

Documented Code For glossaries v4.22

Nicola L.C. Talbot

Dickimaw Books

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

2016-04-19

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 v4.22: L^AT_EX2_ε Package to Assist Generating Glossaries”.

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

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

INSTALL Installation instructions.

CHANGES Change log.

README Package summary.

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

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

Contents

1	Main Package Code	4
1.1	Package Definition	4
1.2	Package Options	5
1.3	Predefined Text	29
1.4	Xindy	39
1.5	Loops and conditionals	48
1.6	Defining new glossaries	54
1.7	Defining new entries	58
1.8	Resetting and unsetting entry flags	83
1.9	Keeping Track of How Many Times an Entry Has Been Unset	86
1.10	Loading files containing glossary entries	90
1.11	Using glossary entries in the text	91
1.12	Adding an entry to the glossary without generating text	150
1.13	Creating associated files	152
1.14	Writing information to associated files	167
1.15	Glossary Entry Cross-References	173
1.16	Displaying the glossary	175
1.17	Acronyms	204
1.18	Predefined acronym styles	209
1.19	Predefined Glossary Styles	240
1.20	Debugging Commands	241
1.21	Compatibility with version 2.07 and below	246
2	Prefix Support (glossaries-prefix Code)	248
3	Glossary Styles	255
3.1	Glossary hyper-navigation definitions (glossary-hypernav package)	255
3.2	In-line Style (glossary-inline.sty)	257
3.3	List Style (glossary-list.sty)	259
3.4	Glossary Styles using longtable (the glossary-long package)	263
3.5	Glossary Styles using longtable and booktabs (the glossary-longbooktabs package)	269
3.6	Glossary Styles using longtable (the glossary-longragged package)	273
3.7	Glossary Styles using multicol (glossary-mcols.sty)	278
3.8	Glossary Styles using supertabular environment (glossary-super package)	284
3.9	Glossary Styles using supertabular environment (glossary-superragged package)	290
3.10	Tree Styles (glossary-tree.sty)	296

4 Backwards Compatibility	306
4.1 glossaries-compatible-207	306
4.2 glossaries-compatible-307	312
5 Accessibility Support (glossaries-accsupp Code)	326
5.1 Defining Replacement Text	327
5.2 Accessing Replacement Text	330
5.3 Displaying the Glossary	346
5.4 Acronyms	347
5.5 Debugging Commands	362
6 Multi-Lingual Support	364
6.1 Polyglossia Captions	364
Glossary	366
Change History	367
Index	389

1 Main Package Code

1.1 Package Definition

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

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries}[2016/04/19 v4.22 (NLCT)]
```

Required packages:

```
3 \RequirePackage{ifthen}
4 \RequirePackage{xkeyval}[2006/11/18]
5 \RequirePackage{mfirstuc}
```

The textcase package has much better case changing handling, so use `\MakeTextUppercase` instead of `\MakeUppercase`

```
6 \RequirePackage{textcase}
7 \renewcommand*{\mfirstucMakeUppercase}{\MakeTextUppercase}%
8 \RequirePackage{xfor}
```

```
9 \RequirePackage{datatool-base}
```

Need to use `\new@ifnextchar` instead of `\@ifnextchar` in commands that have a final optional argument (such as `\gls`) so require `.` Thanks to Morten Høgholm for suggesting this. (This has replaced using the `xspace` package.)

```
10 \RequirePackage{amsgen}
```

As from v3.0, now loading `etoolbox`:

```
11 \RequirePackage{etoolbox}
```

Check if doc has been loaded.

```
f@gls@docloaded
```

```
12 \newif\if@gls@docloaded
13 \@ifpackageloaded{doc}%
14 {%
15   \@gls@docloadedtrue
16 }%
17 {%
18   \@ifclassloaded{nlctdoc}{\@gls@docloadedtrue}{\@gls@docloadedfalse}%
19 }
20 \if@gls@docloaded
```

\doc has been loaded, so some modifications need to be made to ensure both packages can work together. The amount of conflict has been reduced as from v4.11 and no longer involves patching internal commands.

\PrintChanges needs to use doc's version of theglossary, so save that.

org@theglossary

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

@endtheglossary

```
22 \let\glsorg@endtheglossary\endtheglossary
```

\PrintChanges Now redefine \PrintChanges so that it uses the original theglossary environment.

```
23 \let\glsorg@PrintChanges\PrintChanges
24 \renewcommand{\PrintChanges}{%
25   \begingroup
26     \let\theglossary\glsorg@theglossary
27     \let\endtheglossary\glsorg@endtheglossary
28     \glsorg@PrintChanges
29   \endgroup
30 }
```

End of doc stuff.

```
31 \fi
```

1.2 Package Options

toc The toc package option will add the glossaries to the table of contents. This is a boolean key, if the value is omitted it is taken to be true.

```
32 \define@boolkey{glossaries.sty}[gls]{toc}[true]{}%
```

numberline The numberline package option adds \numberline to \addcontentsline. Note that this option only has an effect if used in with toc=true.

```
33 \define@boolkey{glossaries.sty}[gls]{numberline}[true]{}%
```

\@@glossarysec The sectional unit used to start the glossary is stored in \@@glossarysec. If chapters are defined, this is initialised to chapter, otherwise it is initialised to section.

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

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

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

s@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”:

aryentrynumbers

```
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
```

eautionumberlist

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

eautionumberlist

Automatically activates number list for entries containing the see key.

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

\@gls@loadlong

```
82 \newcommand*{\@gls@loadlong}{\RequirePackage{glossary-long}}
```

nolong

This option prevents from being loaded. This means that the glossary styles that use the `longtable` environment will not be available. This option is provided to reduce overhead caused by loading unrequired packages.

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

\@gls@loadsuper

The package isn't loaded if isn't installed.

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

nosuper This option prevents from being loaded. This means that the glossary styles that use the supertabular environment will not be available. This option is provided to reduce overhead caused by loading unrequired packages.

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

\@gls@loadlist

```
88 \newcommand*{\@gls@loadlist}{\RequirePackage{glossary-list}}
```

nolist This option prevents from being loaded (to reduce overheads if required). Naturally, the styles defined in will not be available if this option is used.

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

\@gls@loadtree

```
90 \newcommand*{\@gls@loadtree}{\RequirePackage{glossary-tree}}
```

notree This option prevents from being loaded (to reduce overheads if required). Naturally, the styles defined in will not be available if this option is used.

```
91 \@gls@declareoption{notree}{\renewcommand*{\@gls@loadtree}{}}
```

nostyles Provide an option to suppress all the predefined styles (in the event that the user has custom styles that are not dependent on the predefined styles).

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

postdescription The description terminator is given by `\glspostdescription` (except for the 3 and 4 column styles). This is a full stop by default. The spacefactor is adjusted in case the description ends with an upper case letter. (Patch provided by Michael Pock.)

```
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

rycounterwithin  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 }

s@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

efault@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 }

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

                  Allow user to hook into sort mechanism. The first argument <sort cs> is the temporary control
                  sequence containing the sort value before it has been sanitized and had makeindex/xindy
                  special characters escaped.
128 \newcommand*{\glsprestandardsort}[3]{%
129   \glsdosanitizesort
130 }

```

upsort@standard Set up the macros for default sorting.

```

131 \newcommand*{\@gls@setupsort@standard}{%
    Store entry information when it's defined.
132 \def\do@glo@storeentry{\@glo@storeentry}%
    No count register required for standard sort.
133 \def\@gls@defsortcount##1{%
    Sort according to sort key (\@glo@sort) if provided otherwise sort according to the entry's
    name (\@glo@name). (First argument glossary type, second argument entry label.)
134 \def\@gls@defsort##1##2{%
135 \ifx\@glo@sort\@glsdefaultsort
136 \let\@glo@sort\@glo@name
137 \fi
138 \let\glsdosanitizesort\@gls@sanitizesort
139 \glsprestandardsort{\@glo@sort}{##1}{##2}%
140 \expandafter\protected@xdef\csname glo@##2@sort\endcsname{\@glo@sort}%
141 }%
    Don't need to do anything when the entry is used.
142 \def\@gls@setsort##1{%
143 }
    Set standard sort as the default:
144 \@gls@setupsort@standard

```

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

```

145 \newcommand*{\glsortnumberfmt[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 }

```

s@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 }
```

s@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 [section 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 [section 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}%
    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 }

```

`@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}{}

```

dtoacronymlists

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

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

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

IfListOfAcronyms

<code>\glsIfListOfAcronyms{<label>}{<true part>}{<false part>}</code>

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 }
```

lsisacronymlist Convenient boolean.

```
273 \newif\if@glsisacronymlist
```

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

```
274 \newcommand*{\gls@ckisacronymlist}[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 [section 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}{}%
```

lareNoHyperList

```
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 }
```

ossariesWarning Prints a warning message.

```
301 \newcommand*{\GlossariesWarning}[1]{%
302   \PackageWarning{glossaries}{#1}%
303 }
```

esWarningNoLine 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 }
```

nonglossdefined Issue a warning if overriding \printglossary

```
311 \newcommand*\@gls@warnnonglossdefined{%
312   \GlossariesWarning{Overriding \string\printglossary}%
313 }
```

theglossdefined Issue a warning if overriding theglossary

```
314 \newcommand*\@gls@warnontheglossdefined{%
315   \GlossariesWarning{Overriding 'theglossary' environment}%
316 }
```

noredefwarn Suppress warning on redefinition of \printglossary

```
317 \@gls@declareoption{noredefwarn}{%
318   \renewcommand*\@gls@warnnonglossdefined}{}%
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.

ls@sanitizedesc

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

lssetexpandfield `\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 }
```

setnoexpandfield `\glssetnoexpandfield{<field>}`

Sets field to never expand.

```

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

```

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

```

333 \glsetexpandfield{type}

```

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

```

334 \glsetnoexpandfield{desc}

```

escplural@field

```

335 \glsetnoexpandfield{descplural}

```

ls@sanitizename

```

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

```

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

```

337 \glsetnoexpandfield{name}

```

@sanitizesymbol

```

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

```

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

```

339 \glsetnoexpandfield{symbol}

```

bolplural@field

```

340 \glsetnoexpandfield{symbolplural}

```

Sanitizing stuff:

ls@sanitizesort

```

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

```

ls@sanitizesort

```

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

```

@nosanitizesort

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

dx@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 }
```

@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 }
```

idxstripaccents

```
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     \glsssetnoexpandfield{sortvalue}%
439     \renewcommand*{\@gls@noidx@setsanitizesort}{%
440       \glssanitizesorttrue
441       \glsssetnoexpandfield{sortvalue}%
442     }%
443   \else
444     \glsssetexpandfield{sortvalue}%
445     \renewcommand*{\@gls@noidx@setsanitizesort}{%
446       \glssanitizesortfalse
447       \glsssetexpandfield{sortvalue}%
448     }%
449   \fi
450 }

```

Default setting:

```

451 \glssanitizesorttrue
452 \glsssetnoexpandfield{sortvalue}%

```

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

```

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

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

```

`sanitize`

```

467 \define@key{glossaries.sty}{sanitize}[description=true,symbol=true,name=true]{%
468   \ifthenelse{\equal{#1}{none}}{%
469     {%
470       \GlossariesWarning{sanitize package option deprecated}%
471       \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

notranslatorhook \@gls@notranslatorhook has been removed.

s@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 }

dtranslatordict Checks if given translator dictionary has been loaded.
500 \newcommand{\glsifusedtranslatordict}[3]{%
501 \glsifusetranslator
502 {\ifcsdef{ver@glossaries-dictionary-#1.dict}{#2}{#3}}%
503 {#3}%
504 }

notranslate Provide a synonym for translate=false that can be passed via the document class.
505 \@gls@declareoption{notranslate}{%
506 \glstranslatefalse
507 \let\@gls@usetranslator\relax
508 \let\glsifusetranslator\@secondoftwo
509 }

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

```

```

512 {%
513   \ifcase\nr\relax
514     \glstranslatetrue
515     \renewcommand*\@gls@usetranslator{%
516       \@ifpackageloaded{polyglossia}%
517       {%
518         \let\glsifusetranslator\@secondoftwo
519       }%
520       {%
521         \@ifpackageloaded{babel}%
522         {%
523           \IfFileExists{translator.sty}%
524           {%
525             \RequirePackage{translator}%
526             \let\glsifusetranslator\@firstoftwo
527           }%
528           {}%
529         }%
530         {}%
531       }%
532     }%
533   \or
534     \glstranslatefalse
535     \let\@gls@usetranslator\relax
536     \let\glsifusetranslator\@secondoftwo
537   \or
538     \glstranslatetrue
539     \let\@gls@usetranslator\relax
540     \let\glsifusetranslator\@secondoftwo
541   \fi
542 }

```

Set the default value:

```

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

```

```

559   }
560 }

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

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

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

footnote  Set the long form of the acronym in footnote on first use.
566 \define@boolkey{glossaries.sty}[glsacr]{footnote}[true]{}
567   \ifbool{glsacrdescription}%
568   {}%
569   {%
570     \renewcommand*{\@gls@sanitizedesc}{}%
571   }%
572   \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
573 }

description  Allow acronyms to have a description (needs to be set using the description key in the optional
              argument of \newacronym).
574 \define@boolkey{glossaries.sty}[glsacr]{description}[true]{}
575   \renewcommand*{\@gls@sanitizesymbol}{}%
576   \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
577 }

smallcaps  Define \newacronym to set the short form in small capitals.
578 \define@boolkey{glossaries.sty}[glsacr]{smallcaps}[true]{}
579   \renewcommand*{\@gls@sanitizesymbol}{}%
580   \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
581 }

smaller  Define \newacronym to set the short form using \smaller which obviously needs to be de-
          fined 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@glsnumberstrue`

`y@main@language` Define what language to use for each glossary type (if a language is not defined for a particular glossary type the language specified for the main glossary is used.)
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   \glsxindytrue
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=glslnumbers=false that can be passed via the document class options.

```
614 \@gls@declareoption{xindynoglsnumbers}{%
615   \glsxindytrue
616   \gls{xindy@glslnumbersfalse}
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*{\@gls@writefiles}{\@gls@writefiles}%
638   \else
639     \let\@gls@writefiles\@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][\@newterm]{%
684       \newglossaryentry{##2}{%
685         {type={index},name={##2},description={\nopostdesc},##1}%
686       }%
687     }%
```

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@option
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 }
```

`\ls@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 }
```

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

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

`\sable@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 @glotype@#1@title\endcsname}}
```

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

`\entryname`
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:

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

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

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

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

`acrpluralsuffix`
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.

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

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

ssarytocaptions Does nothing if translator hasn't been loaded.

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

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

```
777 \ifglstranslate
```

Load tracklang

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

Load translator if required.

```
779 \@gls@usetranslator
```

If using , \glossaryname should be defined in terms of \translate, but if babel is also loaded, it will redefine \glossaryname whenever the language is set, so override it. (Don't use \addto as doesn't define it.)

```
780 \@ifpackageloaded{translator}
```

```
781 {%
```

If the language options have been specified through the document class, then translator can pick them up. If not, translator will default to English and any language option passed to babel won't be detected, so if \trans@languages is just English and \bbl@loaded isn't simply english, then don't use the translator dictionaries.

```
782 \ifboolexpr
```

```
783 {
```

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

```
785 and not
```

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

```
787 }
```

```
788 {%
```

```
789 \let\glsifusetranslator\@secondoftwo
```

```
790 }%
```

```
791 {%
```

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

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

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

```
795 {%
```

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

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

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

```
799 }%
```

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

```
801 }%
```

```
802 }%
```

```
803 }%
```

```
804 }%
```

```
805 {}%
```

Check for tracked languages

```
806 \AnyTrackedLanguages
```

```

807 {%
808   \ForEachTrackedDialect{\this@dialect}{%
809     \IfTrackedLanguageFileExists{\this@dialect}%
810     {glossaries-}% prefix
811     {.ldf}%
812     {%
813       \RequireGlossariesLang{\CurrentTrackedTag}%
814     }%
815     {%
816       \PackageWarningNoLine{glossaries}%
817       {No language module detected for ‘\this@dialect’.\MessageBreak
818       Language modules need to be installed separately.\MessageBreak
819       Please check on CTAN for a bundle called\MessageBreak
820       ‘glossaries-\CurrentTrackedLanguage’ or similar}%
821     }%
822   }%
823 }%
824 {}%

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

```

`\nopostdesc` Provide a means to suppress description terminator for a given entry. (Useful for entries with no description.) Has no effect outside the glossaries.

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


`\@nopostdesc` Suppress next description terminator.

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

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

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

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

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

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

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

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

```
866 \@onlypremakeg\setStyleFile
```

The name of the makeindex or xindy style file is given by `\istfilename`. This file is created by `\writeist` (which is used by `\makeglossaries`) so redefining this command will only have an effect if it is done *before* `\makeglossaries`. As from v1.17, use `\setStyleFile` instead of directly redefining `\istfilename`.

`\istfilename`

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

`gls@istfilebase`

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

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

`\@istfilename`

```
873 \newcommand*{\@istfilename}[1]{}
```

This command is the value of the `page_compositor` `makeindex` key. Again, any redefinition of this command must take place *before* `\writeist` otherwise it will have no effect. As from 1.17, use `\glsSetCompositor` instead of directly redefining `\glscompositor`.

`\glscompositor`

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

`\glsSetCompositor` Sets the compositor.

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

Only use before `\makeglossaries`

```
877 \@onlypremakeg\glsSetCompositor
```

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

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

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

`\glsAlphaCompositor` Sets the alpha compositor.

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

Can only be used before `\makeglossaries`

```
886 \@onlypremakeg\glsSetAlphaCompositor
```

`\gls@suffixF` Suffix to use for a two page list. This overrides the separator and the closing page number if set to something other than an empty macro.

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

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

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

Only has an effect when used before `\makeglossaries`

```
890 \@onlypremakeg\glsSetSuffixF
```

`\gls@suffixFF` Suffix to use for a three page list. This overrides the separator and the closing page number if set to something other than an empty macro.

```
891 \newcommand*{\gls@suffixFF}{}

\glsSetSuffixFF Sets the suffix to use for a three page list.
892 \newcommand*{\glsSetSuffixFF}[1]{%
893   \renewcommand*{\gls@suffixFF}{#1}%
894 }

glsnumberformat The command \glsnumberformat indicates the default format for the page numbers in the glossary. (Note that this is not the same as \glossaryentrynumbers, but applies to individual numbers or groups of numbers within an entry's associated number list.) If hyperlinks are defined, it will use \glshypernumber, otherwise it will simply display its argument "as is".
895 \ifcsundef{hyperlink}%
896 {%
897   \newcommand*{\glsnumberformat}[1]{#1}%
898 }%
899 {%
900   \newcommand*{\glsnumberformat}[1]{\glshypernumber{#1}}%
901 }

Individual numbers in an entry's associated number list are delimited using \delimN (which corresponds to the delim_n makeindex keyword). The default value is a comma followed by a space.

\delimN
902 \newcommand{\delimN}{, }

A range of numbers within an entry's associated number list is delimited using \delimR (which corresponds to the delim_r makeindex keyword). The default is an en-dash.

\delimR
903 \newcommand{\delimR}{--}

The glossary preamble is given by \glossarypreamble. This will appear after the glossary sectioning command, and before the theglossary environment. It is designed to allow the user to add information pertaining to the glossary (e.g. "page numbers in italic indicate the primary definition") therefore \glossarypreamble shouldn't be affected by the glossary style. (So if you define your own glossary style, don't have it change \glossarypreamble.) The preamble is empty by default. If you have multiple glossaries, and you want a different preamble for each glossary, you will need to use \printglossary for each glossary type, instead of \printglossaries, and redefine \glossarypreamble before each \printglossary.

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

glossarypreamble `\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{\memUChead{#1}}{\memUChead{#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.

`\glossarysection`

```

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.

`glossarysection`

```

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

```

Do automatic labelling if required

```

984 \@glossaryseclabel
985 }

```

`gls@doclearpage` The `\gls@doclearpage` command is used to issue a `\clearpage` (or `\cleardoublepage`) depending on whether the glossary sectional unit is a chapter. If the sectional unit is something else, do nothing.

```

986 \newcommand*\gls@doclearpage{%
987 \ifthenelse{\equal{\@glossarysec}{chapter}}{%
988 {%
989 \ifcsundef{cleardoublepage}%
990 {%
991 \clearpage
992 }%
993 {%
994 \ifcsdef{if@openright}%
995 {%
996 \if@openright
997 \cleardoublepage
998 \else
999 \clearpage
1000 \fi
1001 }%
1002 {%
1003 \cleardoublepage

```

```

1004      }%
1005    }%
1006  }%
1007  {}%
1008 }

```

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

```

1009 \newcommand*\glsclearpage{\gls@doclearpage}

```

The glossary is added to the table of contents if `glstoc` flag set. If it is set, `\@gls@toc` will add a line to the `.toc` file, otherwise it will do nothing. (The first argument to `\@gls@toc` is the title for the table of contents, the second argument is the sectioning type.)

`\@gls@toc`

```

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

```

1.4 Xindy

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

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

```

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

```

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

```

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

```

`\glsdyattributelist` Comma-separated list of attributes.

```

1025 \ifglsxindy
1026   \edef\glsdyattributelist{}%
1027 \fi

```

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

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

`\@gls@ifinlist`

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

`sAddXdyCounters` Need to know all the counters that will be used in location numbers for Xindy. Argument may be a single counter name or a comma-separated list of counter names.

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

Only has an effect before \writeist:
1056   \@onlypremakeg\GlsAddXdyCounters
1057 \else
1058   \newcommand*\GlsAddXdyCounters[1]{%
1059     \glsnoxindywarning\GlsAddXdyAttribute
1060   }
1061 \fi
```

`saddxdycounters` 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 }

```

AddXdyAttribute Adds an attribute.

```

1068 \ifglxsindy

```

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

```

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

```

Add to xindy attribute list

```

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

```

Add to xindy markup location.

```

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

```

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

```

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

```

`\definedattributes` Add known attributes for all defined counters

```

1103 \ifglxindy
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{hyperterm}
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 \ifglxindy
1134 \newcommand*\GlsAddXdyAlphabet[2]{%
1135 \edef\xdyuseralphabets{%
1136 \xdyuseralphabets ^^J
1137 (define-alphabet "#1" (#2))}}

```

```

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

```

This code is only required for xindy:

```

1142 \ifglxindy

```

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

```

1143   \newcommand*{\@gls@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 `\@gls@xdy@Lclass@<name>`. Set up predefined formats.

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

```

1152   \protected@edef\@gls@roman{\@roman{0}\string"
1153     \string"roman-numbers-lowercase\string" :sep \string"}}%
1154   \@onelevel@sanitize\@gls@roman
1155   \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
1156     :sep \string"}%
1157   \@onelevel@sanitize\@tmp
1158   \ifx\@tmp\@gls@roman
1159     \expandafter
1160     \edef\csname @gls@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 }%
```

`\ix-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 }
```

`\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 }%
```

`\userlocationdefs` List of additional location definitions (separated by `^^J`)

```

1191 \def\xdyuserlocationdefs{}
```

`\userlocationnames` 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 \ifglsxindy
1195 \newcommand*\GlsAddXdyLocation[3][{}]{%
1196     \def\xdy@tmp{#1}%
1197     \ifx\xdy@tmp\xdy@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     \glsnnoxindywarning\GlsAddXdyLocation}
1222 \fi

```

ationclassorder Define location class order

```

1223 \ifglxsindy
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.

ationClassOrder

```

1236 \ifglxsindy
1237 \newcommand*\GlsSetXdyLocationClassOrder[1]{%
1238     \def\@xdylocationclassorder{#1}}
1239 \else
1240 \newcommand*\GlsSetXdyLocationClassOrder[1]{%

```

```

1241 \glsnoxywarning\GlsSetXdyLocationClassOrder}
1242 \fi

\@xdysortrules Define sort rules
1243 \ifglxindy
1244 \def\@xdysortrules{}
1245 \fi

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

xyrequiredstyles Define list of required styles (this should be a comma-separated list of xindy styles)
1256 \ifglxindy
1257 \def\@xdyrequiredstyles{tex}
1258 \fi

\GlsAddXdyStyle Add a xindy style to the list of required styles
1259 \ifglxindy
1260 \newcommand*\GlsAddXdyStyle[1]{%
1261 \edef\@xdyrequiredstyles{\@xdyrequiredstyles,#1}}%
1262 \else
1263 \newcommand*\GlsAddXdyStyle[1]{%
1264 \glsnoxywarning\GlsAddXdyStyle}
1265 \fi

GlsSetXdyStyles Reset the list of required styles
1266 \ifglxindy
1267 \newcommand*\GlsSetXdyStyles[1]{%
1268 \edef\@xdyrequiredstyles{#1}}
1269 \else
1270 \newcommand*\GlsSetXdyStyles[1]{%
1271 \glsnoxywarning\GlsSetXdyStyles}
1272 \fi

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

```

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

```

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

```

`\@gls@codepage` The xindy codepage setting is required by `makeglossaries`, so provide a command for `makeglossaries` to pick up the information from the auxiliary file. This command is not needed by the `glossaries` package, so define it to ignore its arguments.

```

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

```

`\sSetXdyCodePage` 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   \ifglxindy@glsnumbers
1303     \def\@xdylettergroups{(define-letter-group
1304       \string"glnumbers\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"\@glsfirstletter\string"))}
1310 \else
1311   \def\@xdylettergroups{}
1312 \fi
1313 \fi

```

sAddLetterGroup Add a new letter group. The first argument is the name of the letter group. The second argument is the xindy code specifying prefixes and ordering.

```

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#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 }

```

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

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

```


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

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

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

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

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

```
\if glossary exists {⟨type⟩} {⟨true-text⟩} {⟨false-text⟩}
```

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

```
1338 \newcommand{\if glossary exists}[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 re-defining `\glsdetoklabel` can cause things to go badly wrong, I'm not going to mention it in the main user guide. Only advanced users who know what they're doing ought to attempt it.

`\glsdetoklabel`

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

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

```
\if glossary entry exists {⟨label⟩} {⟨true text⟩} {⟨false text⟩}
```

where `⟨label⟩` is the entry's label.

```
1342 \newcommand{\if glossary entry exists}[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   \ifglsentryexists{#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   \ifglsentryexists{#1}{#2}{%
1361     \GlossariesWarning{Glossary entry '\glsdetoklabel{#1}'
1362       has not been defined}%
1363   }%
1364 }
```

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

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

```

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

```

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

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

```

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

```

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

```

1381 \newcommand{\ifglshaschildren}[3]{%
1382   \glsdoifexists{#1}%
1383   {%
1384     \def\do@glshaschildren{#3}%
1385     \edef\@gls@thislabel{\glsdetoklabel{#1}}%
1386     \expandafter\forglentries\expandafter
1387       [\csname glo@\@gls@thislabel @type\endcsname]
1388     {\glo@label}%
1389     {%
1390       \letcs\glo@parent{glo@\glo@label @parent}%
1391       \ifdefequal\@gls@thislabel\glo@parent
1392       {%
1393         \def\do@glshaschildren{#2}%
1394         \@endfortrue
1395       }%
1396     }%
1397   }%
1398   \do@glshaschildren
1399 }%
1400 }

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

1437 \newcommand*{\ifglshasshort}[3]{%

```

```

1438 \letcs{\@glo@short}{glo\glsdetoklabel{#1}@short}%
1439 \ifdefempty\@glo@short
1440   {#3}%
1441   {%
1442     \ifdefequal\@glo@short\@gls@default@value
1443     {#3}%
1444     {#2}%
1445   }%
1446 }

```

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

```

1447 \newcommand*{\ifglshasfield}[4]{%
1448   \glsdoifexists{#2}%
1449   {%
1450     \letcs{\@glo@thisvalue}{glo\glsdetoklabel{#2}@#1}%
      First check supplied field label is defined.
1451     \ifdef\@glo@thisvalue
1452     {%
      Is defined, so now check if empty.
1453       \ifdefempty\@glo@thisvalue
1454       {%
      Is empty, so doesn't have field set.
1455         #4%
1456       }%
1457     {%
      Not empty, so check if set to \@gls@default@value
1458       \ifdefequal\@glo@thisvalue\@gls@default@value{#4}{#3}%
1459     }%
1460   }%
1461   {%
      Field given isn't defined, so check if mapping exists.
1462     \@gls@fetchfield{\@gls@thisfield}{#1}%
      If \@gls@thisfield is defined, we've found a map. If not, the field supplied doesn't exist.
1463     \ifdef\@gls@thisfield
1464     {%
      Is defined, so now check if empty.
1465       \letcs{\@glo@thisvalue}{glo\glsdetoklabel{#2}@ \@gls@thisfield}%
1466       \ifdefempty\@glo@thisvalue
1467       {%
      Is empty so field hasn't been set.
1468         #4%
1469       }%
1470     {%

```

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

```
1471         \ifdefequal\@glo@thisvalue\@gls@default@value{#4}{#3}%
1472     }%
1473 }%
1474 {%
    Not defined.
1475     \GlossariesWarning{Unknown entry field '#1'}%
1476     #4%
1477 }%
1478 }%
1479 }%
1480 }
```

1.6 Defining new glossaries

A comma-separated list of glossary names is stored in \@glo@types. When a new glossary type is created, its identifying name is added to this list. This is used by commands that iterate through all glossaries (such as \makeglossaries and \printglossaries).

\@glo@types

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

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

```
1482 \newcommand*\@gls@provide@newglossary{%
1483     \protected@write\@auxout{}\string\providecommand\string\@newglossary[4]{}%
    Only need to do this once.
1484     \let\@gls@provide@newglossary\relax
1485 }
```

\defglsentryfmt Allow different glossaries to have different display styles.

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

\gls@doentryfmt

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

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

```
1490 \newcommand*\@gls@forbidtexext}[1]{%
1491     \ifboolexpr{test {\ifdefstring{#1}{tex}}
1492         or test {\ifdefstring{#1}{TEX}}}
1493     {%
1494         \def#1{nottex}%
    }
```

```

1495 \PackageError{glossaries}%
1496   {Forbidden ‘.tex’ extension replaced with ‘.nottex’}%
1497   {I’m sorry, I can’t allow you to do something so reckless.\MessageBreak
1498     Don’t use ‘.tex’ as an extension for a temporary file.}%
1499 }%
1500 {%
1501 }%
1502 }

```

`\gls@gobbleopt` Discard optional argument.

```

1503 \newcommand*{\gls@gobbleopt}{\new@ifnextchar[{\@gls@gobbleopt}{}]}
1504 \def\@gls@gobbleopt[#1]{%

```

A new glossary type is defined using `\newglossary`. Syntax:

```
\newglossary[⟨log-ext⟩]{⟨name⟩}{⟨in-ext⟩}{⟨out-ext⟩} {⟨title⟩}[⟨counter⟩]
```

where `⟨log-ext⟩` is the extension of the makeindex transcript file, `⟨in-ext⟩` is the extension of the glossary input file (read in by `\printglossary` and created by makeindex), `⟨out-ext⟩` is the extension of the glossary output file which is read in by makeindex (lines are written to this file by the `\glossary` command), `⟨title⟩` is the title of the glossary that is used in `\glossarysection` and `⟨counter⟩` is the default counter to be used by entries belonging to this glossary. The `makeglossaries` Perl script reads in the relevant extensions from the auxiliary file, and passes the appropriate file names and switches to makeindex.

`\newglossary`

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

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

```

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

```

`\ns@newglossary` Define the unstarred version.

```

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

```

Check if default has been set

```

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

```

Add this to the list of glossary types:

```
1516   \toks@{#2}\edef\@glo@types{\@glo@types\the\toks@,}%

```

Define a comma-separated list of labels for this glossary type, so that all the entries for this glossary can be reset with a single command. When a new entry is created, its label is added to this list.

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

Store the file extensions:

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

Store the title:

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

1525 \@gls@provide@newglossary
1526 \protected@write\auxout{}{\string\@newglossary{#2}{#1}{#3}{#4}}%
```

How to display this entry in the document text (uses `\glsentry` by default). This can be re-defined by the user later if required (see `\defglsentry`). This may already have been defined if this has been specified as a list of acronyms.

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

Define sort counter if required:

```
1532 \@gls@defsortcount{#2}%
```

Find out if the final optional argument has been specified, and use it to set the counter associated with this glossary. (Uses `\glscounter` if no optional argument is present.)

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

`\altnewglossary`

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

Only define new glossaries in the preamble:

```
1543 \@onlypreamble{\newglossary}
```

Only define new glossaries before `\makeglossaries`

```
1544 \@onlypremakeg\newglossary
```


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

`\@newglossary`

```
1545 \newcommand*{\@newglossary}[4]{}

```

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

`@gls@setcounter`

```
1546 \def\@gls@setcounter#1[#2]{%
1547   \expandafter\def\csname @glotype@#1@counter\endcsname{#2}%

```

Add counter to xindy list, if not already added:

```
1548   \ifglsxindy
1549     \GlsAddXdyCounters{#2}%
1550   \fi
1551 }

```

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

`@gls@getcounter`

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

```

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

```
1555 \glsdefmain

```

Define the “acronym” glossaries if required.

```
1556 \@gls@do@acronymsdef

```

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

```
1557 \@gls@do@symbolsdef
1558 \@gls@do@numbersdef
1559 \@gls@do@indexdef

```

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

```
1560 \newcommand*{\newignoredglossary}[1]{%
1561   \ifdefempty\@ignored@glossaries
1562   {%
1563     \edef\@ignored@glossaries{#1}%
1564   }%
1565   {%
1566     \eappto\@ignored@glossaries{,#1}%

```

```

1567 }%
1568 \csgdef{glolist@#1}{,}%
1569 \ifcsundef{gls@#1@entryfmt}%
1570 {%
1571   \defglsentryfmt[#1]{\glsentryfmt}%
1572 }%
1573 {}%
1574 \ifdefempty\@gls@nohyperlist
1575 {%
1576   \renewcommand*{\@gls@nohyperlist}{#1}%
1577 }%
1578 {%
1579   \eappto\@gls@nohyperlist{, #1}%
1580 }%
1581 }

```

`ignored@glossaries` List of ignored glossaries.

```

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

```

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

```

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

```

1.7 Defining new entries

New glossary entries are defined using `\newglossaryentry`. This command requires a label and a key-value list that defines the relevant information for that entry. The definition for these keys follows. Note that the name, description and symbol keys will be sanitized later, depending on the value of the package option `sanitize` (this means that if some of the keys haven't been defined, they can be constructed from the name and description key before they are sanitized).

name The name key indicates the name of the term being defined. This is how the term will appear in the glossary. The name key is required when defining a new glossary entry.

```

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

```

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

```

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

```

descriptionplural

```

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

```

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

```

1597 \define@key{glossentry}{sort}{%
1598 \def\@glo@sort{#1}}

```

text The text key determines how the term should appear when used in the document (i.e. outside of the glossary). If omitted, the value of the name key is used instead.

```

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

```

plural The plural key determines how the plural form of the term should be displayed in the document. If omitted, the plural is constructed by appending `\glspluralsuffix` to the value of the text key.

```

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

```

first The first key determines how the entry should be displayed in the document when it is first used. If omitted, it is taken to be the same as the value of the text key.

```

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

```

firstplural The firstplural key is used to set the plural form for first use, in the event that the plural is required the first time the term is used. If omitted, it is constructed by appending `\glspluralsuffix` to the value of the first key.

```

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

```

s@default@value

```

1611 \newcommand*{\@gls@default@value}{\relax}

```

symbol The symbol key is ignored by most of the predefined glossary styles, and defaults to `\relax` if omitted. It is provided for glossary styles that require an associated symbol, as well as a name and description. To make this value appear in the glossary, you need to redefine

`\glossentry`. If you want this value to appear in the text when the term is used by commands like `\gls`, you will need to change `\glsentryfmt` (or use for `\defglsentryfmt` individual glossaries).

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

`symbolplural`

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

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

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

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

```
1620 \define@key{glossentry}{counter}{%
1621   \ifcsundef{c@#1}%
1622   {%
1623     \PackageError{glossaries}%
1624     {There is no counter called ‘#1’}%
1625     {%
1626       The counter key should have the name of a valid counter
1627       as its value%
1628     }%
1629   }%
1630   {%
1631     \def\@glo@counter{#1}%
1632   }%
1633 }
```

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

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

`checkseeallowed`

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

ed@preambleonly

```
1647 \newcommand*{\gls@checkseeallowed@preambleonly}{%
1648   \GlossariesWarning{glossaries}%
1649   {'see' key doesn't have any effect when used in the document
1650   environment. Move the definition to the preamble
1651   after \string\makeglossaries\space
1652   or \string\makenoidxglossaries}%
1653 }
```

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

```
1654 \define@key{glossentry}{parent}{%
1655 \def\@glo@parent{#1}}
```

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

```
1656 \define@choicekey{glossentry}{nonumberlist}[\val\nr]{true,false}[true]{%
1657   \ifcase\nr\relax
1658     \def\@glo@prefix{\glsnonextpages}%
1659   \else
1660     \def\@glo@prefix{\glsnextpages}%
1661   \fi
1662 }
```

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

user1

```
1663 \define@key{glossentry}{user1}{%
1664   \def\@glo@useri{#1}%
1665 }
```

user2

```
1666 \define@key{glossentry}{user2}{%
1667   \def\@glo@userii{#1}%
1668 }
```

user3

```
1669 \define@key{glossentry}{user3}{%
1670   \def\@glo@useriii{#1}%
1671 }
```

user4

```
1672 \define@key{glossentry}{user4}{%
1673   \def\@glo@useriv{#1}%
1674 }
```

user5

```
1675 \define@key{glossentry}{user5}{%
1676   \def\@glo@userv{#1}%
1677 }
```

user6

```
1678 \define@key{glossentry}{user6}{%
1679   \def\@glo@uservi{#1}%
1680 }
```

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.

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

shortplural This key is provided for use by \newacronym.

```
1684 \define@key{glossentry}{shortplural}{%
1685   \def\@glo@shortpl{#1}%
1686 }
```

long This key is provided for use by \newacronym.

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

longplural This key is provided for use by \newacronym.

```
1690 \define@key{glossentry}{longplural}{%
1691   \def\@glo@longpl{#1}%
1692 }
```

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

```
1693 \newcommand*{\@glsnname}{%
1694   \PackageError{glossaries}{name key required in
1695     \string\newglossaryentry\space for entry '\@glo@label'}{You
1696     haven't specified the entry name}}
```

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

```
1697 \newcommand*{\@glsnodelsc}{%
1698   \PackageError{glossaries}
1699   {%
1700     description key required in \string\newglossaryentry\space
1701     for entry '\@glo@label'%
1702   }%
1703   {%
1704     You haven't specified the entry description%
1705   }%
1706 }
```

lsdefaultplural Now obsolete. Don't use.

```
1707 \newcommand*{\@glsdefaultplural}{}
```

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

```
1708 \newcommand*{\@gls@missingnumberlist}[1]{%
1709   ??%
1710   \ifglssavenumberlist
1711     \GlossariesWarning{Missing number list for entry ‘#1’.
1712       Maybe makeglossaries + rerun required.}%
1713   \else
1714     \PackageError{glossaries}%
1715       {Package option ‘savenumberlist=true’ required.}%
1716     {%
1717       You must use the ‘savenumberlist’ package option
1718       to reference location lists.%
1719     }%
1720   \fi
1721 }
```

@glsdefaultsort Define command to set default sort.

```
1722 \newcommand*{\@glsdefaultsort}{\@glo@name}
```

\gls@level Register to increment entry levels.

```
1723 \newcount\gls@level
```

@noexpand@field

```
1724 \newcommand{\@@gls@noexpand@field}[3]{%
1725   \expandafter\global\expandafter
1726     \let\csname glo@#1@#2\endcsname#3%
1727 }
```

noexpand@fields

```
1728 \newcommand{\@gls@noexpand@fields}[4]{%
1729   \ifcsdef{gls@assign@#3@field}
1730     {%
1731       \ifdefequal{#4}{\@gls@default@value}%
1732       {%
1733         \edef\@gls@value{\expandonce{#1}}%
1734         \csuse{gls@assign@#3@field}{#2}{\@gls@value}%
1735       }%
1736     }%
1737     \csuse{gls@assign@#3@field}{#2}{#4}%
1738   }%
1739 }%
1740 {%
1741   \ifdefequal{#4}{\@gls@default@value}%
1742   {%
1743     \edef\@gls@value{\expandonce{#1}}%
1744     \@@gls@noexpand@field{#2}{#3}{\@gls@value}%
1745   }%
1746   {%
```

```

1747      \@@gls@noexpand@field{#2}{#3}{#4}%
1748    }%
1749  }%
1750 }

```

ls@expand@field

```

1751 \newcommand{\@@gls@expand@field}[3]{%
1752   \expandafter
1753     \protected@xdef\csname glo@#1@#2\endcsname{#3}%
1754 }

```

s@expand@fields

```

1755 \newcommand{\@gls@expand@fields}[4]{%
1756   \ifcsdef{gls@assign@#3@field}
1757   {%
1758     \ifdefequal{#4}{\@gls@default@value}%
1759     {%
1760       \edef\@gls@value{\expandonce{#1}}%
1761       \csuse{gls@assign@#3@field}{#2}{\@gls@value}%
1762     }%
1763   }%
1764   \expandafter\@gls@startswithexpandonce#4\relax\relax@gls@endcheck
1765   {%
1766     \@@gls@expand@field{#2}{#3}{#4}%
1767   }%
1768   {%
1769     \csuse{gls@assign@#3@field}{#2}{#4}%
1770   }%
1771 }%
1772 }%
1773 {%
1774   \ifdefequal{#4}{\@gls@default@value}%
1775   {%
1776     \@@gls@expand@field{#2}{#3}{#1}%
1777   }%
1778   {%
1779     \@@gls@expand@field{#2}{#3}{#4}%
1780   }%
1781 }%
1782 }

```

swithexpandonce

```

1783 \def\@gls@expandonce{\expandonce}
1784 \def\@gls@startswithexpandonce#1#2@gls@endcheck#3#4{%
1785   \def\@gls@tmp{#1}%
1786   \ifdefequal{\@gls@expandonce}{\@gls@tmp}{#3}{#4}%
1787 }

```


`gls@assign@field` `\gls@assign@field{<def value>}{<label>}{<field>}{<tmp cs>}`

Assigns an entry field. Expansion performed by default (except for name, symbol and description where backward compatibility required). If *<tmp cs>* is *<gls@default@value>*, *<def value>* is used instead.

```
1788 \let\gls@assign@field\@gls@expand@fields
```

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

```
1789 \newcommand*{\gls@expand@fields}{%
1790   \let\gls@assign@field\@gls@expand@fields
1791 }
```

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

```
1792 \newcommand*{\gls@snoexpand@fields}{%
1793   \let\gls@assign@field\@gls@noexpand@fields
1794 }
```

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

```
1795 \newrobustcmd{\newglossaryentry}[2]{%
```

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

```
1796   \glsdoifnoexists{#1}%
1797   {%
1798     \gls@defglossaryentry{#1}{#2}%
1799   }%
1800 }
```

`newglossaryentry` The definition of `\newglossaryentry` is changed at the start of the document environment. The see key doesn't work for entries that have been defined in the document environment.

```
1801 \newcommand*{\gls@defdocnewglossaryentry}{%
1802   \let\gls@checkseeallowed\gls@checkseeallowed@preambleonly
1803   \let\newglossaryentry\new@glossaryentry
1804 }
```

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

```
1805 \newrobustcmd{\provideglossaryentry}[2]{%
1806   \ifglsentryexists{#1}%
1807   {%
1808     {%
1809       \gls@defglossaryentry{#1}{#2}%
1810     }%
1811 }
1812 \@onlypreamble{\provideglossaryentry}
```

w@glossaryentry For use in document environment.

```
1813 \newrobustcmd{\new@glossaryentry}[2]{%
1814   \ifundef\@gls@deffile
1815   {%
1816     \global\newwrite\@gls@deffile
1817     \immediate\openout\@gls@deffile=\jobname.glsdefs
1818   }%
1819   {}%
1820   \ifglentryexists{#1}{}%
1821   {%
1822     \gls@defglossaryentry{#1}{#2}%
1823   }%
1824   \@gls@writedef{#1}%
1825 }
1826 \AtBeginDocument
1827 {
1828   \makeatletter
1829   \InputIfFileExists{\jobname.glsdefs}{\}{}%
1830   \makeatother
1831   \gls@defdocnewglossaryentry
1832 }
1833 \AtEndDocument{\ifdef\@gls@deffile{\closeout\@gls@deffile}{}}
```

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

```
1834 \newcommand*{\@gls@writedef}[1]{%
1835   \immediate\write\@gls@deffile
1836   {%
1837     \string\ifglentryexists{#1}{}\glspercentchar^^J%
1838     \expandafter\@gobble\string{\glspercentchar^^J%
1839       \string\gls@defglossaryentry{\glsdetoklabel{#1}}\glspercentchar^^J%
1840       \expandafter\@gobble\string{\glspercentchar%
1841     }%
1842   }
```

Write key value information:

```
1842   \@for\@gls@map:=\@gls@keymap\do
1843   {%
1844     \edef\glo@value{\expandafter\expandonce
1845       \csname glo@\glsdetoklabel{#1}\@expandafter
1846       \@secondoftwo\@gls@map\endcsname}%
1847     \@onelevel@sanitize\glo@value
1848     \immediate\write\@gls@deffile
1849     {%
1850       \expandafter\@firstoftwo\@gls@map
1851       =\expandafter\@gobble\string{\glo@value\expandafter\@gobble\string\},%
1852       \glspercentchar%
1853     }%
1854   }
```

Provide hook:

```
1855   \glswritedefhook
```

```

1856 \immediate\write\@gls@deffile
1857 {%
1858     \glspercentchar^^J%
1859     \expandafter\@gobble\string\}\glspercentchar^^J%
1860     \expandafter\@gobble\string\}\glspercentchar%
1861 }%
1862 }

```

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

```

1863 \newcommand*{\@gls@keymap}{%
1864 {name}{name},%
1865 {sort}{sortvalue},% unescaped sort value
1866 {type}{type},%
1867 {first}{first},%
1868 {firstplural}{firstpl},%
1869 {text}{text},%
1870 {plural}{plural},%
1871 {description}{desc},%
1872 {descriptionplural}{descplural},%
1873 {symbol}{symbol},%
1874 {symbolplural}{symbolplural},%
1875 {user1}{useri},%
1876 {user2}{userii},%
1877 {user3}{useriii},%
1878 {user4}{useriv},%
1879 {user5}{userv},%
1880 {user6}{uservi},%
1881 {long}{long},%
1882 {longplural}{longpl},%
1883 {short}{short},%
1884 {shortplural}{shortpl},%
1885 {counter}{counter},%
1886 {parent}{parent}%
1887 }

```

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

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

```

1888 \newcommand*{\@gls@fetchfield}[2]{%

```

Ensure user field name is fully expanded

```

1889 \edef\@gls@thisval{#2}%

```

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

```

1890 \@for\@gls@map:=\@gls@keymap\do{%
1891 \edef\@this@key{\expandafter\@firstoftwo\@gls@map}%
1892 \ifdefequal{\@this@key}{\@gls@thisval}%
1893 {%

```

Found it.

```
1894 \edef#1{\expandafter\@secondoftwo\@gls@map}%
```

Break out of loop.

```
1895 \endfortrue
1896 }%
1897 {}%
1898 }%
1899 }
```

```
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.

```
1900 \newcommand*{\glsaddstoragekey}{\ifstar\@sglsaddstoragekey\@glsaddstoragekey}
```

Starred version switches on expansion for this key.

```
1901 \newcommand*{\@sglsaddstoragekey}[1]{%
1902 \key@ifundefined{glossentry}{#1}%
1903 {%
1904 \expandafter\newcommand\expandafter*\expandafter
1905 {\csname gls@assign@#1@field@endcsname}[2]{%
1906 \@@gls@expand@field{##1}{#1}{##2}%
1907 }%
1908 }%
1909 {}%
1910 \@glsaddstoragekey{#1}%
1911 }
```

Unstarred version doesn't override default expansion.

```
1912 \newcommand*{\@glsaddstoragekey}[3]{%
```

Check the specified key doesn't already exist.

```
1913 \key@ifundefined{glossentry}{#1}%
1914 {%
```

Set up the key.

```
1915 \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
1916 \appto\@gls@keymap{,{#1}{#1}}%
```

Set the default value.

```
1917 \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
```

Assignment code.

```
1918 \appto\@newglossaryentryposthook{%
1919 \letcs{\@glo@tmp}{@glo@#1}%
1920 \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
1921 }%
```

Define the no-link commands.

```

1922 \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
1923 }%
1924 {%
1925 \PackageError{glossaries}{Key ‘#1’ already exists}{}%
1926 }%
1927 }

```

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

Allow user to add their own custom keys.

```

1928 \newcommand*{\glsaddkey}{\@ifstar\@sglsaddkey\@glsaddkey}

```

Starred version switches on expansion for this key.

```

1929 \newcommand*{\@sglsaddkey}[1]{%
1930 \key@ifundefined{glossentry}{#1}%
1931 {%
1932 \expandafter\newcommand\expandafter*\expandafter
1933 {\csname gls@assign@#1@field\endcsname}[2]{%
1934 \@gls@expand@field{##1}{#1}{##2}%
1935 }%
1936 }%
1937 }%
1938 \@glsaddkey{#1}%
1939 }

```

Unstarred version doesn't override default expansion.

```

1940 \newcommand*{\@glsaddkey}[7]{%

```

Check the specified key doesn't already exist.

```

1941 \key@ifundefined{glossentry}{#1}%
1942 {%

```

Set up the key.

```

1943 \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
1944 \appto\@gls@keymap{, {#1}{#1}}%

```

Set the default value.

```

1945 \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%

```

Assignment code.

```

1946 \appto\@newglossaryentryposthook{%
1947 \letcs{\@glo@tmp}{@glo@#1}%
1948 \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
1949 }%

```

Define the no-link commands.

```

1950 \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
1951 \newcommand*{#4}[1]{\@Gls@entry@field{##1}{#1}}%

```

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

```

1952 \ifcsdef{@gls@user@#1@}%
1953 {%
1954   \PackageError{glossaries}%
1955   {Can't define '\string#5' as helper command
1956   '\expandafter\string\csname @gls@user@#1@\endcsname' already exists}%
1957   }%
1958 }%
1959 {%
1960   \expandafter\newcommand\expandafter*\expandafter
1961   {\csname @gls@user@#1@\endcsname}[2][ ]{%
1962     \new@ifnextchar[%
1963       {\csuse{@gls@user@#1@}{##1}{##2}}%
1964       {\csuse{@gls@user@#1@}{##1}{##2}[ ]}}%
1965   \csdef{@gls@user@#1@}##1##2[##3]{%
1966     \@gls@field@link{##1}{##2}{#3{##2}##3}%
1967   }%
1968   \newrobustcmd*{#5}{%
1969     \expandafter\@gls@hyp@opt\csname @gls@user@#1@\endcsname}%
1970   }%

```

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

```

1971 \ifcsdef{@Gls@user@#1@}%
1972 {%
1973   \PackageError{glossaries}%
1974   {Can't define '\string#6' as helper command
1975   '\expandafter\string\csname @Gls@user@#1@\endcsname' already exists}%
1976   }%
1977 }%
1978 {%
1979   \expandafter\newcommand\expandafter*\expandafter
1980   {\csname @Gls@user@#1@\endcsname}[2][ ]{%
1981     \new@ifnextchar[%
1982       {\csuse{@Gls@user@#1@}{##1}{##2}}%
1983       {\csuse{@Gls@user@#1@}{##1}{##2}[ ]}}%
1984   \csdef{@Gls@user@#1@}##1##2[##3]{%
1985     \@gls@field@link{##1}{##2}{#4{##2}##3}%
1986   }%
1987   \newrobustcmd*{#6}{%
1988     \expandafter\@gls@hyp@opt\csname @Gls@user@#1@\endcsname}%
1989   }%

```

Finally the all caps version:

```

1990 \ifcsdef{@GLS@user@#1@}%
1991 {%
1992   \PackageError{glossaries}%
1993   {Can't define '\string#7' as helper command
1994   '\expandafter\string\csname @GLS@user@#1@\endcsname' already exists}%

```

```

1995     {}%
1996   }%
1997   {%

1998     \expandafter\newcommand\expandafter*\expandafter
1999     {\csname @GLS@user@#1\endcsname}[2] [] {%
2000       \new@ifnextchar[%
2001         {\csuse{@GLS@user@#1@}{##1}{##2}}%
2002         {\csuse{@GLS@user@#1@}{##1}{##2} []}}%
2003     \csdef{@GLS@user@#1@}##1##2[##3]{%
2004       \@gls@field@link{##1}{##2}{\mfirstucMakeUppercase{#3{##2}##3}}%
2005     }%
2006     \newrobustcmd*{#7}{%
2007       \expandafter\@gls@hyp@opt\csname @GLS@user@#1\endcsname}%
2008   }%
2009 }%
2010 {%
2011   \PackageError{glossaries}{Key ‘#1’ already exists}{}%
2012 }%
2013 }

```

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

```

2014 \newcommand{\glsfieldxdef}[3]{%
2015   \glsdoifexists{#1}%
2016   {%
2017     \edef\@glo@label{\glsdetoklabel{#1}}%
2018     \ifcsdef{glo@\@glo@label @#2}%
2019     {%
2020       \expandafter\xdef\csname glo@\@glo@label @#2\endcsname{#3}%
2021     }%
2022     {%
2023       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2024     }%
2025   }%
2026 }

```

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

```

2027 \newcommand{\glsfielddedef}[3]{%
2028   \glsdoifexists{#1}%
2029   {%
2030     \edef\@glo@label{\glsdetoklabel{#1}}%
2031     \ifcsdef{glo@\@glo@label @#2}%
2032     {%

```

```

2033     \expandafter\edef\csname glo@\@glo@label @#2\endcsname{#3}%
2034 }%
2035 {%
2036     \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2037 }%
2038 }%
2039 }

```

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

```

2040 \newcommand{\glsfieldgdef}[3]{%
2041   \glsdoifexists{#1}%
2042   {%
2043     \edef\@glo@label{\glsdetoklabel{#1}}%
2044     \ifcsdef{glo@\@glo@label @#2}%
2045     {%
2046       \expandafter\gdef\csname glo@\@glo@label @#2\endcsname{#3}%
2047     }%
2048     {%
2049       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2050     }%
2051   }%
2052 }

```

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

```

2053 \newcommand{\glsfieldddef}[3]{%
2054   \glsdoifexists{#1}%
2055   {%
2056     \edef\@glo@label{\glsdetoklabel{#1}}%
2057     \ifcsdef{glo@\@glo@label @#2}%
2058     {%
2059       \expandafter\def\csname glo@\@glo@label @#2\endcsname{#3}%
2060     }%
2061     {%
2062       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2063     }%
2064   }%
2065 }

```

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

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


```

2066 \newcommand{\glsfieldfetch}[3]{%
2067   \glsdoifexists{#1}%
2068   {%
2069     \edef\@glo@label{\glsdetoklabel{#1}}%
2070     \ifcsdef{glo@\@glo@label @#2}%
2071     {%
2072       \letcs#3{glo@\@glo@label @#2}%
2073     }%
2074     {%
2075       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2076     }%
2077   }%
2078 }

```

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

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

```

2079 \newcommand{\ifglsfieldeq}[5]{%
2080   \glsdoifexists{#1}%
2081   {%
2082     \edef\@glo@label{\glsdetoklabel{#1}}%
2083     \ifcsdef{glo@\@glo@label @#2}%
2084     {%
2085       \ifcsstring{glo@\@glo@label @#2}{#3}{#4}{#5}%
2086     }%
2087     {%
2088       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2089     }%
2090   }%
2091 }

```

`\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.

```

2092 \newcommand{\ifglsfielddefeq}[5]{%
2093   \glsdoifexists{#1}%
2094   {%
2095     \edef\@glo@label{\glsdetoklabel{#1}}%
2096     \ifcsdef{glo@\@glo@label @#2}%
2097     {%
2098       \expandafter\ifdefstrequal
2099       \csname glo@\@glo@label @#2\endcsname{#3}{#4}{#5}%
2100     }%
2101     {%
2102       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2103     }%

```

```

2104 }%
2105 }

```

```

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

```

As above but uses \ifcsstrequal instead of \ifdefstrequal

```

2106 \newcommand{\ifglsfieldcseq}[5]{%
2107   \glsdoifexists{#1}%
2108   {%
2109     \edef\@glo@label{\glsdetoklabel{#1}}%
2110     \ifcsdef{glo@\@glo@label @#2}%
2111     {%
2112       \ifcsstrequal{glo@\@glo@label @#2}{#3}{#4}{#5}%
2113       }%
2114     {%
2115       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2116     }%
2117   }%
2118 }

```

gls.writedefhook

```

2119 \newcommand*{\gls.writedefhook}{}

```

gls@assign@desc

```

2120 \newcommand*{\gls@assign@desc}[1]{%
2121   \gls@assign@field{}{#1}{desc}{\@glo@desc}%
2122   \gls@assign@field{\@glo@desc}{#1}{descplural}{\@glo@descplural}%
2123 }

```

ewglossaryentry

```

2124 \newcommand{\longnewglossaryentry}[3]{%
2125   \glsdoifnoexists{#1}%
2126   {%
2127     \bgroup
2128     \let\@org@newglossaryentryprehook\@newglossaryentryprehook
2129     \long\def\@newglossaryentryprehook{%
2130       \long\def\@glo@desc{#3\leavevmode\unskip\nopostdesc}%
2131       \@org@newglossaryentryprehook
2132     }%
2133     \renewcommand*{\gls@assign@desc}[1]{%
2134       \global\cslet{glo@\glsdetoklabel{#1}@desc}{\@glo@desc}%
2135       \global\cslet{glo@\glsdetoklabel{#1}@descplural}{\@glo@desc}%
2136     }
2137     \gls@defglossaryentry{#1}{#2}%
2138   \egroup
2139 }
2140 }

```

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

```
2141 \@onlypreamble{\longnewglossaryentry}
```

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

```
2142 \newcommand{\longprovideglossaryentry}[3]{%
2143   \ifglentryexists{#1}{}%
2144   {\longnewglossaryentry{#1}{#2}{#3}}%
2145 }
2146 \@onlypreamble{\longprovideglossaryentry}
```

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

Defines a new entry without checking if it already exists.

```
2147 \newcommand{\gls@defglossaryentry}[2]{%
```

Store label

```
2148   \edef\@glo@label{\glsdetoklabel{#1}}%
```

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

```
2149   \let\glslabel\@glo@label
```

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

```
2150   \let\@glo@name\@gls@name
```

```
2151   \let\@glo@desc\@gls@desc
```

```
2152   \let\@glo@descplural\@gls@default@value
```

```
2153   \let\@glo@type\@gls@default@value
```

```
2154   \let\@glo@symbol\@gls@default@value
```

```
2155   \let\@glo@symbolplural\@gls@default@value
```

```
2156   \let\@glo@text\@gls@default@value
```

```
2157   \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.)

```
2158   \let\@glo@first\@gls@default@value
```

```
2159   \let\@glo@firstplural\@gls@default@value
```

Set the default sort:

```
2160   \let\@glo@sort\@gls@default@value
```

Set the default counter:

```
2161   \let\@glo@counter\@gls@default@value
```

```
2162   \def\@glo@see{}%
```

```
2163   \def\@glo@parent{}%
```

```

2164 \def\@glo@prefix{}%
2165 \def\@glo@useri{}%
2166 \def\@glo@userii{}%
2167 \def\@glo@useriii{}%
2168 \def\@glo@useriv{}%
2169 \def\@glo@userv{}%
2170 \def\@glo@uservi{}%

2171 \def\@glo@short{}%
2172 \def\@glo@shortpl{}%
2173 \def\@glo@long{}%
2174 \def\@glo@longpl{}%

```

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

```
2175 \@newglossaryentryprehook
```

Extract key-val information from third parameter:

```
2176 \setkeys{glossentry}{#2}%
```

Check there is a default glossary.

```

2177 \ifundef\glsdefaulttype
2178 {%
2179   \PackageError{glossaries}%
2180   {No default glossary type (have you used 'nomain'?)}%
2181   {If you use package option 'nomain' you must define
2182    a new glossary before you can define entries}%
2183 }%
2184 {}%

```

Assign type. This must be fully expandable

```

2185 \gls@assign@field{\glsdefaulttype}{\@glo@label}{type}{\@glo@type}%
2186 \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.

```

2187 \ifcsundef{glolist@\@glo@type}%
2188 {%
2189   \PackageError{glossaries}%
2190   {Glossary type '\@glo@type' has not been defined}%
2191   {You need to define a new glossary type, before making entries
2192    in it}%
2193 }%
2194 {%

```

Check if it's an ignored glossary

```

2195 \ifignoredglossary\@glo@type
2196 {%

```

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

```

2197 \ifx\@glo@desc\glsnodesc
2198 \let\@glo@desc\@empty

```

```

2199      \fi
2200    }%
2201    {%
2202    }%
2203    \protected@edef\@glo@list@\csname glo@list@\@glo@type\endcsname}%
2204    \expandafter\xdef\csname glo@list@\@glo@type\endcsname{%
2205      \@glo@list@\@glo@label},}%
2206    }%

  Initialise level to 0.
2207    \gls@level=0\relax

  Has this entry been assigned a parent?
2208    \ifx\@glo@parent\@empty

    Doesn't have a parent. Set \glo@<label>@parent to empty.
2209      \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{%
2210        \else

    Has a parent. Check to ensure this entry isn't its own parent.
2211      \ifdefequal\@glo@label\@glo@parent%
2212      {%
2213        \PackageError{glossaries}{Entry '@@glo@label' can't be its own parent}{}%
2214        \def\@glo@parent{%
2215          \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{%
2216            }%
2217        }%

    Check the parent exists:
2218      \ifglentryexists{\@glo@parent}%
2219      {%

    Parent exists. Set \glo@<label>@parent.
2220      \expandafter\xdef\csname glo@\@glo@label @parent\endcsname{%
2221        \@glo@parent}%

    Determine level.
2222      \gls@level=\csname glo@\@glo@parent @level\endcsname\relax
2223      \advance\gls@level by 1\relax

    If name hasn't been specified, use same as the parent name
2224      \ifx\@glo@name\@gls@name
2225      \expandafter\let\expandafter\@glo@name
2226      \csname glo@\@glo@parent @name\endcsname

    If name and plural haven't been specified, use same as the parent
2227      \ifx\@glo@plural\@gls@default@value
2228      \expandafter\let\expandafter\@glo@plural
2229      \csname glo@\@glo@parent @plural\endcsname
2230      \fi
2231    \fi
2232  }%
2233  {%

```

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

```

2234 \PackageError{glossaries}%
2235 {%
2236     Invalid parent '\@glo@parent'
2237     for entry '\@glo@label' - parent doesn't exist%
2238 }%
2239 {%
2240     Parent entries must be defined before their children%
2241 }%
2242 \def\@glo@parent{%
2243     \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{%
2244 }%
2245 }%
2246 \fi

```

Set the level for this entry

```

2247 \expandafter\xdef\csname glo@\@glo@label @level\endcsname{\number\gls@level}%

```

Define commands associated with this entry:

```

2248 \gls@assign@field{\@glo@name}{\@glo@label}{sortvalue}{\@glo@sort}%
2249 \letcs\@glo@sort{glo@\@glo@label @sortvalue}%
2250 \gls@assign@field{\@glo@name}{\@glo@label}{text}{\@glo@text}%
2251 \expandafter\gls@assign@field\expandafter
2252     {\csname glo@\@glo@label @text\endcsname\glspluralsuffix}%
2253     {\@glo@label}{plural}{\@glo@plural}%
2254 \expandafter\gls@assign@field\expandafter
2255     {\csname glo@\@glo@label @text\endcsname}%
2256     {\@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.

```

2257 \ifx\@glo@first\@gls@default@value
2258     \expandafter\gls@assign@field\expandafter
2259         {\csname glo@\@glo@label @plural\endcsname}%
2260         {\@glo@label}{firstpl}{\@glo@firstplural}%
2261 \else
2262     \expandafter\gls@assign@field\expandafter
2263         {\csname glo@\@glo@label @first\endcsname\glspluralsuffix}%
2264         {\@glo@label}{firstpl}{\@glo@firstplural}%
2265 \fi

2266 \ifcsundef{@glo@type@\@glo@type @counter}%
2267 {%
2268     \def\@glo@defaultcounter{\glscounter}%
2269 }%
2270 {%
2271     \letcs\@glo@defaultcounter{@glo@type@\@glo@type @counter}%
2272 }%
2273 \gls@assign@field{\@glo@defaultcounter}{\@glo@label}{counter}{\@glo@counter}%
2274 \gls@assign@field{}{\@glo@label}{useri}{\@glo@useri}%
2275 \gls@assign@field{}{\@glo@label}{userii}{\@glo@userii}%

```

```

2276 \gls@assign@field{}{\@glo@label}{useriii}{\@glo@useriii}%
2277 \gls@assign@field{}{\@glo@label}{useriv}{\@glo@useriv}%
2278 \gls@assign@field{}{\@glo@label}{userv}{\@glo@userv}%
2279 \gls@assign@field{}{\@glo@label}{uservi}{\@glo@uservi}%
2280 \gls@assign@field{}{\@glo@label}{short}{\@glo@short}%
2281 \gls@assign@field{}{\@glo@label}{shortpl}{\@glo@shortpl}%
2282 \gls@assign@field{}{\@glo@label}{long}{\@glo@long}%
2283 \gls@assign@field{}{\@glo@label}{longpl}{\@glo@longpl}%
2284 \ifx\@glo@name\@glsnname
2285 \@glsnname
2286 \let\@glo@name\@gls@default@value
2287 \fi
2288 \gls@assign@field{}{\@glo@label}{name}{\@glo@name}%

```

Set default numberlist if not defined:

```

2289 \ifcsundef{glo@\@glo@label @numberlist}%
2290 {%
2291 \csxdef{glo@\@glo@label @numberlist}{%
2292 \noexpand\@gls@missingnumberlist{\@glo@label}}%
2293 }%
2294 {}%

```

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.

```

2295 \def\@glo@@desc{\@glo@first}%
2296 \ifx\@glo@desc\@glo@@desc
2297 \let\@glo@desc\@glo@first
2298 \fi
2299 \ifx\@glo@desc\@glsnname
2300 \@glsnname
2301 \let\@glo@desc\@gls@default@value
2302 \fi
2303 \gls@assign@desc{\@glo@label}%

```

Set the sort key for this entry:

```

2304 \@gls@defsort{\@glo@type}{\@glo@label}%

2305 \def\@glo@@symbol{\@glo@text}%
2306 \ifx\@glo@symbol\@glo@@symbol
2307 \let\@glo@symbol\@glo@text
2308 \fi
2309 \gls@assign@field{\relax}{\@glo@label}{symbol}{\@glo@symbol}%
2310 \expandafter
2311 \gls@assign@field\expandafter
2312 {\csname glo@\@glo@label @symbol\endcsname}
2313 {\@glo@label}{symbolplural}{\@glo@symbolplural}%

```

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

```

2314 \expandafter\xdef\csname glo@\@glo@label @flagfalse\endcsname{%
2315 \noexpand\global

```

```

2316      \noexpand\let\expandafter\noexpand
2317      \csname ifglo@\@glo@label @flag\endcsname\noexpand\iffalse
2318  }%
2319  \expandafter\xdef\csname glo@\@glo@label @flagtrue\endcsname{%
2320      \noexpand\global
2321      \noexpand\let\expandafter\noexpand
2322      \csname ifglo@\@glo@label @flag\endcsname\noexpand\iftrue
2323  }%
2324  \csname glo@\@glo@label @flagfalse\endcsname

Sort out any cross-referencing if required.

2325  \ifdefined\@glo@see
2326  {}%
2327  {%
2328      \protected@edef\@do@glsee{%
2329          \noexpand\@gl@fixbraces\noexpand\@glo@list\@glo@see
2330          \noexpand\@nil
2331          \noexpand\expandafter\noexpand\@glsee\noexpand\@glo@list{\@glo@label}}%
2332      \@do@glsee
2333  }%

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

2334  \ifignoredglossary\@glo@type
2335  {%
2336      \csdef{glo@\@glo@label @index}{}%
2337  }
2338  {%
2339      \do@glo@storeentry{\@glo@label}%
2340  }%

Define entry counters if enabled:

2341  \@newglossaryentry@defcounters

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

2342  \@newglossaryentryposthook
2343  }

```

aryentryprehook Allow extra information to be added to glossary entries:

```
2344 \newcommand*{\@newglossaryentryprehook}{}
```

ryentryposthook Allow extra information to be added to glossary entries:

```
2345 \newcommand*{\@newglossaryentryposthook}{}
```

try@defcounters

```
2346 \newcommand*{\@newglossaryentry@defcounters}{}
```

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

```
2347 \newcommand*{\glsmoveentry}[2]{%
2348   \edef\@glo@thislabel{\glsetoklabel{#1}}%
```



```

2349 \edef\glo@type{\csname glo@\@glo@thislabel @type\endcsname}%
2350 \def\glo@list{,}%
2351 \forlslentries[\glo@type]{\glo@label}%
2352 {%
2353     \ifdefequal\@glo@thislabel\glo@label
2354         {\eappto\glo@list{\glo@label,}}%
2355     }%
2356 \cslet\glolist@\glo@type{\glo@list}%
2357 \csdef{glo@\@glo@thislabel @type}{#2}%
2358 }

```

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

```

2359 \ifglxindy
2360 \newcommand*{\@glossaryentryfield}{\string\glossentry}
2361 \else
2362 \newcommand*{\@glossaryentryfield}{\string\glossentry}
2363 \fi

```

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

```

2364 \ifglxindy
2365 \newcommand*{\@glossarysubentryfield}{%
2366     \string\subglossentry}
2367 \else
2368 \newcommand*{\@glossarysubentryfield}{%
2369     \string\subglossentry}
2370 \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.)

```

2371 \newcommand{\@glo@storeentry}[1]{%

```

Escape makeindex/xindy special characters in the label:

```

2372 \edef\@glo@esclabel{#1}%
2373 \@gls@checkmkidxchars\@glo@esclabel

```

Get the sort string and escape any special characters

```

2374 \protected@edef\@glo@sort{\csname glo@#1@sort\endcsname}%
2375 \@gls@checkmkidxchars\@glo@sort

```

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

```

2376 \@gls@checkmkidxchars\@glo@prefix

```

Get the parent, if one exists

```

2377 \edef\@glo@parent{\csname glo@#1@parent\endcsname}%

```

Write the information to the glossary file.

```

2378 \ifglxsindy

```

Store using xindy syntax.

```

2379 \ifx\@glo@parent\@empty

```

Entry doesn't have a parent

```

2380 \expandafter\protected\xdef\csname glo@#1@index\endcsname{%
2381 (\string"\@glo@sort\string" %
2382 \string"\@glo@prefix\@glossaryentryfield{\@glo@esclabel}\string") %
2383 }%
2384 \else

```

Entry has a parent

```

2385 \expandafter\protected\xdef\csname glo@#1@index\endcsname{%
2386 \csname glo@\@glo@parent @index\endcsname
2387 (\string"\@glo@sort\string" %
2388 \string"\@glo@prefix\@glossarysubentryfield
2389 {\csname glo@#1@level\endcsname}{\@glo@esclabel}\string") %
2390 }%
2391 \fi
2392 \else

```

Store using makeindex syntax.

```

2393 \ifx\@glo@parent\@empty

```

Sanitize \@glo@prefix

```

2394 \@onelevel@sanitize\@glo@prefix

```

Entry doesn't have a parent

```

2395 \expandafter\protected\xdef\csname glo@#1@index\endcsname{%
2396 \@glo@sort\@gls@actualchar\@glo@prefix
2397 \@glossaryentryfield{\@glo@esclabel}%
2398 }%
2399 \else

```

Entry has a parent

```

2400 \expandafter\protected\xdef\csname glo@#1@index\endcsname{%
2401 \csname glo@\@glo@parent @index\endcsname\@gls@levelchar
2402 \@glo@sort\@gls@actualchar\@glo@prefix
2403 \@glossarysubentryfield
2404 {\csname glo@#1@level\endcsname}{\@glo@esclabel}%
2405 }%
2406 \fi
2407 \fi
2408 }

```

1.8 Resetting and unsetting entry flags

Each glossary entry is assigned a conditional of the form `\ifglo@<label>@flag` which determines whether or not the entry has been used (see also `\ifglsused` defined below). These flags can be set and unset using the following macros, but first we need to know if we're in `amsmath`'s align environment's measuring pass.

`@ifnotmeasuring`

```
2409 \AtBeginDocument{%
2410   \ifpackageloaded{amsmath}%
2411   {\let\gls@ifnotmeasuring\@gls@ifnotmeasuring}%
2412   }%
2413 }
2414 \newcommand*{\@gls@ifnotmeasuring}[1]{%
2415   \ifmeasuring@
2416   \else
2417     #1%
2418   \fi
2419 }
2420 \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.

```
2421 \newcommand*{\glsreset}[1]{%
2422   \gls@ifnotmeasuring
2423   {%
2424     \glsdoifexists{#1}%
2425     {%
2426       \@glsreset{#1}%
2427     }%
2428   }%
2429 }
```

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

```
2430 \newcommand*{\glslocalreset}[1]{%
2431   \gls@ifnotmeasuring
2432   {%
2433     \glsdoifexists{#1}%
2434     {%
2435       \@glslocalreset{#1}%
2436     }%
2437   }%
2438 }
```

`\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.

```
2439 \newcommand*{\glsunset}[1]{%
2440   \gls@ifnotmeasuring
```

```

2441  {%
2442    \glsdoifexists{#1}%
2443    {%
2444      \@glsunset{#1}%
2445    }%
2446  }%
2447 }

```

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

```

2448 \newcommand*{\glslocalunset}[1]{%
2449   \gls@ifnotmeasuring
2450   {%
2451     \glsdoifexists{#1}%
2452     {%
2453       \@glslocalunset{#1}%
2454     }%
2455   }%
2456 }

```

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

```

2457 \newcommand*{\@glslocalunset}{\@@glslocalunset}

```

`@@glslocalunset` Local unset without checks.

```

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

```

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

```

2461 \newcommand*{\@glsunset}{\@@glsunset}

```

`\@@glsunset` Global unset without checks.

```

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

```

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

```

2465 \newcommand*{\@glslocalreset}{\@@glslocalreset}

```

`@@glslocalreset` Local reset without checks.

```

2466 \newcommand*{\@@glslocalreset}[1]{%
2467   \expandafter\let\csname ifglo@glsdetoklabel{#1}@flag\endcsname\iffalse
2468 }

```

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

```

2469 \newcommand*{\@glsreset}{\@@glsreset}

```

`\@@glsreset` Global reset without checks.

```
2470 \newcommand*{\@@glsreset}[1]{%
2471   \expandafter\global\csname glo@glsdetoklabel{#1}@flagfalse\endcsname
2472 }
```

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

`\glsresetall`

```
2473 \newcommand*{\glsresetall}[1][\@glo@types]{%
2474   \forallglsentries[#1]{\@glsentry}%
2475   {%
2476     \glsreset{\@glsentry}%
2477   }%
2478 }
```

As above, but with only a local effect:

`lslocalresetall`

```
2479 \newcommand*{\glslocalresetall}[1][\@glo@types]{%
2480   \forallglsentries[#1]{\@glsentry}%
2481   {%
2482     \glslocalreset{\@glsentry}%
2483   }%
2484 }
```

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

`\glsunsetall`

```
2485 \newcommand*{\glsunsetall}[1][\@glo@types]{%
2486   \forallglsentries[#1]{\@glsentry}%
2487   {%
2488     \glsunset{\@glsentry}%
2489   }%
2490 }
```

As above, but with only a local effect:

`lslocalunsetall`

```
2491 \newcommand*{\glslocalunsetall}[1][\@glo@types]{%
2492   \forallglsentries[#1]{\@glsentry}%
2493   {%
2494     \glslocalunset{\@glsentry}%
2495   }%
2496 }
```

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

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

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

```
2497 \newcommand*{\@newglossaryentry@defcounters}{%
2498   \csdef{glo@\@glo@label @currcount}{0}%
2499   \csdef{glo@\@glo@label @prevcount}{0}%
2500 }
```

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

```
2501 \newcommand*{\glsenableentrycount}{%
```

Enable new entry fields.

```
2502 \let\@newglossaryentry@defcounters\@newglossaryentry@defcounters
```

Disable `\newglossaryentry` in the document environment.

```
2503 \renewcommand*{\gls@defdocnewglossaryentry}{%
2504   \renewcommand*\newglossaryentry[2]{%
2505     \PackageError{glossaries}{\string\newglossaryentry\space
2506     may only be used in the preamble when entry counting has
2507     been activated}{If you use \string\glsenableentrycount\space
2508     you must place all entry definitions in the preamble not in
2509     the document environment}%
2510   }%
2511 }
```

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

```
2512 \newcommand*{\glsentrycurrcount}[1]{%
2513   \ifcsundef{glo@\glsdetoklabel{##1}@currcount}%
2514   {0}{\@gls@entry@field{##1}{currcount}}%
2515 }%
2516 \newcommand*{\glsentryprevcount}[1]{%
2517   \ifcsundef{glo@\glsdetoklabel{##1}@prevcount}%
2518   {0}{\@gls@entry@field{##1}{prevcount}}%
2519 }
```

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

```
2520 \renewcommand*{\@glsunset}[1]{%
2521   \@glsunset{##1}%
2522   \@gls@increment@currcount{##1}%
2523 }
```

```

2524 \renewcommand*{\@glslocalunset}[1]{%
2525   \@glslocalunset{##1}%
2526   \@gls@local@increment@currcount{##1}%
2527 }%
2528 \renewcommand*{\@glsreset}[1]{%
2529   \@glsreset{##1}%
2530   \csgdef{glo@glsdetoklabel{##1}@currcount}{0}%
2531 }%
2532 \renewcommand*{\@glslocalreset}[1]{%
2533   \@glslocalreset{##1}%
2534   \csdef{glo@glsdetoklabel{##1}@currcount}{0}%
2535 }%

```

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.)

```

2536 \def\@cgl's@##1##2[##3]{%
2537   \ifnum\glsentryprevcount{##2}=1\relax
2538     \cgl'sformat{##2}{##3}%
2539     \glsunset{##2}%
2540   \else
2541     \@gls@{##1}{##2}[##3]%
2542   \fi
2543 }%

```

Similarly for the analogous commands. No case change plural:

```

2544 \def\@cgl'spl@##1##2[##3]{%
2545   \ifnum\glsentryprevcount{##2}=1\relax
2546     \cgl'splformat{##2}{##3}%
2547     \glsunset{##2}%
2548   \else
2549     \@cgl'spl@{##1}{##2}[##3]%
2550   \fi
2551 }%

```

First letter uppercase singular:

```

2552 \def\@cGls@##1##2[##3]{%
2553   \ifnum\glsentryprevcount{##2}=1\relax
2554     \cGlsformat{##2}{##3}%
2555     \glsunset{##2}%
2556   \else
2557     \@Gls@{##1}{##2}[##3]%
2558   \fi
2559 }%

```

First letter uppercase plural:

```

2560 \def\@cGlspl@##1##2[##3]{%
2561   \ifnum\glsentryprevcount{##2}=1\relax
2562     \cGlsplformat{##2}{##3}%
2563     \glsunset{##2}%
2564   \else
2565     \@Glspl@{##1}{##2}[##3]%

```

```

2566 \fi
2567 }%

```

Write information to aux file at the end of the document

```

2568 \AtEndDocument{\@gls@write@entrycounts}%

```

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

```

2569 \renewcommand*{\@gls@entry@count}[2]{%
2570 \csgdef{glo@glstetoklabel{##1}@prevcount}{##2}%
2571 }%

```

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

```

2572 \let\glsenableentrycount\relax
2573 }
2574 \@onlypreamble\glsenableentrycount

```

`ement@currcount`

```

2575 \newcommand*{\@gls@increment@currcount}[1]{%
2576 \csxdef{glo@glstetoklabel{##1}@currcount}{%
2577 \number\numexpr\glsentrycurrcount{##1}+1}%
2578 }

```

`ement@currcount`

```

2579 \newcommand*{\@gls@local@increment@currcount}[1]{%
2580 \csedef{glo@glstetoklabel{##1}@currcount}{%
2581 \number\numexpr\glsentrycurrcount{##1}+1}%
2582 }

```

`ite@entrycounts`

Write the entry counts to the aux file. Use `\immediate` since this occurs right at the end of the document. Only write information for entries that have been used. (Some users have a file containing vast numbers of entries, many of which may not be used. There's no point writing information about the entries that haven't been used and it will only slow things down.)

```

2583 \newcommand*{\@gls@write@entrycounts}{%
2584 \immediate\write\@auxout
2585 {\string\providecommand*{\string\@gls@entry@count}[2]{}}%
2586 \forallglsentries{\@glsentry}{%
2587 \ifglsused{\@glsentry}%
2588 {\immediate\write\@auxout
2589 {\string\@gls@entry@count{\@glsentry}{\glsentrycurrcount{\@glsentry}}}%
2590 {}}%
2591 }%
2592 }

```

`gls@entry@count`

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

```

2593 \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.)

