%
9270 }%

```

Sub-entries display description:

```

9271 \renewcommand{\glossarysubentryfield}[6]{%
9272   \gls@inlinesubsep%
9273   \glsinlinesubnameformat{##2}{##3}%
9274   \glsesubentryitem{##2}\glsinlinesubdescformat{##4}{##5}{##6}%
9275   \def\gls@inlinesubsep{\glsinlinesubseparator}%
9276 }%
9277 }

```

Backward compatible list style.

```

9278 \compatglossarystyle{list}{%
9279   \renewcommand*{\glossaryentryfield}[5]{%
9280     \item[\glsentryitem{##1}\glstarget{##1}{##2}]
9281     ##3\glspostdescription\space ##5}%

```

Sub-entries continue on the same line:

```

9282 \renewcommand*{\glossarysubentryfield}[6]{%

```



```

9283 \glssubentryitem{##2}%
9284 \glstarget{##2}{\strut}##4\glspostdescription\space ##6.}%
9285 }

```

Backward compatible listgroup style.

```

9286 \compatglossarystyle{listgroup}{%
9287 \csuse{@glscmpstyle@list}%
9288 }%

```

Backward compatible listhypergroup style.

```

9289 \compatglossarystyle{listhypergroup}{%
9290 \csuse{@glscmpstyle@list}%
9291 }%

```

Backward compatible altlist style.

```

9292 \compatglossarystyle{altlist}{%
9293 \renewcommand*{\glossaryentryfield}[5]{%
9294 \item[\glssubentryitem{##1}\glstarget{##1}{##2}]%
9295 \mbox{}\par\nobreak\@afterheading
9296 ##3\glspostdescription\space ##5}%
9297 \renewcommand{\glossarysubentryfield}[6]{%
9298 \par
9299 \glssubentryitem{##2}%
9300 \glstarget{##2}{\strut}##4\glspostdescription\space ##6}%
9301 }%

```

Backward compatible altlistgroup style.

```

9302 \compatglossarystyle{altlistgroup}{%
9303 \csuse{@glscmpstyle@altlist}%
9304 }%

```

Backward compatible altlisthypergroup style.

```

9305 \compatglossarystyle{altlisthypergroup}{%
9306 \csuse{@glscmpstyle@altlist}%
9307 }%

```

Backward compatible listdotted style.

```

9308 \compatglossarystyle{listdotted}{%
9309 \renewcommand*{\glossaryentryfield}[5]{%
9310 \item[\makebox[\glslistdottedwidth][l]{%
9311 \glssubentryitem{##1}\glstarget{##1}{##2}%
9312 \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##3}%
9313 \renewcommand*{\glossarysubentryfield}[6]{%
9314 \item[\makebox[\glslistdottedwidth][l]{%
9315 \glssubentryitem{##2}%
9316 \glstarget{##2}{##3}%
9317 \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##4}%
9318 }%

```

Backward compatible sublistdotted style.

```

9319 \compatglossarystyle{sublistdotted}{%
9320 \csuse{@glscmpstyle@listdotted}%

```

```

9321 \renewcommand*{\glossaryentryfield}[5]{%
9322 \item[\glentryitem{##1}\glstarget{##1}{##2}]}%
9323 }%

```

Backward compatible long style.

```

9324 \compatglossarystyle{long}{%
9325 \renewcommand*{\glossaryentryfield}[5]{%
9326 \glentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
9327 \renewcommand*{\glossarysubentryfield}[6]{%
9328 &
9329 \glssubentryitem{##2}%
9330 \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
9331 }%

```

Backward compatible longborder style.

```

9332 \compatglossarystyle{longborder}{%
9333 \csuse{@glscmpstyle@long}%
9334 }%

```

Backward compatible longheader style.

```

9335 \compatglossarystyle{longheader}{%
9336 \csuse{@glscmpstyle@long}%
9337 }%

```

Backward compatible longheaderborder style.

```

9338 \compatglossarystyle{longheaderborder}{%
9339 \csuse{@glscmpstyle@long}%
9340 }%

```

Backward compatible long3col style.

```

9341 \compatglossarystyle{long3col}{%
9342 \renewcommand*{\glossaryentryfield}[5]{%
9343 \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\\}%
9344 \renewcommand*{\glossarysubentryfield}[6]{%
9345 &
9346 \glssubentryitem{##2}%
9347 \glstarget{##2}{\strut}##4 & ##6\\}%
9348 }%

```

Backward compatible long3colborder style.

```

9349 \compatglossarystyle{long3colborder}{%
9350 \csuse{@glscmpstyle@long3col}%
9351 }%

```

Backward compatible long3colheader style.

```

9352 \compatglossarystyle{long3colheader}{%
9353 \csuse{@glscmpstyle@long3col}%
9354 }%

```

Backward compatible long3colheaderborder style.

```

9355 \compatglossarystyle{long3colheaderborder}{%
9356 \csuse{@glscmpstyle@long3col}%
9357 }%

```

Backward compatible long4col style.

```
9358 \compatglossarystyle{long4col}{%
9359   \renewcommand*{\glossaryentryfield}[5]{%
9360     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%
9361   \renewcommand*{\glossarysubentryfield}[6]{%
9362     &
9363     \glssubentryitem{##2}%
9364     \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%
9365 }%
```

Backward compatible long4colheader style.

```
9366 \compatglossarystyle{long4colheader}{%
9367   \csuse{@glscmpstyle@long4col}%
9368 }%
```

Backward compatible long4colborder style.

```
9369 \compatglossarystyle{long4colborder}{%
9370   \csuse{@glscmpstyle@long4col}%
9371 }%
```

Backward compatible long4colheaderborder style.

```
9372 \compatglossarystyle{long4colheaderborder}{%
9373   \csuse{@glscmpstyle@long4col}%
9374 }%
```

Backward compatible altlong4col style.

```
9375 \compatglossarystyle{altlong4col}{%
9376   \csuse{@glscmpstyle@long4col}%
9377 }%
```

Backward compatible altlong4colheader style.

```
9378 \compatglossarystyle{altlong4colheader}{%
9379   \csuse{@glscmpstyle@long4col}%
9380 }%
```

Backward compatible altlong4colborder style.

```
9381 \compatglossarystyle{altlong4colborder}{%
9382   \csuse{@glscmpstyle@long4col}%
9383 }%
```

Backward compatible altlong4colheaderborder style.

```
9384 \compatglossarystyle{altlong4colheaderborder}{%
9385   \csuse{@glscmpstyle@long4col}%
9386 }%
```

Backward compatible long style.

```
9387 \compatglossarystyle{longragged}{%
9388   \renewcommand*{\glossaryentryfield}[5]{%
9389     \glentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
9390     \tabularnewline}%
9391   \renewcommand*{\glossarysubentryfield}[6]{%
9392     &
```

```

9393     \glssubentryitem{##2}%
9394     \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
9395     \tabularnewline}%
9396 }%

```

Backward compatible longraggedborder style.

```

9397 \compatglossarystyle{longraggedborder}{%
9398   \csuse{@glscmpstyle@longragged}%
9399 }%

```

Backward compatible longraggedheader style.

```

9400 \compatglossarystyle{longraggedheader}{%
9401   \csuse{@glscmpstyle@longragged}%
9402 }%

```

Backward compatible longraggedheaderborder style.

```

9403 \compatglossarystyle{longraggedheaderborder}{%
9404   \csuse{@glscmpstyle@longragged}%
9405 }%

```

Backward compatible longragged3col style.

```

9406 \compatglossarystyle{longragged3col}{%
9407   \renewcommand*{\glossaryentryfield}[5]{%
9408     \glssubentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%
9409   \renewcommand*{\glossarysubentryfield}[6]{%
9410     &
9411     \glssubentryitem{##2}%
9412     \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%
9413 }%

```

Backward compatible longragged3colborder style.

```

9414 \compatglossarystyle{longragged3colborder}{%
9415   \csuse{@glscmpstyle@longragged3col}%
9416 }%

```

Backward compatible longragged3colheader style.

```

9417 \compatglossarystyle{longragged3colheader}{%
9418   \csuse{@glscmpstyle@longragged3col}%
9419 }%

```

Backward compatible longragged3colheaderborder style.

```

9420 \compatglossarystyle{longragged3colheaderborder}{%
9421   \csuse{@glscmpstyle@longragged3col}%
9422 }%

```

Backward compatible altlongragged4col style.

```

9423 \compatglossarystyle{altlongragged4col}{%
9424   \renewcommand*{\glossaryentryfield}[5]{%
9425     \glssubentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%
9426   \renewcommand*{\glossarysubentryfield}[6]{%
9427     &
9428     \glssubentryitem{##2}%

```

```

9429 \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%
9430 }%

```

Backward compatible altlongragged4colheader style.

```

9431 \compatglossarystyle{altlongragged4colheader}{%
9432 \csuse{@glscmpstyle@altlong4col}%
9433 }%

```

Backward compatible altlongragged4colborder style.

```

9434 \compatglossarystyle{altlongragged4colborder}{%
9435 \csuse{@glscmpstyle@altlong4col}%
9436 }%

```

Backward compatible altlongragged4colheaderborder style.

```

9437 \compatglossarystyle{altlongragged4colheaderborder}{%
9438 \csuse{@glscmpstyle@altlong4col}%
9439 }%

```

Backward compatible index style.

```

9440 \compatglossarystyle{index}{%
9441 \renewcommand*{\glossaryentryfield}[5]{%
9442 \item\glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
9443 \ifx\relax##4\relax
9444 \else
9445 \space(##4)%
9446 \fi
9447 \space ##3\glspostdescription \space ##5}%
9448 \renewcommand*{\glossarysubentryfield}[6]{%
9449 \ifcase##1\relax
9450 % level 0
9451 \item
9452 \or
9453 % level 1
9454 \subitem
9455 \glssubentryitem{##2}%
9456 \else
9457 % all other levels
9458 \subsubitem
9459 \fi
9460 \textbf{\glstarget{##2}{##3}}%
9461 \ifx\relax##5\relax
9462 \else
9463 \space(##5)%
9464 \fi
9465 \space##4\glspostdescription\space ##6}%
9466 }%

```

Backward compatible indexgroup style.

```

9467 \compatglossarystyle{indexgroup}{%
9468 \csuse{@glscmpstyle@index}%
9469 }%

```

Backward compatible indexhypergroup style.

```
9470 \compatglossarystyle{indexhypergroup}{%
9471 \csuse{@glscmpstyle@index}%
9472 }%
```

Backward compatible tree style.

```
9473 \compatglossarystyle{tree}{%
9474 \renewcommand{\glossaryentryfield}[5]{%
9475 \hangindent0pt\relax
9476 \parindent0pt\relax
9477 \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
9478 \ifx\relax##4\relax
9479 \else
9480 \space(##4)%
9481 \fi
9482 \space ##3\glspostdescription \space ##5\par}%
9483 \renewcommand{\glossarysubentryfield}[6]{%
9484 \hangindent##1\glstreeindent\relax
9485 \parindent##1\glstreeindent\relax
9486 \ifnum##1=1\relax
9487 \glssubentryitem{##2}%
9488 \fi
9489 \textbf{\glstarget{##2}{##3}}%
9490 \ifx\relax##5\relax
9491 \else
9492 \space(##5)%
9493 \fi
9494 \space##4\glspostdescription\space ##6\par}%
9495 }%
```

Backward compatible treegroup style.

```
9496 \compatglossarystyle{treegroup}{%
9497 \csuse{@glscmpstyle@tree}%
9498 }%
```

Backward compatible treehypergroup style.

```
9499 \compatglossarystyle{treehypergroup}{%
9500 \csuse{@glscmpstyle@tree}%
9501 }%
```

Backward compatible treenoname style.

```
9502 \compatglossarystyle{treenoname}{%
9503 \renewcommand{\glossaryentryfield}[5]{%
9504 \hangindent0pt\relax
9505 \parindent0pt\relax
9506 \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
9507 \ifx\relax##4\relax
9508 \else
9509 \space(##4)%
9510 \fi
9511 \space ##3\glspostdescription \space ##5\par}%

```

```

9512 \renewcommand{\glossarysubentryfield}[6]{%
9513   \hangindent##1\glstreeindent\relax
9514   \parindent##1\glstreeindent\relax
9515   \ifnum##1=1\relax
9516     \glssubentryitem{##2}%
9517   \fi
9518   \glstarget{##2}{\strut}%
9519   ##4\glspostdescription\space ##6\par}%
9520 }%

```

Backward compatible treenonamegroup style.

```

9521 \compatglossarystyle{treenonamegroup}{%
9522   \csuse{@glscmpstyle@treenoname}%
9523 }%

```

Backward compatible treenonamehypergroup style.

```

9524 \compatglossarystyle{treenonamehypergroup}{%
9525   \csuse{@glscmpstyle@treenoname}%
9526 }%

```

Backward compatible alttree style.

```

9527 \compatglossarystyle{alttree}{%
9528   \renewcommand{\glossaryentryfield}[5]{%
9529     \ifnum \@gls@prevlevel=0\relax
9530     \else
9531       \settowidth{\glstreeindent}{\textbf{\@glswidestname\space}}%
9532       \hangindent\glstreeindent
9533       \parindent\glstreeindent
9534     \fi
9535     \makebox[0pt][r]{\makebox[\glstreeindent][l]{%
9536       \glssubentryitem{##1}\textbf{\glstarget{##1}{##2}}}%
9537     \ifx\relax##4\relax
9538     \else
9539       (##4)\space
9540     \fi
9541     ##3\glspostdescription \space ##5\par
9542     \def\@gls@prevlevel{0}%
9543   }%
9544   \renewcommand{\glossarysubentryfield}[6]{%
9545     \ifnum##1=1\relax
9546       \glssubentryitem{##2}%
9547     \fi
9548     \ifnum \@gls@prevlevel=##1\relax
9549     \else
9550       \@ifundefined{@glswidestname\romannumeral##1}{%
9551         \settowidth{\gls@tmplen}{\textbf{\@glswidestname\space}}{%
9552         \settowidth{\gls@tmplen}{\textbf{%
9553           \csname @glswidestname\romannumeral##1\endcsname\space}}}%
9554       \ifnum \@gls@prevlevel<##1\relax
9555         \setlength\glstreeindent\gls@tmplen
9556         \addtolength\glstreeindent\parindent

```

```

9557         \parindent\glstreeindent
9558     \else
9559         \@ifundefined{@glswidestname\romannumeral\@gls@prevlevel}{%
9560             \settowidth{\glstreeindent}{\textbf{%
9561                 \@glswidestname\space}}}{%
9562             \settowidth{\glstreeindent}{\textbf{%
9563                 \csname @glswidestname\romannumeral\@gls@prevlevel
9564                     \endcsname\space}}}{%
9565             \addtolength\parindent{-\glstreeindent}%
9566             \setlength\glstreeindent\parindent
9567         \fi
9568     \fi
9569     \hangindent\glstreeindent
9570     \makebox[0pt][r]{\makebox[\@gls@tmplen][l]{%
9571         \textbf{\glstarget{##2}{##3}}}}}%
9572     \ifx##5\relax\relax
9573     \else
9574         (##5)\space
9575     \fi
9576     ##4\glspostdescription\space ##6\par
9577     \def\@gls@prevlevel{##1}%
9578 }%
9579 }%

```

Backward compatible alttreegroup style.

```

9580 \compatglossarystyle{alttreegroup}{%
9581 \csuse{@glscompstyle@alttree}%
9582 }%

```

Backward compatible alttreehypergroup style.

```

9583 \compatglossarystyle{alttreehypergroup}{%
9584 \csuse{@glscompstyle@alttree}%
9585 }%

```

Backward compatible mcolindex style.

```

9586 \compatglossarystyle{mcolindex}{%
9587 \csuse{@glscompstyle@index}%
9588 }%

```

Backward compatible mcolindexgroup style.

```

9589 \compatglossarystyle{mcolindexgroup}{%
9590 \csuse{@glscompstyle@index}%
9591 }%

```

Backward compatible mcolindexhypergroup style.

```

9592 \compatglossarystyle{mcolindexhypergroup}{%
9593 \csuse{@glscompstyle@index}%
9594 }%

```

Backward compatible mcoltree style.

```

9595 \compatglossarystyle{mcoltree}{%
9596 \csuse{@glscompstyle@tree}%

```


9597 }%

Backward compatible mcoltreegroup style.

9598 \compatglossarystyle{mcolindextreegroup}{%
9599 \csuse{@glscompstyle@tree}%
9600 }%

Backward compatible mcoltreehypergroup style.

9601 \compatglossarystyle{mcolindextreehypergroup}{%
9602 \csuse{@glscompstyle@tree}%
9603 }%

Backward compatible mcoltreenoname style.

9604 \compatglossarystyle{mcoltreenoname}{%
9605 \csuse{@glscompstyle@tree}%
9606 }%

Backward compatible mcoltreenonamegroup style.

9607 \compatglossarystyle{mcoltreenonamegroup}{%
9608 \csuse{@glscompstyle@tree}%
9609 }%

Backward compatible mcoltreenonamehypergroup style.

9610 \compatglossarystyle{mcoltreenonamehypergroup}{%
9611 \csuse{@glscompstyle@tree}%
9612 }%

Backward compatible mcolalmtree style.

9613 \compatglossarystyle{mcolalmtree}{%
9614 \csuse{@glscompstyle@almtree}%
9615 }%

Backward compatible mcolalmtreegroup style.

9616 \compatglossarystyle{mcolalmtreegroup}{%
9617 \csuse{@glscompstyle@almtree}%
9618 }%

Backward compatible mcolalmtreehypergroup style.

9619 \compatglossarystyle{mcolalmtreehypergroup}{%
9620 \csuse{@glscompstyle@almtree}%
9621 }%

Backward compatible superragged style.

9622 \compatglossarystyle{superragged}{%
9623 \renewcommand*{\glossaryentryfield}[5]{%
9624 \glsentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
9625 \tabularnewline}%
9626 \renewcommand*{\glossarysubentryfield}[6]{%
9627 &
9628 \glssubentryitem{##2}%
9629 \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
9630 \tabularnewline}%
9631 }%

Backward compatible superraggedborder style.

```
9632 \compatglossarystyle{superraggedborder}{%  
9633 \csuse{@glscmpstyle@superragged}%  
9634 }%
```

Backward compatible superraggedheader style.

```
9635 \compatglossarystyle{superraggedheader}{%  
9636 \csuse{@glscmpstyle@superragged}%  
9637 }%
```

Backward compatible superraggedheaderborder style.

```
9638 \compatglossarystyle{superraggedheaderborder}{%  
9639 \csuse{@glscmpstyle@superragged}%  
9640 }%
```

Backward compatible superragged3col style.

```
9641 \compatglossarystyle{superragged3col}{%  
9642 \renewcommand*{\glossaryentryfield}[5]{%  
9643 \glstentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%  
9644 \renewcommand*{\glossarysubentryfield}[6]{%  
9645 &  
9646 \glssubentryitem{##2}%  
9647 \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%  
9648 }%
```

Backward compatible superragged3colborder style.

```
9649 \compatglossarystyle{superragged3colborder}{%  
9650 \csuse{@glscmpstyle@superragged3col}%  
9651 }%
```

Backward compatible superragged3colheader style.

```
9652 \compatglossarystyle{superragged3colheader}{%  
9653 \csuse{@glscmpstyle@superragged3col}%  
9654 }%
```

Backward compatible superragged3colheaderborder style.

```
9655 \compatglossarystyle{superragged3colheaderborder}{%  
9656 \csuse{@glscmpstyle@superragged3col}%  
9657 }%
```

Backward compatible altsuperragged4col style.

```
9658 \compatglossarystyle{altsuperragged4col}{%  
9659 \renewcommand*{\glossaryentryfield}[5]{%  
9660 \glstentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%  
9661 \renewcommand*{\glossarysubentryfield}[6]{%  
9662 &  
9663 \glssubentryitem{##2}%  
9664 \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%  
9665 }%
```

Backward compatible altsuperragged4colheader style.

```
9666 \compatglossarystyle{altsuperragged4colheader}{%
```

```
9667 \csuse{@glscompstyle@altsuperragged4col}%
9668 }%
```

Backward compatible altsuperragged4colborder style.

```
9669 \compatglossarystyle{altsuperragged4colborder}{%
9670 \csuse{@glscompstyle@altsuperragged4col}%
9671 }%
```

Backward compatible altsuperragged4colheaderborder style.

```
9672 \compatglossarystyle{altsuperragged4colheaderborder}{%
9673 \csuse{@glscompstyle@altsuperragged4col}%
9674 }%
```

Backward compatible super style.

```
9675 \compatglossarystyle{super}{%
9676 \renewcommand*{\glossaryentryfield}[5]{%
9677 \glstentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
9678 \renewcommand*{\glossarysubentryfield}[6]{%
9679 &
9680 \glssubentryitem{##2}%
9681 \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
9682 }%
```

Backward compatible superborder style.

```
9683 \compatglossarystyle{superborder}{%
9684 \csuse{@glscompstyle@super}%
9685 }%
```

Backward compatible superheader style.

```
9686 \compatglossarystyle{superheader}{%
9687 \csuse{@glscompstyle@super}%
9688 }%
```

Backward compatible superheaderborder style.

```
9689 \compatglossarystyle{superheaderborder}{%
9690 \csuse{@glscompstyle@super}%
9691 }%
```

Backward compatible super3col style.

```
9692 \compatglossarystyle{super3col}{%
9693 \renewcommand*{\glossaryentryfield}[5]{%
9694 \glstentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\\}%
9695 \renewcommand*{\glossarysubentryfield}[6]{%
9696 &
9697 \glssubentryitem{##2}%
9698 \glstarget{##2}{\strut}##4 & ##6\\}%
9699 }%
```

Backward compatible super3colborder style.

```
9700 \compatglossarystyle{super3colborder}{%
9701 \csuse{@glscompstyle@super3col}%
9702 }%
```

Backward compatible super3colheader style.

```
9703 \compatglossarystyle{super3colheader}{%  
9704 \csuse{@glscmpstyle@super3col}%  
9705 }%
```

Backward compatible super3colheaderborder style.

```
9706 \compatglossarystyle{super3colheaderborder}{%  
9707 \csuse{@glscmpstyle@super3col}%  
9708 }%
```

Backward compatible super4col style.

```
9709 \compatglossarystyle{super4col}{%  
9710 \renewcommand*{\glossaryentryfield}[5]{%  
9711 \glstryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%  
9712 \renewcommand*{\glossarysubentryfield}[6]{%  
9713 &  
9714 \glssubentryitem{##2}%  
9715 \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%  
9716 }%
```

Backward compatible super4colheader style.

```
9717 \compatglossarystyle{super4colheader}{%  
9718 \csuse{@glscmpstyle@super4col}%  
9719 }%
```

Backward compatible super4colborder style.

```
9720 \compatglossarystyle{super4colborder}{%  
9721 \csuse{@glscmpstyle@super4col}%  
9722 }%
```

Backward compatible super4colheaderborder style.

```
9723 \compatglossarystyle{super4colheaderborder}{%  
9724 \csuse{@glscmpstyle@super4col}%  
9725 }%
```

Backward compatible altsuper4col style.

```
9726 \compatglossarystyle{altsuper4col}{%  
9727 \csuse{@glscmpstyle@super4col}%  
9728 }%
```

Backward compatible altsuper4colheader style.

```
9729 \compatglossarystyle{altsuper4colheader}{%  
9730 \csuse{@glscmpstyle@super4col}%  
9731 }%
```

Backward compatible altsuper4colborder style.

```
9732 \compatglossarystyle{altsuper4colborder}{%  
9733 \csuse{@glscmpstyle@super4col}%  
9734 }%
```

Backward compatible altsuper4colheaderborder style.

```
9735 \compatglossarystyle{altsuper4colheaderborder}{%  
9736 \csuse{@glscmpstyle@super4col}%  
9737 }%
```

7 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.

```
9738 \NeedsTeXFormat{LaTeX2e}
```

Package version number now in line with main glossaries package number but will only be updated when glossaries-accsupp.sty is modified.