```

2594 \newrobustcmd*{\cgl}{\@gls@hyp@opt\@cgl}

```


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

```
2595 \newcommand*{\@cgl}{2} [] {%
2596   \new@ifnextchar[{\@cgl@{#1}{#2}}{\@cgl@{#1}{#2} []}%
2597 }
```

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

```
2598 \def\@cgl@#1#2[#3]{%
2599   \GlossariesWarning{\string\cgl\space is defaulting to
2600     \string\gl\space since you haven't enabled entry counting}%
2601   \@gl@{#1}{#2}[#3]%
2602 }
```

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

```
2603 \newcommand*{\cglformat}{2} {%
2604   \ifglshaslong{#1}{\glentrylong{#1}}{\glentryfirst{#1}}#2%
2605 }
```

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

```
2606 \newrobustcmd*{\cGls}{\@glshyp@opt\@cGls}
```

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

```
2607 \newcommand*{\@cGls}{2} [] {%
2608   \new@ifnextchar[{\@cGls@{#1}{#2}}{\@cGls@{#1}{#2} []}%
2609 }
```

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

```
2610 \def\@cGls@#1#2[#3]{%
2611   \GlossariesWarning{\string\cGls\space is defaulting to
2612     \string\Gls\space since you haven't enabled entry counting}%
2613   \@Gls@{#1}{#2}[#3]%
2614 }
```

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

```
2615 \newcommand*{\cGlsformat}{2} {%
2616   \ifglshaslong{#1}{\Glentrylong{#1}}{\Glentryfirst{#1}}#2%
2617 }
```

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

```
2618 \newrobustcmd*{\cglspl}{\@glshyp@opt\@cglspl}
```

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

```
2619 \newcommand*{\@cglsp1}[2] [] {%
2620   \new@ifnextchar[{\@cglsp1@{#1}{#2}}{\@cglsp1@{#1}{#2} []}%
2621 }
```

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

```
2622 \def\@cglsp1@#1#2[#3] {%
2623   \GlossariesWarning{\string\cglsp1\space is defaulting to
2624     \string\glsp1\space since you haven't enabled entry counting}%
2625   \@glsp1@{#1}{#2}[#3]%
2626 }
```

`\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.

```
2627 \newcommand*{\cglsp1format}[2] {%
2628   \ifglshaslong{#1}{\gl Sentrylongpl{#1}}{\gl Sentryfirstplural{#1}}#2%
2629 }
```

`\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.)

```
2630 \newrobustcmd*{\cGlsp1}{\@glshyp@opt\@cGlsp1}
```

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

```
2631 \newcommand*{\@cGlsp1}[2] [] {%
2632   \new@ifnextchar[{\@cGlsp1@{#1}{#2}}{\@cGlsp1@{#1}{#2} []}%
2633 }
```

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

```
2634 \def\@cGlsp1@#1#2[#3] {%
2635   \GlossariesWarning{\string\cGlsp1\space is defaulting to
2636     \string\Glsp1\space since you haven't enabled entry counting}%
2637   \@Glsp1@{#1}{#2}[#3]%
2638 }
```

`\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.

```
2639 \newcommand*{\cGlsp1format}[2] {%
2640   \ifglshaslong{#1}{\Gl Sentrylongpl{#1}}{\Gl Sentryfirstplural{#1}}#2%
2641 }
```

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

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

`\loadglsentries[⟨type⟩]{⟨filename⟩}`

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

`\loadglsentries`

```
2642 \newcommand*{\loadglsentries}[2][\@gls@default]{%
2643   \let\@gls@default\glsdefaulttype
2644   \def\glsdefaulttype{#1}\input{#2}%
2645   \let\glsdefaulttype\@gls@default
2646 }
```

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

```
2647 \@onlypreamble{\loadglsentries}
```

1.11 Using glossary entries in the text

Any term that has been defined using `\newglossaryentry` (or `\newacronym`) can be displayed in the text (i.e. outside of the glossary) using one of the commands defined in this section. Unless you use `\glslink`, the way the term appears in the text is determined by `\glsdisplayfirst` (if it is the first time the term has been used) or `\glsdisplay` (for subsequent use). Any formatting commands (such as `\textbf` is governed by `\glstextformat`. By default this just displays the link text “as is”.

`\glstextformat`

```
2648 \newcommand*{\glstextformat}[1]{#1}
```

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

```
2649 \newcommand*{\glsentryfmt}{%
2650   \@gls@default@entryfmt\glsdisplayfirst\glsdisplay
2651 }
```

Format that provides backwards compatibility:

```
2652 \newcommand*{\@gls@default@entryfmt}[2]{%
2653   \ifdefempty\glscustomtext
2654   {%
2655     \glsifplural
2656     {%
```

Plural form

```
2657     \glscapscase
2658     {%
```

Don't adjust case

```
2659      \ifglused\glslabel
2660      {%
```

Subsequent use

```
2661      #2{\glsentryplural{\glslabel}}%
2662      {\glsentrydescplural{\glslabel}}%
2663      {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2664      }%
2665      {%
```

First use

```
2666      #1{\glsentryfirstplural{\glslabel}}%
2667      {\glsentrydescplural{\glslabel}}%
2668      {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2669      }%
2670      }%
2671      {%
```

Make first letter upper case

```
2672      \ifglused\glslabel
2673      {%
```

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 `\defglentryfmt`, which avoids the issues caused by fragile commands.)

```
2674      \ifbool{glcompatible-3.07}%
2675      {%
2676      \protected@edef\@glo@etext{%
2677      #2{\glsentryplural{\glslabel}}%
2678      {\glsentrydescplural{\glslabel}}%
2679      {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2680      \xmakefirstuc\@glo@etext
2681      }%
2682      {%
2683      #2{\Glsentryplural{\glslabel}}%
2684      {\Glsentrydescplural{\glslabel}}%
2685      {\Glsentrysymbolplural{\glslabel}}{\glsinsert}%
2686      }%
2687      }%
2688      {%
```

First use

```
2689      \ifbool{glcompatible-3.07}%
2690      {%
2691      \protected@edef\@glo@etext{%
2692      #1{\glsentryfirstplural{\glslabel}}%
2693      {\glsentrydescplural{\glslabel}}%
2694      {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2695      \xmakefirstuc\@glo@etext
2696      }%
```

```

2697         {%
2698         #1{\Glsentryfirstplural{\glslabel}}}%
2699         {\Glsentrydescplural{\glslabel}}}%
2700         {\Glsentrysymbolplural{\glslabel}}{\glsinsert}%
2701     }%
2702 }%
2703 }%
2704 {%

```

Make all upper case

```

2705     \ifglscased\glslabel
2706     {%

```

Subsequent use

```

2707         \mfirstucMakeUppercase{#2{\Glsentryplural{\glslabel}}}%
2708         {\Glsentrydescplural{\glslabel}}}%
2709         {\Glsentrysymbolplural{\glslabel}}{\glsinsert}}}%
2710     }%
2711     {%

```

First use

```

2712         \mfirstucMakeUppercase{#1{\Glsentryfirstplural{\glslabel}}}%
2713         {\Glsentrydescplural{\glslabel}}}%
2714         {\Glsentrysymbolplural{\glslabel}}{\glsinsert}}}%
2715     }%
2716 }%
2717 }%
2718 {%

```

Singular form

```

2719     \glscapscase
2720     {%

```

Don't adjust case

```

2721     \ifglscased\glslabel
2722     {%

```

Subsequent use

```

2723         #2{\Glsentrytext{\glslabel}}}%
2724         {\Glsentrydesc{\glslabel}}}%
2725         {\Glsentrysymbol{\glslabel}}{\glsinsert}%
2726     }%
2727     {%

```

First use

```

2728         #1{\Glsentryfirst{\glslabel}}}%
2729         {\Glsentrydesc{\glslabel}}}%
2730         {\Glsentrysymbol{\glslabel}}{\glsinsert}%
2731     }%
2732 }%
2733 {%

```

Make first letter upper case

```
2734      \ifglused\glslabel
2735      {%
```

Subsequent use

```
2736      \ifbool{glscompatible-3.07}%
2737      {%
2738      \protected@edef\@glo@etext{%
2739      #2{\glsentrytext{\glslabel}}%
2740      {\glsentrydesc{\glslabel}}%
2741      {\glsentrysymbol{\glslabel}}{\glsinsert}}%
2742      \xmakefirstuc\@glo@etext
2743      }%
2744      {%
2745      #2{\Glsentrytext{\glslabel}}%
2746      {\glsentrydesc{\glslabel}}%
2747      {\glsentrysymbol{\glslabel}}{\glsinsert}}%
2748      }%
2749      }%
2750      {%
```

First use

```
2751      \ifbool{glscompatible-3.07}%
2752      {%
2753      \protected@edef\@glo@etext{%
2754      #1{\glsentryfirst{\glslabel}}%
2755      {\glsentrydesc{\glslabel}}%
2756      {\glsentrysymbol{\glslabel}}{\glsinsert}}%
2757      \xmakefirstuc\@glo@etext
2758      }%
2759      {%
2760      #1{\Glsentryfirst{\glslabel}}%
2761      {\glsentrydesc{\glslabel}}%
2762      {\glsentrysymbol{\glslabel}}{\glsinsert}}%
2763      }%
2764      }%
2765      }%
2766      {%
```

Make all upper case

```
2767      \ifglused\glslabel
2768      {%
```

Subsequent use

```
2769      \mfirstucMakeUppercase{#2{\glsentrytext{\glslabel}}%
2770      {\glsentrydesc{\glslabel}}%
2771      {\glsentrysymbol{\glslabel}}{\glsinsert}}%
2772      }%
2773      {%
```

First use

```

2774      \mfirstucMakeUppercase{#1{\glentryfirst{\glslabel}}}%
2775      {\glentrydesc{\glslabel}}}%
2776      {\glentrysymbol{\glslabel}}{\glinsert}}}%
2777  }%
2778 }%
2779 }%
2780 }%
2781 {%

```

Custom text provided in \glsdisp

```

2782  \ifglused{\glslabel}%
2783  {%

```

Subsequent use

```

2784      #2{\glscustomtext}%
2785      {\glentrydesc{\glslabel}}}%
2786      {\glentrysymbol{\glslabel}}{}%
2787  }%
2788  {%

```

First use

```

2789      #1{\glscustomtext}%
2790      {\glentrydesc{\glslabel}}}%
2791      {\glentrysymbol{\glslabel}}{}%
2792  }%
2793 }%
2794 }

```

`\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.

```

2795 \newcommand*{\glsgenentryfmt}{%
2796   \ifdefempty\glscustomtext
2797   {%
2798     \gl@ifplural
2799     {%

```

Plural form

```

2800     \glscapscase
2801     {%

```

Don't adjust case

```

2802     \ifglused\glslabel
2803     {%

```

Subsequent use

```

2804     \glentryplural{\glslabel}\glinsert
2805     }%
2806     {%

```

First use

```

2807     \glentryfirstplural{\glslabel}\glinsert
2808     }%

```

2809 }%
2810 {%

Make first letter upper case

2811 \ifglsused\glslabel
2812 {%

Subsequent use.

2813 \Glsentryplural{\glslabel}\glsinsert
2814 }%
2815 {%

First use

2816 \Glsentryfirstplural{\glslabel}\glsinsert
2817 }%
2818 }%
2819 {%

Make all upper case

2820 \ifglsused\glslabel
2821 {%

Subsequent use

2822 \mfirstucMakeUppercase
2823 {\glsentryplural{\glslabel}\glsinsert}%
2824 }%
2825 {%

First use

2826 \mfirstucMakeUppercase
2827 {\glsentryfirstplural{\glslabel}\glsinsert}%
2828 }%
2829 }%
2830 }%
2831 {%

Singular form

2832 \glscapscase
2833 {%

Don't adjust case

2834 \ifglsused\glslabel
2835 {%

Subsequent use

2836 \glsentrytext{\glslabel}\glsinsert
2837 }%
2838 {%

First use

2839 \glsentryfirst{\glslabel}\glsinsert
2840 }%
2841 }%
2842 {%

Make first letter upper case

```
2843      \ifglsused\glslabel
2844      {%
```

Subsequent use

```
2845      \Glsentrytext{\glslabel}\glsinsert
2846      }%
2847      {%
```

First use

```
2848      \Glsentryfirst{\glslabel}\glsinsert
2849      }%
2850      }%
2851      {%
```

Make all upper case

```
2852      \ifglsused\glslabel
2853      {%
```

Subsequent use

```
2854      \mfirstucMakeUppercase{\glsentrytext{\glslabel}\glsinsert}%
2855      }%
2856      {%
```

First use

```
2857      \mfirstucMakeUppercase{\glsentryfirst{\glslabel}\glsinsert}%
2858      }%
2859      }%
2860      }%
2861      }%
2862      {%
```

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

```
2863      \glscustomtext\glsinsert
2864      }%
2865 }
```

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

```
2866 \newcommand*{\glsgenacfmt}{%
2867   \ifdefempty\glscustomtext
2868   {%
2869     \ifglsused\glslabel
2870     {%
```

Subsequent use:

```
2871     \glsifplural
2872     {%
```

Subsequent plural form:

```
2873     \glscapscase
2874     {%
```

Subsequent plural form, don't adjust case:

```
2875      \acronymfont{\glsentryshortpl{\glslabel}}\glsinsert
2876      }%
2877      {%
```

Subsequent plural form, make first letter upper case:

```
2878      \acronymfont{\Glsentryshortpl{\glslabel}}\glsinsert
2879      }%
2880      {%
```

Subsequent plural form, all caps:

```
2881      \mfirstucMakeUppercase
2882      {\acronymfont{\glsentryshortpl{\glslabel}}\glsinsert}%
2883      }%
2884      }%
2885      {%
```

Subsequent singular form

```
2886      \glscapscase
2887      {%
```

Subsequent singular form, don't adjust case:

```
2888      \acronymfont{\glsentryshort{\glslabel}}\glsinsert
2889      }%
2890      {%
```

Subsequent singular form, make first letter upper case:

```
2891      \acronymfont{\Glsentryshort{\glslabel}}\glsinsert
2892      }%
2893      {%
```

Subsequent singular form, all caps:

```
2894      \mfirstucMakeUppercase
2895      {\acronymfont{\glsentryshort{\glslabel}}\glsinsert}%
2896      }%
2897      }%
2898      }%
2899      {%
```

First use:

```
2900      \glsifplural
2901      {%
```

First use plural form:

```
2902      \glscapscase
2903      {%
```

First use plural form, don't adjust case:

```
2904      \genplacrformat{\glslabel}{\glsinsert}%
2905      }%
2906      {%
```

First use plural form, make first letter upper case:

```
2907      \Genplacrfullformat{\glslabel}{\glsinsert}%
2908      }%
2909      {%
```

First use plural form, all caps:

```
2910      \mfirstucMakeUppercase
2911      {\genplacrfullformat{\glslabel}{\glsinsert}}%
2912      }%
2913      }%
2914      {%
```

First use singular form

```
2915      \glscapscase
2916      {%
```

First use singular form, don't adjust case:

```
2917      \genacrfullformat{\glslabel}{\glsinsert}%
2918      }%
2919      {%
```

First use singular form, make first letter upper case:

```
2920      \Genacrfullformat{\glslabel}{\glsinsert}%
2921      }%
2922      {%
```

First use singular form, all caps:

```
2923      \mfirstucMakeUppercase
2924      {\genacrfullformat{\glslabel}{\glsinsert}}%
2925      }%
2926      }%
2927      }%
2928      }%
2929      {%
```

User supplied text.

```
2930      \glscustomtext
2931      }%
2932 }
```

genacrfullformat

```
\genacrfullformat{\langle label \rangle}{\langle insert \rangle}
```

The full format used by \glsgenacfmt (singular).

```
2933 \newcommand*{\genacrfullformat}[2]{%
2934   \glstrylong{#1}#2\space
2935   (\protect\firstacronymfont{\glstryshort{#1}})%
2936 }
```

Genacrfullformat

```
\Genacrfullformat{\langle label \rangle}{\langle insert \rangle}
```

As above but makes the first letter upper case.

```
2937 \newcommand*{\Genacrfullformat}[2]{%
2938   \protected@edef\gls@text{\genacrfullformat{#1}{#2}}%
2939   \xmakefirstuc\gls@text
2940 }
```

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

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

```
2941 \newcommand*{\genplacrfullformat}[2]{%
2942   \glsentrylongpl{#1}#2\space
2943   (\protect\firstacronymfont{\glsentryshortpl{#1}})%
2944 }
```

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

As above but makes the first letter upper case.

```
2945 \newcommand*{\Genplacrfullformat}[2]{%
2946   \protected@edef\gls@text{\genplacrfullformat{#1}{#2}}%
2947   \xmakefirstuc\gls@text
2948 }
```

`\glsdisplayfirst` Deprecated. Kept for backward compatibility.

```
2949 \newcommand*{\glsdisplayfirst}[4]{#1#4}
```

`\glsdisplay` Deprecated. Kept for backward compatibility.

```
2950 \newcommand*{\glsdisplay}[4]{#1#4}
```

`\defglsdisplay` Deprecated. Kept for backward compatibility.

```
2951 \newcommand*{\defglsdisplay}[2][\glsdefaulttype]{%
2952   \GlossariesWarning{\string\defglsdisplay\space is now obsolete.^^J
2953   Use \string\defglsentryfmt\space instead}%
2954   \expandafter\def\csname gls@#1@display\endcsname##1##2##3##4{#2}%
2955   \edef\@gls@doentrydef{%
2956     \noexpand\defglsentryfmt[#1]{%
2957       \noexpand\ifcsdef\gls@#1@displayfirst}%
2958     {%
2959       \noexpand\@@gls@default@entryfmt
2960       {\noexpand\csuse\gls@#1@displayfirst}}%
2961     {\noexpand\csuse\gls@#1@display}}%
2962   }%
2963   {%
2964     \noexpand\@@gls@default@entryfmt
2965     {\noexpand\glsdisplayfirst}%
2966     {\noexpand\csuse\gls@#1@display}}%
2967   }%
```

```

2968 }%
2969 }%
2970 \@gls@doentrydef
2971 }

```

`glsdisplayfirst` Deprecated. Kept for backward compatibility.

```

2972 \newcommand*{\defglsdisplayfirst}[2][\glsdefaulttype]{%
2973   \GlossariesWarning{\string\defglsdisplayfirst\space is now obsolete.^^J
2974   Use \string\defglsentryfmt\space instead}%
2975   \expandafter\def\csname gls@#1@displayfirst\endcsname##1##2##3##4{#2}%
2976   \edef\@gls@doentrydef{%
2977     \noexpand\defglsentryfmt[#1]{%
2978       \noexpand\ifcsdef{gls@#1@display}%
2979       {%
2980         \noexpand\@gls@default@entryfmt
2981         {\noexpand\csuse{gls@#1@displayfirst}}}%
2982         {\noexpand\csuse{gls@#1@display}}}%
2983       }%
2984       {%
2985         \noexpand\@gls@default@entryfmt
2986         {\noexpand\csuse{gls@#1@displayfirst}}}%
2987         {\noexpand\glsdisplay}%
2988       }%
2989     }%
2990   }%
2991   \@gls@doentrydef
2992 }

```

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.

```

2993 \define@key{glslink}{counter}{%
2994   \ifcsundef{c@#1}%
2995   {%
2996     \PackageError{glossaries}%
2997     {There is no counter called '#1'}%
2998   }%

```

```

2999      The counter key should have the name of a valid counter
3000      as its value%
3001  }%
3002 }%
3003 {%
3004   \def\@gls@counter{#1}%
3005 }%
3006 }

```

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.

```

3007 \define@key{glslink}{format}{%
3008   \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.

```

3009 \define@boolkey{glslink}{hyper}[true]{}

```

Initialise hyper key.

```

3010 \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.

```

3011 \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.

```

3012 \newcommand*\glslinkvar[3]{#1}

```

\glsifhyper Now deprecated.

```

3013 \newcommand*\glsifhyper[2]{%
3014   \glslinkvar{#1}{#2}{#1}%
3015   \GlossariesWarning{\string\glsifhyper\space is deprecated. Did
3016     you mean \string\glsifhyperon\space or \string\glslinkvar?}%
3017 }

```

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

```

3018 \newcommand*\@gls@hyp@opt[1]{%

```

```

3019 \let\glslinkvar\@firstofthree
3020 \let\@gls@hyp@opt@cs#1\relax
3021 \@ifstar{\s@gls@hyp@opt}%
3022 {\@ifnextchar+{\@firstoftwo{\p@gls@hyp@opt}}{\#1}}%
3023 }

```

\s@gls@hyp@opt Starred version

```

3024 \newcommand*{\s@gls@hyp@opt}[1] [] {%
3025 \let\glslinkvar\@secondofthree
3026 \@gls@hyp@opt@cs[hyper=false,#1]}

```

\p@gls@hyp@opt Plus version

```

3027 \newcommand*{\p@gls@hyp@opt}[1] [] {%
3028 \let\glslinkvar\@thirdofthree
3029 \@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

```

3030 \newrobustcmd*{\glslink}{%
3031 \@gls@hyp@opt\@gls@link
3032 }

```

\@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.

```

3033 \newcommand*{\@gls@link}[3] [] {%
3034 \glsdoifexistsordo{#2}%
3035 {%
3036 \let\do@gls@link@checkfirsthyper\relax
3037 \@gls@link[#1]{#2}{#3}%
3038 }%

```

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

```

3039 \glstextformat{#3}%
3040 }%

```

```

3041 \glspostlinkhook
3042 }

```

glspostlinkhook

```

3043 \newcommand*{\glspostlinkhook}{}
3044 % \end{macrocode}
3045 \end{macro}
3046 %
3047 %
3048 %\begin{macro}{\@gls@link@checkfirsthyper}
3049 % Check for first use and switch off \gloskey[glslink]{hyper} key
3050 % if hyperlink not wanted. (Should be off if first use and
3051 % hyper=false is on or if first use and both the entry is in an acronym
3052 % list and the acrfootnote setting is on.)
3053 % This assumes the glossary type is stored in \cs{glstype} and the
3054 % label is stored in \cs{glslabel}.
3055 %\changes{4.08}{2014-07-30}{new}
3056 % \begin{macrocode}
3057 \newcommand*{\@gls@link@checkfirsthyper}{%
3058 \ifglused{\glslabel}%
3059 {%
3060 }%
3061 {%
3062 \gls@checkisacronymlist\glstype
3063 \ifglshyperfirst
3064 \ifglsisacronymlist
3065 \ifglssacrfootnote
3066 \KV@glslink@hyperfalse
3067 \fi
3068 \fi
3069 \else
3070 \KV@glslink@hyperfalse
3071 \fi
3072 }%
3073 }
3074 }

```

Allow user to hook into this

```

3073 \glslinkcheckfirsthyperhook
3074 }

```

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

```

3075 \newcommand*{\glslinkcheckfirsthyperhook}{}

```

linkpostsetkeys

```

3076 \newcommand*{\glslinkpostsetkeys}{}

```

\glsifhyperon Check the value of the hyper key:

```

3077 \newcommand{\glsifhyperon}[2]{\ifKV@glslink@hyper#1\else#2\fi}

```

ablehyperinlist Disable hyperlink if in the “nohyper” list.


```

3078 \newcommand*{\do@glsglslink@hyperinlist}{%
3079   \expandafter\DTLifinlist\expandafter{\glstypel}{\@glsglslink@nohyperlist}%
3080   {\KV@glsglslink@hyperfalse}}}%
3081 }

```

lt@glsglslink@opts Hook to set default options for \@glsglslink.

```

3082 \newcommand*{\@glsglslink@setdefault@glsglslink@opts}{}

```

\@glsglslink

```

3083 \def\@glsglslink[#1]#2#3{%

```

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

```

3084   \leavevmode
3085   \edef\glsglslinklabel{\glsglslinkdetoklabel{#2}}%

```

Save options in \@glsglslink@opts and label in \@glsglslink@label

```

3086   \def\@glsglslink@opts{#1}%
3087   \let\@glsglslink@label\glsglslinklabel
3088   \def\@glsglslinknumberformat{\glsglslinknumberformat}%
3089   \edef\@glsglslink@counter{\csname glo@glsglslinklabel @counter\endcsname}%

```

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

```

3090   \edef\glsglslinktype{\csname glo@glsglslinklabel @type\endcsname}%

```

Save original setting

```

3091   \let\org@ifKV@glsglslink@hyper\ifKV@glsglslink@hyper

```

Set defaults:

```

3092   \@glsglslink@setdefault@glsglslink@opts

```

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

```

3093   \do@glsglslink@disablehyperinlist

```

Macros must set this before calling \@glsglslink. The commands that check the first use flag should set this to \@glsglslink@checkfirsthyper otherwise it should be set to \relax.

```

3094   \do@glsglslink@checkfirsthyper
3095   \setkeys{glsglslink}{#1}%

```

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

```

3096   \glsglslinkpostsetkeys

```

Store the entry’s counter in \theglsentrycounter

```

3097   \@glsglslink@saveentrycounter

```

Define sort key if necessary:

```

3098   \@glsglslink@setsort{\glsglslinklabel}%

```

(De-tok’ing done by \@do@wrglossary)

```

3099   \@do@wrglossary{#2}%
3100   \ifKV@glsglslink@hyper
3101     \@glsglslink{\glsglslinkprefix\glsglslinklabel}{\glsglslinktextformat{#3}}%
3102   \else

```

```

3103      \glsdonohyperlink{\glolinkprefix\glslabel}{\glstextformat{#3}}%
3104      \fi

      Restore original setting
3105      \let\ifKV@glslink@hyper\org@ifKV@glslink@hyper
3106  }

\glolinkprefix
3107 \newcommand*{\glolinkprefix}{glo:}

glsentrycounter  Set default value of entry counter
3108 \def\glsentrycounter{\glscounter}%

saveentrycounter  Need to check if using equation counter in align environment:
3109 \newcommand*{\@gls@saveentrycounter}{%
3110   \def\@gls@Hcounter{}%

      Are we using equation counter?
3111   \ifthenelse{\equal{\@gls@counter}{equation}}{%
3112     {

      If we're in align environment, \xatlevel@ will be defined. (Can't test for \@currentvir as
      may be inside an inner environment.)
3113     \ifcsundef{xatlevel@}%
3114     {%
3115       \edef\theglsentrycounter{\expandafter\noexpand
3116         \csname the\@gls@counter\endcsname}%
3117     }%
3118     {%
3119       \ifx\xatlevel@\@empty
3120         \edef\theglsentrycounter{\expandafter\noexpand
3121           \csname the\@gls@counter\endcsname}%
3122       \else
3123         \savecounters@
3124         \advance\c@equation by 1\relax
3125         \edef\theglsentrycounter{\csname the\@gls@counter\endcsname}%

      Check if hyperref version of this counter
3126         \ifcsundef{theH\@gls@counter}%
3127         {%
3128           \def\@gls@Hcounter{\theglsentrycounter}%
3129         }%
3130         {%
3131           \def\@gls@Hcounter{\csname theH\@gls@counter\endcsname}%
3132         }%
3133         \protected@edef\theHglentrycounter{\@gls@Hcounter}%
3134         \restorecounters@
3135       \fi
3136     }%
3137   }%
3138   {%

```

Not using equation counter so no special measures:

```

3139 \edef\theglsentrycounter{\expandafter\noexpand
3140 \csname the\@gls@counter\endcsname}%
3141 }%
```

Check if hyperref version of this counter

```

3142 \ifx\@gls@Hcounter\@empty
3143 \ifcsundef{theH\@gls@counter}%
3144 {%
3145 \def\theHglsentrycounter{\theglsentrycounter}%
3146 }%
3147 {%
3148 \protected@edef\theHglsentrycounter{\expandafter\noexpand
3149 \csname theH\@gls@counter\endcsname}%
3150 }%
3151 \fi
3152 }
```

`t@glo@numformat` Set the formatting information in the format required by `makeindex`. The first argument is the format specified by the user (via the `format` key), the second argument is the name of the counter used to indicate the location, the third argument is a control sequence which stores the required format and the fourth argument (new to v3.0) is the hyper-prefix.

```

3153 \def\@set@glo@numformat#1#2#3#4{%
3154 \expandafter\@glo@check@mkidxrangechar#3\@nil
3155 \protected@edef#1{%
3156 \@glo@prefix setentrycounter[#4]{#2}%
3157 \expandafter\string\csname\@glo@suffix\endcsname
3158 }%
3159 \@gls@checkmkidxchars#1%
3160 }
```

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.

```

3161 \def\@glo@check@mkidxrangechar#1#2\@nil{%
3162 \if#1(\relax
3163 \def\@glo@prefix{(%
3164 \if\relax#2\relax
3165 \def\@glo@suffix{glsnumberformat}%
3166 \else
3167 \def\@glo@suffix{#2}%
3168 \fi
3169 \else
3170 \if#1)\relax
3171 \def\@glo@prefix{)}%
3172 \if\relax#2\relax
3173 \def\@glo@suffix{glsnumberformat}%
3174 \else
```

```

3175     \def\@glo@suffix{#2}%
3176 \fi
3177 \else
3178     \def\@glo@prefix{}\def\@glo@suffix{#1#2}%
3179 \fi
3180 \fi}

```

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

```

3181 \newcommand*{\@gls@escbsdq}[1]{%
3182   \def\@gls@checkedmkidx{}%
3183   \let\gls@xdystring=#1\relax
3184   \@onelevel@sanitize\gls@xdystring
3185   \edef\do@gls@xdycheckbackslash{%
3186     \noexpand\@gls@xdycheckbackslash\gls@xdystring\noexpand\@nil
3187     \@backslashchar\@backslashchar\noexpand\null}%
3188   \do@gls@xdycheckbackslash
3189   \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdystring}%
3190   \def\@gls@checkedmkidx{}%
3191   \expandafter\@gls@xdycheckquote\gls@xdystring\@nil""\null
3192   \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdystring}%

```

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

```

3193   \@for\@gls@tmp:=\gls@protected@pagefmts\do
3194   {%
3195     \edef\@gls@sanitized@tmp{\expandafter\@gobble\string\\expandonce\@gls@tmp}%
3196     \@onelevel@sanitize\@gls@sanitized@tmp
3197     \edef\gls@dosubst{%
3198       \noexpand\DTLsubstituteall\noexpand\gls@xdystring
3199       {\@gls@sanitized@tmp}{\expandonce\@gls@tmp}%
3200     }%
3201     \gls@dosubst
3202   }%

```

Assign to required control sequence

```

3203   \let#1=\gls@xdystring
3204 }

```

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

`checkmkidxchars`

```

3205 \newcommand{\@gls@checkmkidxchars}[1]{%
3206   \ifglxsindy
3207     \@gls@escbsdq{#1}%
3208   \else
3209     \def\@gls@checkedmkidx{}%
3210     \expandafter\@gls@checkquote#1\@nil""\null
3211     \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3212     \def\@gls@checkedmkidx{}%
3213     \expandafter\@gls@checkescquote#1\@nil\""\null

```

```

3214 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3215 \def\@gls@checkedmkidx{}%
3216 \expandafter\@gls@checkescactual#1\@nil\?\?\null
3217 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3218 \def\@gls@checkedmkidx{}%
3219 \expandafter\@gls@checkactual#1\@nil??\null
3220 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3221 \def\@gls@checkedmkidx{}%
3222 \expandafter\@gls@checkbar#1\@nil||\null
3223 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3224 \def\@gls@checkedmkidx{}%
3225 \expandafter\@gls@checkescbar#1\@nil\\|\null
3226 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3227 \def\@gls@checkedmkidx{}%
3228 \expandafter\@gls@checklevel#1\@nil!!\null
3229 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3230 \fi
3231 }

```

Update the control sequence and strip trailing \@nil:

s@updatechecked

```

3232 \def\@gls@updatechecked#1\@nil#2{\def#2{#1}}

```

\@gls@tmpb Define temporary token

```

3233 \newtoks\@gls@tmpb

```

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

```

3234 \def\@gls@checkquote#1"#2"#3\null{%
3235 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3236 \toks@={#1}%
3237 \ifx\null#2\null
3238 \ifx\null#3\null
3239 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3240 \def\@gls@checkquote{\relax}%
3241 \else
3242 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3243 \@gls@quotechar\@gls@quotechar\@gls@quotechar\@gls@quotechar}%
3244 \def\@gls@checkquote{\@gls@checkquote#3\null}%
3245 \fi
3246 \else
3247 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3248 \@gls@quotechar\@gls@quotechar}%
3249 \ifx\null#3\null
3250 \def\@gls@checkquote{\@gls@checkquote#2""\null}%
3251 \else
3252 \def\@gls@checkquote{\@gls@checkquote#2"#3\null}%
3253 \fi
3254 \fi

```

```

3255 \@@gls@checkquote
3256 }

```

s@checkescquote Do the same for \":

```

3257 \def\@gls@checkescquote#1\"#2\"#3\null{%
3258 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3259 \toks@={#1}%
3260 \ifx\null#2\null
3261 \ifx\null#3\null
3262 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3263 \def\@@gls@checkescquote{\relax}%
3264 \else
3265 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3266 \@gls@quotechar\string\"@gls@quotechar
3267 \@gls@quotechar\string\"@gls@quotechar}%
3268 \def\@@gls@checkescquote{\@gls@checkescquote#3\null}%
3269 \fi
3270 \else
3271 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3272 \@gls@quotechar\string\"@gls@quotechar}%
3273 \ifx\null#3\null
3274 \def\@@gls@checkescquote{\@gls@checkescquote#2\"\" \null}%
3275 \else
3276 \def\@@gls@checkescquote{\@gls@checkescquote#2\"#3\null}%
3277 \fi
3278 \fi
3279 \@@gls@checkescquote
3280 }

```

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

```

3281 \def\@gls@checkescactual#1\?#2\?#3\null{%
3282 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3283 \toks@={#1}%
3284 \ifx\null#2\null
3285 \ifx\null#3\null
3286 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3287 \def\@@gls@checkescactual{\relax}%
3288 \else
3289 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3290 \@gls@quotechar\string\"@gls@actualchar
3291 \@gls@quotechar\string\"@gls@actualchar}%
3292 \def\@@gls@checkescactual{\@gls@checkescactual#3\null}%
3293 \fi
3294 \else
3295 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3296 \@gls@quotechar\string\"@gls@actualchar}%
3297 \ifx\null#3\null
3298 \def\@@gls@checkescactual{\@gls@checkescactual#2\?\?\null}%
3299 \else

```

```

3300 \def\@gls@checkescactual{\@gls@checkescactual#2\?#3\null}%
3301 \fi
3302 \fi
3303 \@gls@checkescactual
3304 }

```

gls@checkeschar Similarly for \|:

```

3305 \def\@gls@checkeschar#1\|#2\|#3\null{%
3306 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3307 \toks@={#1}%
3308 \ifx\null#2\null
3309 \ifx\null#3\null
3310 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3311 \def\@gls@checkeschar{\relax}%
3312 \else
3313 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3314 \@gls@quotechar\string"\@gls@encapchar
3315 \@gls@quotechar\string"\@gls@encapchar}%
3316 \def\@gls@checkeschar{\@gls@checkeschar#3\null}%
3317 \fi
3318 \else
3319 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3320 \@gls@quotechar\string"\@gls@encapchar}%
3321 \ifx\null#3\null
3322 \def\@gls@checkeschar{\@gls@checkeschar#2\|\|\null}%
3323 \else
3324 \def\@gls@checkeschar{\@gls@checkeschar#2\|#3\null}%
3325 \fi
3326 \fi
3327 \@gls@checkeschar
3328 }

```

s@checkesclevel Similarly for \!:

```

3329 \def\@gls@checkesclevel#1\!#2\!#3\null{%
3330 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3331 \toks@={#1}%
3332 \ifx\null#2\null
3333 \ifx\null#3\null
3334 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3335 \def\@gls@checkesclevel{\relax}%
3336 \else
3337 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3338 \@gls@quotechar\string"\@gls@levelchar
3339 \@gls@quotechar\string"\@gls@levelchar}%
3340 \def\@gls@checkesclevel{\@gls@checkesclevel#3\null}%
3341 \fi
3342 \else
3343 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3344 \@gls@quotechar\string"\@gls@levelchar}%

```

```

3345 \ifx\null#3\null
3346 \def\@gls@checkesclevel{\@gls@checkesclevel#2\!\!\null}%
3347 \else
3348 \def\@gls@checkesclevel{\@gls@checkesclevel#2\!#3\null}%
3349 \fi
3350 \fi
3351 \@gls@checkesclevel
3352 }

```

\@gls@checkbar and for |:

```

3353 \def\@gls@checkbar#1|#2|#3\null{%
3354 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3355 \toks@={#1}%
3356 \ifx\null#2\null
3357 \ifx\null#3\null
3358 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3359 \def\@gls@checkbar{\relax}%
3360 \else
3361 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3362 \@gls@quotechar\@gls@encapchar\@gls@quotechar\@gls@encapchar}%
3363 \def\@gls@checkbar{\@gls@checkbar#3\null}%
3364 \fi
3365 \else
3366 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3367 \@gls@quotechar\@gls@encapchar}%
3368 \ifx\null#3\null
3369 \def\@gls@checkbar{\@gls@checkbar#2||\null}%
3370 \else
3371 \def\@gls@checkbar{\@gls@checkbar#2|#3\null}%
3372 \fi
3373 \fi
3374 \@gls@checkbar
3375 }

```

\@gls@checklevel and for !:

```

3376 \def\@gls@checklevel#1!#2!#3\null{%
3377 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3378 \toks@={#1}%
3379 \ifx\null#2\null
3380 \ifx\null#3\null
3381 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3382 \def\@gls@checklevel{\relax}%
3383 \else
3384 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3385 \@gls@quotechar\@gls@levelchar\@gls@quotechar\@gls@levelchar}%
3386 \def\@gls@checklevel{\@gls@checklevel#3\null}%
3387 \fi
3388 \else
3389 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@

```



```

3390 \@gls@quotechar\@gls@levelchar}%
3391 \ifx\null#3\null
3392 \def\@gls@checklevel{\@gls@checklevel#2!\null}%
3393 \else
3394 \def\@gls@checklevel{\@gls@checklevel#2!#3\null}%
3395 \fi
3396 \fi
3397 \@gls@checklevel
3398 }

```

gls@checkactual and for ?:

```

3399 \def\@gls@checkactual#1?#2?#3\null{%
3400 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3401 \toks@={#1}%
3402 \ifx\null#2\null
3403 \ifx\null#3\null
3404 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3405 \def\@gls@checkactual{\relax}%
3406 \else
3407 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3408 \@gls@quotechar\@gls@actualchar\@gls@quotechar\@gls@actualchar}%
3409 \def\@gls@checkactual{\@gls@checkactual#3\null}%
3410 \fi
3411 \else
3412 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3413 \@gls@quotechar\@gls@actualchar}%
3414 \ifx\null#3\null
3415 \def\@gls@checkactual{\@gls@checkactual#2??\null}%
3416 \else
3417 \def\@gls@checkactual{\@gls@checkactual#2?#3\null}%
3418 \fi
3419 \fi
3420 \@gls@checkactual
3421 }

```

s@xdycheckquote As before but for use with xindy

```

3422 \def\@gls@xdycheckquote#1"#2"#3\null{%
3423 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3424 \toks@={#1}%
3425 \ifx\null#2\null
3426 \ifx\null#3\null
3427 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3428 \def\@gls@xdycheckquote{\relax}%
3429 \else
3430 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3431 \string"\string"}%
3432 \def\@gls@xdycheckquote{\@gls@xdycheckquote#3\null}%
3433 \fi
3434 \else

```

```

3435 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3436 \string\}%
3437 \ifx\null#3\null
3438 \def\@gls@xdycheckquote{\@gls@xdycheckquote#2""\null}%
3439 \else
3440 \def\@gls@xdycheckquote{\@gls@xdycheckquote#2"#3\null}%
3441 \fi
3442 \fi
3443 \@gls@xdycheckquote
3444 }

```

ycheckbackslash Need to escape all backslashes for xindy. Define command that will define \@gls@xdycheckbackslash

```

3445 \edef\def@gls@xdycheckbackslash{%
3446 \noexpand\def\noexpand\@gls@xdycheckbackslash##1\@backslashchar
3447 ##2\@backslashchar##3\noexpand\null{%
3448 \noexpand\@gls@tmpb=\noexpand\expandafter
3449 {\noexpand\@gls@checkedmkidx}%
3450 \noexpand\toks@={##1}%
3451 \noexpand\ifx\noexpand\null##2\noexpand\null
3452 \noexpand\ifx\noexpand\null##3\noexpand\null
3453 \noexpand\edef\noexpand\@gls@checkedmkidx{%
3454 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
3455 \noexpand\def\noexpand\@gls@xdycheckbackslash{\relax}%
3456 \noexpand\else
3457 \noexpand\edef\noexpand\@gls@checkedmkidx{%
3458 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
3459 \@backslashchar\@backslashchar\@backslashchar\@backslashchar}%
3460 \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3461 \noexpand\@gls@xdycheckbackslash##3\noexpand\null}%
3462 \noexpand\fi
3463 \noexpand\else
3464 \noexpand\edef\noexpand\@gls@checkedmkidx{%
3465 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
3466 \@backslashchar\@backslashchar}%
3467 \noexpand\ifx\noexpand\null##3\noexpand\null
3468 \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3469 \noexpand\@gls@xdycheckbackslash##2\@backslashchar
3470 \@backslashchar\noexpand\null}%
3471 \noexpand\else
3472 \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3473 \noexpand\@gls@xdycheckbackslash##2\@backslashchar
3474 ##3\noexpand\null}%
3475 \noexpand\fi
3476 \noexpand\fi
3477 \noexpand\@gls@xdycheckbackslash
3478 }%
3479 }

```

Now go ahead and define \@gls@xdycheckbackslash

```

3480 \def@gls@xdycheckbackslash

```

lsdohypertarget

```
3481 \newlength\gls@tmplen
3482 \newcommand*{\glsdohypertarget}[2]{%
3483   \settoheight{\gls@tmplen}{#2}%
3484   \raisebox{\gls@tmplen}{\hypertarget{#1}{}}#2%
3485 }
```

\glsdohyperlink

```
3486 \newcommand*{\glsdohyperlink}[2]{\hyperlink{#1}{#2}}
```

lsdonohyperlink

```
3487 \newcommand*{\glsdonohyperlink}[2]{#2}
```

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

```
3488 \ifcsundef{hyperlink}%
3489 {%
3490   \let\@glslink\glsdonohyperlink
3491 }%
3492 {%
3493   \let\@glslink\glsdohyperlink
3494 }
```

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

```
3495 \ifcsundef{hypertarget}%
3496 {%
3497   \let\@glstarget\@secondoftwo
3498 }%
3499 {%
3500   \let\@glstarget\glsdohypertarget
3501 }
```

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

`\glsdisablehyper`

```
3502 \newcommand{\glsdisablehyper}{%
3503   \KV@glslink@hyperfalse
3504   \let\@glslink\glsdonohyperlink
3505   \let\@glstarget\@secondoftwo
3506 }
```

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

`\glsenablehyper`

```
3507 \newcommand{\glsenablehyper}{%
3508   \KV@glslink@hypertrue
3509   \let\@glslink\glsdohyperlink
```

```

3510 \let\@glstarget\glsdohypertarget
3511 }

```

Provide some convenience commands if not already defined:

```

3512 \providecommand{\@firstofthree}[3]{#1}
3513 \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
3514 \newrobustcmd*{\gls}{\@gls@hyp@opt\@gls}

```

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

```

\@gls
3515 \newcommand*{\@gls}[2][{}]{%
3516   \new@ifnextchar[{\@gls@{#1}{#2}}{\@gls@{#1}{#2}[]}%
3517 }

```

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

```

3518 \def\@gls@#1#2[#3]{%
3519   \glsdoifexists{#2}%
3520   {%
3521     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
3522     \let\glsifplural\@secondoftwo
3523     \let\glsapscase\@firstofthree
3524     \let\glscustomtext\@empty
3525     \def\glsinsert{#3}%

```

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

```

3526   \def\@gls@text{\csname gls@\gls type @entryfmt\endcsname}%

```

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

```

3527   \@gls@link[#1]{#2}{\@gls@text}%

```

Indicate that this entry has now been used

```
3528 \ifKV@glslink@local
3529 \glsllocalunset{#2}%
3530 \else
3531 \glunset{#2}%
3532 \fi
3533 }%

3534 \glspostlinkhook
3535 }
```

`\Gls` behaves like `\gls`, but the first letter of the link text is converted to uppercase (note that if the first letter has an accent, the accented letter will need to be grouped when you define the entry). It is mainly intended for terms that start a sentence:

`\Gls`

```
3536 \newrobustcmd*{\Gls}{\@Gls@hyp@opt\@Gls}
```

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

```
3537 \newcommand*{\@Gls}[2][{}]{%
3538 \new@ifnextchar[{\@Gls@{#1}{#2}}{\@Gls@{#1}{#2}[]}%
3539 }
```

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

```
3540 \def\@Gls@#1#2[#3]{%
3541 \glstoifexists{#2}%
3542 {%
3543 \let\do@glslink@checkfirsthyper\@glslink@checkfirsthyper

3544 \let\glslifplural\@secondoftwo
3545 \let\glscapscase\@secondofthree
3546 \let\glscustomtext\@empty
3547 \def\glinsert{#3}%
```

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

```
3548 \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%
```

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

```
3549 \@glslink[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3550 \ifKV@glslink@local
3551 \glsllocalunset{#2}%
3552 \else
3553 \glunset{#2}%
3554 \fi
3555 }%
```

```

3556 \glspostlinkhook
3557 }

```

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

\GLS

```

3558 \newrobustcmd*{\GLS}{\@gls@hyp@opt\@GLS}

```

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

```

3559 \newcommand*{\@GLS}[2][\@GLS@{#1}{#2}]{\@GLS@{#1}{#2}[]}
3560 \new@ifnextchar[\@GLS@{#1}{#2}]{\@GLS@{#1}{#2}[]}
3561 }

```

\@GLS@ Read in the final optional argument:

```

3562 \def\@GLS@#1#2[#3]{%
3563   \glsdoifexists{#2}%
3564   {%
3565     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
3566     \let\glsifplural\@secondoftwo
3567     \let\glsupcase\@thirdofthree
3568     \let\glscustomtext\@empty
3569     \def\glsinsert{#3}%

```

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

```

3570 \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.

```

3571 \@gls@link[#1]{#2}{\@glo@text}%

```

Indicate that this entry has now been used

```

3572 \ifKV@glslink@local
3573   \glslocalunset{#2}%
3574 \else
3575   \glsunset{#2}%
3576 \fi
3577 }%

```

```

3578 \glspostlinkhook
3579 }

```

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

\glspl

```

3580 \newrobustcmd*{\glspl}{\@gls@hyp@opt\@glspl}

```

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

```

3581 \newcommand*{\@glspl}[2][\@glspl@{#1}{#2}]{\@glspl@{#1}{#2}[]}
3582 \new@ifnextchar[\@glspl@{#1}{#2}]{\@glspl@{#1}{#2}[]}
3583 }

```

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

```
3584 \def\@glsp1@#1#2[#3]{%
3585   \glstoifexists{#2}%
3586   {%
3587     \let\do@gl@link@checkfirsthyper\@gl@link@checkfirsthyper

3588     \let\gl@ifplural\@firstoftwo
3589     \let\gl@scapscase\@firstofthree
3590     \let\glscustomtext\@empty
3591     \def\glinsert{#3}%
```

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

```
3592   \def\@glo@text{\csname gls@\glstyle @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.

```
3593   \@gl@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3594   \ifKV@gl@link@local
3595     \gl@localunset{#2}%
3596   \else
3597     \gl@unset{#2}%
3598   \fi
3599 }%

3600 \glspostlinkhook
3601 }
```

`\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`

```
3602 \newrobustcmd*{\Glspl}{\@gl@hyp@opt\@Glspl}
```

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

```
3603 \newcommand*{\@Glspl}[2][ ]{%
3604   \new@ifnextchar[\@Glspl@{#1}{#2}}{\@Glspl@{#1}{#2}[ ]}%
3605 }
```

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

```
3606 \def\@Glspl@#1#2[#3]{%
3607   \glstoifexists{#2}%
3608   {%
3609     \let\do@gl@link@checkfirsthyper\@gl@link@checkfirsthyper

3610     \let\gl@ifplural\@firstoftwo
3611     \let\gl@scapscase\@secondofthree
3612     \let\glscustomtext\@empty
3613     \def\glinsert{#3}%
```

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

```

3614 \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 op-
tion is used.
3615 \@gls@link[#1]{#2}{\@glo@text}%
Indicate that this entry has now been used
3616 \ifKV@glslink@local
3617 \glslocalunset{#2}%
3618 \else
3619 \glsunset{#2}%
3620 \fi
3621 }%
3622 \glspostlinkhook
3623 }

```

\GLSp1 behaves like \glsp1 except that all the link text is converted to uppercase.

\GLSp1

```

3624 \newrobustcmd*{\GLSp1}{\@gls@hyp@opt\@GLSp1}
Defined the un-starred form. Need to determine if there is a final optional argument
3625 \newcommand*{\@GLSp1}[2][]{%
3626 \new@ifnextchar[{\@GLSp1@{#1}{#2}}{\@GLSp1@{#1}{#2}[]}%
3627 }

```

\@GLSp1 Read in the final optional argument:

```

3628 \def\@GLSp1@#1#2[#3]{%
3629 \glsdoifexists{#2}%
3630 {%
3631 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
3632 \let\glsifplural\@firstoftwo
3633 \let\glscapscase\@thirdofthree
3634 \let\glscustomtext\@empty
3635 \def\glsinsert{#3}%

```

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

```

3636 \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 op-
tion is used.
3637 \@gls@link[#1]{#2}{\@glo@text}%

```


Indicate that this entry has now been used

```

3638 \ifKV@glslink@local
3639 \glsllocalunset{#2}%
3640 \else
3641 \glunset{#2}%
3642 \fi
3643 }%

3644 \glspostlinkhook
3645 }

```

`\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:

```

3646 \newrobustcmd*{\glsdisp}{\@gls@hyp@opt\@glsdisp}

```

Defined the un-starred form.

`\@glsdisp`

```

3647 \newcommand*{\@glsdisp}[3] [] {%
3648 \glsoifexists{#2}{%

3649 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3650 \let\glsifplural\@secondoftwo
3651 \let\glscapscase\@firstofthree
3652 \def\glscustomtext{#3}%
3653 \def\glsinsert{}%

```

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

```

3654 \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.

```

3655 \@gls@link[#1]{#2}{\@glo@text}%

```

Indicate that this entry has now been used

```

3656 \ifKV@glslink@local
3657 \glsllocalunset{#2}%
3658 \else
3659 \glunset{#2}%
3660 \fi
3661 }%

3662 \glspostlinkhook
3663 }

```

checkfirsthyper Instead of just setting \do@gl@link@checkfirsthyper to \relax in \@gl@field@link, set it to \@gl@link@nocheckfirsthyper in case some other action needs to take place.

```
3664 \newcommand*{\@gl@link@nocheckfirsthyper}{}
```

@gl@field@link

```
3665 \newcommand{\@gl@field@link}[3]{%
3666   \glsoifexists{#2}%
3667   {%
3668     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
3669     \@gl@link[#1]{#2}{#3}%
3670   }%

3671   \glspostlinkhook
3672 }
```

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

\glstext

```
3673 \newrobustcmd*{\glstext}{\@gl@hyp@opt\@glstext}
```

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

```
3674 \newcommand*{\@glstext}[2] [] {%
3675   \new@ifnextchar[{\@glstext@{#1}{#2}}{\@glstext@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3676 \def\@glstext@#1#2[#3] {%
3677   \@gl@field@link{#1}{#2}{\glentrytext{#2}{#3}}%
3678 }
```

\GLStext behaves like \glstext except the text is converted to uppercase.

\GLStext

```
3679 \newrobustcmd*{\GLStext}{\@gl@hyp@opt\@GLStext}
```

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

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

Read in the final optional argument:

```
3682 \def\@GLStext@#1#2[#3] {%
3683   \@gl@field@link{#1}{#2}{\mfirstucMakeUppercase{\glentrytext{#2}{#3}}}%
3684 }
```

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

\Glstext

```
3685 \newrobustcmd*{\Glstext}{\@gl@hyp@opt\@Glstext}
```

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

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

Read in the final optional argument:

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

`\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`

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

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

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

Read in the final optional argument:

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

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

`\Glsfirst`

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

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

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

Read in the final optional argument:

```
3700 \def\@Glsfirst@#1#2[#3] {%  
3701   \@gls@field@link{#1}{#2}{\glentryfirst{#2}#3}%  
3702 }
```

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

`\GLSfirst`

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

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

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

Read in the final optional argument:

```
3706 \def\@GLSfirst@#1#2[#3] {%  
3707   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glentryfirst{#2}#3}}%  
3708 }
```

`\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`

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

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

```
3710 \newcommand*{\@glsplural}[2] [] {%
```

```
3711   \new@ifnextchar[{\@glsplural@{#1}{#2}}{\@glsplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3712 \def\@glsplural@#1#2[#3]{%
```

```
3713   \@gls@field@link{#1}{#2}{\glsentryplural{#2}#3}%
```

```
3714 }
```

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

`\Glsplural`

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

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

```
3716 \newcommand*{\@Glsplural}[2] [] {%
```

```
3717   \new@ifnextchar[{\@Glsplural@{#1}{#2}}{\@Glsplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3718 \def\@Glsplural@#1#2[#3]{%
```

```
3719   \@gls@field@link{#1}{#2}{\Glsentryplural{#2}#3}%
```

```
3720 }
```

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

`\GLSplural`

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

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

```
3722 \newcommand*{\@GLSplural}[2] [] {%
```

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

Read in the final optional argument:

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

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

```
3726 }
```

`\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`

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

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

```
3728 \newcommand*{\@glsfirstplural}[2] [] {%
```

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

Read in the final optional argument:

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

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

```
3732 }
```

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

`\Glsfirstplural`

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

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

```
3734 \newcommand*{\@Glsfirstplural}[2] [] {%
```

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

Read in the final optional argument:

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

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

```
3738 }
```

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

`\GLSfirstplural`

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

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

```
3740 \newcommand*{\@GLSfirstplural}[2] [] {%
```

```
3741   \new@ifnextchar[{\@GLSfirstplural@{#1}{#2}}{\@GLSfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3742 \def\@GLSfirstplural@#1#2[#3] {%
```

```
3743   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryfirstplural{#2}#3}}%
```

```
3744 }
```

`\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`

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

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

```
3746 \newcommand*{\@glsname}[2] [] {%
```

```
3747   \new@ifnextchar[{\@glsname@{#1}{#2}}{\@glsname@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3748 \def\@glsname@#1#2[#3] {%
```

```
3749   \@gls@field@link{#1}{#2}{\glsentryname{#2}#3}%
```

```
3750 }
```

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

`\Glsname`

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

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

```
3752 \newcommand*{\@Glsname}[2] [] {%
```

```
3753   \new@ifnextchar[{\@Glsname@{#1}{#2}}{\@Glsname@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3754 \def\@Glsname#1#2[#3]{%
3755   \@gls@field@link{#1}{#2}{\Glsentryname{#2}#3}%
3756 }
```

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

\GLSname

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

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

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

Read in the final optional argument:

```
3760 \def\@GLSname#1#2[#3]{%
3761   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentryname{#2}#3}}%
3762 }
```

\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

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

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

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

Read in the final optional argument:

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

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

\Glsdesc

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

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

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

Read in the final optional argument:

```
3772 \def\@Glsdesc#1#2[#3]{%
3773   \@gls@field@link{#1}{#2}{\Glsentrydesc{#2}#3}%
3774 }
```

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

\GLSdesc

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

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

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

Read in the final optional argument:

```
3778 \def\@GLSdesc@#1#2[#3] {%  
3779   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrydesc{#2}#3}}%  
3780 }
```

`\glsdescplural` behaves like `\gls` except it always uses the value given by the description-plural key and it doesn't mark the entry as used.

`\glsdescplural`

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

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

```
3782 \newcommand*{\@glsdescplural}[2] [] {%  
3783   \new@ifnextchar [{\@glsdescplural@{#1}{#2}}{\@glsdescplural@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3784 \def\@glsdescplural@#1#2[#3] {%  
3785   \@gls@field@link{#1}{#2}{\glsentrydescplural{#2}#3}}%  
3786 }
```

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

`\Glsdescplural`

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

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

```
3788 \newcommand*{\@Glsdescplural}[2] [] {%  
3789   \new@ifnextchar [{\@Glsdescplural@{#1}{#2}}{\@Glsdescplural@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3790 \def\@Glsdescplural@#1#2[#3] {%  
3791   \@gls@field@link{#1}{#2}{\Glsentrydescplural{#2}#3}}%  
3792 }
```

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

`\GLSdescplural`

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

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

```
3794 \newcommand*{\@GLSdescplural}[2] [] {%  
3795   \new@ifnextchar [{\@GLSdescplural@{#1}{#2}}{\@GLSdescplural@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3796 \def\@GLSdescplural@#1#2[#3] {%  
3797   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrydescplural{#2}#3}}%  
3798 }
```

`\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`

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

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

```
3800 \newcommand*{\@glssymbol}[2] [] {%
```

```
3801   \new@ifnextchar[{\@glssymbol@{#1}{#2}}{\@glssymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3802 \def\@glssymbol@#1#2[#3] {%
```

```
3803   \@gls@field@link{#1}{#2}{\glsentrysymbol{#2}#3}%
```

```
3804 }
```

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

`\Glssymbol`

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

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

```
3806 \newcommand*{\@Glssymbol}[2] [] {%
```

```
3807   \new@ifnextchar[{\@Glssymbol@{#1}{#2}}{\@Glssymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3808 \def\@Glssymbol@#1#2[#3] {%
```

```
3809   \@gls@field@link{#1}{#2}{\glsentrysymbol{#2}#3}%
```

```
3810 }
```

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

`\GLSsymbol`

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

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

```
3812 \newcommand*{\@GLSsymbol}[2] [] {%
```

```
3813   \new@ifnextchar[{\@GLSsymbol@{#1}{#2}}{\@GLSsymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3814 \def\@GLSsymbol@#1#2[#3] {%
```

```
3815   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrysymbol{#2}#3}}%
```

```
3816 }
```

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

`\glsymbolplural`

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

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

```
3818 \newcommand*{\@glsymbolplural}[2] [] {%
```

```
3819   \new@ifnextchar[{\@glsymbolplural@{#1}{#2}}{\@glsymbolplural@{#1}{#2} []}]}
```


Read in the final optional argument:

```
3820 \def\@glssymbolplural@#1#2[#3]{%
3821   \@gls@field@link{#1}{#2}{\glstrysymbolplural{#2}#3}%
3822 }
```

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

`Glssymbolplural`

```
3823 \newrobustcmd*{\Glssymbolplural}{\@gls@hyp@opt\@Glssymbolplural}
```

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

```
3824 \newcommand*{\@Glssymbolplural}[2] [] {%
3825   \new@ifnextchar[{\@Glssymbolplural@{#1}{#2}}{\@Glssymbolplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3826 \def\@Glssymbolplural@#1#2[#3]{%
3827   \@gls@field@link{#1}{#2}{\glstrysymbolplural{#2}#3}%
3828 }
```

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

`GLSsymbolplural`

```
3829 \newrobustcmd*{\GLSsymbolplural}{\@gls@hyp@opt\@GLSsymbolplural}
```

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

```
3830 \newcommand*{\@GLSsymbolplural}[2] [] {%
3831   \new@ifnextchar[{\@GLSsymbolplural@{#1}{#2}}{\@GLSsymbolplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3832 \def\@GLSsymbolplural@#1#2[#3]{%
3833   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glstrysymbolplural{#2}#3}}%
3834 }
```

`\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`

```
3835 \newrobustcmd*{\glsuseri}{\@gls@hyp@opt\@glsuseri}
```

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

```
3836 \newcommand*{\@glsuseri}[2] [] {%
3837   \new@ifnextchar[{\@glsuseri@{#1}{#2}}{\@glsuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3838 \def\@glsuseri@#1#2[#3]{%
3839   \@gls@field@link{#1}{#2}{\glstryuseri{#2}#3}%
3840 }
```

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

\Glsuseri

```
3841 \newrobustcmd*{\Glsuseri}{\@gls@hyp@opt\@Glsuseri}
```

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

```
3842 \newcommand*{\@Glsuseri}[2] [] {%
```

```
3843   \new@ifnextchar[{\@Glsuseri@{#1}{#2}}{\@Glsuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3844 \def\@Glsuseri@#1#2[#3] {%
```

```
3845   \@gls@field@link{#1}{#2}{\Glsentryuseri{#2}#3}%
```

```
3846 }
```

\Glsuseri behaves like \glsuseri except that the link text is converted to uppercase.

\GLSuseri

```
3847 \newrobustcmd*{\GLSuseri}{\@gls@hyp@opt\@GLSuseri}
```

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

```
3848 \newcommand*{\@GLSuseri}[2] [] {%
```

```
3849   \new@ifnextchar[{\@GLSuseri@{#1}{#2}}{\@GLSuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3850 \def\@GLSuseri@#1#2[#3] {%
```

```
3851   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentryuseri{#2}#3}}%
```

```
3852 }
```

\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

```
3853 \newrobustcmd*{\glsuserii}{\@gls@hyp@opt\@glsuserii}
```

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

```
3854 \newcommand*{\@glsuserii}[2] [] {%
```

```
3855   \new@ifnextchar[{\@glsuserii@{#1}{#2}}{\@glsuserii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3856 \def\@glsuserii@#1#2[#3] {%
```

```
3857   \@gls@field@link{#1}{#2}{\Glsentryuserii{#2}#3}%
```

```
3858 }
```

\Glsuserii behaves like \glsuserii except that the first letter is converted to uppercase.

\Glsuserii

```
3859 \newrobustcmd*{\Glsuserii}{\@gls@hyp@opt\@Glsuserii}
```

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

```
3860 \newcommand*{\@Glsuserii}[2] [] {%
```

```
3861   \new@ifnextchar[{\@Glsuserii@{#1}{#2}}{\@Glsuserii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3862 \def\@Glsuserii@#1#2[#3] {%
```

```
3863   \@gls@field@link{#1}{#2}{\Glsentryuserii{#2}#3}%
```

```
3864 }
```

\GLSuserii behaves like \glsuserii except that the link text is converted to uppercase.

\GLSuserii

```
3865 \newrobustcmd*{\GLSuserii}{\@gls@hyp@opt\@GLSuserii}
```

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

```
3866 \newcommand*{\@GLSuserii}[2] [] {%
```

```
3867   \new@ifnextchar[{\@GLSuserii@{#1}{#2}}{\@GLSuserii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3868 \def\@GLSuserii@#1#2[#3] {%
```

```
3869   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuserii{#2}#3}}%
```

```
3870 }
```

\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

```
3871 \newrobustcmd*{\glsuseriii}{\@gls@hyp@opt\@glsuseriii}
```

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

```
3872 \newcommand*{\@glsuseriii}[2] [] {%
```

```
3873   \new@ifnextchar[{\@glsuseriii@{#1}{#2}}{\@glsuseriii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3874 \def\@glsuseriii@#1#2[#3] {%
```

```
3875   \@gls@field@link{#1}{#2}{\glsentryuseriii{#2}#3}}%
```

```
3876 }
```

\Glsuseriii behaves like \glsuseriii except that the first letter is converted to uppercase.

\Glsuseriii

```
3877 \newrobustcmd*{\Glsuseriii}{\@gls@hyp@opt\@Glsuseriii}
```

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

```
3878 \newcommand*{\@Glsuseriii}[2] [] {%
```

```
3879   \new@ifnextchar[{\@Glsuseriii@{#1}{#2}}{\@Glsuseriii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3880 \def\@Glsuseriii@#1#2[#3] {%
```

```
3881   \@gls@field@link{#1}{#2}{\Glsentryuseriii{#2}#3}}%
```

```
3882 }
```

\GLSuseriii behaves like \glsuseriii except that the link text is converted to uppercase.

\GLSuseriii

```
3883 \newrobustcmd*{\GLSuseriii}{\@gls@hyp@opt\@GLSuseriii}
```

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

```
3884 \newcommand*{\@GLSuseriii}[2] [] {%
```

```
3885   \new@ifnextchar[{\@GLSuseriii@{#1}{#2}}{\@GLSuseriii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3886 \def\@GLSuseriii@#1#2[#3]{%
3887   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseriii{#2}#3}}%
3888 }
```

\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

```
3889 \newrobustcmd*{\glsuseriv}{\@gls@hyp@opt\@glsuseriv}
```

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

```
3890 \newcommand*{\@glsuseriv}[2][\%]
3891   \new@ifnextchar[{\@glsuseriv@{#1}{#2}}{\@glsuseriv@{#1}{#2}[]}]
```

Read in the final optional argument:

```
3892 \def\@glsuseriv@#1#2[#3]{%
3893   \@gls@field@link{#1}{#2}{\glsentryuseriv{#2}#3}%
3894 }
```

\Glsuseriv behaves like \glsuseriv except that the first letter is converted to uppercase.

\Glsuseriv

```
3895 \newrobustcmd*{\Glsuseriv}{\@gls@hyp@opt\@Glsuseriv}
```

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

```
3896 \newcommand*{\@Glsuseriv}[2][\%]
3897   \new@ifnextchar[{\@Glsuseriv@{#1}{#2}}{\@Glsuseriv@{#1}{#2}[]}]
```

Read in the final optional argument:

```
3898 \def\@Glsuseriv@#1#2[#3]{%
3899   \@gls@field@link{#1}{#2}{\Glsentryuseriv{#2}#3}%
3900 }
```

\GLSuseriv behaves like \glsuseriv except that the link text is converted to uppercase.

\GLSuseriv

```
3901 \newrobustcmd*{\GLSuseriv}{\@gls@hyp@opt\@GLSuseriv}
```

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

```
3902 \newcommand*{\@GLSuseriv}[2][\%]
3903   \new@ifnextchar[{\@GLSuseriv@{#1}{#2}}{\@GLSuseriv@{#1}{#2}[]}]
```

Read in the final optional argument:

```
3904 \def\@GLSuseriv@#1#2[#3]{%
3905   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseriv{#2}#3}}%
3906 }
```

\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

```
3907 \newrobustcmd*{\glsuserv}{\@gls@hyp@opt\@glsuserv}
```

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

```
3908 \newcommand*{\@glsuserv}[2] [] {%
3909   \new@ifnextchar [{\@glsuserv@{#1}{#2}}{\@glsuserv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3910 \def\@glsuserv@#1#2[#3] {%
3911   \@gls@field@link{#1}{#2}{\glsentryuserv{#2}#3}%
3912 }
```

\Glsuserv behaves like \glsuserv except that the first letter is converted to uppercase.

\Glsuserv

```
3913 \newrobustcmd*{\Glsuserv}{\@gls@hyp@opt\@Glsuserv}
```

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

```
3914 \newcommand*{\@GLsuserv}[2] [] {%
3915   \new@ifnextchar [{\@GLsuserv@{#1}{#2}}{\@GLsuserv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3916 \def\@GLsuserv@#1#2[#3] {%
3917   \@gls@field@link{#1}{#2}{\Glsentryuserv{#2}#3}%
3918 }
```

\GLSuserv behaves like \glsuserv except that the link text is converted to uppercase.

\GLSuserv

```
3919 \newrobustcmd*{\GLSuserv}{\@gls@hyp@opt\@GLSuserv}
```

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

```
3920 \newcommand*{\@GLSuserv}[2] [] {%
3921   \new@ifnextchar [{\@GLSuserv@{#1}{#2}}{\@GLSuserv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3922 \def\@GLSuserv@#1#2[#3] {%
3923   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuserv{#2}#3}}%
3924 }
```

\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

```
3925 \newrobustcmd*{\glsuservi}{\@gls@hyp@opt\@glsuservi}
```

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

```
3926 \newcommand*{\@glsuservi}[2] [] {%
3927   \new@ifnextchar [{\@glsuservi@{#1}{#2}}{\@glsuservi@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3928 \def\@glsuservi@#1#2[#3] {%
3929   \@gls@field@link{#1}{#2}{\glsentryuservi{#2}#3}%
3930 }
```

\Glsuservi behaves like \glsuservi except that the first letter is converted to uppercase.

\Glsuservi

```
3931 \newrobustcmd*{\Glsuservi}{\@gls@hyp@opt\@Glsuservi}
```

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

```
3932 \newcommand*{\@Glsuservi}[2] [] {%
```

```
3933   \new@ifnextchar[{\@Glsuservi@{#1}{#2}}{\@Glsuservi@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3934 \def\@Glsuservi@#1#2[#3]{%
```

```
3935   \@gls@field@link{#1}{#2}{\Glsentryuservi{#2}#3}%
```

```
3936 }
```

\Glsuservi behaves like \glsuservi except that the link text is converted to uppercase.

\GLSuservi

```
3937 \newrobustcmd*{\GLSuservi}{\@gls@hyp@opt\@GLSuservi}
```

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

```
3938 \newcommand*{\@GLSuservi}[2] [] {%
```

```
3939   \new@ifnextchar[{\@GLSuservi@{#1}{#2}}{\@GLSuservi@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3940 \def\@GLSuservi@#1#2[#3]{%
```

```
3941   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuservi{#2}#3}}%
```

```
3942 }
```

Now deal with acronym related keys. First the short form:

\acrshort

```
3943 \newrobustcmd*{\acrshort}{\@gls@hyp@opt\@ns@acrshort}
```

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

```
3944 \newcommand*{\@ns@acrshort}[2] [] {%
```

```
3945   \new@ifnextchar[{\@acrshort{#1}{#2}}{\@acrshort{#1}{#2} []}]%
```

```
3946 }
```

Read in the final optional argument:

```
3947 \def\@acrshort#1#2[#3]{%
```

```
3948   \glsdoifexists{#2}%
```

```
3949   {%
```

```
3950     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
```

```
3951     \let\glsifplural\@secondoftwo
```

```
3952     \let\glsapscase\@firstofthree
```

```
3953     \let\glsinsert\@empty
```

```
3954     \def\glscustomtext{%
```

```
3955       \acronymfont{\glsentryshort{#2}}#3%
```

```
3956     }%
```

Call \@gls@link Note that \@gls@link sets \glstype.

```
3957   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
```

```
3958 }%
```

```

3959 \glspostlinkhook
3960 }

```

\Acrshort

```

3961 \newrobustcmd*{\Acrshort}{\@gls@hyp@opt\@ns@Acrshort}

```

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

```

3962 \newcommand*{\ns@Acrshort}[2][\%
3963 \new@ifnextchar[\@Acrshort{#1}{#2}]{\@Acrshort{#1}{#2}[]}%
3964 }

```

Read in the final optional argument:

```

3965 \def\@Acrshort#1#2[#3]{%
3966 \glsdoifexists{#2}%
3967 {%
3968 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
3969 \def\glslabel{#2}%
3970 \let\glsifplural\@secondoftwo
3971 \let\glsapscase\@secondofthree
3972 \let\glsinsert\@empty
3973 \def\glscustomtext{%
3974 \acronymfont{\Glsentryshort{#2}}#3%
3975 }%

```

Call \@gls@link Note that \@gls@link sets \glstype.

```

3976 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3977 }%
3978 \glspostlinkhook
3979 }

```

\ACRshort

```

3980 \newrobustcmd*{\ACRshort}{\@gls@hyp@opt\@ns@ACRshort}

```

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

```

3981 \newcommand*{\ns@ACRshort}[2][\%
3982 \new@ifnextchar[\@ACRshort{#1}{#2}]{\@ACRshort{#1}{#2}[]}%
3983 }

```

Read in the final optional argument:

```

3984 \def\@ACRshort#1#2[#3]{%
3985 \glsdoifexists{#2}%
3986 {%
3987 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

```

```

3988 \def\glslabel{#2}%
3989 \let\glsifplural\@secondoftwo
3990 \let\glsapscase\@thirdofthree
3991 \let\glsinsert\@empty
3992 \def\glscustomtext{%
3993     \mfirstucMakeUppercase{\acronymfont{\glsentryshort{#2}}#3}%
3994 }%

```

Call \@gls@link Note that \@gls@link sets \glstype.

```

3995 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3996 }%

3997 \glspostlinkhook
3998 }

```

Short plural:

\acrshortpl

```

3999 \newrobustcmd*{\acrshortpl}{\@gls@hyp@opt\@ns@acrshortpl}

```

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

```

4000 \newcommand*{\ns@acrshortpl}[2] [] {%
4001 \new@ifnextchar[{\@acrshortpl{#1}{#2}}{\@acrshortpl{#1}{#2} []}%
4002 }

```

Read in the final optional argument:

```

4003 \def\@acrshortpl#1#2[#3]{%
4004 \glsdoifexists{#2}%
4005 {%

4006 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

4007 \def\glslabel{#2}%
4008 \let\glsifplural\@firstoftwo
4009 \let\glsapscase\@firstofthree
4010 \let\glsinsert\@empty
4011 \def\glscustomtext{%
4012 \acronymfont{\glsentryshortpl{#2}}#3%
4013 }%

```

Call \@gls@link Note that \@gls@link sets \glstype.

```

4014 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4015 }%

4016 \glspostlinkhook
4017 }

```

\Acrshortpl

```

4018 \newrobustcmd*{\Acrshortpl}{\@gls@hyp@opt\@ns@Acrshortpl}

```


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

```
4019 \newcommand*{\ns@Acrshortpl}[2][\%
4020   \new@ifnextchar[\@Acrshortpl{#1}{#2}]{\@Acrshortpl{#1}{#2}[]}%
4021 }
```

Read in the final optional argument:

```
4022 \def\@Acrshortpl#1#2[#3]{%
4023   \glsdoifexists{#2}%
4024   {%
4025     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4026     \def\glslabel{#2}%
4027     \let\glsifplural\@firstoftwo
4028     \let\glscapscase\@secondofthree
4029     \let\glsinsert\@empty
4030     \def\glscustomtext{%
4031       \acronymfont{\Glsentryshortpl{#2}}#3%
4032     }%
4033     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4034     \let\glslabel{#2}%
4035     \let\glsifplural\@firstoftwo
4036     \let\glscapscase\@thirdofthree
4037     \let\glsinsert\@empty
4038     \def\glscustomtext{%
4039       \mfirstucMakeUppercase{\acronymfont{\Glsentryshortpl{#2}}#3}%
4040     }%
4041   }
```

Call \@gl@link Note that \@gl@link sets \glstype.

```
4033   \@gl@link{#1}{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4034   }%
4035 \glspostlinkhook
4036 }
```

\ACRshortpl

```
4037 \newrobustcmd*{\ACRshortpl}{\@gl@hyp@opt\ns@ACRshortpl}
```

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

```
4038 \newcommand*{\ns@ACRshortpl}[2][\%
4039   \new@ifnextchar[\@ACRshortpl{#1}{#2}]{\@ACRshortpl{#1}{#2}[]}%
4040 }
```

Read in the final optional argument:

```
4041 \def\@ACRshortpl#1#2[#3]{%
4042   \glsdoifexists{#2}%
4043   {%
4044     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4045     \def\glslabel{#2}%
4046     \let\glsifplural\@firstoftwo
4047     \let\glscapscase\@thirdofthree
4048     \let\glsinsert\@empty
4049     \def\glscustomtext{%
4050       \mfirstucMakeUppercase{\acronymfont{\Glsentryshortpl{#2}}#3}%
4051     }%
4052     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4053     \let\glslabel{#2}%
4054     \let\glsifplural\@firstoftwo
4055     \let\glscapscase\@thirdofthree
4056     \let\glsinsert\@empty
4057     \def\glscustomtext{%
4058       \mfirstucMakeUppercase{\acronymfont{\Glsentryshortpl{#2}}#3}%
4059     }%
4060   }
```

Call \@gls@link Note that \@gls@link sets \glstype.

```
4052   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%  
4053   }%  
  
4054   \glspostlinkhook  
4055 }
```

\acrlong

```
4056 \newrobustcmd*{\acrlong}{\@gls@hyp@opt\@ns@acrlong}
```

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

```
4057 \newcommand*{\ns@acrlong}[2][ ]{%  
4058   \new@ifnextchar[{\@acrlong{#1}{#2}}{\@acrlong{#1}{#2}[ ]}%  
4059 }
```

Read in the final optional argument:

```
4060 \def\@acrlong#1#2[#3]{%  
4061   \glsdoifexists{#2}%  
4062   {%  
  
4063     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper  
  
4064     \def\glslabel{#2}%  
4065     \let\glsifplural\@secondoftwo  
4066     \let\glsapscase\@firstofthree  
4067     \let\glsinsert\@empty
```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```
4068   \def\glscustomtext{%  
4069     \glsentrylong{#2}#3%  
4070   }%
```

Call \@gls@link Note that \@gls@link sets \glstype.

```
4071   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%  
4072   }%  
  
4073   \glspostlinkhook  
4074 }
```

\Acrlong

```
4075 \newrobustcmd*{\Acrlong}{\@gls@hyp@opt\@ns@Acrlong}
```

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

```
4076 \newcommand*{\ns@Acrlong}[2][ ]{%  
4077   \new@ifnextchar[{\@Acrlong{#1}{#2}}{\@Acrlong{#1}{#2}[ ]}%  
4078 }
```

Read in the final optional argument:

```
4079 \def\@Acrlong#1#2[#3]{%  
4080   \glsdoifexists{#2}%  
4081   {%
```

```

4082 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4083 \def\glslabel{#2}%
4084 \let\gl@ifplural\@secondoftwo
4085 \let\gl@scapscase\@secondofthree
4086 \let\gl@insert\@empty

```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```

4087 \def\glscustomtext{%
4088 \gl@entrylong{#2}#3%
4089 }%

```

Call \@gl@link. Note that \@gl@link sets \glstype.

```

4090 \@gl@link[#1]{#2}{\csname gls\glstype @entryfmt\endcsname}%
4091 }%
4092 \gl@postlinkhook
4093 }

```

\ACRlong

```

4094 \newrobustcmd*{\ACRlong}{\@gl@hyp@opt\@ns@ACRlong}

```

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

```

4095 \newcommand*{\ns@ACRlong}[2][\%
4096 \new@ifnextchar{\@ACRlong{#1}{#2}}{\@ACRlong{#1}{#2}[]}%
4097 }

```

Read in the final optional argument:

```

4098 \def\@ACRlong#1#2[#3]{%
4099 \gl@doifexists{#2}%
4100 {%

```

```

4101 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4102 \def\glslabel{#2}%
4103 \let\gl@ifplural\@secondoftwo
4104 \let\gl@scapscase\@thirdofthree
4105 \let\gl@insert\@empty

```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```

4106 \def\glscustomtext{%
4107 \mfirstucMakeUppercase{\gl@entrylong{#2}#3}%
4108 }%

```

Call \@gl@link. Note that \@gl@link sets \glstype.

```

4109 \@gl@link[#1]{#2}{\csname gls\glstype @entryfmt\endcsname}%
4110 }%
4111 \gl@postlinkhook
4112 }

```

Short plural:

`\acrlongpl`

```
4113 \newrobustcmd*{\acrlongpl}{\@gls@hyp@opt\@ns@acrlongpl}
```

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

```
4114 \newcommand*{\ns@acrlongpl}[2][\%  
4115   \new@ifnextchar[\@acrlongpl{#1}{#2}]{\@acrlongpl{#1}{#2}[]}%  
4116 }
```

Read in the final optional argument:

```
4117 \def\@acrlongpl#1#2[#3]{%  
4118   \glsdoifexists{#2}%  
4119   {%  
  
4120     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper  
  
4121     \def\glslabel{#2}%  
4122     \let\glsifplural\@firstoftwo  
4123     \let\glsapscase\@firstofthree  
4124     \let\glsinsert\@empty
```

Bug fix v4.02 removed `\acronymfont` from `\glscustomtext` (`\acronymfont` only designed for short form).

```
4125   \def\glscustomtext{%  
4126     \glsentrylongpl{#2}#3%  
4127   }%
```

Call `\@gls@link`. Note that `\@gls@link` sets `\glsstyle`.

```
4128   \@gls@link[#1]{#2}{\csname gls@\glsstyle @entryfmt\endcsname}%  
4129   }%  
  
4130   \glspostlinkhook  
4131 }
```

`\Acrlongpl`

```
4132 \newrobustcmd*{\Acrlongpl}{\@gls@hyp@opt\@ns@Acrlongpl}
```

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

```
4133 \newcommand*{\ns@Acrlongpl}[2][\%  
4134   \new@ifnextchar[\@Acrlongpl{#1}{#2}]{\@Acrlongpl{#1}{#2}[]}%  
4135 }
```

Read in the final optional argument:

```
4136 \def\@Acrlongpl#1#2[#3]{%  
4137   \glsdoifexists{#2}%  
4138   {%  
  
4139     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
```

```

4140 \def\glslabel{#2}%
4141 \let\glsifplural\@firstoftwo
4142 \let\glsupcase\@secondofthree
4143 \let\glsinsert\@empty

```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```

4144 \def\glscustomtext{%
4145 \Glsentrylongpl{#2}#3%
4146 }%

```

Call \@gls@link. Note that \@gls@link sets \glsstyle.

```

4147 \@gls@link[#1]{#2}{\csname gls@\glsstyle @entryfmt\endcsname}%
4148 }%

```

```

4149 \glspostlinkhook
4150 }

```

\ACRlongpl

```

4151 \newrobustcmd*{\ACRlongpl}{\@gls@hyp@opt\@ns@ACRlongpl}

```

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

```

4152 \newcommand*{\@ns@ACRlongpl}[2][{}]{%
4153 \new@ifnextchar[{\@ACRlongpl{#1}{#2}}{\@ACRlongpl{#1}{#2}[]}%
4154 }

```

Read in the final optional argument:

```

4155 \def\@ACRlongpl#1#2[#3]{%
4156 \glsdoifexists{#2}%
4157 {%

```

```

4158 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

```

```

4159 \def\glslabel{#2}%
4160 \let\glsifplural\@firstoftwo
4161 \let\glsupcase\@thirdofthree
4162 \let\glsinsert\@empty

```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```

4163 \def\glscustomtext{%
4164 \mfirstucMakeUppercase{\Glsentrylongpl{#2}#3}%
4165 }%

```

Call \@gls@link. Note that \@gls@link sets \glsstyle.

```

4166 \@gls@link[#1]{#2}{\csname gls@\glsstyle @entryfmt\endcsname}%
4167 }%

```

```

4168 \glspostlinkhook
4169 }

```

Displaying entry details without adding information to the glossary

These commands merely display entry information without adding entries in the associated file or having hyperlinks.

`gls@entry@field` Generic version.

```
\@gls@entry@field{<label>}{<field>}
```

```
4170 \newcommand*{\@gls@entry@field}[2]{%
4171   \csname glo@glsdetoklabel{#1}@#2\endcsname
4172 }
```

`glsletentryfield`

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

```
4173 \newcommand*{\glsletentryfield}[3]{%
4174   \letcs{#1}{glo@glsdetoklabel{#2}@#3}%
4175 }
```

`Gls@entry@field` Generic first letter uppercase version.

```
\@Gls@entry@field{<label>}{<field>}
```

```
4176 \newcommand*{\@Gls@entry@field}[2]{%
4177   \glsdoifexistsordo{#1}%
4178   {%
4179     \letcs\@glo@text{glo@glsdetoklabel{#1}@#2}%
4180     \ifdef\@glo@text
4181     {%
4182       \xmakefirstuc{\@glo@text}%
4183     }%
4184     {%
4185       ??\PackageError{glossaries}{The field ‘#2’ doesn’t exist for glossary
4186       entry ‘\glsdetoklabel{#1}’}{Check you have correctly spelt the entry
4187       label and the field name}%
4188     }%
4189   }%
4190   {%
4191     ??%
4192   }%
4193 }
```

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
4194 \newcommand*{\glsentryname}[1]{\@gls@entry@field{#1}{name}}

\Glsentryname
4195 \newrobustcmd*{\Glsentryname}[1]{%
4196   \@Gls@entryname{#1}%
4197 }

\@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:
4198 \newcommand*{\@Gls@entryname}[1]{%
4199   \@Gls@entry@field{#1}{name}%
4200 }

ls@acentryname Now the behaviour when \setacronymstyle is used:
4201 \newcommand*{\@Gls@acentryname}[1]{%
4202   \ifglshaslong{#1}%
4203   {%
4204     \letcs\@glo@text{glo@\glsdetoklabel{#1}@name}%
4205     \expandafter\@gls@getbody\@glo@text{}\@nil
4206     \expandafter\ifx\@gls@body\glsentrylong\relax
4207       \expandafter\Glsentrylong\@gls@rest
4208     \else
4209       \expandafter\ifx\@gls@body\glsentryshort\relax
4210         \expandafter\Glsentryshort\@gls@rest
4211       \else
4212         \expandafter\ifx\@gls@body\acronymfont\relax
          Temporarily make \glsentryshort behave like \Glsentryshort. (This is on the assump-
          tion that the argument of \acronymfont is \glsentryshort{\langle label \rangle}, as that's the behaviour
          of the predefined acronym styles.) This is scoped to localise the effect of the assignment.
4213         {%
4214           \let\glsentryshort\Glsentryshort
4215           \@glo@text
4216         }%
4217       \else
4218         \xmakefirstuc{\@glo@text}%
4219       \fi
4220     \fi
4221   \fi
4222 }%
4223 {%
  Not an acronym
4224   \@Gls@entry@field{#1}{name}%
4225 }%
4226 }

```

Get the entry description (as specified by the description when the entry was defined). The argument is the label associated with the entry. Note that unless you used `description=false` in the `sanitize` package option you may get unexpected results if the description key contained any commands.

`\glentrydesc`

```
4227 \newcommand*{\glentrydesc}[1]{\@gls@entry@field{#1}{desc}}
```

`\Glsentrydesc`

```
4228 \newrobustcmd*{\Glsentrydesc}[1]{%
4229   \@Gls@entry@field{#1}{desc}%
4230 }
```

Plural form:

`entrydescplural`

```
4231 \newcommand*{\glentrydescplural}[1]{%
4232   \@gls@entry@field{#1}{descplural}%
4233 }
```

`entrydescplural`

```
4234 \newrobustcmd*{\Glsentrydescplural}[1]{%
4235   \@Gls@entry@field{#1}{descplural}%
4236 }
```

Get the entry text, as specified by the text key when the entry was defined. The argument is the label associated with the entry:

`\glentrytext`

```
4237 \newcommand*{\glentrytext}[1]{\@gls@entry@field{#1}{text}}
```

`\Glsentrytext`

```
4238 \newrobustcmd*{\Glsentrytext}[1]{%
4239   \@Gls@entry@field{#1}{text}%
4240 }
```

Get the plural form:

`\glentryplural`

```
4241 \newcommand*{\glentryplural}[1]{%
4242   \@gls@entry@field{#1}{plural}%
4243 }
```

`\Glsentryplural`

```
4244 \newrobustcmd*{\Glsentryplural}[1]{%
4245   \@Gls@entry@field{#1}{plural}%
4246 }
```


Get the symbol associated with this entry. The argument is the label associated with the entry.

`\glsentrysymbol`

```
4247 \newcommand*{\glsentrysymbol}[1]{%
4248   \@gls@entry@field{#1}{symbol}%
4249 }
```

`\Glsentrysymbol`

```
4250 \newrobustcmd*{\Glsentrysymbol}[1]{%
4251   \@Gls@entry@field{#1}{symbol}%
4252 }
```

Plural form:

`trysymbolplural`

```
4253 \newcommand*{\glsentrysymbolplural}[1]{%
4254   \@gls@entry@field{#1}{symbolplural}%
4255 }
```

`trysymbolplural`

```
4256 \newrobustcmd*{\Glsentrysymbolplural}[1]{%
4257   \@Gls@entry@field{#1}{symbolplural}%
4258 }
```

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`

```
4259 \newcommand*{\glsentryfirst}[1]{%
4260   \@gls@entry@field{#1}{first}%
4261 }
```

`\Glsentryfirst`

```
4262 \newrobustcmd*{\Glsentryfirst}[1]{%
4263   \@Gls@entry@field{#1}{first}%
4264 }
```

Get the plural form (as specified by the firstplural key when the entry was defined).

`ntryfirstplural`

```
4265 \newcommand*{\glsentryfirstplural}[1]{%
4266   \@gls@entry@field{#1}{firstpl}%
4267 }
```

`ntryfirstplural`

```
4268 \newrobustcmd*{\Glsentryfirstplural}[1]{%
4269   \@Gls@entry@field{#1}{firstpl}%
4270 }
```

sentrytitlecase

```
4271 \newrobustcmd*{\@glsentrytitlecase}[2]{%
4272   \glsfieldfetch{#1}{#2}{\@gls@value}%
4273   \xcapitalisewords{\@gls@value}%
4274 }
4275 \ifdef\texorpdfstring
4276 {
4277   \newcommand*{\glsentrytitlecase}[2]{%
4278     \texorpdfstring
4279       {\@glsentrytitlecase{#1}{#2}}%
4280       {\@gls@entry@field{#1}{#2}}%
4281   }
4282 }
4283 {
4284   \newcommand*{\glsentrytitlecase}[2]{\@glsentrytitlecase{#1}{#2}}
4285 }
```

Display the glossary type with which this entry is associated (as specified by the type key used when the entry was defined)

\glsentrytype

```
4286 \newcommand*{\glsentrytype}[1]{\@gls@entry@field{#1}{type}}
```

Display the sort text used for this entry. Note that the sort key is sanitize, so unexpected results may occur if the sort key contained commands.

\glsentrysort

```
4287 \newcommand*{\glsentrysort}[1]{%
4288   \@gls@entry@field{#1}{sort}%
4289 }
```

\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.

```
4290 \newcommand*{\glsentryuseri}[1]{%
4291   \@gls@entry@field{#1}{useri}%
4292 }
```

\Glsentryuseri

```
4293 \newrobustcmd*{\Glsentryuseri}[1]{%
4294   \@Gls@entry@field{#1}{useri}%
4295 }
```

\glsentryuserii Get the second user key (as specified by the user2 when the entry was defined). The argument is the label associated with the entry.

```
4296 \newcommand*{\glsentryuserii}[1]{%
4297   \@gls@entry@field{#1}{userii}%
4298 }
```

```

\Glsentryuserii
4299 \newrobustcmd*{\Glsentryuserii}[1]{%
4300   \@Gls@entry@field{#1}{userii}%
4301 }

\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.
4302 \newcommand*{\glentryuseriii}[1]{%
4303   \@Gls@entry@field{#1}{useriii}%
4304 }

Glsentryuseriii
4305 \newrobustcmd*{\Glsentryuseriii}[1]{%
4306   \@Gls@entry@field{#1}{useriii}%
4307 }

\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.
4308 \newcommand*{\glentryuseriv}[1]{%
4309   \@Gls@entry@field{#1}{useriv}%
4310 }

\Glsentryuseriv
4311 \newrobustcmd*{\Glsentryuseriv}[1]{%
4312   \@Gls@entry@field{#1}{useriv}%
4313 }

\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.
4314 \newcommand*{\glentryuserv}[1]{%
4315   \@Gls@entry@field{#1}{userv}%
4316 }

\Glsentryuserv
4317 \newrobustcmd*{\Glsentryuserv}[1]{%
4318   \@Gls@entry@field{#1}{userv}%
4319 }

\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.
4320 \newcommand*{\glentryuservi}[1]{%
4321   \@Gls@entry@field{#1}{uservi}%
4322 }

\Glsentryuservi
4323 \newrobustcmd*{\Glsentryuservi}[1]{%
4324   \@Gls@entry@field{#1}{uservi}%
4325 }

```

`\glsentryshort` Get the short key (as specified by the short the entry was defined). The argument is the label associated with the entry.

```
4326 \newcommand*{\glsentryshort}[1]{\@gls@entry@field{#1}{short}}
```

`\Glsentryshort`

```
4327 \newrobustcmd*{\Glsentryshort}[1]{%
4328   \@Gls@entry@field{#1}{short}%
4329 }
```

`\glsentryshortpl` Get the short plural key (as specified by the shortplural the entry was defined). The argument is the label associated with the entry.

```
4330 \newcommand*{\glsentryshortpl}[1]{\@gls@entry@field{#1}{shortpl}}
```

`\Glsentryshortpl`

```
4331 \newrobustcmd*{\Glsentryshortpl}[1]{%
4332   \@Gls@entry@field{#1}{shortpl}%
4333 }
```

`\glsentrylong` Get the long key (as specified by the long the entry was defined). The argument is the label associated with the entry.

```
4334 \newcommand*{\glsentrylong}[1]{\@gls@entry@field{#1}{long}}
```

`\Glsentrylong`

```
4335 \newrobustcmd*{\Glsentrylong}[1]{%
4336   \@Gls@entry@field{#1}{long}%
4337 }
```

`\glsentrylongpl` Get the long plural key (as specified by the longplural the entry was defined). The argument is the label associated with the entry.

```
4338 \newcommand*{\glsentrylongpl}[1]{\@gls@entry@field{#1}{longpl}}
```

`\Glsentrylongpl`

```
4339 \newrobustcmd*{\Glsentrylongpl}[1]{%
4340   \@Gls@entry@field{#1}{longpl}%
4341 }
```

Short cut macros to access full form:

`\glsentryfull`

```
4342 \newcommand*{\glsentryfull}[1]{%
4343   \acrfullformat{\glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%
4344 }
```

`\Glsentryfull`

```
4345 \newrobustcmd*{\Glsentryfull}[1]{%
4346   \acrfullformat{\Glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%
4347 }
```

`\glsentryfullpl`

```
4348 \newcommand*{\glsentryfullpl}[1]{%
4349   \acrfullformat{\glsentrylongpl{#1}}{\acronymfont{\glsentryshortpl{#1}}}%
4350 }
```

`\Glsentryfullpl`

```
4351 \newrobustcmd*{\Glsentryfullpl}[1]{%
4352   \acrfullformat{\Glsentrylongpl{#1}}{\acronymfont{\glsentryshortpl{#1}}}%
4353 }
```

`entrynumberlist` Displays the number list as is.

```
4354 \newcommand*{\glsentrynumberlist}[1]{%
4355   \glsdoifexists{#1}%
4356   {%
4357     \@gls@entry@field{#1}{numberlist}%
4358   }%
4359 }
```

`splaynumberlist` Formats the number list for the given entry label. Doesn't work with hyperref.

```
4360 \@ifpackageloaded{hyperref} {%
4361   \newcommand*{\glsdisplaynumberlist}[1]{%
4362     \GlossariesWarning
4363     {%
4364       \string\glsdisplaynumberlist\space
4365       doesn't work with hyperref.^^JUsing
4366       \string\glsentrynumberlist\space instead%
4367     }%
4368     \glsentrynumberlist{#1}%
4369   }%
4370 }%
4371 {%
4372   \newcommand*{\glsdisplaynumberlist}[1]{%
4373     \glsdoifexists{#1}%
4374     {%
4375       \bgroup

4376       \edef\@glo@label{\glsdetoklabel{#1}}%
4377       \let\@org@glsglsnumberformat\glsglsnumberformat
4378       \def\glsglsnumberformat##1{##1}%
4379       \protected@edef\the@numberlist{%
4380         \csname glo@\@glo@label @numberlist\endcsname}%
4381       \def\@gls@numlist@sep{}%
4382       \def\@gls@numlist@nextsep{}%
4383       \def\@gls@numlist@lastsep{}%
4384       \def\@gls@thislist{}%
4385       \def\@gls@donext@def{}%
4386       \renewcommand\do[1]{%
4387         \protected@edef\@gls@thislist{%
4388           \@gls@thislist
```

```

4389         \noexpand\@gls@numlist@sep
4390         ##1%
4391     }%
4392     \let\@gls@numlist@sep\@gls@numlist@nextsep
4393     \def\@gls@numlist@nextsep{\@glsnumlistsep}%
4394     \@gls@donext@def
4395     \def\@gls@donext@def{%
4396         \def\@gls@numlist@lastsep{\@glsnumlistlastsep}%
4397     }%
4398 }%
4399 \expandafter \glsnumlistparser \expandafter{\the@numberlist}%
4400 \let\@gls@numlist@sep\@gls@numlist@lastsep
4401 \@gls@thislist
4402 \egroup
4403 }%
4404 }
4405 }

```

\glsnumlistsep

```

4406 \newcommand*{\glsnumlistsep}{, }

```

\glsnumlistlastsep

```

4407 \newcommand*{\glsnumlistlastsep}{ \& }

```

\gls hyperlink Provide a hyperlink to a glossary entry without adding information to the glossary file. The entry needs to be added using a command like \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.

```

4408 \newcommand*{\gls hyperlink}[2][\glsentrytext{\@glo@label}]{%
4409   \def\@glo@label{#2}%
4410   \@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:

```

4411 \define@key{glossadd}{counter}{\def\@gls@counter{#1}}
4412 \define@key{glossadd}{format}{\def\@glsnumberformat{#1}}

```

This key is only used by \glsaddall:

```

4413 \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`

```
4414 \newrobustcmd*{\glsadd}[2] [] {%
    Need to move to horizontal mode if not already in it, but only if not in preamble.
4415   \@gls@adjustmode
4416   \glsdoifexists{#2}%
4417   {%
4418     \def\@glsnumberformat{glsnumberformat}%
4419     \edef\@gls@counter{\csname glo@\glsdetoklabel{#2}@counter\endcsname}%
4420     \setkeys{glossadd}{#1}%
    Store the entry's counter in \theglsentrycounter
4421     \@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.
4422     \@do@wrglossary{#2}%
4423   }%
4424 }
```

`@gls@adjustmode`

```
4425 \newcommand*{\@gls@adjustmode}{}
4426 \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`

```
4427 \newrobustcmd*{\glsaddall}[1] [] {%
4428   \edef\@glo@type{\@glo@types}%
4429   \setkeys{glossadd}{#1}%
4430   \forallglsentries[\@glo@type]{\@glo@entry}{%
4431     \glsadd[#1]{\@glo@entry}%
4432   }%
4433 }
```

`\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.

```
4434 \newrobustcmd*{\glsaddallunused}[1] [\@glo@types] {%
4435   \forallglsentries[#1]{\@glo@entry}%
4436   {%
4437     \ifglsused{\@glo@entry}{\glsadd[format=glsignore]{\@glo@entry}}%
```

```
4438 }%
4439 }
```

`\glsignore`

```
4440 \newcommand*{\glsignore}[1]{}
```

1.13 Creating associated files

The `\writeist` command creates the associated customized `.ist` makeindex style file. While defining this command, some characters have their catcodes temporarily changed to ensure they get written to the `.ist` file correctly. The makeindex actual character (usually `@`) is redefined to be a `?`, to allow internal commands to be written to the glossary file output file.

The special characters are stored in `\@gls@actualchar`, `\@gls@encapchar`, `\@gls@levelchar` and `\@gls@quotechar` to make them easier to use later, but don't change these values, because the characters are encoded in the command definitions that are used to escape the special characters (which means that the user no longer needs to worry about makeindex special characters).

The symbols and numbers label for group headings are hardwired into the `.ist` file as `glssymbols` and `glsnumbers`, the group titles can be translated (so that `\glssymbolsgroupname` replaces `glssymbols` and `\glsnumbersgroupname` replaces `glsnumbers`) using the command `\glsgetgrouptitle` which is defined in `.` This is done to prevent any problem characters in `\glssymbolsgroupname` and `\glsnumbersgroupname` from breaking hyperlinks.

`\glsopenbrace` Define `\glsopenbrace` to make it easier to write an opening brace to a file.

```
4441 \edef\glsopenbrace{\expandafter\@gobble\string\{}
```

`\glsclosebrace` Define `\glsclosebrace` to make it easier to write an opening brace to a file.

```
4442 \edef\glsclosebrace{\expandafter\@gobble\string\}}
```

`\glsbackslash` Define `\glsbackslash` to make it easier to write a backslash to a file.

```
4443 \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.

```
4444 \edef\glsquote#1{\string"#1\string"}
```

`\glsperscentchar` Define `\glsperscentchar` to make it easier to write a percent character to a file.

```
4445 \edef\glsperscentchar{\expandafter\@gobble\string\%}
```

`\glstildechar` Define `\glstildechar` to make it easier to write a tilde character to a file.

```
4446 \edef\glstildechar{\string~}
```

`@glsfirstletter` Define the first letter to come after the digits 0,...,9. Only required for `xindy`.

```
4447 \ifglsxindy
```

```
4448 \newcommand*{\@glsfirstletter}{A}
```

```
4449 \fi
```


letterAfterDigits Sets the first letter to come after the digits 0,...,9.

```

4450 \ifglsxindy
4451   \newcommand*{\GlsSetXdyFirstLetterAfterDigits}[1]{%
4452     \renewcommand*{\@glstfirstletter}{#1}}
4453 \else
4454   \newcommand*{\GlsSetXdyFirstLetterAfterDigits}[1]{%
4455     \glstnoindywarning\GlsSetXdyFirstLetterAfterDigits}
4456 \fi

```

\@glstminrange Define the minimum number of successive location references to merge into a range.

```

4457 \newcommand*{\@glstminrange}{2}

```

yMinRangeLength Set the minimum range length. The value must either be none or a positive integer. The glossaries package doesn't check if the argument is valid, that is left to xindy.

```

4458 \ifglsxindy
4459   \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
4460     \renewcommand*{\@glstminrange}{#1}}
4461 \else
4462   \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
4463     \glstnoindywarning\GlsSetXdyMinRangeLength}
4464 \fi

```

\writeist

```

4465 \ifglsxindy
    Code to use if xindy is required.
4466   \def\writeist{%
    Define write register if not already defined
4467     \ifundef{\glswrite}{\newwrite\glswrite}{}%
    Update attributes list
4468     \@glst@addpredefinedattributes
    Open the file.
4469     \openout\glswrite=\istfilename
    Write header comment at the start of the file
4470     \write\glswrite{;; xindy style file created by the glossaries
4471       package}%
4472     \write\glswrite{;; for document '\jobname' on
4473       \the\year-\the\month-\the\day}%
    Specify the required styles
4474     \write\glswrite{^^J; required styles^^J}
4475     \@for\@xdystyle:=\@xdyrequiredstyles\do{%
4476       \ifx\@xdystyle\@empty
4477       \else
4478         \protected@write\glswrite{{(require
4479           \string"\@xdystyle.xdy\string")}}%
4480       \fi
4481     }%

```

List the allowed attributes (possible values used by the format key)

```
4482 \write\glswrite{^^J%
4483 ; list of allowed attributes (number formats)^^J}%
4484 \write\glswrite{(define-attributes ((\@xdyattributes)))}%
```

Define any additional alphabets

```
4485 \write\glswrite{^^J; user defined alphabets^^J}%
4486 \write\glswrite{\@xdyuseralphabets}%
```

Define location classes.

```
4487 \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.

```
4488 \@for\@gls@classI:=\@gls@xdy@locationlist\do{%
```

Case were $\langle Hprefix \rangle$ is empty:

```
4489 \protected@write\glswrite{}\{(define-location-class
4490 \string"\@gls@classI\string"^^J\space\space\space
4491 (
4492 :sep "{"{"
4493 \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
4494 :sep "}"
4495 )
4496 ^^J\space\space\space
4497 :min-range-length \@glsminrange^^J%
4498 )
4499 }%
```

Nested iteration over all classes:

```
4500 {%
4501 \@for\@gls@classII:=\@gls@xdy@locationlist\do{%
4502 \protected@write\glswrite{}\{(define-location-class
4503 \string"\@gls@classII-\@gls@classI\string"
4504 ^^J\space\space\space
4505 (
4506 :sep "{"
4507 \csname @gls@xdy@Lclass@\@gls@classII\endcsname\space
4508 :sep "{"{"
4509 \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
4510 :sep "}"
4511 )
4512 ^^J\space\space\space
4513 :min-range-length \@glsminrange^^J%
4514 )
4515 }%
4516 }%
4517 }%
4518 }%
```

User defined location classes (needs checking for new location format).

```

4519 \write\glswrite{^^J; user defined location classes}%
4520 \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.)

```

4521 \write\glswrite{^^J; define cross-reference class^^J}%
4522 \write\glswrite{(define-crossref-class \string"see\string"
4523 :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.)

```

4524 \write\glswrite{(markup-crossref-list
4525 :class \string"see\string"^^J\space\space\space
4526 :open \string"\string\glsseeformat\string"
4527 :close \string"{}\string")}%

```

List the order to sort the classes.

```

4528 \write\glswrite{^^J; define the order of the location classes}%
4529 \write\glswrite{(define-location-class-order
4530 (\@xdylocationclassorder))}%

```

Specify what to write to the start and end of the glossary file.

```

4531 \write\glswrite{^^J; define the glossary markup^^J}%
4532 \write\glswrite{(markup-index^^J\space\space\space
4533 :open \string"\string
4534 \glossarysection[\string\glossarytoctitle]{\string
4535 \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`)

```

4536 \@for\@this@ctr:=\@xdycounters\do{%
4537 {%
4538 \@for\@this@attr:=\@xdyattributelist\do{%
4539 \protected@write\glswrite{}\string\providecommand*%
4540 \expandafter\string
4541 \csname glsX\@this@ctr X\@this@attr\endcsname[2]%
4542 {%
4543 \string\setentrycounter
4544 [\expandafter\@gobble\string\#1]{\@this@ctr}%
4545 \expandafter\string
4546 \csname\@this@attr\endcsname
4547 {\expandafter\@gobble\string\#2}%
4548 }%
4549 }%
4550 }%
4551 }%
4552 }%

```

Add the end part of the open tag and the rest of the markup-index information:

```
4553 \write\glswrite{%  
4554 \string\begin  
4555 {theglossary}\string\glossaryheader\glstildechar n\string" ^^J\space  
4556 \space\space:close \string"\glpercentchar\glstildechar n\string  
4557 \end{theglossary}\string\glossarypostamble  
4558 \glstildechar n\string" ^^J\space\space\space  
4559 :tree)}}%
```

Specify what to put between letter groups

```
4560 \write\glswrite{(markup-letter-group-list  
4561 :sep \string"\string\glsgroupskip\glstildechar n\string")}%
```

Specify what to put between entries

```
4562 \write\glswrite{(markup-indexentry  
4563 :open \string"\string\relax \string\glresetentrylist  
4564 \glstildechar n\string")}%
```

Specify how to format entries

```
4565 \write\glswrite{(markup-locclass-list :open  
4566 \string"\glsoopenbrace\string\glossaryentrynumbers  
4567 \glsoopenbrace\string\relax\space \string"^^J\space\space\space  
4568 :sep \string", \string"  
4569 :close \string"\glsclosebrace\glsclosebrace\string")}%
```

Specify how to separate location numbers

```
4570 \write\glswrite{(markup-locref-list  
4571 :sep \string"\string\delimN\space\string")}%
```

Specify how to indicate location ranges

```
4572 \write\glswrite{(markup-range  
4573 :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.

```
4574 \@onelevel@sanitize\gls@suffixF  
4575 \@onelevel@sanitize\gls@suffixFF  
  
4576 \ifx\gls@suffixF\@empty  
4577 \else  
4578 \write\glswrite{(markup-range  
4579 :close "\gls@suffixF" :length 1 :ignore-end)}}%  
4580 \fi  
4581 \ifx\gls@suffixFF\@empty  
4582 \else  
4583 \write\glswrite{(markup-range  
4584 :close "\gls@suffixFF" :length 2 :ignore-end)}}%  
4585 \fi
```

Specify how to format locations.

```
4586 \write\glswrite{^^J; define format to use for locations^^J}%  
4587 \write\glswrite{\@xdylocref}%
```

Specify how to separate letter groups.

```
4588 \write\glswrite{^^J; define letter group list format^^J}%
4589 \write\glswrite{(markup-letter-group-list
4590 :sep \string"\string\glsgroupskip\glstildechar n\string")}%
```

Define letter group headings.

```
4591 \write\glswrite{^^J; letter group headings^^J}%
4592 \write\glswrite{(markup-letter-group
4593 :open-head \string"\string\glsgroupheading
4594 \glsoopenbrace\string"^^J\space\space\space
4595 :close-head \string"\glsclosebrace\string")}%
```

Define additional letter groups.

```
4596 \write\glswrite{^^J; additional letter groups^^J}%
4597 \write\glswrite{@xdylettergroups}%
```

Define additional sort rules

```
4598 \write\glswrite{^^J; additional sort rules^^J}
4599 \write\glswrite{@xdysortrules}%
```

Close the style file

```
4600 \closeout\glswrite
```

Suppress any further calls.

```
4601 \let\writeist\relax
4602 }
4603 \else
```

Code to use if makeindex is required.

```
4604 \edef\@gls@actualchar{\string?}
4605 \edef\@gls@encapchar{\string|}
4606 \edef\@gls@levelchar{\string!}
4607 \edef\@gls@quotechar{\string"}
4608 \def\writeist{\relax
4609 \ifundefined{\glswrite}{\newwrite\glswrite}{\relax
4610 \openout\glswrite=\istfilename
4611 \write\glswrite{\glspersentchar\space makeindex style file
4612 created by the glossaries package}
4613 \write\glswrite{\glspersentchar\space for document
4614 '\jobname' on \the\year-\the\month-\the\day}
4615 \write\glswrite{actual '@gls@actualchar'}
4616 \write\glswrite{encap '@gls@encapchar'}
4617 \write\glswrite{level '@gls@levelchar'}
4618 \write\glswrite{quote '@gls@quotechar'}
4619 \write\glswrite{keyword \string"\string\glossaryentry\string"}
4620 \write\glswrite{preamble \string"\string\glossarysection[\string
4621 \glossarytoctitle]{\string\glossarytitle}\string
4622 \glossarypreamble\string\n\string\begin{theglossary}\string
4623 \glossaryheader\string\n\string"}
4624 \write\glswrite{postamble \string"\string%\string\n\string
4625 \end{theglossary}\string\glossarypostamble\string\n
4626 \string"}
```

```

4627 \write\glswrite{group_skip \string"\string\glsgroupskip\string\n
4628 \string"}
4629 \write\glswrite{item_0 \string"\string%\string\n\string"}
4630 \write\glswrite{item_1 \string"\string%\string\n\string"}
4631 \write\glswrite{item_2 \string"\string%\string\n\string"}
4632 \write\glswrite{item_01 \string"\string%\string\n\string"}
4633 \write\glswrite{item_x1
4634 \string"\string\relax \string\glsgroupresetentrylist\string\n
4635 \string"}
4636 \write\glswrite{item_12 \string"\string%\string\n\string"}
4637 \write\glswrite{item_x2
4638 \string"\string\relax \string\glsgroupresetentrylist\string\n
4639 \string"}

4640 \write\glswrite{delim_0 \string"\string\{\string
4641 \glsgroupentrynumbers\string\{\string\relax \string"}
4642 \write\glswrite{delim_1 \string"\string\{\string
4643 \glsgroupentrynumbers\string\{\string\relax \string"}
4644 \write\glswrite{delim_2 \string"\string\{\string
4645 \glsgroupentrynumbers\string\{\string\relax \string"}
4646 \write\glswrite{delim_t \string"\string\}\string\}\string"}
4647 \write\glswrite{delim_n \string"\string\delimN \string"}
4648 \write\glswrite{delim_r \string"\string\delimR \string"}
4649 \write\glswrite{headings_flag 1}
4650 \write\glswrite{heading_prefix
4651 \string"\string\glsgroupheading\string\{\string"}
4652 \write\glswrite{heading_suffix
4653 \string"\string\}\string\relax
4654 \string\glsgroupresetentrylist \string"}
4655 \write\glswrite{symhead_positive \string"glssymbols\string"}
4656 \write\glswrite{numhead_positive \string"glsgroupnumbers\string"}
4657 \write\glswrite{page_compositor \string"glsgroupcompositor\string"}
4658 \@glsgroupescbsdq\glsgroupsuffixF
4659 \@glsgroupescbsdq\glsgroupsuffixFF
4660 \ifx\glsgroupsuffixF\@empty
4661 \else
4662 \write\glswrite{suffix_2p \string"\glsgroupsuffixF\string"}
4663 \fi
4664 \ifx\glsgroupsuffixFF\@empty
4665 \else
4666 \write\glswrite{suffix_3p \string"\glsgroupsuffixFF\string"}
4667 \fi
4668 \closeout\glswrite
4669 \let\writeist\relax
4670 }
4671 \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`

```
4672 \newcommand{\noist}{%
```

Update attributes list

```
4673 \@gls@addpredefinedattributes
```

```
4674 \let\writeist\relax
```

```
4675 }
```

`\@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`

```
4676 \newcommand*{\@makeglossary}[1]{%
```

```
4677 \ifglossaryexists{#1}%
```

```
4678 {%
```

Only create a new write if `savewrites=false` otherwise create a token to collect the information.

```
4679 \ifglssavewrites
```

```
4680 \expandafter\newtoks\csname glo@#1@filetok\endcsname
```

```
4681 \else
```

```
4682 \expandafter\newwrite\csname glo@#1@file\endcsname
```

```
4683 \expandafter\@glsopenfile\csname glo@#1@file\endcsname{#1}%
```

```
4684 \fi
```

```
4685 \@gls@renewglossary
```

```
4686 \writeist
```

```
4687 }%
```

```
4688 {%
```

```
4689 \PackageError{glossaries}%
```

```
4690 {Glossary type ‘#1’ not defined}%
```

```
4691 {New glossaries must be defined before using \string\makeglossary}%
```

```
4692 }%
```

```
4693 }
```

`\@glsopenfile` Open write file associated with the given glossary.

```
4694 \newcommand*{\@glsopenfile}[2]{%
```

```
4695 \immediate\openout#1=\jobname.\csname @glotype@#2@out\endcsname
```

```
4696 \PackageInfo{glossaries}{Writing glossary file
```

```
4697 \jobname.\csname @glotype@#2@out\endcsname}%
```

```
4698 }
```

`\@closegls`

```

4699 \newcommand*{\@closegls}[1]{%
4700   \closeout\csname glo@#1@file\endcsname
4701 }
4702 %   \end{macrocode}
4703 %\end{macro}
4704 %
4705 %\begin{macro}{\@gls@automake}
4706 %\changes{4.08}{2014-07-30}{new}
4707 %   \begin{macrocode}
4708 \ifglsxindy
4709   \newcommand*{\@gls@automake}[1]{%
4710     \ifglossaryexists{#1}
4711     {%
4712       \@closegls{#1}%
4713       \ifdefstring{\glsorder}{letter}%
4714       {\def\@gls@order{-M ord/letorder }}%
4715       {\let\@gls@order\@empty}%
4716       \ifcsundef{\xdy@#1@language}%
4717       {\let\@gls@langmod\@xdy@main@language}%
4718       {\letcs\@gls@langmod{\xdy@#1@language}}%
4719       \edef\@gls@dothiswrite{\noexpand\write18{xindy
4720         -I xindy
4721         \@gls@order
4722         -L \@gls@langmod\space
4723         -M \gls@istfilebase\space
4724         -C \gls@codepage\space
4725         -t \jobname.\csuse{\@glotype@#1@log}
4726         -o \jobname.\csuse{\@glotype@#1@in}
4727         \jobname.\csuse{\@glotype@#1@out}}}%
4728     }%
4729     \@gls@dothiswrite
4730   }%
4731   {%
4732     \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
4733   }%
4734 }
4735 \else
4736   \newcommand*{\@gls@automake}[1]{%
4737     \ifglossaryexists{#1}
4738     {%
4739       \@closegls{#1}%
4740       \ifdefstring{\glsorder}{letter}%
4741       {\def\@gls@order{-l }}%
4742       {\let\@gls@order\@empty}%
4743       \edef\@gls@dothiswrite{\noexpand\write18{makeindex \@gls@order
4744         -s \istfilename\space
4745         -t \jobname.\csuse{\@glotype@#1@log}
4746         -o \jobname.\csuse{\@glotype@#1@in}
4747         \jobname.\csuse{\@glotype@#1@out}}}%

```



```

4748     }%
4749     \@gls@dothiswrite
4750   }%
4751   {%
4752     \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
4753   }%
4754 }
4755 \fi

```

`\makeglossaries` Issue warning that `\makeglossaries` hasn't been used.

```

4756 \newcommand*{\@warn@nomakeglossaries}{}

```

Only use this if warning if `\printglossary` has been used without `\makeglossaries`

```

4757 \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`

```

4758 \newcommand*{\makeglossaries}{%

```

Define the write used for style file also used for all other output files if `savewrites=true`.

```

4759   \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.

```

4760   \protected@write\@auxout{}{\string\providecommand\string\@glsorder[1]{} }
4761   \protected@write\@auxout{}{\string\providecommand\string\@istfilename[1]{} }

```

Write the name of the style file to the aux file (needed by `makeglossaries`)

```

4762   \protected@write\@auxout{}{\string\@istfilename{\istfilename}}%
4763   \protected@write\@auxout{}{\string\@glsorder{\glsorder}}

```

Iterate through each glossary type and activate it.

```

4764   \@for\@glo@type:=\@glo@types\do{%
4765     \ifthenelse{\equal{\@glo@type}{} }{}{%
4766       \@makeglossary{\@glo@type}}%
4767   }%

```

New glossaries must be created before `\makeglossaries` so disable `\newglossary`.

```

4768   \renewcommand*\newglossary[4] []{%
4769     \PackageError{glossaries}{New glossaries
4770 must be created before \string\makeglossaries}{You need
4771 to move \string\makeglossaries\space after all your
4772 \string\newglossary\space commands}}%

```

Any subsequent instances of this command should have no effect

```

4773   \let\@makeglossary\relax
4774   \let\makeglossary\relax
4775   \let\makeglossaries\relax

```

Disable all commands that have no effect after `\makeglossaries`

```
4776 \disable@onlypremakeg
```

Allow see key:

```
4777 \let\gls@checkseeallowed\relax
```

Suppress warning about no `\makeglossaries`

```
4778 \let\warn@nomakeglossaries\relax
```

Activate warning about missing `\printglossary`

```
4779 \def\warn@noprintglossary{%
4780   \GlossariesWarningNoLine{No \string\printglossary\space
4781     or \string\printglossaries\space
4782     found.^^J(Remove \string\makeglossaries\space if you don't want
4783     any glossaries.)^^JThis document will not have a glossary}%
4784 }%
```

Declare list parser for `\glsdisplaynumberlist`

```
4785 \ifglssavenumberlist
4786   \edef\@gls@dodeflistparser{\noexpand\DeclareListParser
4787     {\noexpand\glsnumlistparser}{\delimN}}%
4788   \@gls@dodeflistparser
4789 \fi
```

Prevent user from also using `\makenoidxglossaries`

```
4790 \let\makenoidxglossaries\@no@makeglossaries
```

Prohibit sort key in `printgloss` family:

```
4791 \renewcommand*{\@printgloss@setsort}{%
4792   \let\@glo@assign@sortkey\@glo@no@assign@sortkey
4793 }%
```

Check the automake setting:

```
4794 \ifglsautomake
4795   \renewcommand*{\@gls@doautomake}{%
4796     \@for\@gls@type:=\@glo@types\do{%
4797       \ifdefempty{\@gls@type}{}%
4798       {\@gls@automake{\@gls@type}}%
4799     }%
4800   }%
4801 \fi
4802 }
```

Must occur in the preamble:

```
4803 \@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
4804 \let\makeglossary\makeglossaries

    If \makeglossaries hasn't been used, issue a warning. Also issue a warning if neither
    \printglossaries nor \printglossary have been used.
4805 \AtEndDocument{%
4806   \warn@nomakeglossaries
4807   \warn@noprintglossary
4808 }

noidxglossaries Analogous to \makeglossaries this activates the commands needed for \printnoidxglossary
4809 \newcommand*{\makenoidxglossaries}{%

    Redefine empty glossary warning:
4810   \renewcommand{\@gls@noref@warn}[1]{%
4811     \GlossariesWarning{Empty glossary for
4812       \string\printnoidxglossary[type={##1}].
4813     Rerun may be required (or you may have forgotten to use
4814     commands like \string\gls).}%
4815   }%

    Don't escape makeindex/xindy characters
4816   \let\@gls@checkmkidxchars\@gobble

    Write glossary information to aux instead of glossary files
4817   \let\@do@wrglossary\gls@noidxglossary

    Switch on group headings that use the character code:
4818   \let\@gls@getgrouptitle\@gls@noidx@getgrouptitle

    Allow see key:
4819   \let\gls@checkseeallowed\relax

    Redefine cross-referencing macro:
4820   \renewcommand{\@do@seeglossary}[2]{%
4821     \edef\@gls@label{\glsdetoklabel{##1}}%
4822     \protected@write\@auxout{}{%
4823       \string\@gls@reference
4824       {\csname glo@\@gls@label @type\endcsname}%
4825       {\@gls@label}%
4826       {%
4827         \string\glsseeformat##2}%
4828       }%
4829     }%
4830   }%

    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.
4831   \AtBeginDocument
4832   {%
4833     \write\@auxout{\string\providecommand\string\@gls@reference[3]{}}%
4834   }%

```

Change warning about no glossares

```
4835 \def\warn@noprintglossary{%
4836   \GlossariesWarningNoLine{No \string\printnoidxglossary\space
4837     or \string\printnoidxglossaries ^^J
4838     found. (Remove \string\makenoidxglossaries\space if you
4839     don't want any glossaries.)^^JThis document will not have a glossary}%
4840 }%
```

Suppress warning about no \makeglossaries

```
4841 \let\warn@nomakeglossaries\relax
```

Prevent user from also using \makeglossaries

```
4842 \let\makeglossaries\@no@makeglossaries
```

Allow sort key in printgloss family:

```
4843 \renewcommand*{\@printgloss@setsort}{%
4844   \let\@glo@assign@sortkey\@glo@assign@sortkey
```

Initialise default sort order:

```
4845   \def\@glo@sorttype{\@glo@default@sorttype}%
4846 }%
```

All entries must be defined in the preamble:

```
4847 \renewcommand*\new@glossaryentry[2]{%
4848   \PackageError{glossaries}{Glossary entries must be
4849     defined in the preamble^^Jwhen you use
4850     \string\makenoidxglossaries}%
4851   {Either move your definitions to the preamble or use
4852     \string\makeglossaries}%
4853 }%
```

Redefine \glentrynumberlist

```
4854 \renewcommand*{\glentrynumberlist}[1]{%
4855   \letcs{\@gls@loclist}{glo@\glsetoklabel{##1}@loclist}%
4856   \ifdef\@gls@loclist
4857     {%
4858       \glsnoidxloclist{\@gls@loclist}%
4859     }%
4860     {%
4861       ??\glsoifexists{##1}%
4862       {%
4863         \GlossariesWarning{Missing location list for '##1'. Either
4864           a rerun is required or you haven't referenced the entry.}%
4865       }%
4866     }%
4867 }%
```

Redefine \gldisplaynumberlist

```
4868 \renewcommand*{\gldisplaynumberlist}[1]{%
4869   \letcs{\@gls@loclist}{glo@\glsetoklabel{##1}@loclist}%
4870   \ifdef\@gls@loclist
4871     {%
```

```

4872 \def\@gls@noidxloclist@sep{%
4873 \def\@gls@noidxloclist@sep{%
4874 \def\@gls@noidxloclist@sep{%
4875 \glsnumlistsep
4876 }%
4877 \def\@gls@noidxloclist@finalsep{\glsnumlistlastsep}%
4878 }%
4879 }%
4880 \def\@gls@noidxloclist@finalsep{}}%
4881 \def\@gls@noidxloclist@prev{}}%
4882 \forlistloop{\glsnoidxdisplayloclisthandler}{\@gls@loclist}%
4883 \@gls@noidxloclist@finalsep
4884 \@gls@noidxloclist@prev
4885 }%
4886 {%
4887 ??\glsdoifexists{##1}%
4888 {%
4889 \GlossariesWarning{Missing location list for ‘##1’. Either
4890 a rerun is required or you haven’t referenced the entry.}%
4891 }%
4892 }%
4893 }%

```

Provide a generic way of iterating through the number list:

```

4894 \renewcommand*\@glsnumberlistloop}[3]{%
4895 \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
4896 \let\@gls@org\glsnoidxdisplayloc\glsnoidxdisplayloc
4897 \let\@gls@org\glsseeformat\glsseeformat
4898 \let\glsnoidxdisplayloc##2\relax
4899 \let\glsseeformat##3\relax
4900 \ifdef\@gls@loclist
4901 {%
4902 \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
4903 }%
4904 {%
4905 ??\glsdoifexists{##1}%
4906 {%
4907 \GlossariesWarning{Missing location list for ‘##1’. Either
4908 a rerun is required or you haven’t referenced the entry.}%
4909 }%
4910 }%
4911 \let\glsnoidxdisplayloc\@gls@org\glsnoidxdisplayloc
4912 \let\glsseeformat\@gls@org\glsseeformat
4913 }%

```

Modify sanitize sort function

```

4914 \let\@gls@sanitizesort\@gls@noidx@sanitizesort
4915 \let\@gls@nosanitizesort\@gls@noidx@nosanitizesort
4916 \@gls@noidx@setsanitizesort
4917 }

```

Preamble-only command:

```
4918 \@onlypreamble{\makenoidxglossaries}
```

```
lsnumberlistloop \glsnumberlistloop{<label>}{<handler>}
```

```
4919 \newcommand*{\glsnumberlistloop}[2]{%
4920   \PackageError{glossaries}{\string\glsnumberlistloop\space
4921     only works with \string\makenoidxglossaries}{}%
4922 }
```

listloophandler Handler macro for \glsnumberlistloop. (The argument should be in the form \glsnoidxdisplayloc{<prefix>})

```
4923 \newcommand*{\glsnoidxnumberlistloophandler}[1]{%
4924   #1%
4925 }
```

@makeglossaries Can't use both \makeglossaries and \makenoidxglossaries

```
4926 \newcommand*{\@no@makeglossaries}{%
4927   \PackageError{glossaries}{You can't use both
4928     \string\makeglossaries\space and \string\makenoidxglossaries}%
4929   {Either use one or other (or none) of those commands but not both
4930     together.}%
4931 }
```

@gls@noref@warn Warning when no instances of \@gls@reference found.

```
4932 \newcommand{\@gls@noref@warn}[1]{%
4933   \GlossariesWarning{\string\makenoidxglossaries\space
4934     is required to make \string\printnoidxglossary[type={#1}] work}%
4935 }
```

s@noidxglossary Write the glossary information to the aux file:

```
4936 \newcommand*{\gls@noidxglossary}{%
4937   \protected@write\@auxout{}{%
4938     \string\@gls@reference
4939       {\csname glo@\@gls@label @type\endcsname}%
4940       {\@gls@label}%
4941       {\string\glsnoidxdisplayloc
4942         {\@glo@counterprefix}%
4943         {\@gls@counter}%
4944         {\@glsnumberformat}%
4945         {\@glslocref}%
4946       }%
4947   }%
4948 }
```

1.14 Writing information to associated files

`\istfile` Deprecated.

```
4949 \def\istfile{\glswrite}
```

At the end of the document, the files should be created if `savewrites=true`.

```
4950 \AtEndDocument{%
```

```
4951   \glswritefiles
```

```
4952 }
```

`\@glswritefiles` Only write the files if `savewrites=true`

```
4953 \newcommand*{\@glswritefiles}{%
```

Iterate through all the glossaries

```
4954   \forallglossaries{\@glo@type}{%
```

Check for empty glossaries (patch provided by Patrick Häcker)

```
4955     \ifcsundef{glo@\@glo@type @filetok}%
```

```
4956     {%
```

```
4957       \def\gls@tmp{}%
```

```
4958     }%
```

```
4959     {%
```

```
4960       \edef\gls@tmp{\expandafter\the
```

```
4961         \csname glo@\@glo@type @filetok\endcsname}%
```

```
4962     }%
```

```
4963     \ifx\gls@tmp\@empty
```

```
4964       \ifx\@glo@type\glsdefaulttype
```

```
4965         \GlossariesWarningNoLine{Glossary ‘\@glo@type’ has no
```

```
4966           entries.^^JRemember to use package option ‘nomain’ if
```

```
4967 you
```

```
4968           don’t want to^^Juse the main glossary}%
```

```
4969       \else
```

```
4970         \GlossariesWarningNoLine{Glossary ‘\@glo@type’ has no
```

```
4971           entries}%
```

```
4972       \fi
```

```
4973     \else
```

```
4974       \@glsopenfile{\glswrite}{\@glo@type}%
```

```
4975       \immediate\write\glswrite{%
```

```
4976         \expandafter\the
```

```
4977         \csname glo@\@glo@type @filetok\endcsname}%
```

```
4978       \immediate\closeout\glswrite
```

```
4979     \fi
```

```
4980   }%
```

```
4981 }
```

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`

```
4982 \if@gls@docloaded
4983 \else
4984   \renewcommand*{\glossary}[1][main]{\gls@glossary{#1}}
4985 \fi
```

The associated number should be stored in `\theglentrycounter` before using `\gls@glossary`.

`\gls@glossary`

```
4986 \newcommand*{\gls@glossary}[1]{%
4987   \@gls@glossary{#1}%
4988 }
```

`\@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.

```
4989 \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.

`s@renewglossary`

```
4990 \newcommand{\@gls@renewglossary}{%
4991   \gdef\@gls@glossary##1{\@bsphack\beginingroup\gls@wrglossary{##1}}%
4992   \let\@gls@renewglossary\empty
4993 }
```

The `\gls@wrglossary` command is defined to have two arguments. The first argument is the glossary type, the second argument is the glossary entry (the format of which is set in `\glslink`).

`\gls@wrglossary`

```
4994 \newcommand*{\gls@wrglossary}[2]{%
4995   \ifglssavewrites
4996     \protected@edef\@gls@tmp{\the\csname glo@#1@filetok\endcsname#2}%
4997     \expandafter\global\expandafter\csname glo@#1@filetok\endcsname
4998       \expandafter{\@gls@tmp~J}%
4999   \else
```



```

5000 \ifcsdef{glo@#1@file}%
5001 {%
5002 \expandafter\protected@write\csname glo@#1@file\endcsname{%
5003 \gls@disablepagerefexpansion}{#2}%
5004 }%
5005 {%
5006 \ifignoredglossary{#1}{}%
5007 {%
5008 \GlossariesWarning{No file defined for glossary ‘#1’}%
5009 }%
5010 }%
5011 \fi
5012 \endgroup\@esphack
5013 }

```

\do@wrglossary

```

5014 \newcommand*{\do@wrglossary}[1]{%
5015 \glswriteentry{#1}{\do@wrglossary{#1}}%
5016 }

```

\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.

```

5017 \newcommand*{\glswriteentry}[2]{%
5018 \ifglsindexonlyfirst
5019 \ifglsused{#1}{#2}%
5020 \else
5021 #2%
5022 \fi
5023 }

```

\protected@pagefmts List of page formats to be protected against expansion.

```

5024 \newcommand{\gls@protected@pagefmts}{%
5025 \gls@numberpage,\gls@alphpage,\gls@Alphpage,\gls@romanpage,\gls@Romanpage,\gls@arabicpage%
5026 }

```

\gls@disablepagerefexpansion

```

5027 \newcommand*{\gls@disablepagerefexpansion}{%
5028 \@for\@gls@this:=\gls@protected@pagefmts\do
5029 {%
5030 \expandafter\let\@gls@this\relax
5031 }%
5032 }

```

\gls@alphpage

```

5033 \newcommand*{\gls@alphpage}{\@alph\c@page}

```

\gls@Alphpage

```

5034 \newcommand*{\gls@Alphpage}{\@Alph\c@page}

```

`\gls@numberpage`

```
5035 \newcommand*{\gls@numberpage}{\number\c@page}
```

`\gls@arabicpage`

```
5036 \newcommand*{\gls@arabicpage}{\@arabic\c@page}
```

`\gls@romanpage`

```
5037 \newcommand*{\gls@romanpage}{\romannumeral\c@page}
```

`\gls@Romanpage`

```
5038 \newcommand*{\gls@Romanpage}{\@Roman\c@page}
```

`protectedpagefmt`

```
\glsaddprotectedpagefmt{<cs name>}
```

Added a page format to the list of protected page formats. The argument should be the name (without a backslash) of the command that takes a \TeX register as the argument (`\<csname>\c@page` must be valid).

```
5039 \newcommand*{\glsaddprotectedpagefmt}[1]{%
5040   \eappto\gls@protected@pagefmts{,\expandonce{\csname gls#1page\endcsname}}%
5041   \csedef{gls#1page}{\expandonce{\csname#1\endcsname}\noexpand\c@page}%
5042   \eappto\@wrglossarynumberhook{%
5043     \noexpand\let\expandonce{\csname org@gls#1\endcsname}%
5044     \expandonce{\csname#1\endcsname}%
5045     \noexpand\def\expandonce{\csname#1\endcsname}{%
5046       \noexpand\@wrglossary@pageformat
5047       \expandonce{\csname gls#1page\endcsname}%
5048       \expandonce{\csname org@gls#1\endcsname}%
5049     }%
5050   }%
5051 }
```

`ssarynumberhook` Hook used by `\@do@wrglossary`

```
5052 \newcommand*\@wrglossarynumberhook{}
```

`sary@pageformat`

```
5053 \newcommand{\@wrglossary@pageformat}[3]{%
5054   \ifx#3\c@page #1\else #2#3\fi
5055 }
```

`owprimitivemods` Conditional to determine whether or not `\@do@wrglossary` should be allowed to temporarily redefine `\the` and `\number`.

```
5056 \newif\ifglswrallowprimitivemods
5057 \glswrallowprimitivemodstrue
```

`@do@wrglossary` Write the glossary entry in the appropriate format. (Need to set `\@glsnumberformat` and `\@gls@counter` prior to use.) The argument is the entry's label.

```
5058 \newcommand*{\@do@wrglossary}[1]{%
5059   \begingroup
```

First a bit of hackery to prevent premature expansion of `\c@page`. Store original definitions:

```
5060   \let\orgthe\the
5061   \let\orgnumber\number

5062   \let\orgarabic\@arabic
5063   \let\orgromannumeral\romannumeral
5064   \let\orgalph\@alph
5065   \let\orgAlph\@Alph
5066   \let\orgRoman\@Roman
```

Redefine:

```
5067   \ifglswrallowprimitivemods
5068     \def\the##1{%
5069       \ifx##1\c@page \gls@numberpage\else\orgthe##1\fi}%
5070     \def\number##1{%
5071       \ifx##1\c@page \gls@numberpage\else\orgnumber##1\fi}%
5072     \fi
5073     \def\@arabic##1{%
5074       \ifx##1\c@page \gls@arabicpage\else\orgarabic##1\fi}%
5075     \def\romannumeral##1{%
5076       \ifx##1\c@page \gls@romanpage\else\orgromannumeral##1\fi}%
5077     \def\@Roman##1{%
5078       \ifx##1\c@page \gls@Romanpage\else\orgRoman##1\fi}%
5079     \def\@alph##1{%
5080       \ifx##1\c@page \gls@alphpage\else\orgalph##1\fi}%
5081     \def\@Alph##1{%
5082       \ifx##1\c@page \gls@Alphpage\else\orgAlph##1\fi}%

```

Add hook to allow for other number formats:

```
5083   \@wrglossarynumberhook
```

Prevent expansion:

```
5084   \gls@disablepagerefexpansion
```

Now store location in `\@glslocref`:

```
5085   \protected@xdef\@glslocref{\theHglentrycounter}%
5086   \endgroup
```

Escape any special characters

```
5087   \@gls@checkmkidxchars\@glslocref
```

Check if the hyper-location is the same as the location and set the hyper prefix.

```
5088   \expandafter\ifx\theHglentrycounter\theHglentrycounter\relax
5089   \def\@glo@counterprefix{}%
5090   \else
5091     \protected@edef\@glsHlocref{\theHglentrycounter}%
5092     \@gls@checkmkidxchars\@glsHlocref
```

```

5093 \edef\@do@glS@getcounterprefix{\noexpand\@glS@getcounterprefix
5094 {\@glS@locref}{\@glS@Hlocref}%
5095 }%
5096 \@do@glS@getcounterprefix
5097 \fi

```

De-tok label if required

```

5098 \edef\@glS@label{\glSdetoklabel{#1}}%

```

Write the information to file:

```

5099 \@do@wrglossary
5100 }

```

@do@wrglossary

```

5101 \newcommand*{\@do@wrglossary}{%

```

Determine whether to use xindy or makeindex syntax

```

5102 \ifglSxindy

```

Need to determine if the formatting information starts with a (or) indicating a range.

```

5103 \expandafter\@glo@check@mkidxrangechar\@glS@numberformat\@nil
5104 \def\@glo@range{}%
5105 \expandafter\if\@glo@prefix(\relax
5106 \def\@glo@range{:open-range}%
5107 \else
5108 \expandafter\if\@glo@prefix)\relax
5109 \def\@glo@range{:close-range}%
5110 \fi
5111 \fi

```

Write to the glossary file using xindy syntax.

```

5112 \glS@glossary{\csname glo@\@glS@label @type\endcsname}{%
5113 (indexentry :tkey (\csname glo@\@glS@label @index\endcsname)
5114 :locref \string"{\@glo@counterprefix}{\@glS@locref}\string" %
5115 :attr \string"\@glS@counter\@glo@suffix\string"
5116 \@glo@range
5117 )
5118 }%
5119 \else

```

Convert the format information into the format required for makeindex

```

5120 \@set@glo@numformat{\@glo@numfmt}{\@glS@counter}{\@glS@numberformat}%
5121 {\@glo@counterprefix}%

```

Write to the glossary file using makeindex syntax.

```

5122 \glS@glossary{\csname glo@\@glS@label @type\endcsname}{%
5123 \string\glossaryentry{\csname glo@\@glS@label @index\endcsname
5124 \@glS@encapchar\@glo@numfmt}{\@glS@locref}}%
5125 \fi
5126 }

```

`etcounterprefix` Get the prefix that needs to be prepended to counter in order to get the hyper counter. (For example, with the standard article class and `hyperref`, `\theequation` needs to be prefixed with `\section num` to get the equivalent `\theHequation`.) NB this assumes that the prefix ends with a dot, which is the standard. (Otherwise it makes the xindy location classes more complicated.)

```

5127 \newcommand*\@gls@getcounterprefix[2]{%
5128   \edef\@gls@thisloc{#1}\edef\@gls@thisHloc{#2}%
5129   \ifx\@gls@thisloc\@gls@thisHloc
5130     \def\@glo@counterprefix{}%
5131   \else
5132     \def\@gls@get@counterprefix##1.#1##2\end@getprefix{%
5133       \def\@glo@tmp{##2}%
5134       \ifx\@glo@tmp\@empty
5135         \def\@glo@counterprefix{}%
5136       \else
5137         \def\@glo@counterprefix{##1}%
5138       \fi
5139     }%
5140   \@gls@get@counterprefix#2.#1\end@getprefix

```

Warn if no prefix can be formed.

```

5141   \ifx\@glo@counterprefix\@empty
5142     \GlossariesWarning{Hyper target ‘#2’ can’t be formed by
5143       prefixing^^Jlocation ‘#1’. You need to modify the
5144       definition of \string\theH\@gls@counter^^Jotherwise you
5145       will get the warning: “name{\@gls@counter.#1}’ has been^^J
5146       referenced but does not exist"%}
5147   \fi
5148 \fi
5149 }

```

1.15 Glossary Entry Cross-References

`@do@seeglossary` Write the glossary entry with a cross reference. The first argument is the entry’s label, the second must be in the form `[\tag]{\list}`, where `\tag` is a tag such as “see” and `\list` is a list of labels.

```

5150 \newcommand{\@do@seeglossary}[2]{%
5151   \def\@gls@xref{#2}%
5152   \@onelevel@sanitize\@gls@xref
5153   \@gls@checkmkidxchars\@gls@xref
5154   \ifglxindy
5155     \gls@glossary{\csname glo@#1@type\endcsname}{%
5156       (indexentry
5157         :tkey (\csname glo@#1@index\endcsname)
5158         :xref (\string"\@gls@xref\string")
5159         :attr \string"see\string"
5160       )
5161     }%

```

```

5162 \else
5163   \gls@glossary{\csname glo@#1@type\endcsname}{%
5164     \string\glossaryentry{\csname glo@#1@index\endcsname
5165       \gls@encapchar glsseeformat\@gls@xref}{Z}}}%
5166 \fi
5167 }

```

`\@gls@fixbraces` If no optional argument is specified, list needs to be enclosed in a set of braces.

```

5168 \def\@gls@fixbraces#1#2#3\@nil{%
5169   \ifx#2[\relax
5170     \@gls@fixbraces#1#2#3\@end@fixbraces
5171   \else
5172     \def#1{{#2#3}}%
5173   \fi
5174 }

```

`@@gls@fixbraces`

```

5175 \def@@gls@fixbraces#1[#2]#3\@end@fixbraces{%
5176   \def#1{[#2]{#3}}%
5177 }

```

`\glssee` `\glssee{<label>}{<cross-ref list>}`

```

5178 \DeclareRobustCommand*\glssee[3][\seename]{%
5179   \@do@seeglossary{#2}{#1}{#3}}
5180 \newcommand*\@glssee[3][\seename]{%
5181   \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.

```

5182 \DeclareRobustCommand*\glsseeformat[3][\seename]{%
5183   \emph{#1} \glsseelist{#2}}

```

`\glsseelist` `\glsseelist{<list>}` formats list of entry labels.

```

5184 \DeclareRobustCommand*\glsseelist[1]{%

```

If there is only one item in the list, set the last separator to do nothing.

```

5185   \let\@gls@dolast\relax

```

Don’t display separator on the first iteration of the loop

```

5186   \let\@gls@donext\relax

```

Iterate through the labels

```

5187   \@for\@gls@thislabel:=#1\do{%

```

Check if on last iteration of loop

```

5188     \ifx\@xfor@nextelement\@nnil
5189       \@gls@dolast
5190     \else
5191       \@gls@donext
5192     \fi

```

Display the entry for this label. (Expanding label as it's a temporary control sequence that's used elsewhere.)

```
5193 \expandafter\glsseeitem\expandafter{\@gls@thislabel}%
```

Update separators

```
5194 \let\@gls@dolast\glsseelastsep
```

```
5195 \let\@gls@donext\glsseesep
```

```
5196 }%
```

```
5197 }
```

`\glsseelastsep` Separator to use between penultimate and ultimate entries in a cross-referencing list.

```
5198 \newcommand*{\glsseelastsep}{\space\andname\space}
```

`\glsseesep` Separator to use between entires in a cross-referencing list.

```
5199 \newcommand*{\glsseesep}{, }
```

`\glsseeitem` `\glsseeitem{<label>}` formats individual entry in a cross-referencing list.

```
5200 \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.)

```
5201 \newcommand*{\glsseeitemformat}[1]{\glsentrytext{#1}}
```

1.16 Displaying the glossary

An individual glossary is displayed in the text using `\printglossary[<key-val list>]`. If the type key is omitted, the default glossary is displayed. The optional argument can be used to specify an alternative glossary, and can also be used to set the style, title and entry in the table of contents. Available keys are defined below.

`\save@numberlist` Provide command to store number list.

```
5202 \newcommand*{\gls@save@numberlist}[1]{%
```

```
5203 \ifglssavenumberlist
```

```
5204 \toks@{#1}%
```

```
5205 \edef\@do@writeaux@info{%
```

```
5206 \noexpand\csgdef{glo@\glscurrententrylabel @numberlist}{\the\toks@}%
```

```
5207 }%
```

```
5208 \@onelevel@sanitize\@do@writeaux@info
```

```
5209 \protected@write\@auxout{}{\@do@writeaux@info}%
```

```
5210 \fi
```

```
5211 }
```

`\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.)

```
5212 \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.

```
5213 \ifcsundef{printglossary}{}%  
5214 {%
```

If `\printglossary` is already defined, issue a warning and undefine it.

```
5215 \@gls@warnonglossdefined  
5216 \undef\printglossary  
5217 }
```

`\printglossary` has an optional argument. The default value is to set the glossary type to the main glossary.

```
5218 \newcommand*{\printglossary}[1][type=\glsdefaulttype]{%  
5219 \@printglossary{#1}{\@print@glossary}%  
5220 }
```

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`

```
5221 \newcommand*{\printglossaries}{%  
5222 \forallglossaries{\@glo@type}{\printglossary[type=\@glo@type]}%  
5223 }
```

`ntnoidxglossary` Provide an alternative to `\printglossary` that doesn't require an external indexing application. Entries won't be sorted and the location list will be empty.

```
5224 \newcommand*{\printnoidxglossary}[1][type=\glsdefaulttype]{%  
5225 \@printglossary{#1}{\@print@noidx@glossary}%  
5226 }
```

`noidxglossaries` Analogous to `\printglossaries`

```
5227 \newcommand*{\printnoidxglossaries}{%  
5228 \forallglossaries{\@glo@type}{\printnoidxglossary[type=\@glo@type]}%  
5229 }
```

`ntgloss@setsort` Initialise to do nothing.

```
5230 \newcommand*{\@printgloss@setsort}{}%
```

`preglossaryhook`

```
5231 \newcommand*{\@gls@preglossaryhook}{}%
```


`\@printglossary` Sets up the glossary for either `\printglossary` or `\printnoidxglossary`. The first argument is the options list, the second argument is the handler macro that deals with the actual glossary.

```
5232 \newcommand{\@printglossary}[2]{%
```

Set up defaults.

```
5233 \def\@glo@type{\glsdefaulttype}%
```

```
5234 \def\glossarytitle{\csname @glo@type\@glo@type @title\endcsname}%
```

```
5235 \def\glossarytoctitle{\glossarytitle}%
```

```
5236 \let\org@glossarytitle\glossarytitle
```

```
5237 \def\@glossarystyle{%
```

```
5238 \ifx\@glossary@default@style\relax
```

```
5239 \GlossariesWarning{No default glossary style provided \MessageBreak
```

```
5240 for the glossary '\@glo@type'. \MessageBreak
```

```
5241 Using deprecated fallback. \MessageBreak
```

```
5242 To fix this set the style with \MessageBreak
```

```
5243 \string\setglossarystyle\space or use the \MessageBreak
```

```
5244 style key=value option}%
```

```
5245 \fi
```

```
5246 }%
```

```
5247 \def\gls@dotoc@title{\gls@settoc@title{\@glo@type}}%
```

Store current value of `\glossaryentrynumbers`. (This may be changed via the optional argument)

```
5248 \let\@org@glossaryentrynumbers\glossaryentrynumbers
```

Localise the effects of the optional argument

```
5249 \bgroup
```

Activate or deactivate sort key:

```
5250 \@printgloss@setsort
```

Determine settings specified in the optional argument.

```
5251 \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)

```
5252 \ifx\glossarytitle\org@glossarytitle
```

```
5253 \else
```

```
5254 \expandafter\let\csname @glo@type\@glo@type @title\endcsname
```

```
5255 \glossarytitle
```

```
5256 \fi
```

Allow a high-level user command to indicate the current glossary

```
5257 \let\currentglossary\@glo@type
```

Enable individual number lists to be suppressed.

```
5258 \let\org@glossaryentrynumbers\glossaryentrynumbers
```

```
5259 \let\glsnonextpages\glsnonextpages
```

Enable individual number list to be activated:

```
5260 \let\glsnextpages\@glsnextpages
```

Enable suppression of description terminators.

```
5261 \let\nopostdesc\@nopostdesc
```

Set up the entry for the TOC

```
5262 \gls@dotocitle
```

Set the glossary style

```
5263 \@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):

```
5264 \let\gls@org@glossaryentryfield\glossentry
5265 \let\gls@org@glossarysubentryfield\subglossentry
5266 \renewcommand{\glossentry}[1]{%
5267   \xdef\glscurrententrylabel{\glsdetoklabel{##1}}%
5268   \gls@org@glossaryentryfield{##1}%
5269 }%
5270 \renewcommand{\subglossentry}[2]{%
5271   \xdef\glscurrententrylabel{\glsdetoklabel{##2}}%
5272   \gls@org@glossarysubentryfield{##1}{##2}%
5273 }%
```

```
5274 \@gls@preglossaryhook
```

Now do the handler macro that deals with the actual glossary:

```
5275 #2%
```

End the current scope

```
5276 \egroup
```

Reset `\glossaryentrynumbers`

```
5277 \global\let\glossaryentrynumbers\@org@glossaryentrynumbers
```

Suppress warning about no `\printglossary`

```
5278 \global\let\warn@noprntglossary\relax
5279 }
```

`@print@glossary` Internal workings of `\printglossary` dealing with reading the external file.

```
5280 \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.)

```
5281 \makeatletter
```

Input the glossary file, if it exists.

```
5282 \@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.

```
5283 \IfFileExists{\jobname.\csname @glo@type@\glo@type @in\endcsname}%
5284 {}%
5285 {\null}%
```

If xindy is being used, need to write the language dependent information to the .aux file for makeglossaries.

```
5286 \ifglxindy
5287 \ifcsundef{@xdy@\glo@type @language}%
5288 {%
5289     \edef\@do@auxoutstuff{%
5290         \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.

```
5291         \noexpand\immediate\noexpand\write\@auxout{%
5292             \string\providecommand\string\@xdylanguage[2]{}%
5293         \noexpand\immediate\noexpand\write\@auxout{%
5294             \string\@xdylanguage{\glo@type}\@xdy@main@language}%
5295     }%
5296 }%
5297 }%
5298 {%
5299     \edef\@do@auxoutstuff{%
5300         \noexpand\AtEndDocument{%
5301             \noexpand\immediate\noexpand\write\@auxout{%
5302                 \string\providecommand\string\@xdylanguage[2]{}%
5303             \noexpand\immediate\noexpand\write\@auxout{%
5304                 \string\@xdylanguage{\glo@type}\csname @xdy@\glo@type
5305                     @language\endcsname}%
5306             }%
5307         }%
5308     }%
5309     \@do@auxoutstuff
5310     \edef\@do@auxoutstuff{%
5311         \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.

```
5312         \noexpand\immediate\noexpand\write\@auxout{%
5313             \string\providecommand\string\@gls@codepage[2]{}%
5314         \noexpand\immediate\noexpand\write\@auxout{%
5315             \string\@gls@codepage{\glo@type}\@gls@codepage}%
5316     }%
5317 }%
5318 \@do@auxoutstuff
5319 \fi
```

Activate warning if \makeglossaries hasn't been used.

```

5320 \renewcommand*{\@warn@nomakeglossaries}{%
5321   \GlossariesWarningNoLine{\string\makeglossaries\space
5322     hasn't been used,^^Jthe glossaries will not be updated}%
5323 }%
5324 }

```

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 $\backslash@gl{sref@<type>}$. The actual sorting is done by $\backslash@glo@sortentries\{<handler>\}\{<type>\}$.