```
9739 \ProvidesPackage{glossaries-accsupp}[2015/07/08 v4.16 (NLCT)]
```

```
9740 Experimental glossaries accessibility
```

Pass all options to glossaries:

```
9741 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
9742 \ProcessOptions
```

compatibleglossentry Override style compatibility macros:

```
9743 \def\compatibleglossentry#1#2{%
9744   \toks@{#2}%
9745   \protected@edef\@do@glossentry{%
9746     \noexpand\accsuppglossaryentryfield{#1}%
9747     {\noexpand\glsnamefont
9748       {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@name\endcsname}}}%
9749     {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@desc\endcsname}%
9750     {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@symbol\endcsname}%
9751     {\the\toks@}%
9752   }%
9753   \@do@glossentry
9754 }
```

compatiblesubglossentry

```
9755 \def\compatiblesubglossentry#1#2#3{%
9756   \toks@{#3}%
9757   \protected@edef\@do@subglossentry{%
9758     \noexpand\accsuppglossarysubentryfield{\number#1}%
9759     {#2}%
9760     {\noexpand\glsnamefont
9761       {\expandafter\expandonce\csname glo@\glsdetoklabel{#2}@name\endcsname}}}%
9762     {\expandafter\expandonce\csname glo@\glsdetoklabel{#2}@desc\endcsname}%
9763     {\expandafter\expandonce\csname glo@\glsdetoklabel{#2}@symbol\endcsname}%
9764     {\the\toks@}%
9765   }%
9766   \@do@subglossentry
9767 }
```

Required packages:

```
9768 \RequirePackage{glossaries}
```

```
9769 \RequirePackage{accsupp}
```

7.1 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:

```
9770 \define@key{glossentry}{access}{%
9771   \def\@glo@access{#1}%
9772 }
```

textaccess The replacement text corresponding to the text key:

```
9773 \define@key{glossentry}{textaccess}{%
9774   \def\@glo@textaccess{#1}%
9775 }
```

firstaccess The replacement text corresponding to the first key:

```
9776 \define@key{glossentry}{firstaccess}{%
9777   \def\@glo@firstaccess{#1}%
9778 }
```

pluralaccess The replacement text corresponding to the plural key:

```
9779 \define@key{glossentry}{pluralaccess}{%
9780   \def\@glo@pluralaccess{#1}%
9781 }
```

firstpluralaccess The replacement text corresponding to the firstplural key:

```
9782 \define@key{glossentry}{firstpluralaccess}{%
9783   \def\@glo@firstpluralaccess{#1}%
9784 }
```

symbolaccess The replacement text corresponding to the symbol key:

```
9785 \define@key{glossentry}{symbolaccess}{%
9786   \def\@glo@symbolaccess{#1}%
9787 }
```

symbolpluralaccess The replacement text corresponding to the symbolplural key:

```
9788 \define@key{glossentry}{symbolpluralaccess}{%
9789   \def\@glo@symbolpluralaccess{#1}%
9790 }
```

descriptionaccess The replacement text corresponding to the description key:

```
9791 \define@key{glossentry}{descriptionaccess}{%
9792   \def\@glo@descaccess{#1}%
9793 }
```

descriptionpluralaccess The replacement text corresponding to the descriptionplural key:

```
9794 \define@key{glossentry}{descriptionpluralaccess}{%
9795   \def\@glo@descpluralaccess{#1}%
9796 }
```

shortaccess The replacement text corresponding to the short key:

```
9797 \define@key{glossentry}{shortaccess}{%
9798   \def\@glo@shortaccess{#1}%
9799 }
```

shortpluralaccess The replacement text corresponding to the shortplural key:

```
9800 \define@key{glossentry}{shortpluralaccess}{%
9801   \def\@glo@shortpluralaccess{#1}%
9802 }
```

longaccess The replacement text corresponding to the long key:

```
9803 \define@key{glossentry}{longaccess}{%
9804   \def\@glo@longaccess{#1}%
9805 }
```

longpluralaccess The replacement text corresponding to the longplural key:

```
9806 \define@key{glossentry}{longpluralaccess}{%
9807   \def\@glo@longpluralaccess{#1}%
9808 }
```

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:

```
9809 \appto\@gls@keymap{,%
9810   {access}{access},%
9811   {textaccess}{textaccess},%
9812   {firstaccess}{firstaccess},%
9813   {pluralaccess}{pluralaccess},%
9814   {firstpluralaccess}{firstpluralaccess},%
9815   {symbolaccess}{symbolaccess},%
9816   {symbolpluralaccess}{symbolpluralaccess},%
9817   {descaccess}{descaccess},%
9818   {descpluralaccess}{descpluralaccess},%
9819   {shortaccess}{shortaccess},%
9820   {shortpluralaccess}{shortpluralaccess},%
9821   {longaccess}{longaccess},%
9822   {longpluralaccess}{longpluralaccess}%
9823 }
```

\@gls@noaccess Indicates that no replacement text has been provided.

```
9824 \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):

```

9825 \let\@gls@oldnewglossaryentryprehook\@newglossaryentryprehook
9826 \renewcommand*{\@newglossaryentryprehook}{%
9827   \@gls@oldnewglossaryentryprehook
9828   \def\@glo@access{\@glo@symbol}%

```

Initialise the other keys:

```

9829   \def\@glo@textaccess{\@glo@access}%
9830   \def\@glo@firstaccess{\@glo@access}%
9831   \def\@glo@pluralaccess{\@glo@textaccess}%
9832   \def\@glo@firstpluralaccess{\@glo@pluralaccess}%
9833   \def\@glo@symbolaccess{\relax}%
9834   \def\@glo@symbolpluralaccess{\@glo@symbolaccess}%
9835   \def\@glo@descaccess{\relax}%
9836   \def\@glo@descpluralaccess{\@glo@descaccess}%
9837   \def\@glo@shortaccess{\relax}%
9838   \def\@glo@shortpluralaccess{\@glo@shortaccess}%
9839   \def\@glo@longaccess{\relax}%
9840   \def\@glo@longpluralaccess{\@glo@longaccess}%
9841 }

```

Add to the end hook:

```

9842 \let\@gls@oldnewglossaryentryposthook\@newglossaryentryposthook
9843 \renewcommand*{\@newglossaryentryposthook}{%
9844   \@gls@oldnewglossaryentryposthook

```

Store the access information:

```

9845   \expandafter
9846     \protected@xdef\csname glo@\@glo@label @access\endcsname{%
9847     \@glo@access}%
9848   \expandafter
9849     \protected@xdef\csname glo@\@glo@label @textaccess\endcsname{%
9850     \@glo@textaccess}%
9851   \expandafter
9852     \protected@xdef\csname glo@\@glo@label @firstaccess\endcsname{%
9853     \@glo@firstaccess}%
9854   \expandafter
9855     \protected@xdef\csname glo@\@glo@label @pluralaccess\endcsname{%
9856     \@glo@pluralaccess}%
9857   \expandafter
9858     \protected@xdef\csname glo@\@glo@label @firstpluralaccess\endcsname{%
9859     \@glo@firstpluralaccess}%
9860   \expandafter
9861     \protected@xdef\csname glo@\@glo@label @symbolaccess\endcsname{%
9862     \@glo@symbolaccess}%
9863   \expandafter
9864     \protected@xdef\csname glo@\@glo@label @symbolpluralaccess\endcsname{%
9865     \@glo@symbolpluralaccess}%
9866   \expandafter

```



```

9867 \protected@xdef\csname glo@\@glo@label @descaccess\endcsname{%
9868 \@glo@descaccess}%
9869 \expandafter
9870 \protected@xdef\csname glo@\@glo@label @descpluralaccess\endcsname{%
9871 \@glo@descpluralaccess}%
9872 \expandafter
9873 \protected@xdef\csname glo@\@glo@label @shortaccess\endcsname{%
9874 \@glo@shortaccess}%
9875 \expandafter
9876 \protected@xdef\csname glo@\@glo@label @shortpluralaccess\endcsname{%
9877 \@glo@shortpluralaccess}%
9878 \expandafter
9879 \protected@xdef\csname glo@\@glo@label @longaccess\endcsname{%
9880 \@glo@longaccess}%
9881 \expandafter
9882 \protected@xdef\csname glo@\@glo@label @longpluralaccess\endcsname{%
9883 \@glo@longpluralaccess}%
9884 }

```

7.2 Accessing Replacement Text

`\glsentryaccess` Get the value of the access key for the entry with the given label:

```

9885 \newcommand*{\glsentryaccess}[1]{%
9886 \@gls@entry@field{#1}{access}%
9887 }

```

`\glsentrytextaccess` Get the value of the textaccess key for the entry with the given label:

```

9888 \newcommand*{\glsentrytextaccess}[1]{%
9889 \@gls@entry@field{#1}{textaccess}%
9890 }

```

`\glsentryfirstaccess` Get the value of the firstaccess key for the entry with the given label:

```

9891 \newcommand*{\glsentryfirstaccess}[1]{%
9892 \@gls@entry@field{#1}{firstaccess}%
9893 }

```

`\glsentrypluralaccess` Get the value of the pluralaccess key for the entry with the given label:

```

9894 \newcommand*{\glsentrypluralaccess}[1]{%
9895 \@gls@entry@field{#1}{pluralaccess}%
9896 }

```

`\glsentryfirstpluralaccess` Get the value of the firstpluralaccess key for the entry with the given label:

```

9897 \newcommand*{\glsentryfirstpluralaccess}[1]{%
9898 \csname glo@#1@firstpluralaccess\endcsname
9899 }

```

`\glsentrysymbolaccess` Get the value of the symbolaccess key for the entry with the given label:

```

9900 \newcommand*{\glsentrysymbolaccess}[1]{%

```

```

9901 \@gls@entry@field{#1}{symbolaccess}%
9902 }

symbolpluralaccess  Get the value of the symbolpluralaccess key for the entry with the given label:
9903 \newcommand*{\glsentrysymbolpluralaccess}[1]{%
9904 \@gls@entry@field{#1}{symbolpluralaccess}%
9905 }

\glsentrydescaccess  Get the value of the descriptionaccess key for the entry with the given label:
9906 \newcommand*{\glsentrydescaccess}[1]{%
9907 \@gls@entry@field{#1}{descaccess}%
9908 }

entrydescpluralaccess  Get the value of the descriptionpluralaccess key for the entry with the given label:
9909 \newcommand*{\glsentrydescpluralaccess}[1]{%
9910 \@gls@entry@field{#1}{descaccess}%
9911 }

\glsentryshortaccess  Get the value of the shortaccess key for the entry with the given label:
9912 \newcommand*{\glsentryshortaccess}[1]{%
9913 \@gls@entry@field{#1}{shortaccess}%
9914 }

entryshortpluralaccess  Get the value of the shortpluralaccess key for the entry with the given label:
9915 \newcommand*{\glsentryshortpluralaccess}[1]{%
9916 \@gls@entry@field{#1}{shortpluralaccess}%
9917 }

\glsentrylongaccess  Get the value of the longaccess key for the entry with the given label:
9918 \newcommand*{\glsentrylongaccess}[1]{%
9919 \@gls@entry@field{#1}{longaccess}%
9920 }

entrylongpluralaccess  Get the value of the longpluralaccess key for the entry with the given label:
9921 \newcommand*{\glsentrylongpluralaccess}[1]{%
9922 \@gls@entry@field{#1}{longpluralaccess}%
9923 }

\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.)
9924 \newcommand*{\glsaccsupp}[2]{%
9925 \BeginAccSupp{ActualText=#1}#2\EndAccSupp{}%
9926 }

```

`\xglsaccsupp` Fully expands replacement text before calling `\glsaccsupp`

```

9927 \newcommand*{\xglsaccsupp}[2]{%
9928   \protected@edef\xgls@replacementtext{#1}%
9929   \expandafter\xglsaccsupp\expandafter{\xgls@replacementtext}{#2}%
9930 }

```

`\@gls@access@display`

```

9931 \newcommand*{\@gls@access@display}[2]{%
9932   \protected@edef\@glo@access{#2}%
9933   \ifx\@glo@access\xgls@noaccess
9934     #1%
9935   \else
9936     \xglsaccsupp{\@glo@access}{#1}%
9937   \fi
9938 }

```

`\@glsname@access@display` Displays the first argument with the accessibility text for the entry with the label given by the second argument (if set).

```

9939 \DeclareRobustCommand*\@glsname@access@display[2]{%
9940   \@gls@access@display{#1}{\glsentryaccess{#2}}%
9941 }

```

`\@gls@text@access@display` As above but for the `\textaccess` replacement text.

```

9942 \DeclareRobustCommand*\@gls@text@access@display[2]{%
9943   \@gls@access@display{#1}{\glsentrytextaccess{#2}}%
9944 }

```

`\@gls@plural@access@display` As above but for the `\pluralaccess` replacement text.

```

9945 \DeclareRobustCommand*\@gls@plural@access@display[2]{%
9946   \@gls@access@display{#1}{\glsentrypluralaccess{#2}}%
9947 }

```

`\@gls@first@access@display` As above but for the `\firstaccess` replacement text.

```

9948 \DeclareRobustCommand*\@gls@first@access@display[2]{%
9949   \@gls@access@display{#1}{\glsentryfirstaccess{#2}}%
9950 }

```

`\@gls@first@plural@access@display` As above but for the `\firstpluralaccess` replacement text.

```

9951 \DeclareRobustCommand*\@gls@first@plural@access@display[2]{%
9952   \@gls@access@display{#1}{\glsentryfirstpluralaccess{#2}}%
9953 }

```

`\@gls@symbol@access@display` As above but for the `\symbolaccess` replacement text.

```

9954 \DeclareRobustCommand*\@gls@symbol@access@display[2]{%
9955   \@gls@access@display{#1}{\glsentrysymbolaccess{#2}}%
9956 }

```

pluralaccessdisplay As above but for the symbolpluralaccess replacement text.

```

9957 \DeclareRobustCommand*\glsymbolpluralaccessdisplay}[2]{%
9958   \@gls@access@display{#1}{\glsentrysymbolpluralaccess{#2}}}%
9959 }

```

descriptionaccessdisplay As above but for the descriptionaccess replacement text.

```

9960 \DeclareRobustCommand*\glsdescriptionaccessdisplay}[2]{%
9961   \@gls@access@display{#1}{\glsentrydescaccess{#2}}}%
9962 }

```

descriptionpluralaccessdisplay As above but for the descriptionpluralaccess replacement text.

```

9963 \DeclareRobustCommand*\glsdescriptionpluralaccessdisplay}[2]{%
9964   \@gls@access@display{#1}{\glsentrydescpluralaccess{#2}}}%
9965 }

```

shortaccessdisplay As above but for the shortaccess replacement text.

```

9966 \DeclareRobustCommand*\glsshortaccessdisplay}[2]{%
9967   \@gls@access@display{#1}{\glsentryshortaccess{#2}}}%
9968 }

```

shortpluralaccessdisplay As above but for the shortpluralaccess replacement text.

```

9969 \DeclareRobustCommand*\glsshortpluralaccessdisplay}[2]{%
9970   \@gls@access@display{#1}{\glsentryshortpluralaccess{#2}}}%
9971 }

```

longaccessdisplay As above but for the longaccess replacement text.

```

9972 \DeclareRobustCommand*\glslongaccessdisplay}[2]{%
9973   \@gls@access@display{#1}{\glsentrylongaccess{#2}}}%
9974 }

```

longpluralaccessdisplay As above but for the longpluralaccess replacement text.

```

9975 \DeclareRobustCommand*\glslongpluralaccessdisplay}[2]{%
9976   \@gls@access@display{#1}{\glsentrylongpluralaccess{#2}}}%
9977 }

```

\glsaccessdisplay Gets the replacement text corresponding to the named key given by the first argument and calls the appropriate command defined above.

```

9978 \DeclareRobustCommand*\glsaccessdisplay}[3]{%
9979   \@ifundefined{gls#1accessdisplay}%
9980   {%
9981     \PackageError{glossaries-accsupp}{No accessibility support
9982       for key ‘#1’}{}%
9983   }%
9984   {%
9985     \csname gls#1accessdisplay\endcsname{#2}{#3}%
9986   }%
9987 }

```

ls@default@entryfmt Redefine the default entry format to use accessibility information

```

9988 \renewcommand*{\@@gls@default@entryfmt}[2]{%
9989   \ifdefempty\glscustomtext
9990   {%
9991     \glsifplural
9992     {%

```

Plural form

```

9993     \glscapscase
9994     {%

```

Don't adjust case

```

9995     \ifglsused\glslabel
9996     {%

```

Subsequent use

```

9997         #2{\glspluralaccessdisplay
9998           {\glsentryplural{\glslabel}}{\glslabel}}%
9999         {\glsdescriptionpluralaccessdisplay
10000         {\glsentrydescplural{\glslabel}}{\glslabel}}%
10001         {\glsymbolpluralaccessdisplay
10002         {\glsentrysymbolplural{\glslabel}}{\glslabel}}
10003         {\glsinsert}}%
10004     }%
10005     {%

```

First use

```

10006         #1{\glsfirstpluralaccessdisplay
10007           {\glsentryfirstplural{\glslabel}}{\glslabel}}%
10008         {\glsdescriptionpluralaccessdisplay
10009         {\glsentrydescplural{\glslabel}}{\glslabel}}%
10010         {\glsymbolpluralaccessdisplay
10011         {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10012         {\glsinsert}}%
10013     }%
10014 }%
10015 {%

```

Make first letter upper case

```

10016     \ifglsused\glslabel
10017     {%

```

Subsequent use.

```

10018         #2{\glspluralaccessdisplay
10019           {\Glsentryplural{\glslabel}}{\glslabel}}%
10020         {\glsdescriptionpluralaccessdisplay
10021         {\glsentrydescplural{\glslabel}}{\glslabel}}%
10022         {\glsymbolpluralaccessdisplay
10023         {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10024         {\glsinsert}}%
10025     }%
10026     {%

```

First use

```

10027      #1{\glsfirstpluralaccessdisplay
10028          {\Glsentryfirstplural{\glslabel}}{\glslabel}}%
10029      {\glsdescriptionpluralaccessdisplay
10030          {\glsentrydescplural{\glslabel}}{\glslabel}}%
10031      {\glssymbolpluralaccessdisplay
10032          {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10033      {\glsinsert}}%
10034      }%
10035      }%
10036      {%

```

Make all upper case

```

10037      \ifglsused\glslabel
10038      {%

```

Subsequent use

```

10039      \MakeUppercase{%
10040      #2{\glspluralaccessdisplay
10041          {\glsentryplural{\glslabel}}{\glslabel}}%
10042      {\glsdescriptionpluralaccessdisplay
10043          {\glsentrydescplural{\glslabel}}{\glslabel}}%
10044      {\glssymbolpluralaccessdisplay
10045          {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10046      {\glsinsert}}%
10047      }%
10048      {%

```

First use

```

10049      \MakeUppercase{%
10050      #1{\glsfirstpluralaccessdisplay
10051          {\glsentryfirstplural{\glslabel}}{\glslabel}}%
10052      {\glsdescriptionpluralaccessdisplay
10053          {\glsentrydescplural{\glslabel}}{\glslabel}}%
10054      {\glssymbolpluralaccessdisplay
10055          {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10056      {\glsinsert}}%
10057      }%
10058      }%
10059      }%
10060      {%

```

Singular form

```

10061      \glscapscase
10062      {%

```

Don't adjust case

```

10063      \ifglsused\glslabel
10064      {%

```

Subsequent use

```

10065      #2{\glstextaccessdisplay
10066          {\glentrytext{\glslabel}}{\glslabel}}%
10067      {\glsdescriptionaccessdisplay
10068          {\glentrydesc{\glslabel}}{\glslabel}}%
10069      {\glssymbolaccessdisplay
10070          {\glentrysymbol{\glslabel}}{\glslabel}}%
10071      {\glsinsert}%
10072  }%
10073  {%

```

First use

```

10074      #1{\glsfirstaccessdisplay
10075          {\glentryfirst{\glslabel}}{\glslabel}}%
10076      {\glsdescriptionaccessdisplay
10077          {\glentrydesc{\glslabel}}{\glslabel}}%
10078      {\glssymbolaccessdisplay
10079          {\glentrysymbol{\glslabel}}{\glslabel}}%
10080      {\glsinsert}%
10081  }%
10082  }%
10083  {%

```

Make first letter upper case

```

10084      \ifglsused\glslabel
10085      {%

```

Subsequent use

```

10086      #2{\glstextaccessdisplay
10087          {\Glsentrytext{\glslabel}}{\glslabel}}%
10088      {\glsdescriptionaccessdisplay
10089          {\glentrydesc{\glslabel}}{\glslabel}}%
10090      {\glssymbolaccessdisplay
10091          {\glentrysymbol{\glslabel}}{\glslabel}}%
10092      {\glsinsert}%
10093  }%
10094  {%

```

First use

```

10095      #1{\glsfirstaccessdisplay
10096          {\Glsentryfirst{\glslabel}}{\glslabel}}%
10097      {\glsdescriptionaccessdisplay
10098          {\glentrydesc{\glslabel}}{\glslabel}}%
10099      {\glssymbolaccessdisplay
10100          {\glentrysymbol{\glslabel}}{\glslabel}}%
10101      {\glsinsert}%
10102  }%
10103  }%
10104  {%

```

Make all upper case

```

10105      \ifglsused\glslabel
10106      {%

```

Subsequent use

```

10107      \MakeUppercase{%
10108          #2{\glstextaccessdisplay
10109              {\glentrytext{\glslabel}}{\glslabel}}%
10110              {\glsdescriptionaccessdisplay
10111                  {\glentrydesc{\glslabel}}{\glslabel}}%
10112              {\glssymbolaccessdisplay
10113                  {\glentrysymbol{\glslabel}}{\glslabel}}%
10114              {\glsinsert}}}%
10115      }%
10116      {%

```

First use

```

10117      \MakeUppercase{%
10118          #1{\glsfirstaccessdisplay
10119              {\glentryfirst{\glslabel}}{\glslabel}}%
10120              {\glsdescriptionaccessdisplay
10121                  {\glentrydesc{\glslabel}}{\glslabel}}%
10122              {\glssymbolaccessdisplay
10123                  {\glentrysymbol{\glslabel}}{\glslabel}}%
10124              {\glsinsert}}}%
10125      }%
10126      }%
10127      }%
10128      }%
10129      {%

```

Custom text provided in \glsdisp

```

10130      \ifglsused{\glslabel}%
10131      {%

```

Subsequent use

```

10132      #2{\glscustomtext}%
10133      {\glsdescriptionaccessdisplay
10134          {\glentrydesc{\glslabel}}{\glslabel}}%
10135      {\glssymbolaccessdisplay
10136          {\glentrysymbol{\glslabel}}{\glslabel}}%
10137      {\glsinsert}}%
10138      }%
10139      {%

```

First use

```

10140      #1{\glscustomtext}%
10141      {\glsdescriptionaccessdisplay
10142          {\glentrydesc{\glslabel}}{\glslabel}}%
10143      {\glssymbolaccessdisplay
10144          {\glentrysymbol{\glslabel}}{\glslabel}}%
10145      {\glsinsert}}%
10146      }%
10147      }%
10148      }

```


`\glsgenentryfmt` Redefine to use accessibility information.

```
10149 \renewcommand*{\glsgenentryfmt}{%
10150   \ifdefempty\glscustomtext
10151   {%
10152     \glsifplural
10153     {%
```

Plural form

```
10154     \glscapscase
10155     {%
```

Don't adjust case

```
10156     \ifglsused\glslabel
10157     {%
```

Subsequent use

```
10158     \glspluralaccessdisplay
10159     {\glentryplural{\glslabel}}{\glslabel}%
10160     \glsinsert
10161     }%
10162     {%
```

First use

```
10163     \glsfirstpluralaccessdisplay
10164     {\glentryfirstplural{\glslabel}}{\glslabel}%
10165     \glsinsert
10166     }%
10167     }%
10168     {%
```

Make first letter upper case

```
10169     \ifglsused\glslabel
10170     {%
```

Subsequent use.

```
10171     \glspluralaccessdisplay
10172     {\Glsentryplural{\glslabel}}{\glslabel}%
10173     \glsinsert
10174     }%
10175     {%
```

First use

```
10176     \glsfirstpluralaccessdisplay
10177     {\Glsentryfirstplural{\glslabel}}{\glslabel}%
10178     \glsinsert
10179     }%
10180     }%
10181     {%
```

Make all upper case

```
10182     \ifglsused\glslabel
10183     {%
```

Subsequent use

```
10184      \glspluralaccessdisplay
10185      {\mfirstucMakeUppercase{\glsentryplural{\glslabel}}}%
10186      {\glslabel}%
10187      \mfirstucMakeUppercase{\glsinsert}%
10188  }%
10189  {%
```

First use

```
10190      \glsfirstpluralaccessdisplay
10191      {\mfirstucMakeUppercase{\glsentryfirstplural{\glslabel}}}%
10192      {\glslabel}%
10193      \mfirstucMakeUppercase{\glsinsert}%
10194  }%
10195  }%
10196  }%
10197  {%
```

Singular form

```
10198      \glscapscase
10199      {%
```

Don't adjust case

```
10200      \ifglsused\glslabel
10201      {%
```

Subsequent use

```
10202      \glstextaccessdisplay{\glsentrytext{\glslabel}}{\glslabel}%
10203      \glsinsert
10204  }%
10205  {%
```

First use

```
10206      \glsfirstaccessdisplay{\glsentryfirst{\glslabel}}{\glslabel}%
10207      \glsinsert
10208  }%
10209  }%
10210  {%
```

Make first letter upper case

```
10211      \ifglsused\glslabel
10212      {%
```

Subsequent use

```
10213      \glstextaccessdisplay{\Glsentrytext{\glslabel}}{\glslabel}%
10214      \glsinsert
10215  }%
10216  {%
```

First use

```
10217      \glsfirstaccessdisplay{\Glsentryfirst{\glslabel}}{\glslabel}%
10218      \glsinsert
```

```

10219     }%
10220     }%
10221     {%

```

Make all upper case

```

10222     \ifglused\glslabel
10223     {%

```

Subsequent use

```

10224     \glstextaccessdisplay
10225     {\mfirstucMakeUppercase{\glentrytext{\glslabel}}}{\glslabel}%
10226     \mfirstucMakeUppercase{\glsinsert}%
10227     }%
10228     {%

```

First use

```

10229     \glsfirstaccessdisplay
10230     {\mfirstucMakeUppercase{\glentryfirst{\glslabel}}}{\glslabel}%
10231     \mfirstucMakeUppercase{\glsinsert}%
10232     }%
10233     }%
10234     }%
10235     }%
10236     {%

```

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.

```

10237     \glscustomtext\glsinsert
10238     }%
10239 }

```

\glsгенacfmt Redefine to include accessibility information.

```

10240 \renewcommand*{\glsгенacfmt}{%
10241     \ifdefempty\glscustomtext
10242     {%
10243         \ifglused\glslabel
10244         {%

```

Subsequent use:

```

10245     \glsifplural
10246     {%

```

Subsequent plural form:

```

10247     \glscapscase
10248     {%

```

Subsequent plural form, don't adjust case:

```

10249     \acronymfont
10250     {\glsshortpluralaccessdisplay
10251     {\glentryshortpl{\glslabel}}{\glslabel}}%
10252     \glsinsert

```

10253 }%
 10254 {%

Subsequent plural form, make first letter upper case:

10255 \acronymfont
 10256 {\glsshortpluralaccessdisplay
 10257 {\Glsentryshortpl{\glslabel}}{\glslabel}}%
 10258 \glsinsert
 10259 }%
 10260 {%

Subsequent plural form, all caps:

10261 \mfirstucMakeUppercase
 10262 {\acronymfont
 10263 {\glsshortpluralaccessdisplay
 10264 {\Glsentryshortpl{\glslabel}}{\glslabel}}%
 10265 \glsinsert}%
 10266 }%
 10267 }%
 10268 {%

Subsequent singular form

10269 \glscapscase
 10270 {%

Subsequent singular form, don't adjust case:

10271 \acronymfont
 10272 {\glsshortaccessdisplay{\Glsentryshort{\glslabel}}{\glslabel}}%
 10273 \glsinsert
 10274 }%
 10275 {%

Subsequent singular form, make first letter upper case:

10276 \acronymfont
 10277 {\glsshortaccessdisplay{\Glsentryshort{\glslabel}}{\glslabel}}%
 10278 \glsinsert
 10279 }%
 10280 {%

Subsequent singular form, all caps:

10281 \mfirstucMakeUppercase
 10282 {\acronymfont{%
 10283 \glsshortaccessdisplay{\Glsentryshort{\glslabel}}{\glslabel}}%
 10284 \glsinsert}%
 10285 }%
 10286 }%
 10287 }%
 10288 {%

First use:

10289 \glsifplural
 10290 {%

First use plural form:

```
10291      \glscapscase
10292      {%
```

First use plural form, don't adjust case:

```
10293      \genplacrfullformat{\glslabel}{\glsinsert}%
10294      }%
10295      {%
```

First use plural form, make first letter upper case:

```
10296      \Genplacrfullformat{\glslabel}{\glsinsert}%
10297      }%
10298      {%
```

First use plural form, all caps:

```
10299      \mfirstucMakeUppercase
10300      {\genplacrfullformat{\glslabel}{\glsinsert}}%
10301      }%
10302      }%
10303      {%
```

First use singular form

```
10304      \glscapscase
10305      {%
```

First use singular form, don't adjust case:

```
10306      \genacrfullformat{\glslabel}{\glsinsert}%
10307      }%
10308      {%
```

First use singular form, make first letter upper case:

```
10309      \Genacrfullformat{\glslabel}{\glsinsert}%
10310      }%
10311      {%
```

First use singular form, all caps:

```
10312      \mfirstucMakeUppercase
10313      {\genacrfullformat{\glslabel}{\glsinsert}}%
10314      }%
10315      }%
10316      }%
10317      }%
10318      {%
```

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.

```
10319      \glscustomtext
10320      }%
10321 }
```

`\genacrfullformat` Redefine to include accessibility information.

```
10322 \renewcommand*{\genacrfullformat}[2]{%
```

```

10323 \glslongaccessdisplay{\glsentrylong{#1}}{#1}#2\space
10324 (\glsshortaccessdisplay{\protect\firstacronymfont{\glsentryshort{#1}}}{#1})%
10325 }

```

\Genacrfullformat Redefine to include accessibility information.

```

10326 \renewcommand*{\Genacrfullformat}[2]{%
10327 \glslongaccessdisplay{\Glsentrylong{#1}}{#1}#2\space
10328 (\glsshortaccessdisplay{\protect\firstacronymfont{\Glsentryshort{#1}}}{#1})%
10329 }

```

\genplacrfullformat Redefine to include accessibility information.

```

10330 \renewcommand*{\genplacrfullformat}[2]{%
10331 \glslongpluralaccessdisplay{\glsentrylongpl{#1}}{#1}#2\space
10332 (\glsshortpluralaccessdisplay
10333 {\protect\firstacronymfont{\glsentryshortpl{#1}}}{#1})%
10334 }

```

\Genplacrfullformat Redefine to include accessibility information.

```

10335 \renewcommand*{\Genplacrfullformat}[2]{%
10336 \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}#2\space
10337 (\glsshortpluralaccessdisplay
10338 {\protect\firstacronymfont{\Glsentryshortpl{#1}}}{#1})%
10339 }

```

\@acrshort

```

10340 \def\@acrshort#1#2[#3]{%
10341 \glsdoifexists{#2}%
10342 {%
10343 \let\do@gls@link@checkfirsthyper\relax

10344 \let\glsifplural\@secondoftwo
10345 \let\glsifcaps\@firstofthree
10346 \let\glsinsert\@empty
10347 \def\glscustomtext{%
10348 \acronymfont{\glsshortaccessdisplay{\glsentryshort{#2}}{#2}}#3%
10349 }%

```

Call \@gls@link

```

10350 \@gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
10351 }%

10352 \glspostlinkhook
10353 }

```

\@Acrshort

```

10354 \def\@Acrshort#1#2[#3]{%
10355 \glsdoifexists{#2}%
10356 {%
10357 \let\do@gls@link@checkfirsthyper\relax

```

```

10358 \let\glsifplural\@secondoftwo
10359 \let\glscapscase\@secondofthree
10360 \let\glsinsert\@empty
10361 \def\glscustomtext{%
10362 \acronymfont{\glsshortaccessdisplay{\Glsentryshort{#2}}{#2}}#3%
10363 }%

Call \@gls@link
10364 \@gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
10365 }%

10366 \glspostlinkhook
10367 }

```

\@ACRshort

```

10368 \def\@ACRshort#1#2[#3]{%
10369 \glsdoifexists{#2}%
10370 {%
10371 \let\do@gls@link@checkfirsthyper\relax

10372 \let\glsifplural\@secondoftwo
10373 \let\glscapscase\@thirdofthree
10374 \let\glsinsert\@empty
10375 \def\glscustomtext{%
10376 \acronymfont{\glsshortaccessdisplay
10377 {\MakeUppercase{\Glsentryshort{#2}}}{#2}}#3%
10378 }%

Call \@gls@link
10379 \@gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
10380 }%

10381 \glspostlinkhook
10382 }

```

\@acrlong

```

10383 \def\@acrlong#1#2[#3]{%
10384 \glsdoifexists{#2}%
10385 {%
10386 \let\do@gls@link@checkfirsthyper\relax

10387 \let\glsifplural\@secondoftwo
10388 \let\glscapscase\@firstofthree
10389 \let\glsinsert\@empty
10390 \def\glscustomtext{%
10391 \acronymfont{\glslongaccessdisplay{\Glsentrylong{#2}}{#2}}#3%
10392 }%

Call \@gls@link
10393 \@gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
10394 }%

```

```

10395 \glspostlinkhook
10396 }

\@Acrlong
10397 \def\@Acrlong#1#2[#3]{%
10398   \glstoifexists{#2}%
10399   {%
10400     \let\do@gl@link@checkfirsthyper\relax

10401     \let\gl@ifplural\@secondoftwo
10402     \let\gl@scapscase\@firstofthree
10403     \let\gl@insert\@empty
10404     \def\gl@customtext{%
10405       \acronymfont{\gl@longaccessdisplay{\Glsentrylong{#2}}{#2}}#3%
10406     }%

    Call \@gl@link
10407     \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
10408   }%

10409   \glspostlinkhook
10410 }

\@ACRlong
10411 \def\@ACRlong#1#2[#3]{%
10412   \glstoifexists{#2}%
10413   {%
10414     \let\do@gl@link@checkfirsthyper\relax

10415     \let\gl@ifplural\@secondoftwo
10416     \let\gl@scapscase\@firstofthree
10417     \let\gl@insert\@empty
10418     \def\gl@customtext{%
10419       \acronymfont{\gl@longaccessdisplay{%
10420         \MakeUppercase{\Glsentrylong{#2}}}{#2}}#3}%
10421     }%

    Call \@gl@link
10422     \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
10423   }%

10424   \glspostlinkhook
10425 }

```

7.3 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.

```

10426 \renewcommand*{\glossentryname}[1]{%
10427   \glsdoifexists{#1}%
10428   {%
10429     \glsnamefont{\glsnameaccessdisplay{\glsentryname{#1}}{#1}}%
10430   }%
10431 }

10432 \renewcommand*{\glossentryname}[1]{%
10433   \glsdoifexists{#1}%
10434   {%
10435     \glsnamefont{\glsnameaccessdisplay{\Glsentryname{#1}}{#1}}%
10436   }%
10437 }

10438 \renewcommand*{\glossentrydesc}[1]{%
10439   \glsdoifexists{#1}%
10440   {%
10441     \glsdescriptionaccessdisplay{\glsentrydesc{#1}}{#1}%
10442   }%
10443 }

10444 \renewcommand*{\Glossentrydesc}[1]{%
10445   \glsdoifexists{#1}%
10446   {%
10447     \glsdescriptionaccessdisplay{\Glsentrydesc{#1}}{#1}%
10448   }%
10449 }

10450 \renewcommand*{\glossentrysymbol}[1]{%
10451   \glsdoifexists{#1}%
10452   {%
10453     \glssymbolaccessdisplay{\glsentrysymbol{#1}}{#1}%
10454   }%
10455 }

10456 \renewcommand*{\Glossentrysymbol}[1]{%
10457   \glsdoifexists{#1}%
10458   {%
10459     \glssymbolaccessdisplay{\Glsentrysymbol{#1}}{#1}%
10460   }%
10461 }

```

`pglossaryentryfield`

```

10462 \newcommand*{\accsuppglossaryentryfield}[5]{%
10463   \glossaryentryfield{#1}%
10464   {\glsnameaccessdisplay{#2}{#1}}%
10465   {\glsdescriptionaccessdisplay{#3}{#1}}%
10466   {\glssymbolaccessdisplay{#4}{#1}}{#5}%
10467 }

```

ossarysubentryfield

```

10468 \newcommand*\accsuppglossarysubentryfield}[6]{%
10469   \glossarysubentryfield{#1}{#2}%
10470   {\glslnameaccessdisplay{#3}{#2}}%
10471   {\glslsdescriptionaccessdisplay{#4}{#2}}%
10472   {\glssymbolaccessdisplay{#5}{#2}}{#6}%
10473 }

```

7.4 Acronyms

Redefine acronym styles provided by glossaries:

long-short *<long>* (*<short>*) acronym style.

```

10474 \renewacronymstyle{long-short}%
10475 {%

```

Check for long form in case this is a mixed glossary.

```

10476   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
10477 }%
10478 {%
10479   \renewcommand*\GenericAcronymFields{description={\the\glslongtok}}%
10480   \renewcommand*\Genacrfullformat}[2]{%
10481     \glslongaccessdisplay{\glsenentrylong{##1}}{##1}##2\space
10482     (\glsshortaccessdisplay
10483       {\protect\firstacronymfont{\glsenentryshort{##1}}}{##1})%
10484   }%
10485   \renewcommand*\Genacrfullformat}[2]{%
10486     \glslongaccessdisplay{\Glsenentrylong{##1}}{##1}##2\space
10487     (\glsshortaccessdisplay
10488       {\protect\firstacronymfont{\glsenentryshort{##1}}}{##1})%
10489   }%
10490   \renewcommand*\Genplacrfullformat}[2]{%
10491     \glslongpluralaccessdisplay{\glsenentrylongpl{##1}}{##1}##2\space
10492     (\glsshortpluralaccessdisplay
10493       {\protect\firstacronymfont{\glsenentryshortpl{##1}}}{##1})%
10494   }%
10495   \renewcommand*\Genplacrfullformat}[2]{%
10496     \glslongpluralaccessdisplay{\Glsenentrylongpl{##1}}{##1}##2\space
10497     (\glsshortpluralaccessdisplay
10498       {\protect\firstacronymfont{\glsenentryshortpl{##1}}}{##1})%
10499   }%
10500   \renewcommand*\acronymentry}[1]{%
10501     \glsshortaccessdisplay{\acronymfont{\glsenentryshort{##1}}}{##1}}
10502   \renewcommand*\acronymsort}[2]{##1}%
10503   \renewcommand*\acronymfont}[1]{##1}%
10504   \renewcommand*\firstacronymfont}[1]{\acronymfont{##1}}%
10505   \renewcommand*\acrpluralsuffix{\glspluralsuffix}%
10506 }

```

short-long *<short>* (*<long>*) acronym style.

```
10507 \renewacronymstyle{short-long}%
10508 {%
    Check for long form in case this is a mixed glossary.
10509   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
10510 }%
10511 {%
10512   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
10513   \renewcommand*{\genacrfullformat}[2]{%
10514     \glsshortaccessdisplay
10515     {\protect\firstacronymfont{\glsenentryshort{##1}}}{##1}##2\space
10516     (\glslongaccessdisplay{\glsenentrylong{##1}}{##1})%
10517   }%
10518   \renewcommand*{\Genacrfullformat}[2]{%
10519     \glsshortaccessdisplay
10520     {\protect\firstacronymfont{\Glsenentryshort{##1}}}{##1}##2\space
10521     (\glslongaccessdisplay{\glsenentrylong{##1}}{##1})%
10522   }%
10523   \renewcommand*{\genplacrfullformat}[2]{%
10524     \glsshortpluralaccessdisplay
10525     {\protect\firstacronymfont{\glsenentryshortpl{##1}}}{##1}##2\space
10526     (\glslongpluralaccessdisplay
10527     {\glsenentrylongpl{##1}}{##1})%
10528   }%
10529   \renewcommand*{\Genplacrfullformat}[2]{%
10530     \glsshortpluralaccessdisplay
10531     {\protect\firstacronymfont{\Glsenentryshortpl{##1}}}{##1}##2\space
10532     (\glslongpluralaccessdisplay{\glsenentrylongpl{##1}}{##1})%
10533   }%
10534   \renewcommand*{\acronymentry}[1]{%
10535     \glsshortaccessdisplay{\acronymfont{\glsenentryshort{##1}}}{##1}%
10536   \renewcommand*{\acronymsort}[2]{##1}%
10537   \renewcommand*{\acronymfont}[1]{##1}%
10538   \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
10539   \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
10540 }
```

long-short-desc *<long>* (*<short>*) acronym style that has an accompanying description (which the user needs to supply).

```
10541 \renewacronymstyle{long-short-desc}%
10542 {%
10543   \GlsUseAcrEntryDisplayStyle{long-short}%
10544 }%
10545 {%
10546   \GlsUseAcrStyleDefs{long-short}%
10547   \renewcommand*{\GenericAcronymFields}{}%
10548   \renewcommand*{\acronymsort}[2]{##2}%
10549   \renewcommand*{\acronymentry}[1]{%
```

```

10550    \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10551    (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}})%
10552 }

```

long-sc-short-desc *<long>* (\textsc{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

10553 \renewacronymstyle{long-sc-short-desc}%
10554 {%
10555   \GlsUseAcrEntryDispStyle{long-sc-short}%
10556 }%
10557 {%
10558   \GlsUseAcrStyleDefs{long-sc-short}%
10559   \renewcommand*{\GenericAcronymFields}{}%
10560   \renewcommand*{\acronymsort}[2]{##2}%
10561   \renewcommand*{\acronymentry}[1]{%
10562     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10563     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}})%
10564 }

```

long-sm-short-desc *<long>* (\textsmaller{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

10565 \renewacronymstyle{long-sm-short-desc}%
10566 {%
10567   \GlsUseAcrEntryDispStyle{long-sm-short}%
10568 }%
10569 {%
10570   \GlsUseAcrStyleDefs{long-sm-short}%
10571   \renewcommand*{\GenericAcronymFields}{}%
10572   \renewcommand*{\acronymsort}[2]{##2}%
10573   \renewcommand*{\acronymentry}[1]{%
10574     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10575     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}})%
10576 }

```

short-long-desc *<short>* ({<long>}) acronym style that has an accompanying description (which the user needs to supply).

```

10577 \renewacronymstyle{short-long-desc}%
10578 {%
10579   \GlsUseAcrEntryDispStyle{short-long}%
10580 }%
10581 {%
10582   \GlsUseAcrStyleDefs{short-long}%
10583   \renewcommand*{\GenericAcronymFields}{}%
10584   \renewcommand*{\acronymsort}[2]{##2}%
10585   \renewcommand*{\acronymentry}[1]{%
10586     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10587     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}})%
10588 }

```

sc-short-long-desc *<long>* (\textsc{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

10589 \renewacronymstyle{sc-short-long-desc}%
10590 {%
10591   \GlsUseAcrEntryDispStyle{sc-short-long}%
10592 }%
10593 {%
10594   \GlsUseAcrStyleDefs{sc-short-long}%
10595   \renewcommand*{\GenericAcronymFields}{}%
10596   \renewcommand*{\acronymsort}[2]{##2}%
10597   \renewcommand*{\acronymentry}[1]{%
10598     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10599     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10600 }
```

sm-short-long-desc *<long>* (\textsmaller{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

10601 \renewacronymstyle{sm-short-long-desc}%
10602 {%
10603   \GlsUseAcrEntryDispStyle{sm-short-long}%
10604 }%
10605 {%
10606   \GlsUseAcrStyleDefs{sm-short-long}%
10607   \renewcommand*{\GenericAcronymFields}{}%
10608   \renewcommand*{\acronymsort}[2]{##2}%
10609   \renewcommand*{\acronymentry}[1]{%
10610     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10611     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10612 }
```

dua *<long>* only acronym style.

```

10613 \renewacronymstyle{dua}%
10614 {%

    Check for long form in case this is a mixed glossary.

10615   \ifdefempty\glscustomtext
10616   {%
10617     \ifglshaslong{\glslabel}%
10618     {%
10619       \glsifplural
10620       {%

        Plural form:

10621         \glscapscase
10622         {%

        Plural form, don't adjust case:

10623         \glslongpluralaccessdisplay{\glsentrylongpl{\glslabel}}{\glslabel}%
10624         \glsinsert
10625       }%
    }%
  }%
}
```

```

10626      {%
    Plural form, make first letter upper case:
10627      \glslongpluralaccessdisplay{\Glsentrylongpl{\glslabel}}{\glslabel}%
10628      \glsinsert
10629      }%
10630      {%
    Plural form, all caps:
10631      \glslongpluralaccessdisplay
10632      {\mfirstucMakeUppercase{\glsentrylongpl{\glslabel}}}{\glslabel}%
10633      \mfirstucMakeUppercase{\glsinsert}%
10634      }%
10635      }%
10636      {%
    Singular form
10637      \glscapscase
10638      {%
    Singular form, don't adjust case:
10639      \glslongaccessdisplay{\glsentrylong{\glslabel}}{\glslabel}\glsinsert
10640      }%
10641      {%
    Subsequent singular form, make first letter upper case:
10642      \glslongaccessdisplay{\Glsentrylong{\glslabel}}{\glslabel}\glsinsert
10643      }%
10644      {%
    Subsequent singular form, all caps:
10645      \glslongaccessdisplay
10646      {\mfirstucMakeUppercase
10647      {\glsentrylong{\glslabel}\glsinsert}}{\glslabel}%
10648      \mfirstucMakeUppercase{\glsinsert}%
10649      }%
10650      }%
10651      }%
10652      {%
    Not an acronym:
10653      \glsgenentryfmt
10654      }%
10655      }%
10656      {\glscustomtext\glsinsert}%
10657      }%
10658      {%
10659      \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
10660      \renewcommand*{\acrfullfmt}[3]{%
10661      \glslink[##1]{##2}{%
10662      \glslongaccessdisplay{\glsentrylong{##2}}{##2}##3\space
10663      (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2})}%

```

```

10664 \renewcommand*{\Acrfullfmt}[3]{%
10665   \glslink[##1]{##2}{%
10666     \glslongaccessdisplay{\Glsentrylong{##2}}{##2}##3\space
10667     (\glsshortaccessdisplay{\acronymfont{\Glsentryshort{##2}}}{##2}}}%
10668 \renewcommand*{\ACRfullfmt}[3]{%
10669   \glslink[##1]{##2}{%
10670     \glslongaccessdisplay
10671     {\mfirstucMakeUppercase{\Glsentrylong{##2}}{##2}##3\space
10672     (\glsshortaccessdisplay{\acronymfont{\Glsentryshort{##2}}}{##2}})}}%
10673 \renewcommand*{\acrfullplfmt}[3]{%
10674   \glslink[##1]{##2}{%
10675     \glslongpluralaccessdisplay
10676     {\Glsentrylongpl{##2}}{##2}##3\space
10677     (\glsshortpluralaccessdisplay
10678     {\acronymfont{\Glsentryshortpl{##2}}}{##2}})}}%
10679 \renewcommand*{\Acrfullplfmt}[3]{%
10680   \glslink[##1]{##2}{%
10681     \glslongpluralaccessdisplay
10682     {\Glsentrylongpl{##2}}{##2}##3\space
10683     (\glsshortpluralaccessdisplay
10684     {\acronymfont{\Glsentryshortpl{##2}}}{##2}})}}%
10685 \renewcommand*{\ACRfullplfmt}[3]{%
10686   \glslink[##1]{##2}{%
10687     \glslongpluralaccessdisplay
10688     {\mfirstucMakeUppercase{\Glsentrylongpl{##2}}{##2}##3\space
10689     (\glsshortpluralaccessdisplay
10690     {\acronymfont{\Glsentryshortpl{##2}}}{##2}})}}%
10691 \renewcommand*{\glsentryfull}[1]{%
10692   \glslongaccessdisplay{\Glsentrylong{##1}}\space
10693   (\glsshortaccessdisplay{\acronymfont{\Glsentryshort{##1}}}{##1}})%
10694 }%
10695 \renewcommand*{\Glsentryfull}[1]{%
10696   \glslongaccessdisplay{\Glsentrylong{##1}}{##1}\space
10697   (\glsshortaccessdisplay{\acronymfont{\Glsentryshort{##1}}}{##1}})%
10698 }%
10699 \renewcommand*{\glsentryfullpl}[1]{%
10700   \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}\space
10701   (\glsshortpluralaccessdisplay{\acronymfont{\Glsentryshortpl{##1}}}{##1}})%
10702 }%
10703 \renewcommand*{\Glsentryfullpl}[1]{%
10704   \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}\space
10705   (\glsshortpluralaccessdisplay{\acronymfont{\Glsentryshortpl{##1}}}{##1}})%
10706 }%
10707 \renewcommand*{\acronymentry}[1]{%
10708   \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##1}}}{##1}}%
10709 \renewcommand*{\acronymsort}[2]{##1}%
10710 \renewcommand*{\acronymfont}[1]{##1}%
10711 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
10712 }

```

dua-desc <long> only acronym style with user-supplied description.

```
10713 \renewacronymstyle{dua-desc}%
10714 {%
10715   \GlsUseAcrEntryDispStyle{dua}%
10716 }%
10717 {%
10718   \GlsUseAcrStyleDefs{dua}%
10719   \renewcommand*{\GenericAcronymFields}{}%
10720   \renewcommand*{\acronymentry}[1]{%
10721     \glslongaccessdisplay{\acronymfont{\glsentrylong{##1}}}{##1}}%
10722   \renewcommand*{\acronymsort}[2]{##2}%
10723 }%
```

footnote <short>\footnote{<long>} acronym style.

```
10724 \renewacronymstyle{footnote}%
10725 {%
```

Check for long form in case this is a mixed glossary.

```
10726   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
10727 }%
10728 {%
10729   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
```

Need to ensure hyperlinks are switched off on first use:

```
10730   \glshyperfirstfalse
10731   \renewcommand*{\genacrfullformat}[2]{%
10732     \glsshortaccessdisplay
10733     {\protect\firstacronymfont{\glsentryshort{##1}}}{##1}##2%
10734     \protect\footnote{\glslongaccessdisplay{\glsentrylong{##1}}}{##1}}%
10735   }%
10736   \renewcommand*{\Genacrfullformat}[2]{%
10737     \glsshortaccessdisplay
10738     {\firstacronymfont{\Glsentryshort{##1}}}{##1}##2%
10739     \protect\footnote{\glslongaccessdisplay{\glsentrylong{##1}}}{##1}}%
10740   }%
10741   \renewcommand*{\genplacrfullformat}[2]{%
10742     \glsshortpluralaccessdisplay
10743     {\protect\firstacronymfont{\glsentryshortpl{##1}}}{##1}##2%
10744     \protect\footnote{\glslongpluralaccessdisplay{\glsentrylongpl{##1}}}{##1}}%
10745   }%
10746   \renewcommand*{\Genplacrfullformat}[2]{%
10747     \glsshortpluralaccessdisplay
10748     {\firstacronymfont{\Glsentryshortpl{##1}}}{##1}##2%
10749     \protect\footnote{\glslongpluralaccessdisplay{\glsentrylongpl{##1}}}{##1}}%
10750   }%
10751   \renewcommand*{\acronymentry}[1]{%
10752     \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}%
10753   \renewcommand*{\acronymsort}[2]{##1}%
10754   \renewcommand*{\acronymfont}[1]{##1}%
10755   \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
```


Don't use footnotes for \acrfull:

```

10756 \renewcommand*{\acrfullfmt}[3]{%
10757   \glslink[##1]{##2}{%
10758     \glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}##3\space
10759     (\glslongaccessdisplay{\glsentrylong{##2}}{##2}))}%
10760 \renewcommand*{\Acrfullfmt}[3]{%
10761   \glslink[##1]{##2}{%
10762     \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##2}}}{##2}##3\space
10763     (\glslongaccessdisplay{\glsentrylong{##2}}{##2}))}%
10764 \renewcommand*{\ACRfullfmt}[3]{%
10765   \glslink[##1]{##2}{%
10766     \glsshortaccessdisplay
10767       {\mfirstucMakeUppercase
10768         {\acronymfont{\glsentryshort{##2}}}{##2}##3\space
10769         (\glslongaccessdisplay{\glsentrylong{##2}}{##2}))}%
10770 \renewcommand*{\acrfullplfmt}[3]{%
10771   \glslink[##1]{##2}{%
10772     \glsshortpluralaccessdisplay
10773       {\acronymfont{\glsentryshortpl{##2}}}{##2}##3\space
10774       (\glslongpluralaccessdisplay{\glsentrylongpl{##2}}{##2}))}%
10775 \renewcommand*{\ACRfullplfmt}[3]{%
10776   \glslink[##1]{##2}{%
10777     \glsshortpluralaccessdisplay
10778       {\acronymfont{\Glsentryshortpl{##2}}}{##2}##3\space
10779       (\glslongpluralaccessdisplay{\glsentrylongpl{##2}}{##2}))}%
10780 \renewcommand*{\ACRfullplfmt}[3]{%
10781   \glslink[##1]{##2}{%
10782     \glsshortpluralaccessdisplay
10783       {\mfirstucMakeUppercase
10784         {\acronymfont{\glsentryshortpl{##2}}}{##2}##3\space
10785         (\glslongpluralaccessdisplay{\glsentrylongpl{##2}}{##2}))}%

```

Similarly for \glsentryfull etc:

```

10786 \renewcommand*{\glsentryfull}[1]{%
10787   \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}\space
10788   (\glslongaccessdisplay{\glsentrylong{##1}}{##1}))%
10789 \renewcommand*{\Glsentryfull}[1]{%
10790   \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##1}}}{##1}\space
10791   (\glslongaccessdisplay{\glsentrylong{##1}}{##1}))%
10792 \renewcommand*{\glsentryfullpl}[1]{%
10793   \glsshortpluralaccessdisplay
10794     {\acronymfont{\glsentryshortpl{##1}}}{##1}\space
10795     (\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}))%
10796 \renewcommand*{\Glsentryfullpl}[1]{%
10797   \glsshortpluralaccessdisplay
10798     {\acronymfont{\Glsentryshortpl{##1}}}{##1}\space
10799     (\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}))%
10800 }

```

footnote-sc \textsc{<short>}\footnote{<long>} acronym style.

```
10801 \renewacronymstyle{footnote-sc}%
10802 {%
10803   \GlsUseAcrEntryDispStyle{footnote}%
10804 }%
10805 {%
10806   \GlsUseAcrStyleDefs{footnote}%
10807   \renewcommand{\acronymentry}[1]{%
10808     \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}
10809   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
10810   \renewcommand*{\acrpluralsuffix}{\glstextup{\glspluralsuffix}}%
10811 }%
```

footnote-sm \textsmaller{<short>}\footnote{<long>} acronym style.

```
10812 \renewacronymstyle{footnote-sm}%
10813 {%
10814   \GlsUseAcrEntryDispStyle{footnote}%
10815 }%
10816 {%
10817   \GlsUseAcrStyleDefs{footnote}%
10818   \renewcommand{\acronymentry}[1]{%
10819     \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}
10820   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
10821   \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
10822 }%
```

footnote-desc <short>\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```
10823 \renewacronymstyle{footnote-desc}%
10824 {%
10825   \GlsUseAcrEntryDispStyle{footnote}%
10826 }%
10827 {%
10828   \GlsUseAcrStyleDefs{footnote}%
10829   \renewcommand*{\GenericAcronymFields}{}%
10830   \renewcommand*{\acronymsort}[2]{##2}%
10831   \renewcommand*{\acronymentry}[1]{%
10832     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10833     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10834 }
```

footnote-sc-desc \textsc{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```
10835 \renewacronymstyle{footnote-sc-desc}%
10836 {%
10837   \GlsUseAcrEntryDispStyle{footnote-sc}%
10838 }%
10839 }
```

```

10840 \GlsUseAcrStyleDefs{footnote-sc}%
10841 \renewcommand*{\GenericAcronymFields}{}%
10842 \renewcommand*{\acronymsort}[2]{##2}%
10843 \renewcommand*{\acronymentry}[1]{%
10844     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10845     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10846 }

```

footnote-sm-desc \textsmaller{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

10847 \renewacronymstyle{footnote-sm-desc}%
10848 {%
10849     \GlsUseAcrEntryDispStyle{footnote-sm}%
10850 }%
10851 {%
10852     \GlsUseAcrStyleDefs{footnote-sm}%
10853     \renewcommand*{\GenericAcronymFields}{}%
10854     \renewcommand*{\acronymsort}[2]{##2}%
10855     \renewcommand*{\acronymentry}[1]{%
10856         \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10857         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10858 }

```

Use \newacronymhook to modify the key list to set the access text to the long version by default.