$\backslash@glo@sortentries$

```

5325 \newcommand*{\@glo@sortentries}[2]{%
5326   \def\@glo@sortinglist{}%
5327   \def\@glo@sortinghandler{#1}%
5328   \edef\@glo@type{#2}%
5329   \forlistcsloop{\@glo@do@sortentries}\@gl{sref@#2}%
5330   \csdef{\@gl{sref@#2}}{}%
5331   \@for\@this@label:=\@glo@sortinglist\do{%

```

Has this entry already been added?

```

5332     \xifinlistcs{\@this@label}\@gl{sref@#2}%
5333     {}%
5334     {%
5335       \listcsxadd{\@gl{sref@#2}}{\@this@label}%
5336     }%
5337     \ifcsdef{\@glo@sortingchildren@\@this@label}%
5338     {%
5339       \@glo@addchildren{#2}\@this@label}%
5340     }%
5341     {}%
5342   }%
5343 }

```

$\backslash@glo@addchildren$

$$\backslash@glo@addchildren\{<type>\}\{<parent>\}$$

```

5344 \newcommand*{\@glo@addchildren}[2]{%

```

Scope to allow nesting.

```

5345   \bgroup
5346     \letcs{\@glo@childlist}\@glo@sortingchildren@#2}%
5347     \@for\@this@childlabel:=\@glo@childlist\do
5348     {%

```

Check this label hasn't already been added.

```
5349      \xifinlistcs{\@this@childlabel}{@glsref@#1}%
5350      {%
5351      {%
5352      \listcsxadd{@glsref@#1}{\@this@childlabel}%
5353      }%
```

Does this child have children?

```
5354      \ifcsdef{@glo@sortingchildren@\@this@childlabel}%
5355      {%
5356      \@glo@addchildren{#1}{\@this@childlabel}%
5357      }%
5358      {%
5359      }%
5360      }%
5361 \egroup
5362 }
```

@do@sortentries

```
5363 \newcommand*{\@glo@do@sortentries}[1]{%
5364 \ifglshasparent{#1}%
5365 {%
```

This entry has a parent, so add it to the child list

```
5366 \edef\@glo@parent{\csuse{glo@glstetoklabel{#1}@parent}}%
5367 \ifcsundef{@glo@sortingchildren@\@glo@parent}%
5368 {%
5369 \csdef{@glo@sortingchildren@\@glo@parent}{}%
5370 }%
5371 {}%
5372 \expandafter\@glo@sortedinsert
5373 \csname @glo@sortingchildren@\@glo@parent\endcsname{#1}%
```

Has the parent been added?

```
5374 \xifinlistcs{\@glo@parent}{@glsref@\@glo@type}%
5375 {%
```

Yes, it has so do nothing.

```
5376 }%
5377 {%
```

No, it hasn't so add it now.

```
5378 \expandafter\@glo@do@sortentries\expandafter{\@glo@parent}%
5379 }%
5380 }%
5381 {%
5382 \@glo@sortedinsert{\@glo@sortinglist}{#1}%
5383 }%
5384 }
```

glo@sortedinsert `\@glo@sortedinsert{<list>}{<entry label>}`

Insert into list.

```
5385 \newcommand*{\@glo@sortedinsert}[2]{%
5386   \dtl@insertinto{#2}{#1}{\@glo@sortinghandler}%
5387 }%
```

The sort handlers need to be in the form required by datatool's `\dtl@sortlist` macro. These must set the count register `\dtl@sortresult` to either -1 ($\#1$ less than $\#2$), 0 ($\#1 = \#2$) or $+1$ ($\#1$ greater than $\#2$).

orthandler@word

```
5388 \newcommand*{\@glo@sorthandler@word}[2]{%
5389   \letcs\@gls@sort@A{glo@glstdetoklabel{#1}@sort}%
5390   \letcs\@gls@sort@B{glo@glstdetoklabel{#2}@sort}%
5391   \edef\glo@do@compare{%
5392     \noexpand\dtlwordindexcompare{\noexpand\dtl@sortresult}%
5393     {\expandonce\@gls@sort@B}%
5394     {\expandonce\@gls@sort@A}%
5395   }%
5396   \glo@do@compare
5397 }
```

thandler@letter

```
5398 \newcommand*{\@glo@sorthandler@letter}[2]{%
5399   \letcs\@gls@sort@A{glo@glstdetoklabel{#1}@sort}%
5400   \letcs\@gls@sort@B{glo@glstdetoklabel{#2}@sort}%
5401   \edef\glo@do@compare{%
5402     \noexpand\dtlletterindexcompare{\noexpand\dtl@sortresult}%
5403     {\expandonce\@gls@sort@B}%
5404     {\expandonce\@gls@sort@A}%
5405   }%
5406   \glo@do@compare
5407 }
```

orthandler@case Case-sensitive sort.

```
5408 \newcommand*{\@glo@sorthandler@case}[2]{%
5409   \letcs\@gls@sort@A{glo@glstdetoklabel{#1}@sort}%
5410   \letcs\@gls@sort@B{glo@glstdetoklabel{#2}@sort}%
5411   \edef\glo@do@compare{%
5412     \noexpand\dtlcompare{\noexpand\dtl@sortresult}%
5413     {\expandonce\@gls@sort@B}%
5414     {\expandonce\@gls@sort@A}%
5415   }%
5416   \glo@do@compare
5417 }
```

thandler@nocase Case-insensitive sort.

```

5418 \newcommand*{\@glo@sorthandler@nocase}[2]{%
5419   \letcs\@gls@sort@A{glo\glsdetoklabel{#1}@sort}%
5420   \letcs\@gls@sort@B{glo\glsdetoklabel{#2}@sort}%
5421   \edef\glo@do@compare{%
5422     \noexpand\dtlicompare{\noexpand\dtl@sortresult}%
5423     {\expandonce\@gls@sort@B}%
5424     {\expandonce\@gls@sort@A}%
5425   }%
5426   \glo@do@compare
5427 }

```

@sortmacro@word Sort macro for ‘word’

```

5428 \newcommand*{\@glo@sortmacro@word}[1]{%
5429   \ifdefstring{\@glo@default@sorttype}{standard}%
5430   {%
5431     \@glo@sortentries{\@glo@sorthandler@word}{#1}%
5432   }%
5433   {%
5434     \PackageError{glossaries}{Conflicting sort options:^^J
5435       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5436       \string\printnoidxglossary[sort=word]}{}%
5437   }%
5438 }

```

ortmacro@letter Sort macro for ‘letter’

```

5439 \newcommand*{\@glo@sortmacro@letter}[1]{%
5440   \ifdefstring{\@glo@default@sorttype}{standard}%
5441   {%
5442     \@glo@sortentries{\@glo@sorthandler@letter}{#1}%
5443   }%
5444   {%
5445     \PackageError{glossaries}{Conflicting sort options:^^J
5446       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5447       \string\printnoidxglossary[sort=letter]}{}%
5448   }%
5449 }

```

tmacro@standard Sort macro for ‘standard’. (Use either ‘word’ or ‘letter’ order.)

```

5450 \newcommand*{\@glo@sortmacro@standard}[1]{%
5451   \ifdefstring{\@glo@default@sorttype}{standard}%
5452   {%
5453     \ifcsdef{\@glo@sorthandler@\glsorder}%
5454     {%
5455       \@glo@sortentries{\csuse{\@glo@sorthandler@\glsorder}}{#1}%
5456     }%
5457     {%
5458       \PackageError{glossaries}{Unknown sort handler ‘\glsorder’}{}%
5459     }%
5460   }%

```

```

5461 {%
5462   \PackageError{glossaries}{Conflicting sort options:^^J
5463     \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5464     \string\printnoidxglossary[sort=standard]}{}%
5465 }%
5466 }

```

@sortmacro@case Sort macro for ‘case’

```

5467 \newcommand*{\@glo@sortmacro@case}[1]{%
5468   \ifdefstring{\@glo@default@sorttype}{standard}%
5469   {%
5470     \@glo@sortentries{\@glo@sorthandler@case}{#1}%
5471   }%
5472   {%
5473     \PackageError{glossaries}{Conflicting sort options:^^J
5474       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5475       \string\printnoidxglossary[sort=case]}{}%
5476   }%
5477 }

```

@sortmacro@nocase Sort macro for ‘nocase’

```

5478 \newcommand*{\@glo@sortmacro@nocase}[1]{%
5479   \ifdefstring{\@glo@default@sorttype}{standard}%
5480   {%
5481     \@glo@sortentries{\@glo@sorthandler@nocase}{#1}%
5482   }%
5483   {%
5484     \PackageError{glossaries}{Conflicting sort options:^^J
5485       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5486       \string\printnoidxglossary[sort=nocase]}{}%
5487   }%
5488 }

```

@sortmacro@def Sort macro for ‘def’. The order of definition is given in \glo@list@<type>.

```

5489 \newcommand*{\@glo@sortmacro@def}[1]{%
5490   \def\@glo@sortinglist{}%
5491   \for\gl@sentries[#1]{\@gl@thislabel}%
5492   {%
5493     \xifinlistcs{\@gl@thislabel}{\@gl@ref@#1}%
5494     {%
5495       \listead{\@glo@sortinglist}{\@gl@thislabel}%
5496     }%
5497   }%

```

Hasn't been referenced.

```

5498   }%
5499 }%
5500 \cslet{\@gl@ref@#1}{\@glo@sortinglist}%
5501 }

```


ortmacro@def@do This won't include parent entries that haven't been referenced.

```
5502 \newcommand*{\@glo@sortmacro@def@do}[1]{%
5503   \ifinlistcs{#1}{\@glsref@\@glo@type}%
5504   {}%
5505   {%
5506     \listcsadd{\@glsref@\@glo@type}{#1}%
5507   }%
5508   \ifcsdef{\@glo@sortingchildren@#1}%
5509   {%
5510     \@glo@addchildren{\@glo@type}{#1}%
5511   }%
5512   {}%
5513 }
```

o@sortmacro@use Sort macro for 'use'. (No sorting is required, as the entries are already in order of use, so do nothing.)

```
5514 \newcommand*{\@glo@sortmacro@use}[1]{}
```

@noidx@glossary Glossary handler for \printnoidxglossary which doesn't use an indexing application. Since \printnoidxglossary may occur at the start of the document, we can't just check if an entry has been used. Instead, the first pass needs to write information to the aux file every time an entry is referenced. This needs to be read in on the second run and stored in a list corresponding to the appropriate glossary.

```
5515 \newcommand*{\@print@noidx@glossary}{%
5516   \ifcsdef{\@glsref@\@glo@type}%
5517   {%
```

Sort the entries:

```
5518   \ifcsdef{\@glo@sortmacro@\@glo@sorttype}%
5519   {%
5520     \csuse{\@glo@sortmacro@\@glo@sorttype}{\@glo@type}%
5521   }%
5522   {%
5523     \PackageError{glossaries}{Unknown sort handler '\@glo@sorttype'}{ }%
5524   }%
```

Do the glossary heading and preamble

```
5525   \glossarysection[\glossarytoctitle]{\glossarytitle}%
5526   \glossarypreamble
5527   \begin{theglossary}%
5528   \glossaryheader
5529   \glsresetentrylist
5530   \def\@gls@currentlettergroup{ }
```

Iterate through the entries.

```
5531   \forlistcsloop{\@gls@noidx@do}{\@glsref@\@glo@type}%
```

Finally end the glossary and do the postamble:

```
5532   \end{theglossary}%
5533   \glossarypostamble
```

```

5534 }%
5535 {%
5536   \@gls@noref@warn{\@glo@type}%
5537 }%
5538 }

```

\glo@grabfirst

```

5539 \def\glo@grabfirst#1#2\@nil{%
5540   \def\@gls@firsttok{#1}%
5541   \ifdefempty\@gls@firsttok
5542   {%
5543     \def\@glo@thislettergrp{0}%
5544   }%
5545   {%

```

Sanitize it:

```

5546   \@onelevel@sanitize\@gls@firsttok

```

Fetch the first letter:

```

5547   \expandafter\@glo@grabfirst\@gls@firsttok{}{}\@nil
5548 }%
5549 }

```

\@glo@grabfirst

```

5550 \def\@glo@grabfirst#1#2\@nil{%
5551   \ifdefempty\@glo@thislettergrp
5552   {%
5553     \def\@glo@thislettergrp{glssymbols}%
5554   }%
5555   {%
5556     \count@=\uccode'#1\relax
5557     \ifnum\count@=0\relax
5558       \def\@glo@thislettergrp{glssymbols}%
5559     \else
5560       \ifdefstring\@glo@sorttype{case}%
5561       {%
5562         \count@='#1\relax
5563       }%
5564       {%
5565     }%
5566     \edef\@glo@thislettergrp{\the\count@}%
5567   \fi
5568 }%
5569 }

```

\@gls@noidx@do Handler for list iteration used by \@print@noidx@glossary. The argument is the entry label.
This only allows one sublevel.

```

5570 \newcommand{\@gls@noidx@do}[1]{%

```

Get this entry's location list

```

5571   \global\letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%

```

Does this entry have a parent?

```
5572 \ifglshasparent{#1}%  
5573 {%
```

Has a parent.

```
5574 \gls@level=\csuse{glo@\glsdetoklabel{#1}@level}\relax  
5575 \ifdefvoid{\@gls@loclist}  
5576 {%  
5577 \subglossentry{\gls@level}{#1}{}%  
5578 }%  
5579 {%  
5580 \subglossentry{\gls@level}{#1}%  
5581 }%  
5582 \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%  
5583 }%  
5584 }%  
5585 }%  
5586 {%
```

Doesn't have a parent Get this entry's sort key

```
5587 \letcs{\@gls@sort}{glo@\glsdetoklabel{#1}@sort}%
```

Fetch the first letter:

```
5588 \expandafter\glo@grabfirst\@gls@sort{}\@nil  
5589 \ifdefequal{\@glo@thislettergrp}{\@gls@currentlettergroup}%  
5590 }%  
5591 {%
```

Do the group header:

```
5592 \ifdefempty{\@gls@currentlettergroup}{\@gls@groupskip}%  
5593 \gls@groupheading{\@glo@thislettergrp}%  
5594 }%  
5595 \let\@gls@currentlettergroup\@glo@thislettergrp
```

Do this entry:

```
5596 \ifdefvoid{\@gls@loclist}  
5597 {%  
5598 \glossentry{#1}{}%  
5599 }%  
5600 {%  
5601 \glossentry{#1}%  
5602 }%  
5603 \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%  
5604 }%  
5605 }%  
5606 }%  
5607 }
```

```
\glsnoidxloclist \glsnoidxloclist{<list cs>}
```

Display location list.

```

5608 \newcommand*\glsnoidxloclist}[1]{%
5609   \def\@gls@noidxloclist@sep{}%
5610   \def\@gls@noidxloclist@prev{}%
5611   \forlistloop{\glsnoidxloclisthandler}{#1}%
5612 }

```

`xloclisthandler` Handler for location list iterator.

```

5613 \newcommand*\glsnoidxloclisthandler}[1]{%
5614   \ifdefstring{\@gls@noidxloclist@prev}{#1}%
5615   {%

```

Same as previous location so skip.

```

5616   }%
5617   {%
5618     \@gls@noidxloclist@sep
5619     #1%
5620     \def\@gls@noidxloclist@sep{\delimN}%
5621     \def\@gls@noidxloclist@prev{#1}%
5622   }%
5623 }

```

`yloclisthandler` Handler for location list iterator when used with `\glsdisplaynumberlist`.

```

5624 \newcommand*\glsnoidxdisplayloclisthandler}[1]{%
5625   \ifdefstring{\@gls@noidxloclist@prev}{#1}%
5626   {%

```

Same as previous location so skip.

```

5627   }%
5628   {%
5629     \@gls@noidxloclist@sep
5630     \@gls@noidxloclist@prev
5631     \def\@gls@noidxloclist@prev{#1}%
5632   }%
5633 }

```

`snoidxdisplayloc` `\glsnoidxdisplayloc{<prefix>}{<counter>}{<format>}{<location>}`

Display a location in the location list.

```

5634 \newcommand*\glsnoidxdisplayloc[4]{%
5635   \setentrycounter[#1]{#2}%
5636   \csuse{#3}{#4}%
5637 }

```

`\@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.

```
5638 \newcommand*{\@gls@reference}[3]{%
```

Add to label list

```
5639 \glsdoifexistsorwarn{#2}%
5640 {%
5641 \ifcsundef{@glsref@#1}{\csgdef{@glsref@#1}{}}{}%
5642 \ifinlistcs{#2}{@glsref@#1}%
5643 {}%
5644 {\listcsgadd{@glsref@#1}{#2}}%
```

Add to location list

```
5645 \ifcsundef{glo@glstdetoklabel{#2}@loclist}%
5646 {\csgdef{glo@glstdetoklabel{#2}@loclist}{}}%
5647 {}%
5648 \listcsgadd{glo@glstdetoklabel{#2}@loclist}{#3}%
5649 }%
5650 }
```

The keys that can be used in the optional argument to `\printglossary` or `\printnoidxglossary` are as follows: The type key sets the glossary type.

```
5651 \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`.

```
5652 \define@key{printgloss}{title}{%
5653 \def@glossarytitle{#1}%
5654 \let@gls@dotoc@title\relax
5655 }
```

The toctitle sets the text used for the relevant entry in the table of contents.

```
5656 \define@key{printgloss}{toctitle}{%
5657 \def@glossarytoctitle{#1}%
5658 \let@gls@dotoc@title\relax
5659 }
```

The style key sets the glossary style (but only for the given glossary).

```
5660 \define@key{printgloss}{style}{%
5661 \ifcsundef{@glsstyle@#1}%
5662 {%
5663 \PackageError{glossaries}%
5664 {Glossary style ‘#1’ undefined}{}%
5665 }%
5666 {%
5667 \def@glossarystyle{\setglossentrycompatibility
5668 \csname @glsstyle@#1\endcsname}%
5669 }%
5670 }
```

The `numberedsection` key determines if this glossary should be in a numbered section.

```

5671 \define@choicekey{printgloss}{numberedsection}[\val\nr]{%
5672 false,nolabel,autolabel,nameref}[nolabel]{%
5673   \ifcase\nr\relax
5674     \renewcommand*{\@@glossarysecstar}{*}%
5675     \renewcommand*{\@@glossaryseclabel}{}%
5676   \or
5677     \renewcommand*{\@@glossarysecstar}{}%
5678     \renewcommand*{\@@glossaryseclabel}{}%
5679   \or
5680     \renewcommand*{\@@glossarysecstar}{}%
5681     \renewcommand*{\@@glossaryseclabel}{\label{\glsautoprefix\@glo@type}}%
5682   \or
5683     \renewcommand*{\@@glossarysecstar}{*}%
5684     \renewcommand*{\@@glossaryseclabel}{%
5685       \protected@edef\@currentlabelname{\glossarytoctitle}%
5686       \label{\glsautoprefix\@glo@type}}%
5687   \fi
5688 }
```

The `nogroupskip` key determines whether or not there should be a vertical gap between glossary groups.

```

5689 \define@choicekey{printgloss}{nogroupskip}{true,false}[true]{%
5690   \csuse{glsnogroupskip#1}%
5691 }
```

The `nopostdot` key has the same effect as the package option of the same name.

```

5692 \define@choicekey{printgloss}{nopostdot}{true,false}[true]{%
5693   \csuse{glsnopostdot#1}%
5694 }
```

The `entrycounter` key is the same as the package option but localised to the current glossary.

```

5695 \define@choicekey{printgloss}{entrycounter}{true,false}[true]{%
5696   \csuse{glsentrycounter#1}%
5697   \ifglsentrycounter
5698     \ifx\@gls@counterwithin\@empty
5699       \newcounter{glossaryentry}%
5700     \else
5701       \newcounter{glossaryentry}[\@gls@counterwithin]%
5702     \fi
5703     \def\theHglossaryentry{\currentglossary.\theglossaryentry}%
5704     \renewcommand*{\glsresetentrycounter}{%
5705       \setcounter{glossaryentry}{0}%
5706     }%
5707     \renewcommand*{\glsstepentry}[1]{%
5708       \refstepcounter{glossaryentry}%
5709       \label{glsentry-\glsdetoklabel{##1}}%
5710     }%
5711     \renewcommand*{\glsentrycounterlabel}{\theglossaryentry.\space}%
5712     \renewcommand*{\glsentryitem}[1]{%
```

```

5713     \glsstepentry{##1}\glsentrycounterlabel
5714 }%
5715 \else
5716   \renewcommand*\glsresetentrycounter{}{%
5717   \renewcommand*\glsstepentry}[1]{}%
5718   \renewcommand*\glsentrycounterlabel{}{%
5719   \renewcommand*\glsentryitem}[1]{\glsresetsubentrycounter}
5720 \fi
5721 }

```

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.

```

5722 \define@choicekey{printgloss}{subentrycounter}{true,false}[true]{%
5723   \csuse{glssubentrycounter#1}%
5724   \ifglssubentrycounter
5725     \ifundef\c@glossarysubentry
5726     {%
5727       \ifglentrycounter
5728         \newcounter{glossarysubentry}[glossaryentry]%
5729       \else
5730         \newcounter{glossarysubentry}
5731       \fi
5732     }{}%
5733     \renewcommand*\glsstepsubentry}[1]{%
5734       \edef\currentglssubentry{\glsdetoklabel{##1}}%
5735       \refstepcounter{glossarysubentry}%
5736       \label{glsentry-\currentglssubentry}%
5737     }%
5738     \renewcommand*\glsresetsubentrycounter{}{%
5739       \setcounter{glossarysubentry}{0}%
5740     }%
5741     \renewcommand*\glssubentryitem}[1]{%
5742       \glsstepsubentry{##1}\glssubentrycounterlabel
5743     }%
5744     \renewcommand*\glssubentrycounterlabel{\theGlossarysubentry}\space}%
5745     \def\theHglossarysubentry{\currentglssubentry.\theGlossarysubentry}
5746   \else
5747     \renewcommand*\glssubentryitem}[1]{}%
5748     \renewcommand*\glsstepsubentry}[1]{}%
5749     \renewcommand*\glsresetsubentrycounter{}{%
5750     \renewcommand*\glssubentrycounterlabel{}{%
5751   \fi
5752 }

```

The nonnumberlist key determines if this glossary should have a number list.

```

5753 \define@boolkey{printgloss}[gls]{nonnumberlist}[true]{%
5754 \ifglslnonnumberlist
5755   \def\glossaryentrynumbers##1{%
5756 \else

```

```

5757 \def\glossaryentrynumbers##1{##1}%
5758 \fi}

```

The sort key sets the glossary sort handler (`\printnoidxglossary` only).

```

5759 \define@key{printgloss}{sort}{\@glo@assign@sortkey{#1}}

```

`@assign@sortkey` Issue error if used with `\printglossary`

```

5760 \newcommand*{\@glo@no@assign@sortkey}[1]{%
5761 \PackageError{glossaries}{‘sort’ key not permitted with
5762 \string\printglossary}%
5763 {The ‘sort’ key may only be used with \string\printnoidxglossary}%
5764 }

```

`@assign@sortkey` For use with `\printnoidxglossary`

```

5765 \newcommand*{\@glo@assign@sortkey}[1]{%
5766 \def\@glo@sorttype{#1}%
5767 }

```

`@glsnonextpages` Suppresses the next number list only. Global assignments required as it may not occur in the same level of grouping as the next numberlist. (For example, if `\glsnonextpages` is placed in the entry’s description and 3 column tabular style glossary is used.) `\org@glossaryentrynumbers` needs to be set at the start of each glossary, in the event that `\glossaryentrynumber` is re-defined.

```

5768 \newcommand*{\@glsnonextpages}{%
5769 \gdef\glossaryentrynumbers##1{%
5770 \glsresetentrylist
5771 }%
5772 }

```

`\@glsnextpages` Activate the next number list only. Global assignments required as it may not occur in the same level of grouping as the next numberlist. (For example, if `\glsnextpages` is placed in the entry’s description and 3 column tabular style glossary is used.) `\org@glossaryentrynumbers` needs to be set at the start of each glossary, in the event that `\glossaryentrynumber` is re-defined.

```

5773 \newcommand*{\@glsnextpages}{%
5774 \gdef\glossaryentrynumbers##1{%
5775 ##1\glsresetentrylist}}

```

`sresetentrylist` Resets `\glossaryentrynumbers`

```

5776 \newcommand*{\glsresetentrylist}{%
5777 \global\let\glossaryentrynumbers\org@glossaryentrynumbers}

```

`\glsnonextpages` Outside of `\printglossary` this does nothing.

```

5778 \newcommand*{\glsnonextpages}{}

```

`\glsnextpages` Outside of `\printglossary` this does nothing.

```

5779 \newcommand*{\glsnextpages}{}

```


glossaryentry If the `entrycounter` package option has been used, define a counter to number each level 0 entry.

```

5780 \ifglentrycounter
5781   \ifx\@gls@counterwithin\@empty
5782     \newcounter{glossaryentry}
5783   \else
5784     \newcounter{glossaryentry}[\@gls@counterwithin]
5785   \fi
5786   \def\theHglossaryentry{\currentglossary.\theglossaryentry}
5787 \fi

```

lossarysubentry If the `subentrycounter` package option has been used, define a counter to number each level 1 entry.

```

5788 \ifglssubentrycounter
5789   \ifglentrycounter
5790     \newcounter{glossarysubentry}[glossaryentry]
5791   \else
5792     \newcounter{glossarysubentry}
5793   \fi
5794   \def\theHglossarysubentry{\currentglssubentry.\theglossarysubentry}
5795 \fi

```

subentrycounter Resets the `glossarysubentry` counter.

```

5796 \ifglssubentrycounter
5797   \newcommand*{\glsresetsubentrycounter}{%
5798     \setcounter{glossarysubentry}{0}%
5799   }
5800 \else
5801   \newcommand*{\glsresetsubentrycounter}{}
5802 \fi

```

subentrycounter Resets the `glossareentry` counter.

```

5803 \ifglentrycounter
5804   \newcommand*{\glsresetentrycounter}{%
5805     \setcounter{glossaryentry}{0}%
5806   }
5807 \else
5808   \newcommand*{\glsresetentrycounter}{}
5809 \fi

```

\glsstepentry Advance the `glossaryentry` counter if in use. The argument is the label associated with the entry.

```

5810 \ifglentrycounter
5811   \newcommand*{\glsstepentry}[1]{%
5812     \refstepcounter{glossaryentry}%
5813     \label{glsentry-\glsdetoklabel{#1}}%
5814   }
5815 \else

```

```

5816 \newcommand*{\glsstepentry}[1]{%
5817 \fi

```

`\glsstepsubentry` Advance the glossarysubentry counter if in use. The argument is the label associated with the subentry.

```

5818 \ifglssubentrycounter
5819 \newcommand*{\glsstepsubentry}[1]{%
5820 \edef\currentglssubentry{\glsdetoklabel{#1}}%
5821 \refstepcounter{glossarysubentry}%
5822 \label{glssentry-\currentglssubentry}%
5823 }
5824 \else
5825 \newcommand*{\glsstepsubentry}[1]{%
5826 \fi

```

`\glsrefentry` Reference the entry or sub-entry counter if in use, otherwise just do `\gls`.

```

5827 \ifglssentrycounter
5828 \newcommand*{\glsrefentry}[1]{\ref{glssentry-\glsdetoklabel{#1}}}
5829 \else
5830 \ifglssubentrycounter
5831 \newcommand*{\glsrefentry}[1]{\ref{glssentry-\glsdetoklabel{#1}}}
5832 \else
5833 \newcommand*{\glsrefentry}[1]{\gls{#1}}
5834 \fi
5835 \fi

```

`\glsentrycounterlabel` Defines how to display the glossaryentry counter.

```

5836 \ifglssentrycounter
5837 \newcommand*{\glsentrycounterlabel}{\theglossaryentry.\space}
5838 \else
5839 \newcommand*{\glsentrycounterlabel}{}
5840 \fi

```

`\glsentrysubentrycounterlabel` Defines how to display the glossarysubentry counter.

```

5841 \ifglssubentrycounter
5842 \newcommand*{\glsentrysubentrycounterlabel}{\theglossarysubentry)\space}
5843 \else
5844 \newcommand*{\glsentrysubentrycounterlabel}{}
5845 \fi

```

`\glsentryitem` Step and display glossaryentry counter, if appropriate.

```

5846 \ifglssentrycounter
5847 \newcommand*{\glsentryitem}[1]{%
5848 \glsstepentry{#1}\glsentrycounterlabel
5849 }
5850 \else
5851 \newcommand*{\glsentryitem}[1]{\glsresetsubentrycounter}
5852 \fi

```

`glssubentryitem` Step and display `glossarysubentry` counter, if appropriate.

```
5853 \ifglssubentrycounter
5854   \newcommand*{\glssubentryitem}[1]{%
5855     \glssubentrycounterlabel
5856   }
5857 \else
5858   \newcommand*{\glssubentryitem}[1]{}
5859 \fi
```

`theglossary` If the `theglossary` environment has already been defined, a warning will be issued. This environment should be redefined by glossary styles.

```
5860 \ifcsundef{theglossary}%
5861 {%
5862   \newenvironment{theglossary}{}{}%
5863 }%
5864 {%
5865   \@gls@warnontheglossdefined
5866   \renewenvironment{theglossary}{}{}%
5867 }
```

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`

```
5868 \newcommand*{\glossaryheader}{}%
```

`\glstarget` `\glstarget{<label>}{<name>}`

Provide user interface to `\@glstarget` to make it easier to modify the glossary style in the document.

```
5869 \newcommand*{\glstarget}[2]{\@glstarget{\glslinkprefix#1}{#2}}
```

As from version 3.08, glossary information is now written to the external files using `\glossentry` and `\subglossentry` instead of `\glossaryentryfield` and `\glossarysubentryfield`. The default definition provides backward compatibility for glossary styles that use the old forms.

`atibleglossentry`

`\glossentry{<label>}{<page-list>}`

```
5870 \providecommand*{\compatibleglossentry}[2]{%
5871   \toks@{#2}%
5872   \protected@edef\do@glossentry{\noexpand\glossaryentryfield{#1}%
5873     {\noexpand\glsnamefont
5874       {\expandafter\expandonce\csname glo@#1@name\endcsname}}}%
5875 }
```

```

5875     {\expandafter\expandonce\csname glo@#1@desc\endcsname}%
5876     {\expandafter\expandonce\csname glo@#1@symbol\endcsname}%
5877     {\the\toks@}%
5878   }%
5879   \@do@glossentry
5880 }

```

\glossentryname

```

5881 \newcommand*{\glossentryname}[1]{%
5882   \glsdoifexistsorwarn{#1}%
5883   {%
5884     \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
5885     \expandafter\glsnamefont\expandafter{\glo@name}%
5886   }%
5887 }

```

\Glossentryname

```

5888 \newcommand*{\Glossentryname}[1]{%
5889   \glsdoifexistsorwarn{#1}%
5890   {%
5891     \glsnamefont{\Glsentryname{#1}}%
5892   }%
5893 }

```

\glossentrydesc

```

5894 \newcommand*{\glossentrydesc}[1]{%
5895   \glsdoifexistsorwarn{#1}%
5896   {%
5897     \glsentrydesc{#1}%
5898   }%
5899 }

```

\Glossentrydesc

```

5900 \newcommand*{\Glossentrydesc}[1]{%
5901   \glsdoifexistsorwarn{#1}%
5902   {%
5903     \Glsentrydesc{#1}%
5904   }%
5905 }

```

\glossentrysymbol

```

5906 \newcommand*{\glossentrysymbol}[1]{%
5907   \glsdoifexistsorwarn{#1}%
5908   {%
5909     \glsentrysymbol{#1}%
5910   }%
5911 }

```

lossentrysymbol

```
5912 \newcommand*\Glossentrysymbol}[1]{%
5913   \glsdoifexistsorwarn{#1}%
5914   {%
5915     \Glsentrysymbol{#1}%
5916   }%
5917 }
```

blesubglossentry

`\subglossentry{<level>}{<label>}{<page-list>}`

```
5918 \providecommand*\compatiblesubglossentry}[3]{%
5919   \toks@{#3}%
5920   \protected@edef\@do@subglossentry{\noexpand\glossarysubentryfield{\number#1}%
5921     {#2}%
5922     {\noexpand\glsnamefont
5923       {\expandafter\expandonce\csname glo@#2@name\endcsname}}%
5924     {\expandafter\expandonce\csname glo@#2@desc\endcsname}%
5925     {\expandafter\expandonce\csname glo@#2@symbol\endcsname}%
5926     {\the\toks@}}%
5927   }%
5928   \@do@subglossentry
5929 }
```

rycompatibility

```
5930 \newcommand*\setglossentrycompatibility{%
5931   \let\glossentry\compatibleglossentry
5932   \let\subglossentry\compatiblesubglossentry
5933 }
5934 \setglossentrycompatibility
```

ossaryentryfield

`\glossaryentryfield{<label>}{<name>}{<description>}{<symbol>}{<page-list>}`

This command formerly governed how each entry row should be formatted in the glossary.
Now deprecated.

```
5935 \newcommand{\glossaryentryfield}[5]{%
5936   \GlossariesWarning
5937   {Deprecated use of \string\glossaryentryfield.^^J
5938     I recommend you change to \string\glossentry.^^J
5939     If you've just upgraded, try removing your gls auxiliary
5940     files^^J and recompile}%
5941   \noindent\textbf{\glstarget{#1}{#2}} #4 #3. #5\par}
```

arysubentryfield

`\glossarysubentryfield{<level>}{<label>}{<name>}{<description>}{<symbol>}{<page-list>}`

This command governs how each subentry should be formatted in the glossary. Glossary styles need to redefine this command. Most of the predefined styles ignore *<symbol>*. The first argument is a number indicating the level. (The level should be greater than or equal to 1.)

```
5942 \newcommand*{\glossarysubentryfield}[6]{%
5943   \GlossariesWarning
5944   {Deprecated use of \string\glossarysubentryfield.^^J
5945    I recommend you change to \string\subglossentry.^^J
5946    If you've just upgraded, try removing your gls auxiliary
5947    files^^J and recompile}%
5948   \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`

```
5949 \newcommand*{\glsgroupskip}{}%
```

Each of the 28 glossary groups described above is preceded by a group heading. This is formatted by the command `\glsgroupheading` which takes one argument which is the *label* assigned to that group (not the title). The corresponding labels are: `glssymbols`, `glsnumbers`, A, ..., Z. Glossary styles must redefine this command. (In between groups, `\glsgroupheading` comes immediately after `\glsgroupskip`.)

`\glsgroupheading`

```
5950 \newcommand*{\glsgroupheading}[1]{}
```

It is possible to “trick” `makeindex` into treating entries as though they belong to the same group, even if the terms don’t start with the same letter, by modifying the sort key. For example, all entries belonging to one group could be defined so that the sort key starts with an a, while entries belonging to another group could be defined so that the sort key starts with a b, and so on. If you want each group to have a heading, you would then need to modify the translation control sequences `\glsgetgrouptitle` and `\glsgetgrouplabel` so that the label is translated into the required title (and vice-versa).

`\glsgetgrouptitle{<label>}`

This command produces the title for the glossary group whose label is given by *<label>*. By default, the group labelled `glssymbols` produces `\glssymbolsgroupname`, the group labelled `glsnumbers` produces `\glsnumbersgroupname` and all the other groups simply produce their label. As mentioned above, the group labels are: `glssymbols`, `glsnumbers`, A, ..., Z. If you want to redefine the group titles, you will need to redefine this command. Languages

other than English may produce labels that are non-expandable, so we need to check for that otherwise it will create a “missing \endcsname inserted” error.

lsgetgrouptitle

```
5951 \newcommand*{\glsgetgrouptitle}[1]{%
5952   \@gls@getgrouptitle{#1}{\@gls@grptitle}%
5953   \@gls@grptitle
5954 }
```

s@getgrouptitle Gets the group title specified by the label (first argument) and stores in the second argument, which must be a control sequence.

```
5955 \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.

```
5956 \dtl@ifsingle{#1}%
5957 {%
5958   \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
5959 }%
5960 {%
5961   \ifboolexpr{test{\ifstrequal{#1}{glssymbols}}
5962               or test{\ifstrequal{#1}{glsnumbers}}}%
5963   {%
5964     \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
5965   }%
5966   {%
5967     \def#2{#1}%
5968   }%
5969 }%
5970 }
```

othergrouptitle Version for the no-indexing app option:

```
5971 \newcommand*{\@gls@noidx@getgrouptitle}[2]{%
5972   \DTLifint{#1}%
5973   {\edef#2{\char#1\relax}}%
5974   {%
5975     \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
5976   }%
5977 }
```

`\glsgetgrouplabel{<title>}`

This command does the reverse to the previous command. The argument is the group title, and it produces the group label. Note that if you redefine \glsgetgrouptitle, you will also need to redefine \glsgetgrouplabel.

lsgetgrouplabel

```
5978 \newcommand*{\glsgetgrouplabel}[1]{%
```

```

5979 \ifthenelse{\equal{#1}{\glssymbolsgroupname}}{\glssymbols}{%
5980 \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`

```

5981 \newcommand*{\setentrycounter}[2][]{%
5982   \def\@glo@counterprefix{#1}%
5983   \ifx\@glo@counterprefix\empty
5984     \def\@glo@counterprefix{.}%
5985   \else
5986     \def\@glo@counterprefix{.#1.}%
5987   \fi
5988   \def\glsentrycounter{#2}%
5989 }

```

The current glossary style can be set using `\setglossarystyle{<style>}`.

`\setglossarystyle`

```

5990 \newcommand*{\setglossarystyle}[1]{%
5991   \ifcsundef{@glsstyle@#1}%
5992   {%
5993     \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
5994   }%
5995   {%
5996     \csname @glsstyle@#1\endcsname
5997   }%

```

Set the default style if it's not already set.

```

5998   \ifx\@glossary@default@style\relax
5999     \protected@edef\@glossary@default@style{#1}%
6000   \fi
6001 }

```

`\glossarystyle`

```

6002 \newcommand*{\glossarystyle}[1]{%
6003   \ifcsundef{@glsstyle@#1}%
6004   {%
6005     \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
6006   }%
6007   {%
6008     \GlossariesWarning
6009     {Deprecated command \string\glossarystyle.^~J
6010      I recommend you switch to \string\setglossarystyle\space unless
6011      you want to maintain backward compatibility}%
6012     \setglossentrycompatibility
6013     \csname @glsstyle@#1\endcsname

```



```

6014 \ifcsdef{@glscompstyle@#1}%
6015 {\setglossentrycompatibility\csuse{@glscompstyle@#1}}%
6016 {}%
6017 }%

```

Set the default style if it isn't already set so that `\printglossary` can warn if the fallback style is in use.

```

6018 \ifx@glossary@default@style\relax
6019 \protected@edef@glossary@default@style{#1}%
6020 \fi
6021 }

```

`\newglossarystyle` New glossary styles can be defined using:

```
\newglossarystyle{<name>}{<definition>}
```

The *<definition>* argument should redefine `\theglossary`, `\glossaryheader`, `\glsgroupheading`, `\glossaryentryfield` and `\glsgroupskip` (see [section 1.19](#) for the definitions of predefined styles). Glossary styles should not redefine `\glossarypreamble` and `\glossarypostamble`, as the user should be able to switch between styles without affecting the pre- and postambles.

```

6022 \newcommand{\newglossarystyle}[2]{%
6023 \ifcsundef{@glsstyle@#1}%
6024 {%
6025 \expandafter\def\csname @glsstyle@#1\endcsname{#2}%
6026 }%
6027 {%
6028 \PackageError{glossaries}{Glossary style ‘#1’ is already defined}{}%
6029 }%
6030 }

```

`\newglossarystyle` Code for this macro supplied by Marco Daniel.

```

6031 \newcommand{\renewglossarystyle}[2]{%
6032 \ifcsundef{@glsstyle@#1}%
6033 {%
6034 \PackageError{glossaries}{Glossary style ‘#1’ isn’t already defined}{}%
6035 }%
6036 {%
6037 \csdef{@glsstyle@#1}{#2}%
6038 }%
6039 }

```

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`

```
6040 \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`

```
6041 \ifcsundef{hyperlink}%
6042 {%
6043   \def\glshypernumber#1{#1}%
6044 }%
6045 {%
6046   \def\glshypernumber#1{\@glshypernumber#1\nohyperpage{}}\@nil}
6047 }
```

`@glshypernumber` This code was provided by Heiko Oberdiek to allow material to be attached to the location.

```
6048 \def\@glshypernumber#1\nohyperpage#2#3\@nil{%
6049   \ifx\#1\%
6050     \else
6051       \@delimR#1\delimR\delimR\%
6052     \fi
6053   \ifx\#2\%
6054     \else
6055       #2%
6056     \fi
6057   \ifx\#3\%
6058     \else
6059       \@glshypernumber#3\@nil
6060     \fi
6061 }
```

`\@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`

```
6062 \def\@delimR#1\delimR #2\delimR #3\%
6063 \ifx\#2\%
6064   \@delimN{#1}%
6065 \else
6066   \delimR{#1}%
6067 \fi
```

```

6065 \else
6066   \@gls@numberlink{#1}\delimR\@gls@numberlink{#2}%
6067 \fi}

```

\@delimN displays a list of individual numbers, instead of a range:

\@delimN

```

6068 \def\@delimN#1{\@delimN#1\delimN \delimN\}
6069 \def\@delimN#1\delimN #2\delimN#3\{\{%
6070 \ifx\#3\}%
6071   \@gls@numberlink{#1}%
6072 \else
6073   \@gls@numberlink{#1}\delimN\@gls@numberlink{#2}%
6074 \fi
6075 }

```

The following code is modified from hyperref's \HyInd@pagelink where the name of the counter being used is given by \@gls@counter.

```

6076 \def\@gls@numberlink#1{%
6077 \begingroup
6078 \toks@={}%
6079 \@gls@removespaces#1 \@nil
6080 \endgroup}

6081 \def\@gls@removespaces#1 #2\@nil{%
6082 \toks@=\expandafter{\the\toks@#1}%
6083 \ifx\#2\}%
6084   \edef\x{\the\toks@}%
6085   \ifx\x\empty
6086     \else

6087       \hyperlink{\glstrycounter\@glo@counterprefix\the\toks@}%
6088         {\the\toks@}%
6089     \fi
6090   \else
6091     \@gls@ReturnAfterFi{%
6092       \@gls@removespaces#2\@nil
6093     }%
6094   \fi
6095 }
6096 \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

```

6097 \newcommand*{\hyperrm}[1]{\textrm{\glshypernumber{#1}}}

```

\hypersf

```

6098 \newcommand*{\hypersf}[1]{\textsf{\glshypernumber{#1}}}

```

```

\hypertt
6099 \newcommand*{\hypertt}[1]{\texttt{\glshypernumber{#1}}}

\hyperbf
6100 \newcommand*{\hyperbf}[1]{\textbf{\glshypernumber{#1}}}

\hypermd
6101 \newcommand*{\hypermd}[1]{\textmd{\glshypernumber{#1}}}

\hyperit
6102 \newcommand*{\hyperit}[1]{\textit{\glshypernumber{#1}}}

\hypersl
6103 \newcommand*{\hypersl}[1]{\textsl{\glshypernumber{#1}}}

\hyperup
6104 \newcommand*{\hyperup}[1]{\textup{\glshypernumber{#1}}}

\hypersc
6105 \newcommand*{\hypersc}[1]{\textsc{\glshypernumber{#1}}}

\hyperemph
6106 \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.

```

6107 \newcommand{\oldacronym}[4][\gls@label]{%
6108   \def\gls@label{#2}%
6109   \newacronym[#4]{#1}{#2}{#3}%
6110   \ifcsundef{xspace}%

```

```

6111 {%
6112   \expandafter\edef\csname#1\endcsname{%
6113     \noexpand\@ifstar{\noexpand\Gls{#1}}{\noexpand\gls{#1}}}%
6114   }%
6115 }%
6116 {%
6117   \expandafter\edef\csname#1\endcsname{%
6118     \noexpand\@ifstar{\noexpand\Gls{#1}\noexpand\xspace}{%
6119       \noexpand\gls{#1}\noexpand\xspace}%
6120     }%
6121   }%
6122 }

```

`\newacronym[<key-val list>]{<label>}{<abbrev>}{<long>}`

This is a quick way of defining acronyms, using `\newglossaryentry` with the appropriate values. It sets the glossary type to `\acronymtype` which will be acronym if the package option `acronym` has been used, otherwise it will be the default glossary. Since `\newacronym` merely calls `\newglossaryentry`, the acronym is treated like any other glossary entry.

If you prefer a different format, you can redefine `\newacronym` as required. The optional argument can be used to override any of the settings.

This is just a stub. It's redefined by commands like `\SetDefaultAcronymStyle`.

`\newacronym`

```

6123 \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.

```

6124 \newcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}

```

If `garamondx` has been loaded, need to use `\textulc` instead of `\textup`.

`\glstextup`

```

6125 \newrobustcmd*{\glstextup}[1]{\ifdef\textulc{\textulc{#1}}{\textup{#1}}}

```

The following are defined for compatibility with version 2.07 and earlier.

`\glsshortkey`

```

6126 \newcommand*{\glsshortkey}{short}

```

`\sshortpluralkey`

```

6127 \newcommand*{\glsshortpluralkey}{shortplural}

```

```

\glslongkey
6128 \newcommand*{\glslongkey}{long}

\glslongpluralkey
6129 \newcommand*{\glslongpluralkey}{longplural}

\acrfull  Full form of the acronym.
6130 \newrobustcmd*{\acrfull}{\@gls@hyp@opt\ns@acrfull}

6131 \newcommand*\ns@acrfull[2] [] {%
6132   \new@ifnextchar[{\@acrfull{#1}{#2}}%
6133     {\@acrfull{#1}{#2} []}%
6134 }

\@acrfull  Low-level macro:
6135 \def\@acrfull#1#2[#3] {%
  Make it easier for acronym styles to change this:
6136   \acrfullfmt{#1}{#2}{#3}%
6137 }

  Using \acrlinkfullformat and \acrfullformat is now deprecated as it can cause com-
  plications with the first letter upper case variants, but the package needs to provide backward
  compatibility support.

\acrfullfmt  No case change full format.
6138 \newcommand*{\acrfullfmt}[3] {%
6139   \acrlinkfullformat{\@acrlong}{\@acrshort}{#1}{#2}{#3}%
6140 }

\acrlinkfullformat  Format for full links like \acrfull. Syntax: \acrlinkfullformat{<long cs>}{<short cs>}{<options>}{<label>}{<in
6141 \newcommand{\acrlinkfullformat}[5] {%
6142   \acrfullformat{#1}{#3}{#4}[#5]{#2}{#3}{#4} []}%
6143 }

\acrfullformat  Default full form is <long> (<short>).
6144 \newcommand{\acrfullformat}[2] {#1\glsspace(#2)}

\glsspace  Robust space to ensure it's written to the .glsdefs file.
6145 \newrobustcmd{\glsspace}{\space}

  Default format for full acronym

\Acrfull
6146 \newrobustcmd*{\Acrfull}{\@gls@hyp@opt\ns@Acrfull}

6147 \newcommand*\ns@Acrfull[2] [] {%
6148   \new@ifnextchar[{\@Acrfull{#1}{#2}}%
6149     {\@Acrfull{#1}{#2} []}%
6150 }

```

Low-level macro:

```
6151 \def\@Acrfull#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6152 \Acrfullfmt{#1}{#2}{#3}%  
6153 }
```

`\Acrfullfmt` First letter upper case full format.

```
6154 \newcommand*\@Acrfullfmt[3]{%  
6155 \acrlinkfullformat{\@Acrlong}{\@acrshort}{#1}{#2}{#3}%  
6156 }
```

`\ACRfull`

```
6157 \newrobustcmd*\@ACRfull{\@gls@hyp@opt\ns@ACRfull}  
  
6158 \newcommand*\ns@ACRfull[2][{}]{%  
6159 \new@ifnextchar[{\@ACRfull{#1}{#2}}%  
6160 {\@ACRfull{#1}{#2}[]}%  
6161 }
```

Low-level macro:

```
6162 \def\@ACRfull#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6163 \ACRfullfmt{#1}{#2}{#3}%  
6164 }
```

`\ACRfullfmt` All upper case full format.

```
6165 \newcommand*\@ACRfullfmt[3]{%  
6166 \acrlinkfullformat{\@ACRlong}{\@ACRshort}{#1}{#2}{#3}%  
6167 }
```

Plural:

`\acrfullpl`

```
6168 \newrobustcmd*\@acrfullpl{\@gls@hyp@opt\ns@acrfullpl}  
  
6169 \newcommand*\ns@acrfullpl[2][{}]{%  
6170 \new@ifnextchar[{\@acrfullpl{#1}{#2}}%  
6171 {\@acrfullpl{#1}{#2}[]}%  
6172 }
```

Low-level macro:

```
6173 \def\@acrfullpl#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6174 \acrfullplfmt{#1}{#2}{#3}%  
6175 }
```

```

\acrfullplfmt  No case change plural full format.
6176 \newcommand*{\acrfullplfmt}[3]{%
6177   \acrlinkfullformat{\@acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%
6178 }

```

\Acrfullpl

```

6179 \newrobustcmd*{\Acrfullpl}{\@gls@hyp@opt\ns@Acrfullpl}

6180 \newcommand*\ns@Acrfullpl[2][{}]{%
6181   \new@ifnextchar[{\@Acrfullpl{#1}{#2}}]{%
6182     {\@Acrfullpl{#1}{#2}[]}%
6183 }

```

Low-level macro:

```

6184 \def\@Acrfullpl#1#2[#3]{%
  Make it easier for acronym styles to change this:
6185   \Acrfullplfmt{#1}{#2}{#3}%
6186 }

```

\Acrfullplfmt First letter upper case plural full format.

```

6187 \newcommand*{\Acrfullplfmt}[3]{%
6188   \acrlinkfullformat{\@Acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%
6189 }

```

\ACRfullpl

```

6190 \newrobustcmd*{\ACRfullpl}{\@gls@hyp@opt\ns@ACRfullpl}

6191 \newcommand*\ns@ACRfullpl[2][{}]{%
6192   \new@ifnextchar[{\@ACRfullpl{#1}{#2}}]{%
6193     {\@ACRfullpl{#1}{#2}[]}%
6194 }

```

Low-level macro:

```

6195 \def\@ACRfullpl#1#2[#3]{%
  Make it easier for acronym styles to change this:
6196   \ACRfullplfmt{#1}{#2}{#3}%
6197 }

```

\ACRfullplfmt All upper case plural full format.

```

6198 \newcommand*{\ACRfullplfmt}[3]{%
6199   \acrlinkfullformat{\@ACRlongpl}{\@ACRshortpl}{#1}{#2}{#3}%
6200 }

```


1.18 Predefined acronym styles

`\acronymfont` This is only used with the additional acronym styles:
6201 `\newcommand{\acronymfont}[1]{#1}`

`\firstacronymfont` This is only used with the additional acronym styles:
6202 `\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.
6203 `\newcommand*{\acrnameformat}[2]{\acronymfont{#1}}`

Define some tokens used by `\newacronym`:

`\glskeylisttok`
6204 `\newtoks\glskeylisttok`

`\glslabeltok`
6205 `\newtoks\glslabeltok`

`\glsshorttok`
6206 `\newtoks\glsshorttok`

`\glslongtok`
6207 `\newtoks\glslongtok`

`\newacronymhook` Provide a hook for `\newacronym`:
6208 `\newcommand*{\newacronymhook}{}`

`\genericNewAcronym` New improved version of setting the acronym style.
6209 `\newcommand*{\SetGenericNewAcronym}{%`
Change the behaviour of `\Glsentryname` to workaround expansion issues that cause a problem for `\makefirstuc`
6210 `\let\@Gls@entryname\@Gls@acrentryname`
Change the way acronyms are defined:
6211 `\renewcommand{\newacronym}[4][\{%`
6212 `\ifdefempty{\@glsacronymlists}%`
6213 `{%`
6214 `\def\@glo@type{\acronymtype}%`
6215 `\setkeys{glossentry}{##1}%`
6216 `\DeclareAcronymList{\@glo@type}%`
6217 `}%`
6218 `{}}%`
6219 `\glskeylisttok{##1}%`
6220 `\glslabeltok{##2}%`
6221 `\glsshorttok{##3}%`
6222 `\glslongtok{##4}%`

```

6223 \newacronymhook
6224 \protected@edef\@do@newglossaryentry{%
6225   \noexpand\newglossaryentry{\the\glslabeltok}%
6226   {%
6227     type=\acronymtype,%
6228     name={\expandonce{\acronymentry{##2}}},%
6229     sort={\acronymsort{\the\glsshorttok}{\the\glslongtok}},%
6230     text={\the\glsshorttok},%
6231     short={\the\glsshorttok},%
6232     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6233     long={\the\glslongtok},%
6234     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6235     \GenericAcronymFields,%
6236     \the\glskeylisttok
6237   }%
6238 }%
6239 \@do@newglossaryentry
6240 }%

```

Make sure that \acrfull etc reflects the new style:

```

6241 \renewcommand*{\acrfullfmt}[3]{%
6242   \glslink[##1]{##2}{\genacrfullformat{##2}{##3}}}%
6243 \renewcommand*{\Acrfullfmt}[3]{%
6244   \glslink[##1]{##2}{\Genacrfullformat{##2}{##3}}}%
6245 \renewcommand*{\ACRfullfmt}[3]{%
6246   \glslink[##1]{##2}{%
6247     \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}%
6248 \renewcommand*{\acrfullplfmt}[3]{%
6249   \glslink[##1]{##2}{\genplacrfullformat{##2}{##3}}}%
6250 \renewcommand*{\Acrfullplfmt}[3]{%
6251   \glslink[##1]{##2}{\Genplacrfullformat{##2}{##3}}}%
6252 \renewcommand*{\ACRfullplfmt}[3]{%
6253   \glslink[##1]{##2}{%
6254     \mfirstucMakeUppercase{\genplacrfullformat{##2}{##3}}}%

```

Make sure that \glsentryfull etc reflects the new style:

```

6255 \renewcommand*{\glsentryfull}[1]{\genacrfullformat{##1}{}}%
6256 \renewcommand*{\Glsentryfull}[1]{\Genacrfullformat{##1}{}}%
6257 \renewcommand*{\glsentryfullpl}[1]{\genplacrfullformat{##1}{}}%
6258 \renewcommand*{\Glsentryfullpl}[1]{\Genplacrfullformat{##1}{}}%
6259 }

```

\icAcronymFields Fields used by \SetGenericNewAcronym that can be changed by the acronym style.

```

6260 \newcommand*{\GenericAcronymFields}{description={\the\glslongtok}}

```

\acronymentry `\acronymentry{<label>}`

Display style for the name field in the list of acronyms.

```

6261 \newcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{#1}}}

```

`\acronymsort` `\acronymsort{<short>}{<long>}`

Default sort format for acronyms.

```
6262 \newcommand*{\acronymsort}[2]{#1}
```

`\setacronymstyle` `\setacronymstyle{<style name>}`

```
6263 \newcommand*{\setacronymstyle}[1]{%
6264   \ifcsundef{@glsacr@dispstyle@#1}
6265   {%
6266     \PackageError{glossaries}{Undefined acronym style ‘#1’}{}%
6267   }%
6268   {%
6269     \ifdefempty{\@glsacronymlists}%
6270     {%
6271       \DeclareAcronymList{\acronymtype}%
6272     }%
6273     {}%
6274     \SetGenericNewAcronym
6275     \GlsUseAcrStyleDefs{#1}%
6276     \@for\@gls@type:=\@glsacronymlists\do{%
6277       \defglentryfmt[\@gls@type]{\GlsUseAcrEntryDispStyle{#1}}%
6278     }%
6279   }%
6280 }
```

`\newacronymstyle` `\newacronymstyle{<style name>}{<entry format definition>}{<display definitions>}`

Defines a new acronym style called *<style name>*.

```
6281 \newcommand*{\newacronymstyle}[3]{%
6282   \ifcsdef{@glsacr@dispstyle@#1}%
6283   {%
6284     \PackageError{glossaries}{Acronym style ‘#1’ already exists}{}%
6285   }%
6286   {%
6287     \csdef{@glsacr@dispstyle@#1}{#2}%
6288     \csdef{@glsacr@styledefs@#1}{#3}%
6289   }%
6290 }
```

`\renewacronymstyle` Redefines the given acronym style.

```
6291 \newcommand*{\renewacronymstyle}[3]{%
6292   \ifcsdef{@glsacr@dispstyle@#1}%
6293   {%
```

```

6294 \csdef{@glsacr@dispstyle@#1}{#2}%
6295 \csdef{@glsacr@styledefs@#1}{#3}%
6296 }%
6297 {%
6298 \PackageError{glossaries}{Acronym style ‘#1’ doesn’t exist}{}%
6299 }%
6300 }

```

rEntryDispStyle

```

6301 \newcommand*{\GlsUseAcrEntryDispStyle}[1]{\csuse{@glsacr@dispstyle@#1}}

```

UseAcrStyleDefs

```

6302 \newcommand*{\GlsUseAcrStyleDefs}[1]{\csuse{@glsacr@styledefs@#1}}

```

Predefined acronym styles:

long-short *<long>* (*<short>*) acronym style.

```

6303 \newacronymstyle{long-short}%
6304 {%

```

Check for long form in case this is a mixed glossary.

```

6305 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6306 }%
6307 {%
6308 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6309 \renewcommand*{\genacrfullformat}[2]{%
6310 \glsentrylong{##1}##2\space
6311 (\protect\firstacronymfont{\glsentryshort{##1}})%
6312 }%
6313 \renewcommand*{\Genacrfullformat}[2]{%
6314 \Glsentrylong{##1}##2\space
6315 (\protect\firstacronymfont{\glsentryshort{##1}})%
6316 }%
6317 \renewcommand*{\genplacrfullformat}[2]{%
6318 \glsentrylongpl{##1}##2\space
6319 (\protect\firstacronymfont{\glsentryshortpl{##1}})%
6320 }%
6321 \renewcommand*{\Genplacrfullformat}[2]{%
6322 \Glsentrylongpl{##1}##2\space
6323 (\protect\firstacronymfont{\glsentryshortpl{##1}})%
6324 }%
6325 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6326 \renewcommand*{\acronymsort}[2]{##1}%
6327 \renewcommand*{\acronymfont}[1]{##1}%
6328 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
6329 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6330 }

```

long-sp-short Similar to the previous style but allows the space between the long and short form to be customized.

```

6331 \newacronymstyle{long-sp-short}%
6332 {%
    Check for long form in case this is a mixed glossary.
6333 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6334 }%
6335 {%
6336 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6337 \renewcommand*{\genacrfullformat}[2]{%
6338 \glentrylong{##1}##2\glsacspace{##1}%
6339 (\protect\firstacronymfont{\glentryshort{##1}})%
6340 }%
6341 \renewcommand*{\Genacrfullformat}[2]{%
6342 \Glsentrylong{##1}##2\glsacspace{##1}%
6343 (\protect\firstacronymfont{\glentryshort{##1}})%
6344 }%
6345 \renewcommand*{\genplacrfullformat}[2]{%
6346 \glentrylongpl{##1}##2\glsacspace{##1}%
6347 (\protect\firstacronymfont{\glentryshortpl{##1}})%
6348 }%
6349 \renewcommand*{\Genplacrfullformat}[2]{%
6350 \Glsentrylongpl{##1}##2\glsacspace{##1}%
6351 (\protect\firstacronymfont{\glentryshortpl{##1}})%
6352 }%
6353 \renewcommand*{\acronymentry}[1]{\acronymfont{\glentryshort{##1}}}%
6354 \renewcommand*{\acronymsort}[2]{##1}%
6355 \renewcommand*{\acronymfont}[1]{##1}%
6356 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
6357 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6358 }

```

`\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.

```

6359 \newcommand*{\glsacspace}[1]{%
6360 \settowidth{\dimen@}{(\firstacronymfont{\glentryshort{##1}})}%
6361 \ifdim\dimen@<3em~\else\space\fi
6362 }

```

`short-long` (*short*) (*long*) acronym style.

```

6363 \newacronymstyle{short-long}%
6364 {%
    Check for long form in case this is a mixed glossary.
6365 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6366 }%
6367 {%
6368 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6369 \renewcommand*{\genacrfullformat}[2]{%
6370 \protect\firstacronymfont{\glentryshort{##1}}##2\space
6371 (\glentrylong{##1})%

```

```

6372 }%
6373 \renewcommand*{\Genacrfullformat}[2]{%
6374   \protect\firstacronymfont{\Glsentryshort{##1}}##2\space
6375   (\glsentrylong{##1})%
6376 }%
6377 \renewcommand*{\genplacrfullformat}[2]{%
6378   \protect\firstacronymfont{\glsentryshortpl{##1}}##2\space
6379   (\glsentrylongpl{##1})%
6380 }%
6381 \renewcommand*{\Genplacrfullformat}[2]{%
6382   \protect\firstacronymfont{\Glsentryshortpl{##1}}##2\space
6383   (\glsentrylongpl{##1})%
6384 }%

6385 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6386 \renewcommand*{\acronymsort}[2]{##1}%
6387 \renewcommand*{\acronymfont}[1]{##1}%
6388 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
6389 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6390 }

```

long-sc-short *<long>* (\textsc{<short>}) acronym style.

```

6391 \newacronymstyle{long-sc-short}%
6392 {%
6393   \GlsUseAcrEntryDisplayStyle{long-short}%
6394 }%
6395 {%
6396   \GlsUseAcrStyleDefs{long-short}%
6397   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6398   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6399 }

```

long-sm-short *<long>* (\textsmaller{<short>}) acronym style.

```

6400 \newacronymstyle{long-sm-short}%
6401 {%
6402   \GlsUseAcrEntryDisplayStyle{long-short}%
6403 }%
6404 {%
6405   \GlsUseAcrStyleDefs{long-short}%
6406   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6407   \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6408 }

```

sc-short-long *<short>* (\textsc{<long>}) acronym style.

```

6409 \newacronymstyle{sc-short-long}%
6410 {%
6411   \GlsUseAcrEntryDisplayStyle{short-long}%
6412 }%
6413 {%

```

```

6414 \GlsUseAcrStyleDefs{short-long}%
6415 \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6416 \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6417 }

```

sm-short-long *<short>* (*\textsmaller{<long>}*) acronym style.

```

6418 \newacronymstyle{sm-short-long}%
6419 {%
6420 \GlsUseAcrEntryDisplayStyle{short-long}%
6421 }%
6422 {%
6423 \GlsUseAcrStyleDefs{short-long}%
6424 \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6425 \renewcommand*{\acrpluralsuffix}{\glacrpluralsuffix}%
6426 }

```

long-short-desc *<long>* (*{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

6427 \newacronymstyle{long-short-desc}%
6428 {%
6429 \GlsUseAcrEntryDisplayStyle{long-short}%
6430 }%
6431 {%
6432 \GlsUseAcrStyleDefs{long-short}%
6433 \renewcommand*{\GenericAcronymFields}{}%
6434 \renewcommand*{\acronymsort}[2]{##2}%
6435 \renewcommand*{\acronymentry}[1]{%
6436 \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6437 }

```

g-sp-short-desc *<long>* (*{<short>}*) acronym style that has an accompanying description (which the user needs to supply). The space between the long and short form is given by `\glsacspace`.

```

6438 \newacronymstyle{long-sp-short-desc}%
6439 {%
6440 \GlsUseAcrEntryDisplayStyle{long-sp-short}%
6441 }%
6442 {%
6443 \GlsUseAcrStyleDefs{long-sp-short}%
6444 \renewcommand*{\GenericAcronymFields}{}%
6445 \renewcommand*{\acronymsort}[2]{##2}%
6446 \renewcommand*{\acronymentry}[1]{%
6447 \glentrylong{##1}\glsacspace{##1}(\acronymfont{\glentryshort{##1}})}%
6448 }

```

g-sc-short-desc *<long>* (*\textsc{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

6449 \newacronymstyle{long-sc-short-desc}%
6450 {%

```

```

6451 \GlsUseAcrEntryDispStyle{long-sc-short}%
6452 }%
6453 {%
6454 \GlsUseAcrStyleDefs{long-sc-short}%
6455 \renewcommand*{\GenericAcronymFields}{}%
6456 \renewcommand*{\acronymsort}[2]{##2}%
6457 \renewcommand*{\acronymentry}[1]{%
6458     \glstrylong{##1}\space (\acronymfont{\glstryshort{##1}})}%
6459 }

```

g-sm-short-desc *<long>* (\textsmaller{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

6460 \newacronymstyle{long-sm-short-desc}%
6461 {%
6462 \GlsUseAcrEntryDispStyle{long-sm-short}%
6463 }%
6464 {%
6465 \GlsUseAcrStyleDefs{long-sm-short}%
6466 \renewcommand*{\GenericAcronymFields}{}%
6467 \renewcommand*{\acronymsort}[2]{##2}%
6468 \renewcommand*{\acronymentry}[1]{%
6469     \glstrylong{##1}\space (\acronymfont{\glstryshort{##1}})}%
6470 }

```

short-long-desc *<short>* ({<long>}) acronym style that has an accompanying description (which the user needs to supply).

```

6471 \newacronymstyle{short-long-desc}%
6472 {%
6473 \GlsUseAcrEntryDispStyle{short-long}%
6474 }%
6475 {%
6476 \GlsUseAcrStyleDefs{short-long}%
6477 \renewcommand*{\GenericAcronymFields}{}%
6478 \renewcommand*{\acronymsort}[2]{##2}%
6479 \renewcommand*{\acronymentry}[1]{%
6480     \glstrylong{##1}\space (\acronymfont{\glstryshort{##1}})}%
6481 }

```

short-long-desc *<long>* (\textsc{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

6482 \newacronymstyle{sc-short-long-desc}%
6483 {%
6484 \GlsUseAcrEntryDispStyle{sc-short-long}%
6485 }%
6486 {%
6487 \GlsUseAcrStyleDefs{sc-short-long}%
6488 \renewcommand*{\GenericAcronymFields}{}%
6489 \renewcommand*{\acronymsort}[2]{##2}%
6490 \renewcommand*{\acronymentry}[1]{%

```



```

6491 \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6492 }

```

short-long-desc <long> (\textsmaller{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

6493 \newacronymstyle{sm-short-long-desc}%
6494 {%
6495 \GlsUseAcrEntryDispStyle{sm-short-long}%
6496 }%
6497 {%
6498 \GlsUseAcrStyleDefs{sm-short-long}%
6499 \renewcommand*{\GenericAcronymFields}{}%
6500 \renewcommand*{\acronymsort}[2]{##2}%
6501 \renewcommand*{\acronymentry}[1]{%
6502 \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6503 }

```

dua <long> only acronym style.

```

6504 \newacronymstyle{dua}%
6505 {%

```

Check for long form in case this is a mixed glossary.

```

6506 \ifdefempty\glscustomtext
6507 {%
6508 \ifglshaslong{\glslabel}%
6509 {%
6510 \glsifplural
6511 {%

```

Plural form:

```

6512 \glscapscase
6513 {%

```

Plural form, don't adjust case:

```

6514 \glsentrylongpl{\glslabel}\glsinsert
6515 }%
6516 {%

```

Plural form, make first letter upper case:

```

6517 \Glsentrylongpl{\glslabel}\glsinsert
6518 }%
6519 {%

```

Plural form, all caps:

```

6520 \mfirstucMakeUppercase
6521 {\glsentrylongpl{\glslabel}\glsinsert}%
6522 }%
6523 }%
6524 {%

```

Singular form

```
6525      \glscapscase
6526      {%
```

Singular form, don't adjust case:

```
6527      \glentrylong{\glslabel}\glsinsert
6528      }%
6529      {%
```

Subsequent singular form, make first letter upper case:

```
6530      \Glsentrylong{\glslabel}\glsinsert
6531      }%
6532      {%
```

Subsequent singular form, all caps:

```
6533      \mfirstucMakeUppercase
6534      {\glentrylong{\glslabel}\glsinsert}%
6535      }%
6536      }%
6537      }%
6538      {%
```

Not an acronym:

```
6539      \glsgenentryfmt
6540      }%
6541      }%
6542      {\glscustomtext\glsinsert}%
6543      }%
6544      {%
6545      \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%

6546      \renewcommand*{\acrfullfmt}[3]{%
6547      \glslink[##1]{##2}{\glentrylong{##2}##3\space
6548      (\acronymfont{\glentryshort{##2}})}}%
6549      \renewcommand*{\Acrfullfmt}[3]{%
6550      \glslink[##1]{##2}{\Glsentrylong{##2}##3\space
6551      (\acronymfont{\glentryshort{##2}})}}%
6552      \renewcommand*{\ACRfullfmt}[3]{%
6553      \glslink[##1]{##2}{%
6554      \mfirstucMakeUppercase{\glentrylong{##2}##3\space
6555      (\acronymfont{\glentryshort{##2}})}}}%

6556      \renewcommand*{\acrfullplfmt}[3]{%
6557      \glslink[##1]{##2}{\glentrylongpl{##2}##3\space
6558      (\acronymfont{\glentryshortpl{##2}})}}%

6559      \renewcommand*{\Acrfullplfmt}[3]{%
6560      \glslink[##1]{##2}{\Glsentrylongpl{##2}##3\space
6561      (\acronymfont{\glentryshortpl{##2}})}}%
6562      \renewcommand*{\ACRfullplfmt}[3]{%
6563      \glslink[##1]{##2}{%
```

```

6564      \mfirstucMakeUppercase{\glentrylongpl{##2}##3\space
6565      (\acronymfont{\glentryshortpl{##2}})}}}%
6566 \renewcommand*{\glentryfull}[1]{%
6567   \glentrylong{##1}\space(\acronymfont{\glentryshort{##1}})%
6568 }%
6569 \renewcommand*{\Glsentryfull}[1]{%
6570   \Glsentrylong{##1}\space(\acronymfont{\glentryshort{##1}})%
6571 }%
6572 \renewcommand*{\glentryfullpl}[1]{%
6573   \glentrylongpl{##1}\space(\acronymfont{\glentryshortpl{##1}})%
6574 }%
6575 \renewcommand*{\Glsentryfullpl}[1]{%
6576   \Glsentrylongpl{##1}\space(\acronymfont{\glentryshortpl{##1}})%
6577 }%
6578 \renewcommand*{\acronymentry}[1]{\acronymfont{\glentryshort{##1}}}%
6579 \renewcommand*{\acronymsort}[2]{##1}%
6580 \renewcommand*{\acronymfont}[1]{##1}%
6581 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6582 }

```

dua-desc *<long>* only acronym style with user-supplied description.

```

6583 \newacronymstyle{dua-desc}%
6584 {%
6585   \GlsUseAcrEntryDispStyle{dua}%
6586 }%
6587 {%
6588   \GlsUseAcrStyleDefs{dua}%
6589   \renewcommand*{\GenericAcronymFields}{}%
6590   \renewcommand*{\acronymentry}[1]{\acronymfont{\glentrylong{##1}}}%
6591   \renewcommand*{\acronymsort}[2]{##2}%
6592 }%

```

footnote *<short>*\footnote{*<long>*} acronym style.

```

6593 \newacronymstyle{footnote}%
6594 {%
6595   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6596 }%
6597 {%
6598   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6599   \glshyperfirstfalse
6600   \renewcommand*{\genacrfullformat}[2]{%
6601     \protect\firstacronymfont{\glentryshort{##1}}##2%
6602     \protect\footnote{\glentrylong{##1}}%
6603   }%
6604   \renewcommand*{\Genacrfullformat}[2]{%

```

```

6605 \firstacronymfont{\Glsentryshort{##1}}##2%
6606 \protect\footnote{\glentrylong{##1}}%
6607 }%
6608 \renewcommand*{\genplacrfullformat}[2]{%
6609 \protect\firstacronymfont{\glentryshortpl{##1}}##2%
6610 \protect\footnote{\glentrylongpl{##1}}%
6611 }%
6612 \renewcommand*{\Genplacrfullformat}[2]{%
6613 \protect\firstacronymfont{\Glsentryshortpl{##1}}##2%
6614 \protect\footnote{\glentrylongpl{##1}}%
6615 }%
6616 \renewcommand*{\acronymentry}[1]{\acronymfont{\glentryshort{##1}}}%
6617 \renewcommand*{\acronymsort}[2]{##1}%
6618 \renewcommand*{\acronymfont}[1]{##1}%
6619 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%

```

Don't use footnotes for \acrfull:

```

6620 \renewcommand*{\acrfullfmt}[3]{%
6621 \glslink[##1]{##2}{\acronymfont{\glentryshort{##2}}##3\space
6622 (\glentrylong{##2})}%
6623 \renewcommand*{\Acrfullfmt}[3]{%
6624 \glslink[##1]{##2}{\acronymfont{\Glsentryshort{##2}}##3\space
6625 (\glentrylong{##2})}%
6626 \renewcommand*{\ACRfullfmt}[3]{%
6627 \glslink[##1]{##2}{%
6628 \mfirstucMakeUppercase{\acronymfont{\glentryshort{##2}}##3\space
6629 (\glentrylong{##2})}}}%
6630 \renewcommand*{\acrfullplfmt}[3]{%
6631 \glslink[##1]{##2}{\acronymfont{\glentryshortpl{##2}}##3\space
6632 (\glentrylongpl{##2})}%
6633 \renewcommand*{\Acrfullplfmt}[3]{%
6634 \glslink[##1]{##2}{\acronymfont{\Glsentryshortpl{##2}}##3\space
6635 (\glentrylongpl{##2})}%
6636 \renewcommand*{\ACRfullplfmt}[3]{%
6637 \glslink[##1]{##2}{%
6638 \mfirstucMakeUppercase{\acronymfont{\glentryshortpl{##2}}##3\space
6639 (\glentrylongpl{##2})}}}%

```

Similarly for \glentryfull etc:

```

6640 \renewcommand*{\glentryfull}[1]{%
6641 \acronymfont{\glentryshort{##1}}\space(\glentrylong{##1})}%
6642 \renewcommand*{\Glsentryfull}[1]{%
6643 \acronymfont{\Glsentryshort{##1}}\space(\glentrylong{##1})}%
6644 \renewcommand*{\glentryfullpl}[1]{%
6645 \acronymfont{\glentryshortpl{##1}}\space(\glentrylongpl{##1})}%
6646 \renewcommand*{\Glsentryfullpl}[1]{%
6647 \acronymfont{\Glsentryshortpl{##1}}\space(\glentrylongpl{##1})}%
6648 }

```

footnote-sc \textsc{<short>}\footnote{<long>} acronym style.

```

6649 \newacronymstyle{footnote-sc}%
6650 {%
6651   \GlsUseAcrEntryDisplayStyle{footnote}%
6652 }%
6653 {%
6654   \GlsUseAcrStyleDefs{footnote}%
6655   \renewcommand{\acronymentry}[1]{\acronymfont{\glentryshort{##1}}}
6656   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6657   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6658 }%

```

footnote-sm \textsmaller{<short>}\footnote{<long>} acronym style.

```

6659 \newacronymstyle{footnote-sm}%
6660 {%
6661   \GlsUseAcrEntryDisplayStyle{footnote}%
6662 }%
6663 {%
6664   \GlsUseAcrStyleDefs{footnote}%
6665   \renewcommand{\acronymentry}[1]{\acronymfont{\glentryshort{##1}}}
6666   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6667   \renewcommand*{\acrpluralsuffix}{\glacrpluralsuffix}%
6668 }%

```

footnote-desc <short>\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

6669 \newacronymstyle{footnote-desc}%
6670 {%
6671   \GlsUseAcrEntryDisplayStyle{footnote}%
6672 }%
6673 {%
6674   \GlsUseAcrStyleDefs{footnote}%
6675   \renewcommand*{\GenericAcronymFields}{}%
6676   \renewcommand*{\acronymsort}[2]{##2}%
6677   \renewcommand*{\acronymentry}[1]{%
6678     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6679 }

```

footnote-sc-desc \textsc{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

6680 \newacronymstyle{footnote-sc-desc}%
6681 {%
6682   \GlsUseAcrEntryDisplayStyle{footnote-sc}%
6683 }%
6684 {%
6685   \GlsUseAcrStyleDefs{footnote-sc}%
6686   \renewcommand*{\GenericAcronymFields}{}%
6687   \renewcommand*{\acronymsort}[2]{##2}%
6688   \renewcommand*{\acronymentry}[1]{%
6689     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%

```

6690 }

ootnote-sm-desc \textsmaller{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

6691 \newacronymstyle{footnote-sm-desc}%

6692 {%

6693 \GlsUseAcrEntryDispStyle{footnote-sm}%

6694 }%

6695 {%

6696 \GlsUseAcrStyleDefs{footnote-sm}%

6697 \renewcommand*{\GenericAcronymFields}{}%

6698 \renewcommand*{\acronymsort}[2]{##2}%

6699 \renewcommand*{\acronymentry}[1]{%

6700 \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%

6701 }

AcronymSynonyms

6702 \newcommand*{\DefineAcronymSynonyms}{%

Short form

\acs

6703 \let\acs\acrshort

First letter uppercase short form

\Acs

6704 \let\Acs\Acrshort

Plural short form

\acsp

6705 \let\acsp\acrshortpl

First letter uppercase plural short form

\Acsp

6706 \let\Acsp\Acrshortpl

Long form

\acl

6707 \let\acl\aclong

Plural long form

\aclp

6708 \let\aclp\aclongpl

First letter upper case long form

```

\Acl
6709 \let\Acl\Acrlong
      First letter upper case plural long form

\Aclp
6710 \let\Aclp\Acrlongpl
      Full form

\acf
6711 \let\acf\acrfull
      Plural full form

\acfp
6712 \let\acfp\acrfullpl
      First letter upper case full form

\Acf
6713 \let\Acf\Acrfull
      First letter upper case plural full form

\Acfp
6714 \let\Acfp\Acrfullpl
      Standard form

\ac
6715 \let\ac\gls
      First upper case standard form

\Ac
6716 \let\Ac\Gls
      Standard plural form

\acp
6717 \let\acp\glspl
      Standard first letter upper case plural form

\Acp
6718 \let\Acp\Glspl
6719 }
      Define synonyms if required
6720 \ifglsacrshortcuts
6721 \DefineAcronymSynonyms
6722 \fi

```

These commands for setting the style are now deprecated but are kept for backward compatibility.

`\setAcronymDisplayStyle` Sets the default acronym display style for given glossary.

```
6723 \newcommand*{\SetDefaultAcronymDisplayStyle}[1]{%
6724   \def\glsentryfmt[#1]{\glsentryfmt}%
6725 }
```

`\setNewAcronymDef` Sets up the acronym definition for the default style. The information is provided by the tokens `\glslabeltok`, `\glsshorttok`, `\glslongtok` and `\glskeylisttok`.

```
6726 \newcommand*{\DefaultNewAcronymDef}{%
6727   \edef\@do@newglossaryentry{%
6728     \noexpand\newglossaryentry{\the\glslabeltok}%
6729     {%
6730       type=\acronymtype,%
6731       name={\the\glsshorttok},%
6732       sort={\the\glsshorttok},%
6733       text={\the\glsshorttok},%
6734       first={\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
6735       plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6736       firstplural={\acrfullformat{\noexpand\expandonce\noexpand\@glo@longpl}%
6737                     {\noexpand\expandonce\noexpand\@glo@shortpl}},%
6738       short={\the\glsshorttok},%
6739       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6740       long={\the\glslongtok},%
6741       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6742       description={\the\glslongtok},%
6743       descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
```

Remaining options specified by the user:

```
6744       \the\glskeylisttok
6745     }%
6746   }%
6747   \let\@org@gls@assign@firstpl\gls@assign@firstpl
6748   \let\@org@gls@assign@plural\gls@assign@plural
6749   \let\@org@gls@assign@descplural\gls@assign@descplural
6750   \def\gls@assign@firstpl##1##2{%
6751     \@@gls@expand@field{##1}{firstpl}{##2}%
6752   }%
6753   \def\gls@assign@plural##1##2{%
6754     \@@gls@expand@field{##1}{plural}{##2}%
6755   }%
6756   \def\gls@assign@descplural##1##2{%
6757     \@@gls@expand@field{##1}{descplural}{##2}%
6758   }%
6759   \@do@newglossaryentry
6760   \let\gls@assign@firstpl\@org@gls@assign@firstpl
6761   \let\gls@assign@plural\@org@gls@assign@plural
6762   \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
6763 }
```


ultAcronymStyle Set up the default acronym style:

```
6764 \newcommand*{\SetDefaultAcronymStyle}{%
```

Set the display style:

```
6765 \@for\@gls@type:=\@glsacronymlists\do{%
```

```
6766 \SetDefaultAcronymDisplayStyle{\@gls@type}%
```

```
6767 }%
```

Set up the definition of \newacronym:

```
6768 \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).

```
6769 \ifx\@glsacronymlists\@empty
```

```
6770 \def\@glo@type{\acronymtype}%
```

```
6771 \setkeys{glossentry}{##1}%
```

```
6772 \DeclareAcronymList{\@glo@type}%
```

```
6773 \SetDefaultAcronymDisplayStyle{\@glo@type}%
```

```
6774 \fi
```

```
6775 \glskeylisttok{##1}%
```

```
6776 \glslabeltok{##2}%
```

```
6777 \glsshorttok{##3}%
```

```
6778 \glslongtok{##4}%
```

```
6779 \newacronymhook
```

```
6780 \DefaultNewAcronymDef
```

```
6781 }%
```

```
6782 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
```

```
6783 }
```

\acrfootnote Used by the footnote acronym styles.

```
6784 \newcommand*{\acrfootnote}[3]{\acrlinkfootnote{#1}{#2}{#3}}
```

acrlinkfootnote

```
6785 \newcommand*{\acrlinkfootnote}[3]{%
```

```
6786 \footnote{\glslink[#1]{#2}{#3}}%
```

```
6787 }
```

acrnolinkfootnote

```
6788 \newcommand*{\acrnolinkfootnote}[3]{%
```

```
6789 \footnote{#3}%
```

```
6790 }
```

acronymDisplayStyle Sets the acronym display style for given glossary for the description and footnote combination.

```
6791 \newcommand*{\SetDescriptionFootnoteAcronymDisplayStyle}[1]{%
```

```
6792 \defglsentryfmt[#1]{%
```

```
6793 \ifdefempty\glscustomtext
```

```
6794 {%
```

```
6795 \ifglsused{\glslabel}%
```

```

6796      {%
6797      \acronymfont{\glsgetentryfmt}%
6798      }%
6799      {%
6800      \firstacronymfont{\glsgetentryfmt}%
6801      \ifglshassymbol{\glslabel}%
6802      {%
6803      \expandafter\protect\expandafter\acrfootnote\expandafter
6804      {\@gls@link@opts}{\@gls@link@label}%
6805      {%
6806      \glsifplural
6807      {\glsentrysymbolplural{\glslabel}}%
6808      {\glsentrysymbol{\glslabel}}%
6809      }%
6810      }%
6811      }%
6812      }%
6813      {\glscustomtext\glsinsert}%
6814      }%
6815      }

```

teNewAcronymDef

```

6816 \newcommand*{\DescriptionFootnoteNewAcronymDef}{%
6817 \edef\@do@newglossaryentry{%
6818 \noexpand\newglossaryentry{\the\glslabeltok}%
6819 {%
6820 type=\acronymtype,%
6821 name={\noexpand\acronymfont{\the\glsshorttok}},%
6822 sort={\the\glsshorttok},%
6823 first={\the\glsshorttok},%
6824 firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6825 text={\the\glsshorttok},%
6826 plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6827 short={\the\glsshorttok},%
6828 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6829 long={\the\glslongtok},%
6830 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6831 symbol={\the\glslongtok},%
6832 symbolplural={\noexpand\expandonce\noexpand\@glo@longpl},%
6833 \the\glskeylisttok
6834 }%
6835 }%
6836 \let\@org@gls@assign@firstpl\gls@assign@firstpl
6837 \let\@org@gls@assign@plural\gls@assign@plural
6838 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
6839 \def\gls@assign@firstpl##1##2{%
6840 \@@gls@expand@field{##1}{firstpl}{##2}%
6841 }%
6842 \def\gls@assign@plural##1##2{%

```

```

6843 \@@gls@expand@field{##1}{plural}{##2}%
6844 }%
6845 \def\gls@assign@symbolplural##1##2{%
6846 \@@gls@expand@field{##1}{symbolplural}{##2}%
6847 }%
6848 \do@newglossaryentry
6849 \let\gls@assign@plural\@org@gls@assign@plural
6850 \let\gls@assign@firstpl\@org@gls@assign@firstpl
6851 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
6852 }

```

oteAcronymStyle If a description and footnote are both required, store the long form in the symbol key. Store the short form in text key. Note that since the long form is stored in the symbol key, if you want the long form to appear in the list of acronyms, you need to use a glossary style that displays the symbol key.

```

6853 \newcommand*\SetDescriptionFootnoteAcronymStyle{%
6854 \renewcommand{\newacronym}[4][\]{%
6855 \ifx\@glsacronymlists\@empty
6856 \def\@glo@type{\acronymtype}%
6857 \setkeys{glossentry}{##1}%
6858 \DeclareAcronymList{\@glo@type}%
6859 \SetDescriptionFootnoteAcronymDisplayStyle{\@glo@type}%
6860 \fi
6861 \glskeylisttok{##1}%
6862 \glslabeltok{##2}%
6863 \glsshorttok{##3}%
6864 \glslongtok{##4}%
6865 \newacronymhook
6866 \DescriptionFootnoteNewAcronymDef
6867 }%

```

If footnote package option is specified, set the first use to append the long form (stored in symbol) as a footnote.

```

6868 \@for\@gls@type:=\@glsacronymlists\do{%
6869 \SetDescriptionFootnoteAcronymDisplayStyle{\@gls@type}%
6870 }%

```

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.

```

6871 \ifglsacrsmallcaps
6872 \renewcommand*\acronymfont[1]{\textsc{##1}}%
6873 \renewcommand*\acrpluralsuffix{\glsupacrpluralsuffix}%
6874 \else
6875 \ifglsacrsmaller
6876 \renewcommand*\acronymfont[1]{\textsmaller{##1}}%
6877 \fi
6878 \fi

```

Check for package option clash

```

6879 \ifglsacrdue
6880 \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’
6881 can’t both be set}{}%
6882 \fi
6883 }%

```

nymDisplayStyle Sets the acronym display style for given glossary with description and dua combination.

```

6884 \newcommand*{\SetDescriptionDUAAcronymDisplayStyle}[1]{%
6885 \defglsentryfmt[#1]{\glsentryfmt}%
6886 }

```

UANewAcronymDef

```

6887 \newcommand*{\DescriptionDUANewAcronymDef}{%
6888 \edef\@do@newglossaryentry{%
6889 \noexpand\newglossaryentry{\the\glslabeltok}%
6890 {%
6891 type=\acronymtype,%
6892 name={\the\glslongtok},%
6893 sort={\the\glslongtok},%
6894 text={\the\glslongtok},%
6895 first={\the\glslongtok},%
6896 plural={\noexpand\expandonce\noexpand\@glo@longpl},%
6897 firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
6898 short={\the\glsshorttok},%
6899 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6900 long={\the\glslongtok},%
6901 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6902 symbol={\the\glsshorttok},%
6903 symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6904 \the\glskeylisttok
6905 }%
6906 }%
6907 \let\@org@gls@assign@firstpl\gls@assign@firstpl
6908 \let\@org@gls@assign@plural\gls@assign@plural
6909 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
6910 \def\gls@assign@firstpl##1##2{%
6911 \@@gls@expand@field{##1}{firstpl}{##2}%
6912 }%
6913 \def\gls@assign@plural##1##2{%
6914 \@@gls@expand@field{##1}{plural}{##2}%
6915 }%
6916 \def\gls@assign@symbolplural##1##2{%
6917 \@@gls@expand@field{##1}{symbolplural}{##2}%
6918 }%
6919 \@do@newglossaryentry
6920 \let\gls@assign@firstpl\@org@gls@assign@firstpl
6921 \let\gls@assign@plural\@org@gls@assign@plural
6922 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
6923 }

```

DUAACronymStyle Description, don't use acronym and no footnote. Note that the short form is stored in the symbol key, so if the short form needs to be displayed in the glossary, use a style the displays the symbol.

```

6924 \newcommand*{\SetDescriptionDUAACronymStyle}{%
6925   \ifglsmallcaps
6926     \PackageError{glossaries}{Option clash: 'smallcaps' and 'dua'
6927       can't both be set}{}%
6928   \else
6929     \ifglsmaller
6930       \PackageError{glossaries}{Option clash: 'smaller' and 'dua'
6931         can't both be set}{}%
6932     \fi
6933   \fi
6934   \renewcommand{\newacronym}[4][{}]{%
6935     \ifx\@glsacronymlists\@empty
6936       \def\@glo@type{\acronymtype}%
6937       \setkeys{glossentry}{##1}%
6938       \DeclareAcronymList{\@glo@type}%
6939       \SetDescriptionDUAACronymDisplayStyle{\@glo@type}%
6940     \fi
6941     \glskeylisttok{##1}%
6942     \glslabeltok{##2}%
6943     \glsshorttok{##3}%
6944     \glslongtok{##4}%
6945     \newacronymhook
6946     \DescriptionDUANewAcronymDef
6947   }%

```

Set display.

```

6948   \@for\@gls@type:=\@glsacronymlists\do{%
6949     \SetDescriptionDUAACronymDisplayStyle{\@gls@type}%
6950   }%
6951 }%

```

nymDisplayStyle Sets the acronym display style for given glossary using the description setting (but not footnote or dua).

```

6952 \newcommand*{\SetDescriptionAcronymDisplayStyle}[1]{%
6953   \defglentryfmt[#1]{%
6954     \ifdefempty\glscustomtext
6955     {%
6956       \ifglused{\glslabel}%
6957     }%

```

Move the inserted text outside of \acronymfont

```

6958       \let\gls@org@insert\glsinsert
6959       \let\glsinsert\@empty
6960       \acronymfont{\glsgenentryfmt}\gls@org@insert
6961     }%

```

```

6962     {%
6963         \glsgenentryfmt
6964         \ifglshassymbol{\glslabel}%
6965         {%
6966             \glsifplural
6967             {%
6968                 \def\@glo@symbol{\glsentrysymbolplural{\glslabel}}%
6969             }%
6970             {%
6971                 \def\@glo@symbol{\glsentrysymbol{\glslabel}}%
6972             }%
6973             \space(\protect\firstacronymfont
6974             {\glscapscase
6975              {\@glo@symbol}
6976              {\@glo@symbol}
6977              {\mfirstucMakeUppercase{\@glo@symbol}}})%
6978         }%
6979     }%
6980 }%
6981 }%
6982 {\glscustomtext\glsinsert}%
6983 }%
6984 }

```

onNewAcronymDef

```

6985 \newcommand*{\DescriptionNewAcronymDef}{%
6986     \edef\@do@newglossaryentry{%
6987         \noexpand\newglossaryentry{\the\glslabeltok}%
6988         {%
6989             type=\acronymtype,%
6990             name={\noexpand
6991                 \acrnameformat{\the\glsshorttok}{\the\glslongtok}},%
6992             sort={\the\glsshorttok},%
6993             first={\the\glslongtok},%
6994             firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
6995             text={\the\glsshorttok},%
6996             plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6997             short={\the\glsshorttok},%
6998             shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6999             long={\the\glslongtok},%
7000             longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7001             symbol={\noexpand\@glo@text},%
7002             symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7003             \the\glskeylisttok}%
7004     }%
7005     \let\@org@gls@assign@firstpl\gls@assign@firstpl
7006     \let\@org@gls@assign@plural\gls@assign@plural
7007     \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7008     \def\gls@assign@firstpl##1##2{%

```

```

7009 \@@gls@expand@field{##1}{firstpl}{##2}%
7010 }%
7011 \def\gls@assign@plural##1##2{%
7012 \@@gls@expand@field{##1}{plural}{##2}%
7013 }%
7014 \def\gls@assign@symbolplural##1##2{%
7015 \@@gls@expand@field{##1}{symbolplural}{##2}%
7016 }%
7017 \do@newglossaryentry
7018 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7019 \let\gls@assign@plural\@org@gls@assign@plural
7020 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7021 }

```

ionAcronymStyle Option description is used, but not dua or footnote. Store long form in first key and short form in text and symbol key. The name is stored using \acrnameformat to allow the user to override the way the name is displayed in the list of acronyms.

```

7022 \newcommand*{\SetDescriptionAcronymStyle}{%
7023 \renewcommand{\newacronym}[4][\]{%
7024 \ifx\@glsacronymlists\@empty
7025 \def\@glo@type{\acronymtype}%
7026 \setkeys{glossentry}{##1}%
7027 \DeclareAcronymList{\@glo@type}%
7028 \SetDescriptionAcronymDisplayStyle{\@glo@type}%
7029 \fi
7030 \glskeylisttok{##1}%
7031 \glslabeltok{##2}%
7032 \glsshorttok{##3}%
7033 \glslongtok{##4}%
7034 \newacronymhook
7035 \DescriptionNewAcronymDef
7036 }%

```

Set display.

```

7037 \@for\@gls@type:=\@glsacronymlists\do{%
7038 \SetDescriptionAcronymDisplayStyle{\@gls@type}%
7039 }%

```

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.

```

7040 \ifglsacrsmallcaps
7041 \renewcommand{\acronymfont}[1]{\textsc{##1}}
7042 \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7043 \else
7044 \ifglsacrsmaller
7045 \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
7046 \fi
7047 \fi
7048 }%

```

`\nymDisplayStyle` Sets the acronym display style for given glossary with footnote setting (but not description or dua).

```

7049 \newcommand*{\SetFootnoteAcronymDisplayStyle}[1]{%
7050   \defglentryfmt[#1]{%

7051     \ifdefempty\glscustomtext
7052     {%

        Move the inserted text outside of \acronymfont

7053       \let\gls@org@insert\glsinsert
7054       \let\glsinsert\@empty
7055       \ifglused{\glslabel}%
7056       {%
7057         \acronymfont{\glsgenentryfmt}\gls@org@insert
7058       }%
7059       {%
7060         \firstacronymfont{\glsgenentryfmt}\gls@org@insert
7061         \ifglshaslong{\glslabel}%
7062         {%
7063           \expandafter\protect\expandafter\acrfootnote\expandafter
7064           {\@gls@link@opts}{\@gls@link@label}%
7065           {%
7066             \glsifplural
7067             {\glsentrylongpl{\glslabel}}%
7068             {\glsentrylong{\glslabel}}%
7069           }%
7070         }%

7071       }%
7072     }%
7073   }%
7074   {\glscustomtext\glsinsert}%
7075 }%
7076 }

```

`\teNewAcronymDef`

```

7077 \newcommand*{\FootnoteNewAcronymDef}{%
7078   \edef\@do@newglossaryentry{%
7079     \noexpand\newglossaryentry{\the\glslabeltok}%
7080     {%
7081       type=\acronymtype,%
7082       name={\noexpand\acronymfont{\the\glsshorttok}},%
7083       sort={\the\glsshorttok},%
7084       text={\the\glsshorttok},%
7085       plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7086       first={\the\glsshorttok},%
7087       firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7088       short={\the\glsshorttok},%
7089       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7090       long={\the\glslongtok},%

```



```

7091     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7092     description={\the\glslongtok},%
7093     descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7094     \the\glskeylisttok
7095   }%
7096 }%
7097 \let\@org@gls@assign@plural\gls@assign@plural
7098 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7099 \let\@org@gls@assign@descplural\gls@assign@descplural
7100 \def\gls@assign@firstpl##1##2{%
7101   \@gls@expand@field{##1}{firstpl}{##2}%
7102 }%
7103 \def\gls@assign@plural##1##2{%
7104   \@gls@expand@field{##1}{plural}{##2}%
7105 }%
7106 \def\gls@assign@descplural##1##2{%
7107   \@gls@expand@field{##1}{descplural}{##2}%
7108 }%
7109 \do@newglossaryentry
7110 \let\gls@assign@plural\@org@gls@assign@plural
7111 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7112 \let\gls@assign@descplural\@org@gls@assign@descplural
7113 }

```

oteAcronymStyle If footnote package option is specified, set the first use to append the long form (stored in description) as a footnote. Use the description key to store the long form.

```

7114 \newcommand*{\SetFootnoteAcronymStyle}{%
7115   \renewcommand{\newacronym}[4][]{%
7116     \ifx\@glsacronymlists\@empty
7117       \def\@glo@type{\acronymtype}%
7118       \setkeys{glossentry}{##1}%
7119       \DeclareAcronymList{\@glo@type}%
7120       \SetFootnoteAcronymDisplayStyle{\@glo@type}%
7121     \fi
7122     \glskeylisttok{##1}%
7123     \glslabeltok{##2}%
7124     \glsshorttok{##3}%
7125     \glslongtok{##4}%
7126     \newacronymhook
7127     \FootnoteNewAcronymDef
7128   }%

```

Set display

```

7129   \@for\@gls@type:=\@glsacronymlists\do{%
7130     \SetFootnoteAcronymDisplayStyle{\@gls@type}%
7131   }%

```

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.

```

7132   \ifglsacrsmallcaps

```

```

7133     \renewcommand*{\acronymfont}[1]{\textsc{##1}}%
7134     \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7135 \else
7136     \ifglssacrsmaller
7137         \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
7138     \fi
7139 \fi

    Check for option clash
7140 \ifglssacrdua
7141     \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’
7142         can’t both be set}{}%
7143 \fi
7144 }%

```

`\parenifnotempty` Do a space followed by the argument if the argument doesn’t expand to empty or `\relax`. If argument isn’t empty (or `\relax`), apply the macro to it given in the second argument.

```

7145 \DeclareRobustCommand*{\glsdoparenifnotempty}[2]{%
7146     \protected@edef\gls@tmp{#1}%
7147     \ifdefempty\gls@tmp
7148     {}%
7149     {%
7150         \ifx\gls@tmp\@gls@default@value
7151         \else
7152             \space (#2{#1})%
7153         \fi
7154     }%
7155 }

```

`\nymDisplayStyle` Sets the acronym display style for given glossary where neither footnote nor description is required, but smallcaps or smaller specified.

```

7156 \newcommand*{\SetSmallAcronymDisplayStyle}[1]{%
7157     \defglsenentryfmt[#1]{%

7158     \ifdefempty\glscustomtext
7159     {%

```

Move the inserted text outside of `\acronymfont`

```

7160         \let\gls@org@insert\glsinsert
7161         \let\glsinsert\@empty
7162         \ifglssused{\glslabel}%
7163         {%
7164             \acronymfont{\glsgenentryfmt}\gls@org@insert
7165         }%
7166         {%
7167             \glsgenentryfmt
7168             \ifglshassymbol{\glslabel}%
7169             {%
7170                 \glsifplural
7171                 {%

```

```

7172         \def\@glo@symbol{\glentrysymbolplural{\glslabel}}%
7173     }%
7174     {%
7175         \def\@glo@symbol{\glentrysymbol{\glslabel}}%
7176     }%
7177     \space
7178     (\glscapscase
7179     {\firstacronymfont{\@glo@symbol}}%
7180     {\firstacronymfont{\@glo@symbol}}%
7181     {\firstacronymfont{\mfirstucMakeUppercase{\@glo@symbol}}})%
7182 }%
7183 {}%
7184 }%
7185 }%
7186 {\glscustomtext\glsinsert}%
7187 }%
7188 }

```

11NewAcronymDef

```

7189 \newcommand*{\SmallNewAcronymDef}{%
7190     \edef\@do@newglossaryentry{%
7191         \noexpand\newglossaryentry{\the\glslabeltok}%
7192         {%
7193             type=\acronymtype,%
7194             name={\noexpand\acronymfont{\the\glsshorttok}},%
7195             sort={\the\glsshorttok},%
7196             text={\the\glsshorttok},%
7197             plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7198             first={\the\glslongtok},%
7199             firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7200             short={\the\glsshorttok},%
7201             shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7202             long={\the\glslongtok},%
7203             longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7204             description={\noexpand\@glo@first},%
7205             descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7206             symbol={\the\glsshorttok},%
7207             symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7208             \the\glskeylisttok
7209         }%
7210     }%
7211     \let\@org@gls@assign@firstpl\gls@assign@firstpl
7212     \let\@org@gls@assign@plural\gls@assign@plural
7213     \let\@org@gls@assign@descplural\gls@assign@descplural

```

```

7214 \let\org@gl@s@assign@symbolplural\gl@s@assign@symbolplural
7215 \def\gl@s@assign@firstpl##1##2{%
7216   \@@gl@s@expand@field{##1}{firstpl}{##2}%
7217 }%
7218 \def\gl@s@assign@plural##1##2{%
7219   \@@gl@s@expand@field{##1}{plural}{##2}%
7220 }%
7221 \def\gl@s@assign@descplural##1##2{%
7222   \@@gl@s@expand@field{##1}{descplural}{##2}%
7223 }%
7224 \def\gl@s@assign@symbolplural##1##2{%
7225   \@@gl@s@expand@field{##1}{symbolplural}{##2}%
7226 }%
7227 \do@newglossaryentry
7228 \let\gl@s@assign@firstpl\org@gl@s@assign@firstpl
7229 \let\gl@s@assign@plural\org@gl@s@assign@plural
7230 \let\gl@s@assign@descplural\org@gl@s@assign@descplural
7231 \let\gl@s@assign@symbolplural\org@gl@s@assign@symbolplural
7232 }

```

allAcronymStyle Neither footnote nor description required, but smallcaps or smaller specified. Use the symbol key to store the short form and first to store the long form.

```

7233 \newcommand*{\SetSmallAcronymStyle}{%
7234   \renewcommand{\newacronym}[4][]{%
7235     \ifx\org@gl@s@acronymlists\@empty
7236       \def\@glo@type{\acronymtype}%
7237       \setkeys{glossentry}{##1}%
7238       \DeclareAcronymList{\@glo@type}%
7239       \SetSmallAcronymDisplayStyle{\@glo@type}%
7240     \fi
7241     \gl@keylisttok{##1}%
7242     \gl@labeltok{##2}%
7243     \gl@shorttok{##3}%
7244     \gl@longtok{##4}%
7245     \newacronymhook
7246     \SmallNewAcronymDef
7247   }%

```

Change the display since first only contains long form.

```

7248 \@for\@gl@s@type:=\org@gl@s@acronymlists\do{%
7249   \SetSmallAcronymDisplayStyle{\@gl@s@type}%
7250 }%

```

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.

```

7251 \ifgl@acrsmallcaps
7252   \renewcommand*{\acronymfont}[1]{\textsc{##1}}
7253   \renewcommand*{\acrpluralsuffix}{\gl@supacrpluralsuffix}%
7254 \else
7255   \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}

```

```

7256 \fi
      check for option clash
7257 \ifglsacrdua
7258   \ifglsacrsmallcaps
7259     \PackageError{glossaries}{Option clash: ‘smallcaps’ and ‘dua’
7260       can’t both be set}{}%
7261   \else
7262     \PackageError{glossaries}{Option clash: ‘smaller’ and ‘dua’
7263       can’t both be set}{}%
7264   \fi
7265 \fi
7266 }%

```

DUADisplayStyle Sets the acronym display style for given glossary with dua setting.

```

7267 \newcommand*{\SetDUADisplayStyle}[1]{%
7268   \defglsentryfmt[#1]{\glsentryfmt}%
7269 }

```

UANewAcronymDef

```

7270 \newcommand*{\DUANewAcronymDef}{%
7271   \edef\@do@newglossaryentry{%
7272     \noexpand\newglossaryentry{\the\glslabeltok}%
7273     {%
7274       type=\acronymtype,%
7275       name={\the\glsshorttok},%
7276       text={\the\glslongtok},%
7277       first={\the\glslongtok},%
7278       plural={\noexpand\expandonce\noexpand\@glo@longpl},%
7279       firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7280       short={\the\glsshorttok},%
7281       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7282       long={\the\glslongtok},%
7283       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7284       description={\the\glslongtok},%
7285       descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7286       symbol={\the\glsshorttok},%
7287       symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7288       \the\glskeylisttok
7289     }%
7290   }%
7291   \let\@org@gls@assign@firstpl\gls@assign@firstpl
7292   \let\@org@gls@assign@plural\gls@assign@plural
7293   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7294   \let\@org@gls@assign@descplural\gls@assign@descplural
7295   \def\gls@assign@firstpl##1##2{%
7296     \@@gls@expand@field{##1}{firstpl}{##2}%
7297   }%
7298   \def\gls@assign@plural##1##2{%
7299     \@@gls@expand@field{##1}{plural}{##2}%

```

```

7300 }%
7301 \def\gls@assign@symbolplural##1##2{%
7302   \@gls@expand@field{##1}{symbolplural}{##2}%
7303 }%
7304 \def\gls@assign@descplural##1##2{%
7305   \@gls@expand@field{##1}{descplural}{##2}%
7306 }%
7307 \do@newglossaryentry
7308 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7309 \let\gls@assign@plural\@org@gls@assign@plural
7310 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7311 \let\gls@assign@descplural\@org@gls@assign@descplural
7312 }

```

\SetDUASStyle Always expand acronyms.

```

7313 \newcommand*{\SetDUASStyle}{%
7314   \renewcommand{\newacronym}[4][]{%
7315     \ifx\@glsacronymlists\@empty
7316       \def\@glo@type{\acronymtype}%
7317       \setkeys{glossentry}{##1}%
7318       \DeclareAcronymList{\@glo@type}%
7319       \SetDUADisplayStyle{\@glo@type}%
7320     \fi
7321     \glskeylisttok{##1}%
7322     \glslabeltok{##2}%
7323     \glsshorttok{##3}%
7324     \glslongtok{##4}%
7325     \newacronymhook
7326     \DUANewAcronymDef
7327   }%
7328   \SetDUASStyle
7329   \SetDUADisplayStyle{\@gls@type}%
7330 }%
7331 }

```

SetAcronymStyle

```

7332 \newcommand*{\SetAcronymStyle}{%
7333   \SetDefaultAcronymStyle
7334   \ifglsacrdescription
7335     \ifglsacrfootnote
7336       \SetDescriptionFootnoteAcronymStyle
7337     \else
7338       \ifglsacrdua
7339         \SetDescriptionDUAAcronymStyle
7340       \else
7341         \SetDescriptionAcronymStyle
7342       \fi
7343     \fi

```

```

7344 \else
7345   \ifglsacrfootnote
7346     \SetFootnoteAcronymStyle
7347   \else
7348     \ifthenelse{\boolean{glsacrsmalldcaps}}\OR
7349       \boolean{glsacrsmalldl}}{
7350       {%
7351         \SetSmallAcronymStyle
7352       }%
7353     {%
7354       \ifglsacrdua
7355         \SetDUASStyle
7356       \fi
7357     }%
7358   \fi
7359 \fi
7360 }

```

Set the acronym style according to the package options

```

7361 \SetAcronymStyle

```

Allow user to define their own custom acronyms. (For compatibility with versions before v3.0, the short form is stored in the user1 key, the plural short form is stored in the user2 key, the long form is stored in the user3 key and the plural long form is stored in the user4 key.) Defaults to displaying only the acronym with the long form as the description.

`\setacronymdisplaystyle` Sets the acronym display style.

```

7362 \newcommand*{\SetCustomDisplayStyle}[1]{%
7363   \defglsentryfmt[#1]{\glsentryfmt}%
7364 }

```

`\setacronymfields`

```

7365 \newcommand*{\CustomAcronymFields}{%
7366   name={\the\glsshorttok},%
7367   description={\the\glslongtok},%
7368   first={\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
7369   firstplural={\acrfullformat
7370     {\noexpand\glsentrylongpl{\the\glslabeltok}}}%
7371     {\noexpand\glsentryshortpl{\the\glslabeltok}}},%

7372   text={\the\glsshorttok},%
7373   plural={\the\glsshorttok\noexpand\acrpluralsuffix}%
7374 }

```

`\setnewacronymdef`

```

7375 \newcommand*{\CustomNewAcronymDef}{%
7376   \protected@edef\do@newglossaryentry{%
7377     \noexpand\newglossaryentry{\the\glslabeltok}%
7378     {%

```

```

7379     type=\acronymtype,%
7380     short={\the\glsshorttok},%
7381     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7382     long={\the\glslongtok},%
7383     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7384     user1={\the\glsshorttok},%
7385     user2={\the\glsshorttok\noexpand\acrpluralsuffix},%
7386     user3={\the\glslongtok},%
7387     user4={\the\glslongtok\noexpand\acrpluralsuffix},%
7388     \CustomAcronymFields,%
7389     \the\glskeylisttok
7390   }%
7391 }%
7392 \@do@newglossaryentry
7393 }

```

\SetCustomStyle

```

7394 \newcommand*{\SetCustomStyle}{%
7395   \renewcommand{\newacronym}[4][]{%
7396     \ifx\@glsacronymlists\@empty
7397       \def\@gls@type{\acronymtype}%
7398       \setkeys{glossentry}{##1}%
7399       \DeclareAcronymList{\@gls@type}%
7400       \SetCustomDisplayStyle{\@gls@type}%
7401     \fi
7402     \glskeylisttok{##1}%
7403     \glslabeltok{##2}%
7404     \glsshorttok{##3}%
7405     \glslongtok{##4}%
7406     \newacronymhook
7407     \CustomNewAcronymDef
7408   }%
7409   Set the display
7410   \@for\@gls@type:=\@glsacronymlists\do{%
7411     \SetCustomDisplayStyle{\@gls@type}%
7412   }%
7413 }

```

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.

```

7413 \RequirePackage{glossary-hypernav}

```

The styles that use list-like environments. These are not loaded if the nolist option is used:

```

7414 \@gls@loadlist

```


The styles that use the longtable environment. These are not loaded if the nolong package option is used.

```
7415 \@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.

```
7416 \@gls@loadsuper
```

The tree-like styles. These are not loaded if the notree package option is used.

```
7417 \@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.

```
7418 \ifx\@glossary@default@style\relax
```

```
7419 \else
```

```
7420   \setglossarystyle{\@glossary@default@style}
```

```
7421 \fi
```

1.20 Debugging Commands

```
\showgloparent \showgloparent{\label{}}
```

```
7422 \newcommand*{\showgloparent}[1]{%
```

```
7423   \expandafter\show\csname glo@\glsdetoklabel{#1}@parent\endcsname
```

```
7424 }
```

```
\showglolevel \showglolevel{\label{}}
```

```
7425 \newcommand*{\showglolevel}[1]{%
```

```
7426   \expandafter\show\csname glo@\glsdetoklabel{#1}@level\endcsname
```

```
7427 }
```

```
\showglotext \showglotext{\label{}}
```

```
7428 \newcommand*{\showglotext}[1]{%
```

```
7429   \expandafter\show\csname glo@\glsdetoklabel{#1}@text\endcsname
```

```
7430 }
```

```
\showgloplural \showgloplural{\label{}}
```

```

7431 \newcommand*{\showgloplural}[1]{%
7432   \expandafter\show\csname glo@\glsdetoklabel{#1}@plural\endcsname
7433 }

```

\showglofirst \showglofirst{<label>}

```

7434 \newcommand*{\showglofirst}[1]{%
7435   \expandafter\show\csname glo@\glsdetoklabel{#1}@first\endcsname
7436 }

```

\showglofirstpl \showglofirstpl{<label>}

```

7437 \newcommand*{\showglofirstpl}[1]{%
7438   \expandafter\show\csname glo@\glsdetoklabel{#1}@firstpl\endcsname
7439 }

```

\showgloftype \showgloftype{<label>}

```

7440 \newcommand*{\showgloftype}[1]{%
7441   \expandafter\show\csname glo@\glsdetoklabel{#1}@type\endcsname
7442 }

```

\showglocounter \showglocounter{<label>}

```

7443 \newcommand*{\showglocounter}[1]{%
7444   \expandafter\show\csname glo@\glsdetoklabel{#1}@counter\endcsname
7445 }

```

\showglouser \showglouser{<label>}

```

7446 \newcommand*{\showglouser}[1]{%
7447   \expandafter\show\csname glo@\glsdetoklabel{#1}@user\endcsname
7448 }

```

\showglouserii \showglouserii{<label>}

```

7449 \newcommand*{\showglouserii}[1]{%
7450   \expandafter\show\csname glo@glstdetoklabel{#1}@userii\endcsname
7451 }

```

\showglouseriii \showglouseriii{<label>}

```

7452 \newcommand*{\showglouseriii}[1]{%
7453   \expandafter\show\csname glo@glstdetoklabel{#1}@useriii\endcsname
7454 }

```

\showglouseriv \showglouseriv{<label>}

```

7455 \newcommand*{\showglouseriv}[1]{%
7456   \expandafter\show\csname glo@glstdetoklabel{#1}@useriv\endcsname
7457 }

```

\showglouserv \showglouserv{<label>}

```

7458 \newcommand*{\showglouserv}[1]{%
7459   \expandafter\show\csname glo@glstdetoklabel{#1}@userv\endcsname
7460 }

```

\showglouservi \showglouservi{<label>}

```

7461 \newcommand*{\showglouservi}[1]{%
7462   \expandafter\show\csname glo@glstdetoklabel{#1}@uservi\endcsname
7463 }

```

\showgloname \showgloname{<label>}

```

7464 \newcommand*{\showgloname}[1]{%
7465   \expandafter\show\csname glo@glstdetoklabel{#1}@name\endcsname
7466 }

```

\showglodesc \showglodesc{<label>}

```

7467 \newcommand*{\showglodesc}[1]{%
7468   \expandafter\show\csname glo@\glsdetoklabel{#1}@desc\endcsname
7469 }

```

showglodescplural `\showglodescplural{<label>}`

```

7470 \newcommand*{\showglodescplural}[1]{%
7471   \expandafter\show\csname glo@\glsdetoklabel{#1}@descplural\endcsname
7472 }

```

\showglosort `\showglosort{<label>}`

```

7473 \newcommand*{\showglosort}[1]{%
7474   \expandafter\show\csname glo@\glsdetoklabel{#1}@sort\endcsname
7475 }

```

\showglosymbol `\showglosymbol{<label>}`

```

7476 \newcommand*{\showglosymbol}[1]{%
7477   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbol\endcsname
7478 }

```

showglosymbolplural `\showglosymbolplural{<label>}`

```

7479 \newcommand*{\showglosymbolplural}[1]{%
7480   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolplural\endcsname
7481 }

```

\showgloshort `\showgloshort{<label>}`

```

7482 \newcommand*{\showgloshort}[1]{%
7483   \expandafter\show\csname glo@\glsdetoklabel{#1}@short\endcsname
7484 }

```

\showglolong `\showglolong{<label>}`

```

7485 \newcommand*{\showglolong}[1]{%
7486   \expandafter\show\csname glo@\glsdetoklabel{#1}@long\endcsname
7487 }

```

\showgloindex \showgloindex{<label>}

```

7488 \newcommand*{\showgloindex}[1]{%
7489   \expandafter\show\csname glo@\glsdetoklabel{#1}@index\endcsname
7490 }

```

\showgloflag \showgloflag{<label>}

```

7491 \newcommand*{\showgloflag}[1]{%
7492   \expandafter\show\csname ifglo@\glsdetoklabel{#1}@flag\endcsname
7493 }

```

\showgloloclist \showgloloclist{<label>}

```

7494 \newcommand*{\showgloloclist}[1]{%
7495   \expandafter\show\csname glo@\glsdetoklabel{#1}@loclist\endcsname
7496 }

```

\showglofield \showglofield{<label>}{<field>}

```

7497 \newcommand*{\showglofield}[2]{%
7498   \csshow{glo@\glsdetoklabel{#1}@#2}%
7499 }

```

showacronymlists \showacronymlists

Show list of glossaries that have been flagged as a list of acronyms.

```

7500 \newcommand*{\showacronymlists}{%
7501   \show\@glsacronymlists
7502 }

```

\showglossaries \showglossaries

Show list of defined glossaries.

```

7503 \newcommand*{\showglossaries}{%

```

```
7504 \show\@glo@types
7505 }
```

`\showglossaryin` `\showglossaryin{<glossary-label>}`

Show the ‘in’ extension for the given glossary.

```
7506 \newcommand*{\showglossaryin}[1]{%
7507 \expandafter\show\csname @glotype@#1@in\endcsname
7508 }
```

`\showglossaryout` `\showglossaryout{<glossary-label>}`

Show the ‘out’ extension for the given glossary.

```
7509 \newcommand*{\showglossaryout}[1]{%
7510 \expandafter\show\csname @glotype@#1@out\endcsname
7511 }
```

`showglossarytitle` `\showglossarytitle{<glossary-label>}`

Show the title for the given glossary.

```
7512 \newcommand*{\showglossarytitle}[1]{%
7513 \expandafter\show\csname @glotype@#1@title\endcsname
7514 }
```

`wglossarycounter` `\showglossarycounter{<glossary-label>}`

Show the counter for the given glossary.

```
7515 \newcommand*{\showglossarycounter}[1]{%
7516 \expandafter\show\csname @glotype@#1@counter\endcsname
7517 }
```

`wglossaryentries` `\showglossaryentries{<glossary-label>}`

Show the list of entry labels for the given glossary.

```
7518 \newcommand*{\showglossaryentries}[1]{%
7519 \expandafter\show\csname glolist@#1\endcsname
7520 }
```

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.

```
7521 \csname ifglscpatible-2.07\endcsname
7522   \RequirePackage{glossaries-compatible-207}
7523 \fi
```

2 Prefix Support (glossaries-prefix Code)

This package provides a means of adding prefixes to your glossary entries. For example, you may want to use “`\gls{<label>}`” on first use but use “`\an\gls{<label>}`” on subsequent use.

```
7524 \NeedsTeXFormat{LaTeX2e}
```

```
7525 \ProvidesPackage{glossaries-prefix}[2016/04/19 v4.22 (NLCT)]
```

Pass all options to glossaries:

```
7526 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
7527 \ProcessOptions
```

Load glossaries:

```
7528 \RequirePackage{glossaries}
```

Add the new keys:

```
7529 \define@key{glossentry}{prefixfirst}{\def\@glo@entryprefixfirst{#1}}%
```

```
7530 \define@key{glossentry}{prefixfirstplural}{\def\@glo@entryprefixfirstplural{#1}}%
```

```
7531 \define@key{glossentry}{prefix}{\def\@glo@entryprefix{#1}}%
```

```
7532 \define@key{glossentry}{prefixplural}{\def\@glo@entryprefixplural{#1}}%
```

Add them to `\@gls@keymap`:

```
7533 \appto\@gls@keymap{,%
```

```
7534   {prefixfirst}{prefixfirst},%
```

```
7535   {prefixfirstplural}{prefixfirstplural},%
```

```
7536   {prefix}{prefix},%
```

```
7537   {prefixplural}{prefixplural}}%
```

```
7538 }
```

Set the default values:

```
7539 \appto\@newglossaryentryprehook{%
```

```
7540   \def\@glo@entryprefix{}}%
```

```
7541   \def\@glo@entryprefixplural{}}%
```

```
7542   \let\@glo@entryprefixfirst\@gls@default@value
```

```
7543   \let\@glo@entryprefixfirstplural\@gls@default@value
```

```
7544 }
```

Set the assignment code:

```
7545 \appto\@newglossaryentryposthook{%
```

```
7546   \gls@assign@field{ }\@glo@label{prefix}{\@glo@entryprefix}}%
```

```
7547   \gls@assign@field{ }\@glo@label{prefixplural}{\@glo@entryprefixplural}}%
```

If `prefixfirst` has not been supplied, make it the same as `prefix`.

```
7548 \expandafter\gls@assign@field\expandafter
```

```
7549   {\csname glo@\@glo@label @prefix\endcsname}{\@glo@label}{prefixfirst}}%
```

```
7550   {\@glo@entryprefixfirst}}%
```


If prefixfirstplural has not been supplied, make it the same as prefixplural.

```

7551 \expandafter\gls@assign@field\expandafter
7552   {\csname glo@\@glo@label @prefixplural\endcsname}{\@glo@label}%
7553   {prefixfirstplural}{\@glo@entryprefixfirstplural}%
7554 }

```

Define commands to access these fields:

entryprefixfirst

```

7555 \newcommand*\glsentryprefixfirst[1]{\csuse{glo@#1@prefixfirst}}

```

entryfirstplural

```

7556 \newcommand*\glsentryfirstplural[1]{\csuse{glo@#1@prefixfirstplural}}

```

\glsentryprefix

```

7557 \newcommand*\glsentryprefix[1]{\csuse{glo@#1@prefix}}

```

entryprefixplural

```

7558 \newcommand*\glsentryprefixplural[1]{\csuse{glo@#1@prefixplural}}

```

Now for the initial upper case variants:

entryprefixfirst

```

7559 \newrobustcmd*\Glsentryprefixfirst[1]{%
7560   \protected@edef\@glo@text{\csname glo@#1@prefixfirst\endcsname}%
7561   \xmakefirstuc\@glo@text
7562 }

```

entryfirstplural

```

7563 \newrobustcmd*\Glsentryfirstplural[1]{%
7564   \protected@edef\@glo@text{\csname glo@#1@prefixfirstplural\endcsname}%
7565   \xmakefirstuc\@glo@text
7566 }

```

\Glsentryprefix

```

7567 \newrobustcmd*\Glsentryprefix[1]{%
7568   \protected@edef\@glo@text{\csname glo@#1@prefix\endcsname}%
7569   \xmakefirstuc\@glo@text
7570 }

```

entryprefixplural

```

7571 \newrobustcmd*\Glsentryprefixplural[1]{%
7572   \protected@edef\@glo@text{\csname glo@#1@prefixplural\endcsname}%
7573   \xmakefirstuc\@glo@text
7574 }

```

Define commands to determine if the prefix keys have been set:

\ifglshasprefix

```
7575 \newcommand*{\ifglshasprefix}[3]{%
7576   \ifcempty{glo@#1@prefix}%
7577   {#3}%
7578   {#2}%
7579 }
```

hasprefixplural

```
7580 \newcommand*{\ifglshasprefixplural}[3]{%
7581   \ifcempty{glo@#1@prefixplural}%
7582   {#3}%
7583   {#2}%
7584 }
```

shasprefixfirst

```
7585 \newcommand*{\ifglshasprefixfirst}[3]{%
7586   \ifcempty{glo@#1@prefixfirst}%
7587   {#3}%
7588   {#2}%
7589 }
```

efixfirstplural

```
7590 \newcommand*{\ifglshasprefixfirstplural}[3]{%
7591   \ifcempty{glo@#1@prefixfirstplural}%
7592   {#3}%
7593   {#2}%
7594 }
```

Define commands that insert the prefix before commands like \gls:

\pgls

```
7595 \newrobustcmd{\pgls}{\@gls@hyp@opt\@pgls}
```

\@pgls Unstarred version.

```
7596 \newcommand*{\@pgls}[2][ ]{%
7597   \new@ifnextchar[%
7598   {\@pgls@{#1}{#2}}%
7599   {\@pgls@{#1}{#2}[ ]}%
7600 }
```

\@pgls@ Read in the final optional argument:

```
7601 \def\@pgls@#1#2[#3]{%
7602   \glsdoifexists{#2}%
7603   {%
7604     \ifglsused{#2}%
7605     {%
7606       \glsentryprefix{#2}%
7607     }%

```

```

7608     {%
7609     \glsentryprefixfirst{#2}%
7610     }%
7611     \@gls@{#1}{#2}[#3]%
7612     }%
7613 }

```

Similarly for the plural version:

```

\pglsp1
7614 \newrobustcmd{\pglsp1}{\@gls@hyp@opt\@pglsp1}

```

\@pglsp1 Unstarred version.

```

7615 \newcommand*{\@pglsp1}[2][{}]{%
7616   \new@ifnextchar[%
7617   {\@pglsp1@{#1}{#2}}%
7618   {\@pglsp1@{#1}{#2}[]}%
7619 }

```

\@pglsp1@ Read in the final optional argument:

```

7620 \def\@pglsp1@#1#2[#3]{%
7621   \glsdoifexists{#2}%
7622   {%
7623     \ifglsused{#2}%
7624     {%
7625       \glsentryprefixplural{#2}%
7626     }%
7627     {%
7628       \glsentryprefixfirstplural{#2}%
7629     }%
7630     \@glspl@{#1}{#2}[#3]%
7631   }%
7632 }

```

Now for the first letter upper case versions:

```

\Pgls
7633 \newrobustcmd{\Pgls}{\@gls@hyp@opt\@Pgls}

```

\@Pgls Unstarred version.

```

7634 \newcommand*{\@Pgls}[2][{}]{%
7635   \new@ifnextchar[%
7636   {\@Pgls@{#1}{#2}}%
7637   {\@Pgls@{#1}{#2}[]}%
7638 }

```

\@Pgls@ Read in the final optional argument:

```

7639 \def\@Pgls@#1#2[#3]{%

```

```

7640 \glsdoifexists{#2}%
7641 {%
7642   \ifglsused{#2}%
7643   {%
7644     \ifglshasprefix{#2}%
7645     {%
7646       \Glsentryprefix{#2}%
7647       \@gls@{#1}{#2}[#3]%
7648     }%
7649     {\@Gls@{#1}{#2}[#3]}%
7650   }%
7651   {%
7652     \ifglshasprefixfirst{#2}%
7653     {%
7654       \Glsentryprefixfirst{#2}%
7655       \@gls@{#1}{#2}[#3]%
7656     }%
7657     {\@Gls@{#1}{#2}[#3]}%
7658   }%
7659 }%
7660 }

```

Similarly for the plural version:

```

\Pglspl
7661 \newrobustcmd{\Pglspl}{\@gls@hyp@opt\@Pglspl}

```

\@Pglspl Unstarred version.

```

7662 \newcommand*{\@Pglspl}[2] [] {%
7663   \new@ifnextchar[%
7664   {\@Pglspl@{#1}{#2}}%
7665   {\@Pglspl@{#1}{#2} []}%
7666 }

```

\@Pglspl@ Read in the final optional argument:

```

7667 \def\@Pglspl@#1#2[#3] {%
7668   \glsdoifexists{#2}%
7669   {%
7670     \ifglsused{#2}%
7671     {%
7672       \ifglshasprefixplural{#2}%
7673       {%
7674         \Glsentryprefixplural{#2}%
7675         \@glspl@{#1}{#2}[#3]%
7676       }%
7677       {\@Glspl@{#1}{#2}[#3]}%
7678     }%
7679     {%
7680       \ifglshasprefixfirstplural{#2}%

```

```

7681      {%
7682      \Glsentryprefixfirstplural{#2}%
7683      \@glsp1@{#1}{#2}[#3]%
7684      }%
7685      {\@Glspl@{#1}{#2}[#3]}%
7686      }%
7687  }%
7688 }

```

Finally the all upper case versions:

\PGLS

```

7689 \newrobustcmd{\PGLS}{\@gls@hyp@opt\PGLS}

```

\@PGLS Unstarred version.

```

7690 \newcommand*{\@PGLS}[2][{}]{%
7691   \new@ifnextchar[%
7692   {\@PGLS@{#1}{#2}}%
7693   {\@PGLS@{#1}{#2}[]}%
7694 }

```

\@PGLS@ Read in the final optional argument:

```

7695 \def\@PGLS@#1#2[#3]{%
7696   \glsdoifexists{#2}%
7697   {%
7698     \ifglsused{#2}%
7699     {%
7700       \mfirstucMakeUppercase{\glsentryprefix{#2}}%
7701     }%
7702     {%
7703       \mfirstucMakeUppercase{\glsentryprefixfirst{#2}}%
7704     }%
7705     \@GLS@{#1}{#2}[#3]%
7706   }%
7707 }

```

Plural version:

\PGLSp1

```

7708 \newrobustcmd{\PGLSp1}{\@gls@hyp@opt\PGLSp1}

```

\@PGLSp1 Unstarred version.

```

7709 \newcommand*{\@PGLSp1}[2][{}]{%
7710   \new@ifnextchar[%
7711   {\@PGLSp1@{#1}{#2}}%
7712   {\@PGLSp1@{#1}{#2}[]}%
7713 }

```

\@PGLSp1@ Read in the final optional argument:

```
7714 \def\@PGLSp1@#1#2[#3]{%
7715   \glsdoifexists{#2}%
7716   {%
7717     \ifglsused{#2}%
7718     {%
7719       \mfirstucMakeUppercase{\glsentryprefixplural{#2}}%
7720     }%
7721     {%
7722       \mfirstucMakeUppercase{\glsentryprefixfirstplural{#2}}%
7723     }%
7724     \@GLSp1@{#1}{#2}[#3]%
7725   }%
7726 }
```

3 Glossary Styles

3.1 Glossary hyper-navigation definitions (glossary-hypernav package)

Package Definition:

```
7727 \ProvidesPackage{glossary-hypernav}[2016/04/19 v4.22 (NLCT)]
```

The commands defined in this package are provided to help navigate around the groups within a glossary (see [section 1.16](#).) `\printglossary` (and `\printglossaries`) set `\@glo@type` to the label of the current glossary. This is used to create a unique hypertarget in the event of multiple glossaries.

```
\glsnavhyperlink[<type>]{<label>}{<text>}
```

This command makes `<text>` a hyperlink to the glossary group whose label is given by `<label>` for the glossary given by `<type>`.

`glsnavhyperlink`

```
7728 \newcommand*{\glsnavhyperlink}[3][\@glo@type]{%
7729   \edef\gls@grplabel{#2}\protected@edef\gls@grptitle{#3}%
7730   \@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`

```
7731 \newcommand*{\glsnavhypertarget}[3][\@glo@type]{%
  Add this group to the aux file for re-run check.
7732   \protected@write\@auxout{}{\string\@gls@hypergroup{#1}{#2}}%
  Add the target.
7733   \@glstarget{glsn:#1@#2}{#3}%
  Check list of know groups to determine if a re-run is required.
7734   \expandafter\let
7735     \expandafter\@gls@list\csname @gls@hypergroup@#1\endcsname
  Iterate through list and terminate loop if this group is found.
7736   \@for\@gls@elem:=\@gls@list\do{%
7737     \ifthenelse{equal{\@gls@elem}{#2}}{\@endfortrue}{}%
}
```

Check if list terminated prematurely.

```
7738 \if@endfor
7739 \else
```

This group was not included in the list, so issue a warning.

```
7740 \GlossariesWarningNoLine{Navigation panel
7741     for glossary type ‘#1’^^Jmissing group ‘#2’}%
7742 \gdef\gls@hypergrouprerun{%
7743     \GlossariesWarningNoLine{Navigation panel
7744     has changed. Rerun LaTeX}}%
7745 \fi
7746 }
```

hypergrouprerun Give a warning at the end if re-run required

```
7747 \let\gls@hypergrouprerun\relax
7748 \AtEndDocument{\gls@hypergrouprerun}
```

@gls@hypergroup This adds to (or creates) the command \@gls@hypergroup_{list}@*<glossary type>* which lists all groups for a given glossary, so that the navigation bar only contains those groups that are present. However it requires at least 2 runs to ensure the information is up-to-date.

```
7749 \newcommand*{\@gls@hypergroup}[2]{%
7750 \ifundefined{\@gls@hypergrouplist@#1}{%
7751     \expandafter\xdef\csname @gls@hypergrouplist@#1\endcsname{#2}%
7752 }{%
7753     \expandafter\let\expandafter\@gls@tmp
7754         \csname @gls@hypergrouplist@#1\endcsname
7755     \expandafter\xdef\csname @gls@hypergrouplist@#1\endcsname{%
7756         \@gls@tmp,#2}%
7757 }%
7758 }
```

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

```
7759 \newcommand*{\glsnavigation}{%
7760 \def\@gls@between{}%
7761 \ifcsundef{\@gls@hypergrouplist@\@glo@type}%
7762 {%
7763     \def\@gls@list{}%
7764 }%
7765 {%
7766     \expandafter\let\expandafter\@gls@list
7767         \csname @gls@hypergrouplist@\@glo@type\endcsname
7768 }%
7769 \@for\@gls@tmp:=\@gls@list\do{%
```



```

7770 \gls@between
7771 \gls@getgrouptitle{\gls@tmp}{\gls@grptitle}%
7772 \glsnavhyperlink{\gls@tmp}{\gls@grptitle}%
7773 \let\gls@between\glshypernavsep
7774 }%
7775 }

```

`\glshypernavsep` Separator for the hyper navigation bar.

```

7776 \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`

```

7777 \newcommand*{\glssymbolnav}{%
7778 \glsnavhyperlink{glssymbols}{\glsgetgrouptitle{glssymbols}}%
7779 \glshypernavsep
7780 \glsnavhyperlink{glsnumbers}{\glsgetgrouptitle{glsnumbers}}%
7781 \glshypernavsep
7782 }

```

3.2 In-line Style (glossary-inline.sty)

This defines an in-line style where the entries are comma-separated with just the name and description displayed.

```

7783 \ProvidesPackage{glossary-inline}[2016/04/19 v4.22 (NLCT)]

```

`inline` Define the inline style.

```

7784 \newglossarystyle{inline}{%
    Start of glossary sets up first empty separator between entries. (This is then changed by
    \glossentry)
7785 \renewenvironment{theglossary}%
7786 {%
7787 \def\gls@inlinesep{}%
7788 \def\gls@inlinesubsep{}%
7789 \def\gls@inlinepostchild{}%
7790 }%
7791 {\glspostinline}%

    No header:
7792 \renewcommand*{\glossaryheader}{}%

    No group headings (if heading is required, add \glsinlinedopostchild to start definition
    in case heading follows a child entry):
7793 \renewcommand*{\glsgroupheading}[1]{}%

```

Just display separator followed by name and description:

```

7794 \renewcommand{\glossentry}[2]{%
7795   \glsinlinedopostchild
7796   \gls@inlinesep
7797   \glsentryitem{##1}%
7798   \glsinlinenameformat{##1}{%
7799     \glossentryname{##1}%
7800   }%
7801   \ifglshasdescsuppressed{##1}%
7802   {%
7803     \glsinlineemptydescformat
7804     {%
7805       \glossentrysymbol{##1}%
7806     }%
7807     {%
7808       ##2%
7809     }%
7810   }%
7811   {%
7812     \ifglshasdesc{##1}%
7813     {\glsinlinedescformat{\glossentrydesc{##1}}{\glossentrysymbol{##1}}{##2}}%
7814     {\glsinlineemptydescformat{\glossentrysymbol{##1}}{##2}}%
7815   }%
7816   \ifglshaschildren{##1}%
7817   {%
7818     \glsresetsubentrycounter
7819     \glsinlineparentchildseparator
7820     \def\gls@inlinesubsep{%
7821       \def\gls@inlinepostchild{\glsinlinepostchild}%
7822     }%
7823     {}%
7824     \def\gls@inlinesep{\glsinlineseparator}%
7825   }%

```

Sub-entries display description:

```

7826 \renewcommand{\subglossentry}[3]{%
7827   \gls@inlinesubsep%
7828   \glsinlinesubnameformat{##2}{%
7829     \glossentryname{##2}%
7830   }%
7831   \glsinlinesubdescformat{\glossentrydesc{##2}}{\glossentrysymbol{##2}}{##3}%
7832   \def\gls@inlinesubsep{\glsinlinesubseparator}%
7833 }%

```

Nothing special between groups:

```

7834 \renewcommand*{\glsgroupskip}{}%
7835 }

```

linedopostchild

```

7836 \newcommand*{\glsinlinedopostchild}{%

```

```

7837 \gls@inlinepostchild
7838 \def\gls@inlinepostchild{}%
7839 }

```

`inlineseparator` Separator to use between entries.

```
7840 \newcommand*\glsinlineseparator}{;\space}
```

`inlinesubseparator` Separator to use between sub-entries.

```
7841 \newcommand*\glsinlinesubseparator}{,\space}
```

`parentchildseparator` Separator to use between parent and children.

```
7842 \newcommand*\glsinlineparentchildseparator}{:\space}
```

`inlinepostchild` Hook to use between child and next entry

```
7843 \newcommand*\glsinlinepostchild}{}
```

`\glspostinline` Terminator for inline glossary.

```
7844 \newcommand*\glspostinline}{\glspostdescription\space}
```

`inlinenameformat` Formats the name of the entry (first argument label, second argument name):

```
7845 \newcommand*\glsinlinenameformat}[2]{\glstarget{#1}{#2}}
```

`inlinedescformat` Formats the entry's description, symbol and location list:

```
7846 \newcommand*\glsinlinedescformat}[3]{\space#1}
```

`emptydescformat` Formats the entry's symbol and location list when the description is empty:

```
7847 \newcommand*\glsinlineemptydescformat}[2]{}
```

`inlinesubnameformat` Formats the name of the subentry (first argument label, second argument name):

```
7848 \newcommand*\glsinlinesubnameformat}[2]{\glstarget{#1}{}}
```

`inlinesubdescformat` Formats the subentry's description, symbol and location list:

```
7849 \newcommand*\glsinlinesubdescformat}[3]{#1}
```

3.3 List Style (glossary-list.sty)

The style file defines glossary styles that use the description environment. Note that since the entry name is placed in the optional argument to the `\item` command, it will appear in a bold font by default.

```
7850 \ProvidesPackage{glossary-list}[2016/04/19 v4.22 (NLCT)]
```

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```

7851 \providecommand{\indexspace}{%
7852 \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
7853 }

```

tgrouphaderfmt Provide a way of adjusting the format of the group headings.

```
7854 \newcommand*{\glslistgrouphaderfmt}[1]{#1}
```

tnavigationitem Provide a way of adjusting the format of the navigation header. This puts the navigation line inside the optional argument of item to prevent unwanted space occurring at the start, but this can cause a problem if the navigation line is too long. With this command, it makes it easier for the user to customise the style without having to remember to modify `\glossaryheader` after the style has been set.

```
7855 \newcommand*{\glslistnavigationitem}[1]{\item[#1]}
```

list The list glossary style uses the description environment. The group separator `\glsgroupskip` is redefined as `\indexspace` which produces a gap between groups. The glossary heading and the group headings do nothing. Sub-entries immediately follow the main entry without the sub-entry name. This style does not use the entry's symbol. This is used as the default style for the glossaries package.

```
7856 \newglossarystyle{list}{%
  Use description environment:
7857   \renewenvironment{theglossary}%
7858     {\begin{description}}{\end{description}}%
  No header at the start of the environment:
7859   \renewcommand*{\glossaryheader}{}%
  No group headings:
7860   \renewcommand*{\glsgroupheading}[1]{}%
  Main (level 0) entries start a new item in the list:
7861   \renewcommand*{\glossentry}[2]{%
7862     \item[\glsentryitem{##1}]%
7863       \glstarget{##1}{\glossentryname{##1}}]
7864       \glossentrydesc{##1}\glspostdescription\space ##2}%
  Sub-entries continue on the same line:
7865   \renewcommand*{\subglossentry}[3]{%
7866     \glssubentryitem{##2}%
7867     \glstarget{##2}{\strut}\space
7868     \glossentrydesc{##2}\glspostdescription\space ##3.}%
7869 %   \end{macrocode}
7870 % Add vertical space between groups:
7871 %\changes{3.03}{2012/09/21}{added check for glsnogroupskip}
7872 %   \begin{macrocode}
7873   \renewcommand*{\glsgroupskip}{\ifglslsnogroupskip\else\indexspace\fi}%
7874 }
```

listgroup The listgroup style is like the list style, but the glossary groups have headings.

```
7875 \newglossarystyle{listgroup}{%
  Base it on the list style:
7876   \setglossarystyle{list}%

```

Each group has a heading:

```
7877 \renewcommand*{\glsgroupheading}[1]{%
7878   \item[\glslistgroupheaderfmt{\glsgrouptitle{##1}}]}
```

listhypergroup The listhypergroup style is like the listgroup style, but has a set of links to the groups at the start of the glossary.

```
7879 \newglossarystyle{listhypergroup}{%
```

Base it on the list style:

```
7880 \setglossarystyle{list}%
```

Add navigation links at the start of the environment.

```
7881 \renewcommand*{\glossaryheader}{%
7882   \glslstnavigationitem{\glslnavigation}}%
```

Each group has a heading with a hypertarget:

```
7883 \renewcommand*{\glsgroupheading}[1]{%
7884   \item[\glslistgroupheaderfmt
7885         {\glslnavhypertarget{##1}{\glsgrouptitle{##1}}}]}
```

altlist The altlist glossary style is like the list style, but places the description on a new line. Sub-entries follow in separate paragraphs without the sub-entry name. This style does not use the entry's symbol.

```
7886 \newglossarystyle{altlist}{%
```

Base it on the list style:

```
7887 \setglossarystyle{list}%
```

Main (level 0) entries start a new item in the list with a line break after the entry name:

```
7888 \renewcommand*{\glossentry}[2]{%
7889   \item[\glssentryitem{##1}%
7890         \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.

```
7891   \mbox{} \par \nobreak \@afterheading
7892   \glossentrydesc{##1} \glspostdescription \space ##2}%
```

Sub-entries start a new paragraph:

```
7893 \renewcommand{\subglossentry}[3]{%
7894   \par
7895   \glssubentryitem{##2}%
7896   \glstarget{##2}{\strut} \glossentrydesc{##2} \glspostdescription \space ##3}%
7897 }
```

altlistgroup The altlistgroup glossary style is like the altlist style, but the glossary groups have headings.

```
7898 \newglossarystyle{altlistgroup}{%
```

Base it on the altlist style:

```
7899 \setglossarystyle{altlist}%
```

Each group has a heading:

```
7900 \renewcommand*{\glsgroupheading}[1]{%
7901   \item[\glslistgroupheaderfmt{\glsgetgrouptitle{##1}}]}
```

altlisthypergroup The altlisthypergroup glossary style is like the altlistgroup style, but has a set of links to the groups at the start of the glossary.

```
7902 \newglossarystyle{altlisthypergroup}{%
```

Base it on the altlist style:

```
7903 \setglossarystyle{altlist}%
```

Add navigation links at the start of the environment.

```
7904 \renewcommand*{\glossaryheader}{%
7905   \glslistnavigationitem{\glsnavigation}}%
```

Each group has a heading with a hypertarget:

```
7906 \renewcommand*{\glsgroupheading}[1]{%
7907   \item[\glslistgroupheaderfmt
7908     {\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.

```
7909 \newglossarystyle{listdotted}{%
```

Base it on the list style:

```
7910 \setglossarystyle{list}%
```

Each main (level 0) entry starts a new item:

```
7911 \renewcommand*{\glossentry}[2]{%
7912   \item[]\makebox[\glslistdottedwidth][l]{%
7913     \glsentryitem{##1}%
7914     \glstarget{##1}{\glossentryname{##1}}%
7915     \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##1}}%
```

Sub entries have the same format as main entries:

```
7916 \renewcommand*{\subglossentry}[3]{%
7917   \item[]\makebox[\glslistdottedwidth][l]{%
7918     \glssubentryitem{##2}%
7919     \glstarget{##2}{\glossentryname{##2}}%
7920     \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##2}}%
7921 }
```

listdottedwidth

```
7922 \newlength\glslistdottedwidth
7923 \setlength{\glslistdottedwidth}{.5\hsize}
```

sublistdotted This style is similar to the glostylelistdotted style, except that the main entries just have the name displayed.

```
7924 \newglossarystyle{sublistdotted}{%
```

Base it on the listdotted style:

```
7925 \setglossarystyle{listdotted}%
```

Main (level 0) entries just display the name:

```
7926 \renewcommand*{\glossentry}[2]{%
7927   \item[\glentryitem{##1}\glstarget{##1}{\glossentryname{##1}}}%
7928 }
```

3.4 Glossary Styles using longtable (the glossary-long package)

The glossary styles defined in the package used the longtable environment in the glossary.

```
7929 \ProvidesPackage{glossary-long}[2016/04/19 v4.22 (NLCT)]
```

Requires the package:

```
7930 \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`.)

```
7931 \@ifundefined{glsdescwidth}{%
7932   \newlength{glsdescwidth}
7933   \setlength{glsdescwidth}{0.6\hsize}
7934 }{}
```

`\glspagelistwidth` This is a length that governs the width of the page list column.

```
7935 \@ifundefined{glspagelistwidth}{%
7936   \newlength{glspagelistwidth}
7937   \setlength{glspagelistwidth}{0.1\hsize}
7938 }{}
```

`long` The long glossary style command which uses the longtable environment:

```
7939 \newglossarystyle{long}{%
```

Use longtable with two columns:

```
7940   \renewenvironment{theglossary}%
7941     {\begin{longtable}[lp{glsdescwidth}]}%
7942     {\end{longtable}}%
```

Do nothing at the start of the environment:

```
7943   \renewcommand*{\glossaryheader}{}%
```

No heading between groups:

```
7944   \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries displayed in a row:

```
7945   \renewcommand{\glossentry}[2]{%
7946     \glentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
7947     \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline
7948   }%
```

Sub entries displayed on the following row without the name:

```

7949 \renewcommand{\subglossentry}[3]{%
7950     &
7951     \glssubentryitem{##2}%
7952     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
7953     ##3\tabularnewline
7954 }%
```

Blank row between groups:

```

7955 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else &
7956 \tabularnewline\fi}%
7957 }
```

longborder The longborder style is like the above, but with horizontal and vertical lines:

```

7958 \newglossarystyle{longborder}{%
```

Base it on the `glostylelong` style:

```

7959 \setglossarystyle{long}%
```

Use `longtable` with two columns with vertical lines between each column:

```

7960 \renewenvironment{theglossary}{%
7961     \begin{longtable}{|l|p{\glsdescwidth}|}{\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```

7962 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
7963 }
```

longheader The longheader style is like the long style but with a header:

```

7964 \newglossarystyle{longheader}{%
```

Base it on the `glostylelong` style:

```

7965 \setglossarystyle{long}%
```

Set the table's header:

```

7966 \renewcommand*{\glossaryheader}{%
7967     \bfseries \entryname & \bfseries \descriptionname\tabularnewline\endhead}%
7968 }
```

longheaderborder The longheaderborder style is like the long style but with a header and border:

```

7969 \newglossarystyle{longheaderborder}{%
```

Base it on the `glostylelongborder` style:

```

7970 \setglossarystyle{longborder}%
```

Set the table's header and add horizontal line to table's foot:

```

7971 \renewcommand*{\glossaryheader}{%
7972     \hline\bfseries \entryname & \bfseries
7973     \descriptionname\tabularnewline\hline
7974     \endhead
7975     \hline\endfoot}%
7976 }
```


long3col The long3col style is like long but with 3 columns

```
7977 \newglossarystyle{long3col}{%
```

Use a longtable with 3 columns:

```
7978 \renewenvironment{theglossary}%
```

```
7979 {\begin{longtable}{lp{\glstdescwidth}p{\glspagelistwidth}}}%
```

```
7980 {\end{longtable}}%
```

No table header:

```
7981 \renewcommand*{\glossaryheader}{}%
```

No headings between groups:

```
7982 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
7983 \renewcommand{\glossentry}[2]{%
```

```
7984 \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
```

```
7985 \glossentrydesc{##1} & ##2\tabularnewline
```

```
7986 }%
```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```
7987 \renewcommand{\subglossentry}[3]{%
```

```
7988 &
```

```
7989 \glssubentryitem{##2}%
```

```
7990 \glstarget{##2}{\strut}\glossentrydesc{##2} &
```

```
7991 ##3\tabularnewline
```

```
7992 }%
```

Blank row between groups:

```
7993 \renewcommand*{\glsgroupskip}{}%
```

```
7994 \ifglsgroupskip\else & &\tabularnewline\fi}%
```

```
7995 }
```

long3colborder The long3colborder style is like the long3col style but with a border:

```
7996 \newglossarystyle{long3colborder}{%
```

Base it on the glostylelong3col style:

```
7997 \setglossarystyle{long3col}%
```

Use a longtable with 3 columns with vertical lines around them:

```
7998 \renewenvironment{theglossary}%
```

```
7999 {\begin{longtable}{|lp{\glstdescwidth}|p{\glspagelistwidth}|}}%
```

```
8000 {\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
8001 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
```

```
8002 }
```

long3colheader The long3colheader style is like long3col but with a header row:

```
8003 \newglossarystyle{long3colheader}{%
```

Base it on the `glostylelong3col` style:

```
8004 \setglossarystyle{long3col}%
```

Set the table's header:

```
8005 \renewcommand*{\glossaryheader}{%
8006   \bfseries\entryname&\bfseries\descriptionname&
8007   \bfseries\pagelistname\tabularnewline\endhead}%
8008 }
```

`colheaderborder` The `long3colheaderborder` style is like the above but with a border

```
8009 \newglossarystyle{long3colheaderborder}{%
```

Base it on the `glostylelong3colborder` style:

```
8010 \setglossarystyle{long3colborder}%
```

Set the table's header and add horizontal line at table's foot:

```
8011 \renewcommand*{\glossaryheader}{%
8012   \hline
8013   \bfseries\entryname&\bfseries\descriptionname&
8014   \bfseries\pagelistname\tabularnewline\hline\endhead
8015   \hline\endfoot}%
8016 }
```

`long4col` The `long4col` style has four columns where the third column contains the value of the associated symbol key.

```
8017 \newglossarystyle{long4col}{%
```

Use a longtable with 4 columns:

```
8018 \renewenvironment{theglossary}%
8019   {\begin{longtable}{l1l1l}}%
8020   {\end{longtable}}%
```

No table header:

```
8021 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8022 \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):

```
8023 \renewcommand{\glossentry}[2]{%
8024   \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8025   \glossentrydesc{##1} &
8026   \glossentrysymbol{##1} &
8027   ##2\tabularnewline
8028   }%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
8029 \renewcommand{\subglossentry}[3]{%
8030   &
```

```

8031 \glssubentryitem{##2}%
8032 \glstarget{##2}{\strut}\glossentrydesc{##2} &
8033 \glossentrysymbol{##2} & ##3\tabularnewline
8034 }%

```

Blank row between groups:

```

8035 \renewcommand*{\glsgroupskip}{%
8036 \ifglsgnোগroupskip\else & & \tabularnewline\fi}%
8037 }

```

long4colheader The long4colheader style is like long4col but with a header row.

```

8038 \newglossarystyle{long4colheader}{%

```

Base it on the glostylelong4col style:

```

8039 \setglossarystyle{long4col}%

```

Table has a header:

```

8040 \renewcommand*{\glossaryheader}{%
8041 \bfseries\entryname&\bfseries\descriptionname&
8042 \bfseries \symbolname&
8043 \bfseries\pagelistname\tabularnewline\endhead}%
8044 }

```

long4colborder The long4colborder style is like long4col but with a border.

```

8045 \newglossarystyle{long4colborder}{%

```

Base it on the glostylelong4col style:

```

8046 \setglossarystyle{long4col}%

```

Use a longtable with 4 columns surrounded by vertical lines:

```

8047 \renewenvironment{theglossary}%
8048 {\begin{longtable}{|l|l|l|l|}}%
8049 {\end{longtable}}%

```

Add horizontal lines to the head and foot of the table:

```

8050 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8051 }

```

colheaderborder The long4colheaderborder style is like the above but with a border.

```

8052 \newglossarystyle{long4colheaderborder}{%

```

Base it on the glostylelong4col style:

```

8053 \setglossarystyle{long4col}%

```

Use a longtable with 4 columns surrounded by vertical lines:

```

8054 \renewenvironment{theglossary}%
8055 {\begin{longtable}{|l|l|l|l|}}%
8056 {\end{longtable}}%

```

Add table header and horizontal line at the table's foot:

```

8057 \renewcommand*{\glossaryheader}{%
8058 \hline\bfseries\entryname&\bfseries\descriptionname&

```

```

8059     \bfseries \symbolname&
8060     \bfseries\pagelistname\tabularnewline\hline\endhead
8061     \hline\endfoot}%
8062 }

```

altlong4col The altlong4col style is like the long4col style but can have multiline descriptions and page lists.

```

8063 \newglossarystyle{altlong4col}{%
    Base it on the glostylelong4col style:
8064     \setglossarystyle{long4col}%
    Use a longtable with 4 columns where the second and last columns may have multiple lines
    in each row:
8065     \renewenvironment{theglossary}%
8066         {\begin{longtable}{lp{\glstdescwidth}lp{\glspagelistwidth}}}%
8067         {\end{longtable}}%
8068 }

```

altlong4colheader The altlong4colheader style is like altlong4col but with a header row.

```

8069 \newglossarystyle{altlong4colheader}{%
    Base it on the glostylelong4colheader style:
8070     \setglossarystyle{long4colheader}%
    Use a longtable with 4 columns where the second and last columns may have multiple lines
    in each row:
8071     \renewenvironment{theglossary}%
8072         {\begin{longtable}{lp{\glstdescwidth}lp{\glspagelistwidth}}}%
8073         {\end{longtable}}%
8074 }

```

altlong4colborder The altlong4colborder style is like altlong4col but with a border.

```

8075 \newglossarystyle{altlong4colborder}{%
    Base it on the glostylelong4colborder style:
8076     \setglossarystyle{long4colborder}%
    Use a longtable with 4 columns where the second and last columns may have multiple lines
    in each row:
8077     \renewenvironment{theglossary}%
8078         {\begin{longtable}{|lp{\glstdescwidth}|lp{\glspagelistwidth}|}%
8079         {\end{longtable}}%
8080 }

```

altlong4colheaderborder The altlong4colheaderborder style is like the above but with a header as well as a border.