```

10859 \renewcommand*{\newacronymhook}{%
10860     \edef\@gls@keylist{shortaccess=\the\glslongtok,%
10861         \the\glskeylisttok}%
10862     \expandafter\glskeylisttok\expandafter{\@gls@keylist}%
10863 }

```

DefaultNewAcronymDef Modify default style to use access text:

```

10864 \renewcommand*{\DefaultNewAcronymDef}{%
10865     \edef\@do@newglossaryentry{%
10866         \noexpand\newglossaryentry{\the\glslabeltok}%
10867         {%
10868             type=\acronymtype,%
10869             name={\the\glsshorttok},%
10870             description={\the\glslongtok},%
10871             descriptionaccess=\relax,%
10872             text={\the\glsshorttok},%
10873             access={\noexpand\@glo@textaccess},%
10874             sort={\the\glsshorttok},%
10875             short={\the\glsshorttok},%
10876             shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10877             shortaccess={\the\glslongtok},%
10878             long={\the\glslongtok},%
10879             longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10880             descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%

```

```

10881 first={\noexpand\glslongaccessdisplay
10882   {\the\glslongtok}{\the\glslabeltok}\space
10883   (\noexpand\glsshortaccessdisplay
10884     {\the\glsshorttok}{\the\glslabeltok})},%
10885 plural={\the\glsshorttok\acrpluralsuffix},%
10886 firstplural={\noexpand\glslongpluralaccessdisplay
10887   {\noexpand\@glo@longpl}{\the\glslabeltok}\space
10888   (\noexpand\glsshortpluralaccessdisplay
10889     {\noexpand\@glo@shortpl}{\the\glslabeltok})},%
10890 firstaccess=\relax,
10891 firstpluralaccess=\relax,
10892 textaccess={\noexpand\@glo@shortaccess},%
10893 \the\glskeylisttok
10894 }%
10895 }%
10896 \let\@org@gls@assign@firstpl\gls@assign@firstpl
10897 \let\@org@gls@assign@plural\gls@assign@plural
10898 \let\@org@gls@assign@descplural\gls@assign@descplural
10899 \def\gls@assign@firstpl##1##2{%
10900   \@@gls@expand@field{##1}{firstpl}{##2}%
10901 }%
10902 \def\gls@assign@plural##1##2{%
10903   \@@gls@expand@field{##1}{plural}{##2}%
10904 }%
10905 \def\gls@assign@descplural##1##2{%
10906   \@@gls@expand@field{##1}{descplural}{##2}%
10907 }%
10908 \do@newglossaryentry
10909 \let\gls@assign@firstpl\@org@gls@assign@firstpl
10910 \let\gls@assign@plural\@org@gls@assign@plural
10911 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
10912 }

```

otnoteNewAcronymDef

```

10913 \renewcommand*{\DescriptionFootnoteNewAcronymDef}{%
10914   \edef\@do@newglossaryentry{%
10915     \noexpand\newglossaryentry{\the\glslabeltok}%
10916     {%
10917       type=\acronymtype,%
10918       name={\noexpand\acronymfont{\the\glsshorttok}},%
10919       sort={\the\glsshorttok},%
10920       text={\the\glsshorttok},%
10921       short={\the\glsshorttok},%
10922       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10923       shortaccess={\the\glslongtok},%
10924       long={\the\glslongtok},%
10925       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10926       access={\noexpand\@glo@textaccess},%
10927       plural={\the\glsshorttok\noexpand\acrpluralsuffix},%

```

```

10928     symbol={\the\glslongtok},%
10929     symbolplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10930     firstpluralaccess=\relax,
10931     textaccess={\noexpand\@glo@shortaccess},%
10932     \the\glskeylisttok
10933 }%
10934 }%
10935 \let\@org@gls@assign@firstpl\gls@assign@firstpl
10936 \let\@org@gls@assign@plural\gls@assign@plural
10937 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
10938 \def\gls@assign@firstpl##1##2{%
10939   \@@gls@expand@field{##1}{firstpl}{##2}%
10940 }%
10941 \def\gls@assign@plural##1##2{%
10942   \@@gls@expand@field{##1}{plural}{##2}%
10943 }%
10944 \def\gls@assign@symbolplural##1##2{%
10945   \@@gls@expand@field{##1}{symbolplural}{##2}%
10946 }%
10947 \do@newglossaryentry
10948 \let\gls@assign@plural\@org@gls@assign@plural
10949 \let\gls@assign@firstpl\@org@gls@assign@firstpl
10950 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
10951 }

```

ptionNewAcronymDef

```

10952 \renewcommand*{\DescriptionNewAcronymDef}{%
10953   \edef\@do@newglossaryentry{%
10954     \noexpand\newglossaryentry{\the\glslabeltok}%
10955     {%
10956       type=\acronymtype,%
10957       name={\noexpand
10958         \acrnameformat{\the\glsshorttok}{\the\glslongtok}},%
10959       access={\noexpand\@glo@textaccess},%
10960       sort={\the\glsshorttok},%
10961       short={\the\glsshorttok},%
10962       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10963       shortaccess={\the\glslongtok},%
10964       long={\the\glslongtok},%
10965       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10966       first={\the\glslongtok},%
10967       firstaccess=\relax,
10968       firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10969       text={\the\glsshorttok},%
10970       textaccess={\the\glslongtok},%
10971       plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10972       symbol={\noexpand\@glo@text},%
10973       symbolaccess={\noexpand\@glo@textaccess},%
10974       symbolplural={\noexpand\@glo@plural},%

```

```

10975     firstpluralaccess=\relax,
10976     textaccess={\noexpand\@glo@shortaccess},%
10977     \the\glskeylisttok}%
10978 }%
10979 \let\@org@gls@assign@firstpl\gls@assign@firstpl
10980 \let\@org@gls@assign@plural\gls@assign@plural
10981 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
10982 \def\gls@assign@firstpl##1##2{%
10983     \@@gls@expand@field{##1}{firstpl}{##2}%
10984 }%
10985 \def\gls@assign@plural##1##2{%
10986     \@@gls@expand@field{##1}{plural}{##2}%
10987 }%
10988 \def\gls@assign@symbolplural##1##2{%
10989     \@@gls@expand@field{##1}{symbolplural}{##2}%
10990 }%
10991 \do@newglossaryentry
10992 \let\gls@assign@firstpl\@org@gls@assign@firstpl
10993 \let\gls@assign@plural\@org@gls@assign@plural
10994 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
10995 }

```

otnoteNewAcronymDef

```

10996 \renewcommand*{\FootnoteNewAcronymDef}{%
10997     \edef\@do@newglossaryentry{%
10998         \noexpand\newglossaryentry{\the\glslabeltok}%
10999         {%
11000             type=\acronymtype,%
11001             name={\noexpand\acronymfont{\the\glsshorttok}},%
11002             sort={\the\glsshorttok},%
11003             text={\the\glsshorttok},%
11004             textaccess={\the\glslongtok},%
11005             access={\noexpand\@glo@textaccess},%
11006             plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11007             short={\the\glsshorttok},%
11008             shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11009             long={\the\glslongtok},%
11010             longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11011             description={\the\glslongtok},%
11012             descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11013             \the\glskeylisttok
11014         }%
11015     }%
11016     \let\@org@gls@assign@plural\gls@assign@plural
11017     \let\@org@gls@assign@firstpl\gls@assign@firstpl
11018     \let\@org@gls@assign@descplural\gls@assign@descplural
11019     \def\gls@assign@firstpl##1##2{%
11020         \@@gls@expand@field{##1}{firstpl}{##2}%
11021     }%

```

```

11022 \def\gls@assign@plural##1##2{%
11023   \@@gls@expand@field{##1}{plural}{##2}%
11024 }%
11025 \def\gls@assign@descplural##1##2{%
11026   \@@gls@expand@field{##1}{descplural}{##2}%
11027 }%
11028 \@do@newglossaryentry
11029 \let\gls@assign@plural\@org@gls@assign@plural
11030 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11031 \let\gls@assign@descplural\@org@gls@assign@descplural
11032 }

```

\SmallNewAcronymDef

```

11033 \renewcommand*{\SmallNewAcronymDef}{%
11034   \edef\@do@newglossaryentry{%
11035     \noexpand\newglossaryentry{\the\glslabeltok}%
11036     {%
11037       type=\acronymtype,%
11038       name={\noexpand\acronymfont{\the\glsshorttok}},%
11039       access={\noexpand\@glo@symbolaccess},%
11040       sort={\the\glsshorttok},%
11041       short={\the\glsshorttok},%
11042       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11043       shortaccess={\the\glslongtok},%
11044       long={\the\glslongtok},%
11045       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11046       text={\noexpand\@glo@short},%
11047       textaccess={\noexpand\@glo@shortaccess},%
11048       plural={\noexpand\@glo@shortpl},%
11049       first={\the\glslongtok},%
11050       firstaccess=\relax,%
11051       firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11052       description={\noexpand\@glo@first},%
11053       descriptionplural={\noexpand\@glo@firstplural},%
11054       symbol={\the\glsshorttok},%
11055       symbolaccess={\the\glslongtok},%
11056       symbolplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11057       \the\glskeylisttok
11058     }%
11059   }%
11060   \let\@org@gls@assign@firstpl\gls@assign@firstpl
11061   \let\@org@gls@assign@plural\gls@assign@plural
11062   \let\@org@gls@assign@descplural\gls@assign@descplural
11063   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
11064   \def\gls@assign@firstpl##1##2{%
11065     \@@gls@expand@field{##1}{firstpl}{##2}%
11066   }%
11067   \def\gls@assign@plural##1##2{%
11068     \@@gls@expand@field{##1}{plural}{##2}%

```

```

11069 }%
11070 \def\gls@assign@descplural##1##2{%
11071   \@@gls@expand@field{##1}{descplural}{##2}%
11072 }%
11073 \def\gls@assign@symbolplural##1##2{%
11074   \@@gls@expand@field{##1}{symbolplural}{##2}%
11075 }%
11076 \do@newglossaryentry
11077 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11078 \let\gls@assign@plural\@org@gls@assign@plural
11079 \let\gls@assign@descplural\@org@gls@assign@descplural
11080 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11081 }

```

The following are kept for compatibility with versions before 3.0:

\glsshortaccesskey

```

11082 \newcommand*{\glsshortaccesskey}{\glsshortkey access}%

```

\glsshortpluralaccesskey

```

11083 \newcommand*{\glsshortpluralaccesskey}{\glsshortpluralkey access}%

```

\glslongaccesskey

```

11084 \newcommand*{\glslongaccesskey}{\glslongkey access}%

```

\glslongpluralaccesskey

```

11085 \newcommand*{\glslongpluralaccesskey}{\glslongpluralkey access}%

```

7.5 Debugging Commands

\showglongnameaccess

```

11086 \newcommand*{\showglongnameaccess}[1]{%
11087   \expandafter\show\csname glo@\glsdetoklabel{#1}@textaccess\endcsname
11088 }

```

\showglotextaccess

```

11089 \newcommand*{\showglotextaccess}[1]{%
11090   \expandafter\show\csname glo@\glsdetoklabel{#1}@textaccess\endcsname
11091 }

```

\showglopluralaccess

```

11092 \newcommand*{\showglopluralaccess}[1]{%
11093   \expandafter\show\csname glo@\glsdetoklabel{#1}@pluralaccess\endcsname
11094 }

```

\showglofirstaccess

```

11095 \newcommand*{\showglofirstaccess}[1]{%
11096   \expandafter\show\csname glo@\glsdetoklabel{#1}@firstaccess\endcsname
11097 }

```


lofirstpluralaccess

```
11098 \newcommand*{\showglofirstpluralaccess}[1]{%
11099   \expandafter\show\csname glo@\glsdetoklabel{#1}@firstpluralaccess\endcsname
11100 }
```

showglosymbolaccess

```
11101 \newcommand*{\showglosymbolaccess}[1]{%
11102   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolaccess\endcsname
11103 }
```

osymbolpluralaccess

```
11104 \newcommand*{\showglosymbolpluralaccess}[1]{%
11105   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolpluralaccess\endcsname
11106 }
```

\showglodescaccess

```
11107 \newcommand*{\showglodescaccess}[1]{%
11108   \expandafter\show\csname glo@\glsdetoklabel{#1}@descaccess\endcsname
11109 }
```

glodescpluralaccess

```
11110 \newcommand*{\showglodescpluralaccess}[1]{%
11111   \expandafter\show\csname glo@\glsdetoklabel{#1}@descpluralaccess\endcsname
11112 }
```

\showgloshortaccess

```
11113 \newcommand*{\showgloshortaccess}[1]{%
11114   \expandafter\show\csname glo@\glsdetoklabel{#1}@shortaccess\endcsname
11115 }
```

loshortpluralaccess

```
11116 \newcommand*{\showgloshortpluralaccess}[1]{%
11117   \expandafter\show\csname glo@\glsdetoklabel{#1}@shortpluralaccess\endcsname
11118 }
```

\showglolongaccess

```
11119 \newcommand*{\showglolongaccess}[1]{%
11120   \expandafter\show\csname glo@\glsdetoklabel{#1}@longaccess\endcsname
11121 }
```

glolongpluralaccess

```
11122 \newcommand*{\showglolongpluralaccess}[1]{%
11123   \expandafter\show\csname glo@\glsdetoklabel{#1}@longpluralaccess\endcsname
11124 }
```

8 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.

```
11125 \NeedsTeXFormat{LaTeX2e}
11126 \ProvidesPackage{glossaries-babel}[2015/07/08 v4.16 (NLCT)]
```

Load tracklang to obtain language settings.

```
11127 \RequirePackage{tracklang}
11128 \let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
11129 \AnyTrackedLanguages
11130 {%
11131   \ForEachTrackedDialect{\this@dialect}{%
11132     \IfTrackedLanguageFileExists{\this@dialect}%
11133     {glossaries-}% prefix
11134     {.ldf}%
11135     {%
11136       \RequireGlossariesLang{\CurrentTrackedTag}%
11137     }%
11138     {%
11139       \PackageWarningNoLine{glossaries}%
11140       {No language module detected for ‘\this@dialect’.\MessageBreak
11141       Language modules need to be installed separately.\MessageBreak
11142       Please check on CTAN for a bundle called\MessageBreak
11143       ‘glossaries-\CurrentTrackedLanguage’ or similar}%
11144     }%
11145   }%
11146 }%
11147 {}%
```

8.1 Polyglossia Captions

Language support has now been split off into independent language modules.

```
11148 \NeedsTeXFormat{LaTeX2e}
11149 \ProvidesPackage{glossaries-polyglossia}[2015/07/08 v4.16 (NLCT)]
```

Load tracklang to obtain language settings.

```
11150 \RequirePackage{tracklang}
11151 \let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
11152 \AnyTrackedLanguages
11153 {%
11154   \ForEachTrackedDialect{\this@dialect}{%
11155     \IfTrackedLanguageFileExists{\this@dialect}%
11156     {glossaries-}% prefix
11157     {.ldf}%
```

```

11158      {%
11159      \RequireGlossariesLang{\CurrentTrackedTag}%
11160      }%
11161      {%
11162      \PackageWarningNoLine{glossaries}%
11163      {No language module detected for ‘\this@dialect’.\MessageBreak
11164      Language modules need to be installed separately.\MessageBreak
11165      Please check on CTAN for a bundle called\MessageBreak
11166      ‘glossaries-\CurrentTrackedLanguage’ or similar}%
11167      }%
11168      }%
11169      }%
11170      {}%

```

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

??		<code>\glsmakefirstuc</code> : new	259
	super: fixed typo in <code>\subglossentry</code>	1.06	
	(<code>\glossentrydesc</code>)	285	General: now requires <code>etoolbox</code> .
1.01	General: Added range facility in		<code>\capitalisewords</code> : new
	format key	109	<code>\xcapitalisewords</code> : new
	<code>\writeist</code> : Added spaces after		260
	<code>\delimN</code> and <code>\delimR</code> in ist	1.07	
	file	156	<code>\@gls@link</code> : fixed bug caused by
1.03			<code>\theglssentrycounter</code> set-
	<code>\makefirstuc</code> : changed ‘pro-		ting the page number too soon
	ected@edef to ‘def	258
1.04	General: Added <code>\glstextformat</code>	93	<code>\glsadd</code> : fixed bug caused by
1.05			<code>\theglssentrycounter</code> set-
	<code>\glossarysection</code> : added	1.08	ting the page number too soon
	<code>\@mkboth</code> to <code>\glossarysection</code>	
	37	General: Added babel support . . .
	<code>\gls@defglossaryentry</code> :		<code>\capitalisewords</code> : made robust
	Changed the default value of	
	the sort key to just the value of		<code>listgroup</code> : changed <code>listgroup</code>
	the name key	76	style to use <code>\glsgetgrouptitle</code>
		

altlistgroup: changed altlistgroup style to use \glsgetgrouptitle 267 \makefirstuc: made robust ... 257	\hyperlink (memoir defines \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) 76 descriptionplural support added 76 symbolplural support added .. 76 \Glsentrydescplural: New .. 147 \glsentrydescplural: New .. 147 \Glsentrysymbolplural: New 148 \glsentrysymbolplural: New 148 \SetDescriptionFootnoteAcronymStyle: Added \protect before \footnote and \glslink . 231 \SetFootnoteAcronymStyle: Added \protect before \footnote and \glslink . 237 symbolplural: new 61
1.09 \@mfu@nocaplist: new 260 \capitalisewords: added check for words that shouldn't be capitalised 259 \gMFUnocap: new 260 \mfu@checkword: new 260 \MFUclear: new 260	
1.1 \@glossarysection: numbered sections and auto label added 39 \@gls@tmpb: changed \toksdef to \newtoks 111 \@gls@toc: numberline added .. 40 \@p@glossarysection: numbered sections and auto label added 39 General: amsgen now loaded (\new@ifnextchar needed) .. 4 translate: translate option added 22 \setglossarysection: new ... 38 numberedsection: numbered-section package option added . 6 numberline: numberline option added 6	1.13 General: fixed bug that ignored 3rd parameter 124-132 \ACRfullpl: new 212 \Acrfullpl: new 211 \acrfullpl: new 211 \acrpluralsuffix: New 209 \gls@defglossaryentry: Changed default first value .. 76 Changed default firstplural value 76 Removed restriction on only using \newglossaryentry in the preamble 81 \newacronym: Removed restriction on only using \newacronym in the preamble 209
1.10 \ecapitalisewords: new 260 \emakefirstuc: new 259	
1.12 \@GLSpl: now uses \glsentrydescplural and \glsentrysymbolplural instead of \glsentrydesc and \glsentrysymbol 123 \@GLspl@: now uses \glsentrydescplural and \glsentrysymbolplural instead of \glsentrydesc and \glsentrysymbol 122 \@GLspl@: now uses \glsentrydescplural and \glsentrysymbolplural instead of \glsentrydesc and \glsentrysymbol 121 General: added check for \hypertarget separate to	\@gls@hypergroup: new 262 General: added nonumberlist key to \printglossary 195 added numberedsection key to \printglossary 193 \firstacronymfont: new 212 \glsautoprefix: new 6 \glsnavhyperlink: changed 'edef to 'protected@edef ... 261

\glsnavhypertarget: added write to aux file	261	scroll off the top of the page	117
\glsnavigation: changed to only use labels for groups that are present	262	\gls@defglossaryentry: Changed def to let	76
1.15		1.17	
\@gls@link: added \glslabel	106	\@@do@wrglossary: new	174
\gls@defglossaryentry: check for \@glo@first in descrip- tion	80	\@do@seeglossary: new	177
check for \@glo@text in sym- bol	80	\@glo@storeentry: new	82
\gls@hypergroup: new	262	\@gls@glossary: changed defi- nition to use \index instead of \@index	172
\glsnavhypertarget: added check if rerun required	261	\@glsdefaultplural: new	63
\glssettoctitle: new	30	\@glsdefaultsort: new	64
\printglossary: changed the way the TOC title is set	179	\@gls@hypernumber: new	205
1.16		\@gls@noname: new	63
\@GLS@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	120	\@gls@nonextpages: new	195
\@GLSpl: Test glossary type is \acronymtype in addition to checking if footnote option has been used	123	General: added xindy support ...	25
\@Gls@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	119	parent: new	62
\@Glspl@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	122	see: new	61
\@gls@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	118	\gls@defglossaryentry: added nonumberlist key	77
\@glsdisp: Test glossary type is \acronymtype in addition to checking if footnote option has been used	123	added parent key	77
\@glspl@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	121	added see key	76
\@glstarget: raised the hyper- target so the target text doesn't		Stored main part of entry format when entry is defined	81
		\gls@suffixF: new	35
		\gls@suffixFF: new	36
		\gls@wrglossary: modified to allow for xindy support	172
		\gls@hyperlink: new	153
		\gls@hypernumber: modified to allow material to be attached to location	205
		\glsnavhyperlink: replaced 'hy- perlink to '@glslink	261
		\glsnavhypertarget: replaced 'hypertarget to '@glstarget .	261
		\glssee: new	177
		\glsseeformat: new	177
		\glsSetSuffixF: new	35
		\glsSetSuffixFF: new	36
		\ifglsxindy: new	25
		\istfilename: added xindy sup- port	34
		\newglossarystyle: made \newglossarystyle long .	204
		\nopostdesc: new	33
		nonumberlist: new	62

<code>\printglossary</code> : added check to determine if <code>\printglossary</code> is already defined 179	<code>\forall glossaries</code> : replaced <code>\ifthenelse</code> with <code>\ifx</code> 49
added print language to aux file 179	<code>\forall glossentries</code> : replaced <code>\ifthenelse</code> with <code>\ifx</code> 50
order: order package option added 25	<code>\glsdefmain</code> : new 12
<code>\writeist</code> : added xindy support 156	<code>\glsdescwidth</code> : changed <code>\linewidth</code> to <code>\hsize</code> . 269, 284
1.18	<code>\glslistdottedwidth</code> : changed <code>\linewidth</code> to <code>\hsize</code> 268
<code>\@gls@loadlist</code> : new 8	<code>\glspagelistwidth</code> : changed <code>\linewidth</code> to <code>\hsize</code> . 269, 285
<code>\@gls@loadlong</code> : new 8	<code>nomain</code> : added <code>nomain</code> package option 13
<code>\@gls@loadsuper</code> : new 8	<code>\writeist</code> : removed <code>item_02</code> - no such <code>makeindex</code> key 160
<code>\@gls@loadtree</code> : new 8	2.02
<code>\gls@defglossaryentry</code> : Changed default value of sort to <code>\glsdefaultsort</code> 76	<code>\@printglossary</code> : suppressed warning globally rather than locally 181
moved sort sanitization to <code>\newglossaryentry</code> 80	<code>\glossarysection</code> : changed <code>\@mkboth</code> to <code>\glossarymark</code> 37
<code>\glstarget</code> : new 198	<code>\gls glossarymark</code> : New 38
<code>\oldacronym</code> : new 208	2.03
<code>nolist</code> : new 8	<code>\@GLS@</code> : Added check for hyper-first 120
<code>notlong</code> : new 8	<code>\@GLSpl</code> : Added check for hyper-first 123
sort: moved sanitization to <code>\newglossaryentry</code> 60	<code>\@Gls@</code> : Added check for hyper-first 119
<code>nostyles</code> : new 8	<code>\@Glspl@</code> : Added check for hyper-first 122
<code>nosuper</code> : new 8	<code>\@gls@</code> : Added check for hyper-first 118
<code>notree</code> : new 8	<code>\@gls@@link</code> : new 105
1.19	<code>\@gls@link</code> : added <code>\leavevmode</code> 106
<code>\gls clearpage</code> : new 40	Moved entry existence check to avoid duplicate code 106
<code>\glsdisp</code> : new 123	<code>\@glsdisp</code> : Added check for hyper-first 123
<code>\SetDescriptionAcronymStyle</code> : changed <code>\acronymfont</code> to use <code>\textsmaller</code> instead of <code>\smaller</code> 235	<code>\@glspl@</code> : Added check for hyper-first 121
<code>\SetDescriptionFootnoteAcronymStyle</code> : changed <code>\acronymfont</code> to use <code>\textsmaller</code> instead of <code>\smaller</code> 231	<code>\gls glossarymark</code> : Added check to see if it's already defined .. 38
<code>\SetFootnoteAcronymStyle</code> : changed <code>\acronymfont</code> to use <code>\textsmaller</code> instead of <code>\smaller</code> 237	<code>hyperfirst</code> : new 23
<code>\SetSmallAcronymStyle</code> : changed <code>\acronymfont</code> to use <code>\textsmaller</code> instead of <code>\smaller</code> 240	2.04
2.01	<code>\@GLS@</code> : Changed test to check if glossary type has been identified as a list of acronyms ... 120
<code>\@gls@link</code> : moved <code>\@do@wrglossary</code> before term is displayed to prevent unwanted <code>whatsit</code> 107	

\@GLSp1: Changed test to check if glossary type has been identi- fied as a list of acronyms ...	123	\gls@<type>@displayfirst have been defined.	57
\@GLs@: Changed test to check if glossary type has been identi- fied as a list of acronyms ...	119	\SetAcronymLists:new	15
\@GLspl@: Changed test to check if glossary type has been identi- fied as a list of acronyms ..	122	\SetDefaultAcronymDisplayStyle: new	227
\@glossaryentryfield:new ..	82	\SetDefaultAcronymStyle: new	228
\@glossarysubentryfield: new	82	\SetDescriptionAcronymDisplayStyle: new	233
\@gls@: Changed test to check if glossary type has been identi- fied as a list of acronyms ...	118	\SetDescriptionDUAAcronymDisplayStyle: new	231
\@glsacronymlists:new	14	\SetDescriptionFootnoteAcronymDisplayStyle: new	229
\@glsdisp: Changed test to check if glossary type has been identi- fied as a list of acronyms ..	123	\SetDUADisplayStyle:new ..	241
\@glspl@: Changed test to check if glossary type has been identi- fied as a list of acronyms ..	121	\SetFootnoteAcronymDisplayStyle: new	235
\@newglossaryentryposthook: new	81	\SetSmallAcronymDisplayStyle: new	238
\@newglossaryentryprehook: new	81		
acronymlists:new	15		
\DeclareAcronymList:new ...	14		
\DefineAcronymSynonyms:new	225		
\gls@defglossaryentry: added user1-6 keys	77		
\glsadd: fixed bug that ignored counter	154		
\Glsentryuseri:new	149		
\glsentryuseri:new	149		
\Glsentryuserii:new	150		
\glsentryuserii:new	150		
\Glsentryuseriii:new	150		
\glsentryuseriii:new	150		
\Glsentryuseriv:new	150		
\glsentryuseriv:new	150		
\Glsentryuserv:new	150		
\glsentryuserv:new	150		
\Glsentryuservi:new	151		
\glsentryuservi:new	150		
\ns@newglossary: added check to determine if \gls@<type>@display and			
		\@glsdisp: Added closing brace. Patch provided by Sergiu Dotenco	123
		Removed spurious brace. Patch provided by Sergiu Dotenco	124
		\writeist: Added \string be- fore opening and closing braces. Patch provided by Sergiu Dotenco	161
	2.05		
		\altnewglossary:new	57
		\CustomAcronymFields:new ..	243
		\CustomNewAcronymDef:new ..	243
		\SetCustomDisplayStyle:new	243
		\SetCustomStyle:new	244
	2.06		
		General: glssadd format key stored in \@glsnumberformat (was mistakenly stored in \@glo@format)	153
	2.07		
		\@@do@wrglossary: added check for hyper location prefix ...	175
		modified to use new format ..	174
		\@@glossarysec: replaced \@ifundefined with \ifcsundef	6
	3.0		
		\@do@seeglossary: Sanitize and escape cross-referencing in- formation	177

\@gls@counterwithin:new ... 10	\glossarysection: replaced
\@gls@ifinlist:new 41	\@ifundefined with
\@gls@link:added \@gls@saveentrycounter\ifcsundef 37	\glossarystyle: replaced
..... 107	\@ifundefined with
added \@gls@setsort 107	\ifcsundef 204
\@gls@saveentrycounter:new 108	\gls@codepage: replaced
\@gls@setupsort@def:new ... 11	\@ifundefined with
\@gls@setupsort@standard:	\ifcsundef 25
new 10	\gls@defglossaryentry: added
\@gls@setupsort@use:new ... 11	\@gls@defsort 80
\@gls@xdy@locationlist:new 44	added short and long keys 77
\@glslink:replaced \@ifundefined	replaced \@ifundefined with
with \ifcsundef 117	\ifcsundef 77
\@glsnextpages:new 195	\gls@docclearpage: replaced
\@makeglossary: Added check	\@ifundefined with
for savewrites 162	\ifcsundef 39
\@print@glossary: replaced	\gls@wrglossary: modified to
\@ifundefined with	take into account savewrites 172
\ifcsundef 182	\glsadd:added \@gls@saveentrycounter
\@printglossary: added 154
\currentglossary 181	\GlsAddXdyCounters:new 41
added \glsnextpages 181	\glentrycounterlabel:new 197
make toctitle default to title .. 181	\glentryitem:new 198
\@set@glo@numformat: added	\Glsentrylong:new 151
4th argument 109	\glentrylong:new 151
\@xdy@attributelist:new 41	\Glsentrylongpl:new 151
General: added prefix to hyperlink	\glentrylongpl:new 151
..... 206	\Glsentryshort:new 151
etoolbox now loaded 5	\glentryshort:new 151
replaced \@ifundefined with	\Glsentryshortpl:new 151
\ifcsundef ... 29,32,103,193	\glentryshortpl:new 151
\acrfootnote:new 229	\glsgetgrouptitle: re-
\ACRfull: added starred version 210	placed \@ifundefined with
\Acrfull: added starred version 210	\ifcsundef 202
\acrfull: added starred version 209	\gls@glossarymark: replaced
\ACRfullpl: added starred ver-	\@ifundefined with
sion 212	\ifcsundef 38
\Acrfullpl: added starred ver-	\glshyperlink: changed de-
sion 211	fault from \glentryname to
\acrfullpl: added starred ver-	\glentrytext 153
sion 211	\glshypernumber: replaced
\acrlinkfootnote:new 229	\@ifundefined with
\acrno linkfootnote:new ... 229	\ifcsundef 205
savewrites:new 26	\glsnumberformat: replaced
see:added \@glo@seeautonumberlist	\@ifundefined with
..... 61	\ifcsundef 36
seeautonumberlist:new 8	\glsrefentry:new 197

<code>\glsresetsubentrycounter:</code>		<code>\ifcsundef</code>	179
new	196	<code>\SetDescriptionFootnoteAcronymDisplayStyle:</code>	
<code>\glsseeitem:</code> hyperlink uses		expanded options link op-	
<code>\glsseeitemformat</code> instead		tions	229
of <code>\glsentryname</code>	178	<code>\setentrycounter:</code> added op-	
<code>\glsseeitemformat:new</code>	178	tional argument	203
<code>\glssortnumberfmt:new</code>	11	<code>\showacronymlists:new</code>	249
<code>\glsstepentry:new</code>	197	<code>\showglocounter:new</code>	246
<code>\glsstepsubentry:new</code>	197	<code>\showglodesc:new</code>	247
<code>\glssubentrycounterlabel:</code>		<code>\showglodescplural:new</code> ...	248
new	198	<code>\showglofirst:new</code>	246
<code>\glssubentryitem:new</code>	198	<code>\showglofirstpl:new</code>	246
<code>theglossary:replaced\@ifundefined</code>		<code>\showgloflag:new</code>	249
with <code>\ifcsundef</code>	198	<code>\showgloindex:new</code>	249
<code>short:new</code>	63	<code>\showglolevel:new</code>	245
<code>shortplural:new</code>	63	<code>\showgloname:new</code>	247
<code>\ifglossaryexists:</code> re-		<code>\showgloparent:new</code>	245
placed <code>\@ifundefined</code> with		<code>\showgloplural:new</code>	245
<code>\ifcsundef</code>	50	<code>\showglosort:new</code>	248
<code>\ifglsentryexists:</code> re-		<code>\showglossaries:new</code>	249
placed <code>\@ifundefined</code> with		<code>\showglossarycounter:new</code> .	250
<code>\ifcsundef</code>	51	<code>\showglossaryentries:new</code> .	250
<code>\istfile:deprecated</code>	170	<code>\showglossaryin:new</code>	250
<code>glossaryentry:new</code>	196	<code>\showglossaryout:new</code>	250
<code>glossarysubentry:new</code>	196	<code>\showglossarytitle:new</code> ...	250
<code>\newglossaryentry:</code> replaced		<code>\showglosymbol:new</code>	248
<code>\DeclareRobustCommand</code>		<code>\showglosymbolplural:new</code> .	248
with <code>\newrobustcmd</code>	66	<code>\showglotext:new</code>	245
<code>\newglossarystyle:</code> re-		<code>\showglotype:new</code>	246
placed <code>\@ifundefined</code> with		<code>\showglouseri:new</code>	246
<code>\ifcsundef</code>	204	<code>\showglouserii:new</code>	246
<code>\ns@newglossary:</code> added		<code>\showglouseriii:new</code>	247
<code>\@gls@defsortcount</code>	57	<code>\showglouseriv:new</code>	247
replaced <code>\@ifundefined</code> with		<code>\showglouserv:new</code>	247
<code>\ifcsundef</code>	57	<code>\showglouservi:new</code>	247
<code>entrycounter:new</code>	9	<code>subentrycounter:new</code>	10
<code>entrycounterwithin:new</code>	9	<code>\writeist:</code> added xindy-only	
<code>\oldacronym:replaced\@ifundefined</code>		macro definitions to glossary	
with <code>\ifcsundef</code>	208	open tag	158
<code>compatible-2.07:</code> compatible-		modified to support new for-	
2.07 option added	27	mat	156
<code>long:new</code>	63		
<code>longplural:new</code>	63	3.01	
<code>nonumberlist:</code> now boolean ...	62	<code>\@glswritefiles:</code> added check	
<code>sort:new</code>	10	for empty glossaries	170
<code>counter:replaced\@ifundefined</code>		General: made robust	119
with <code>\ifcsundef</code>	61	<code>\ACRfull:</code> made robust	210
<code>\printglossary:</code> replaced		<code>\Acrfull:</code> made robust	210
<code>\@ifundefined</code> with		<code>\acrfull:</code> made robust	209

\acrfullformat:	removed	
\acronymfont as it should already be set in the second argument.	210
\ACRfullpl: made robust	212
\Acrfullpl: made robust	211
\acrfullpl: made robust	211
\ACRlong: made robust	142
\Acrlong: made robust	142
\acrlong: made robust	141
\ACRlongpl: made robust	144
\Acrlongpl: made robust	144
\acrlongpl: made robust	143
\ACRshort: made robust	139
\Acrshort: made robust	138
\acrshort: made robust	137
\ACRshortpl: made robust	140
\Acrshortpl: made robust	140
\acrshortpl: made robust	139
\Gls: made robust	119
\glsadd: made robust	154
\glsaddall: made robust	154
\GLSdesc: made robust	129
\Glsdesc: made robust	129
\glsdesc: made robust	128
\GLSdescplural: made robust	130
\Glsdescplural: made robust	130
\glsdescplural: made robust	129
\glsfirst: made robust	125
\GLSfirstplural: made robust	127
\Glsfirstplural: made robust	127
\glsfirstplural: made robust	127
\glslink: made robust	105
\GLSname: made robust	128
\Glsname: made robust	128
\glsname: made robust	128
\GLSpl: made robust	122
\Glspl: made robust	121
\glspl: made robust	120
\GLSplplural: made robust	126
\GLSsymbol: made robust	131
\Glsymbol: made robust	131
\glsymbol: made robust	130
\GLSsymbolplural: made robust	132
\Glsymbolplural: made robust	131
\glsymbolplural: made robust	131
\Glstext: made robust	125
\glstext: made robust	124
\GLSuseri: made robust	133
\Glsuseri: made robust	132
\glsuseri: made robust	132
\GLSuserii: made robust	134
\Glsuserii: made robust	133
\glsuserii: made robust	133
\GLSuseriii: made robust	134
\Glsuseriii: made robust	134
\glsuseriii: made robust	134
\GLSuseriv: made robust	135
\Glsuseriv: made robust	135
\glsuseriv: made robust	135
\GLSuserv: made robust	136
\Glsuserv: made robust	136
\glsuserv: made robust	136
\GLSuservi: made robust	137
\Glsuservi: made robust	137
\glsuservi: made robust	137
3.02		
\@@do@wrglossary:	changed	
\@glslocref to \theglsentrycounter	176
\@do@wrglossary:	changed	
\@do@wr@glossary to test for indexonlyfirst option; put old \@do@wr@glossary code into @@do@wrglossary	173
\@gls@missingnumberlist:		
new	64
\@glswritefiles:	added check for existence of token in case \makeglossaries has been omitted 170
\@printglossary:	add a way to fetch current entry label	... 181
savenumberlist:	new 7
ucmark:	new 9
\gls@defglossaryentry:	added numberlist element 80
\gls@save@numberlist:	new	... 179
\gls@wrglossary:	added check for glossary file defined 172
\glsdisplaynumberlist:	new	152
\glsentrycounter:	set default value 107

\Glsentryfull: fixed bug (replaced \glsentryshortpl with \glsentryshort)	152	long: added check for glsnogroupskip	270
\glsentryfullpl: fixed bug (replaced \glsentryshort with \glsentryshortpl)	152	long3col: added check for glsnogroupskip	271
\glsentrynumberlist: new ..	152	long4col: added check for glsnogroupskip	273
\glsmoveentry: new	82	longragged: added check for glsnogroupskip	276
\glsnumlistlastsep: new ...	153	longragged3col: added check for glsnogroupskip	277
\glsnumlistsep: new	153	nopostdot: new	9
\glsresetsubentrycounter: new	197	tree: added check for glsnogroupskip	300
\ifglshaschildren: new	52	treenoname: added check for glsnogroupskip	301
\ifglshasparent: new	52	super: added check for glsnogroupskip	285
\makeglossaries: added list parser	165	super3col: added check for glsnogroupskip	287
indexonlyfirst: new	23	super4col: added check for glsnogroupskip	289
\renewglossarystyle: new ..	204	superragged: added check for glsnogroupskip	292
\showglossaryentries: fixed misspelt command	250	superragged3col: added check for glsnogroupskip	294
\SmallNewAcronymDef: fixed broken short and long plural	239		
3.03		3.04	
\@gls@sanitizesort: new	18	\@do@wrglossary: changed \theglsentrycounter back to \@glslocref	176
\@gls@setupsort@standard: used \@gls@sanitizesort .	10	\@do@wrglossary: modified to compensate for possible incorrect page number	174
\@printglossary: allow title to override default toctitle	180	\@gls@escbsdq: unsanitize \gls@numberpage, \gls@alphpage, \gls@Alphpage and \gls@romanpage	110
General: allow title to set toctitle	192	\@print@glossary: Moved aux write to end of document to prevent unwanted whatsit occurring here.	182
\glsinlinedescformat: new .	265	General: Added check for doc package	5
\glsinlineemptydescformat: new	265	added datatool-base as a required package	4
\glsinlinenameformat: new .	265	added local key	104
\glsinlinepostchild: new ..	265	\gls@Alphpage: new	173
\glsinlinesubdescformat: new	265	\gls@alphpage: new	173
\glsinlinesubnameformat: new	265	\gls@disablepagerefexpansion: new	173
\glspostinline: replaced “.” with \glspostdescription	265		
atlongragged4col: added check for glsnogroupskip ..	279		
altsuperragged4col: added check for glsnogroupskip ..	296		
alttree: added check for glsnogroupskip	304		
index: added check for glsnogroupskip	298		
nogroupskip: new	9		

\gls@numberpage: new	173	\GlsSetXdyCodePage: Added	
\gls@protected@pagefmts:		check for fontspec	49
new	173	\SetDescriptionAcronymDisplayStyle:	
\gls@romanpage: new	173	now using \glsdoparenifnotempty	
\glsdefmain: added check for		233
doc package	12	\setglossarypreamble: new ..	37
\glsorg@endtheglossary: new .	5	3.08a	
\glsorg@theglossary: new	5	\@glo@storeentry: no longer	
altlist: replaced \newline with		need to check for special char-	
paragraph break	267	acters in any of the fields other	
\PrintChanges: new	5	than sort	83
3.05		updated for \glossentry	83
\@do@wrglossary: add Roman		\@glossaryentryfield: switched	
case. Fixed bugs in the else		to \glossentry	82
statements	174	\@glossarysubentryfield:	
\@gls@link: added check for “no-		switched to \subglossentry	82
hypertypes”	107	General: added nogroupskip key	
\@gls@nohyperlist: new	16	to \printglossary	193
mcolalttree: replaced ‘2’ with		removed definition of	
\glsmcols	283	\@glossaryentryfield ..	344
mcolindex: replaced ‘2’ with		removed definition of	
\glsmcols	281	\@glossarysubentryfield	344
mcoltree: replaced ‘2’ with		\compatibleglossentry: new	199
\glsmcols	282	\compatiblesubglossentry:	
mcoltreename: replaced ‘2’		new	200
with \glsmcols	283	\glossaryentryfield: depre-	
\gls@protected@pagefmts:		cated	201
added Roman to list	173	\Glossentrydesc: new	200
\gls@Romanpage: new	173	\glossentrydesc: new	199
\GlsDeclareNoHyperList: new	16	\Glossentryname: new	199
\glsgetgrouplabel: fixed bug		\glossentryname: new	199