```

8081 \newglossarystyle{altlong4colheaderborder}{%
    Base it on the glostylelong4colheaderborder style:
8082     \setglossarystyle{long4colheaderborder}%

```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8083 \renewenvironment{theglossary}%
8084   {\begin{longtable}{|l|p{\glstdescwidth}|l|p{\glspagelistwidth}|}%
8085   {\end{longtable}}%
8086 }
```

3.5 Glossary Styles using longtable and booktabs (the glossary-longbooktabs) package

The styles here are based on David Carlisle's patch at <http://tex.stackexchange.com/a/56890>

```
8087 \ProvidesPackage{glossary-longbooktabs}[2016/04/19 v4.22 (NLCT)]
```

Requires booktabs package:

```
8088 \RequirePackage{booktabs}
```

and the base packages for long styles:

```
8089 \RequirePackage{glossary-long}
```

```
8090 \RequirePackage{glossary-longragged}
```

(longtable and array loaded by those packages).

long-booktabs The long-booktabs style is similar to the longheader style but uses the booktabs rules and patches longtable to check for group skip occurring at a page break.

```
8091 \newglossarystyle{long-booktabs}{%
```

If the style change is scoped, the patch will only have a local effect, which may be useful if it conflicts with other tables in the document.

```
8092 \glspatchLToutput
```

As with the longheader style, use the long style as a base.

```
8093 \setglossarystyle{long}%
```

Add a header with rules.

```
8094 \renewcommand*{\glossaryheader}{%
```

```
8095   \toprule \bfseries \entryname & \bfseries
```

```
8096   \descriptionname\tabularnewline\midrule\endhead
```

```
8097   \bottomrule\endfoot}%
```

Check for the nogroupskip package option. If there should be a gap between groups, insert the penalty and the vertical space.

```
8098 \renewcommand*{\glsgroupskip}{%
```

```
8099   \ifglsnogroupskip \else \glspenaltygroupskip\fi}%
```

```
8100 }
```

ng3col-booktabs The long3col-booktabs style is similar to the long3colheader style but uses the booktabs rules and patches longtable to check for group skip occurring at a page break.

```
8101 \newglossarystyle{long3col-booktabs}{%
```

If the style change is scoped, the patch will only have a local effect, which may be useful if it conflicts with other tables in the document.

```
8102 \glspatchLToutput
```

Use the long3col style as a base.

```
8103 \setglossarystyle{long3col}%
```

Add a header with rules.

```
8104 \renewcommand*{\glossaryheader}{%
8105   \toprule \bfseries \entryname &
8106   \bfseries \descriptionname &
8107   \bfseries \pagelistname
8108   \tabularnewline\midrule\endhead
8109   \bottomrule\endfoot}%
```

Check for the nogroupskip package option. If there should be a gap between groups, insert the penalty and the vertical space.

```
8110 \renewcommand*{\glsgroupskip}{%
8111   \ifglsnogroupskip \else \glspenaltygroupskip\fi}%
8112 }
```

ng4col-booktabs The long4col-booktabs style is similar to the long4colheader style but uses the booktabs rules and patches longtable to check for group skip occurring at a page break.

```
8113 \newglossarystyle{long4col-booktabs}{%
```

If the style change is scoped, the patch will only have a local effect, which may be useful if it conflicts with other tables in the document.

```
8114 \glspatchLToutput
```

Use the long4col style as a base.

```
8115 \setglossarystyle{long4col}%
```

Add a header with rules.

```
8116 \renewcommand*{\glossaryheader}{%
8117   \toprule \bfseries \entryname &
8118   \bfseries \descriptionname &
8119   \bfseries \symbolname &
8120   \bfseries \pagelistname
8121   \tabularnewline\midrule\endhead
8122   \bottomrule\endfoot}%
```

Check for the nogroupskip package option. If there should be a gap between groups, insert the penalty and the vertical space.

```
8123 \renewcommand*{\glsgroupskip}{%
8124   \ifglsnogroupskip \else \glspenaltygroupskip\fi}%
8125 }
```

ng4col-booktabs The altlong4col-booktabs style is similar to the altlong4colheader style but uses the booktabs rules and patches longtable to check for group skip occurring at a page break.

```
8126 \newglossarystyle{altlong4col-booktabs}{%
```

If the style change is scoped, the patch will only have a local effect, which may be useful if it conflicts with other tables in the document.

```
8127 \glspatchLToutput
```

Use the long4col-booktabs style as a base.

```
8128 \setglossarystyle{long4col-booktabs}%
```

Change the column specifications:

```
8129 \renewenvironment{theglossary}%
```

```
8130 {\begin{longtable}{lp{\glstdescwidth}lp{\glspagelistwidth}}}%
```

```
8131 {\end{longtable}}%
```

```
8132 }
```

Ragged styles.

ragged-booktabs The longragged-booktabs style is similar to the longragged style but uses the booktabs rules and patches longtable to check for group skip occurring at a page break.

```
8133 \newglossarystyle{longragged-booktabs}{%
```

If the style change is scoped, the patch will only have a local effect, which may be useful if it conflicts with other tables in the document.

```
8134 \glspatchLToutput
```

Use the long-booktabs style as a base.

```
8135 \setglossarystyle{long-booktabs}%
```

Adjust the column specification.

```
8136 \renewenvironment{theglossary}%
```

```
8137 {\begin{longtable}{l>{\raggedright}p{\glstdescwidth}}}%
```

```
8138 {\end{longtable}}%
```

```
8139 }
```

ed3col-booktabs The longragged3col-booktabs style is similar to the longragged3col style but uses the booktabs rules and patches longtable to check for group skip occurring at a page break.

```
8140 \newglossarystyle{longragged3col-booktabs}{%
```

If the style change is scoped, the patch will only have a local effect, which may be useful if it conflicts with other tables in the document.

```
8141 \glspatchLToutput
```

Use the long3col-booktabs style as a base.

```
8142 \setglossarystyle{long3col-booktabs}%
```

Adjust the column specification.

```
8143 \renewenvironment{theglossary}%
```

```
8144 {\begin{longtable}{l>{\raggedright}p{\glstdescwidth}%
```

```
8145 >{\raggedright}p{\glspagelistwidth}}}%
```

```
8146 {\end{longtable}}%
```

```
8147 }
```

ed4col-booktabs The altlongragged4col-booktabs style is similar to the altlongragged4col style but uses the booktabs rules and patches longtable to check for group skip occurring at a page break.

```
8148 \newglossarystyle{altlongragged4col-booktabs}{%
```

If the style change is scoped, the patch will only have a local effect, which may be useful if it conflicts with other tables in the document.

```
8149 \glspatchLToutput
```

Use the altlong4col-booktabs style as a base.

```
8150 \setglossarystyle{altlong4col-booktabs}%
```

Adjust the column specification.

```
8151 \renewenvironment{theglossary}%
```

```
8152 {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}l%
```

```
8153 >{\raggedright}p{\glspagelistwidth}}}%
```

```
8154 {\end{longtable}}%
```

```
8155 }
```

sLTpenaltycheck

```
8156 \newcommand*{\glslTpenaltycheck}{%
```

```
8157 \ifnum\outputpenalty=-50\vskip-\normalbaselineskip\relax\fi
```

```
8158 }
```

enaltygroupskip

```
8159 \newcommand{\glspenaltygroupskip}{%
```

```
8160 \noalign{\penalty-50\vskip\normalbaselineskip}}
```

restoreLToutput Provide a way of restoring \LT@output for the user.

```
8161 \let\@gls@org@LT@output\LT@output
```

```
8162 \newcommand*{\glstoreLToutput}{\let\LT@output\@gls@org@LT@output}
```

This is David's patch, but I've replaced the hard-coded values with \glslTpenaltycheck to make it easier to adjust.

lspatchLToutput

```
8163 \newcommand*{\glspatchLToutput}{%
```

```
8164 \renewcommand*{\LT@output}{%
```

```
8165 \ifnum\outputpenalty <-\@Mi
```

```
8166 \ifnum\outputpenalty > -\LT@end@pen
```

```
8167 \LT@err{floats and marginpars not allowed in a longtable}\@ehc
```

```
8168 \else
```

```
8169 \setbox\z@\vbox{\unvbox\@cclv}%
```

```
8170 \ifdim \ht\LT@lastfoot>\ht\LT@foot
```

```
8171 \dimen@\pagegoal
```

```
8172 \advance\dimen@-\ht\LT@lastfoot
```

```
8173 \ifdim\dimen@<\ht\z@
```

```
8174 \setbox\@cclv\vbox{\unvbox\z@\copy\LT@foot\vss}%
```

```
8175 \makecol
```

```
8176 \@outputpage
```



```

8177         \setbox\z@\vbox{\box\LT@head\glsLTpenaltycheck}%
8178         \fi
8179     \fi
8180     \global\@colroom\@colht
8181     \global\vsizel\@colht
8182     {\unvbox\z@\box\ifvoid\LT@lastfoot\LT@foot\else\LT@lastfoot\fi}%
8183     \fi
8184 \else
8185     \setbox\@cclv\vbox{\unvbox\@cclv\copy\LT@foot\vss}%
8186     \@makecol
8187     \@outputpage
8188     \global\vsizel\@colroom
8189     \copy\LT@head
8190     \glsLTpenaltycheck
8191     \nobreak
8192 \fi
8193 }%
8194 }

```

3.6 Glossary Styles using longtable (the glossary-longragged package)

The glossary styles defined in the package used the longtable environment in the glossary and use ragged right formatting for the multiline columns.

8195 \ProvidesPackage{glossary-longragged}[2016/04/19 v4.22 (NLCT)]

Requires the package:

8196 \RequirePackage{array}

Requires the package:

8197 \RequirePackage{longtable}

`\glsdescwidth` This is a length that governs the width of the description column. This may have already been defined.

```

8198 \@ifundefined{glsdescwidth}{%
8199     \newlength\glsdescwidth
8200     \setlength{\glsdescwidth}{0.6\hsize}
8201 }{}

```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```

8202 \@ifundefined{glspagelistwidth}{%
8203     \newlength\glspagelistwidth
8204     \setlength{\glspagelistwidth}{0.1\hsize}
8205 }{}

```

`longragged` The longragged glossary style is like the long but uses ragged right formatting for the description column.

8206 \newglossarystyle{longragged}{%

Use longtable with two columns:

8207 \renewenvironment{theglossary}{%

8208 \begin{longtable}[l>{\raggedright}p{\glsgdescwidth}}}%

8209 \end{longtable}}%

Do nothing at the start of the environment:

8210 \renewcommand*{\glossaryheader}{}%

No heading between groups:

8211 \renewcommand*{\glsgroupheading}[1]{}%

Main (level 0) entries displayed in a row:

8212 \renewcommand{\glossentry}[2]{%

8213 \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &

8214 \glossentrydesc{##1}\glspostdescription\space ##2%

8215 \tabularnewline

8216 }%

Sub entries displayed on the following row without the name:

8217 \renewcommand{\subglossentry}[3]{%

8218 &

8219 \glssubentryitem{##2}%

8220 \glstarget{##2}{\strut}\glossentrydesc{##2}%

8221 \glspostdescription\space ##3%

8222 \tabularnewline

8223 }%

Blank row between groups:

8224 \renewcommand*{\glsgroupskip}{\ifglsgnোগroupskip\else & \tabularnewline\fi}%

8225 }

ongraggedborder The longraggedborder style is like the above, but with horizontal and vertical lines:

8226 \newglossarystyle{longraggedborder}{%

Base it on the glostylelongragged style:

8227 \setglossarystyle{longragged}%

Use longtable with two columns with vertical lines between each column:

8228 \renewenvironment{theglossary}{%

8229 \begin{longtable}{|l|>{\raggedright}p{\glsgdescwidth}|}}%

8230 \end{longtable}}%

Place horizontal lines at the head and foot of the table:

8231 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%

8232 }

ongraggedheader The longraggedheader style is like the longragged style but with a header:

8233 \newglossarystyle{longraggedheader}{%

Base it on the glostylelongragged style:

8234 \setglossarystyle{longragged}%

Set the table's header:

```
8235 \renewcommand*{\glossaryheader}{%
8236 \bfseries \entryname & \bfseries \descriptionname
8237 \tabularnewline\endhead}%
8238 }
```

gedheaderborder The longraggedheaderborder style is like the longragged style but with a header and border:

```
8239 \newglossarystyle{longraggedheaderborder}{%
```

Base it on the glostylelongraggedborder style:

```
8240 \setglossarystyle{longraggedborder}{%
```

Set the table's header and add horizontal line to table's foot:

```
8241 \renewcommand*{\glossaryheader}{%
8242 \hline\bfseries \entryname & \bfseries \descriptionname
8243 \tabularnewline\hline
8244 \endhead
8245 \hline\endfoot}%
8246 }
```

longragged3col The longragged3col style is like longragged but with 3 columns

```
8247 \newglossarystyle{longragged3col}{%
```

Use a longtable with 3 columns:

```
8248 \renewenvironment{theglossary}%
8249 {\begin{longtable}{l>{\raggedright}p{\glstdescwidth}%
8250 >{\raggedright}p{\glspagelistwidth}}}%
8251 {\end{longtable}}%
```

No table header:

```
8252 \renewcommand*{\glossaryheader}{}%
```

No headings between groups:

```
8253 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
8254 \renewcommand{\glossentry}[2]{%
8255 \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8256 \glossentrydesc{##1} & ##2\tabularnewline
8257 }%
```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```
8258 \renewcommand{\subglossentry}[3]{%
8259 &
8260 \glssubentryitem{##2}%
8261 \glstarget{##2}{\strut}\glossentrydesc{##2} &
8262 ##3\tabularnewline
8263 }%
```

Blank row between groups:

```
8264 \renewcommand*{\glsgroupskip}{%
8265 \ifglsnogroupskip\else & \tabularnewline\fi}%
8266 }
```

agged3colborder The longragged3colborder style is like the longragged3col style but with a border:

```
8267 \newglossarystyle{longragged3colborder}{%
```

Base it on the glostylelongragged3col style:

```
8268 \setglossarystyle{longragged3col}%
```

Use a longtable with 3 columns with vertical lines around them:

```
8269 \renewenvironment{theglossary}%
8270 {\begin{longtable}{|l|>{\raggedright}p{\glsdescwidth}|%
8271 >{\raggedright}p{\glspagelistwidth}|}%
8272 {\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
8273 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8274 }
```

agged3colheader The longragged3colheader style is like longragged3col but with a header row:

```
8275 \newglossarystyle{longragged3colheader}{%
```

Base it on the glostylelongragged3col style:

```
8276 \setglossarystyle{longragged3col}%
```

Set the table's header:

```
8277 \renewcommand*{\glossaryheader}{%
8278 \bfseries\entryname&\bfseries\descriptionname&
8279 \bfseries\pagelistname\tabularnewline\endhead}%
8280 }
```

colheaderborder The longragged3colheaderborder style is like the above but with a border

```
8281 \newglossarystyle{longragged3colheaderborder}{%
```

Base it on the glostylelongragged3colborder style:

```
8282 \setglossarystyle{longragged3colborder}%
```

Set the table's header and add horizontal line at table's foot:

```
8283 \renewcommand*{\glossaryheader}{%
8284 \hline
8285 \bfseries\entryname&\bfseries\descriptionname&
8286 \bfseries\pagelistname\tabularnewline\hline\endhead
8287 \hline\endfoot}%
8288 }
```

tlongragged4col 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.

```
8289 \newglossarystyle{altlongragged4col}{%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8290 \renewenvironment{theglossary}%
8291   {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}l%
8292     >{\raggedright}p{\glspagelistwidth}}}%
8293   {\end{longtable}}%
```

No table header:

```
8294 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8295 \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):

```
8296 \renewcommand{\glossentry}[2]{%
8297   \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8298   \glossentrydesc{##1} & \glossentrysymbol{##1} &
8299   ##2\tabularnewline
8300 }%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
8301 \renewcommand{\subglossentry}[3]{%
8302   &
8303   \glssubentryitem{##2}%
8304   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8305   \glossentrysymbol{##2} & ##3\tabularnewline
8306 }%
```

Blank row between groups:

```
8307 \renewcommand*{\glsgroupskip}{%
8308   \ifglsnogroupskip\else & & \tabularnewline\fi}%
8309 }
```

agged4colheader The altlongragged4colheader style is like altlongragged4col but with a header row.

```
8310 \newglossarystyle{altlongragged4colheader}{%
```

Base it on the glostylealtlongragged4col style:

```
8311 \setglossarystyle{altlongragged4col}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8312 \renewenvironment{theglossary}%
8313   {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}l%
8314     >{\raggedright}p{\glspagelistwidth}}}%
8315   {\end{longtable}}%
```

Table has a header:

```
8316 \renewcommand*{\glossaryheader}{%
8317   \bfseries\entryname&\bfseries\descriptionname&
8318   \bfseries \symbolname&
```

```

8319 \bfseries\pagelistname\tabularnewline\endhead}%
8320 }

```

ragged4colborder The `altlongragged4colborder` style is like `altlongragged4col` but with a border.

```

8321 \newglossarystyle{altlongragged4colborder}{%

```

Base it on the `glostylealtlongragged4col` style:

```

8322 \setglossarystyle{altlongragged4col}%

```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```

8323 \renewenvironment{theglossary}%
8324 {\begin{longtable}{|l|>{\raggedright}p{\glsgdescwidth}|l|}%
8325 >{\raggedright}p{\glspagelistwidth}|}%
8326 {\end{longtable}}%

```

Add horizontal lines to the head and foot of the table:

```

8327 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8328 }

```

colheaderborder The `altlongragged4colheaderborder` style is like the above but with a header as well as a border.

```

8329 \newglossarystyle{altlongragged4colheaderborder}{%

```

Base it on the `glostylealtlongragged4col` style:

```

8330 \setglossarystyle{altlongragged4col}%

```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```

8331 \renewenvironment{theglossary}%
8332 {\begin{longtable}{|l|>{\raggedright}p{\glsgdescwidth}|l|}%
8333 >{\raggedright}p{\glspagelistwidth}|}%
8334 {\end{longtable}}%

```

Add table header and horizontal line at the table's foot:

```

8335 \renewcommand*{\glossaryheader}{%
8336 \hline\bfseries\entryname&\bfseries\descriptionname&
8337 \bfseries \symbolname&
8338 \bfseries\pagelistname\tabularnewline\hline\endhead
8339 \hline\endfoot}%
8340 }

```

3.7 Glossary Styles using multicol (glossary-mcols.sty)

The style file defines glossary styles that use the `multicol` package. These use the tree-like glossary styles in a `multicol` environment.

```

8341 \ProvidesPackage{glossary-mcols}[2016/04/19 v4.22 (NLCT)]

```

Required packages:

```

8342 \RequirePackage{multicol}
8343 \RequirePackage{glossary-tree}

```

`\indexspace` The are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```
8344 \providecommand{\indexspace}{%
8345   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
8346 }
```

`\glsmcols` Define macro in which to store the number of columns. (Defaults to 2.)

```
8347 \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.)

```
8348 \newglossarystyle{mcolindex}{%
8349   \setglossarystyle{index}%
8350   \renewenvironment{theglossary}%
8351     {%
8352       \begin{multicols}{\glsmcols}
8353       \setlength{\parindent}{0pt}%
8354       \setlength{\parskip}{0pt plus 0.3pt}%
8355       \let\item\@idxitem}%
8356   {\end{multicols}}%
8357 }
```

`mcolindexgroup` As `mcolindex` but has headings:

```
8358 \newglossarystyle{mcolindexgroup}{%
8359   \setglossarystyle{mcolindex}%
8360   \renewcommand*\glsgroupheading[1]{%
8361     \item\glstreegroupheaderfmt{\glsgrouptitle{##1}}\indexspace}%
8362 }
```

`indexhypergroup` The `mcolindexhypergroup` style is like the `mcolindexgroup` style but has hyper navigation.

```
8363 \newglossarystyle{mcolindexhypergroup}{%
```

Base it on the `glostylemcolindex` style:

```
8364   \setglossarystyle{mcolindex}%
```

Put navigation links to the groups at the start of the glossary:

```
8365   \renewcommand*\glossaryheader{%
8366     \item\glstreenavigationfmt{\glsnavigation}\indexspace}%
```

Add a heading for each group (with a target). The group's title is in bold followed by a vertical gap.

```
8367   \renewcommand*\glsgroupheading[1]{%
8368     \item\glstreegroupheaderfmt
8369       {\glsnavigationhypertarget{##1}{\glsgrouptitle{##1}}}%
8370   \indexspace}%
8371 }
```

`colindexspannav` Similar to `colindexhypergroup`, but puts the navigation line in the optional argument of `multicols`.

```

8372 \newglossarystyle{colindexspannav}{%
8373   \setglossarystyle{index}%
8374   \renewenvironment{theglossary}%
8375     {%
8376       \begin{multicols}{\glsmcols}[\noindent\glstreenavigationfmt{\glsnavigation}]
8377       \setlength{\parindent}{0pt}%
8378       \setlength{\parskip}{0pt plus 0.3pt}%
8379       \let\item\@idxitem}%
8380   {\end{multicols}}}%

```

Add a heading for each group (with a target). The group's title is in bold followed by a vertical gap.

```

8381 \renewcommand*{\glsgroupheading}[1]{%
8382   \item\glstreegroupheaderfmt
8383     {\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}%
8384   \indexspace}%
8385 }

```

`mcoltree` Multi-column index style. Same as the tree, but puts the glossary in multiple columns.

```

8386 \newglossarystyle{mcoltree}{%
8387   \setglossarystyle{tree}%
8388   \renewenvironment{theglossary}%
8389     {%
8390       \begin{multicols}{\glsmcols}
8391       \setlength{\parindent}{0pt}%
8392       \setlength{\parskip}{0pt plus 0.3pt}%
8393     }%
8394   {\end{multicols}}}%
8395 }

```

`mcoltreegroup` Like the `mcoltree` style but the glossary groups have headings.

```

8396 \newglossarystyle{mcoltreegroup}{%
8397   Base it on the glostylemcoltree style:
8397   \setglossarystyle{mcoltree}%
8398   Each group has a heading (in bold) followed by a vertical gap):
8398   \renewcommand{\glsgroupheading}[1]{\par
8399     \noindent\glstreegroupheaderfmt{\glsgrouptitle{##1}}\par\indexspace}%
8400 }

```

`ltreehypergroup` The `mcoltreehypergroup` style is like the `treegroup` style, but has a set of links to the groups at the start of the glossary.

```

8401 \newglossarystyle{mcoltreehypergroup}{%
8402   Base it on the glostylemcoltree style:
8402   \setglossarystyle{mcoltree}%

```


Put navigation links to the groups at the start of the theglossary environment:

```
8403 \renewcommand*{\glossaryheader}{%
8404 \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
8405 \renewcommand*{\glsgroupheading}[1]{%
8406 \par\noindent
8407 \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8408 \indexspace}%
8409 }
```

mcoltreespannav Similar to the mcoltreehypergroup style but the navigation line is put in the optional argument of the multicols environment.

```
8410 \newglossarystyle{mcoltreespannav}{%
8411 \setglossarystyle{tree}%
8412 \renewenvironment{theglossary}%
8413 {%
8414 \begin{multicols}{\glsmcols}[\noindent\glstreenavigationfmt{\glsnavigation}]
8415 \setlength{\parindent}{0pt}%
8416 \setlength{\parskip}{0pt plus 0.3pt}%
8417 }%
8418 {\end{multicols}}}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
8419 \renewcommand*{\glsgroupheading}[1]{%
8420 \par\noindent
8421 \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8422 \indexspace}%
8423 }
```

mcoltreenoname Multi-column index style. Same as the treenoname, but puts the glossary in multiple columns.

```
8424 \newglossarystyle{mcoltreenoname}{%
8425 \setglossarystyle{treenoname}%
8426 \renewenvironment{theglossary}%
8427 {%
8428 \begin{multicols}{\glsmcols}
8429 \setlength{\parindent}{0pt}%
8430 \setlength{\parskip}{0pt plus 0.3pt}%
8431 }%
8432 {\end{multicols}}}%
8433 }
```

treenonamegroup Like the mcoltreenoname style but the glossary groups have headings.

```
8434 \newglossarystyle{mcoltreenonamegroup}{%
Base it on the glostylemcoltreenoname style:
8435 \setglossarystyle{mcoltreenoname}%
```

Give each group a heading:

```
8436 \renewcommand{\glsgroupheading}[1]{\par
8437 \noindent\glstreegroupheaderfmt{\glsgrouptitle{##1}}\par\indexspace}%
8438 }
```

`onamehypergroup` The `mcoltreenonamehypergroup` style is like the `mcoltreenonamegroup` style, but has a set of links to the groups at the start of the glossary.

```
8439 \newglossarystyle{mcoltreenonamehypergroup}{%
```

Base it on the `glostylemcoltreenoname` style:

```
8440 \setglossarystyle{mcoltreenoname}%
```

Put navigation links to the groups at the start of the `theglossary` environment:

```
8441 \renewcommand*{\glossaryheader}{%
8442 \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
8443 \renewcommand*{\glsgroupheading}[1]{%
8444 \par\noindent
8445 \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}\par
8446 \indexspace}%
8447 }
```

`eenonamespannav` Similar to the `mcoltreenonamehypergroup` style but the navigation line is put in the optional argument of the `multicols` environment.

```
8448 \newglossarystyle{mcoltreenonamespannav}{%
8449 \setglossarystyle{treenoname}%
8450 \renewenvironment{theglossary}%
8451 {%
8452 \begin{multicols}{\glsmcols}\noindent\glstreenavigationfmt{\glsnavigation}]
8453 \setlength{\parindent}{0pt}%
8454 \setlength{\parskip}{0pt plus 0.3pt}%
8455 }%
8456 {\end{multicols}}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
8457 \renewcommand*{\glsgroupheading}[1]{%
8458 \par\noindent
8459 \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}\par
8460 \indexspace}%
8461 }
```

`mcolalttree` Multi-column index style. Same as the `alttree`, but puts the glossary in multiple columns.

```
8462 \newglossarystyle{mcolalttree}{%
8463 \setglossarystyle{alttree}%
8464 \renewenvironment{theglossary}%
8465 {%
8466 \begin{multicols}{\glsmcols}
8467 \def\@gls@prevlevel{-1}%
```

```

8468     \mbox{}\par
8469   }%
8470   {\par\end{multicols}}}%
8471 }

```

colalmtreegroup Like the mcolalmtree style but the glossary groups have headings.

```

8472 \newglossarystyle{mcolalmtreegroup}{%
    Base it on the glostylemcolalmtree style:
8473   \setglossarystyle{mcolalmtree}%
    Give each group a heading.
8474   \renewcommand{\glsgroupheading}[1]{\par
8475     \def\@gls@prevlevel{-1}%
8476     \hangindent0pt\relax
8477     \parindent0pt\relax
8478     \glstreegroupheaderfmt{\glsgrouptitle{##1}}\par\indexspace}%
8479 }

```

treehypergroup The mcolalmtreehypergroup style is like the mcolalmtreegroup style, but has a set of links to the groups at the start of the glossary.

```

8480 \newglossarystyle{mcolalmtreehypergroup}{%
    Base it on the glostylemcolalmtree style:
8481   \setglossarystyle{mcolalmtree}%
    Put the navigation links in the header
8482   \renewcommand*{\glossaryheader}{%
8483     \par
8484     \def\@gls@prevlevel{-1}%
8485     \hangindent0pt\relax
8486     \parindent0pt\relax
8487     \glstreenavigationfmt{\glsnavigation}\par\indexspace}%
    Put a hypertarget at the start of each group
8488   \renewcommand*{\glsgroupheading}[1]{%
8489     \par
8490     \def\@gls@prevlevel{-1}%
8491     \hangindent0pt\relax
8492     \parindent0pt\relax
8493     \glstreegroupheaderfmt{\glssnavhypertarget{##1}{\glsgrouptitle{##1}}}\par
8494     \indexspace}%
8495 }

```

almtreeespannav Similar to the mcolalmtreehypergroup style but the navigation line is put in the optional argument of the multicols environment.

```

8496 \newglossarystyle{mcolalmtreeespannav}{%
8497   \setglossarystyle{almtree}%
8498   \renewenvironment{theglossary}%
8499   {%
8500     \begin{multicols}{\glsmcols}\noindent\glstreenavigationfmt{\glsnavigation}]

```

```

8501      \def\@gls@prevlevel{-1}%
8502      \mbox{}\par
8503  }%
8504  {\par\end{multicols}}}%

  Put a hypertarget at the start of each group
8505  \renewcommand*\@glsgroupheading[1]{%
8506    \par
8507    \def\@gls@prevlevel{-1}%
8508    \hangindent0pt\relax
8509    \parindent0pt\relax
8510    \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8511    \indexspace}
8512  }

```

3.8 Glossary Styles using supertabular environment (glossary-super package)

The glossary styles defined in the package use the supertabular environment.

```

8513 \ProvidesPackage{glossary-super}[2016/04/19 v4.22 (NLCT)]

```

Requires the package:

```

8514 \RequirePackage{supertabular}

```

`\glsdescwidth` This is a length that governs the width of the description column. This may already have been defined if has been loaded.

```

8515 \@ifundefined{glsdescwidth}{%
8516   \newlength\glsdescwidth
8517   \setlength{\glsdescwidth}{0.6\hsize}
8518 }{}

```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined if has been loaded.

```

8519 \@ifundefined{glspagelistwidth}{%
8520   \newlength\glspagelistwidth
8521   \setlength{\glspagelistwidth}{0.1\hsize}
8522 }{}

```

`super` The super glossary style uses the supertabular environment (it uses lengths defined in the package.)

```

8523 \newglossarystyle{super}{%

```

Put the glossary in a supertabular environment with two columns and no head or tail:

```

8524   \renewenvironment{theglossary}%
8525     {\tablehead{}\tabletail{}}%
8526     \begin{supertabular}{lp{\glsdescwidth}}}%
8527     {\end{supertabular}}}%

```

Do nothing at the start of the table:

```
8528 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8529 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```
8530 \renewcommand{\glossentry}[2]{%
8531   \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8532   \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline
8533 }%
```

Sub entries put in a row (no name, description and page list in second column):

```
8534 \renewcommand{\subglossentry}[3]{%
8535   &
8536   \glssubentryitem{##2}%
8537   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
8538   ##3\tabularnewline
8539 }%
```

Blank row between groups:

```
8540 \renewcommand*{\glsgroupskip}{%
8541   \ifglsnogroupskip\else & \tabularnewline\fi}%
8542 }
```

superborder The superborder style is like the above, but with horizontal and vertical lines:

```
8543 \newglossarystyle{superborder}{%
```

Base it on the `glostylesuper` style:

```
8544 \setglossarystyle{super}%
```

Put the glossary in a `supertabular` environment with two columns and a horizontal line in the head and tail:

```
8545 \renewenvironment{theglossary}%
8546   {\tablehead{\hline}\tabletail{\hline}%
8547   \begin{supertabular}{|l|p{\glstdescwidth}|}%
8548   {\end{supertabular}}%
8549 }
```

superheader The superheader style is like the super style, but with a header:

```
8550 \newglossarystyle{superheader}{%
```

Base it on the `glostylesuper` style:

```
8551 \setglossarystyle{super}%
```

Put the glossary in a `supertabular` environment with two columns, a header and no tail:

```
8552 \renewenvironment{theglossary}%
8553   {\tablehead{\bfseries \entryname &
8554   \bfseries\descriptionname\tabularnewline}%
8555   \tabletail{}}%
8556   \begin{supertabular}{lp{\glstdescwidth}}%
```

```

8557 {\end{supertabular}}}%
8558 }

```

superheaderborder The superheaderborder style is like the super style but with a header and border:

```

8559 \newglossarystyle{superheaderborder}{%
    Base it on the glostylesuper style:
8560 \setglossarystyle{super}%
    Put the glossary in a supertabular environment with two columns, a header and horizontal
    lines above and below the table:
8561 \renewenvironment{theglossary}%
8562 {\tablehead{\hline\bfseries \entryname &
8563 \bfseries \descriptionname\tabularnewline\hline}%
8564 \tabletail{\hline}
8565 \begin{supertabular}{|l|p{\glstdescwidth}|}%
8566 {\end{supertabular}}}%
8567 }

```

super3col The super3col style is like the super style, but with 3 columns:

```

8568 \newglossarystyle{super3col}{%
    Put the glossary in a supertabular environment with three columns and no head or tail:
8569 \renewenvironment{theglossary}%
8570 {\tablehead{}\tabletail{}}%
8571 \begin{supertabular}{|lp{\glstdescwidth}p{\glspagelistwidth}|}%
8572 {\end{supertabular}}}%
    Do nothing at the start of the table:
8573 \renewcommand*{\glossaryheader}{}%
    No group headings:
8574 \renewcommand*{\glsgroupheading}[1]{}%
    Main (level 0) entries on a row (name in first column, description in second column, page list
    in last column):
8575 \renewcommand{\glossentry}[2]{%
8576 \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8577 \glossentrydesc{##1} & ##2\tabularnewline
8578 }%
    Sub entries on a row (no name, description in second column, page list in last column):
8579 \renewcommand{\subglossentry}[3]{%
8580 &
8581 \glssubentryitem{##2}%
8582 \glstarget{##2}{\strut}\glossentrydesc{##2} &
8583 ##3\tabularnewline
8584 }%
    Blank row between groups:
8585 \renewcommand*{\glsgroupskip}{%
8586 \ifglsnogroupskip\else & \tabularnewline\fi}%
8587 }

```

super3colborder The **super3colborder** style is like the **super3col** style, but with a border:

```
8588 \newglossarystyle{super3colborder}{%
```

Base it on the **glostylesuper3col** style:

```
8589 \setglossarystyle{super3col}%
```

Put the glossary in a **supertabular** environment with three columns and a horizontal line in the head and tail:

```
8590 \renewenvironment{theglossary}%
```

```
8591 {\tablehead{\hline}\tabletail{\hline}%
```

```
8592 \begin{supertabular}{|l|p{\glsgdescwidth}|p{\glspagelistwidth}|}%
```

```
8593 {\end{supertabular}}%
```

```
8594 }
```

super3colheader The **super3colheader** style is like the **super3col** style but with a header row:

```
8595 \newglossarystyle{super3colheader}{%
```

Base it on the **glostylesuper3col** style:

```
8596 \setglossarystyle{super3col}%
```

Put the glossary in a **supertabular** environment with three columns, a header and no tail:

```
8597 \renewenvironment{theglossary}%
```

```
8598 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
```

```
8599 \bfseries\pagelistname\tabularnewline}\tabletail{}}%
```

```
8600 \begin{supertabular}{lp{\glsgdescwidth}p{\glspagelistwidth}}%
```

```
8601 {\end{supertabular}}%
```

```
8602 }
```

colheaderborder The **super3colheaderborder** style is like the **super3col** style but with a header and border:

```
8603 \newglossarystyle{super3colheaderborder}{%
```

Base it on the **glostylesuper3colborder** style:

```
8604 \setglossarystyle{super3colborder}%
```

Put the glossary in a **supertabular** environment with three columns, a header with horizontal lines and a horizontal line in the tail:

```
8605 \renewenvironment{theglossary}%
```

```
8606 {\tablehead{\hline
```

```
8607 \bfseries\entryname&\bfseries\descriptionname&
```

```
8608 \bfseries\pagelistname\tabularnewline\hline}%
```

```
8609 \tabletail{\hline}%
```

```
8610 \begin{supertabular}{|l|p{\glsgdescwidth}|p{\glspagelistwidth}|}%
```

```
8611 {\end{supertabular}}%
```

```
8612 }
```

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.

```
8613 \newglossarystyle{super4col}{%
```

Put the glossary in a supertabular environment with four columns and no head or tail:

```
8614 \renewenvironment{theglossary}%
8615 {\tablehead{}\tabletail{}}%
8616 \begin{supertabular}{1111}}{%
8617 \end{supertabular}}%
```

Do nothing at the start of the table:

```
8618 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8619 \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:

```
8620 \renewcommand{\glossentry}[2]{%
8621 \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8622 \glossentrydesc{##1} &
8623 \glossentrysymbol{##1} & ##2\tabularnewline
8624 }%
```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```
8625 \renewcommand{\subglossentry}[3]{%
8626 &
8627 \glssubentryitem{##2}%
8628 \glstarget{##2}{\strut}\glossentrydesc{##2} &
8629 \glossentrysymbol{##2} & ##3\tabularnewline
8630 }%
```

Blank row between groups:

```
8631 \renewcommand*{\glsgroupskip}{%
8632 \ifglsgroupskip\else & & \tabularnewline\fi}%
8633 }
```

super4colheader The super4colheader style is like the super4col but with a header row.

```
8634 \newglossarystyle{super4colheader}{%
```

Base it on the glostylessuper4col style:

```
8635 \setglossarystyle{super4col}%
```

Put the glossary in a supertabular environment with four columns, a header and no tail:

```
8636 \renewenvironment{theglossary}%
8637 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8638 \bfseries\symbolname &
8639 \bfseries\pagelistname\tabularnewline}%
8640 \tabletail{}}%
8641 \begin{supertabular}{1111}}{%
8642 {\end{supertabular}}}%
8643 }
```

super4colborder The super4colborder style is like the super4col but with a border.

```
8644 \newglossarystyle{super4colborder}{%
```


Base it on the `glostylesuper4col` style:

```
8645 \setglossarystyle{super4col}%
```

Put the glossary in a `supertabular` environment with four columns and a horizontal line in the head and tail:

```
8646 \renewenvironment{theglossary}%  
8647 {\tablehead{\hline}\tabletail{\hline}%  
8648 \begin{supertabular}{|l|l|l|l|}%  
8649 {\end{supertabular}}%  
8650 }
```

`colheaderborder` The `super4colheaderborder` style is like the `super4col` but with a header and border.

```
8651 \newglossarystyle{super4colheaderborder}{%
```

Base it on the `glostylesuper4col` style:

```
8652 \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:

```
8653 \renewenvironment{theglossary}%  
8654 {\tablehead{\hline\bfseries\entryname&\bfseries\descriptionname&  
8655 \bfseries\symbolname &  
8656 \bfseries\pagelistname\tabularnewline\hline}%  
8657 \tabletail{\hline}%  
8658 \begin{supertabular}{|l|l|l|l|}%  
8659 {\end{supertabular}}%  
8660 }
```

`altsuper4col` The `altsuper4col` glossary style is like `super4col` but has provision for multiline descriptions.

```
8661 \newglossarystyle{altsuper4col}{%
```

Base it on the `glostylesuper4col` style:

```
8662 \setglossarystyle{super4col}%
```

Put the glossary in a `supertabular` environment with four columns and no head or tail:

```
8663 \renewenvironment{theglossary}%  
8664 {\tablehead{}\tabletail{}}%  
8665 \begin{supertabular}{lp{\glsgdescwidth}lp{\glspagelistwidth}}%  
8666 {\end{supertabular}}%  
8667 }
```

`super4colheader` The `altsuper4colheader` style is like the `altsuper4col` but with a header row.

```
8668 \newglossarystyle{altsuper4colheader}{%
```

Base it on the `glostylesuper4colheader` style:

```
8669 \setglossarystyle{super4colheader}%
```

Put the glossary in a `supertabular` environment with four columns, a header and no tail:

```
8670 \renewenvironment{theglossary}%  
8671 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&  
8672 \bfseries\symbolname &
```

```

8673      \bfseries\pagelistname\tabularnewline}\tabletail{}}%
8674      \begin{supertabular}{lp{\glsgdescwidth}lp{\glspagelistwidth}}}%
8675      {\end{supertabular}}}%
8676 }

```

super4colborder The `altsuper4colborder` style is like the `altsuper4col` but with a border.

```

8677 \newglossarystyle{altsuper4colborder}{%
      Base it on the glostylesuper4colborder style:
8678   \setglossarystyle{super4colborder}%
      Put the glossary in a supertabular environment with four columns and a horizontal line in the
      head and tail:
8679   \renewenvironment{theglossary}%
8680     {\tablehead{\hline}\tabletail{\hline}%
8681     \begin{supertabular}%
8682       {lllp{\glsgdescwidth}lllp{\glspagelistwidth}}}%
8683     {\end{supertabular}}}%
8684 }

```

colheaderborder The `altsuper4colheaderborder` style is like the `altsuper4col` but with a header and border.

```

8685 \newglossarystyle{altsuper4colheaderborder}{%
      Base it on the glostylesuper4colheaderborder style:
8686   \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:
8687   \renewenvironment{theglossary}%
8688     {\tablehead{\hline
8689       \bfseries\entryname &
8690       \bfseries\descriptionname &
8691       \bfseries\symbolname &
8692       \bfseries\pagelistname\tabularnewline\hline}%
8693     \tabletail{\hline}%
8694     \begin{supertabular}%
8695       {lllp{\glsgdescwidth}lllp{\glspagelistwidth}}}%
8696     {\end{supertabular}}}%
8697 }

```

3.9 Glossary Styles using supertabular environment (glossary-superragged package)

The glossary styles defined in the package use the supertabular environment. These styles are like those provided by the package, except that the multiline columns have ragged right justification.

```

8698 \ProvidesPackage{glossary-superragged}[2016/04/19 v4.22 (NLCT)]

```

Requires the package:

```
8699 \RequirePackage{array}
```

Requires the package:

```
8700 \RequirePackage{supertabular}
```

`\glsdescwidth` This is a length that governs the width of the description column. This may already have been defined.

```
8701 \@ifundefined{glsdescwidth}{%
8702   \newlength{glsdescwidth
8703   \setlength{glsdescwidth}{0.6\hsize}
8704 }{}
```

`glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```
8705 \@ifundefined{glspagelistwidth}{%
8706   \newlength{glspagelistwidth
8707   \setlength{glspagelistwidth}{0.1\hsize}
8708 }{}
```

`superragged` The superragged glossary style uses the supertabular environment.

```
8709 \newglossarystyle{superragged}{%
```

Put the glossary in a supertabular environment with two columns and no head or tail:

```
8710 \renewenvironment{theglossary}{%
8711   {\tablehead{}\tabletail{}}%
8712   \begin{supertabular}{l>{\raggedright}p{glsdescwidth}}}%
8713   {\end{supertabular}}%
```

Do nothing at the start of the table:

```
8714 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8715 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```
8716 \renewcommand{\glossentry}[2]{%
8717   \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8718   \glossentrydesc{##1}\glspostdescription\space ##2%
8719   \tabularnewline
8720 }%
```

Sub entries put in a row (no name, description and page list in second column):

```
8721 \renewcommand{\subglossentry}[3]{%
8722   &
8723   \glssubentryitem{##2}%
8724   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
8725   ##3%
8726   \tabularnewline
8727 }%
```

Blank row between groups:

```
8728 \renewcommand*{\glsgroupskip}{\ifglsgroupskip\else & \tabularnewline\fi}%  
8729 }
```

`superraggedborder` The `superraggedborder` style is like the above, but with horizontal and vertical lines:

```
8730 \newglossarystyle{superraggedborder}{%
```

Base it on the `glostylesuperragged` style:

```
8731 \setglossarystyle{superragged}%
```

Put the glossary in a `supertabular` environment with two columns and a horizontal line in the head and tail:

```
8732 \renewenvironment{theglossary}%  
8733 {\tablehead{\hline}\tabletail{\hline}%  
8734 \begin{supertabular}{|l|>{\raggedright}p{\glsgdescwidth}|}}%  
8735 {\end{supertabular}}%  
8736 }
```

`superraggedheader` The `superraggedheader` style is like the `super` style, but with a header:

```
8737 \newglossarystyle{superraggedheader}{%
```

Base it on the `glostylesuperragged` style:

```
8738 \setglossarystyle{superragged}%
```

Put the glossary in a `supertabular` environment with two columns, a header and no tail:

```
8739 \renewenvironment{theglossary}%  
8740 {\tablehead{\bfseries \entryname & \bfseries \descriptionname  
8741 \tabularnewline}%  
8742 \tabletail{}}%  
8743 \begin{supertabular}{|l>{\raggedright}p{\glsgdescwidth}|}}%  
8744 {\end{supertabular}}%  
8745 }
```

`superraggedheaderborder` The `superraggedheaderborder` style is like the `superragged` style but with a header and border:

```
8746 \newglossarystyle{superraggedheaderborder}{%
```

Base it on the `glostylesuper` style:

```
8747 \setglossarystyle{superragged}%
```

Put the glossary in a `supertabular` environment with two columns, a header and horizontal lines above and below the table:

```
8748 \renewenvironment{theglossary}%  
8749 {\tablehead{\hline\bfseries \entryname &  
8750 \bfseries \descriptionname\hline}%  
8751 \tabletail{\hline}  
8752 \begin{supertabular}{|l|>{\raggedright}p{\glsgdescwidth}|}}%  
8753 {\end{supertabular}}%  
8754 }
```

`superragged3col` The `superragged3col` style is like the `superragged` style, but with 3 columns:

```
8755 \newglossarystyle{superragged3col}{%
```

Put the glossary in a supertabular environment with three columns and no head or tail:

```
8756 \renewenvironment{theglossary}%
8757   {\tablehead{}\tabletail{}}%
8758   \begin{supertabular}{l>{\raggedright}p{\glstdescwidth}%
8759     >{\raggedright}p{\glspagelistwidth}}}%
8760   {\end{supertabular}}%
```

Do nothing at the start of the table:

```
8761 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8762 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
8763 \renewcommand{\glossentry}[2]{%
8764   \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8765   \glossentrydesc{##1} &
8766   ##2\tabularnewline
8767 }%
```

Sub entries on a row (no name, description in second column, page list in last column):

```
8768 \renewcommand{\subglossentry}[3]{%
8769   &
8770   \glssubentryitem{##2}%
8771   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8772   ##3\tabularnewline
8773 }%
```

Blank row between groups:

```
8774 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else & &\tabularnewline\fi}%
8775 }
```

agged3colborder The `superragged3colborder` style is like the `superragged3col` style, but with a border:

```
8776 \newglossarystyle{superragged3colborder}{%
```

Base it on the `glostylessuperragged3col` style:

```
8777 \setglossarystyle{superragged3col}%
```

Put the glossary in a supertabular environment with three columns and a horizontal line in the head and tail:

```
8778 \renewenvironment{theglossary}%
8779   {\tablehead{\hline}\tabletail{\hline}%
8780   \begin{supertabular}{|l|>{\raggedright}p{\glstdescwidth}|%
8781     >{\raggedright}p{\glspagelistwidth}|}%
8782   {\end{supertabular}}%
8783 }
```

agged3colheader The `superragged3colheader` style is like the `superragged3col` style but with a header row:

```
8784 \newglossarystyle{superragged3colheader}{%
```

Base it on the `glostylesuperragged3col` style:

```
8785 \setglossarystyle{superragged3col}%
```

Put the glossary in a `supertabular` environment with three columns, a header and no tail:

```
8786 \renewenvironment{theglossary}%
8787 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8788 \bfseries\pagelistname\tabularnewline}\tabletail{}}%
8789 \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}%
8790 >{\raggedright}p{\glspagelistwidth}}}%
8791 {\end{supertabular}}%
8792 }
```

`colheaderborder` The `superragged3colheaderborder` style is like the `superragged3col` style but with a header and border:

```
8793 \newglossarystyle{superragged3colheaderborder}{%
```

Base it on the `glostylesuperragged3colborder` style:

```
8794 \setglossarystyle{superragged3colborder}%
```

Put the glossary in a `supertabular` environment with three columns, a header with horizontal lines and a horizontal line in the tail:

```
8795 \renewenvironment{theglossary}%
8796 {\tablehead{\hline
8797 \bfseries\entryname&\bfseries\descriptionname&
8798 \bfseries\pagelistname\tabularnewline\hline}%
8799 \tabletail{\hline}%
8800 \begin{supertabular}{|l|>{\raggedright}p{\glsgdescwidth}|%
8801 >{\raggedright}p{\glspagelistwidth}|}%
8802 {\end{supertabular}}%
8803 }
```

`superragged4col` The `altsuperragged4col` glossary style is like `altsuper4col` style in the package but uses ragged right formatting in the description and page list columns.

```
8804 \newglossarystyle{altsuperragged4col}{%
```

Put the glossary in a `supertabular` environment with four columns and no head or tail:

```
8805 \renewenvironment{theglossary}%
8806 {\tablehead{}\tabletail{}}%
8807 \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}l%
8808 >{\raggedright}p{\glspagelistwidth}}}%
8809 {\end{supertabular}}%
```

Do nothing at the start of the table:

```
8810 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8811 \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:

```
8812 \renewcommand{\glossentry}[2]{}%
```

```

8813 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8814 \glossentrydesc{##1} &
8815 \glossentrysymbol{##1} & ##2\tabularnewline
8816 }%

```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```

8817 \renewcommand{\subglossentry}[3]{%
8818 &
8819 \glssubentryitem{##2}%
8820 \glstarget{##2}{\strut}\glossentrydesc{##2} &
8821 \glossentrysymbol{##2} & ##3\tabularnewline
8822 }%

```

Blank row between groups:

```

8823 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else & & \tabularnewline\fi}%
8824 }

```

ragged4colheader The `altsuperragged4colheader` style is like the `altsuperragged4col` style but with a header row.

```

8825 \newglossarystyle{altsuperragged4colheader}{%

```

Base it on the `glostylealtsuperragged4col` style:

```

8826 \setglossarystyle{altsuperragged4col}%

```

Put the glossary in a `supertabular` environment with four columns, a header and no tail:

```

8827 \renewenvironment{theglossary}%
8828 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8829 \bfseries\symbolname &
8830 \bfseries\pagelistname\tabularnewline}\tabletail{}}%
8831 \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}l%
8832 >{\raggedright}p{\glspagelistwidth}}}%
8833 {\end{supertabular}}%
8834 }

```

ragged4colborder The `altsuperragged4colborder` style is like the `altsuperragged4col` style but with a border.

```

8835 \newglossarystyle{altsuperragged4colborder}{%

```

Base it on the `glostylealtsuperragged4col` style:

```

8836 \setglossarystyle{altsuper4col}%

```

Put the glossary in a `supertabular` environment with four columns and a horizontal line in the head and tail:

```

8837 \renewenvironment{theglossary}%
8838 {\tablehead{\hline}\tabletail{\hline}%
8839 \begin{supertabular}%
8840 {l|>{\raggedright}p{\glsdescwidth}l|}%
8841 >{\raggedright}p{\glspagelistwidth}||}%
8842 {\end{supertabular}}%
8843 }

```

`colheaderborder` The `altsuperragged4colheaderborder` style is like the `altsuperragged4col` style but with a header and border.

```
8844 \newglossarystyle{altsuperragged4colheaderborder}{%
```

Base it on the `glostylealtsuperragged4col` style:

```
8845 \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:

```
8846 \renewenvironment{theglossary}{%
8847   {\tablehead{\hline
8848     \bfseries\entryname &
8849     \bfseries\descriptionname &
8850     \bfseries\symbolname &
8851     \bfseries\pagelistname\tabularnewline\hline}%
8852   \tabletail{\hline}%
8853   \begin{supertabular}%
8854     {ll|>{\raggedright}p{\glstdescwidth}|l|}%
8855     >{\raggedright}p{\glspagelistwidth}|}}%
8856   {\end{supertabular}}}%
8857 }
```

3.10 Tree Styles (`glossary-tree.sty`)

The style file defines glossary styles that have a tree-like structure. These are designed for hierarchical glossaries.

```
8858 \ProvidesPackage{glossary-tree}[2016/04/19 v4.22 (NLCT)]
```

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```
8859 \providecommand{\indexspace}{%
8860   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
8861 }
```

`\glstreenamefmt` Format used to display the name in the tree styles. (This may be counteracted by `\glstnamefont`.) This command was previously also used to format the group headings.

```
8862 \newcommand*{\glstreenamefmt}[1]{\textbf{#1}}
```

`\glstreegroupheaderfmt` Format used to display the group header in the tree styles. Before v4.22, `\glstreenamefmt` was used for the group header, so the default definition uses that to help maintain backward-compatibility, since in previous versions redefining `\glstreenamefmt` would've also affected the group headings.

```
8863 \newcommand*{\glstreegroupheaderfmt}[1]{\glstreenamefmt{#1}}
```

`\glstreenavigationfmt` Format used to display the navigation header in the tree styles.

```
8864 \newcommand*{\glstreenavigationfmt}[1]{\glstreenamefmt{#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.

```
8865 \newglossarystyle{index}{%
```

Set the paragraph indentation and skip and define `\item` to be the same as that used by theindex:

```
8866 \renewenvironment{theglossary}%
8867 {\setlength{\parindent}{0pt}%
8868 \setlength{\parskip}{0pt plus 0.3pt}%
8869 \let\item\@idxitem}%
```

```
8870 {\par}%
```

Do nothing at the start of the environment:

```
8871 \renewcommand*{\glossaryheader}{}%
```

No group headers:

```
8872 \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.

```
8873 \renewcommand*{\glossentry}[2]{%
8874 \item\glstentryitem{##1}\glstreenamfmt{\glstarget{##1}{\glossentryname{##1}}}%
8875 \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
8876 \space \glossentrydesc{##1}\glspostdescription\space ##2%
8877 }%
```

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.

```
8878 \renewcommand{\subglossentry}[3]{%
8879 \ifcase##1\relax
8880 % level 0
8881 \item
8882 \or
8883 % level 1
8884 \subitem
8885 \glssubentryitem{##2}%
8886 \else
8887 % all other levels
8888 \subsubitem
8889 \fi
8890 \glstreenamfmt{\glstarget{##2}{\glossentryname{##2}}}%
8891 \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
8892 \space \glossentrydesc{##2}\glspostdescription\space ##3%
8893 }%
```

Vertical gap between groups is the same as that used by indices:

```
8894 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}}
```

indexgroup The indexgroup style is like the index style but has headings.

```
8895 \newglossarystyle{indexgroup}{%
```

Base it on the glostyleindex style:

```
8896 \setglossarystyle{index}%
```

Add a heading for each group. This puts the group's title in bold followed by a vertical gap.

```
8897 \renewcommand*{\glsgroupheading}[1]{%
8898 \item\glstreegroupheaderfmt{\glsgrouptitle{##1}}%
8899 \indexspace
8900 }%
8901 }
```

indexhypergroup The indexhypergroup style is like the indexgroup style but has hyper navigation.

```
8902 \newglossarystyle{indexhypergroup}{%
```

Base it on the glostyleindex style:

```
8903 \setglossarystyle{index}%
```

Put navigation links to the groups at the start of the glossary:

```
8904 \renewcommand*{\glossaryheader}{%
8905 \item\glstreenavigationfmt{\glsnavigation}\indexspace}%
```

Add a heading for each group (with a target). The group's title is in bold followed by a vertical gap.

```
8906 \renewcommand*{\glsgroupheading}[1]{%
8907 \item\glstreegroupheaderfmt
8908 {\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}%
8909 \indexspace}%
8910 }
```

tree The tree glossary style is similar in style to the index style, but can have arbitrary levels.

```
8911 \newglossarystyle{tree}{%
```

Set the paragraph indentation and skip:

```
8912 \renewenvironment{theglossary}%
8913 {\setlength{\parindent}{0pt}%
8914 \setlength{\parskip}{0pt plus 0.3pt}}%
8915 {}%
```

Do nothing at the start of the theglossary environment:

```
8916 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8917 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries: name in bold, followed by symbol in brackets (if it exists), the description and the page list:

```

8918 \renewcommand{\glossentry}[2]{%
8919   \hangindent0pt\relax
8920   \parindent0pt\relax
8921   \glstentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
8922   \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
8923   \space\glossentrydesc{##1}\glspostdescription\space##2\par
8924 }%
```

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.

```

8925 \renewcommand{\subglossentry}[3]{%
8926   \hangindent##1\glstreeindent\relax
8927   \parindent##1\glstreeindent\relax
8928   \ifnum##1=1\relax
8929     \glssubentryitem{##2}%
8930   \fi
8931   \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
8932   \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
8933   \space\glossentrydesc{##2}\glspostdescription\space ##3\par
8934 }%
```

Vertical gap between groups is the same as that used by indices:

```

8935 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}}
```

treegroup Like the tree style but the glossary groups have headings.

```

8936 \newglossarystyle{treegroup}{%
```

Base it on the `glostyletree` style:

```

8937 \setglossarystyle{tree}%
```

Each group has a heading (in bold) followed by a vertical gap):

```

8938 \renewcommand{\glsgroupheading}[1]{\par
8939   \noindent\glstreegroupheaderfmt{\glsgrouptitle{##1}}\par
8940   \indexspace}%
8941 }
```

treehypergroup The `treehypergroup` style is like the `treegroup` style, but has a set of links to the groups at the start of the glossary.

```

8942 \newglossarystyle{treehypergroup}{%
```

Base it on the `glostyletree` style:

```

8943 \setglossarystyle{tree}%
```

Put navigation links to the groups at the start of the `theglossary` environment:

```

8944 \renewcommand*{\glossaryheader}{%
8945   \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```

8946 \renewcommand*{\glsgroupheading}[1]{%
```

```

8947 \par\noindent
8948 \glstreegroupheaderfmt
8949 {\glsnahypertarget{##1}{\glsgrouptitle{##1}}}\par
8950 \indexspace}%
8951 }

```

`\glstreeindent` Length governing left indent for each level of the tree style.

```

8952 \newlength\glstreeindent
8953 \setlength{\glstreeindent}{10pt}

```

`treenoname` The `treenoname` glossary style is like the tree style, but doesn't print the name or symbol for sub-levels.

```

8954 \newglossarystyle{treenoname}{%

```

Set the paragraph indentation and skip:

```

8955 \renewenvironment{theglossary}%
8956 {\setlength{\parindent}{0pt}%
8957 \setlength{\parskip}{0pt plus 0.3pt}}%
8958 {}%

```

No header:

```

8959 \renewcommand*{\glossaryheader}{}%

```

No group headings:

```

8960 \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.

```

8961 \renewcommand{\glossentry}[2]{%
8962 \hangindent0pt\relax
8963 \parindent0pt\relax
8964 \glssentryitem{##1}\glstreenamfmt{\glstarget{##1}{\glossentryname{##1}}}%
8965 \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
8966 \space\glossentrydesc{##1}\glspostdescription\space##2\par
8967 }%

```

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.

```

8968 \renewcommand{\subglossentry}[3]{%
8969 \hangindent##1\glstreeindent\relax
8970 \parindent##1\glstreeindent\relax
8971 \ifnum##1=1\relax
8972 \glssubentryitem{##2}%
8973 \fi
8974 \glstarget{##2}{\strut}%
8975 \glossentrydesc{##2}\glspostdescription\space##3\par
8976 }%

```

Vertical gap between groups is the same as that used by indices:

```

8977 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
8978 }

```

treenonamegroup Like the `treenoname` style but the glossary groups have headings.

```
8979 \newglossarystyle{treenonamegroup}{%
      Base it on the glostyletreenoname style:
8980   \setglossarystyle{treenoname}%
      Give each group a heading:
8981   \renewcommand{\glsgroupheading}[1]{\par
8982     \noindent\glstreegroupheaderfmt
8983     {\glsgrouptitle{##1}}\par\indexspace}%
8984 }
```

onamehypergroup The `treenonamehypergroup` style is like the `treenonamegroup` style, but has a set of links to the groups at the start of the glossary.

```
8985 \newglossarystyle{treenonamehypergroup}{%
      Base it on the glostyletreenoname style:
8986   \setglossarystyle{treenoname}%
      Put navigation links to the groups at the start of the theglossary environment:
8987   \renewcommand*{\glossaryheader}{%
8988     \par\noindent\glstreenavigationfmt{\glstravigation}\par\indexspace}%
      Each group has a heading (in bold with a target) followed by a vertical gap):
8989   \renewcommand*{\glsgroupheading}[1]{%
8990     \par\noindent
8991     \glstreegroupheaderfmt
8992     {\glstrnavhypertarget{##1}{\glsgrouptitle{##1}}}\par
8993     \indexspace}%
8994 }
```

esttoplevelname Find the widest name over all parentless entries in the given glossary or glossaries.

```
8995 \newrobustcmd*{\glstrfindwidesttoplevelname}[1][\@glo@types]{%
8996   \dimen@=0pt\relax
8997   \gls@tmplen=0pt\relax
8998   \forallglossaries[#1]{\@gls@type}%
8999   {%
9000     \forglstentries[\@gls@type]{\@glo@label}%
9001     {%
9002       \ifglshasparent{\@glo@label}%
9003       }%
9004     {%
9005       \settowidth{\dimen@}%
9006       {\glstreenamfmt{\glstentryname{\@glo@label}}}%
9007       \ifdim\dimen@>\gls@tmplen
9008         \gls@tmplen=\dimen@
9009       \letcs{\@glswidestname}{glo@\glstetoklabel{\@glo@label}@name}%
9010       \fi
9011     }%
9012   }%
9013 }%
9014 }
```

`\glsetwidest` `\glsetwidest` [*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.

```

9015 \newcommand*{\glsetwidest}[2][0]{%
9016   \expandafter\def\csname @glswidestname\romannumeral#1\endcsname{%
9017     #2}%
9018 }

```

`\@glswidestname` Initialise `\@glswidestname`.

```

9019 \newcommand*{\@glswidestname}{}

```

`\glstreenamebox` Used by the `alttree` style to create the box for the name and associated information.

```

9020 \newcommand*{\glstreenamebox}[2]{%
9021   \makebox[#1][l]{#2}%
9022 }

```

`alttree` The `alttree` glossary style is similar in style to the `tree` style, but the indentation is obtained from the width of `\@glswidestname` which is set using `\glsetwidest`.

```

9023 \newglossarystyle{alttree}{%

```

Redefine the glossary environment.

```

9024   \renewenvironment{theglossary}%
9025     {\def\@gls@prevlevel{-1}%
9026       \mbox{}\par}%
9027     {\par}%

```

Set the header and group headers to nothing.

```

9028   \renewcommand*{\glossaryheader}{}%
9029   \renewcommand*{\glsgroupheading}[1]{}%

```

Redefine the way that the level 0 entries are displayed.

```

9030   \renewcommand{\glosseentry}[2]{%
9031     \ifnum\@gls@prevlevel=0\relax
9032     \else

```

Find out how big the indentation should be by measuring the widest entry.

```

9033       \settowidth{\glstreeindent}{\glstreenamefmt{\@glswidestname\space}}%
9034       \fi

```

Set the hangindent and paragraph indent.

```

9035       \hangindent\glstreeindent
9036       \parindent\glstreeindent

```

Put the name to the left of the paragraph block.

```

9037       \makebox[0pt][r]{\glstreenamebox{\glstreeindent}{%
9038         \glsentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}}}%

```

If the symbol is missing, ignore it, otherwise put it in brackets.

```

9039       \ifglshassymbol{##1}{(\glossentrysymbol{##1})\space}{}%

```

Do the description followed by the description terminator and location list.

```

9040       \glossentrydesc{##1}\glspostdescription \space ##2\par

```

Set the previous level to 0.

```
9041 \def\@gls@prevlevel{0}%  
9042 }%
```

Redefine the way sub-entries are displayed.

```
9043 \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.

```
9044 \ifnum##1=1\relax  
9045 \glssubentryitem{##2}%  
9046 \fi
```

If the level hasn't changed, keep the same settings, otherwise adjust \glstreeindent accordingly.

```
9047 \ifnum\@gls@prevlevel=##1\relax  
9048 \else
```

Compute the widest entry for this level, or for level 0 if not defined for this level. Store in \gls@tmplen

```
9049 \@ifundefined{@glswidestname\romannumeral##1}{%  
9050 \settowidth{\gls@tmplen}{\glstreenamfmt{\@glswidestname\space}}}%  
9051 \settowidth{\gls@tmplen}{\glstreenamfmt{%  
9052 \csname @glswidestname\romannumeral##1\endcsname\space}}}%
```

Determine if going up or down a level

```
9053 \ifnum\@gls@prevlevel<##1\relax
```

Depth has increased, so add the width of the widest entry to \glstreeindent.

```
9054 \setlength\glstreeindent\gls@tmplen  
9055 \addtolength\glstreeindent\parindent  
9056 \parindent\glstreeindent  
9057 \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.

```
9058 \@ifundefined{@glswidestname\romannumeral\@gls@prevlevel}{%  
9059 \settowidth{\glstreeindent}{\glstreenamfmt{%  
9060 \@glswidestname\space}}}%  
9061 \settowidth{\glstreeindent}{\glstreenamfmt{%  
9062 \csname @glswidestname\romannumeral\@gls@prevlevel  
9063 \endcsname\space}}}%
```

Subtract this length from the previous level's paragraph indent and set to \glstreeindent.

```
9064 \addtolength\parindent{-\glstreeindent}%  
9065 \setlength\glstreeindent\parindent  
9066 \fi  
9067 \fi
```

Set the hanging indentation.

```
9068 \hangindent\glstreeindent
```

Put the name to the left of the paragraph block

```
9069 \makebox[0pt][r]{\glstreenamebox{\gls@tmplen}{%  
9070 \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}}}%
```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
9071 \ifglshassymbol{##2}{(\glossentrysymbol{##2})\space}{}%
```

Do the description followed by the description terminator and location list.

```
9072 \glossentrydesc{##2}\glspostdescription\space ##3\par
```

Set the previous level macro to the current level.

```
9073 \def\@gls@prevlevel{##1}%  
9074 }%
```

Vertical gap between groups is the same as that used by indices:

```
9075 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%  
9076 }
```

alttreegroup Like the `alttree` style but the glossary groups have headings.

```
9077 \newglossarystyle{alttreegroup}{%
```

Base it on the `glostylealttree` style:

```
9078 \setglossarystyle{alttree}%
```

Give each group a heading.

```
9079 \renewcommand{\glsgroupheading}[1]{\par  
9080 \def\@gls@prevlevel{-1}%  
9081 \hangindent0pt\relax  
9082 \parindent0pt\relax  
9083 \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}}%  
9084 \par\indexspace}%  
9085 }
```

alttreehypergroup The `alttreehypergroup` style is like the `alttreegroup` style, but has a set of links to the groups at the start of the glossary.

```
9086 \newglossarystyle{alttreehypergroup}{%
```

Base it on the `glostylealttree` style:

```
9087 \setglossarystyle{alttree}%
```

Put the navigation links in the header

```
9088 \renewcommand*{\glossaryheader}{%  
9089 \par  
9090 \def\@gls@prevlevel{-1}%  
9091 \hangindent0pt\relax  
9092 \parindent0pt\relax  
9093 \glstreenavigationfmt{\glsnavigation}\par\indexspace}%
```

Put a `hypertarget` at the start of each group

```
9094 \renewcommand*{\glsgroupheading}[1]{%  
9095 \par  
9096 \def\@gls@prevlevel{-1}%
```



```
9097 \hangindent0pt\relax
9098 \parindent0pt\relax
9099 \glstreegroupheaderfmt
9100 {\glsnahypertarget{##1}{\glsgrouptitle{##1}}}\par
9101 \indexspace}}
```

4 Backwards Compatibility

4.1 glossaries-compatible-207

Provides compatibility with version 2.07 and below. This uses original glossaries xindy and makeindex formatting, so can be used with old documents that had customized style files, but hyperlinks may not work properly.

```
9102 \NeedsTeXFormat{LaTeX2e}
9103 \ProvidesPackage{glossaries-compatible-207}[2016/04/19 v4.22 (NLCT)]
```

AddXdyAttribute Adds an attribute in old format.

```
9104 \ifglxsindy
9105   \renewcommand*\GlsAddXdyAttribute[1]{%
9106     \edef\@xdyattributes{\@xdyattributes ^^J \string"#1\string"}%
9107     \expandafter\toks@\expandafter{\@xdylocref}%
9108     \edef\@xdylocref{\the\toks@ ^^J}%
9109     (markup-locref
9110 :open \string"\string~n\string\setentrycounter
9111   {\noexpand\glscounter}%
9112   \expandafter\string\csname#1\endcsname
9113   \expandafter\@gobble\string\{\string" ^^J
9114 :close \string"\expandafter\@gobble\string\}\string" ^^J
9115 :attr \string"#1\string"))}}
```

Only has an effect before `\writeist`:

```
9116 \fi
```

sAddXdyCounters

```
9117 \renewcommand*\GlsAddXdyCounters[1]{%
9118   \GlossariesWarning{\string\GlsAddXdyCounters\space not available
9119     in compatibility mode.}%
9120 }
```

Add predefined attributes

```
9121 \GlsAddXdyAttribute{glsnumberformat}
9122 \GlsAddXdyAttribute{textrm}
9123 \GlsAddXdyAttribute{textsf}
9124 \GlsAddXdyAttribute{texttt}
9125 \GlsAddXdyAttribute{textbf}
9126 \GlsAddXdyAttribute{textmd}
9127 \GlsAddXdyAttribute{textit}
9128 \GlsAddXdyAttribute{textup}
9129 \GlsAddXdyAttribute{textsl}
```

```

9130 \GlsAddXdyAttribute{textsc}
9131 \GlsAddXdyAttribute{emph}
9132 \GlsAddXdyAttribute{glshypernumber}
9133 \GlsAddXdyAttribute{hyperrm}
9134 \GlsAddXdyAttribute{hypersf}
9135 \GlsAddXdyAttribute{hypertt}
9136 \GlsAddXdyAttribute{hyperbf}
9137 \GlsAddXdyAttribute{hypermd}
9138 \GlsAddXdyAttribute{hyperit}
9139 \GlsAddXdyAttribute{hyperup}
9140 \GlsAddXdyAttribute{hypersl}
9141 \GlsAddXdyAttribute{hypersc}
9142 \GlsAddXdyAttribute{hyperemph}

```

sAddXdyLocation Restore v2.07 definition:

```

9143 \ifglxindy
9144 \renewcommand*{\GlsAddXdyLocation}[2]{%
9145   \edef\@xdyuserlocationdefs{%
9146     \@xdyuserlocationdefs ^^J%
9147     (define-location-class \string"#1\string"^^J\space\space
9148     \space(#2))
9149   }%
9150   \edef\@xdyuserlocationnames{%
9151     \@xdyuserlocationnames^^J\space\space\space
9152     \string"#1\string"}%
9153 }
9154 \fi

```

\@do@wrglossary

```

9155 \renewcommand{\@do@wrglossary}[1]{%
  Determine whether to use xindy or makeindex syntax
9156 \ifglxindy
  Need to determine if the formatting information starts with a ( or ) indicating a range.
9157 \expandafter\@glo@check@mkidxrangechar\@glsnumberformat\@nil
9158 \def\@glo@range{}%
9159 \expandafter\if\@glo@prefix(\relax
9160   \def\@glo@range{:open-range}%
9161 \else
9162   \expandafter\if\@glo@prefix)\relax
9163   \def\@glo@range{:close-range}%
9164 \fi
9165 \fi

  Get the location and escape any special characters
9166 \protected@edef\@glslocref{\theglentrycounter}%
9167 \@gls@checkmkidxchars\@glslocref

  Write to the glossary file using xindy syntax.
9168 \glossary[\csname glo@#1@type\endcsname]{%

```

```

9169 (indexentry :tkey (\csname glo@#1@index\endcsname)
9170 :locoref \string"\@glslocoref\string" %
9171 :attr \string"\@glo@suffix\string" \@glo@range
9172 )
9173 }%
9174 \else
    Convert the format information into the format required for makeindex
9175 \@set@glo@numformat\@glo@numfmt\@gls@counter\@glsnumberformat
    Write to the glossary file using makeindex syntax.
9176 \glossary[\csname glo@#1@type\endcsname]{%
9177 \string\glossaryentry{\csname glo@#1@index\endcsname
9178 \@gls@encapchar\@glo@numfmt}{\theglsentrycounter}}%
9179 \fi
9180 }

```

t@glo@numformat Only had 3 arguments in v2.07

```

9181 \def\@set@glo@numformat#1#2#3{%
9182 \expandafter\@glo@check@mkidxrangechar#3\@nil
9183 \protected@edef#1{%
9184 \@glo@prefix setentrycounter[]{#2}%
9185 \expandafter\string\csname\@glo@suffix\endcsname
9186 }%
9187 \@gls@checkmkidxchars#1%
9188 }

```

\writeist Redefine \writeist back to the way it was in v2.07, but change \istfile to \glswrite.

```

9189 \ifglxsindy
9190 \def\writeist{%
9191 \openout\glswrite=\istfilename
9192 \write\glswrite{;; xindy style file created by the glossaries
9193 package in compatible-2.07 mode}%
9194 \write\glswrite{;; for document '\jobname' on
9195 \the\year-\the\month-\the\day}%
9196 \write\glswrite{^^J; required styles^^J}
9197 \@for\@xdystyle:=\@xdyrequiredstyles\do{%
9198 \ifx\@xdystyle\@empty
9199 \else
9200 \protected@write\glswrite{{(require
9201 \string"\@xdystyle.xdy\string")}}%
9202 \fi
9203 }%
9204 \write\glswrite{^^J%
9205 ; list of allowed attributes (number formats)^^J}%
9206 \write\glswrite{(define-attributes ((\@xdyattributes)))}%
9207 \write\glswrite{^^J; user defined alphabets^^J}%
9208 \write\glswrite{\@xdyuseralphabets}%
9209 \write\glswrite{^^J; location class definitions^^J}%
9210 \protected@edef\@gls@roman{\@roman{0}\string"

```

```

9211     \string"roman-numbers-lowercase\string" :sep \string"}}%
9212 \@onelevel@sanitize\@gls@roman
9213 \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
9214     :sep \string"}}%
9215 \@onelevel@sanitize\@tmp
9216 \ifx\@tmp\@gls@roman
9217     \write\glswrite{(define-location-class
9218         \string"roman-page-numbers\string"^^J\space\space\space
9219         (\string"roman-numbers-lowercase\string")
9220         :min-range-length \@glsminrange)}}%
9221 \else
9222     \write\glswrite{(define-location-class
9223         \string"roman-page-numbers\string"^^J\space\space\space
9224         (:sep "\@gls@roman")
9225         :min-range-length \@glsminrange)}}%
9226 \fi
9227 \write\glswrite{(define-location-class
9228     \string"Roman-page-numbers\string"^^J\space\space\space
9229     (\string"roman-numbers-uppercase\string")
9230     :min-range-length \@glsminrange)}}%
9231 \write\glswrite{(define-location-class
9232     \string"arabic-page-numbers\string"^^J\space\space\space
9233     (\string"arabic-numbers\string")
9234     :min-range-length \@glsminrange)}}%
9235 \write\glswrite{(define-location-class
9236     \string"alpha-page-numbers\string"^^J\space\space\space
9237     (\string"alpha\string")
9238     :min-range-length \@glsminrange)}}%
9239 \write\glswrite{(define-location-class
9240     \string"Alpha-page-numbers\string"^^J\space\space\space
9241     (\string"ALPHA\string")
9242     :min-range-length \@glsminrange)}}%
9243 \write\glswrite{(define-location-class
9244     \string"Appendix-page-numbers\string"^^J\space\space\space
9245     (\string"ALPHA\string"
9246     :sep \string"\@glsAlphacompositor\string"
9247     \string"arabic-numbers\string")
9248     :min-range-length \@glsminrange)}}%
9249 \write\glswrite{(define-location-class
9250     \string"arabic-section-numbers\string"^^J\space\space\space
9251     (\string"arabic-numbers\string"
9252     :sep \string"\glscompositor\string"
9253     \string"arabic-numbers\string")
9254     :min-range-length \@glsminrange)}}%
9255 \write\glswrite{^^J; user defined location classes}%
9256 \write\glswrite{\@xdyuserlocationdefs}%
9257 \write\glswrite{^^J; define cross-reference class^^J}%
9258 \write\glswrite{(define-crossref-class \string"see\string"
9259     :unverified )}%

```

```

9260 \write\glswrite{(markup-crossref-list
9261 :class \string"see\string"^^J\space\space\space
9262 :open \string"\string\glsseeformat\string"
9263 :close \string"{}\string")}%
9264 \write\glswrite{^^J; define the order of the location classes}%
9265 \write\glswrite{(define-location-class-order
9266 (\@xdylocationclassorder))}%
9267 \write\glswrite{^^J; define the glossary markup^^J}%
9268 \write\glswrite{(markup-index^^J\space\space\space
9269 :open \string"\string
9270 \glossarysection[\string\glossarytoctitle]{\string
9271 \glossarytitle}\string\glossarypreamble\string~n\string\begin
9272 {theglossary}\string\glossaryheader\string~n\string" ^^J\space
9273 \space\space:close \string"\expandafter\@gobble
9274 \string%\string~n\string
9275 \end{theglossary}\string\glossarypostamble
9276 \string~n\string" ^^J\space\space\space
9277 :tree)}}%
9278 \write\glswrite{(markup-letter-group-list
9279 :sep \string"\string\glsgroupskip\string~n\string")}%
9280 \write\glswrite{(markup-indexentry
9281 :open \string"\string\relax \string\glsresetentrylist
9282 \string~n\string")}%
9283 \write\glswrite{(markup-locclass-list :open
9284 \string"\glsopenbrace\string\glossaryentrynumbers
9285 \glsopenbrace\string\relax\space \string"^^J\space\space\space
9286 :sep \string", \string"
9287 :close \string"\glsclosebrace\glsclosebrace\string")}%
9288 \write\glswrite{(markup-locref-list
9289 :sep \string"\string\delimN\space\string")}%
9290 \write\glswrite{(markup-range
9291 :sep \string"\string\delimR\space\string")}%
9292 \@onelevel@sanitize\gls@suffixF
9293 \@onelevel@sanitize\gls@suffixFF
9294 \ifx\gls@suffixF\@empty
9295 \else
9296 \write\glswrite{(markup-range
9297 :close "\gls@suffixF" :length 1 :ignore-end)}%
9298 \fi
9299 \ifx\gls@suffixFF\@empty
9300 \else
9301 \write\glswrite{(markup-range
9302 :close "\gls@suffixFF" :length 2 :ignore-end)}%
9303 \fi
9304 \write\glswrite{^^J; define format to use for locations^^J}%
9305 \write\glswrite{\@xdylocref}%
9306 \write\glswrite{^^J; define letter group list format^^J}%
9307 \write\glswrite{(markup-letter-group-list
9308 :sep \string"\string\glsgroupskip\string~n\string")}%