(typo in \equal)	203	\Glossentrysymbol: new	200
\nopostdesc: made robust	33	\glossentrysymbol: new	200
nohypertypes: new	16	\gls@assign@desc@field: new	18
3.06		\gls@assign@descplural@field:	
\@xdy@main@language: Changed		new	18
back to using \language	25	\gls@assign@field: new	66
\findrootlanguage: Obsoleted	48	\gls@ifnotmeasuring: new ...	84
3.07		\glsaddallunused: new	155
\@gls@link: fixed bug that failed		\glsexpandfields: new	66
to find entry in list	107	\glsnoexpandfields: new	66
\glossarypreamble: modified to		\glssee: made robust	177
work with \setglossarypreamble		\glsseeformat: made robust ..	177
.....	37	\glsseeitem: made robust	178
\gls@doclearpage: added check		\glsseelist: made robust	178
for openright	39	\ifglsdescs suppressed: new ..	53
\glspostdescription: Added		\ifglsdesc: new	53
spacefactor code	9	\ifglsdescsymbol: new	53

list: updated list style to use \glossentry and \subglossentry	266	\Glsentryfullpl: made robust	152
listdotted: updated listdotted style to use \glossentry and \subglossentry	268	\Glsentrylong: made robust ..	151
altlist: updated altlist style to use \glossentry and \subglossentry	267	\Glsentrylongpl: made robust	151
altlongragged4col: updated to use \glossentry and \subglossentry	279	\Glsentryname: made robust ..	146
alttree: updated to use \glossentry and \subglossentry	302	\Glsentryplural: made robust	148
index: added paragraph break at end of environment	298	\Glsentryshort: made robust ..	151
updated to use \glossentry and \subglossentry	298	\Glsentryshortpl: made robust	151
inline: updated inline style to use \glossentry and \subglossentry	264	\Glsentrysymbol: made robust	148
long: updated to use \glossentry and \subglossentry	270	\Glsentrysymbolplural: made robust	148
longragged: updated to use \glossentry and \subglossentry	276	\Glsentrytext: made robust ..	148
longragged3col: updated to use \glossentry and \subglossentry	277	\Glsentryuseri: made robust ..	149
tree: updated to use \glossentry and \subglossentry	299	\Glsentryuserii: made robust	150
\setglossarystyle: new	203	\Glsentryuseriii: made robust	150
\setglossentrycompatibility: new	200	\Glsentryuseriv: made robust	150
superragged: updated to use \glossentry and \subglossentry	292	\Glsentryuserv: made robust ..	150
3.09a		\Glsentryuservi: made robust	151
\@gls@assign@symbolplural@field: new	18	\glstextup: new	209
\@gls@default@value: new ...	60	\ifglshassymbol: changed test to check for \@gls@default@symbol	53
\Glsentrydesc: made robust ..	147	3.10a	
\Glsentrydescplural: made robust	147	\@gls@keymap: new	68
\Glsentryfirst: made robust ..	149	\@gls@provide@newglossary: new	55
\Glsentryfirstplural: made robust	149	\@gls@writedef: new	67
\Glsentryfull: made robust ..	152	\@gls@defaultplural: Obsolete ..	63
		\@gls@nodelsc: new	63
		\@print@glossary: Added providecommand code to aux file	182, 183
		\gls@assign@type@field: new	17
		\gls@defglossaryentry: Changed to using \@gls@default@value	76
		new	76
		\glswritedefhook: new	75
		\makeglossaries: Added providecommand code to aux file	164
		\new@glossaryentry: new	67
		\ns@newglossary: added \@gls@provide@newglossary	57
		3.11a	
		\@ACRlong: added \glslabel, \glsifplural, \glsapscase,	

\glsinsert and \glscustomtext	344	343
\@ACRshort: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	343	\@acrshort: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	342
\@Acrlong: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	344	\@gls@: add \glslabel, \glsifplural, \glscapscase, \glscustomtext and \glsinsert	118
\@ACRshort: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	343	change to using \glsentryfmt style commands	118
\@GLS@: add \glslabel, \glsifplural, \glscapscase, \glscustomtext and \glsinsert	120	\@gls@noexpand@fields: Fixed bug expand replaced with noexpand	64
change to using \glsentryfmt style commands	120	\@glsdisp: add \glslabel, \glsifplural, \glscapscase, \glscustomtext and \glsinsert	123
removed \MakeUppercase (now moved to \glsentryfmt)	120	change to using \glsentryfmt style commands	123
\@GLSpl: add \glslabel, \glsifplural, \glscapscase, \glscustomtext and \glsinsert	122	\@glspl@: add \glslabel, \glsifplural, \glscapscase, \glscustomtext and \glsinsert	121
change to using \glsentryfmt style commands	123	change to using \glsentryfmt style commands	121
removed \MakeUppercase as now dealt with in \glsentryfmt	123	General: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	138–145
\@Gls@: add \glsifplural, \glscapscase, \glscustomtext and \glsinsert	119	changed to just use \Glsentrydescplural	130
change to using \glsentryfmt style commands	119	changed to just use \glsentrydescplural	130
removed \makefirstuc (now dealt with in \glsentryfmt)	119	changed to just use \Glsentrydesc	129
\@Glspl@: add \glsifplural, \glscapscase, \glscustomtext and \glsinsert	122	changed to just use \glsentrydesc	129
change to using \glsentryfmt style commands	122	changed to just use \Glsentryfirstplural	127
removed \makefirstuc (now dealt with in \glsentryfmt)	122	changed to just use \glsentryfirstplural	127
\@acrlong: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext		changed to just use \Glsentryfirst	125
		changed to just use \glsentryfirst	125, 126
		changed to just use \Glsentryname	128

changed to just use \glentryname	\defglentryfmt:new	55
..... 128	\forentries: replaced \ifx	
changed to just use \Glsentryplural	with \ifdefempty	50
..... 126	\gls@assign@desc:new	75
changed to just use \glentryplural	\gls@defglossaryentry: Fixed	
..... 126, 127	default counter if none sup-	
changed to just use \Glsentrysymbolplural	plied	79
..... 132	\gls@doentryfmt:new	55
changed to just use \glentrysymbolplural	\glsdisplay:obsoleted	102
..... 131, 132	\glsdisplayfirst:obsoleted	102
changed to just use \Glsentrysymbol	\glsgenentryfmt:new	97
..... 131	\glsgetgrouptitle: Added	
changed to just use \glentrysymbol	check in case non-Latin alpha-	
..... 130, 131	bet in use	202
Changed to just use	\glsglossarymark: replaced	
\Glsentrytext	\MakeUppercase with	
changed to just use \glentrytext	\mfirstucMakeUppercase	38
..... 124	\glsnavigation: switched to us-	
changed to just use \Glsentryuseriii	ing \@gls@getgrouptitle	263
..... 134	\ifglshasdesc: replaced	
changed to just use \glentryuseriii	\ifdefempty with \ifcsempy	
..... 134, 135	53
changed to just use \Glsentryuserii	\ifglshaslong:new	53
..... 133	\ifglshasshort:new	53
changed to just use \glentryuserii	\ifglshassymbol: replaced	
..... 133, 134	\ifdefempty with \ifcsempy	
changed to just use \Glsentryuseriv	53
..... 135	\ifglused:replaced \ifthenelse	
changed to just use \glentryuseriv	with \ifbool	51
..... 135, 136	\longnewglossaryentry:new	75
changed to just use \Glsentryuseri	\ns@newglossary: replaced	
..... 133	\glsdisplay and \glsdisplayfirst	
changed to just use \glentryuseri	with \glentryfmt	57
..... 132, 133	compatible-3.07:cnew	27
changed to just use \Glsentryuservi	\SetCustomDisplayStyle: up-	
..... 137	dated to use \defglentryfmt	
changed to just use \glentryuservi	243
..... 137	\SetDefaultAcronymDisplayStyle:	
changed to just use \Glsentryuserv	changed to use \defglentryfmt	
..... 136	227
changed to just use \glentryuserv	\SetDescriptionAcronymDisplayStyle:	
..... 136	updated to use \defglentryfmt	
Now requires textcase	233
..... 4	\SetDescriptionDUAAcronymDisplayStyle:	
acronymlists: replaced	updated to use \defglentryfmt	
\@addtoacronymlists with	231
\DeclareAcronymList	\SetDescriptionFootnoteAcronymDisplayStyle:	
.... 15	updated to use \defglentryfmt	
\defgldisplay:obsoleted	229
... 102		
\defgldisplayfirst: obso-		
leted		
..... 102		

\SetDUADisplayStyle: updated	17
to use \defglentryfmt .. 241	\gls@checkseeallowed: new .. 61
\SetFootnoteAcronymDisplayStyle:	\glsaddallunused: set default to
updated to use \defglentryfmt	\@glo@types 155
..... 235	\Glsentryfull: changed to use
\SetSmallAcronymDisplayStyle:	\acrfullformat 152
updated to use \defglentryfmt	\glentryfull: changed to use
..... 238	\acrfullformat 151
\setupglossaries: new 28	\Glsentryfullpl: changed to
\showglolong: new 248	use \acrfullformat 152
\showgloshort: new 248	\glentryfullpl: changed to
numbers: new 27	use \acrfullformat 152
symbols: new 27	\glsglossarymark: renamed
3.12a	\glossarymark to \glsglossarymark
\gls@defglossaryentry: added	to avoid conflict with memoir 38
\glslabel 76	\glsprestandardsort: new ... 10
\glsaddkey: new 70	\glsetexpandfield: new 17
3.13a	\glsetnoexpandfield: new .. 17
\@gls@assign@symbol@field:	altsuper4colheader: switched
changed to use \glsetnoexpandfield	to \tabularnewline 290
..... 18	altsuper4colheaderborder:
\@gls@assign@symbolplural@field:	switched to \tabularnewline
changed to use \glsetnoexpandfield 291
..... 18	long: switched to \tabularnewline
\@gls@link: removed \relax . 107 270
\@gls@notranslatorhook: new 21	long3col: switched to \tabularnewline
\@gls@setupsort@standard: 271
moved \@gls@santizesort	long3colheader: switched to
to \glsprestandardsort .. 10	\tabularnewline 272
ucmark: added check for memoir . 9	long3colheaderborder: switched
see: added \gls@checkseeallowed	to \tabularnewline 272
..... 61	long4col: switched to \tabularnewline
\glossarysection: changed 272
\glossarymark to \glsglossarymark	long4colheader: switched to
..... 37	\tabularnewline 273
\glossarystyle: fixed bug	longheader: switched to
caused by using \ifdef in-	\tabularnewline 270
stead of \ifcsdef 204	longheaderborder: switched to
\gls@assign@desc@field:	\tabularnewline 271
changed to use \glsetnoexpandfield	\SetFootnoteAcronymDisplayStyle:
..... 18	fixed missing argument bug 236
\gls@assign@descplural@field:	super: switched to \tabularnewline
changed to use \glsetnoexpandfield 285
..... 18	super3col: switched to
\gls@assign@name@field:	\tabularnewline 287
changed to use \glsetnoexpandfield	super3colheader: switched to
..... 18	\tabularnewline 287
\gls@assign@type@field:	super4col: switched to
changed to use \glsetexpandfield	\tabularnewline 288

super4colheader: switched to \tabularnewline 289	\ACRfullplfmt: new 212
super4colheaderborder: switched to \tabularnewline 290	\Acrfullplfmt: new 212
superheader: switched to \tabularnewline 286	\acrfullplfmt: new 211
superheaderborder: switched to \tabularnewline 286	\acronymentry: new 214
3.14a	sanitize: fixed bug that caused an error here 21
\@glswritefiles: renamed \glswritefiles to \@glswritefiles and used "savewrites" option to set \glswritefiles 170	sc-short-long: new 218
General: new 251	sc-short-long-desc: new ... 220
acronyms: new 14	\Genacrfullformat: new 101
\gls@defglossaryentry: added check for existence of default glossary 77	\genacrfullformat: new 101
set the default for firstplural to be the value of plural 79	\GenericAcronymFields: new 214
xindygloss: new 26	\Genplacrfullformat: new .. 101
\longprovideglossaryentry: new 76	\genplacrfullformat: new .. 101
compatible-2.07: added check for 2.07 before setting 3.07 compatibility 27	\Glsentryfull: bug fix: added missing \acronymfont 152
notranslate: new 22	\glsentryfull: bug fix: added missing \acronymfont 151
\provideglossaryentry: new . 66	\Glsentryfullpl: bug fix: added missing \acronymfont 152
4.0	\glsentryfullpl: bug fix: added missing \acronymfont 152
\gls@defglossaryentry: added check for first key 79	\glsgenacfmt: new 99
4.01	\GlsUseAcrEntryDisplayStyle: new 215
General: fixed non-value options so that they can be passed to document class 7	\GlsUseAcrStyleDefs: new .. 215
\CustomAcronymFields: in- serted missing comma 243	short-long: new 217
4.02	short-long-desc: new 220
\@acrfull: now using \acrfullfmt 209	xindynoglsnumbers: new 26
\@gls@indexdef: new 28	sm-short-long: new 218
\@gls@numbersdef: new 27	sm-short-long-desc: new ... 220
\@gls@symbolsdef: new 27	\makeglossaries: made pream- ble only 166
General: Removed \acronymfont 141-145	index: new 28
\ACRfullfmt: new 211	\newacronymstyle: new 215
\Acrfullfmt: new 210	long-sc-short: new 218
\acrfullfmt: new 210	long-sc-short-desc: new ... 219
	long-short: new 215
	long-short-desc: new 218
	long-sm-short: new 218
	long-sm-short-desc: new ... 219
	long-sp-short-desc: new ... 219
	footnote: new 223
	footnote-desc: new 225
	footnote-sc: new 224
	footnote-sc-desc: new 225
	footnote-sm: new 224
	footnote-sm-desc: new 225
	\setacronymstyle: new 214

<code>\SetDescriptionAcronymDisplayStyle:</code>	<code>\@gls@fetchfield:new</code> 68
Moved check for empty custom text to prevent unwanted	<code>\@gls@field@link:new</code> 124
parenthetical material 233	<code>\@gls@link:added\glsdetoklabel</code> 106
<code>\SetDescriptionFootnoteAcronymDisplayStyle:</code>	<code>\@gls@link@opts</code> 106
Moved check for empty custom text to prevent unwanted	and <code>\@gls@link@label</code> to
parenthetical material 229	<code>\@gls@link</code> 106
<code>\SetFootnoteAcronymDisplayStyle:</code>	<code>\@gls@writedef:</code> added
Moved check for empty custom text to prevent unwanted	<code>\glsdetoklabel</code> 67
parenthetical material 235	<code>\@glsdisp:</code> removed <code>\glslabel</code>
<code>\SetGenericNewAcronym:new</code> 213	(defined in <code>\@gls@link</code>) . . . 123
<code>\SetSmallAcronymDisplayStyle:</code>	<code>\@glspl@:</code> removed <code>\glslabel</code>
Moved check for empty custom text to prevent unwanted	(defined in <code>\@gls@link</code>) . . . 121
parenthetical material 238	<code>\@printglossary:</code> added
<code>dua:new</code> 221	<code>\glsdetoklabel</code> 181
<code>dua-desc:new</code> 222	General: changed default to
<code>numberedsection:</code> added	<code>\@empty</code> instead of <code>\relax</code> . . 26
<code>namerefoption</code> 6	removed <code>\glslabel</code> (defined in
4.03	<code>\@gls@link</code>) 138
<code>\@@do@wrglossary:</code> added	<code>sc-short-long-desc:</code> redefined
<code>\glsdetoklabel</code> 175	to use accessibility information 349
<code>\@ACRlong:</code> removed <code>\glslabel</code>	<code>\compatibleglossentry:</code> added
(defined in <code>\@gls@link</code>) . . . 344	<code>\glsdetoklabel</code> 325
<code>\@ACRshort:</code> removed <code>\glslabel</code>	<code>\compatiblesubglossentry:</code>
(defined in <code>\@gls@link</code>) . . . 343	added <code>\glsdetoklabel</code> . . . 325
<code>\@Acrlong:</code> removed <code>\glslabel</code>	<code>\Genacrfullformat:</code> redefined
(defined in <code>\@gls@link</code>) . . . 344	to use accessibility information 342
<code>\@Acrshort:</code> removed <code>\glslabel</code>	<code>\genacrfullformat:</code> redefined
(defined in <code>\@gls@link</code>) . . . 343	to use accessibility information 341
<code>\@GLS@:</code> removed <code>\glslabel</code> (defined in <code>\@gls@link</code>) 120	<code>\Genplacrfullformat:</code> redefined
<code>\@GLSpl:</code> removed <code>\glslabel</code> (defined in <code>\@gls@link</code>) . . . 122	to use accessibility information 342
<code>\@Gls@:</code> removed <code>\glslabel</code> (defined in <code>\@gls@link</code>) 119	<code>\genplacrfullformat:</code> redefined
<code>\@Gls@entry@field:new</code> . . . 145	to use accessibility information 342
<code>\@Glspl@:</code> removed <code>\glslabel</code> (defined in <code>\@gls@link</code>) . . . 122	<code>\glossentryname:</code> added
<code>\@acrlong:</code> removed <code>\glslabel</code> (defined in <code>\@gls@link</code>) . . . 343	<code>\glsdetoklabel</code> 199
<code>\@acrshort:</code> removed <code>\glslabel</code> (defined in <code>\@gls@link</code>) . . . 342	<code>\gls@defglossaryentry:</code> added
<code>\@gls@:</code> removed <code>\glslabel</code> (defined in <code>\@gls@link</code>) 118	<code>\glsdetoklabel</code> 76
<code>\@gls@access@display:new</code> . 331	replaced #1 with <code>\@gls@label</code> 77
<code>\@gls@entry@field:new</code> . . . 145	replaced <code>\ifthenelse</code> with
	<code>\ifdefequal</code> 78
	<code>\glsadd:</code> added <code>\glsdetoklabel</code> 154
	<code>\glsaddkey:</code> switched to using
	<code>\@gls@field@link</code> 71

<code>\glsdetoklabel:new</code>	51	replaced <code>\ifthenelse</code> with	
<code>\glsdisplaynumberlist:added</code>		<code>\ifdefequal</code>	82
<code>\glsdetoklabel</code>	152	<code>\glsrefentry:added\glsdetoklabel</code>	
<code>\glsdoifexistsorwarn:new</code> ..	52	197
<code>\glsentryaccess:switched to</code>		<code>\glsreset:added\glsdetoklabel</code>	
<code>using\@gls@entry@field</code> ..	329	84
<code>\glsentrydescaccess:switched</code>		<code>\glsseelist:added\expandafter</code>	
<code>to using\@gls@entry@field</code>	330	<code>commands</code>	178
<code>\glsentrydescpluralaccess:</code>		<code>\glsstepentry:added\glsdetoklabel</code>	
<code>switched to using\@gls@entry@field</code>	330	197
<code>\glsentryfirstaccess:switched</code>		<code>\glsstepsubentry:added</code>	
<code>to using\@gls@entry@field</code>	329	<code>\glsdetoklabel</code>	197
<code>\glsentryfirstplural:added</code>		<code>\glsunset:added\glsdetoklabel</code>	
<code>\glsdetoklabel</code>	149	85
<code>\glsentrylongaccess:switched</code>		<code>short-long:commented spuri-</code>	
<code>to using\@gls@entry@field</code>	330	<code>ous EOL</code>	217
<code>\glsentrylongpluralaccess:</code>		<code>redefined to use accessibility in-</code>	
<code>switched to using\@gls@entry@field</code>	330	<code>formation</code>	347
<code>\glsentrypluralaccess:</code>		<code>short-long-desc:redefined to</code>	
<code>switched to using\@gls@entry@field</code>	329	<code>use accessibility information</code>	348
<code>\glsentryshortaccess:switched</code>		<code>\ifglsdescsuppressed:added</code>	
<code>to using\@gls@entry@field</code>	330	<code>\glsdetoklabel</code>	53
<code>\glsentryshortpluralaccess:</code>		<code>fixed typo</code>	53
<code>switched to using\@gls@entry@field</code>	330	<code>\ifglsentryexists:added</code>	
<code>\glsentrysymbolaccess:</code>		<code>\glsdetoklabel</code>	51
<code>switched to using\@gls@entry@field</code>	329	<code>\ifglschaschildren:added</code>	
<code>\glsentrysymbolpluralaccess:</code>		<code>\glsdetoklabel</code>	52
<code>switched to using\@gls@entry@field</code>	330	<code>\ifglschasdesc:added\glsdetoklabel</code>	
<code>\glsentrytextaccess:switched</code>		53
<code>to using\@gls@entry@field</code>	329	<code>\ifglschasfield:new</code>	54
<code>\glsacacmt:redefined to use</code>		<code>\ifglschaslong:added\glsdetoklabel</code>	
<code>accessibility information</code> ...	339	53
<code>\glsacacmt:redefined to</code>		<code>\ifglschasparent:added</code>	
<code>use accessibility information</code>	337	<code>\glsdetoklabel</code>	52
<code>\glsacacmt:added\glsdetoklabel</code>	153	<code>\ifglschasshort:added</code>	
<code>\glslocalreset:added</code>		<code>\glsdetoklabel</code>	53
<code>\glsdetoklabel</code>	84	<code>\ifglschassymbol:added</code>	
<code>\glslocalunset:added</code>		<code>\glsdetoklabel</code>	53
<code>\glsdetoklabel</code>	85	<code>replaced \ifcempty with</code>	
<code>\glsmoveentry:added\glsdetoklabel</code>	82	<code>\ifdefempty and replaced</code>	
		<code>\ifx with\ifdefequal</code>	53
		<code>\ifglsused:added\glsdetoklabel</code>	
		51
		<code>sm-short-long-desc:redefined</code>	
		<code>to use accessibility informa-</code>	
		<code>tion</code>	349
		<code>long-sc-short-desc:redefined</code>	
		<code>to use accessibility informa-</code>	
		<code>tion</code>	348

long-short: redefined to use accessibility information	346	\showglolongpluralaccess:	
long-short-desc: redefined to use accessibility information	347	added \glsdetoklabel . . .	361
long-sm-short-desc: redefined to use accessibility information	348	\showglongname: added \glsdetoklabel	247
footnote: redefined to use accessibility information	352	\showglongnameaccess: added \glsdetoklabel	360
footnote-desc: redefined to use accessibility information . . .	354	\showgloparent: added \glsdetoklabel	245
footnote-sc: redefined to use accessibility information	354	\showgloplural: added \glsdetoklabel	245
footnote-sc-desc: redefined to use accessibility information	354	\showglopluralaccess: added \glsdetoklabel	360
footnote-sm: redefined to use accessibility information	354	\showgloshort: added \glsdetoklabel	248
footnote-sm-desc: redefined to use accessibility information	355	\showgloshortaccess: added \glsdetoklabel	361
\renewacronymstyle: new . . .	215	\showgloshortpluralaccess: added \glsdetoklabel . . .	361
\showglocounter: added \glsdetoklabel	246	\showglosort: added \glsdetoklabel	248
\showglodesc: added \glsdetoklabel	247	\showglosymbol: added \glsdetoklabel	248
\showglodescaccess: added \glsdetoklabel	361	\showglosymbolaccess: added \glsdetoklabel	361
\showglodescplural: added \glsdetoklabel	248	\showglosymbolplural: added \glsdetoklabel	248
\showglodescpluralaccess: added \glsdetoklabel . . .	361	\showglosymbolpluralaccess: added \glsdetoklabel . . .	361
\showglofirst: added \glsdetoklabel	246	\showglotext: added \glsdetoklabel	245
\showglofirstaccess: added \glsdetoklabel	360	\showglotextaccess: added \glsdetoklabel	360
\showglofirstpl: added \glsdetoklabel	246	\showgloftype: added \glsdetoklabel	246
\showglofirstpluralaccess: added \glsdetoklabel . . .	361	\showglouserii: added \glsdetoklabel	246
\showgloflag: added \glsdetoklabel	249	\showglouseriii: added \glsdetoklabel	247
\showgloindex: added \glsdetoklabel	249	\showglouseriv: added \glsdetoklabel	247
\showglolevel: added \glsdetoklabel	245	\showglouserv: added \glsdetoklabel	247
\showglolong: added \glsdetoklabel	248	\showglouservi: added \glsdetoklabel	247
\showglolongaccess: added \glsdetoklabel	361	dua: fixed bug in \acrfullfmt .	222
		fixed bug in \Acrfullplfmt .	222
		fixed bug in \acrfullplfmt .	222

redefined to use accessibility in-		
formation	349	
dua-desc: commented spurious		
EOL	223	
redefined to use accessibility in-		
formation	352	
4.04		
\@gls@noidx@nosanitizesort:		
new	19	
\@gls@noidx@sanitizesort:		
new	18	
\@gls@nosanitizesort:new ..	18	
\@gls@sanitizesort:new ...	18	
\@glo@addchildren:new	184	
\@glo@do@sortentries:new ..	184	
\@glo@grabfirst:new	189	
\@glo@sortedinsert:new ...	185	
\@glo@sortentries:new	183	
\@glo@sorthandler@case:new	185	
\@glo@sorthandler@letter:		
new	185	
\@glo@sorthandler@nocase:		
new	186	
\@glo@sorthandler@word:new	185	
\@glo@sortmacro@case:new ..	187	
\@glo@sortmacro@def:new ..	187	
\@glo@sortmacro@def@do:new	188	
\@glo@sortmacro@letter:new	186	
\@glo@sortmacro@nocase:new	187	
\@glo@sortmacro@standard:		
new	186	
\@glo@sortmacro@use:new ..	188	
\@glo@sortmacro@word:new ..	186	
\@gls@getcounterprefix:		
added warning if no prefix can		
be formed	176	
\@gls@getothergrouptitle:		
new	203	
\@gls@noidx@do:new	190	
\@gls@noref@warn:new	170	
\@gls@reference:new	192	
\@gls@warnonglossdefined:		
new	17	
\@gls@warnontheGLOSSdefined:		
new	17	
\@no@makeglossaries:new ..	170	
\@print@glossary:new	182	
\@print@noidx@glossary:new	188	
\@print@gloss@setsort:new ..	180	
\@printglossary:new	180	
General: added sort key to print-		
gloss group	195	
\compatibleglossentry:		
changed \newcommand to		
\def as is may or may not be		
defined	325	
\compatiblesubglossentry:		
changed \newcommand to		
\def as is may or may not be		
defined	325	
\defglsdisplayfirst:fixed un-		
wanted space	102	
\glo@grabfirst:new	189	
\gls@defglossaryentry: re-		
placed \ifx with \ifdefvoid	81	
\glsnoidxdisplayloc:new ..	192	
\glsnoidxdisplayloclisthandler:		
new	191	
\glsnoidxloclist:new	191	
\glsnoidxloclisthandler:		
new	191	
\glsnoidxstripaccents:new ..	19	
alttree: moved hangindent and		
parindent assignments out-		
side level test	302	
\makeglossaries: Moved def-		
inition of \glswrite to		
\makeglossaries	164	
\makenoidxglossaries:new ..	166	
\printglossary: changed to use		
new \@printglossary ...	179	
\printnoidxglossaries:new	180	
\printnoidxglossary:new ..	180	
\showgloclist:new	249	
\warn@noprintglossary: Acti-		
vate warning in \makeglossaries		
.....	179	
\writeist: checked for definition		
of \glswrite	156, 160	
4.06		
\@GLS@: added \glsifhyper ..	120	
\@GLSpl: added \glsifhyper ..	123	
\@Gls@: added \glsifhyper ..	119	
\@Glspl@: added \glsifhyper	122	
\@gls@: added \glsifhyper ..	118	
\@gls@numbersdef: added hook		
to set toc title	28	

\@gls@symbolsdef: added hook to set toc title	27	moved check for first use to \@gls@link	118
\@glsdisp: added \glsifhyper	123	\@gls@doautomake: new	26
\@glspl@: added \glsifhyper	121	\@gls@field@link: added assignment of \do@gls@link@checkfirsthyper	124
General: added \glsifhyper ..	138–145	\@gls@forbidtexext: new	55
acronym: added hook to set toc title	13	\@gls@hyp@opt: new	104
acronyms: added hook to set toc title	14	\@gls@link: removed redundancy	107
\glsdefmain: added hook to set toc title	13	renamed \gls@type to \glstype	107
4.07		\@glsdisp: moved \glsifhyper	123
\@glossarysection: added optional argument when using unstarred version	39	moved check for first use to \@gls@link	123
\@gls@noidx@do: added \global in case it's used in a tabular-like style	190	\@glspl@: moved \glsifhyper	121
\Acrfullplfmt: fixed no case change bug	212	moved check for first use to \@gls@link	121
\glsletentryfield: new	145	\@ignored@glossaries: new ..	59
4.08		General: added entrycounter option to printgloss family ..	193
\@ACRlong: added \do@gls@link@checkfirsthyper	344	added nopostdot option to printgloss family	193
\@ACRshort: added \do@gls@link@checkfirsthyper	343	added subentrycounter option to printgloss family	194
\@Acrlong: added \do@gls@link@checkfirsthyper	344	explicitly initialise hyper key ..	104
\@Acrshort: added \do@gls@link@checkfirsthyper	342	moved \glsifhyper ...	138–145
\@GLS@: moved \glsifhyper ..	120	removed \@sACRlongpl	144
moved check for first use to \@gls@link	120	removed \@sAcrlongpl	144
\@GLSpl: moved \glsifhyper ..	123	removed \@sacrlongpl	143
moved check for first use to \@gls@link	123	removed \@sACRlong	142
\@Gls@: moved \glsifhyper ..	119	removed \@sAcrlong	142
moved check for first use to \@gls@link	119	removed \@sacrlong	141
\@Glspl@: moved \glsifhyper	122	removed \@sACRshortpl ...	141
moved check for first use to \@gls@link	122	removed \@sAcrshortpl ...	139
\@acrlong: added \do@gls@link@checkfirsthyper	343	removed \@sACRshort	139
\@acrshort: added \do@gls@link@checkfirsthyper	342	removed \@sAcrshort	138
\@closegls: new	163	removed \@sacrshort	138
\@gls@: moved \glsifhyper ..	118	removed \@sgls@link	105
		removed \@sGLSdescplural	130
		removed \@sGlsdescplural	130
		removed \@sglsdescplural	129
		removed \@sGLSdesc	129
		removed \@sGlsdesc	129
		removed \@sglsdesc	129
		removed \@sglsdisp	123
		removed \@sGLSfirstplural	127
		removed \@sGlsfirstplural	127

removed \@sglsfirstplural	127	removed sPglS	255
removed \@sGLSfirst	126	removed spglS	253
removed \@sGlsfirst	125	removed sPGLSpl	257
removed \@sglsfirst	125	removed sPglSpl	255
removed \@sGLSname	128	removed spglSpl	254
removed \@sGlsname	128	\ACRfull: removed \@sACRfull	210
removed \@sglsname	128	switched to using \@gls@hyp@opt	210
removed \@sGLSplural	126	\Acrfull: removed \@sAcrfull	210
removed \@sGlsplural	126	switched to using \@gls@hyp@opt	210
removed \@sglsplural	126	\acrfull: removed \@sacrfull	209
removed \@sGLSpl	122	switched to using \@gls@hyp@opt	209
removed \@sGlspl	121	\ACRfullpl: removed \@sACRfullpl	212
removed \@sglspl	120	switched to using \@gls@hyp@opt	212
removed \@sGLSsymbolplural	132	\Acrfullpl: removed \@sAcrfullpl	211
removed \@sGlsymbolplural	132	switched to using \@gls@hyp@opt	211
removed \@sglsymbolplural	131	\ACRlong: switched to using	142
removed \@sGLSsymbol	131	\Acrlong: switched to using	142
removed \@sGlsymbol	131	\acrlong: switched to using	141
removed \@sglsymbol	130	\ACRlongpl: switched to using	144
removed \@sGLStext	124	\Acrlongpl: switched to using	143
removed \@sGLstext	125	\ACRshort: switched to using	139
removed \@sglstext	124	\Acrshort: switched to using	138
removed \@sGLSuseriii	135	\acrshort: switched to using	137
removed \@sGlsuseriii	134	\ACRshortpl: switched to using	140
removed \@sGLSuserii	134	\Acrshortpl: switched to using	140
removed \@sGlsuserii	133		
removed \@sglsuserii	133		
removed \@sGLSuseriv	135		
removed \@sGlsuseriv	135		
removed \@sglsuseriv	135		
removed \@sGLSuseri	133		
removed \@sGlsuseri	132		
removed \@sglsuseri	132		
removed \@sGLSuservi	137		
removed \@sGlsuservi	137		
removed \@sglsuservi	137		
removed \@sGLSuserv	136		
removed \@sGlsuserv	136		
removed \@sglsuserv	136		
removed \@sGLS	120		
removed \@sGls	119		
removed \@sgls	118		
removed \@thirdofthree (defined in kernel)	117		
removed sPGLS	256		

\acrshortpl: switched to using \@gls@hyp@opt 139	\Glsfirstplural: switched to using \@gls@hyp@opt 127
\forallacronyms: new 50	\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 ... 77	\glslinkvar: new 104
\gls@istfilebase: new 34	\GLSname: switched to using \@gls@hyp@opt 128
\glsaddkey: removed \@sGLS@user@<key> 72	\Glsname: switched to using \@gls@hyp@opt 128
removed \@sGls@user@<key> . 71	\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 121
\GLSdesc: switched to using \@gls@hyp@opt 129	\glspl: switched to using \@gls@hyp@opt 120
\Glsdesc: switched to using \@gls@hyp@opt 129	\GLSplural: switched to using \@gls@hyp@opt 126
\glsdesc: switched to using \@gls@hyp@opt 128	\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 129	\GLSsymbol: switched to using \@gls@hyp@opt 131
\glsdisablehyper: added \KV@glslink@hyperfalse to definition 117	\Glssymbol: switched to using \@gls@hyp@opt 131
\glsdisp: switched to using \@gls@hyp@opt 123	\glssymbol: switched to using \@gls@hyp@opt 130
\glsdohyperlink: new 116	\GLSsymbolplural: switched to using \@gls@hyp@opt 132
\glsdohypertarget: new 116	\Glssymbolplural: switched to using \@gls@hyp@opt 131
\glsenablehyper: added \KV@glslink@hypertrue to definition 117	\glssymbolplural: switched to using \@gls@hyp@opt 131
\GLSfirst: switched to using \@gls@hyp@opt 125	\GLStext: switched to using \@gls@hyp@opt 124
\Glsfirst: switched to using \@gls@hyp@opt 125	\Glstext: switched to using \@gls@hyp@opt 125
\glsfirst: switched to using \@gls@hyp@opt 125	\glstext: switched to using \@gls@hyp@opt 124
\GLSfirstplural: switched to using \@gls@hyp@opt 127	\glstreenamefmt: new 297

\GLSuseri: switched to using \@gls@hyp@opt 133	\PglS: changed to use \@gls@hyp@opt 255
\Glsuseri: switched to using \@gls@hyp@opt 132	\pgls: changed to use \@gls@hyp@opt 253
\glsuseri: switched to using \@gls@hyp@opt 132	\PGLSpl: changed to use \@gls@hyp@opt 257
\GLSuserii: switched to using \@gls@hyp@opt 134	\Pglspl: changed to use \@gls@hyp@opt 255
\Glsuserii: switched to using \@gls@hyp@opt 133	\pglspl: changed to use \@gls@hyp@opt 254
\glsuserii: switched to using \@gls@hyp@opt 133	\s@gls@hyp@opt:new 104
\GLSuseriii: switched to using \@gls@hyp@opt 134	\s@newglossary:new 56
\Glsuseriii: switched to using \@gls@hyp@opt 134	automake:new 26
\glsuseriii: switched to using \@gls@hyp@opt 134	4.09
\GLSuseriv: switched to using \@gls@hyp@opt 135	\glsaddkey: fixed bug in user commands 71
\Glsuseriv: switched to using \@gls@hyp@opt 135	4.10
\GLSuseriv: switched to using \@gls@hyp@opt 135	\@Gls@acentryname:new ... 146
\GLSuseriv: switched to using \@gls@hyp@opt 135	\@Gls@entryname:new 146
\GLSuseriv: switched to using \@gls@hyp@opt 135	\@gls@glossary: Renamed \@glossary to \@gls@glossary 172
\GLSuseriv: switched to using \@gls@hyp@opt 135	\glspercentchar:new 156
\GLSuseriv: switched to using \@gls@hyp@opt 135	\glstildechar:new 156
\GLSuseriv: switched to using \@gls@hyp@opt 135	alttree: moved space after sym- bol 303, 304
\GLSuseriv: switched to using \@gls@hyp@opt 135	4.11
\GLSuseriv: switched to using \@gls@hyp@opt 135	\@@do@wrglossary: added hook 175
\GLSuseriv: switched to using \@gls@hyp@opt 135	sanitize: none option 21
\GLSuseriv: switched to using \@gls@hyp@opt 135	\gls@wrglossary: renamed from \@wrglossary to \gls@wrglossary 172
\GLSuseriv: switched to using \@gls@hyp@opt 135	\glsaddprotectedpagefmt: new 174
\GLSuseriv: switched to using \@gls@hyp@opt 135	\glsbackslash:new 155
\GLSuseriv: switched to using \@gls@hyp@opt 135	4.12
\ifignoredglossary:new 59	\@gls@addpredefinedattributes: Added glsignore attribute ... 43
altlongragged4col: fixed bug that displayed description in- stead of symbol 279	\@gls@adjustmode:new 154
\newglossary: added starred ver- sion 56	\@gls@notranslatorhook: re- moved 21
\newignoredglossary:new ... 58	\@gls@toc: added \protect to \numberline 40
\ns@newglossary: added \@glotype@<name>@log ... 57	\@gls@usetranslator:new ... 22
new 56	\glsacrpluralsuffix:new ... 31
\p@gls@hyp@opt:new 104	\glsadd: added check for vertical mode 154
\PGLS: changed to use \@gls@hyp@opt 256	\glsaddallunused: replaced @gobble with glsignore 155

\glsifusedtranslatordict:	\glslocalunset: switched to
new 22	\@glslocalunset 85
\glsignore:new 155	\glsreset: switched to
\glsupacrpluralsuffix:new . 31	\@glsreset 84
\ProvidesGlossariesLang:	\glsunset: switched to
new 31	\@glsunset 85
\RequireGlossariesLang:new 31	4.15
4.13	General: bug fix replaced
\indexspace:new ... 266, 281, 297	\@glo@type with \glstype 144
4.14	4.16
\@glslocalreset:new 86	\@ACRlong:added \glspostlinkhook
\@glslocalunset:new 85 344
\@glsreset:new 86	\@ACRshort:added \glspostlinkhook
\@glsunset:new 85 343
\@newglossaryentry@defcounters:	\@Acrlong:added \glspostlinkhook
new 87 344
\@cGls:new 90	\@Acrshort:added \glspostlinkhook
\@cGls@:new 90 343
\@cGlspl@:new 92	\@GLS@:added \glspostlinkhook
\@cgls:new 90 120
\@cgls@:new 90	\@GLSpl:added \glspostlinkhook
\@cglspl:new 91 123
\@cglspl@:new 91	\@Gls@:added \glspostlinkhook
\@gls@entry@count:new 90 119
\@gls@increment@currcount:	\@Glspl@:added \glspostlinkhook
new 89 122
\@gls@local@increment@currcount:	\@acrlong:added \glspostlinkhook
new 89 344
\@gls@write@entrycounts:	\@acrshort:added \glspostlinkhook
new 89 342
\@glslocalreset:new 85	\@gls@:added \glspostlinkhook
\@glslocalunset:new 85 119
\@glsreset:new 86	\@gls@@link:added \glspostlinkhook
\@glsunset:new 85 105
\@newglossaryentry@defcounters:	\@gls@field@link: added
new 82	\glspostlinkhook 124
\cGls:new 90	\@gls@link: moved definition
\cgls:new 90	of \glsifhyperon outside of
\cGlsformat:new 91	this macro 107
\cglsformat:new 90	\@glsdisp:added \glspostlinkhook
\cGlspl:new 91 124
\cglspl:new 91	\@glspl@:added \glspostlinkhook
\cGlsplformat:new 92 121
\cglsplformat:new 91	General:added \glspostlinkhook
\gls@defdocnewglossaryentry: 138–145
new 66	\glsacspace:new 217
\glsenableentrycount:new .. 87	\glsadd:changed \@do@wrglossary
\glslocalreset: switched to	to \@do@wrglossary 154
\@glslocalreset 84	\glsaddstoragekey:new 69

<code>\glsfielddef:new</code>	73	<code>\glspostlinkhook:new</code>	105
<code>\glsfieldedef:new</code>	72	<code>\glswriteentry:new</code>	173
<code>\glsfieldfetch:new</code>	73	<code>\ifglsfieldcseq:new</code>	75
<code>\glsfieldgdef:new</code>	73	<code>\ifglsfielddefeq:new</code>	74
<code>\glsfieldxdef:new</code>	72	<code>\ifglsfieldeq:new</code>	74
<code>\glsifhyperon</code> : moved defini-		<code>long-sp-short:new</code>	216
tion of <code>\glsifhyperon</code>	106	<code>\showglofield:new</code>	249
<code>\glslinkpostsetkeys:new</code> ..	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	\@cGls	90
\@@do@wrglossary	174	\@cGls@	90
\@@glo@assign@sortkey	195	\@cGlspl@	92
\@@glossarysec	6	\@cgls	90
\@@glossaryseclabel	6	\@cgls@	90
\@@glossarysecstar	6	\@cglspl	91
\@@gls@default@entryfmt	333	\@cglspl@	91
\@@gls@expand@field	65	\@closegls	163
\@@gls@fixbraces	177	\@delimN	206
\@@gls@noidx@nosanitizesort	19	\@delimR	206
\@@gls@noidx@sanitizesort	18	\@disable@onlypremakeg	30
\@@gls@nosanitizesort	18	\@disable@premakecs	30
\@@gls@sanitizesort	18	\@disabled@glsaddxdycounters	42
\@@glslocalreset	86	\@do@seeglossary	177
\@@glslocalunset	85	\@do@wrglossary	173, 306
\@@glsreset	86	\@glo@addchildren	184
\@@glsunset	85	\@glo@default@sorttype	10
\@@newglossaryentry@defcounters		\@glo@do@sortentries	184
	87	\@glo@grabfirst	189
\@ACRlong	344	\@glo@no@assign@sortkey	195
\@ACRshort	343	\@glo@seeautonumberlist	8
\@Acrlong	344	\@glo@sortedinsert	185
\@Acrshort	342	\@glo@sortentries	183
\@GLS@	120	\@glo@sorthandler@case	185
\@GLSpl	122	\@glo@sorthandler@letter	185
\@Gls@	119	\@glo@sorthandler@nocase	186
\@Gls@acrentryname	146	\@glo@sorthandler@word	185
\@Gls@entry@field	145	\@glo@sortmacro@case	187
\@Gls@entryname	146	\@glo@sortmacro@def	187
\@Glspl@	121	\@glo@sortmacro@def@do	188
\@PGLS	256	\@glo@sortmacro@letter	186
\@PGLS@	256	\@glo@sortmacro@nocase	187
\@PGLSpl	257	\@glo@sortmacro@standard	186
\@PGLSpl@	257	\@glo@sortmacro@use	188
\@Pgls	255	\@glo@sortmacro@word	186
\@Pgls@	255	\@glo@storeentry	82
\@Pglspl	255	\@glo@types	55
\@Pglspl@	256	\@glossary@default@style	7
\@acrfull	209	\@glossaryentryfield	82
\@acrlong	343	\@glossarysection	39
\@acrshort	342	\@glossarysubentryfield	82
\@addtoacronymlists	14	\@gls	118
		\@gls@	118

<code>\@gls@link</code>	105	<code>\@gls@makefirstuc</code>	258
<code>\@gls@access@display</code>	331	<code>\@gls@missingnumberlist</code>	64
<code>\@gls@addpredefinedattributes</code>	43	<code>\@gls@noaccess</code>	327
<code>\@gls@adjustmode</code>	154	<code>\@gls@noexpand@field</code>	64
<code>\@gls@assign@symbol@field</code> ...	18	<code>\@gls@noexpand@fields</code>	64
<code>\@gls@assign@symbolplural@field</code>	18	<code>\@gls@nohyperlist</code>	16
<code>\@gls@checkactual</code>	115	<code>\@gls@noidx@do</code>	190
<code>\@gls@checkbar</code>	114	<code>\@gls@noidx@setsanitizesort</code> .	21
<code>\@gls@checkescactual</code>	112	<code>\@gls@noref@warn</code>	170
<code>\@gls@checkescbar</code>	112	<code>\@gls@notranslatorhook</code>	21
<code>\@gls@checkesclevel</code>	113	<code>\@gls@numbersdef</code>	27
<code>\@gls@checkescquote</code>	111	<code>\@gls@onlypremakeg</code>	29
<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>	10	<code>\@gls@sanitizename</code>	18
<code>\@gls@declareoption</code>	7	<code>\@gls@sanitizesort</code>	18
<code>\@gls@default@value</code>	60	<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>	24
<code>\@gls@entry@count</code>	90	<code>\@gls@setcounter</code>	58
<code>\@gls@entry@field</code>	145	<code>\@gls@setupsort@def</code>	11
<code>\@gls@escbsdq</code>	109	<code>\@gls@setupsort@standard</code>	10
<code>\@gls@expand@fields</code>	65	<code>\@gls@setupsort@use</code>	11
<code>\@gls@fetchfield</code>	68	<code>\@gls@startswithexpandonce</code> ..	65
<code>\@gls@field@link</code>	124	<code>\@gls@symbolsdef</code>	27
<code>\@gls@fixbraces</code>	177	<code>\@gls@tmpb</code>	111
<code>\@gls@forbidtext</code>	55	<code>\@gls@toc</code>	40
<code>\@gls@getcounter</code>	58	<code>\@gls@updatechecked</code>	111
<code>\@gls@getcounterprefix</code>	176	<code>\@gls@usetranslator</code>	22
<code>\@gls@getgrouptitle</code>	202	<code>\@gls@warnonglossdefined</code>	17
<code>\@gls@getothergrouptitle</code> ...	203	<code>\@gls@warnonthe glossdefined</code> .	17
<code>\@gls@glossary</code>	172	<code>\@gls@write@entrycounts</code>	89
<code>\@gls@hyp@opt</code>	104	<code>\@gls@writedef</code>	67
<code>\@gls@hypergroup</code>	262	<code>\@gls@xdy@Lclass@Alpha-page-numbers</code>	45
<code>\@gls@ifinlist</code>	41	<code>\@gls@xdy@Lclass@Appendix-page-numbers</code>	45
<code>\@gls@increment@currcount</code> ...	89	<code>\@gls@xdy@Lclass@Roman-page-numbers</code>	45
<code>\@gls@indexdef</code>	28	<code>\@gls@xdy@Lclass@alpha-page-numbers</code>	45
<code>\@gls@keymap</code>	68, 251	<code>\@gls@xdy@Lclass@arabic-page-numbers</code>	45
<code>\@gls@link</code>	106	<code>\@gls@xdy@Lclass@arabic-section-numbers</code>	45
<code>\@gls@loadlist</code>	8	<code>\@gls@xdy@Lclass@roman-page-numbers</code>	44
<code>\@gls@loadlong</code>	8		
<code>\@gls@loadsuper</code>	8		
<code>\@gls@loadtree</code>	8		
<code>\@gls@local@increment@currcount</code>	89		

<code>\ACRlong</code>	142	<code>\Acsp</code>	226
<code>\Acrlong</code>	142	<code>\acsp</code>	226
<code>\acrlong</code>	141	<code>\addglossarytocaptions</code>	32
<code>\ACRlongpl</code>	144	<code>\addto</code>	32
<code>\Acrlongpl</code>	144	<code>align (environment)</code>	84, 108
<code>\acrlongpl</code>	143	<code>altlist (style)</code>	267
<code>\acrnameformat</code>	212, 235	<code>altlistgroup (style)</code>	267
<code>\acrnohreffootnote</code>	229	<code>altlisthypergroup (style)</code>	268
<code>acronym (option)</code>	13	<code>altlong4col (style)</code>	274
acronym styles:		<code>altlong4colborder (style)</code>	274
<code>dua</code>	221, 349	<code>altlong4colheader (style)</code>	274
<code>dua-desc</code>	222, 352	<code>altlong4colheaderborder (style)</code>	274
<code>footnote</code>	223, 352	<code>altlongragged4col (style)</code>	278
<code>footnote-desc</code>	225, 354	<code>altlongragged4colborder (style)</code>	280
<code>footnote-sc</code>	224, 354	<code>altlongragged4colheader (style)</code>	279
<code>footnote-sc-desc</code>	225, 354	<code>altlongragged4colheaderborder</code>	
<code>footnote-sm</code>	224, 354	<code>(style)</code>	280
<code>footnote-sm-desc</code>	225, 355	<code>\altnewglossary</code>	57
<code>long-sc-short</code>	218	<code>altsuper4col (style)</code>	290
<code>long-sc-short-desc</code>	219, 348	<code>altsuper4colborder (style)</code>	290
<code>long-short</code>	215, 346	<code>altsuper4colheader (style)</code>	290
<code>long-short-desc</code>	218, 347	<code>altsuper4colheaderborder</code>	
<code>long-sm-short</code>	218	<code>(style)</code>	291
<code>long-sm-short-desc</code>	219, 348	<code>altsuperragged4col (style)</code>	295
<code>long-sp-short</code>	216	<code>altsuperragged4colborder</code>	
<code>long-sp-short-desc</code>	219	<code>(style)</code>	296
<code>sc-short-long</code>	218	<code>altsuperragged4colheader</code>	
<code>sc-short-long-desc</code>	220, 349	<code>(style)</code>	296
<code>short-long</code>	217, 347	<code>altsuperragged4colheaderborder</code>	
<code>short-long-desc</code>	220, 348	<code>(style)</code>	297
<code>sm-short-long</code>	218	<code>almtree (style)</code>	302
<code>sm-short-long-desc</code>	220, 349	<code>almtreegroup (style)</code>	304
<code>\acronymentry</code>	214	<code>almtreehypergroup (style)</code>	304
<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>	275, 291
<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>	22, 30, 32, 48
<code>\Acrshort</code>	138		
<code>\acrshort</code>	137	C	
<code>\ACRshortpl</code>	140	<code>\capitalisewords</code>	259
<code>\Acrshortpl</code>	140	<code>\cGls</code>	90
<code>\acrshortpl</code>	139	<code>\cgls</code>	90
<code>\Acs</code>	226	<code>\cGlsformat</code>	91
<code>\acs</code>	226	<code>\cglsformat</code>	90
		<code>\cGlspl</code>	91

`\cglspl` 91
`\cGlsplformat` 92
`\cglsplformat` 91
`\compatglossarystyle` 311
`compatible-2.07 (option)` 27
`compatible-3.07 (option)` 27
`\compatibleglossentry` .. 199, 325
`\compatiblesubglossentry` 200, 325
`counter (key)` 61
`counter (option)` 16
`\CustomAcronymFields` 243
`\CustomNewAcronymDef` 243

D

`datatool package` 185
`\DeclareAcronymList` 14
`\DefaultNewAcronymDef` .. 227, 355
`\defentryfmt` 103
`\defglstdisplay` 102
`\defglstdisplayfirst` 102
`\defglsentry` 57
`\defglsentryfmt` 55, 59, 61, 94
`\DefineAcronymSynonyms` 225
`\delimN` 36, 205
`\delimR` 36, 205
`description (environment)` 266
`description (key)` 59
`description (option)` 24
`descriptionaccess (key)` 326
`\DescriptionDUANewAcronymDef` 231
`\DescriptionFootnoteNewAcronymDef`
..... 230, 356
`\descriptionname` 31
`\DescriptionNewAcronymDef` ..
..... 234, 357
`descriptionplural (key)` 60
`descriptionpluralaccess (key)` 327
`\detokenize` 51
`doc package` 5, 12
`document (environment)` 66, 87
`dua (acrstyle)` 221, 349
`dua (option)` 24
`dua-desc (acrstyle)` 222, 352
`\DUANewAcronymDef` 241

E

`\ecapitalisewords` 260
`\emakefirsttuc` 259
`entrycounter (option)` 9
`entrycounterwithin (option)` 9

`\entryname` 31
environments:
 `align` 84, 108
 `description` 266
 `document` 66, 87
 `longtable` 8, 244, 269–280
 `multicols` 281
 `supertabular` ... 8, 244, 284–297
 `theglossary` ... 5, 17, 36, 37,
 198, 204, 282, 283, 299, 300, 302
 `theindex` 297
`equation (counter)` 108
`etoolbox package` 5, 257

F

file types
 `.aux` 182
 `.glo` 82
 `.ist` 155, 162
 `.toc` 40
 `.xdy` 34
 `glo` 250
`\findrootlanguage` 48
`first (key)` 60
`firstaccess (key)` 326
`\firstacronymfont` 99, 212
`firstplural (key)` 60
`firstpluralaccess (key)` 326
`footnote (acrstyle)` 223, 352
`footnote (option)` 24
`footnote-desc (acrstyle)` .. 225, 354
`footnote-sc (acrstyle)` 224, 354
`footnote-sc-desc (acrstyle)` 225, 354
`footnote-sm (acrstyle)` 224, 354
`footnote-sm-desc (acrstyle)` 225, 355
`\FootnoteNewAcronymDef` . 236, 358
`\forallacronyms` 50
`\forallglossaries` 49
`\forallglstentries` 50
`\forglstentries` 50

G

`garamondx package` 209
`\Genacrfullformat` 101, 342
`\genacrfullformat` 101, 341
`\GenericAcronymFields` 214
`\Genplacrfullformat` 101, 342
`\genplacrfullformat` 101, 342
`\glo@grabfirst` 189
`\glolinkprefix` 107

glossareentry (counter) 197
glossaries package 28,
48, 156, 244, 251, 266, 305, 325
glossaries-accsupp package ... 82, 325
\GlossariesWarning 16
\GlossariesWarningNoLine 16
\glossary 56, 162, 171, 203
glossary counters:
 glossaryentry 196
 glossarysubentry 196
glossary keys:
 access 326
 counter 61
 description 59
 descriptionaccess 326
 descriptionplural 60
 descriptionpluralaccess . 327
 first 60
 firstaccess 326
 firstplural 60
 firstpluralaccess 326
 long 63
 longaccess 327
 longplural 63
 longpluralaccess 327
 name 59
 nonumberlist 62
 parent 62
 plural 60
 pluralaccess 326
 see 61
 short 63
 shortaccess 327
 shortplural 63
 shortpluralaccess 327
 sort 60
 symbol 61
 symbolaccess 326
 symbolplural 61
 symbolpluralaccess 326
 text 60
 textaccess 326
 type 61
 user1 62
 user2 62
 user3 62
 user4 62
 user5 62
 user6 63
glossary package 1, 208
glossary styles:
 altlist 267, 268, 313
 altlist 267
 altlistgroup 267, 268, 313
 altlistgroup 267
 altlisthypergroup ... 268, 313
 altlisthypergroup 268
 altlong4col 274, 278, 315
 altlong4col 274
 altlong4colborder ... 274, 315
 altlong4colborder 274
 altlong4colheader ... 274, 315
 altlong4colheader 274
 altlong4colheaderborder .
 274, 315
 altlong4colheaderborder . 274
 altlongragged4col 278–280, 316
 altlongragged4col 278
 altlongragged4colborder .
 280, 317
 altlongragged4colborder . 280
 altlongragged4colheader .
 279, 317
 altlongragged4colheader . 279
 altlongragged4colheaderborder
 280, 317
 altlongragged4colheaderborder
 280
 altsuper4col . 290, 291, 295, 324
 altsuper4col 290
 altsuper4colborder .. 290, 324
 altsuper4colborder 290
 altsuper4colheader .. 290, 324
 altsuper4colheader 290
 altsuper4colheaderborder
 291, 324
 altsuper4colheaderborder 291
 altsuperragged4col 295–297, 322
 altsuperragged4col 295
 altsuperragged4colborder
 296, 323
 altsuperragged4colborder 296
 altsuperragged4colheader
 296, 322
 altsuperragged4colheader 296
 altsuperragged4colheaderborder
 297, 323

altsuperragged4colheaderborder	297
alttree	283, 302, 304, 319
alttree	302
alttreegroup	304, 320
alttreegroup	304
alttreehypergroup	304, 320
alttreehypergroup	304
index	281, 297–299, 317
index	297
indexgroup	298, 299, 317
indexgroup	298
indexhypergroup	299, 318
indexhypergroup	299
inline	312
inline	263
list	7, 266–268, 312
list	266
listdotted	268, 269, 313
listdotted	268
listgroup	266, 267, 313
listgroup	266
listhypergroup	267, 313
listhypergroup	267
long	269–271, 275, 314, 315
long	269
long3col	271, 272, 314
long3col	271
long3colborder	271, 314
long3colborder	271
long3colheader	272, 314
long3colheader	272
long3colheaderborder	272, 314
long3colheaderborder	272
long4col	272–274, 315
long4col	272
long4colborder	273, 315
long4colborder	273
long4colheader	273, 315
long4colheader	273
long4colheaderborder	273, 315
long4colheaderborder	273
longborder	270, 314
longborder	270
longheader	270, 314
longheader	270
longheaderborder	270, 314
longheaderborder	270
longragged	275–277
longragged	275
longragged3col	277, 278, 316
longragged3col	277