```

```

9309 \write\glswrite{^^J; letter group headings^^J}%
9310 \write\glswrite{(markup-letter-group
9311   :open-head \string"\string\glsgroupheading
9312   \glsoopenbrace\string"^^J\space\space\space
9313   :close-head \string"\glsclosebrace\string")}%
9314 \write\glswrite{^^J; additional letter groups^^J}%
9315 \write\glswrite{\@xdylettergroups}%
9316 \write\glswrite{^^J; additional sort rules^^J}
9317 \write\glswrite{\@xdysortrules}%
9318 \noist}
9319 \else
9320 \edef\@gls@actualchar{\string?}
9321 \edef\@gls@encapchar{\string|}
9322 \edef\@gls@levelchar{\string!}
9323 \edef\@gls@quotechar{\string"}
9324 \def\writeist{\relax
9325   \openout\glswrite=\istfilename
9326   \write\glswrite{\expandafter\@gobble\string\% makeindex style file
9327     created by the glossaries package}
9328   \write\glswrite{\expandafter\@gobble\string\% for document
9329     '\jobname' on \the\year-\the\month-\the\day}
9330   \write\glswrite{actual '\@gls@actualchar'}
9331   \write\glswrite{encap '\@gls@encapchar'}
9332   \write\glswrite{level '\@gls@levelchar'}
9333   \write\glswrite{quote '\@gls@quotechar'}
9334   \write\glswrite{keyword \string"\string\glossaryentry\string"}
9335   \write\glswrite{preamble \string"\string\glossarysection[\string
9336     \glossarytoctitle]{\string\glossarytitle}\string
9337     \glossarypreamble\string\n\string\begin{theglossary}\string
9338     \glossaryheader\string\n\string"}
9339   \write\glswrite{postamble \string"\string%\string\n\string
9340     \end{theglossary}\string\glossarypostamble\string\n
9341     \string"}
9342   \write\glswrite{group_skip \string"\string\glsgroupskip\string\n
9343     \string"}
9344   \write\glswrite{item_0 \string"\string%\string\n\string"}
9345   \write\glswrite{item_1 \string"\string%\string\n\string"}
9346   \write\glswrite{item_2 \string"\string%\string\n\string"}
9347   \write\glswrite{item_01 \string"\string%\string\n\string"}
9348   \write\glswrite{item_x1
9349     \string"\string\relax \string\glsresetentrylist\string\n
9350     \string"}
9351   \write\glswrite{item_12 \string"\string%\string\n\string"}
9352   \write\glswrite{item_x2
9353     \string"\string\relax \string\glsresetentrylist\string\n
9354     \string"}
9355   \write\glswrite{delim_0 \string"\string\{\string
9356     \glossaryentrynumbers\string\{\string\relax \string"}
9357   \write\glswrite{delim_1 \string"\string\{\string

```

```

9358     \glossaryentrynumbers\string\{\string\relax \string}
9359     \write\glswrite{delim_2 \string"\string\{\string
9360     \glossaryentrynumbers\string\{\string\relax \string}
9361     \write\glswrite{delim_t \string"\string\}\string\}\string}
9362     \write\glswrite{delim_n \string"\string\delimN \string}
9363     \write\glswrite{delim_r \string"\string\delimR \string}
9364     \write\glswrite{headings_flag 1}
9365     \write\glswrite{heading_prefix
9366         \string"\string\glsgroupheading\string\{\string}
9367     \write\glswrite{heading_suffix
9368         \string"\string\}\string\relax
9369         \string\glsgroupresetentrylist \string}
9370     \write\glswrite{symhead_positive \string"glssymbols\string}
9371     \write\glswrite{numhead_positive \string"glslnumbers\string}
9372     \write\glswrite{page_compositor \string"glscpositor\string}
9373     \@gls@escbsdq\gls@suffixF
9374     \@gls@escbsdq\gls@suffixFF
9375     \ifx\gls@suffixF\@empty
9376     \else
9377         \write\glswrite{suffix_2p \string"\gls@suffixF\string}
9378     \fi
9379     \ifx\gls@suffixFF\@empty
9380     \else
9381         \write\glswrite{suffix_3p \string"\gls@suffixFF\string}
9382     \fi
9383     \noist
9384 }
9385 \fi

```

\noist

```
9386 \renewcommand*{\noist}{\let\writeist\relax}
```

4.2 glossaries-compatible-307

```

9387 \NeedsTeXFormat{LaTeX2e}
9388 \ProvidesPackage{glossaries-compatible-307}[2016/04/19 v4.22 (NLCT)]

```

Compatibility macros for predefined glossary styles:

`\atglossarystyle` Defines a compatibility glossary style.

```

9389 \newcommand{\compatglossarystyle}[2]{%
9390   \ifcsundef{@glscompstyle@#1}%
9391   {%
9392     \csdef{@glscompstyle@#1}{#2}%
9393   }%
9394   {%
9395     \PackageError{glossaries}{Glossary compatibility style ‘#1’ is already defined}{}%
9396   }%
9397 }

```


Backward compatible inline style.

```
9398 \compatglossarystyle{inline}{%
9399   \renewcommand{\glossaryentryfield}[5]{%
9400     \glsinlinedopostchild
9401     \gls@inlinesep
9402     \def\glo@desc{##3}%
9403     \def\@no@post@desc{\nopostdesc}%
9404     \glsentryitem{##1}\glsinlinenameformat{##1}{##2}%
9405     \ifx\glo@desc\@no@post@desc
9406       \glsinlineemptydescformat{##4}{##5}%
9407     \else
9408       \ifstrepty{##3}%
9409         {\glsinlineemptydescformat{##4}{##5}}%
9410         {\glsinlinedescformat{##3}{##4}{##5}}%
9411     \fi
9412     \ifglshaschildren{##1}%
9413     {%
9414       \glsresetsubentrycounter
9415       \glsinlineparentchildseparator
9416       \def\gls@inlinesubsep{}%
9417       \def\gls@inlinepostchild{\glsinlinepostchild}%
9418     }%
9419   }%
9420   \def\gls@inlinesep{\glsinlineseparator}%
9421 }%
```

Sub-entries display description:

```
9422 \renewcommand{\glossarysubentryfield}[6]{%
9423   \gls@inlinesubsep%
9424   \glsinlinesubnameformat{##2}{##3}%
9425   \glssubentryitem{##2}\glsinlinesubdescformat{##4}{##5}{##6}%
9426   \def\gls@inlinesubsep{\glsinlinesubseparator}%
9427 }%
9428 }
```

Backward compatible list style.

```
9429 \compatglossarystyle{list}{%
9430   \renewcommand*{\glossaryentryfield}[5]{%
9431     \item[\glsentryitem{##1}\glstarget{##1}{##2}]
9432     ##3\glspostdescription\space ##5}%
9433 }
```

Sub-entries continue on the same line:

```
9433 \renewcommand*{\glossarysubentryfield}[6]{%
9434   \glssubentryitem{##2}%
9435   \glstarget{##2}{\strut}##4\glspostdescription\space ##6.}%
9436 }
```

Backward compatible listgroup style.

```
9437 \compatglossarystyle{listgroup}{%
9438   \csuse{@glscompstyle@list}%
9439 }%
```

Backward compatible listhypergroup style.

```
9440 \compatglossarystyle{listhypergroup}{%
9441   \csuse{@glscompstyle@list}%
9442 }%
```

Backward compatible altlist style.

```
9443 \compatglossarystyle{altlist}{%
9444   \renewcommand*{\glossaryentryfield}[5]{%
9445     \item[\glsentryitem{##1}\glstarget{##1}{##2}]%
9446     \mbox{}\par\nobreak\@afterheading
9447     ##3\glspostdescription\space ##5}%
9448   \renewcommand{\glossarysubentryfield}[6]{%
9449     \par
9450     \glssubentryitem{##2}%
9451     \glstarget{##2}{\strut}##4\glspostdescription\space ##6}%
9452 }%
```

Backward compatible altlistgroup style.

```
9453 \compatglossarystyle{altlistgroup}{%
9454   \csuse{@glscompstyle@altlist}%
9455 }%
```

Backward compatible altlisthypergroup style.

```
9456 \compatglossarystyle{altlisthypergroup}{%
9457   \csuse{@glscompstyle@altlist}%
9458 }%
```

Backward compatible listdotted style.

```
9459 \compatglossarystyle{listdotted}{%
9460   \renewcommand*{\glossaryentryfield}[5]{%
9461     \item[\makebox[\glslistdottedwidth][l]{%
9462       \glsentryitem{##1}\glstarget{##1}{##2}%
9463       \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##3}%
9464   \renewcommand*{\glossarysubentryfield}[6]{%
9465     \item[\makebox[\glslistdottedwidth][l]{%
9466       \glssubentryitem{##2}%
9467       \glstarget{##2}{##3}%
9468       \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##4}%
9469 }%
```

Backward compatible sublistdotted style.

```
9470 \compatglossarystyle{sublistdotted}{%
9471   \csuse{@glscompstyle@listdotted}%
9472   \renewcommand*{\glossaryentryfield}[5]{%
9473     \item[\glsentryitem{##1}\glstarget{##1}{##2}]}%
9474 }%
```

Backward compatible long style.

```
9475 \compatglossarystyle{long}{%
9476   \renewcommand*{\glossaryentryfield}[5]{%
9477     \glsentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
9478   \renewcommand*{\glossarysubentryfield}[6]{%
9479     \glssubentryitem{##2}\glstarget{##2}{##3} & ##4\glspostdescription\space ##5\\}%
9479 }%
```

```

9479      &
9480      \glssubentryitem{##2}%
9481      \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
9482 }%

```

Backward compatible longborder style.

```

9483 \compatglossarystyle{longborder}{%
9484 \csuse{@glscmpstyle@long}%
9485 }%

```

Backward compatible longheader style.

```

9486 \compatglossarystyle{longheader}{%
9487 \csuse{@glscmpstyle@long}%
9488 }%

```

Backward compatible longheaderborder style.

```

9489 \compatglossarystyle{longheaderborder}{%
9490 \csuse{@glscmpstyle@long}%
9491 }%

```

Backward compatible long3col style.

```

9492 \compatglossarystyle{long3col}{%
9493 \renewcommand*{\glossaryentryfield}[5]{%
9494 \glstarget{##1}{\strut}##4 & ##3 & ##5\\}%
9495 \renewcommand*{\glossarysubentryfield}[6]{%
9496 &
9497 \glssubentryitem{##2}%
9498 \glstarget{##2}{\strut}##4 & ##6\\}%
9499 }%

```

Backward compatible long3colborder style.

```

9500 \compatglossarystyle{long3colborder}{%
9501 \csuse{@glscmpstyle@long3col}%
9502 }%

```

Backward compatible long3colheader style.

```

9503 \compatglossarystyle{long3colheader}{%
9504 \csuse{@glscmpstyle@long3col}%
9505 }%

```

Backward compatible long3colheaderborder style.

```

9506 \compatglossarystyle{long3colheaderborder}{%
9507 \csuse{@glscmpstyle@long3col}%
9508 }%

```

Backward compatible long4col style.

```

9509 \compatglossarystyle{long4col}{%
9510 \renewcommand*{\glossaryentryfield}[5]{%
9511 \glstarget{##1}{\strut}##4 & ##3 & ##4 & ##5\\}%
9512 \renewcommand*{\glossarysubentryfield}[6]{%
9513 &
9514 \glssubentryitem{##2}%

```

9515 \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%
9516 }%

Backward compatible long4colheader style.

9517 \compatglossarystyle{long4colheader}{%
9518 \csuse{@glscompstyle@long4col}%
9519 }%

Backward compatible long4colborder style.

9520 \compatglossarystyle{long4colborder}{%
9521 \csuse{@glscompstyle@long4col}%
9522 }%

Backward compatible long4colheaderborder style.

9523 \compatglossarystyle{long4colheaderborder}{%
9524 \csuse{@glscompstyle@long4col}%
9525 }%

Backward compatible altlong4col style.

9526 \compatglossarystyle{altlong4col}{%
9527 \csuse{@glscompstyle@long4col}%
9528 }%

Backward compatible altlong4colheader style.

9529 \compatglossarystyle{altlong4colheader}{%
9530 \csuse{@glscompstyle@long4col}%
9531 }%

Backward compatible altlong4colborder style.

9532 \compatglossarystyle{altlong4colborder}{%
9533 \csuse{@glscompstyle@long4col}%
9534 }%

Backward compatible altlong4colheaderborder style.

9535 \compatglossarystyle{altlong4colheaderborder}{%
9536 \csuse{@glscompstyle@long4col}%
9537 }%

Backward compatible long style.

9538 \compatglossarystyle{longragged}{%
9539 \renewcommand*{\glossaryentryfield}[5]{%
9540 \glssentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
9541 \tabularnewline}%
9542 \renewcommand*{\glossarysubentryfield}[6]{%
9543 &
9544 \glssubentryitem{##2}%
9545 \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
9546 \tabularnewline}%
9547 }%

Backward compatible longraggedborder style.

9548 \compatglossarystyle{longraggedborder}{%
9549 \csuse{@glscompstyle@longragged}%
9550 }%

Backward compatible longraggedheader style.

```
9551 \compatglossarystyle{longraggedheader}{%  
9552 \csuse{@glscompstyle@longragged}%  
9553 }%
```

Backward compatible longraggedheaderborder style.

```
9554 \compatglossarystyle{longraggedheaderborder}{%  
9555 \csuse{@glscompstyle@longragged}%  
9556 }%
```

Backward compatible longragged3col style.

```
9557 \compatglossarystyle{longragged3col}{%  
9558 \renewcommand*{\glossaryentryfield}[5]{%  
9559 \glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%  
9560 \renewcommand*{\glossarysubentryfield}[6]{%  
9561 &  
9562 \glssubentryitem{##2}%  
9563 \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%  
9564 }%
```

Backward compatible longragged3colborder style.

```
9565 \compatglossarystyle{longragged3colborder}{%  
9566 \csuse{@glscompstyle@longragged3col}%  
9567 }%
```

Backward compatible longragged3colheader style.

```
9568 \compatglossarystyle{longragged3colheader}{%  
9569 \csuse{@glscompstyle@longragged3col}%  
9570 }%
```

Backward compatible longragged3colheaderborder style.

```
9571 \compatglossarystyle{longragged3colheaderborder}{%  
9572 \csuse{@glscompstyle@longragged3col}%  
9573 }%
```

Backward compatible altlongragged4col style.

```
9574 \compatglossarystyle{altlongragged4col}{%  
9575 \renewcommand*{\glossaryentryfield}[5]{%  
9576 \glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%  
9577 \renewcommand*{\glossarysubentryfield}[6]{%  
9578 &  
9579 \glssubentryitem{##2}%  
9580 \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%  
9581 }%
```

Backward compatible altlongragged4colheader style.

```
9582 \compatglossarystyle{altlongragged4colheader}{%  
9583 \csuse{@glscompstyle@altlong4col}%  
9584 }%
```

Backward compatible altlongragged4colborder style.

```
9585 \compatglossarystyle{altlongragged4colborder}{%
```

```

9586 \csuse{@glscompstyle@altlong4col}%
9587 }%

```

Backward compatible altlongragged4colheaderborder style.

```

9588 \compatglossarystyle{altlongragged4colheaderborder}{%
9589 \csuse{@glscompstyle@altlong4col}%
9590 }%

```

Backward compatible index style.

```

9591 \compatglossarystyle{index}{%
9592 \renewcommand*{\glossaryentryfield}[5]{%
9593 \item\glsentryitem{##1}\textbf{\glstarget{##1}{##2}}%
9594 \ifx\relax##4\relax
9595 \else
9596 \space{##4}%
9597 \fi
9598 \space ##3\glspostdescription \space ##5}%
9599 \renewcommand*{\glossarysubentryfield}[6]{%
9600 \ifcase##1\relax
9601 % level 0
9602 \item
9603 \or
9604 % level 1
9605 \subitem
9606 \glssubentryitem{##2}%
9607 \else
9608 % all other levels
9609 \subsubitem
9610 \fi
9611 \textbf{\glstarget{##2}{##3}}%
9612 \ifx\relax##5\relax
9613 \else
9614 \space{##5}%
9615 \fi
9616 \space##4\glspostdescription\space ##6}%
9617 }%

```

Backward compatible indexgroup style.

```

9618 \compatglossarystyle{indexgroup}{%
9619 \csuse{@glscompstyle@index}%
9620 }%

```

Backward compatible indexhypergroup style.

```

9621 \compatglossarystyle{indexhypergroup}{%
9622 \csuse{@glscompstyle@index}%
9623 }%

```

Backward compatible tree style.

```

9624 \compatglossarystyle{tree}{%
9625 \renewcommand{\glossaryentryfield}[5]{%
9626 \hangindent0pt\relax

```

```

9627 \parindent0pt\relax
9628 \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
9629 \ifx\relax##4\relax
9630 \else
9631 \space(##4)%
9632 \fi
9633 \space ##3\glspostdescription \space ##5\par}%
9634 \renewcommand{\glossarysubentryfield}[6]{%
9635 \hangindent##1\glstreeindent\relax
9636 \parindent##1\glstreeindent\relax
9637 \ifnum##1=1\relax
9638 \glssubentryitem{##2}%
9639 \fi
9640 \textbf{\glstarget{##2}{##3}}%
9641 \ifx\relax##5\relax
9642 \else
9643 \space(##5)%
9644 \fi
9645 \space##4\glspostdescription\space ##6\par}%
9646 }%

```

Backward compatible treegroup style.

```

9647 \compatglossarystyle{treegroup}{%
9648 \csuse{@glscmpstyle@tree}%
9649 }%

```

Backward compatible treehypergroup style.

```

9650 \compatglossarystyle{treehypergroup}{%
9651 \csuse{@glscmpstyle@tree}%
9652 }%

```

Backward compatible treenoname style.

```

9653 \compatglossarystyle{treenoname}{%
9654 \renewcommand{\glossaryentryfield}[5]{%
9655 \hangindent0pt\relax
9656 \parindent0pt\relax
9657 \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
9658 \ifx\relax##4\relax
9659 \else
9660 \space(##4)%
9661 \fi
9662 \space ##3\glspostdescription \space ##5\par}%
9663 \renewcommand{\glossarysubentryfield}[6]{%
9664 \hangindent##1\glstreeindent\relax
9665 \parindent##1\glstreeindent\relax
9666 \ifnum##1=1\relax
9667 \glssubentryitem{##2}%
9668 \fi
9669 \glstarget{##2}{\strut}%
9670 ##4\glspostdescription\space ##6\par}%
9671 }%

```

Backward compatible treenonamegroup style.

```
9672 \compatglossarystyle{treenonamegroup}{%
9673   \csuse{@glscompstyle@treenoname}%
9674 }%
```

Backward compatible treenonamehypergroup style.

```
9675 \compatglossarystyle{treenonamehypergroup}{%
9676   \csuse{@glscompstyle@treenoname}%
9677 }%
```

Backward compatible alttree style.

```
9678 \compatglossarystyle{alttree}{%
9679   \renewcommand{\glossaryentryfield}[5]{%
9680     \ifnum\@gls@prevlevel=0\relax
9681     \else
9682       \settowidth{\glstreeindent}{\textbf{\@glswidestname\space}}%
9683       \hangindent\glstreeindent
9684       \parindent\glstreeindent
9685     \fi
9686     \makebox[0pt][r]{\makebox[\glstreeindent][l]{%
9687       \glssentryitem{##1}\textbf{\glstarget{##1}{##2}}}%
9688     \ifx\relax##4\relax
9689     \else
9690       (##4)\space
9691     \fi
9692     ##3\glspostdescription \space ##5\par
9693     \def\@gls@prevlevel{0}%
9694   }%
9695   \renewcommand{\glossarysubentryfield}[6]{%
9696     \ifnum##1=1\relax
9697       \glssubentryitem{##2}%
9698     \fi
9699     \ifnum\@gls@prevlevel=##1\relax
9700     \else
9701       \@ifundefined{@glswidestname\romannumeral##1}{%
9702         \settowidth{\gls@tmplen}{\textbf{\@glswidestname\space}}{%
9703         \settowidth{\gls@tmplen}{\textbf{%
9704           \csname @glswidestname\romannumeral##1\endcsname\space}}}%
9705       \ifnum\@gls@prevlevel<##1\relax
9706         \setlength\glstreeindent\gls@tmplen
9707         \addtolength\glstreeindent\parindent
9708         \parindent\glstreeindent
9709       \else
9710         \@ifundefined{@glswidestname\romannumeral\@gls@prevlevel}{%
9711           \settowidth{\glstreeindent}{\textbf{%
9712             \@glswidestname\space}}{%
9713           \settowidth{\glstreeindent}{\textbf{%
9714             \csname @glswidestname\romannumeral\@gls@prevlevel
9715             \endcsname\space}}}%
9716           \addtolength\parindent{-\glstreeindent}%

```



```

9717         \setlength\glstreeindent\parindent
9718     \fi
9719 \fi
9720 \hangindent\glstreeindent
9721 \makebox[0pt][r]{\makebox[\glstemplen][l]{%
9722     \textbf{\glstarget{##2}{##3}}}%
9723 \ifx##5\relax\relax
9724 \else
9725     (##5)\space
9726 \fi
9727 ##4\glspostdescription\space ##6\par
9728 \def\@gls@prevlevel{##1}%
9729 }%
9730 }%

```

Backward compatible alttreegroup style.

```

9731 \compatglossarystyle{alttreegroup}{%
9732 \csuse{@glscompstyle@alttree}%
9733 }%

```

Backward compatible alttreehypergroup style.

```

9734 \compatglossarystyle{alttreehypergroup}{%
9735 \csuse{@glscompstyle@alttree}%
9736 }%

```

Backward compatible mcolindex style.

```

9737 \compatglossarystyle{mcolindex}{%
9738 \csuse{@glscompstyle@index}%
9739 }%

```

Backward compatible mcolindexgroup style.

```

9740 \compatglossarystyle{mcolindexgroup}{%
9741 \csuse{@glscompstyle@index}%
9742 }%

```

Backward compatible mcolindexhypergroup style.

```

9743 \compatglossarystyle{mcolindexhypergroup}{%
9744 \csuse{@glscompstyle@index}%
9745 }%

```

Backward compatible mcoltree style.

```

9746 \compatglossarystyle{mcoltree}{%
9747 \csuse{@glscompstyle@tree}%
9748 }%

```

Backward compatible mcoltreegroup style.

```

9749 \compatglossarystyle{mcolindextreegroup}{%
9750 \csuse{@glscompstyle@tree}%
9751 }%

```

Backward compatible mcoltreehypergroup style.

```

9752 \compatglossarystyle{mcolindextreehypergroup}{%

```

9753 \csuse{@glscompstyle@tree}%
 9754 }%

Backward compatible mcoltreenoname style.

9755 \compatglossarystyle{mcoltreenoname}{%
 9756 \csuse{@glscompstyle@tree}%
 9757 }%

Backward compatible mcoltreenonamegroup style.

9758 \compatglossarystyle{mcoltreenonamegroup}{%
 9759 \csuse{@glscompstyle@tree}%
 9760 }%

Backward compatible mcoltreenonamehypergroup style.

9761 \compatglossarystyle{mcoltreenonamehypergroup}{%
 9762 \csuse{@glscompstyle@tree}%
 9763 }%

Backward compatible mcolalmtree style.

9764 \compatglossarystyle{mcolalmtree}{%
 9765 \csuse{@glscompstyle@almtree}%
 9766 }%

Backward compatible mcolalmtreegroup style.

9767 \compatglossarystyle{mcolalmtreegroup}{%
 9768 \csuse{@glscompstyle@almtree}%
 9769 }%

Backward compatible mcolalmtreehypergroup style.

9770 \compatglossarystyle{mcolalmtreehypergroup}{%
 9771 \csuse{@glscompstyle@almtree}%
 9772 }%

Backward compatible superragged style.

9773 \compatglossarystyle{superragged}{%
 9774 \renewcommand*{\glossaryentryfield}[5]{%
 9775 \glsentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
 9776 \tabularnewline}%
 9777 \renewcommand*{\glossarysubentryfield}[6]{%
 9778 &
 9779 \glssubentryitem{##2}%
 9780 \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
 9781 \tabularnewline}%
 9782 }%

Backward compatible superraggedborder style.

9783 \compatglossarystyle{superraggedborder}{%
 9784 \csuse{@glscompstyle@superragged}%
 9785 }%

Backward compatible superraggedheader style.

9786 \compatglossarystyle{superraggedheader}{%
 9787 \csuse{@glscompstyle@superragged}%
 9788 }%

Backward compatible superraggedheaderborder style.

```
9789 \compatglossarystyle{superraggedheaderborder}{%
9790 \csuse{@glscompstyle@superragged}%
9791 }%
```

Backward compatible superragged3col style.

```
9792 \compatglossarystyle{superragged3col}{%
9793 \renewcommand*{\glossaryentryfield}[5]{%
9794 \glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%
9795 \renewcommand*{\glossarysubentryfield}[6]{%
9796 &
9797 \glssubentryitem{##2}%
9798 \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%
9799 }%
```

Backward compatible superragged3colborder style.

```
9800 \compatglossarystyle{superragged3colborder}{%
9801 \csuse{@glscompstyle@superragged3col}%
9802 }%
```

Backward compatible superragged3colheader style.

```
9803 \compatglossarystyle{superragged3colheader}{%
9804 \csuse{@glscompstyle@superragged3col}%
9805 }%
```

Backward compatible superragged3colheaderborder style.

```
9806 \compatglossarystyle{superragged3colheaderborder}{%
9807 \csuse{@glscompstyle@superragged3col}%
9808 }%
```

Backward compatible altsuperragged4col style.

```
9809 \compatglossarystyle{altsuperragged4col}{%
9810 \renewcommand*{\glossaryentryfield}[5]{%
9811 \glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%
9812 \renewcommand*{\glossarysubentryfield}[6]{%
9813 &
9814 \glssubentryitem{##2}%
9815 \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%
9816 }%
```

Backward compatible altsuperragged4colheader style.

```
9817 \compatglossarystyle{altsuperragged4colheader}{%
9818 \csuse{@glscompstyle@altsuperragged4col}%
9819 }%
```

Backward compatible altsuperragged4colborder style.

```
9820 \compatglossarystyle{altsuperragged4colborder}{%
9821 \csuse{@glscompstyle@altsuperragged4col}%
9822 }%
```

Backward compatible altsuperragged4colheaderborder style.

```
9823 \compatglossarystyle{altsuperragged4colheaderborder}{%
```

```

9824 \csuse{@glscompstyle@altsuperragged4col}%
9825 }%

```

Backward compatible super style.

```

9826 \compatglossarystyle{super}{%
9827   \renewcommand*{\glossaryentryfield}[5]{%
9828     \glentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
9829   \renewcommand*{\glossarysubentryfield}[6]{%
9830     &
9831     \glssubentryitem{##2}%
9832     \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
9833 }%

```

Backward compatible superborder style.

```

9834 \compatglossarystyle{superborder}{%
9835   \csuse{@glscompstyle@super}%
9836 }%

```

Backward compatible superheader style.

```

9837 \compatglossarystyle{superheader}{%
9838   \csuse{@glscompstyle@super}%
9839 }%

```

Backward compatible superheaderborder style.

```

9840 \compatglossarystyle{superheaderborder}{%
9841   \csuse{@glscompstyle@super}%
9842 }%

```

Backward compatible super3col style.

```

9843 \compatglossarystyle{super3col}{%
9844   \renewcommand*{\glossaryentryfield}[5]{%
9845     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\\}%
9846   \renewcommand*{\glossarysubentryfield}[6]{%
9847     &
9848     \glssubentryitem{##2}%
9849     \glstarget{##2}{\strut}##4 & ##6\\}%
9850 }%

```

Backward compatible super3colborder style.

```

9851 \compatglossarystyle{super3colborder}{%
9852   \csuse{@glscompstyle@super3col}%
9853 }%

```

Backward compatible super3colheader style.

```

9854 \compatglossarystyle{super3colheader}{%
9855   \csuse{@glscompstyle@super3col}%
9856 }%

```

Backward compatible super3colheaderborder style.

```

9857 \compatglossarystyle{super3colheaderborder}{%
9858   \csuse{@glscompstyle@super3col}%
9859 }%

```

Backward compatible super4col style.

```
9860 \compatglossarystyle{super4col}{%
9861   \renewcommand*{\glossaryentryfield}[5]{%
9862     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%
9863   \renewcommand*{\glossarysubentryfield}[6]{%
9864     &
9865     \glssubentryitem{##2}%
9866     \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%
9867 }%
```

Backward compatible super4colheader style.

```
9868 \compatglossarystyle{super4colheader}{%
9869   \csuse{@glscompstyle@super4col}%
9870 }%
```

Backward compatible super4colborder style.

```
9871 \compatglossarystyle{super4colborder}{%
9872   \csuse{@glscompstyle@super4col}%
9873 }%
```

Backward compatible super4colheaderborder style.

```
9874 \compatglossarystyle{super4colheaderborder}{%
9875   \csuse{@glscompstyle@super4col}%
9876 }%
```

Backward compatible altsuper4col style.

```
9877 \compatglossarystyle{altsuper4col}{%
9878   \csuse{@glscompstyle@super4col}%
9879 }%
```

Backward compatible altsuper4colheader style.

```
9880 \compatglossarystyle{altsuper4colheader}{%
9881   \csuse{@glscompstyle@super4col}%
9882 }%
```

Backward compatible altsuper4colborder style.

```
9883 \compatglossarystyle{altsuper4colborder}{%
9884   \csuse{@glscompstyle@super4col}%
9885 }%
```

Backward compatible altsuper4colheaderborder style.

```
9886 \compatglossarystyle{altsuper4colheaderborder}{%
9887   \csuse{@glscompstyle@super4col}%
9888 }%
```

5 Accessibility Support (glossaries-accsupp Code)

The package is experimental. It is intended to provide a means of using the PDF accessibility support in glossary entries. See the documentation for further details about accessibility support.

```
9889 \NeedsTeXFormat{LaTeX2e}
```

Package version number now in line with main glossaries package number.

```
9890 \ProvidesPackage{glossaries-accsupp}[2016/04/19 v4.22 (NLCT)]
```

```
9891 Experimental glossaries accessibility]
```

Pass all options to glossaries:

```
9892 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
9893 \ProcessOptions
```

This package should be loaded before glossaries-extra, so complain if that has already been loaded.

```
9894 \@ifpackageloaded{glossaries-extra}
```

```
9895 {%
```

```
9896 \PackageWarning{glossaries-accsupp}{The ‘glossaries-accsupp’
```

```
9897 package has been loaded after the ‘glossaries-extra’
```

```
9898 package. This can cause a failure to integrate both
```

```
9899 packages. Either use the ‘accsupp’ option when you
```

```
9900 load ‘glossaries-extra’ or load ‘glossaries-accsupp’
```

```
9901 before loading ‘glossaries-extra’}%
```

```
9902 }
```

```
9903 {}
```

tibleglossentry Override style compatibility macros:

```
9904 \def\compatibleglossentry#1#2{%
```

```
9905 \toks@{#2}%
```

```
9906 \protected@edef\@do@glossentry{%
```

```
9907 \noexpand\accsuppglossaryentryfield{#1}%
```

```
9908 {\noexpand\glsnamefont
```

```
9909 {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@name\endcsname}}%
```

```
9910 {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@desc\endcsname}%
```

```
9911 {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@symbol\endcsname}%
```

```
9912 {\the\toks@}%
```

```
9913 }%
```

```
9914 \@do@glossentry
```

```
9915 }
```

lesubglossentry

```
9916 \def\compatiblesubglossentry#1#2#3{%
9917   \toks@{#3}%
9918   \protected@edef\@do@subglossentry{%
9919     \noexpand\accsuppglossarysubentryfield{\number#1}%
9920     {#2}%
9921     {\noexpand\glsnamefont
9922      {\expandafter\expandonce\csname glo@\glsdetoklabel{#2}@name\endcsname}}%
9923     {\expandafter\expandonce\csname glo@\glsdetoklabel{#2}@desc\endcsname}%
9924     {\expandafter\expandonce\csname glo@\glsdetoklabel{#2}@symbol\endcsname}%
9925     {\the\toks@}%
9926   }%
9927   \@do@subglossentry
9928 }
```

Required packages:

```
9929 \RequirePackage{glossaries}
9930 \RequirePackage{accsupp}
```

5.1 Defining Replacement Text

The version 0.1 stored the replacement text in the symbol key. This has been changed to use the new keys defined here. Example of use:

```
\newglossaryentry{dr}{name=Dr,description={},access={Doctor}}
```

access The replacement text corresponding to the name key:

```
9931 \define@key{glossentry}{access}{%
9932   \def\@glo@access{#1}%
9933 }
```

textaccess The replacement text corresponding to the text key:

```
9934 \define@key{glossentry}{textaccess}{%
9935   \def\@glo@textaccess{#1}%
9936 }
```

firstaccess The replacement text corresponding to the first key:

```
9937 \define@key{glossentry}{firstaccess}{%
9938   \def\@glo@firstaccess{#1}%
9939 }
```

pluralaccess The replacement text corresponding to the plural key:

```
9940 \define@key{glossentry}{pluralaccess}{%
9941   \def\@glo@pluralaccess{#1}%
9942 }
```

firstpluralaccess The replacement text corresponding to the firstplural key:

```
9943 \define@key{glossentry}{firstpluralaccess}{%
9944   \def\@glo@firstpluralaccess{#1}%
9945 }
```

symbolaccess The replacement text corresponding to the symbol key:

```
9946 \define@key{glossentry}{symbolaccess}{%
9947   \def\@glo@symbolaccess{#1}%
9948 }
```

symbolpluralaccess The replacement text corresponding to the symbolplural key:

```
9949 \define@key{glossentry}{symbolpluralaccess}{%
9950   \def\@glo@symbolpluralaccess{#1}%
9951 }
```

descriptionaccess The replacement text corresponding to the description key:

```
9952 \define@key{glossentry}{descriptionaccess}{%
9953   \def\@glo@descaccess{#1}%
9954 }
```

descriptionpluralaccess The replacement text corresponding to the descriptionplural key:

```
9955 \define@key{glossentry}{descriptionpluralaccess}{%
9956   \def\@glo@descpluralaccess{#1}%
9957 }
```

shortaccess The replacement text corresponding to the short key:

```
9958 \define@key{glossentry}{shortaccess}{%
9959   \def\@glo@shortaccess{#1}%
9960 }
```

shortpluralaccess The replacement text corresponding to the shortplural key:

```
9961 \define@key{glossentry}{shortpluralaccess}{%
9962   \def\@glo@shortpluralaccess{#1}%
9963 }
```

longaccess The replacement text corresponding to the long key:

```
9964 \define@key{glossentry}{longaccess}{%
9965   \def\@glo@longaccess{#1}%
9966 }
```

longpluralaccess The replacement text corresponding to the longplural key:

```
9967 \define@key{glossentry}{longpluralaccess}{%
9968   \def\@glo@longpluralaccess{#1}%
9969 }
```

There are no equivalent keys for the user1...user6 keys. The replacement text would have to be explicitly put in the value, e.g., user1={\glsacccsupp{inches}{in}}.

Append these new keys to \@gls@keymap:

```

9970 \appto\@gls@keymap{,%
9971   {access}{access},%
9972   {textaccess}{textaccess},%
9973   {firstaccess}{firstaccess},%
9974   {pluralaccess}{pluralaccess},%
9975   {firstpluralaccess}{firstpluralaccess},%
9976   {symbolaccess}{symbolaccess},%
9977   {symbolpluralaccess}{symbolpluralaccess},%
9978   {descaccess}{descaccess},%
9979   {descpluralaccess}{descpluralaccess},%
9980   {shortaccess}{shortaccess},%
9981   {shortpluralaccess}{shortpluralaccess},%
9982   {longaccess}{longaccess},%
9983   {longpluralaccess}{longpluralaccess}%
9984 }
```

\@gls@noaccess Indicates that no replacement text has been provided.

```

9985 \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):

```

9986 \let\@gls@oldnewglossaryentryprehook\@newglossaryentryprehook
9987 \renewcommand*{\@newglossaryentryprehook}{%
9988   \@gls@oldnewglossaryentryprehook
9989   \def\@glo@access{\@glo@symbol}%
```

Initialise the other keys:

```

9990 \def\@glo@textaccess{\@glo@access}%
9991 \def\@glo@firstaccess{\@glo@access}%
9992 \def\@glo@pluralaccess{\@glo@textaccess}%
9993 \def\@glo@firstpluralaccess{\@glo@pluralaccess}%
9994 \def\@glo@symbolaccess{\relax}%
9995 \def\@glo@symbolpluralaccess{\@glo@symbolaccess}%
9996 \def\@glo@descaccess{\relax}%
9997 \def\@glo@descpluralaccess{\@glo@descaccess}%
9998 \def\@glo@shortaccess{\relax}%
9999 \def\@glo@shortpluralaccess{\@glo@shortaccess}%
10000 \def\@glo@longaccess{\relax}%
10001 \def\@glo@longpluralaccess{\@glo@longaccess}%
10002 }
```

Add to the end hook:

```

10003 \let\@gls@oldnewglossaryentryposthook\@newglossaryentryposthook
10004 \renewcommand*{\@newglossaryentryposthook}{%
10005   \@gls@oldnewglossaryentryposthook
```

Store the access information:

```

10006 \expandafter
10007   \protected@xdef\csname glo@\@glo@label @access\endcsname{%
```

```

10008      \@glo@access}%
10009 \expandafter
10010      \protected@xdef\csname glo@\@glo@label @textaccess\endcsname{%
10011      \@glo@textaccess}%
10012 \expandafter
10013      \protected@xdef\csname glo@\@glo@label @firstaccess\endcsname{%
10014      \@glo@firstaccess}%
10015 \expandafter
10016      \protected@xdef\csname glo@\@glo@label @pluralaccess\endcsname{%
10017      \@glo@pluralaccess}%
10018 \expandafter
10019      \protected@xdef\csname glo@\@glo@label @firstpluralaccess\endcsname{%
10020      \@glo@firstpluralaccess}%
10021 \expandafter
10022      \protected@xdef\csname glo@\@glo@label @symbolaccess\endcsname{%
10023      \@glo@symbolaccess}%
10024 \expandafter
10025      \protected@xdef\csname glo@\@glo@label @symbolpluralaccess\endcsname{%
10026      \@glo@symbolpluralaccess}%
10027 \expandafter
10028      \protected@xdef\csname glo@\@glo@label @descaccess\endcsname{%
10029      \@glo@descaccess}%
10030 \expandafter
10031      \protected@xdef\csname glo@\@glo@label @descpluralaccess\endcsname{%
10032      \@glo@descpluralaccess}%
10033 \expandafter
10034      \protected@xdef\csname glo@\@glo@label @shortaccess\endcsname{%
10035      \@glo@shortaccess}%
10036 \expandafter
10037      \protected@xdef\csname glo@\@glo@label @shortpluralaccess\endcsname{%
10038      \@glo@shortpluralaccess}%
10039 \expandafter
10040      \protected@xdef\csname glo@\@glo@label @longaccess\endcsname{%
10041      \@glo@longaccess}%
10042 \expandafter
10043      \protected@xdef\csname glo@\@glo@label @longpluralaccess\endcsname{%
10044      \@glo@longpluralaccess}%
10045 }

```

5.2 Accessing Replacement Text

`\glsentryaccess` Get the value of the access key for the entry with the given label:

```

10046 \newcommand*{\glsentryaccess}[1]{%
10047   \@gls@entry@field{#1}{access}%
10048 }

```

`\glsentrytextaccess` Get the value of the textaccess key for the entry with the given label:

```

10049 \newcommand*{\glsentrytextaccess}[1]{%

```

```
10050 \@gls@entry@field{#1}{textaccess}%
10051 }
```

entryfirstaccess Get the value of the firstaccess key for the entry with the given label:

```
10052 \newcommand*{\glsentryfirstaccess}[1]{%
10053 \@gls@entry@field{#1}{firstaccess}%
10054 }
```

entrypluralaccess Get the value of the pluralaccess key for the entry with the given label:

```
10055 \newcommand*{\glsentrypluralaccess}[1]{%
10056 \@gls@entry@field{#1}{pluralaccess}%
10057 }
```

entryfirstpluralaccess Get the value of the firstpluralaccess key for the entry with the given label:

```
10058 \newcommand*{\glsentryfirstpluralaccess}[1]{%
10059 \csname glo@#1@firstpluralaccess\endcsname
10060 }
```

entrysymbolaccess Get the value of the symbolaccess key for the entry with the given label:

```
10061 \newcommand*{\glsentrysymbolaccess}[1]{%
10062 \@gls@entry@field{#1}{symbolaccess}%
10063 }
```

entrysymbolpluralaccess Get the value of the symbolpluralaccess key for the entry with the given label:

```
10064 \newcommand*{\glsentrysymbolpluralaccess}[1]{%
10065 \@gls@entry@field{#1}{symbolpluralaccess}%
10066 }
```

entrydescaccess Get the value of the descriptionaccess key for the entry with the given label:

```
10067 \newcommand*{\glsentrydescaccess}[1]{%
10068 \@gls@entry@field{#1}{descaccess}%
10069 }
```

entrydescpluralaccess Get the value of the descriptionpluralaccess key for the entry with the given label:

```
10070 \newcommand*{\glsentrydescpluralaccess}[1]{%
10071 \@gls@entry@field{#1}{descaccess}%
10072 }
```

entryshortaccess Get the value of the shortaccess key for the entry with the given label:

```
10073 \newcommand*{\glsentryshortaccess}[1]{%
10074 \@gls@entry@field{#1}{shortaccess}%
10075 }
```

entryshortpluralaccess Get the value of the shortpluralaccess key for the entry with the given label:

```
10076 \newcommand*{\glsentryshortpluralaccess}[1]{%
10077 \@gls@entry@field{#1}{shortpluralaccess}%
10078 }
```

entrylongaccess Get the value of the longaccess key for the entry with the given label:

```
10079 \newcommand*{\glentrylongaccess}[1]{%
10080   \@gls@entry@field{#1}{longaccess}%
10081 }
```

ongpluralaccess Get the value of the longpluralaccess key for the entry with the given label:

```
10082 \newcommand*{\glentrylongpluralaccess}[1]{%
10083   \@gls@entry@field{#1}{longpluralaccess}%
10084 }
```

\glsacccsupp \glsacccsupp{<replacement text>}{<text>}

This can be redefined to use E or Alt instead of ActualText. (I don't have the software to test the E or Alt options.)

```
10085 \newcommand*{\glsacccsupp}[2]{%
10086   \BeginAccSupp{ActualText=#1}#2\EndAccSupp{}%
10087 }
```

\xglsacccsupp Fully expands replacement text before calling \glsacccsupp

```
10088 \newcommand*{\xglsacccsupp}[2]{%
10089   \protected@edef\@gls@replacementtext{#1}%
10090   \expandafter\glsacccsupp\expandafter{\@gls@replacementtext}{#2}%
10091 }
```

@access@display

```
10092 \newcommand*{\@gls@access@display}[2]{%
10093   \protected@edef\@glo@access{#2}%
10094   \ifx\@glo@access\@gls@noaccess
10095     #1%
10096   \else
10097     \xglsacccsupp{\@glo@access}{#1}%
10098   \fi
10099 }
```

meaccessdisplay Displays the first argument with the accessibility text for the entry with the label given by the second argument (if set).

```
10100 \DeclareRobustCommand*{\glsnameaccessdisplay}[2]{%
10101   \@gls@access@display{#1}{\glentryaccess{#2}}%
10102 }
```

xtaccessdisplay As above but for the textaccess replacement text.

```
10103 \DeclareRobustCommand*{\glstextaccessdisplay}[2]{%
10104   \@gls@access@display{#1}{\glentrytextaccess{#2}}%
10105 }
```

alaccessdisplay As above but for the pluralaccess replacement text.

```
10106 \DeclareRobustCommand*{\glspluralaccessdisplay}[2]{%
10107   \@gls@access@display{#1}{\glentrypluralaccess{#2}}%
10108 }
```

staccessdisplay As above but for the firstaccess replacement text.

```

10109 \DeclareRobustCommand*\glfirstaccessdisplay}[2]{%
10110 \@gls@access@display{#1}{\glentryfirstaccess{#2}}%
10111 }

```

alaccessdisplay As above but for the firstpluralaccess replacement text.

```

10112 \DeclareRobustCommand*\glfirstpluralaccessdisplay}[2]{%
10113 \@gls@access@display{#1}{\glentryfirstpluralaccess{#2}}%
10114 }

```

olaccessdisplay As above but for the symbolaccess replacement text.

```

10115 \DeclareRobustCommand*\glssymbolaccessdisplay}[2]{%
10116 \@gls@access@display{#1}{\glentrysymbolaccess{#2}}%
10117 }

```

alaccessdisplay As above but for the symbolpluralaccess replacement text.

```

10118 \DeclareRobustCommand*\glssymbolpluralaccessdisplay}[2]{%
10119 \@gls@access@display{#1}{\glentrysymbolpluralaccess{#2}}%
10120 }

```

onaccessdisplay As above but for the descriptionaccess replacement text.

```

10121 \DeclareRobustCommand*\glsdescriptionaccessdisplay}[2]{%
10122 \@gls@access@display{#1}{\glentrydescaccess{#2}}%
10123 }

```

alaccessdisplay As above but for the descriptionpluralaccess replacement text.

```

10124 \DeclareRobustCommand*\glsdescriptionpluralaccessdisplay}[2]{%
10125 \@gls@access@display{#1}{\glentrydescpluralaccess{#2}}%
10126 }

```

rtaccessdisplay As above but for the shortaccess replacement text.

```

10127 \DeclareRobustCommand*\glsshortaccessdisplay}[2]{%
10128 \@gls@access@display{#1}{\glentryshortaccess{#2}}%
10129 }

```

alaccessdisplay As above but for the shortpluralaccess replacement text.

```

10130 \DeclareRobustCommand*\glsshortpluralaccessdisplay}[2]{%
10131 \@gls@access@display{#1}{\glentryshortpluralaccess{#2}}%
10132 }

```

ngaccessdisplay As above but for the longaccess replacement text.

```

10133 \DeclareRobustCommand*\glslongaccessdisplay}[2]{%
10134 \@gls@access@display{#1}{\glentrylongaccess{#2}}%
10135 }

```

alaccessdisplay As above but for the longpluralaccess replacement text.

```

10136 \DeclareRobustCommand*\glslongpluralaccessdisplay}[2]{%
10137 \@gls@access@display{#1}{\glentrylongpluralaccess{#2}}%
10138 }

```

`\glsaccessdisplay` Gets the replacement text corresponding to the named key given by the first argument and calls the appropriate command defined above.

```

10139 \DeclareRobustCommand*\glsaccessdisplay}[3]{%
10140   \@ifundefined{gls#1accessdisplay}%
10141   {%
10142     \PackageError{glossaries-accsupp}{No accessibility support
10143       for key ‘#1’}{}%
10144   }%
10145   {%
10146     \csname gls#1accessdisplay\endcsname{#2}{#3}%
10147   }%
10148 }

```

`\default@entryfmt` Redefine the default entry format to use accessibility information

```

10149 \renewcommand*\@@gls@default@entryfmt}[2]{%
10150   \ifdefempty\glscustomtext
10151   {%
10152     \glsifplural
10153     {%

```

Plural form

```

10154     \glscapscase
10155     {%

```

Don't adjust case

```

10156     \ifglsused\glslabel
10157     {%

```

Subsequent use

```

10158     #2{\glspluralaccessdisplay
10159       {\glsentryplural{\glslabel}}{\glslabel}}%
10160     {\glsdescriptionpluralaccessdisplay
10161       {\glsentrydescplural{\glslabel}}{\glslabel}}%
10162     {\glsymbolpluralaccessdisplay
10163       {\glsentrysymbolplural{\glslabel}}{\glslabel}}
10164     {\glsinsert}%
10165   }%
10166   {%

```

First use

```

10167     #1{\glsfirstpluralaccessdisplay
10168       {\glsentryfirstplural{\glslabel}}{\glslabel}}%
10169     {\glsdescriptionpluralaccessdisplay
10170       {\glsentrydescplural{\glslabel}}{\glslabel}}%
10171     {\glsymbolpluralaccessdisplay
10172       {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10173     {\glsinsert}%
10174   }%
10175   }%
10176   {%

```

Make first letter upper case

```
10177      \ifglsused\glslabel
10178      {%
```

Subsequent use.

```
10179      #2{\glspluralaccessdisplay
10180          {\Glsentryplural{\glslabel}}{\glslabel}}%
10181          {\glsdescriptionpluralaccessdisplay
10182          {\glsentrydescplural{\glslabel}}{\glslabel}}%
10183          {\glssymbolpluralaccessdisplay
10184          {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10185          {\glsinsert}}%
10186      }%
10187      {%
```

First use

```
10188      #1{\glsfirstpluralaccessdisplay
10189          {\Glsentryfirstplural{\glslabel}}{\glslabel}}%
10190          {\glsdescriptionpluralaccessdisplay
10191          {\glsentrydescplural{\glslabel}}{\glslabel}}%
10192          {\glssymbolpluralaccessdisplay
10193          {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10194          {\glsinsert}}%
10195      }%
10196      }%
10197      {%
```

Make all upper case

```
10198      \ifglsused\glslabel
10199      {%
```

Subsequent use

```
10200      \MakeUppercase{%
10201      #2{\glspluralaccessdisplay
10202          {\glsentryplural{\glslabel}}{\glslabel}}%
10203          {\glsdescriptionpluralaccessdisplay
10204          {\glsentrydescplural{\glslabel}}{\glslabel}}%
10205          {\glssymbolpluralaccessdisplay
10206          {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10207          {\glsinsert}}%
10208      }%
10209      {%
```

First use

```
10210      \MakeUppercase{%
10211      #1{\glsfirstpluralaccessdisplay
10212          {\glsentryfirstplural{\glslabel}}{\glslabel}}%
10213          {\glsdescriptionpluralaccessdisplay
10214          {\glsentrydescplural{\glslabel}}{\glslabel}}%
10215          {\glssymbolpluralaccessdisplay
10216          {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
```

```

10217         {\glsinsert}}}%
10218     }%
10219 }%
10220 }%
10221 {%

```

Singular form

```

10222     \glscapscase
10223     {%

```

Don't adjust case

```

10224     \ifglsused\glslabel
10225     {%

```

Subsequent use

```

10226     #2{\glstextaccessdisplay
10227         {\glsentrytext{\glslabel}}{\glslabel}}%
10228     {\glsdescriptionaccessdisplay
10229         {\glsentrydesc{\glslabel}}{\glslabel}}%
10230     {\glssymbolaccessdisplay
10231         {\glsentrysymbol{\glslabel}}{\glslabel}}%
10232     {\glsinsert}}%
10233 }%
10234 {%

```

First use

```

10235     #1{\glsfirstaccessdisplay
10236         {\glsentryfirst{\glslabel}}{\glslabel}}%
10237     {\glsdescriptionaccessdisplay
10238         {\glsentrydesc{\glslabel}}{\glslabel}}%
10239     {\glssymbolaccessdisplay
10240         {\glsentrysymbol{\glslabel}}{\glslabel}}%
10241     {\glsinsert}}%
10242 }%
10243 }%
10244 {%

```

Make first letter upper case

```

10245     \ifglsused\glslabel
10246     {%

```

Subsequent use

```

10247     #2{\glstextaccessdisplay
10248         {\Glsentrytext{\glslabel}}{\glslabel}}%
10249     {\glsdescriptionaccessdisplay
10250         {\glsentrydesc{\glslabel}}{\glslabel}}%
10251     {\glssymbolaccessdisplay
10252         {\glsentrysymbol{\glslabel}}{\glslabel}}%
10253     {\glsinsert}}%
10254 }%
10255 {%

```


First use

```

10256      #1{\glsfirstaccessdisplay
10257          {\Glsentryfirst{\glslabel}}{\glslabel}}%
10258          {\glsdescriptionaccessdisplay
10259              {\glsentrydesc{\glslabel}}{\glslabel}}%
10260          {\glsymbolaccessdisplay
10261              {\glsentrysymbol{\glslabel}}{\glslabel}}%
10262          {\glsinsert}}%
10263      }%
10264  }%
10265  {%

```

Make all upper case

```

10266      \ifglsused\glslabel
10267      {%

```

Subsequent use

```

10268      \MakeUppercase{%
10269          #2{\glstextaccessdisplay
10270              {\glsentrytext{\glslabel}}{\glslabel}}%
10271              {\glsdescriptionaccessdisplay
10272                  {\glsentrydesc{\glslabel}}{\glslabel}}%
10273                  {\glsymbolaccessdisplay
10274                      {\glsentrysymbol{\glslabel}}{\glslabel}}%
10275                      {\glsinsert}}}%
10276      }%
10277  {%

```

First use

```

10278      \MakeUppercase{%
10279          #1{\glsfirstaccessdisplay
10280              {\glsentryfirst{\glslabel}}{\glslabel}}%
10281              {\glsdescriptionaccessdisplay
10282                  {\glsentrydesc{\glslabel}}{\glslabel}}%
10283                  {\glsymbolaccessdisplay
10284                      {\glsentrysymbol{\glslabel}}{\glslabel}}%
10285                      {\glsinsert}}}%
10286      }%
10287  }%
10288  }%
10289  }%
10290  {%

```

Custom text provided in \glsdisp

```

10291      \ifglsused{\glslabel}%
10292      {%

```

Subsequent use

```

10293      #2{\glscustomtext}%
10294      {\glsdescriptionaccessdisplay
10295          {\glsentrydesc{\glslabel}}{\glslabel}}%

```

```

10296      {\glssymbolaccessdisplay
10297      {\glentrysymbol{\glslabel}}{\glslabel}}%
10298      {\glsinsert}%
10299  }%
10300  {%

```

First use

```

10301      #1{\glscustomtext}%
10302      {\glsdescriptionaccessdisplay
10303      {\glentrydesc{\glslabel}}{\glslabel}}%
10304      {\glssymbolaccessdisplay
10305      {\glentrysymbol{\glslabel}}{\glslabel}}%
10306      {\glsinsert}%
10307  }%
10308  }%
10309 }

```

`\glsgenentryfmt` Redefine to use accessibility information.

```

10310 \renewcommand*{\glsgenentryfmt}{%
10311   \ifdefempty\glscustomtext
10312   {%
10313     \glsifplural
10314     {%

```

Plural form

```

10315     \glscapscase
10316     {%

```

Don't adjust case

```

10317     \ifglused\glslabel
10318     {%

```

Subsequent use

```

10319     \glspluralaccessdisplay
10320     {\glentryplural{\glslabel}}{\glslabel}%
10321     \glsinsert
10322   }%
10323   {%

```

First use

```

10324     \glsfirstpluralaccessdisplay
10325     {\glentryfirstplural{\glslabel}}{\glslabel}%
10326     \glsinsert
10327   }%
10328   }%
10329   {%

```

Make first letter upper case

```

10330     \ifglused\glslabel
10331     {%

```

Subsequent use.

```
10332      \glspluralaccessdisplay
10333      {\Glsentryplural{\glslabel}}{\glslabel}%
10334      \glsinsert
10335      }%
10336      {%
```

First use

```
10337      \glsfirstpluralaccessdisplay
10338      {\Glsentryfirstplural{\glslabel}}{\glslabel}%
10339      \glsinsert
10340      }%
10341      }%
10342      {%
```

Make all upper case

```
10343      \ifglused\glslabel
10344      {%
```

Subsequent use

```
10345      \glspluralaccessdisplay
10346      {\mfirstucMakeUppercase{\glsentryplural{\glslabel}}}%
10347      {\glslabel}%
10348      \mfirstucMakeUppercase{\glsinsert}%
10349      }%
10350      {%
```

First use

```
10351      \glsfirstpluralaccessdisplay
10352      {\mfirstucMakeUppercase{\glsentryfirstplural{\glslabel}}}%
10353      {\glslabel}%
10354      \mfirstucMakeUppercase{\glsinsert}%
10355      }%
10356      }%
10357      }%
10358      {%
```

Singular form

```
10359      \glscapscase
10360      {%
```

Don't adjust case

```
10361      \ifglused\glslabel
10362      {%
```

Subsequent use

```
10363      \glstextaccessdisplay{\glsentrytext{\glslabel}}{\glslabel}%
10364      \glsinsert
10365      }%
10366      {%
```

First use

```
10367      \glsfirstaccessdisplay{\glsentryfirst{\glslabel}}{\glslabel}%
10368      \glsinsert
10369      }%
10370      }%
10371      {%
```

Make first letter upper case

```
10372      \ifglsused\glslabel
10373      {%
```

Subsequent use

```
10374      \glstextaccessdisplay{\Glsentrytext{\glslabel}}{\glslabel}%
10375      \glsinsert
10376      }%
10377      {%
```

First use

```
10378      \glsfirstaccessdisplay{\Glsentryfirst{\glslabel}}{\glslabel}%
10379      \glsinsert
10380      }%
10381      }%
10382      {%
```

Make all upper case

```
10383      \ifglsused\glslabel
10384      {%
```

Subsequent use

```
10385      \glstextaccessdisplay
10386      {\mfirstucMakeUppercase{\glsentrytext{\glslabel}}}{\glslabel}%
10387      \mfirstucMakeUppercase{\glsinsert}%
10388      }%
10389      {%
```

First use

```
10390      \glsfirstaccessdisplay
10391      {\mfirstucMakeUppercase{\glsentryfirst{\glslabel}}}{\glslabel}%
10392      \mfirstucMakeUppercase{\glsinsert}%
10393      }%
10394      }%
10395      }%
10396      }%
10397      {%
```

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.

```
10398      \glscustomtext\glsinsert
10399      }%
10400 }
```

`\glsgenacfmt` Redefine to include accessibility information.

```
10401 \renewcommand*{\glsgenacfmt}{%
10402   \ifdefempty\glscustomtext
10403   {%
10404     \ifglused\glslabel
10405     {%
```

Subsequent use:

```
10406     \glsifplural
10407     {%
```

Subsequent plural form:

```
10408     \glscapscase
10409     {%
```

Subsequent plural form, don't adjust case:

```
10410     \acronymfont
10411     {\glsshortpluralaccessdisplay
10412      {\glentryshortpl{\glslabel}}{\glslabel}}%
10413     \glsinsert
10414   }%
10415   {%
```

Subsequent plural form, make first letter upper case:

```
10416     \acronymfont
10417     {\glsshortpluralaccessdisplay
10418      {\Glsentryshortpl{\glslabel}}{\glslabel}}%
10419     \glsinsert
10420   }%
10421   {%
```

Subsequent plural form, all caps:

```
10422     \mfirstucMakeUppercase
10423     {\acronymfont
10424      {\glsshortpluralaccessdisplay
10425       {\glentryshortpl{\glslabel}}{\glslabel}}%
10426      \glsinsert}%
10427   }%
10428   }%
10429   {%
```

Subsequent singular form

```
10430     \glscapscase
10431     {%
```

Subsequent singular form, don't adjust case:

```
10432     \acronymfont
10433     {\glsshortaccessdisplay{\glentryshort{\glslabel}}{\glslabel}}%
10434     \glsinsert
10435   }%
10436   {%
```

Subsequent singular form, make first letter upper case:

```
10437      \acronymfont
10438      {\glsshortaccessdisplay{\Glsentryshort{\glslabel}}{\glslabel}}%
10439      \glsinsert
10440      }%
10441      {%
```

Subsequent singular form, all caps:

```
10442      \mfirstucMakeUppercase
10443      {\acronymfont{%
10444      \glsshortaccessdisplay{\Glsentryshort{\glslabel}}{\glslabel}}%
10445      \glsinsert}%
10446      }%
10447      }%
10448      }%
10449      {%
```

First use:

```
10450      \glsifplural
10451      {%
```

First use plural form:

```
10452      \glscapscase
10453      {%
```

First use plural form, don't adjust case:

```
10454      \genplacrfullformat{\glslabel}{\glsinsert}%
10455      }%
10456      {%
```

First use plural form, make first letter upper case:

```
10457      \Genplacrfullformat{\glslabel}{\glsinsert}%
10458      }%
10459      {%
```

First use plural form, all caps:

```
10460      \mfirstucMakeUppercase
10461      {\genplacrfullformat{\glslabel}{\glsinsert}}%
10462      }%
10463      }%
10464      {%
```

First use singular form

```
10465      \glscapscase
10466      {%
```

First use singular form, don't adjust case:

```
10467      \genacrfullformat{\glslabel}{\glsinsert}%
10468      }%
10469      {%
```

First use singular form, make first letter upper case:

```
10470      \Genacrfullformat{\glslabel}{\glsinsert}%
10471      }%
10472      {%
```

First use singular form, all caps:

```
10473      \mfirstucMakeUppercase
10474      {\genacrfullformat{\glslabel}{\glsinsert}}%
10475      }%
10476      }%
10477      }%
10478      }%
10479      {%
```

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.

```
10480      \glscustomtext
10481      }%
10482 }
```

enacrfullformat Redefine to include accessibility information.

```
10483 \renewcommand*{\genacrfullformat}[2]{%
10484   \glslongaccessdisplay{\glsentrylong{#1}}{#1}#2\space
10485   (\glsshortaccessdisplay{\protect\firstacronymfont{\glsentryshort{#1}}}{#1}}%
10486 }
```

enacrfullformat Redefine to include accessibility information.

```
10487 \renewcommand*{\Genacrfullformat}[2]{%
10488   \glslongaccessdisplay{\Glsentrylong{#1}}{#1}#2\space
10489   (\glsshortaccessdisplay{\protect\firstacronymfont{\Glsentryshort{#1}}}{#1}}%
10490 }
```

placrfullformat Redefine to include accessibility information.

```
10491 \renewcommand*{\genplacrfullformat}[2]{%
10492   \glslongpluralaccessdisplay{\glsentrylongpl{#1}}{#1}#2\space
10493   (\glsshortpluralaccessdisplay
10494     {\protect\firstacronymfont{\glsentryshortpl{#1}}}{#1}}%
10495 }
```

placrfullformat Redefine to include accessibility information.

```
10496 \renewcommand*{\Genplacrfullformat}[2]{%
10497   \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}#2\space
10498   (\glsshortpluralaccessdisplay
10499     {\protect\firstacronymfont{\glsentryshortpl{#1}}}{#1}}%
10500 }
```

\@acrshort

```
10501 \def\@acrshort#1#2[#3]{%
10502   \glsdoifexists{#2}%
```

```

10503 {%
10504   \let\do@gl@s@link@checkfirsthyper\relax

10505   \let\gl@sifplural\@secondoftwo
10506   \let\gl@scapscase\@firstofthree
10507   \let\gl@sinsert\@empty
10508   \def\glscustomtext{%
10509     \acronymfont{\glsshortaccessdisplay{\glentryshort{#2}}{#2}}#3%
10510   }%

  Call \@gl@s@link
10511   \@gl@s@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10512 }%

10513 \glspostlinkhook
10514 }

```

\@Acrshort

```

10515 \def\@Acrshort#1#2[#3]{%
10516   \glstoifexists{#2}%
10517   {%
10518     \let\do@gl@s@link@checkfirsthyper\relax

10519     \let\gl@sifplural\@secondoftwo
10520     \let\gl@scapscase\@secondofthree
10521     \let\gl@sinsert\@empty
10522     \def\glscustomtext{%
10523       \acronymfont{\glsshortaccessdisplay{\Glsentryshort{#2}}{#2}}#3%
10524     }%

    Call \@gl@s@link
10525     \@gl@s@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10526   }%

10527   \glspostlinkhook
10528 }

```

\@ACRshort

```

10529 \def\@ACRshort#1#2[#3]{%
10530   \glstoifexists{#2}%
10531   {%
10532     \let\do@gl@s@link@checkfirsthyper\relax

10533     \let\gl@sifplural\@secondoftwo
10534     \let\gl@scapscase\@thirdofthree
10535     \let\gl@sinsert\@empty
10536     \def\glscustomtext{%
10537       \acronymfont{\glsshortaccessdisplay
10538         {\MakeUppercase{\glentryshort{#2}}}{#2}}#3%
10539     }%

```



```

Call \@gls@link
10540   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10541   }%

10542   \glspostlinkhook
10543 }

```

\@acrlong

```

10544 \def\@acrlong#1#2[#3]{%
10545   \glsdoifexists{#2}%
10546   {%
10547     \let\do@gls@link@checkfirsthyper\relax

10548     \let\glsifplural\@secondoftwo
10549     \let\glscapscase\@firstofthree
10550     \let\glsinsert\@empty
10551     \def\glscustomtext{%
10552       \acronymfont{\glslongaccessdisplay{\glsentrylong{#2}}{#2}}#3%
10553     }%

```

```

Call \@gls@link
10554   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10555   }%

10556   \glspostlinkhook
10557 }

```

\@Acrlong

```

10558 \def\@Acrlong#1#2[#3]{%
10559   \glsdoifexists{#2}%
10560   {%
10561     \let\do@gls@link@checkfirsthyper\relax

10562     \let\glsifplural\@secondoftwo
10563     \let\glscapscase\@firstofthree
10564     \let\glsinsert\@empty
10565     \def\glscustomtext{%
10566       \acronymfont{\glslongaccessdisplay{\Glsentrylong{#2}}{#2}}#3%
10567     }%

```

```

Call \@gls@link
10568   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10569   }%

10570   \glspostlinkhook
10571 }

```

\@ACRlong

```

10572 \def\@ACRlong#1#2[#3]{%
10573   \glsdoifexists{#2}%
10574   {%
10575     \let\do@gls@link@checkfirsthyper\relax

```

```

10576 \let\glsifplural\@secondoftwo
10577 \let\glsifscapscase\@firstofthree
10578 \let\glsinsert\@empty
10579 \def\glscustomtext{%
10580 \acronymfont{\glslongaccessdisplay{%
10581 \MakeUppercase{\glsentrylong{#2}}}{#2}#3}%
10582 }%

Call \@gls@link
10583 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10584 }%

10585 \glspostlinkhook
10586 }

```

5.3 Displaying the Glossary

We need to redefine the way the glossary entries are formatted to include the accessibility support. The predefined glossary styles use `\glossentryname`, `\glossentrydesc` and `\glossentrysymbol`, but we need to provide compatibility with earlier versions in case users have defined their own styles using `\accsuppglossaryentryfield` and `\accsuppglossarysubentryfield`.

Now redefine `\glossentryname`, `\glossentrydesc` and `\glossentrysymbol` etc so they use the accessibility stuff.

```

10587 \renewcommand*{\glossentryname}[1]{%
10588 \glsdoifexists{#1}%
10589 {%
10590 \glsnamefont{\glsnameaccessdisplay{\glsentryname{#1}}{#1}}%
10591 }%
10592 }

10593 \renewcommand*{\glossentryname}[1]{%
10594 \glsdoifexists{#1}%
10595 {%
10596 \glsnamefont{\glsnameaccessdisplay{\Glsentryname{#1}}{#1}}%
10597 }%
10598 }

10599 \renewcommand*{\glossentrydesc}[1]{%
10600 \glsdoifexists{#1}%
10601 {%
10602 \glsdescriptionaccessdisplay{\glsentrydesc{#1}}{#1}%
10603 }%
10604 }

10605 \renewcommand*{\Glossentrydesc}[1]{%
10606 \glsdoifexists{#1}%
10607 {%
10608 \glsdescriptionaccessdisplay{\Glsentrydesc{#1}}{#1}%
10609 }%
10610 }

```

```

10611 \renewcommand*{\glossentrysymbol}[1]{%
10612   \glsdoifexists{#1}%
10613   {%
10614     \glssymbolaccessdisplay{\glsentrysymbol{#1}}{#1}%
10615   }%
10616 }

10617 \renewcommand*{\Glossentrysymbol}[1]{%
10618   \glsdoifexists{#1}%
10619   {%
10620     \glssymbolaccessdisplay{\Glsentrysymbol{#1}}{#1}%
10621   }%
10622 }

```

ssaryentryfield

```

10623 \newcommand*{\accsuppglossaryentryfield}[5]{%
10624   \glossaryentryfield{#1}%
10625   {\glsnameaccessdisplay{#2}{#1}}%
10626   {\glsdescriptionaccessdisplay{#3}{#1}}%
10627   {\glssymbolaccessdisplay{#4}{#1}}{#5}%
10628 }

```

rysubentryfield

```

10629 \newcommand*{\accsuppglossarysubentryfield}[6]{%
10630   \glossarysubentryfield{#1}{#2}%
10631   {\glsnameaccessdisplay{#3}{#2}}%
10632   {\glsdescriptionaccessdisplay{#4}{#2}}%
10633   {\glssymbolaccessdisplay{#5}{#2}}{#6}%
10634 }

```

5.4 Acronyms

Redefine acronym styles provided by glossaries:

long-short *<long>* (*<short>*) acronym style.

```

10635 \renewacronymstyle{long-short}%
10636 {%

```

Check for long form in case this is a mixed glossary.

```

10637   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
10638 }%
10639 {%
10640   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
10641   \renewcommand*{\genacrfullformat}[2]{%
10642     \glslongaccessdisplay{\glsentrylong{##1}}{##1}##2\space
10643     (\glsshortaccessdisplay
10644       {\protect\firstacronymfont{\glsentryshort{##1}}}{##1})%
10645   }%
10646   \renewcommand*{\Genacrfullformat}[2]{%

```

```

10647 \glslongaccessdisplay{\Glsentrylong{##1}}{##1}##2\space
10648 (\glsshortaccessdisplay
10649 {\protect\firstacronymfont{\glsentryshort{##1}}{##1}})%
10650 }%
10651 \renewcommand*{\genplacrfullformat}[2]{%
10652 \glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}##2\space
10653 (\glsshortpluralaccessdisplay
10654 {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}})%
10655 }%
10656 \renewcommand*{\Genplacrfullformat}[2]{%
10657 \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}##2\space
10658 (\glsshortpluralaccessdisplay
10659 {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}})%
10660 }%
10661 \renewcommand*{\acronymentry}[1]{%
10662 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}{##1}}
10663 \renewcommand*{\acronymsort}[2]{##1}%
10664 \renewcommand*{\acronymfont}[1]{##1}%
10665 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
10666 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
10667 }

```

short-long (*short*) (*long*) acronym style.

```

10668 \renewacronymstyle{short-long}%
10669 {%

```

Check for long form in case this is a mixed glossary.

```

10670 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
10671 }%
10672 {%
10673 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
10674 \renewcommand*{\genacrfullformat}[2]{%
10675 \glsshortaccessdisplay
10676 {\protect\firstacronymfont{\glsentryshort{##1}}{##1}##2\space
10677 (\glslongaccessdisplay{\glsentrylong{##1}}{##1}})%
10678 }%
10679 \renewcommand*{\Genacrfullformat}[2]{%
10680 \glsshortaccessdisplay
10681 {\protect\firstacronymfont{\Glsentryshort{##1}}{##1}##2\space
10682 (\glslongaccessdisplay{\glsentrylong{##1}}{##1}})%
10683 }%
10684 \renewcommand*{\genplacrfullformat}[2]{%
10685 \glsshortpluralaccessdisplay
10686 {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}##2\space
10687 (\glslongpluralaccessdisplay
10688 {\glsentrylongpl{##1}}{##1}})%
10689 }%
10690 \renewcommand*{\Genplacrfullformat}[2]{%
10691 \glsshortpluralaccessdisplay
10692 {\protect\firstacronymfont{\Glsentryshortpl{##1}}{##1}##2\space

```

```

10693 (\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1})%
10694 }%
10695 \renewcommand*{\acronymentry}[1]{%
10696   \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}%
10697 \renewcommand*{\acronymsort}[2]{##1}%
10698 \renewcommand*{\acronymfont}[1]{##1}%
10699 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
10700 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
10701 }

```

long-short-desc *<long>* (*<short>*) acronym style that has an accompanying description (which the user needs to supply).

```

10702 \renewacronymstyle{long-short-desc}%
10703 {%
10704   \GlsUseAcrEntryDispStyle{long-short}%
10705 }%
10706 {%
10707   \GlsUseAcrStyleDefs{long-short}%
10708   \renewcommand*{\GenericAcronymFields}{}%
10709   \renewcommand*{\acronymsort}[2]{##2}%
10710   \renewcommand*{\acronymentry}[1]{%
10711     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10712     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10713 }

```

g-sc-short-desc *<long>* (\textsc{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

10714 \renewacronymstyle{long-sc-short-desc}%
10715 {%
10716   \GlsUseAcrEntryDispStyle{long-sc-short}%
10717 }%
10718 {%
10719   \GlsUseAcrStyleDefs{long-sc-short}%
10720   \renewcommand*{\GenericAcronymFields}{}%
10721   \renewcommand*{\acronymsort}[2]{##2}%
10722   \renewcommand*{\acronymentry}[1]{%
10723     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10724     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10725 }

```

g-sm-short-desc *<long>* (\textsmaller{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

10726 \renewacronymstyle{long-sm-short-desc}%
10727 {%
10728   \GlsUseAcrEntryDispStyle{long-sm-short}%
10729 }%
10730 {%
10731   \GlsUseAcrStyleDefs{long-sm-short}%
10732   \renewcommand*{\GenericAcronymFields}{}%

```

```

10733 \renewcommand*{\acronymsort}[2]{##2}%
10734 \renewcommand*{\acronymentry}[1]{%
10735     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10736     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10737 }

```

short-long-desc *(short)* (*{<long>}*) acronym style that has an accompanying description (which the user needs to supply).