longragged3colborder	278, 316
longragged3colborder	278
longragged3colheader	278, 316
longragged3colheader	278
longragged3colheaderborder	278, 316
longragged3colheaderborder	278
longraggedborder	276, 316
longraggedborder	276
longraggedheader	276, 316
longraggedheader	276
longraggedheaderborder	277, 316
longraggedheaderborder	277
mcolalttree	284, 321
mcolalttree	283
mcolalttreegroup	284, 321
mcolalttreegroup	284
mcolalttreehypergroup	284, 321
mcolalttreehypergroup	284
mcolindex	281, 320
mcolindex	281
mcolindexgroup	281, 320
mcolindexgroup	281
mcolindexhypergroup	281, 320
mcolindexhypergroup	281
mcoltree	282, 320
mcoltree	282
mcoltreegroup	321
mcoltreegroup	282
mcoltreehypergroup	282, 321
mcoltreehypergroup	282
mcoltreenoname	283, 321
mcoltreenoname	282
mcoltreenonamegroup	283, 321
mcoltreenonamegroup	283
mcoltreenonamehypergroup	283, 321
mcoltreenonamehypergroup	283
sublistdotted	313
sublistdotted	269
super	285, 286, 293, 323
super	285
super3col	286–288, 323
super3col	286
super3colborder	287, 323

super3colborder	287	treenonamehypergroup	301
super3colheader	287, 324	glossary-hypernav package	155
super3colheader	287	glossary-list package	7, 8, 266
super3colheaderborder	288, 324	glossary-long package	
super3colheaderborder ...	288	8, 269, 278, 284, 285
super4col	288–290, 324	glossary-longragged package	275
super4col	288	glossary-mcols package	280
super4colborder	289, 324	glossary-super package	
super4colborder	289	8, 269, 284, 291, 295
super4colheader	289, 324	glossary-superragged package	291
super4colheader	289	glossary-tree package	8, 297
super4colheaderborder	289, 324	glossaryentry (counter) .	9, 197, 198
super4colheaderborder ...	289	glossaryentry (counter)	196
superborder	285, 323	\glossaryentryfield .	201, 204, 205
superborder	285	\glossaryentrynumber ...	195, 196
superheader	286, 323	\glossaryentrynumbers	
superheader	286	7, 36, 180, 181
superheaderborder ...	286, 323	\glossaryheader	198, 204
superheaderborder	286	\glossarymark	38
superragged	292, 293, 321	\glossaryname	30, 32
superragged	292	\glossarypostamble	37, 204
superragged3col ..	293–295, 322	\glossarypreamble	37, 204
superragged3col	293	\glossarysection	6, 37, 56
superragged3colborder	294, 322	\glossarystyle	204, 245
superragged3colborder ...	294	glossarysubentry (counter) ...	
superragged3colheader	294, 322	10, 196–198
superragged3colheader ...	294	glossarysubentry (counter) ...	196
superragged3colheaderborder		\glossarysubentryfield	201
.....	295, 322	\glossentry	61, 199
superraggedborder ...	292, 322	\Glossentrydesc	200
superraggedborder	292	\glossentrydesc	199, 344
superraggedheader ...	293, 322	\Glossentryname	199
superraggedheader	293	\glossentryname	199, 344
superraggedheaderborder .		\Glossentrysymbol	200
.....	293, 322	\glossentrysymbol	200, 344
superraggedheaderborder .	293	\GLS	120
superraggedright3colheaderborder		\Gls	119, 121, 257
.....	295	\gls	4, 61, 92,
tree	282, 299, 300, 302, 318	104, 118–120, 124–137, 197, 253	
tree	299	\gls@Alphpage	173
treegroup	282, 300, 318	\gls@alphpage	173
treegroup	300	\gls@assign@desc	75
treehypergroup	300, 318	\gls@assign@desc@field	18
treehypergroup	300	\gls@assign@descplural@field	18
treenoname ...	282, 300, 301, 318	\gls@assign@field	66
treenoname	300	\gls@assign@name@field	18
treenonamegroup	301, 319	\gls@assign@type@field	17
treenonamegroup	301	\gls@checkisacronymlist	15
treenonamehypergroup	301, 319	\gls@checkseeallowed	61

<code>\gls@codepage</code>	25	<code>\glsdefaulttype</code>	13
<code>\gls@defdocnewglossaryentry</code> .	66	<code>\glsdefmain</code>	12
<code>\gls@defglossaryentry</code>	76	<code>\GLSdesc</code>	129
<code>\gls@disablepagerefexpansion</code>	173	<code>\Glsdesc</code>	129
<code>\gls@doclearpage</code>	39	<code>\glsdesc</code>	128, 129
<code>\gls@doentryfmt</code>	55	<code>\GLSdescplural</code>	130
<code>\gls@glossary</code>	172	<code>\Glsdescplural</code>	130
<code>\gls@hypergroup prerun</code>	262	<code>\glsdescplural</code>	129, 130
<code>\gls@ifnotmeasuring</code>	84	<code>\glsdescriptionaccessdisplay</code>	332
<code>\gls@istfilebase</code>	34	<code>\glsdescriptionpluralaccessdisplay</code>	332
<code>\gls@level</code>	64	332
<code>\gls@noidxglossary</code>	170	<code>\glsdescwidth</code> ...	269, 275, 284, 291
<code>\gls@numberpage</code>	173	<code>\glsdetoklabel</code>	51
<code>\gls@protected@pagefmts</code>	173	<code>\glsdisablehyper</code>	117
<code>\gls@Romanpage</code>	173	<code>\glsdisp</code>	123
<code>\gls@romanpage</code>	173	<code>\glsdisplay</code>	92, 102, 118
<code>\gls@save@numberlist</code>	179	<code>\glsdisplayfirst</code>	92, 102, 118
<code>\gls@suffixF</code>	35	<code>\glsdisplaynumberlist</code>	152, 168, 191
<code>\gls@suffixFF</code>	36	<code>\glsdohyperlink</code>	116
<code>\gls@wrglossary</code>	172	<code>\glsdohypertarget</code>	116
<code>\glsaccessdisplay</code>	332	<code>\glsdoifexists</code>	51
<code>\glsaccsupp</code>	330	<code>\glsdoifexistsorwarn</code>	52
<code>\glsacrpluralsuffix</code>	31	<code>\glsdoifnoexists</code>	51
<code>\glsacspace</code>	217	<code>\glsdoparenifnotempty</code>	238
<code>\glsadd</code>	92, 153, 154, 203	<code>\glsenableentrycount</code>	87
<code>\glsadd options</code>		<code>\glsenablehyper</code>	117
counter	154	<code>\glsentryaccess</code>	329
format	154, 205	<code>\glsentrycounter</code>	107
<code>\glsaddall</code>	51, 92, 154	<code>\glsentrycounterlabel</code>	197
<code>\glsaddall options</code>		<code>\glsentrycurrcount</code>	87
types	154	<code>\Glsentrydesc</code>	147
<code>\glsaddallunused</code>	154	<code>\glsentrydesc</code>	147
<code>\glsaddkey</code>	70	<code>\glsentrydescaccess</code>	330
<code>\GlsAddLetterGroup</code>	49	<code>\Glsentrydescplural</code>	147
<code>\glsaddprotectedpagefmt</code>	173	<code>\glsentrydescplural</code>	147
<code>\GlsAddSortRule</code>	47	<code>\glsentrydescpluralaccess</code> ..	330
<code>\glsaddstoragekey</code>	69	<code>\Glsentryfirst</code>	149
<code>\GlsAddXdyAlphabet</code>	44	<code>\glsentryfirst</code>	149
<code>\GlsAddXdyAttribute</code>	42, 305	<code>\glsentryfirstaccess</code>	329
<code>\GlsAddXdyCounters</code>	41, 305	<code>\Glsentryfirstplural</code>	149
<code>\GlsAddXdyLocation</code>	46, 306	<code>\glsentryfirstplural</code>	149
<code>\GlsAddXdyStyle</code>	47	<code>\glsentryfirstpluralaccess</code> .	329
<code>\glsautoprefix</code>	6	<code>\glsentryfmt</code>	59, 61, 93
<code>\glsbackslash</code>	155	<code>\Glsentryfull</code>	152
<code>\glsclearpage</code>	40	<code>\glsentryfull</code> ...	151, 214, 224, 353
<code>\glsclosebrace</code>	155	<code>\Glsentryfullpl</code>	152
<code>\glscompositor</code>	35, 45	<code>\glsentryfullpl</code>	152
<code>\glscounter</code>	16, 57	<code>\glsentryitem</code>	198
<code>\GlsDeclareNoHyperList</code>	16	<code>\Glsentrylong</code>	151

<code>\glentrylong</code>	151	<code>\glsexpandfields</code>	66
<code>\glentrylongaccess</code>	330	<code>\glsfielddef</code>	73
<code>\Glsentrylongpl</code>	151	<code>\glsfieldedef</code>	72
<code>\glentrylongpl</code>	151	<code>\glsfieldfetch</code>	73
<code>\glentrylongpluralaccess</code>	330	<code>\glsfieldgdef</code>	73
<code>\Glsentryname</code>	146	<code>\glsfieldxdef</code>	72
<code>\glentryname</code>	146, 178	<code>\GLSfirst</code>	125
<code>\glentrynumberlist</code>	152, 168	<code>\Glsfirst</code>	125
<code>\Glsentryplural</code>	148	<code>\glsfirst</code>	125
<code>\glentryplural</code>	148	<code>\glsfirstaccessdisplay</code>	331
<code>\glentrypluralaccess</code>	329	<code>\GLSfirstplural</code>	127
<code>\Glsentryprefix</code>	253	<code>\Glsfirstplural</code>	127
<code>\glentryprefix</code>	252	<code>\glsfirstplural</code>	127
<code>\Glsentryprefixfirst</code>	252	<code>\glsfirstpluralaccessdisplay</code>	331
<code>\glentryprefixfirst</code>	252	<code>\glsgenacfmt</code>	99, 339
<code>\Glsentryprefixfirstplural</code>	252	<code>\glsgenentryfmt</code>	97, 337
<code>\glentryprefixfirstplural</code>	252	<code>\glsgetgrouplabel</code>	203
<code>\Glsentryprefixplural</code>	253	<code>\glsgetgrouptitle</code>	155, 202
<code>\glentryprefixplural</code>	252	<code>\glsglossarymark</code>	9, 38
<code>\glentryprevcount</code>	87	<code>\glsgroupheading</code>	202, 204
<code>\Glsentryshort</code>	151	<code>\glsgroupskip</code>	201, 204, 266
<code>\glentryshort</code>	151	<code>\glshyperlink</code>	153
<code>\glentryshortaccess</code>	330	<code>\glshypernavsep</code>	263
<code>\Glsentryshortpl</code>	151	<code>\glshypernumber</code>	36, 205
<code>\glentryshortpl</code>	151	<code>\glsifhyper</code>	104
<code>\glentryshortpluralaccess</code>	330	<code>\glsifhyperon</code>	104, 106
<code>\glentrysort</code>	149	<code>\glsIfListOfAcronyms</code>	15
<code>\Glsentrysymbol</code>	148	<code>\glsifusedtranslatordict</code>	22
<code>\glentrysymbol</code>	148	<code>\glsignore</code>	155
<code>\glentrysymbolaccess</code>	329	<code>\glsinlinedescformat</code>	265
<code>\Glsentrysymbolplural</code>	148	<code>\glsinlinedopostchild</code>	264, 265
<code>\glentrysymbolplural</code>	148	<code>\glsinlineemptydescformat</code>	265
<code>\glentrysymbolpluralaccess</code>	330	<code>\glsinlinenameformat</code>	265
<code>\Glsentrytext</code>	148	<code>\glsinlineparentchildseparator</code>	265
<code>\glentrytext</code>	50, 148, 178		
<code>\glentrytextaccess</code>	329	<code>\glsinlinepostchild</code>	265
<code>\glentrytype</code>	149	<code>\glsinlineseparator</code>	265
<code>\Glsentryuseri</code>	149	<code>\glsinlinesubdescformat</code>	265
<code>\glentryuseri</code>	149	<code>\glsinlinesubnameformat</code>	265
<code>\Glsentryuserii</code>	150	<code>\glsinlinesubseparator</code>	265
<code>\glentryuserii</code>	150	<code>\glskeylisttok</code>	212
<code>\Glsentryuseriii</code>	150	<code>\glslabeltok</code>	212
<code>\glentryuseriii</code>	150	<code>\glsletentryfield</code>	145
<code>\Glsentryuseriv</code>	150	<code>\glslink</code>	92, 105, 118, 153, 203, 205
<code>\glentryuseriv</code>	150	<code>\glslink options</code>	
<code>\Glsentryuserv</code>	150	<code>counter</code>	103, 118, 251
<code>\glentryuserv</code>	150	<code>format</code>	103, 118, 205
<code>\Glsentryuservi</code>	151	<code>hyper</code>	103, 106, 118
<code>\glentryuservi</code>	150	<code>local</code>	104

<code>\glslinkcheckfirsthyperhook</code>	106	<code>\glspercentchar</code>	156
<code>\glslinkpostsetkeys</code>	106	<code>\GLSpl</code>	122
<code>\glslinkvar</code>	104	<code>\Glspl</code>	121, 257
<code>\glslistdottedwidth</code>	268	<code>\glspl</code>	92, 120–122
<code>\glslocalreset</code>	84	<code>\GLSplural</code>	126
<code>\glslocalresetall</code>	86	<code>\Glsplural</code>	126
<code>\glslocalunset</code>	85	<code>\glsplural</code>	126
<code>\glslocalunsetall</code>	87	<code>\glspluralaccessdisplay</code>	331
<code>\glslongaccessdisplay</code>	332	<code>\glspluralsuffix</code>	31, 60
<code>\glslongaccesskey</code>	360	<code>\glspostdescription</code>	9
<code>\glslongkey</code>	209	<code>\glspostinline</code>	265
<code>\glslongpluralaccessdisplay</code>	332	<code>\glspostlinkhook</code>	105
<code>\glslongpluralaccesskey</code>	360	<code>\glsprestandardsort</code>	10
<code>\glslongpluralkey</code>	209	<code>\glsquote</code>	155
<code>\glslongtok</code>	213	<code>\glsrefentry</code>	197
<code>\glsmakefirstuc</code>	259	<code>\glsreset</code>	84
<code>\glsmcols</code>	281	<code>\glsresetall</code>	86
<code>\glsmoveentry</code>	82	<code>\glsresetentrylist</code>	196
<code>\GLSname</code>	128	<code>\glsresetsubentrycounter</code>	196, 197
<code>\Glsname</code>	128	<code>\glssee</code>	177
<code>\glsname</code>	128	<code>\glsseeformat</code>	158, 177
<code>\glsnameaccessdisplay</code>	331	<code>\glsseeitem</code>	178
<code>\glsnamefont</code>	205, 297	<code>\glsseeitemformat</code>	178
<code>\glsnavhyperlink</code>	261	<code>\glsseelastsep</code>	178
<code>\glsnavhypertarget</code>	261	<code>\glsseelist</code>	178
<code>\glsnavigation</code>	262	<code>\glsseesep</code>	178
<code>\glsnextpages</code>	196	<code>\glsSetAlphaCompositor</code>	35
<code>\glsnoexpandfields</code>	66	<code>\glsSetCompositor</code>	35
<code>\glsnoidxdisplayloc</code>	192	<code>\glssetexpandfield</code>	17
<code>\glsnoidxdisplaylocloclisthandler</code>	191	<code>\glssetnoexpandfield</code>	17
<code>\glsnoidxloclist</code>	191	<code>\glsSetSuffixF</code>	35
<code>\glsnoidxloclisthandler</code>	191	<code>\glsSetSuffixFF</code>	36
<code>\glsnoidxnumberlistloophandler</code>	169	<code>\glssettoctitle</code>	30
<code>\glsnoidxstripaccents</code>	19	<code>\glssetwidest</code>	302
<code>\glsnonextpages</code>	196	<code>\GlsSetXdyCodePage</code>	48
<code>\glsnoxindywarning</code>	40	<code>\GlsSetXdyFirstLetterAfterDigits</code>	156
<code>\glsnumberformat</code>	36	<code>\GlsSetXdyLanguage</code>	48
<code>\glsnumberlistloop</code>	169	<code>\GlsSetXdyLocationClassOrder</code>	47
<code>\glsnumbersgroupname</code>	31, 155, 202	<code>\GlsSetXdyMinRangeLength</code>	156
<code>\glsnumlistlastsep</code>	153	<code>\GlsSetXdyStyles</code>	48
<code>\glsnumlistsep</code>	153	<code>\glsshortaccessdisplay</code>	332
<code>\glsopenbrace</code>	155	<code>\glsshortaccesskey</code>	360
<code>\glsorder</code>	24	<code>\glsshortkey</code>	209
<code>\glsorg@endtheglossary</code>	5	<code>\glsshortpluralaccessdisplay</code>	332
<code>\glsorg@theglossary</code>	5	<code>\glsshortpluralaccesskey</code>	360
<code>\glspagelistwidth</code>	269, 275, 285, 292	<code>\glsshortpluralkey</code>	209
<code>\glspar</code>	34	<code>\glsshorttok</code>	212
		<code>\glssortnumberfmt</code>	11

<code>\glsspace</code>	210
<code>\glsstentry</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>	130, 131
<code>\glssymbolaccessdisplay</code>	331
<code>\glssymbolnav</code>	263
<code>\GLSsymbolplural</code>	132
<code>\Glsymbolplural</code>	131
<code>\glssymbolplural</code>	131, 132
<code>\glssymbolpluralaccessdisplay</code>	332
<code>\glssymbolsgroupname</code> .	31, 155, 202
<code>\glstarget</code>	198
<code>\GLStext</code>	124
<code>\Glstext</code>	125
<code>\glstext</code>	124
<code>\glstextaccessdisplay</code>	331
<code>\glstextformat</code>	93
<code>\glstextup</code>	209
<code>\glstildechar</code>	156
<code>\glstreeindent</code>	299–301
<code>\glstreenamefmt</code>	297
<code>\glssunset</code>	85
<code>\glssunsetall</code>	86
<code>\glsupacrpluralsuffix</code>	31
<code>\GlsUseAcrEntryDispStyle</code> ...	215
<code>\GlsUseAcrStyleDefs</code>	215
<code>\GLSuseri</code>	133
<code>\Glsuseri</code>	132
<code>\glssuseri</code>	132, 133
<code>\GLSuserii</code>	134
<code>\Glsuserii</code>	133
<code>\glssuserii</code>	133, 134
<code>\GLSuseriii</code>	134
<code>\Glsuseriii</code>	134
<code>\glssuseriii</code>	134
<code>\GLSuseriv</code>	135
<code>\Glsuseriv</code>	135
<code>\glssuseriv</code>	135
<code>\GLSuserv</code>	136
<code>\Glsuserv</code>	136
<code>\glssuserv</code>	136
<code>\GLSuservi</code>	137
<code>\Glsuservi</code>	137
<code>\glssuservi</code>	137

<code>\glswrite</code>	166
<code>\glswritedefhook</code>	75
<code>\glswriteentry</code>	173
<code>\gMFUnocap</code>	260

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>	5
<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>	74
<code>\ifgl@fieldeq</code>	74
<code>\ifgl@haschildren</code>	52
<code>\ifgl@hasdesc</code>	53
<code>\ifgl@hasfield</code>	54
<code>\ifgl@haslong</code>	53
<code>\ifgl@hasparent</code>	52
<code>\ifgl@hasprefix</code>	253
<code>\ifgl@hasprefixfirst</code>	253
<code>\ifgl@hasprefixfirstplural</code> .	253
<code>\ifgl@hasprefixplural</code>	253
<code>\ifgl@hasshort</code>	53
<code>\ifgl@hasymbol</code>	53
<code>\ifgl@translate</code>	21
<code>\ifgl@sused</code>	51, 84
<code>\ifgl@xindy</code>	25
<code>\ifignoredglossary</code>	59
<code>index (option)</code>	28
<code>index (style)</code>	297
<code>indexgroup (style)</code>	298
<code>indexhypergroup (style)</code>	299

indexonlyfirst (option) 23
\indexspace 266, 281, 297
inline (style) 263
\inputencodingname 25
\istfile 170
\istfilename 34
\item 205, 266, 297, 298

L

link text 93
list (style) 266
listdotted (style) 268
listgroup (style) 266
listhypergroup (style) 267
\loadglsentries 13, 92
long (key) 63
long (style) 269
long-sc-short (acrstyle) 218
long-sc-short-desc (acrstyle) ..
..... 219, 348
long-short (acrstyle) 215, 346
long-short-desc (acrstyle) . 218, 347
long-sm-short (acrstyle) 218
long-sm-short-desc (acrstyle) ..
..... 219, 348
long-sp-short (acrstyle) 216
long-sp-short-desc (acrstyle) .. 219
long3col (style) 271
long3colborder (style) 271
long3colheader (style) 272
long3colheaderborder (style) .. 272
long4col (style) 272
long4colborder (style) 273
long4colheader (style) 273
long4colheaderborder (style) .. 273
longaccess (key) 327
longborder (style) 270
longheader (style) 270
longheaderborder (style) 270
\longnewglossaryentry 60, 75
longplural (key) 63
longpluralaccess (key) 327
\longprovideglossaryentry ... 76
longragged (style) 275
longragged3col (style) 277
longragged3colborder (style) .. 278
longragged3colheader (style) .. 278
longragged3colheaderborder
(style) 278

longraggedborder (style) 276
longraggedheader (style) 276
longraggedheaderborder (style) 277
longtable (environment)
..... 8, 244, 269–280
longtable package 269, 275

M

\makefirstuc 257
makeglossaries
..... 24, 25, 34, 48, 56, 165, 182
\makeglossaries 26,
29, 30, 34, 35, 55, 57, 164, 166
\makeglossary 166
makeindex 363
makeindex 10, 25,
26, 31, 34–36, 56, 58, 60, 83,
109, 112, 155, 158, 160, 162,
171, 175, 176, 201, 202, 306, 307
delim_n 36
delim_r 36
page_compositor 35
special characters ... 110, 111, 155
makeindex (option) 25
\makenoidxglossaries 21, 166
mcolalttree (style) 283
mcolalttreegroup (style) 284
mcolalttreehypergroup (style) . 284
mcolindex (style) 281
mcolindexgroup (style) 281
mcolindexhypergroup (style) ... 281
mcoltree (style) 282
mcoltreegroup (style) 282
mcoltreehypergroup (style) 282
mcoltreenoname (style) 282
mcoltreenonamegroup (style) ... 283
mcoltreenonamehypergroup
(style) 283
memoir class 172
mfirstuc package 1, 260
\mfirstucMakeUppercase 259
\mfu@checkword 260
\MFUclear 260
\MFUnocap 260
multicol package 280
multicols (environment) 281

N

name (key) 59
\new@glossaryentry 67

`\newacronym` 24, 63, 92, 208, 209
`\newacronymhook` 213
`\newacronymstyle` 215
`\newglossary` 16, 56, 58, 162, 165, 192
`\newglossaryentry` .. 60, 66, 92, 208
`\newglossaryentry options`
 `access` 328, 329
 `counter` 61
 `description` 24, 59, 60,
 63, 66, 76, 128, 147, 209, 237, 326
 `descriptionaccess` 330, 332
 `descriptionplural` 129, 327
 `descriptionpluralaccess` .. 330, 332
 `first` 60,
 79, 118, 125, 148, 235, 240, 326
 `firstaccess` 329, 331
 `firstplural` 60, 127, 149, 326
 `firstpluralaccess` 329, 331
 `format` 157
 `long` 99, 151, 327
 `longaccess` 330, 332
 `longplural` 151, 327
 `longpluralaccess` 330, 332
 `name` 59,
 60, 63, 66, 76, 128, 146, 178, 326
 `nonumberlist` 62
 `parent` 62, 66
 `plural` 60, 79, 126, 326
 `pluralaccess` 329, 331
 `prefix` 252
 `prefixfirst` 252
 `prefixfirstplural` 252
 `prefixplural` 252
 `see` 8, 61, 165, 167
 `short` 99, 151, 327
 `shortaccess` 330, 332
 `shortplural` 151, 327
 `shortpluralaccess` 330, 332
 `sort` 60, 149, 201, 202
 `symbol` 59, 61, 130, 230–
 232, 235, 240, 272, 288, 326, 328
 `symbolaccess` 329, 331
 `symbolplural` 131, 326
 `symbolpluralaccess` 330, 332
 `text` 60, 118, 124, 147, 230, 235, 326
 `textaccess` 329, 331
 `type` 13, 61, 92, 149
 `user1` 132, 149, 327
 `user2` 133, 150

`user3` 134, 150
 `user4` 135, 150
 `user5` 136, 150
 `user6` 137, 150, 327
`\newglossarystyle` 204
`\newignoredglossary` 58
`nogroupskip (option)` 9
`nohypertypes (option)` 16
`\noist` 162, 251, 311
`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` 56
`numberedsection (option)` 6
`numberline (option)` 6
`numbers (option)` 27

O

`\oldacronym` 208
`order (option)` 25

P

`\p@gl@s@hyp@opt` 104
`package options:`
 `acronym` 13, 14, 30, 179, 208
 `true` 14
 `acronym` 13
 `acronymlists` 15
 `acronyms` 14
 `automake` 26
 `compatible-2.07` 27
 `compatible-3.07` 27
 `counter` 16
 `counter` 16
 `description` 234, 235
 `description` 24
 `dua` 233–235
 `dua` 24
 `entrycounter` 194, 196
 `true` 9

entrycounter	9
entrycounterwithin	9
footnote	118–123, 231, 233, 234, 237
footnote	24
hyperfirst	
false	118–123
hyperfirst	23
index	28
indexonlyfirst	370
indexonlyfirst	23
makeindex	158, 251
makeindex	25
nogroupskip	9
nohypertypes	16
nolist	244
nolist	8
nolong	244, 269
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	6
numberline	6
numbers	27
order	25
sanitize	20, 59, 146, 147
sanitize	21
sanitizesort	17
sanitizesort	20
savenuumberlist	7
savewrites	26, 368
false	162
true	164, 170
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	10
symbols	27
toc	5
true	6
toc	5
translate	22
false	22
translate	22
translator	21
ucmark	9
xindy	25, 26, 158, 251
xindy	25
xindygloss	26
xindynoglsnumbers	26
\pagelistname	31
parent (key)	62
\PGLS	256
\Pgls	255
\pgls	253
\PGLSpl	257
\Pglspl	255
\pglspl	254
\phantomsection	37–39
plural (key)	60
pluralaccess (key)	326
polyglossia package	22, 32
\printacronyms	14
\PrintChanges	5
\printglossaries	
.	12, 36, 55, 58, 166, 179, 180, 261
\printglossary	36,
.	37, 56, 58, 166, 179, 181, 192, 261
\printglossary options	
entrycounter	193
nogroupskip	193
nonumberlist	195
nopostdot	193
numberedsection	193
style	193

subentrycounter	194	\SetDescriptionFootnoteAcronymStyle	230
title	192	\SetDUADisplayStyle	241
toctitle	192	\SetDUASStyle	242
type	13, 179, 192	\setentrycounter	203
\printnoidxglossaries	180	\SetFootnoteAcronymDisplayStyle	235
\printnoidxglossary	180, 192	\SetFootnoteAcronymStyle ...	237
\printnoidxglossary options		\SetGenericNewAcronym	213
sort	195	\setglossarypreamble	37
\provideglossaryentry	66	\setglossarysection	38
\ProvidesGlossariesLang	31	\setglossarystyle	203
R			
\renewacronymstyle	215	\setglossentrycompatibility	200
\renewglossarystyle	204	\SetSmallAcronymDisplayStyle	238
\RequireGlossariesLang	31	\SetSmallAcronymStyle	240
\roman	44	\setStyleFile	34
S			
\s@gl@s@hyp@opt	104	\setupglossaries	28
\s@newglossary	56	short (key)	63
sanitize (option)	21	short-long (acrstyle)	217, 347
sanitizesort (option)	20	short-long-desc (acrstyle) .	220, 348
savenumberlist (option)	7	shortaccess (key)	327
savewrites (option)	26	shortplural (key)	63
sc-short-long (acrstyle)	218	shortpluralaccess (key)	327
sc-short-long-desc (acrstyle) .		shotcuts (option)	24
.....	220, 349	\showacronymlists	249
\scantokens	50	\showglocounter	246
\section	50	\showglodesc	247
section (option)	6	\showglodescaccess	361
see (key)	61	\showglodescplural	248
seeautonumberlist (option)	8	\showglodescpluralaccess ...	361
\seename	31	\showglofield	249
\SetAcronymLists	15	\showglofirst	246
\SetAcronymStyle	14, 242	\showglofirstaccess	360
\setacronymstyle	214	\showglofirstpl	246
\SetCustomDisplayStyle	243	\showglofirstpluralaccess ..	361
\SetCustomStyle	244	\showgloflag	249
\SetDefaultAcronymDisplayStyle		\showgloindex	249
.....	227	\showglolevel	245
\SetDefaultAcronymStyle	228	\showgloloclist	249
\SetDescriptionAcronymDisplayStyle		\showglolong	248
.....	233	\showglolongaccess	361
\SetDescriptionAcronymStyle	234	\showglolongpluralaccess ...	361
\SetDescriptionDUAAcronymDisplayStyle		\showgloname	247
.....	231	\showglonameaccess	360
\SetDescriptionDUAAcronymStyle		\showgloparent	245
.....	232	\showgloplural	245
\SetDescriptionFootnoteAcronymDisplayStyle		\showglopluralaccess	360
.....	229	\showgloshort	248
		\showgloshortaccess	361

W	
<code>\warn@nomakeglossaries</code>	164
<code>\warn@noprintglossary</code>	179
<code>\writeist</code> 34, 41, 43, 46, 156, 305, 307	
X	
<code>\xcapitalisewords</code>	260
<code>\xglaccsupp</code>	331
<code>xindy</code>	363
<code>xindy</code> 10, 25, 26, 34, 35, 40, 44, 46–	
	49, 83, 115, 116, 156, 158, 171,
	175, 176, 182, 201, 251, 306, 307
<code>xindy (option)</code>	25
<code>xindygloss (option)</code>	26
<code>xindynoglsnumbers (option)</code>	26
<code>\xmakefirstuc</code>	259
<code>xspace package</code>	4, 208