```

10738 \renewacronymstyle{short-long-desc}%
10739 {%
10740     \GlsUseAcrEntryDispStyle{short-long}%
10741 }%
10742 {%
10743     \GlsUseAcrStyleDefs{short-long}%
10744     \renewcommand*{\GenericAcronymFields}{}%
10745     \renewcommand*{\acronymsort}[2]{##2}%
10746     \renewcommand*{\acronymentry}[1]{%
10747         \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10748         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10749 }

```

short-long-desc *(long)* (\textsc{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

10750 \renewacronymstyle{sc-short-long-desc}%
10751 {%
10752     \GlsUseAcrEntryDispStyle{sc-short-long}%
10753 }%
10754 {%
10755     \GlsUseAcrStyleDefs{sc-short-long}%
10756     \renewcommand*{\GenericAcronymFields}{}%
10757     \renewcommand*{\acronymsort}[2]{##2}%
10758     \renewcommand*{\acronymentry}[1]{%
10759         \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10760         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10761 }

```

short-long-desc *(long)* (\textsmaller{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

10762 \renewacronymstyle{sm-short-long-desc}%
10763 {%
10764     \GlsUseAcrEntryDispStyle{sm-short-long}%
10765 }%
10766 {%
10767     \GlsUseAcrStyleDefs{sm-short-long}%
10768     \renewcommand*{\GenericAcronymFields}{}%
10769     \renewcommand*{\acronymsort}[2]{##2}%
10770     \renewcommand*{\acronymentry}[1]{%
10771         \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10772         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%

```

10773 }

dua *<long>* only acronym style.

10774 \renewacronymstyle{dua}%
10775 {%

Check for long form in case this is a mixed glossary.

10776 \ifdefempty\glscustomtext
10777 {%
10778 \ifglshaslong{\glslabel}%
10779 {%
10780 \glsifplural
10781 {%

Plural form:

10782 \glscapscase
10783 {%

Plural form, don't adjust case:

10784 \glslongpluralaccessdisplay{\glentrylongpl{\glslabel}}{\glslabel}%
10785 \glsinsert
10786 }%
10787 {%

Plural form, make first letter upper case:

10788 \glslongpluralaccessdisplay{\Glentrylongpl{\glslabel}}{\glslabel}%
10789 \glsinsert
10790 }%
10791 {%

Plural form, all caps:

10792 \glslongpluralaccessdisplay
10793 {\mfirstucMakeUppercase{\glentrylongpl{\glslabel}}}{\glslabel}%
10794 \mfirstucMakeUppercase{\glsinsert}%
10795 }%
10796 }%
10797 {%

Singular form

10798 \glscapscase
10799 {%

Singular form, don't adjust case:

10800 \glslongaccessdisplay{\glentrylong{\glslabel}}{\glslabel}\glsinsert
10801 }%
10802 {%

Subsequent singular form, make first letter upper case:

10803 \glslongaccessdisplay{\Glentrylong{\glslabel}}{\glslabel}\glsinsert
10804 }%
10805 {%

Subsequent singular form, all caps:

```

10806         \glslongaccessdisplay
10807         {\mfirstucMakeUppercase
10808          {\glsentrylong{\glslabel}\glsinsert}}{\glslabel}%
10809         \mfirstucMakeUppercase{\glsinsert}%
10810         }%
10811     }%
10812 }%
10813 {%

```

Not an acronym:

```

10814         \glsgenentryfmt
10815     }%
10816 }%
10817 {\glscustomtext\glsinsert}%
10818}%
10819{%
10820 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
10821 \renewcommand*{\acrfullfmt}[3]{%
10822     \glslink[##1]{##2}{%
10823         \glslongaccessdisplay{\glsentrylong{##2}}{##2}##3\space
10824         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}})%
10825 \renewcommand*{\Acrfullfmt}[3]{%
10826     \glslink[##1]{##2}{%
10827         \glslongaccessdisplay{\Glsentrylong{##2}}{##2}##3\space
10828         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}})%
10829 \renewcommand*{\ACRfullfmt}[3]{%
10830     \glslink[##1]{##2}{%
10831         \glslongaccessdisplay
10832         {\mfirstucMakeUppercase{\glsentrylong{##2}}{##2}##3\space
10833         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}})%
10834 \renewcommand*{\acrfullplfmt}[3]{%
10835     \glslink[##1]{##2}{%
10836         \glslongpluralaccessdisplay
10837         {\glsentrylongpl{##2}}{##2}##3\space
10838         (\glsshortpluralaccessdisplay
10839         {\acronymfont{\glsentryshortpl{##2}}}{##2}})%
10840 \renewcommand*{\Acrfullplfmt}[3]{%
10841     \glslink[##1]{##2}{%
10842         \glslongpluralaccessdisplay
10843         {\Glsentrylongpl{##2}}{##2}##3\space
10844         (\glsshortpluralaccessdisplay
10845         {\acronymfont{\glsentryshortpl{##2}}}{##2}})%
10846 \renewcommand*{\ACRfullplfmt}[3]{%
10847     \glslink[##1]{##2}{%
10848         \glslongpluralaccessdisplay
10849         {\mfirstucMakeUppercase{\glsentrylongpl{##2}}{##2}##3\space
10850         (\glsshortpluralaccessdisplay
10851         {\acronymfont{\glsentryshortpl{##2}}}{##2}})%
10852 \renewcommand*{\glsentryfull}[1]{%

```



```

10853 \glslongaccessdisplay{\glsentrylong{##1}}\space
10854 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})%
10855 }%
10856 \renewcommand*{\Glsentryfull}[1]{%
10857 \glslongaccessdisplay{\Glsentrylong{##1}}{##1}\space
10858 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})%
10859 }%
10860 \renewcommand*{\Glsentryfullpl}[1]{%
10861 \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}\space
10862 (\glsshortpluralaccessdisplay{\acronymfont{\glsentryshortpl{##1}}}{##1})%
10863 }%
10864 \renewcommand*{\Glsentryfullpl}[1]{%
10865 \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}\space
10866 (\glsshortpluralaccessdisplay{\acronymfont{\glsentryshortpl{##1}}}{##1})%
10867 }%
10868 \renewcommand*{\acronymentry}[1]{%
10869 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}%
10870 \renewcommand*{\acronymsort}[2]{##1}%
10871 \renewcommand*{\acronymfont}[1]{##1}%
10872 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
10873 }

```

dua-desc <long> only acronym style with user-supplied description.

```

10874 \renewacronymstyle{dua-desc}%
10875 {%
10876 \GlsUseAcrEntryDispStyle{dua}%
10877 }%
10878 {%
10879 \GlsUseAcrStyleDefs{dua}%
10880 \renewcommand*{\GenericAcronymFields}{}%
10881 \renewcommand*{\acronymentry}[1]{%
10882 \glslongaccessdisplay{\acronymfont{\glsentrylong{##1}}}{##1}}%
10883 \renewcommand*{\acronymsort}[2]{##2}%
10884 }%

```

footnote <short>\footnote{<long>} acronym style.

```

10885 \renewacronymstyle{footnote}%
10886 {%
10887 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
10888 }%
10889 {%
10890 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%

```

Need to ensure hyperlinks are switched off on first use:

```

10891 \glshyperfirstfalse
10892 \renewcommand*{\genacrfullformat}[2]{%
10893 \glsshortaccessdisplay
10894 {\protect\firstacronymfont{\glsentryshort{##1}}}{##1}##2%

```

```

10895 \protect\footnote{\glslongaccessdisplay{\glsentrylong{##1}}{##1}}%
10896 }%
10897 \renewcommand*{\Genacrfullformat}[2]{%
10898 \glsshortaccessdisplay
10899   {\firstacronymfont{\Glsentryshort{##1}}{##1}##2%
10900 \protect\footnote{\glslongaccessdisplay{\glsentrylong{##1}}{##1}}%
10901 }%
10902 \renewcommand*{\genplacrfullformat}[2]{%
10903 \glsshortpluralaccessdisplay
10904   {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}##2%
10905 \protect\footnote{\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}}%
10906 }%
10907 \renewcommand*{\Genplacrfullformat}[2]{%
10908 \glsshortpluralaccessdisplay
10909   {\protect\firstacronymfont{\Glsentryshortpl{##1}}{##1}##2%
10910 \protect\footnote{\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}}%
10911 }%
10912 \renewcommand*{\acronymentry}[1]{%
10913 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}{##1}}%
10914 \renewcommand*{\acronymsort}[2]{##1}%
10915 \renewcommand*{\acronymfont}[1]{##1}%
10916 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%

```

Don't use footnotes for \acrfull:

```

10917 \renewcommand*{\acrfullfmt}[3]{%
10918 \glslink[##1]{##2}{%
10919 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}{##2}##3\space
10920 (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}}%
10921 \renewcommand*{\Acrfullfmt}[3]{%
10922 \glslink[##1]{##2}{%
10923 \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##2}}{##2}##3\space
10924 (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}}%
10925 \renewcommand*{\ACRfullfmt}[3]{%
10926 \glslink[##1]{##2}{%
10927 \glsshortaccessdisplay
10928   {\mfirstucMakeUppercase
10929   {\acronymfont{\glsentryshort{##2}}{##2}##3\space
10930   (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}}%
10931 \renewcommand*{\acrfullplfmt}[3]{%
10932 \glslink[##1]{##2}{%
10933 \glsshortpluralaccessdisplay
10934   {\acronymfont{\glsentryshortpl{##2}}{##2}##3\space
10935   (\glslongpluralaccessdisplay{\glsentrylongpl{##2}}{##2})}}}%
10936 \renewcommand*{\Acrfullplfmt}[3]{%
10937 \glslink[##1]{##2}{%
10938 \glsshortpluralaccessdisplay
10939   {\acronymfont{\Glsentryshortpl{##2}}{##2}##3\space
10940   (\glslongpluralaccessdisplay{\glsentrylongpl{##2}}{##2})}}}%
10941 \renewcommand*{\ACRfullplfmt}[3]{%
10942 \glslink[##1]{##2}{%

```

```

10943 \glsshortpluralaccessdisplay
10944 {\mfirstucMakeUppercase
10945 {\acronymfont{\glentryshortpl{##2}}{##2}##3\space
10946 (\glslongpluralaccessdisplay{\glentrylongpl{##2}}{##2}}}%

```

Similarly for \glentryfull etc:

```

10947 \renewcommand*{\glentryfull}[1]{%
10948 \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}{##1}\space
10949 (\glslongaccessdisplay{\glentrylong{##1}}{##1}}}%
10950 \renewcommand*{\Glsentryfull}[1]{%
10951 \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##1}}{##1}\space
10952 (\glslongaccessdisplay{\glentrylong{##1}}{##1}}}%
10953 \renewcommand*{\glentryfullpl}[1]{%
10954 \glsshortpluralaccessdisplay
10955 {\acronymfont{\glentryshortpl{##1}}{##1}\space
10956 (\glslongpluralaccessdisplay{\glentrylongpl{##1}}{##1}}}%
10957 \renewcommand*{\Glsentryfullpl}[1]{%
10958 \glsshortpluralaccessdisplay
10959 {\acronymfont{\Glsentryshortpl{##1}}{##1}\space
10960 (\glslongpluralaccessdisplay{\glentrylongpl{##1}}{##1}}}%
10961 }

```

footnote-sc \textsc{<short>}\footnote{<long>} acronym style.

```

10962 \renewacronymstyle{footnote-sc}%
10963 {%
10964 \GlsUseAcrEntryDispStyle{footnote}%
10965 }%
10966 {%
10967 \GlsUseAcrStyleDefs{footnote}%
10968 \renewcommand{\acronymentry}[1]{%
10969 \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}{##1}}
10970 \renewcommand{\acronymfont}[1]{\textsc{##1}}%
10971 \renewcommand*{\acrpluralsuffix}{\glstextup{\glspluralsuffix}}%
10972 }%

```

footnote-sm \textsmaller{<short>}\footnote{<long>} acronym style.

```

10973 \renewacronymstyle{footnote-sm}%
10974 {%
10975 \GlsUseAcrEntryDispStyle{footnote}%
10976 }%
10977 {%
10978 \GlsUseAcrStyleDefs{footnote}%
10979 \renewcommand{\acronymentry}[1]{%
10980 \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}{##1}}
10981 \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
10982 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
10983 }%

```

footnote-desc <short>\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

10984 \renewacronymstyle{footnote-desc}%
10985 {%
10986   \GlsUseAcrEntryDisplayStyle{footnote}%
10987 }%
10988 {%
10989   \GlsUseAcrStyleDefs{footnote}%
10990   \renewcommand*{\GenericAcronymFields}{}%
10991   \renewcommand*{\acronymsort}[2]{##2}%
10992   \renewcommand*{\acronymentry}[1]{%
10993     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10994     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10995 }

```

ootnote-sc-desc \textsc{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

10996 \renewacronymstyle{footnote-sc-desc}%
10997 {%
10998   \GlsUseAcrEntryDisplayStyle{footnote-sc}%
10999 }%
11000 {%
11001   \GlsUseAcrStyleDefs{footnote-sc}%
11002   \renewcommand*{\GenericAcronymFields}{}%
11003   \renewcommand*{\acronymsort}[2]{##2}%
11004   \renewcommand*{\acronymentry}[1]{%
11005     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11006     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11007 }

```

ootnote-sm-desc \textsmaller{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

11008 \renewacronymstyle{footnote-sm-desc}%
11009 {%
11010   \GlsUseAcrEntryDisplayStyle{footnote-sm}%
11011 }%
11012 {%
11013   \GlsUseAcrStyleDefs{footnote-sm}%
11014   \renewcommand*{\GenericAcronymFields}{}%
11015   \renewcommand*{\acronymsort}[2]{##2}%
11016   \renewcommand*{\acronymentry}[1]{%
11017     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11018     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11019 }

```

Use \newacronymhook to modify the key list to set the access text to the long version by default.

```

11020 \renewcommand*{\newacronymhook}{%
11021   \edef\@gls@keylist{shortaccess=\the\glslongtok,%
11022     \the\glskeylisttok}%
11023   \expandafter\glskeylisttok\expandafter{\@gls@keylist}%

```

11024 }

1tNewAcronymDef Modify default style to use access text:

```
11025 \renewcommand*{\DefaultNewAcronymDef}{%
11026   \edef\@do@newglossaryentry{%
11027     \noexpand\newglossaryentry{\the\glslabeltok}%
11028     {%
11029       type=\acronymtype,%
11030       name={\the\glsshorttok},%
11031       description={\the\glslongtok},%
11032       descriptionaccess=\relax,%
11033       text={\the\glsshorttok},%
11034       access={\noexpand\@glo@textaccess},%
11035       sort={\the\glsshorttok},%
11036       short={\the\glsshorttok},%
11037       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11038       shortaccess={\the\glslongtok},%
11039       long={\the\glslongtok},%
11040       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11041       descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11042       first={\noexpand\glslongaccessdisplay
11043         {\the\glslongtok}{\the\glslabeltok}\space
11044         (\noexpand\glsshortaccessdisplay
11045           {\the\glsshorttok}{\the\glslabeltok})},%
11046       plural={\the\glsshorttok\acrpluralsuffix},%
11047       firstplural={\noexpand\glslongpluralaccessdisplay
11048         {\noexpand\@glo@longpl}{\the\glslabeltok}\space
11049         (\noexpand\glsshortpluralaccessdisplay
11050           {\noexpand\@glo@shortpl}{\the\glslabeltok})},%
11051       firstaccess=\relax,%
11052       firstpluralaccess=\relax,%
11053       textaccess={\noexpand\@glo@shortaccess},%
11054       \the\glskeylisttok
11055     }%
11056   }%
11057   \let\@org@gls@assign@firstpl\gls@assign@firstpl
11058   \let\@org@gls@assign@plural\gls@assign@plural
11059   \let\@org@gls@assign@descplural\gls@assign@descplural
11060   \def\gls@assign@firstpl##1##2{%
11061     \@gls@expand@field{##1}{firstpl}{##2}%
11062   }%
11063   \def\gls@assign@plural##1##2{%
11064     \@gls@expand@field{##1}{plural}{##2}%
11065   }%
11066   \def\gls@assign@descplural##1##2{%
11067     \@gls@expand@field{##1}{descplural}{##2}%
11068   }%
11069   \@do@newglossaryentry
11070   \let\gls@assign@firstpl\@org@gls@assign@firstpl
```

```

11071 \let\gls@assign@plural\@org@gls@assign@plural
11072 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11073 }

```

teNewAcronymDef

```

11074 \renewcommand*{\DescriptionFootnoteNewAcronymDef}{%
11075 \edef\@do@newglossaryentry{%
11076 \noexpand\newglossaryentry{\the\glslabeltok}%
11077 {%
11078 type=\acronymtype,%
11079 name={\noexpand\acronymfont{\the\glsshorttok}},%
11080 sort={\the\glsshorttok},%
11081 text={\the\glsshorttok},%
11082 short={\the\glsshorttok},%
11083 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11084 shortaccess={\the\glslongtok},%
11085 long={\the\glslongtok},%
11086 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11087 access={\noexpand\@glo@textaccess},%
11088 plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11089 symbol={\the\glslongtok},%
11090 symbolplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11091 firstpluralaccess=\relax,
11092 textaccess={\noexpand\@glo@shortaccess},%
11093 \the\glskeylisttok
11094 }%
11095 }%
11096 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11097 \let\@org@gls@assign@plural\gls@assign@plural
11098 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
11099 \def\gls@assign@firstpl##1##2{%
11100 \@@gls@expand@field{##1}{firstpl}{##2}%
11101 }%
11102 \def\gls@assign@plural##1##2{%
11103 \@@gls@expand@field{##1}{plural}{##2}%
11104 }%
11105 \def\gls@assign@symbolplural##1##2{%
11106 \@@gls@expand@field{##1}{symbolplural}{##2}%
11107 }%
11108 \do@newglossaryentry
11109 \let\gls@assign@plural\@org@gls@assign@plural
11110 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11111 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11112 }

```

onNewAcronymDef

```

11113 \renewcommand*{\DescriptionNewAcronymDef}{%
11114 \edef\@do@newglossaryentry{%
11115 \noexpand\newglossaryentry{\the\glslabeltok}%

```

```

11116 {%
11117     type=\acronymtype,%
11118     name={\noexpand
11119         \acrnameformat{\the\glsshorttok}{\the\glslongtok}},%
11120     access={\noexpand\@glo@textaccess},%
11121     sort={\the\glsshorttok},%
11122     short={\the\glsshorttok},%
11123     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11124     shortaccess={\the\glslongtok},%
11125     long={\the\glslongtok},%
11126     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11127     first={\the\glslongtok},%
11128     firstaccess=\relax,
11129     firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11130     text={\the\glsshorttok},%
11131     textaccess={\the\glslongtok},%
11132     plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11133     symbol={\noexpand\@glo@text},%
11134     symbolaccess={\noexpand\@glo@textaccess},%
11135     symbolplural={\noexpand\@glo@plural},%
11136     firstpluralaccess=\relax,
11137     textaccess={\noexpand\@glo@shortaccess},%
11138     \the\glskeylisttok}%
11139 }%
11140 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11141 \let\@org@gls@assign@plural\gls@assign@plural
11142 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
11143 \def\gls@assign@firstpl##1##2{%
11144     \@gls@expand@field{##1}{firstpl}{##2}%
11145 }%
11146 \def\gls@assign@plural##1##2{%
11147     \@gls@expand@field{##1}{plural}{##2}%
11148 }%
11149 \def\gls@assign@symbolplural##1##2{%
11150     \@gls@expand@field{##1}{symbolplural}{##2}%
11151 }%
11152 \@do@newglossaryentry
11153 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11154 \let\gls@assign@plural\@org@gls@assign@plural
11155 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11156 }

```

teNewAcronymDef

```

11157 \renewcommand*{\FootnoteNewAcronymDef}{%
11158     \edef\@do@newglossaryentry{%
11159         \noexpand\newglossaryentry{\the\glslabeltok}%
11160         {%
11161             type=\acronymtype,%
11162             name={\noexpand\acronymfont{\the\glsshorttok}},%

```

```

11163     sort={\the\glsshorttok},%
11164     text={\the\glsshorttok},%
11165     textaccess={\the\glslongtok},%
11166     access={\noexpand\@glo@textaccess},%
11167     plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11168     short={\the\glsshorttok},%
11169     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11170     long={\the\glslongtok},%
11171     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11172     description={\the\glslongtok},%
11173     descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11174     \the\glskeylisttok
11175   }%
11176 }%
11177 \let\@org@gls@assign@plural\gls@assign@plural
11178 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11179 \let\@org@gls@assign@descplural\gls@assign@descplural
11180 \def\gls@assign@firstpl##1##2{%
11181   \@@gls@expand@field{##1}{firstpl}{##2}%
11182 }%
11183 \def\gls@assign@plural##1##2{%
11184   \@@gls@expand@field{##1}{plural}{##2}%
11185 }%
11186 \def\gls@assign@descplural##1##2{%
11187   \@@gls@expand@field{##1}{descplural}{##2}%
11188 }%
11189 \do@newglossaryentry
11190 \let\gls@assign@plural\@org@gls@assign@plural
11191 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11192 \let\gls@assign@descplural\@org@gls@assign@descplural
11193 }

```

11NewAcronymDef

```

11194 \renewcommand*{\SmallNewAcronymDef}{%
11195   \edef\@do@newglossaryentry{%
11196     \noexpand\newglossaryentry{\the\glslabeltok}%
11197     {%
11198       type=\acronymtype,%
11199       name={\noexpand\acronymfont{\the\glsshorttok}},%
11200       access={\noexpand\@glo@symbolaccess},%
11201       sort={\the\glsshorttok},%
11202       short={\the\glsshorttok},%
11203       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11204       shortaccess={\the\glslongtok},%
11205       long={\the\glslongtok},%
11206       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11207       text={\noexpand\@glo@short},%
11208       textaccess={\noexpand\@glo@shortaccess},%
11209       plural={\noexpand\@glo@shortpl},%

```



```

11210     first={\the\glslongtok},%
11211     firstaccess=\relax,
11212     firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11213     description={\noexpand\@glo@first},%
11214     descriptionplural={\noexpand\@glo@firstplural},%
11215     symbol={\the\glsshorttok},%
11216     symbolaccess={\the\glslongtok},%
11217     symbolplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11218     \the\glskeylisttok
11219 }%
11220 }%
11221 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11222 \let\@org@gls@assign@plural\gls@assign@plural
11223 \let\@org@gls@assign@descplural\gls@assign@descplural
11224 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
11225 \def\gls@assign@firstpl##1##2{%
11226   \@@gls@expand@field{##1}{firstpl}{##2}%
11227 }%
11228 \def\gls@assign@plural##1##2{%
11229   \@@gls@expand@field{##1}{plural}{##2}%
11230 }%
11231 \def\gls@assign@descplural##1##2{%
11232   \@@gls@expand@field{##1}{descplural}{##2}%
11233 }%
11234 \def\gls@assign@symbolplural##1##2{%
11235   \@@gls@expand@field{##1}{symbolplural}{##2}%
11236 }%
11237 \do@newglossaryentry
11238 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11239 \let\gls@assign@plural\@org@gls@assign@plural
11240 \let\gls@assign@descplural\@org@gls@assign@descplural
11241 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11242 }

```

The following are kept for compatibility with versions before 3.0:

sshortaccesskey

```

11243 \newcommand*{\glsshortaccesskey}{\glsshortkey access}%

```

pluralaccesskey

```

11244 \newcommand*{\glsshortpluralaccesskey}{\glsshortpluralkey access}%

```

lslongaccesskey

```

11245 \newcommand*{\glslongaccesskey}{\glslongkey access}%

```

pluralaccesskey

```

11246 \newcommand*{\glslongpluralaccesskey}{\glslongpluralkey access}%

```

5.5 Debugging Commands

owglonameaccess

```
11247 \newcommand*{\showglonameaccess}[1]{%
11248   \expandafter\show\csname glo@\glsdetoklabel{#1}@textaccess\endcsname
11249 }
```

owglotextaccess

```
11250 \newcommand*{\showglotextaccess}[1]{%
11251   \expandafter\show\csname glo@\glsdetoklabel{#1}@textaccess\endcsname
11252 }
```

glopluralaccess

```
11253 \newcommand*{\showglopluralaccess}[1]{%
11254   \expandafter\show\csname glo@\glsdetoklabel{#1}@pluralaccess\endcsname
11255 }
```

wglofirstaccess

```
11256 \newcommand*{\showglofirstaccess}[1]{%
11257   \expandafter\show\csname glo@\glsdetoklabel{#1}@firstaccess\endcsname
11258 }
```

rstpluralaccess

```
11259 \newcommand*{\showglofirstpluralaccess}[1]{%
11260   \expandafter\show\csname glo@\glsdetoklabel{#1}@firstpluralaccess\endcsname
11261 }
```

glosymbolaccess

```
11262 \newcommand*{\showglosymbolaccess}[1]{%
11263   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolaccess\endcsname
11264 }
```

bolpluralaccess

```
11265 \newcommand*{\showglosymbolpluralaccess}[1]{%
11266   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolpluralaccess\endcsname
11267 }
```

owglodescaccess

```
11268 \newcommand*{\showglodescaccess}[1]{%
11269   \expandafter\show\csname glo@\glsdetoklabel{#1}@descaccess\endcsname
11270 }
```

escpluralaccess

```
11271 \newcommand*{\showglodescpluralaccess}[1]{%
11272   \expandafter\show\csname glo@\glsdetoklabel{#1}@descpluralaccess\endcsname
11273 }
```

wgloshortaccess

```
11274 \newcommand*{\showgloshortaccess}[1]{%  
11275   \expandafter\show\csname glo@\glsdetoklabel{#1}@shortaccess\endcsname  
11276 }
```

ortpluralaccess

```
11277 \newcommand*{\showgloshortpluralaccess}[1]{%  
11278   \expandafter\show\csname glo@\glsdetoklabel{#1}@shortpluralaccess\endcsname  
11279 }
```

owglolongaccess

```
11280 \newcommand*{\showglolongaccess}[1]{%  
11281   \expandafter\show\csname glo@\glsdetoklabel{#1}@longaccess\endcsname  
11282 }
```

ongpluralaccess

```
11283 \newcommand*{\showglolongpluralaccess}[1]{%  
11284   \expandafter\show\csname glo@\glsdetoklabel{#1}@longpluralaccess\endcsname  
11285 }
```

6 Multi-Lingual Support

Many thanks to everyone who contributed to the translations both via email and on comp.text.tex. Language support has now been split off into independent language modules.

```
11286 \NeedsTeXFormat{LaTeX2e}
11287 \ProvidesPackage{glossaries-babel}[2016/04/19 v4.22 (NLCT)]
```

Load tracklang to obtain language settings.

```
11288 \RequirePackage{tracklang}
11289 \let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
11290 \AnyTrackedLanguages
11291 {%
11292   \ForEachTrackedDialect{\this@dialect}{%
11293     \IfTrackedLanguageFileExists{\this@dialect}%
11294       {glossaries-}% prefix
11295       {.ldf}%
11296       {%
11297         \RequireGlossariesLang{\CurrentTrackedTag}%
11298       }%
11299       {%
11300         \PackageWarningNoLine{glossaries}%
11301           {No language module detected for ‘\this@dialect’.\MessageBreak
11302             Language modules need to be installed separately.\MessageBreak
11303             Please check on CTAN for a bundle called\MessageBreak
11304             ‘glossaries-\CurrentTrackedLanguage’ or similar}%
11305       }%
11306     }%
11307   }%
11308 }
```

6.1 Polyglossia Captions

Language support has now been split off into independent language modules.

```
11309 \NeedsTeXFormat{LaTeX2e}
11310 \ProvidesPackage{glossaries-polyglossia}[2016/04/19 v4.22 (NLCT)]
```

Load tracklang to obtain language settings.

```
11311 \RequirePackage{tracklang}
11312 \let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
11313 \AnyTrackedLanguages
```

```

11314 {%
11315     \ForEachTrackedDialect{\this@dialect}{%
11316         \IfTrackedLanguageFileExists{\this@dialect}%
11317         {glossaries-}% prefix
11318         {.ldf}%
11319         {%
11320             \RequireGlossariesLang{\CurrentTrackedTag}%
11321         }%
11322         {%
11323             \PackageWarningNoLine{glossaries}%
11324             {No language module detected for ‘\this@dialect’.\MessageBreak
11325             Language modules need to be installed separately.\MessageBreak
11326             Please check on CTAN for a bundle called\MessageBreak
11327             ‘glossaries-\CurrentTrackedLanguage’ or similar}%
11328         }%
11329     }%
11330 }%
11331 {}%

```

Glossary

`makeindex` An indexing application. [9](#), [24](#), [25](#), [167](#)

`xindy` An flexible indexing application with multilingual support written in Perl. [9](#), [24](#), [25](#), [167](#)

Change History

1.01 (2007-05-17)	numberline: numberline option added .. 5
General: Added range facility in format key 107	1.12 (2008-03-08)
\writeist: Added spaces after \delimN and \delimR in ist file 153	\@GLSpl: now uses \glentrydescplural and \glentrysymbolplural instead of \glentrydesc and \glentrysymbol 120
1.04 (2007-08-03)	\@Glspl@: now uses \glentrydescplural and \glentrysymbolplural instead of \glentrydesc and \glentrysymbol 120
1.05 (2007-08-10)	\@Glspl@: now uses \glentrydescplural and \glentrysymbolplural instead of \glentrydesc and \glentrysymbol 119
\glossarysection: added \@mkboth to \glossarysection 36	General: added check for \hypertarget separate to \hyperlink (memoir defines \hyperlink but not \hypertarget) 115
\gls@defglossaryentry: Changed the default value of the sort key to just the value of the name key 75	descriptionplural: new 59
1.07 (2007-09-13)	\gls@defglossaryentry: Changed default first plural to be first key with s appended (was text key with s appended) 75
\@gls@link: fixed bug caused by \theglentrycounter setting the page number too soon 105	descriptionplural support added 75
\glsadd: fixed bug caused by \theglentrycounter setting the page number too soon 151	symbolplural support added 75
1.08 (2007-10-13)	\Glsentrydescplural: New 144
General: Added babel support 30	\glentrydescplural: New 144
listgroup: changed listgroup style to use \glsgrouptitle 260	\Glsentrysymbolplural: New 145
altlistgroup: changed altlistgroup style to use \glsgrouptitle 261	\glentrysymbolplural: New 145
1.1 (2008-02-22)	\SetDescriptionFootnoteAcronymStyle: Added \protect before \footnote and \glslink 227
\@glossarysection: numbered sections and auto label added 37	\SetFootnoteAcronymStyle: Added \protect before \footnote and \glslink 233
\@gls@tmpb: changed \toksdef to \newtoks 109	symbolplural: new 60
\@gls@toc: numberline added 39	1.13 (2008-05-10)
\@p@glossarysection: numbered sections and auto label added 38	General: fixed bug that ignored 3rd parameter 122–129
General: amsgen now loaded (\new@ifnextchar needed) 4	\ACRfullpl: new 208
translate: translate option added 21	
\setglossarysection: new 37	
numberedsection: numberedsection package option added 6	

<code>\Acrfullpl: new</code>	208	<code>\@gls@: Test glossary type is \acronymtype</code> in addition to checking if footnote op- tion has been used	116
<code>\acrfullpl: new</code>	207	<code>\@glsdisp: Test glossary type is</code> <code>\acronymtype</code> in addition to check- ing if footnote option has been used	121
<code>\acrpluralsuffix: New</code>	205	<code>\@glspl@: Test glossary type is</code> <code>\acronymtype</code> in addition to check- ing if footnote option has been used	119
<code>\gls@defglossaryentry: Changed de-</code> fault first value	75	<code>\@glstarget: raised the hypertarget so</code> the target text doesn't scroll off the top of the page	115
Changed default firstplural value	75	<code>\gls@defglossaryentry: Changed def</code> to let	75
Removed restriction on only using <code>\newglossaryentry</code> in the preamble	80		
<code>\newacronym: Removed restriction on</code> only using <code>\newacronym</code> in the pream- ble	205		
1.14 (2008-06-17)		1.17 (2008-12-26)	
<code>\@gls@hypergroup: new</code>	256	<code>\@@do@wrglossary: new</code>	171
General: added nonnumberlist key to <code>\printglossary</code>	191	<code>\@do@seeglossary: new</code>	173
added numberedsection key to <code>\printglossary</code>	190	<code>\@glo@storeentry: new</code>	81
<code>\firstacronymfont: new</code>	209	<code>\@gls@glossary: changed definition to</code> use <code>\index</code> instead of <code>\@index</code>	168
<code>\glsautoprefix: new</code>	6	<code>\@glsdefaultplural: new</code>	62
<code>\glsnavhyperlink: changed \edef to</code> <code>\protected@edef</code>	255	<code>\@glsdefaultsort: new</code>	63
<code>\glsnavhypertarget: added write to aux</code> file	255	<code>\@glshypernumber: new</code>	202
<code>\glsnavigation: changed to only use la-</code> bels for groups that are present	256	<code>\@glsnoname: new</code>	62
1.15 (2008-08-15)		<code>\@glsnonextpages: new</code>	192
<code>\@gls@link: added \glslabel</code>	105	General: added xindy support	24
<code>\gls@defglossaryentry: check for</code> <code>\@glo@first</code> in description	79	parent: new	61
check for <code>\@glo@text</code> in symbol	79	see: new	60
<code>\gls@hypergroup: new</code>	256	<code>\gls@defglossaryentry: added non-</code> umberlist key	76
<code>\glsnavhypertarget: added check if re-</code> run required	255	added parent key	75
<code>\glssettoctitle: new</code>	29	added see key	75
<code>\printglossary: changed the way the</code> TOC title is set	176	Stored main part of entry format when entry is defined	80
1.16 (2008-08-27)		<code>\gls@suffixF: new</code>	34
<code>\@GLS@: Test glossary type is \acronymtype</code> in addition to checking if footnote op- tion has been used	118	<code>\gls@suffixFF: new</code>	35
<code>\@GLSpl: Test glossary type is</code> <code>\acronymtype</code> in addition to check- ing if footnote option has been used	120	<code>\gls@wrglossary: modified to allow for</code> xindy support	168
<code>\@Gls@: Test glossary type is \acronymtype</code> in addition to checking if footnote op- tion has been used	117	<code>\glshyperlink: new</code>	150
<code>\@Glspl@: Test glossary type is</code> <code>\acronymtype</code> in addition to check- ing if footnote option has been used	120	<code>\glshypernumber: modified to allow ma-</code> terial to be attached to location	202
		<code>\glsnavhyperlink: replaced \hyperlink</code> to <code>\@glslink</code>	255
		<code>\glsnavhypertarget: replaced</code> <code>\hypertarget</code> to <code>\@glstarget</code> ...	255
		<code>\glssee: new</code>	174
		<code>\glsseeformat: new</code>	174
		<code>\glsSetSuffixF: new</code>	34
		<code>\glsSetSuffixFF: new</code>	35

\ifglxsindy: new	24	before term is displayed to prevent unwanted whatsit	105
\istfilename: added xindy support ...	33		
\newglossarystyle: made \newglossarystyle long	201	\foralllglossaries: replaced \ifthenelse with \ifx	48
\nopostdesc: new	32	\forallgsentries: replaced \ifthenelse with \ifx	48
nonumberlist: new	61	\glsgdefmain: new	12
\printglossary: added check to determine if \printglossary is already defined	176	\glsgdescwidth: changed \linewidth to \hsize	263, 284
added print language to aux file	176	\glsglistdottedwidth: changed \linewidth to \hsize	262
order: order package option added	24	\glsgpagelistwidth: changed \linewidth to \hsize	263, 284
\writeist: added xindy support	153	nomain: added nomain package option .	12
1.18 (2009-01-14)		\writeist: removed item_02 - no such makeindex key	157
\@glsgloadlist: new	8	2.02 (2007-07-13)	
\@glsgloadlong: new	7	\@printglossary: suppressed warning globally rather than locally	178
\@glsgloadsuper: new	7	2.02 (2009-07-13)	
\@glsgloadtree: new	8	\@glossarysection: changed \@mkboth to \glossarymark	36
\glsgdefglossaryentry: Changed default value of sort to \@glsgdefaultsort	75	\glsgglossarymark: New	36
moved sort sanitization to \newglossaryentry	79	2.03 (2009-09-23)	
\glsgtarget: new	195	\@GLSG: Added check for hyperfirst	118
\oldacronym: new	204	\@GLSGpl: Added check for hyperfirst ...	120
nolist: new	8	\@GLSG: Added check for hyperfirst	117
nolong: new	7	\@GLSGpl: Added check for hyperfirst ..	120
sort: moved sanitization to \newglossaryentry	59	\@glsg: Added check for hyperfirst	116
nostyles: new	8	\@glsg@link: new	103
nosuper: new	8	\@glsg@link: added \leavevmode	105
notree: new	8	Moved entry existence check to avoid duplicate code	105
1.19 (2009-03-02)		\@glsgdisp: Added check for hyperfirst .	121
\glsgclearpage: new	39	\@glsgpl: Added check for hyperfirst ..	119
\glsgdisp: new	121	\glsgglossarymark: Added check to see if it's already defined	36
\SetDescriptionAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	231	hyperfirst: new	23
\SetDescriptionFootnoteAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	227	2.04 (2009-11-10)	
\SetFootnoteAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	233	\@GLSG: Changed test to check if glossary type has been identified as a list of acronyms	118
\SetSmallAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	236	\@GLSGpl: Changed test to check if glossary type has been identified as a list of acronyms	120
2.01 (2009 May 30)		\@GLSG: Changed test to check if glossary type has been identified as a list of acronyms	117
\@glsg@link: moved \@do@wrglossary			

\@Glspl@: Changed test to check if glossary type has been identified as a list of acronyms	120	\SetDUADisplayStyle: new	237
\@glossaryentryfield: new	81	\SetFootnoteAcronymDisplayStyle: new	232
\@glossarysubentryfield: new	81	\SetSmallAcronymDisplayStyle: new	234
\@gls@: Changed test to check if glossary type has been identified as a list of acronyms	116	2.05 (2010-02-06)	
\@glsacronymlists: new	13	\@glsdisp: Added closing brace. Patch provided by Sergiu Dotenco	121
\@glsdisp: Changed test to check if glossary type has been identified as a list of acronyms	121	Removed spurious brace. Patch provided by Sergiu Dotenco	121
\@glspl@: Changed test to check if glossary type has been identified as a list of acronyms	119	\writeist: Added \string before opening and closing braces. Patch provided by Segiu Dotenco	158
\@newglossaryentryposthook: new ..	80	2.06 (2010-06-14)	
\@newglossaryentryprehook: new ..	80	\altnewglossary: new	56
acronymlists: new	15	\CustomAcronymFields: new	239
\DeclareAcronymList: new	14	\CustomNewAcronymDef: new	239
\DefineAcronymSynonyms: new	222	\SetCustomDisplayStyle: new	239
\gls@defglossaryentry: added userl-6 keys	76	\SetCustomStyle: new	240
\glsadd: fixed bug that ignored counter	151	2.07 (2010-07-10)	
\Glsentryuseri: new	146	General: glsadd format key stored in \@glsnumberformat (was mistakenly stored in \@glo@format)	150
\glsentryuseri: new	146	3.0 (2010-07-12)	
\Glsentryuserii: new	147	\@makeglossary: Added check for savewrites	159
\glsentryuserii: new	146	\gls@wrglossary: modified to take into account savewrites	168
\Glsentryuseriii: new	147	3.0 (2010/03/31)	
\glsentryuseriii: new	147	\@set@glo@numformat: added 4th argument	107
\Glsentryuseriv: new	147	3.0 (2011-04-02)	
\glsentryuseriv: new	147	\@do@wrglossary: added check for hyper location prefix	171
\Glsentryuserv: new	147	modified to use new format	171
\glsentryuserv: new	147	\@glossarysec: replaced \@ifundefined with \ifcsundef	5
\Glsentryuservi: new	147	\@do@seeglossary: Sanitize and escape cross-referencing information	173
\glsentryuservi: new	147	\@gls@counterwithin: new	9
\ns@newglossary: added check to determine if \gls@{type}@display and \gls@{type}@displayfirst have been defined.	56	\@gls@ifinlist: new	40
\SetAcronymLists: new	15	\@gls@link: added \@gls@saveentrycounter	105
\SetDefaultAcronymDisplayStyle: new	224	added \@gls@setsort	105
\SetDefaultAcronymStyle: new	225	\@gls@saveentrycounter: new	106
\SetDescriptionAcronymDisplayStyle: new	229	\@gls@setupsort@def: new	10
\SetDescriptionDUAAcronymDisplayStyle: new	228	\@gls@setupsort@standard: new	10
\SetDescriptionFootnoteAcronymDisplayStyle: new	225	\@gls@setupsort@use: new	11
		\@gls@xdy@locationlist: new	43

\@glslink: replaced \@ifundefined with \ifcsundef	115	\glsentryshort: new	148
\@glsnextpages: new	192	\Glsentryshortpl: new	148
\@print@glossary: replaced \@ifundefined with \ifcsundef	179	\glsentryshortpl: new	148
\@printglossary: added \currentglossary	177	\glsgetgrouptitle: replaced \@ifundefined with \ifcsundef	199
added \glsnextpages	178	\gls glossarymark: replaced \@ifundefined with \ifcsundef	36
make toctitle default to title	177	\gls hyperlink: changed default from \glsentryname to \glsentrytext	150
\@xdyattributelist: new	39	\gls hypernumber: replaced \@ifundefined with \ifcsundef	202
General: added prefix to hyperlink	203	\gls numberformat: replaced \@ifundefined with \ifcsundef	35
etoolbox now loaded	4	\gls refentry: new	194
replaced \@ifundefined with \ifcsundef	28, 31, 101, 189	\gls resetsubentrycounter: new ...	193
\acrfootnote: new	225	\gls seeitem: hyperlink uses \glsseeitemformat instead of \glsentryname	175
\ACRfull: added starred version	207	\gls seeitemformat: new	175
\Acrfull: added starred version	206	\gls sortnumberfmt: new	10
\acrfull: added starred version	206	\gls stepentry: new	193
\ACRfullpl: added starred version ...	208	\gls steps subentry: new	194
\Acrfullpl: added starred version ...	208	\gls subentrycounterlabel: new ...	194
\acrfullpl: added starred version ...	207	\gls subentryitem: new	195
\acrlinkfootnote: new	225	theglossary: replaced \@ifundefined with \ifcsundef	195
\acrno linkfootnote: new	225	short: new	62
save writes: new	25	shortplural: new	62
see: added \@glo@seeautonumberlist	60	\if glossaryexists: replaced \@ifundefined with \ifcsundef	49
seeautonumberlist: new	7	\if glsentryexists: replaced \@ifundefined with \ifcsundef	49
\glossarysection: replaced \@ifundefined with \ifcsundef	36	\istfile: deprecated	167
\glossarystyle: replaced \@ifundefined with \ifcsundef	200	glossaryentry: new	193
\gls@codepage: replaced \@ifundefined with \ifcsundef	24	glossarysubentry: new	193
\gls@def glossaryentry: added \@gls@defsort	79	\new glossaryentry: replaced \DeclareRobustCommand with \newrobustcmd	65
added short and long keys	76	\new glossarystyle: replaced \@ifundefined with \ifcsundef	201
replaced \@ifundefined with \ifcsundef	76	\ns@new glossary: added \@gls@defsortcount	56
\gls@docclearpage: replaced \@ifundefined with \ifcsundef	38	replaced \@ifundefined with \ifcsundef	56
\glsadd: added \@gls@saveentrycounter	151	entrycounter: new	9
\GlsAddXdyCounters: new	40	entrycounterwithin: new	9
\glsentrycounterlabel: new	194	\oldacronym: replaced \@ifundefined with \ifcsundef	204
\glsentryitem: new	194	compatible-2.07: compatible-2.07 op- tion added	26
\Glsentrylong: new	148	long: new	62
\glsentrylong: new	148		
\Glsentrylongpl: new	148		
\glsentrylongpl: new	148		
\Glsentryshort: new	148		

longplural: new	62	\Acrfull: made robust	206
nonumberlist: now boolean	61	\acrfull: made robust	206
sort: new	9	\acrfullformat: removed \acronymfont	
counter: replaced \@ifundefined with		as it should already be set in the sec-	
\ifcsundef	60	ond argument.	206
\printglossary: replaced \@ifundefined		\ACRfullpl: made robust	208
with \ifcsundef	176	\Acrfullpl: made robust	208
\SetDescriptionFootnoteAcronymDisplayStyle		\acrfullpl: made robust	207
expanded options link options	225	\ACRlong: made robust	139
\setentrycounter: added optional ar-		\Acrlong: made robust	138
gument	200	\acrlong: made robust	138
\showacronymlists: new	245	\ACRlongpl: made robust	141
\showglocounter: new	242	\Acrlongpl: made robust	140
\showglodesc: new	243	\acrlongpl: made robust	140
\showglodescplural: new	244	\ACRshort: made robust	135
\showglofirst: new	242	\Acrshort: made robust	135
\showglofirsttpl: new	242	\acrshort: made robust	134
\showgloflag: new	245	\ACRshortpl: made robust	137
\showgloindex: new	245	\Acrshortpl: made robust	136
\showglolevel: new	241	\acrshortpl: made robust	136
\showglongame: new	243	\Gls: made robust	117
\showgloparent: new	241	\glsadd: made robust	151
\showgloplural: new	241	\glsaddall: made robust	151
\showglosort: new	244	\GLSdesc: made robust	126
\showglossaries: new	245	\Glsdesc: made robust	126
\showglossarycounter: new	246	\glsdesc: made robust	126
\showglossaryentries: new	246	\GLSdescplural: made robust	127
\showglossaryin: new	246	\Glsdescplural: made robust	127
\showglossaryout: new	246	\glsdescplural: made robust	127
\showglossarytitle: new	246	\glsfirst: made robust	123
\showglosymbol: new	244	\GLSfirstplural: made robust	125
\showglosymbolplural: new	244	\Glsfirstplural: made robust	125
\showglotext: new	241	\glsfirstplural: made robust	124
\showglotype: new	242	\glslink: made robust	103
\showglouseri: new	242	\GLSname: made robust	126
\showglouserii: new	242	\Glsname: made robust	125
\showglouseriii: new	243	\glsname: made robust	125
\showglouseriv: new	243	\GLSpl: made robust	120
\showglouserv: new	243	\Glspl: made robust	119
\showglouservi: new	243	\glspl: made robust	118
subentrycounter: new	9	\GLSplural: made robust	124
\writeist: added xindy-only macro defi-		\GLSsymbol: made robust	128
nitions to glossary open tag	155	\Glsymbol: made robust	128
modified to support new format	153	\glssymbol: made robust	128
3.01 (2011-04-12)		\GLSsymbolplural: made robust	129
\@glswritefiles: added check for		\Glsymbolplural: made robust	129
empty glossaries	167	\glssymbolplural: made robust	128
General: made robust	118	\Glstext: made robust	122
\ACRfull: made robust	207	\glstext: made robust	122

\GLSuseri: made robust	130	\glstentryfullpl: fixed bug (re-	
\Glsuseri: made robust	130	placed \glstentryshort with	
\glsuseri: made robust	129	\glstentryshortpl)	149
\GLSuserii: made robust	131	\glstentrynumberlist: new	149
\Glsuserii: made robust	130	\glsmoveentry: new	80
\glsuserii: made robust	130	\glsresetsubentrycounter: new ...	193
\GLSuseriii: made robust	131	\ifglshaschildren: new	51
\Glsuseriii: made robust	131	\ifglshasparent: new	51
\glsuseriii: made robust	131	\makeglossaries: added list parser ..	162
\GLSuseriv: made robust	132	indexonlyfirst: new	23
\Glsuseriv: made robust	132	\renewglossarystyle: new	201
\glsuseriv: made robust	132	\showglossaryentries: fixed misspelt	
\GLSuserv: made robust	133	command	246
\Glsuserv: made robust	133	\SmallNewAcronymDef: fixed broken	
\glsuserv: made robust	132	short and long plural	235
\GLSuservi: made robust	134	3.03 (2012/09/21)	
\Glsuservi: made robust	134	\@gls@sanitizesort: new	17
\glsuservi: made robust	133	\@gls@setupsort@standard: used	
		\@gls@sanitizesort	10
3.02 (2012-05-19)		\@printglossary: allow title to override	
\glsnumlistlastsep: new	150	default toctitle	177
\glsnumlistsep: new	150	General: allow title to set toctitle	189
3.02 (2012-05-21)		\glsinlinedescformat: new	259
\@do@wrglossary: changed \@gls@loc		\glsinlineemptydescformat: new ..	259
ref to \theglstentrycounter	172	\glsinlinenameformat: new	259
\@do@wrglossary: changed \@do@wr@glossary		\glsinlinepostchild: new	259
to test for indexonlyfirst option; put		\glsinlinesubdescformat: new	259
old \@do@wr@glossary code into		\glsinlinesubnameformat: new	259
\@do@wrglossary	169	\glspostinline: replaced “.” with	
\@gls@missingnumberlist: new	63	\glspostdescription	259
\@gls@writefiles: added check		altlongragged4col: added check for	
for existence of token in case		glsnogroupskip	277
\makeglossaries has been		altsuperragged4col: added check for	
omitted	167	glsnogroupskip	295
\@printglossary: add a way to fetch		alttree: added check for glsnogroupskip	
current entry label	178	304
savenumberlist: new	7	index: added check for glsnogroupskip	298
ucmark: new	8	nogroupskip: new	8
\gls@defglossaryentry: added num-		long: added check for glsnogroupskip	264
berlist element	79	long3col: added check for glsnogroup-	
\gls@save@numberlist: new	175	skip	265
\gls@wrglossary: added check for glos-		long4col: added check for glsnogroup-	
sary file defined	169	skip	267
\glsdisplaynumberlist: new	149	longragged: added check for glsnogroup-	
\glstentrycounter: set default value ..	106	skip	274
\Glstentryfull: fixed bug (re-		longragged3col: added check for	
placed \glstentryshortpl with		glsnogroupskip	276
\glstentryshort)	148	nopostdot: new	8
		tree: added check for glsnogroupskip	299

treenoname: added check for glsnogroup-skip	300	mcolindexspannav: replaced '2' with \glsmcols	280
super: added check for glsnogroupskip	285	mcoltree: replaced '2' with \glsmcols	280
super3col: added check for glsnogroup-skip	286	mcoltreenoname: replaced '2' with \glsmcols	281
super4col: added check for glsnogroup-skip	288	mcoltreespannav: replaced '2' with \glsmcols	281
superragged: added check for glsnogroupskip	292	\gls@protected@pagefmts: added Roman to list	169
superragged3col: added check for glsnogroupskip	293	\gls@Romanpage: new	170
3.04 (2012-11-11)		\glsgetgrouplabel: fixed bug (typo in \equal)	199
altlist: replaced \newline with paragraph break	261	\nopostdesc: made robust	32
3.04 (2012-11-18)		3.05 (2013/04/21)	
\@do@wrglossary: changed \theglsentrycount back to \glslocref	172	\@gls@nohyperlist: new	15
\@do@wrglossary: modified to compensate for possible incorrect page number	171	\@gls@DeclareNoHyperList: new	15
\@gls@escbsdq: unsanitize \gls@numberpage, \gls@alphpage, \gls@Alphpage and \gls@romanpage	108	nohypertypes: new	15
\@print@glossary: Moved aux write to end of document to prevent unwanted whatsit occurring here.	179	3.06 (2013/06/17)	
General: Added check for doc package ...	4	\@xdy@main@language: Changed back to using \language	24
added datatool-base as a required package	4	\findrootlanguage: Obsoleted	46
added local key	102	3.07 (2013-07-05)	
\gls@Alphpage: new	169	\@gls@link: fixed bug that failed to find entry in list	105
\gls@alphpage: new	169	\glossarypreamble: modified to work with \setglossarypreamble	35
\gls@disablepagerefexpansion: new	169	\gls@docclearpage: added check for openright	38
\gls@numberpage: new	170	\glspostdescription: Added spacefactor code	8
\gls@protected@pagefmts: new	169	\GlsSetXdyCodePage: Added check for fontspec	47
\gls@romanpage: new	170	\SetDescriptionAcronymDisplayStyle: now using \glsdoparenifnotempty	229
\glsdefmain: added check for doc package	12	\setglossarypreamble: new	36
\glsorg@endtheglossary: new	5	3.08a (2013-08-30)	
\glsorg@theglossary: new	5	list: updated list style to use \glossentry and \subglossentry	260
\PrintChanges: new	5	listdotted: updated listdotted style to use \glossentry and \subglossentry	262
3.05 (2013-04-21)		altlist: updated altlist style to use \glossentry and \subglossentry	261
\@do@wrglossary: add Roman case. Fixed bugs in the else statements ..	171	inline: updated inline style to use \glossentry and \subglossentry	258
\@gls@link: added check for "nohypertypes"	105	3.08a (2013-09-28)	
mcolalttree: replaced '2' with \glsmcols	282	\@gls@storeentry: no longer need to check for special characters in any of the fields other than sort	81
mcolindex: replaced '2' with \glsmcols	279		

updated for \glossentry	82	tree: updated to use \glossentry and	
\@glossaryentryfield: switched to		\subglossentry	299
\glossentry	81	\setglossarystyle:new	200
\@glossarysubentryfield: switched to		\setglossentrycompatibility:new	197
\subglossentry	81	superragged: updated to use	
General: added nogroupskip key to		\glossentry and \subglossentry	291
\printglossary	190	3.09a (2013-10-09)	
removed definition of \@glossaryentryfield		\@gls@assign@symbolplural@field:	
.....	346	new	17
removed definition of \@glossarysubentryfield		\@gls@default@value: new	59
.....	346	\Glsentrydesc: made robust	144
\compatibleglossentry: new	195	\Glsentrydescplural: made robust ..	144
\compatiblesubglossentry: new ...	197	\Glsentryfirst: made robust	145
\glossaryentryfield: deprecated ...	197	\Glsentryfirstplural: made robust .	145
\Glossentrydesc: new	196	\Glsentryfull: made robust	148
\glossentrydesc: new	196	\Glsentryfullpl: made robust	149
\Glossentryname: new	196	\Glsentrylong: made robust	148
\glossentryname: new	196	\Glsentrylongpl: made robust	148
\Glossentrysymbol: new	197	\Glsentryname: made robust	143
\glossentrysymbol: new	196	\Glsentryplural: made robust	144
\gls@assign@desc@field: new	17	\Glsentryshort: made robust	148
\gls@assign@descplural@field: new	17	\Glsentryshortpl: made robust	148
\gls@assign@field: new	65	\Glsentrysymbol: made robust	145
\gls@ifnotmeasuring: new	83	\Glsentrysymbolplural: made robust	145
\glsaddallunused: new	151	\Glsentrytext: made robust	144
\glsexpandfields: new	65	\Glsentryuseri: made robust	146
\glsnoexpandfields: new	65	\Glsentryuserii: made robust	147
\glssee: made robust	174	\Glsentryuseriii: made robust	147
\glsseeformat: made robust	174	\Glsentryuseriv: made robust	147
\glsseeitem: made robust	175	\Glsentryuserv: made robust	147
\glsseelist: made robust	174	\Glsentryuservi: made robust	147
\ifglshdescsuppressed: new	52	\glstextup: new	205
\ifglshasdesc: new	52	\ifglshassymbol: changed test to check	
\ifglshassymbol: new	52	for \@gls@default@symbol	52
altlongragged4col: updated to use		3.10a (2013-09-28)	
\glossentry and \subglossentry	277	\gls@assign@type@field: new	17
alttree: updated to use \glossentry		3.10a (2013-10-13)	
and \subglossentry	302	\@gls@keymap: new	67
index: added paragraph break at end of		\@gls@provide@newglossary: new ...	54
environment	297	\@gls@writedef: new	66
updated to use \glossentry and		\@gls@defaultplural: Obsolete	62
\subglossentry	297	\@glsnodesc: new	62
long: updated to use \glossentry and		\@print@glossary: Added providecom-	
\subglossentry	263	mand code to aux file	179
longragged: updated to use \glossentry		\gls@defglossaryentry: Changed to	
and \subglossentry	274	using \@gls@default@value	75
longragged3col: updated to use		new	75
\glossentry and \subglossentry	275	\glswritedefhook: new	74

\makeglossaries: Added providecom-	\@acrshort: added \glslabel,
mand code to aux file	\glsifplural, \glscapscase,
161	\glsinsert and \glscustomtext 344
\new@glossaryentry: new	\@gls@: add \glslabel, \glsifplural,
66	\glscapscase, \glscustomtext and
\ns@newglossary: added \@gls@provide@newglossary	\glsinsert
56	116
3.11a (2013-10-15)	change to using \glsentryfmt style
\@ACRlong: added \glslabel,	commands
\glsifplural, \glscapscase,	116
\glsinsert and \glscustomtext 346	\@gls@noexpand@fields: Fixed bug
\@ACRshort: added \glslabel,	expand replaced with noexpand
\glsifplural, \glscapscase,	63
\glsinsert and \glscustomtext 344	\@glsdisp: add \glslabel, \glsifplural,
\@Acrlong: added \glslabel,	\glscapscase, \glscustomtext and
\glsifplural, \glscapscase,	\glsinsert
\glsinsert and \glscustomtext 345	121
\@Acrshort: added \glslabel,	change to using \glsentryfmt style
\glsifplural, \glscapscase,	commands
\glsinsert and \glscustomtext 344	121
\@GLS@: add \glslabel, \glsifplural,	\@glspl@: add \glslabel, \glsifplural,
\glscapscase, \glscustomtext and	\glscapscase, \glscustomtext and
\glsinsert	\glsinsert
118	119
change to using \glsentryfmt style	change to using \glsentryfmt style
commands	commands
118	119
removed \MakeUppercase (now moved	General: added \glslabel, \glsifplural,
to \glsentryfmt)	\glscapscase, \glsinsert and
118	\glscustomtext
\@GLSpl: add \glslabel, \glsifplural,	134-141
\glscapscase, \glscustomtext and	changed to just use \Glsentrydescplural
\glsinsert
120	127
change to using \glsentryfmt style	changed to just use \glsentrydescplural
commands
120	127
removed \MakeUppercase as now dealt	changed to just use \Glsentrydesc .
with in \glsentryfmt
120	126, 127
\@Gls@: add \glsifplural, \glscapscase,	changed to just use \Glsentryfirstplural
\glscustomtext and \glsinsert 117
change to using \glsentryfmt style	125
commands	changed to just use \glsentryfirstplural
117
removed \makefirstuc (now dealt	124, 125
with in \glsentryfmt)	changed to just use \Glsentryfirst 123
117	changed to just use \glsentryfirst 123
\@Glspl@: add \glsifplural,	changed to just use \Glsentryname .
\glscapscase, \glscustomtext and
\glsinsert	125, 126
119	changed to just use \Glsentryplural 124
change to using \glsentryfmt style	changed to just use \glsentryplural 124
commands	changed to just use \Glsentrysymbolplural
120
removed \makefirstuc (now dealt	129
with in \glsentryfmt)	changed to just use \glsentrysymbolplural
120
\@acrlong: added \glslabel,	129
\glsifplural, \glscapscase,	changed to just use \Glsentrysymbol 128
\glsinsert and \glscustomtext 345	changed to just use \glsentrysymbol 128
	Changed to just use \Glsentrytext .
	123
	changed to just use \glsentrytext .
	122

changed to just use \Glsentryuseriii 131	\ns@newglossary: replaced \glsdisplay and \glsdisplayfirst with \glsentryfmt 56
changed to just use \glsentryuseriii 131, 132	compatible-3.07: cnew 26
changed to just use \Glsentryuserii 130	\SetCustomDisplayStyle: updated to use \defglentryfmt 239
changed to just use \glsentryuserii 130, 131	\SetDefaultAcronymDisplayStyle: changed to use \defglentryfmt . 224
changed to just use \Glsentryuseriv 132	\SetDescriptionAcronymDisplayStyle: updated to use \defglentryfmt . 229
changed to just use \Glsentryuseri 130	\SetDescriptionDUAAcronymDisplayStyle: updated to use \defglentryfmt . 228
changed to just use \glsentryuseri 129, 130	\SetDescriptionFootnoteAcronymDisplayStyle: updated to use \defglentryfmt . 225
changed to just use \Glsentryuservi 134	\SetDUADisplayStyle: updated to use \defglentryfmt 237
changed to just use \glsentryuservi 133, 134	\SetFootnoteAcronymDisplayStyle: updated to use \defglentryfmt . 232
changed to just use \Glsentryuserv 133	\SetSmallAcronymDisplayStyle: up- dated to use \defglentryfmt ... 234
changed to just use \glsentryuserv 133	\setupglossaries: new 27
Now requires textcase 4	\showglolong: new 244
acronymlists: replaced \@addtoacronymlists with \DeclareAcronymList 15	\showgloshort: new 244
\defglsdisplay: obsoleted 100	numbers: new 26
\defglsdisplayfirst: obsoleted 101	symbols: new 26
\defglentryfmt: new 54	3.12a (2013-10-16)
\forglsentries: replaced \ifx with \ifdefempty 48	\gls@defglossaryentry: added \glslabel 75
\gls@assign@desc: new 74	\glsaddkey: new 69
\gls@defglossaryentry: Fixed default counter if none supplied 78	3.13a (2013-11-05)
\gls@doentryfmt: new 54	\@gls@assign@symbol@field: changed to use \glssetnoexpandfield 17
\glsdisplay: obsoleted 100	\@gls@assign@symbolplural@field: changed to use \glssetnoexpandfield 17
\glsdisplayfirst: obsoleted 100	\@gls@link: removed \relax 106
\glsgenentryfmt: new 95	\@gls@notranslatorhook: new 21
\glsgetgrouptitle: Added check in case non-Latin alphabet in use 199	\@gls@setupsort@standard: moved \@gls@santizesort to \glsprestandardsort 10
\gls glossarymark: replaced \MakeUppercase with \mfirstucMakeUppercase ... 36	ucmark: added check for memoir 9
\glsnavigation: switched to using \@gls@getgrouptitle 257	see: added \gls@checkseeallowed ... 60
\ifglshasdesc: replaced \ifdefempty with \ifcsempy 52	\glossarysection: changed \glossarymark to \gls glossarymark 36
\ifglshaslong: new 52	\glossarystyle: fixed bug caused by us- ing \ifdef instead of \ifcsdef ... 201
\ifglshasshort: new 52	\gls@assign@desc@field: changed to use \glssetnoexpandfield 17
\ifglshassymbol: replaced \ifdefempty with \ifcsempy 52	
\ifglused: replaced \ifthenelse with \ifbool 50	
\longnewglossaryentry: new 74	

<code>\gls@assign@descplural@field:</code> changed to use <code>\glssetnoexpandfield</code> 17	<code>super3colheader:</code> switched to <code>\tabularnewline</code> 287
<code>\gls@assign@name@field:</code> changed to use <code>\glssetnoexpandfield</code> 17	<code>super4col:</code> switched to <code>\tabularnewline</code> 288
<code>\gls@assign@type@field:</code> changed to use <code>\glssetexpandfield</code> 17	<code>super4colheader:</code> switched to <code>\tabularnewline</code> 288
<code>\gls@checkseeallowed:</code> new 60	<code>super4colheaderborder:</code> switched to <code>\tabularnewline</code> 289
<code>\glsaddallunused:</code> set default to <code>\@glo@types</code> 151	<code>superheader:</code> switched to <code>\tabularnewline</code> 285
<code>\Glsentryfull:</code> changed to use <code>\acrfullformat</code> 148	<code>superheaderborder:</code> switched to <code>\tabularnewline</code> 286
<code>\glsentryfull:</code> changed to use <code>\acrfullformat</code> 148	3.14a (2013-11-12)
<code>\Glsentryfullpl:</code> changed to use <code>\acrfullformat</code> 149	<code>\@glswritefiles:</code> renamed <code>\glswritefiles</code> to <code>\@glswritefiles</code> and used “savewrites” option to set <code>\glswritefiles</code> 167
<code>\glsentryfullpl:</code> changed to use <code>\acrfullformat</code> 149	General: new 248
<code>\glsglossarymark:</code> renamed <code>\glossarymark</code> to <code>\glsglossarymark</code> to avoid con- flict with memoir 36	acronyms: new 13
<code>\glsprestandardsort:</code> new 9	<code>\gls@defglossaryentry:</code> added check for existence of default glossary 76
<code>\glssetexpandfield:</code> new 16	set the default for firstplural to be the value of plural 78
<code>\glssetnoexpandfield:</code> new 16	xindygloss: new 25
<code>altsuper4colheader:</code> switched to <code>\tabularnewline</code> 289	<code>\longprovideglossaryentry:</code> new ... 75
<code>altsuper4colheaderborder:</code> switched to <code>\tabularnewline</code> 290	compatible-2.07: added check for 2.07 before setting 3.07 compatibility 26
<code>long:</code> switched to <code>\tabularnewline</code> 263, 264	<code>notranslate:</code> new 21
<code>long3col:</code> switched to <code>\tabularnewline</code> 265	<code>\provideglossaryentry:</code> new 65
<code>long3colheader:</code> switched to <code>\tabularnewline</code> 266	4.0 (2013-11-14)
<code>long3colheaderborder:</code> switched to <code>\tabularnewline</code> 266	<code>\gls@defglossaryentry:</code> added check for first key 78
<code>long4col:</code> switched to <code>\tabularnewline</code> 266	<code>super:</code> fixed typo in <code>\subglossentry</code> (<code>\glossentrydesc</code>) 285
<code>long4colheader:</code> switched to <code>\tabularnewline</code> 267	4.01 (2013-11-16)
<code>longheader:</code> switched to <code>\tabularnewline</code> 264	General: fixed non-value options so that they can be passed to document class . 6
<code>longheaderborder:</code> switched to <code>\tabularnewline</code> 264	<code>\CustomAcronymFields:</code> inserted miss- ing comma 239
<code>\SetFootnoteAcronymDisplayStyle:</code> fixed missing argument bug 232	4.02 (2013-12-05)
<code>super:</code> switched to <code>\tabularnewline</code> . 285	<code>\@acrfull:</code> now using <code>\acrfullfmt</code> .. 206
<code>super3col:</code> switched to <code>\tabularnewline</code> 286	<code>\@gls@indexdef:</code> new 27
	<code>\@gls@numbersdef:</code> new 26
	<code>\@gls@symbolsdef:</code> new 26
	General: Removed <code>\acronymfont</code> . 138–141
	<code>\ACRfullfmt:</code> new 207
	<code>\Acrfullfmt:</code> new 207
	<code>\acrfullfmt:</code> new 206
	<code>\ACRfullplfmt:</code> new 208

\Acrfullplfmt: new	208	\SetDescriptionFootnoteAcronymDisplayStyle:	
\acrfullplfmt: new	208	Moved check for empty custom text to	
\acronymentry: new	210	prevent unwanted parenthetical ma-	
sanitize: fixed bug that caused an error		terial	225
here	20	\SetFootnoteAcronymDisplayStyle:	
sc-short-long: new	214	Moved check for empty custom text to	
sc-short-long-desc: new	216	prevent unwanted parenthetical ma-	
\Genacrfullformat: new	99	terial	232
\genacrfullformat: new	99	\SetGenericNewAcronym: new	209
\GenericAcronymFields: new	210	\SetSmallAcronymDisplayStyle:	
\Genplacrfullformat: new	100	Moved check for empty custom text to	
\genplacrfullformat: new	100	prevent unwanted parenthetical ma-	
\Glsentryfull: bug fix: added missing		terial	234
\acronymfont	148	dua: new	217
\glentryfull: bug fix: added missing		dua-desc: new	219
\acronymfont	148	numberedsection: added nameref op-	
\Glsentryfullpl: bug fix: added miss-		tion	6
ing \acronymfont	149	4.02 (2013-13-05)	
\glentryfullpl: bug fix: added miss-		\makeglossaries: made preamble only	162
ing \acronymfont	149	4.03 (2014-01-17)	
\glsgenacfmt: new	97	General: changed default to \@empty in-	
\GlsUseAcrEntryDispStyle: new ...	212	stead of \relax	26
\GlsUseAcrStyleDefs: new	212	4.03 (2014-01-20)	
short-long: new	213	\@do@wrglossary: added \glsdetoklabel	
short-long-desc: new	216	172
xindynoglsnumbers: new	25	\@ACRlong: removed \glslabel (defined	
sm-short-long: new	215	in \@gls@link)	346
sm-short-long-desc: new	217	\@ACRshort: removed \glslabel (de-	
index: new	27	fined in \@gls@link)	344
\newacronymstyle: new	211	\@Acrlong: removed \glslabel (defined	
long-sc-short: new	214	in \@gls@link)	345
long-sc-short-desc: new	215	\@Acrshort: removed \glslabel (de-	
long-short: new	212	fined in \@gls@link)	344
long-short-desc: new	215	\@GLS@: removed \glslabel (defined in	
long-sm-short: new	214	\@gls@link)	118
long-sm-short-desc: new	216	\@GLSpl: removed \glslabel (defined in	
long-sp-short-desc: new	215	\@gls@link)	120
footnote: new	219	\@Gls@: removed \glslabel (defined in	
footnote-desc: new	221	\@gls@link)	117
footnote-sc: new	220	\@Gls@entry@field: new	142
footnote-sc-desc: new	221	\@Glspl@: removed \glslabel (defined	
footnote-sm: new	221	in \@gls@link)	119
footnote-sm-desc: new	222	\@acrlong: removed \glslabel (defined	
\setacronymstyle: new	211	in \@gls@link)	345
\SetDescriptionAcronymDisplayStyle:		\@acrshort: removed \glslabel (de-	
Moved check for empty custom text to		fined in \@gls@link)	344
prevent unwanted parenthetical ma-		\@gls@: removed \glslabel (defined in	
terial	229	\@gls@link)	116
		\@gls@access@display: new	332

<code>\@gls@entry@field</code> : new	142	<code>\glsentrydescpluralaccess</code> : switched to using <code>\@gls@entry@field</code>	331
<code>\@gls@fetchfield</code> : new	67	<code>\glsentryfirstaccess</code> : switched to using <code>\@gls@entry@field</code>	331
<code>\@gls@field@link</code> : new	122	<code>\glsentryfirstplural</code> : added <code>\glsdetoklabel</code>	145
<code>\@gls@link</code> : added <code>\glsdetoklabel</code> .	105	<code>\glsentrylongaccess</code> : switched to using <code>\@gls@entry@field</code>	332
moved <code>\@gls@link@opts</code> and <code>\@gls@link@label</code> to <code>\@gls@link</code>	105	<code>\glsentrylongpluralaccess</code> : switched to using <code>\@gls@entry@field</code>	332
<code>\@gls@writedef</code> : added <code>\glsdetoklabel</code>	66	<code>\glsentrypluralaccess</code> : switched to using <code>\@gls@entry@field</code>	331
<code>\@glsdisp</code> : removed <code>\glslabel</code> (defined in <code>\@gls@link</code>)	121	<code>\glsentryshortaccess</code> : switched to using <code>\@gls@entry@field</code>	331
<code>\@glspl@</code> : removed <code>\glslabel</code> (defined in <code>\@gls@link</code>)	119	<code>\glsentryshortpluralaccess</code> : switched to using <code>\@gls@entry@field</code>	331
<code>\@printglossary</code> : added <code>\glsdetoklabel</code>	178	<code>\glsentrysymbolaccess</code> : switched to using <code>\@gls@entry@field</code>	331
General: removed <code>\glslabel</code> (defined in <code>\@gls@link</code>)	134	<code>\glsentrysymbolpluralaccess</code> : switched to using <code>\@gls@entry@field</code>	331
sc-short-long-desc: redefined to use accessibility information	350	<code>\glsentrytextaccess</code> : switched to using <code>\@gls@entry@field</code>	330
<code>\compatibleglossentry</code> : added <code>\glsdetoklabel</code>	326	<code>\glsgenacfmt</code> : redefined to use accessibility information	341
<code>\compatiblesubglossentry</code> : added <code>\glsdetoklabel</code>	327	<code>\glsgenentryfmt</code> : redefined to use accessibility information	338
<code>\Genacrfullformat</code> : redefined to use accessibility information	343	<code>\glshyperlink</code> : added <code>\glsdetoklabel</code>	150
<code>\genacrfullformat</code> : redefined to use accessibility information	343	<code>\glslocalreset</code> : added <code>\glsdetoklabel</code>	83
<code>\Genplacrfullformat</code> : redefined to use accessibility information	343	<code>\glslocalunset</code> : added <code>\glsdetoklabel</code>	84
<code>\genplacrfullformat</code> : redefined to use accessibility information	343	<code>\glsmoveentry</code> : added <code>\glsdetoklabel</code>	80
<code>\glossentryname</code> : added <code>\glsdetoklabel</code>	196	replaced <code>\ifthenelse</code> with <code>\ifdefequal</code>	81
<code>\gls@defglossaryentry</code> : added <code>\glsdetoklabel</code>	75	<code>\glsrefentry</code> : added <code>\glsdetoklabel</code>	194
replaced <code>#1</code> with <code>\@gls@label</code>	76	<code>\glsreset</code> : added <code>\glsdetoklabel</code> ...	83
replaced <code>\ifthenelse</code> with <code>\ifdefequal</code>	77	<code>\glsseelist</code> : added <code>\expandafter</code> commands	175
<code>\glsadd</code> : added <code>\glsdetoklabel</code>	151	<code>\glsstepentry</code> : added <code>\glsdetoklabel</code>	193
<code>\glsaddkey</code> : switched to using <code>\@gls@field@link</code>	70	<code>\glsstepsubentry</code> : added <code>\glsdetoklabel</code>	194
<code>\glsdetoklabel</code> : new	49	<code>\glsunset</code> : added <code>\glsdetoklabel</code> ...	83
<code>\glsdisplaynumberlist</code> : added <code>\glsdetoklabel</code>	149	short-long: commented spurious EOL	214
<code>\glsdoifexistsorwarn</code> : new	50	redefined to use accessibility information	348
<code>\glsentryaccess</code> : switched to using <code>\@gls@entry@field</code>	330		
<code>\glsentrydescaccess</code> : switched to using <code>\@gls@entry@field</code>	331		

\showglotextaccess:added\glsdetoklabel	\@gls@noidx@do:new	186
..... 362	\@gls@noref@warn:new	166
\showglotype:added\glsdetoklabel	\@gls@reference:new	188
..... 242	\@gls@warnonglossdefined:new	16
\showglouserii:added\glsdetoklabel	\@gls@warnontheGLOSSdefined:new	16
..... 242	\@no@makeGLOSSaries:new	166
\showglouseriii:added\glsdetoklabel	\@print@GLOSSary:new	178
..... 243	\@print@noidx@GLOSSary:new	185
\showglouseriv:added\glsdetoklabel	\@printgloss@setsort:new	176
..... 243	\@printglossary:new	177
\showglouserv:added\glsdetoklabel	General: added sort key to printgloss	
..... 243	group	192
\showglouservi:added\glsdetoklabel	\compatibleglossentry: changed	
..... 243	\newcommand to \def as is may or may	
dua: fixed bug in \acrfullfmt	not be defined	326
..... 218	\compatiblesubglossentry: changed	
fixed bug in \Acrrfullplfmt	\newcommand to \def as is may or may	
..... 218	not be defined	327
fixed bug in \acrfullplfmt	\defglsdisplayfirst: fixed unwanted	
..... 218	space	101
redefined to use accessibility informa-	\glo@grabfirst:new	186
tion	\gls@defglossaryentry: replaced \ifx	
..... 351	with \ifdefvoid	80
dua-desc: commented spurious EOL	\glsnoidxdisplayloc:new	188
.. 219	\glsnoidxdisplayloc:clisthandler:	
redefined to use accessibility informa-	new	188
tion	\glsnoidxloc:clist:new	187
..... 353	\glsnoidxloc:clisthandler:new	188
4.04 (2014-03-04)	\glsnoidxstripaccents:new	18
\@gls@getcounterprefix: added warn-	alttree: moved hangindent and	
ing if no prefix can be formed	parindent assignments outside level	
..... 173	test	302
4.04 (2014-03-06)	\makeGLOSSaries: Moved definition of	
\@gls@noidx@nosanitizesort:new	\glswrite to \makeGLOSSaries	161
.. 18	\makenoidxGLOSSaries:new	163
\@gls@noidx@sanitizesort:new	\printglossary: changed to use new	
..... 18	\@printglossary	176
\@gls@sanitizesort:new	\printnoidxGLOSSaries:new	176
..... 17	\printnoidxGLOSSary:new	176
\@glo@addchildren:new	\showgloloclist:new	245
..... 180	\warn@noprintglossary: Activate warn-	
\@glo@do@sortentries:new	ing in \makeGLOSSaries	175
..... 181	\writeist: checked for definition of	
\@glo@grabfirst:new	\glswrite	153, 157
..... 186	4.06 (2014-03-12)	
\@glo@sortedinsert:new	\@GLS@: added \glsifhyper	118
..... 182	\@GLSpl: added \glsifhyper	120
\@glo@sortentries:new	\@Gls@: added \glsifhyper	117
..... 180	\@GLspl@: added \glsifhyper	120
\@glo@sorthandler@case:new		
..... 182		
\@glo@sorthandler@letter:new		
... 182		
\@glo@sorthandler@nocase:new		
... 182		
\@glo@sorthandler@word:new		
..... 182		
\@glo@sortmacro@case:new		
..... 184		
\@glo@sortmacro@def:new		
..... 184		
\@glo@sortmacro@def@do:new		
..... 185		
\@glo@sortmacro@letter:new		
..... 183		
\@glo@sortmacro@nocase:new		
..... 184		
\@glo@sortmacro@standard:new		
... 183		
\@glo@sortmacro@use:new		
..... 185		
\@glo@sortmacro@word:new		
..... 183		
\@gls@getothergrouptitle:new		
... 199		

\@gls@: added \glsifhyper	116	\@gls@field@link: added assignment of	
\@gls@numbersdef: added hook to set		\do@gls@link@checkfirsthyper	122
toc title	27	\@gls@forbidtexext: new	54
\@gls@symbolsdef: added hook to set		\@gls@hyp@opt: new	102
toc title	26	\@gls@link: removed redundancy	105
\@glsdisp: added \glsifhyper	121	renamed \gls@type to \glstype ...	105
\@glspl@: added \glsifhyper	119	\@glsdisp: moved \glsifhyper	121
General: added \glsifhyper	134–141	moved check for first use to	
acronym: added hook to set toc title	13	\@gls@link	121
acronyms: added hook to set toc title ...	13	\@glspl@: moved \glsifhyper	119
\glsdefmain: added hook to set toc title	12	moved check for first use to	
4.07 (2014-04-04)		\@gls@link	119
\@glossarysection: added optional ar-		\@ignored@glossaries: new	58
gument when using unstarred version	37	General: added entrycounter option to	
\@gls@noidx@do: added \global in case		printgloss family	190
it's used in a tabular-like style	186	added nopostdot option to printgloss	
\Acrfullplfmt: fixed no case change		family	190
bug	208	added subentrycounter option to	
\glsletentryfield: new	142	printgloss family	191
4.08 (2014-07-30)		explicitly initialise hyper key	102
\@ACRlong: added \do@gls@link@checkfirsthyper		moved \glsifhyper	134–141
.....	345	removed \@sACRlongpl	141
\@ACRshort: added \do@gls@link@checkfirsthyper		removed \@sAcrlongpl	140
.....	344	removed \@sacrlongpl	140
\@Acrlong: added \do@gls@link@checkfirsthyper		removed \@sACRlong	139
.....	345	removed \@sAcrlong	138
\@Acrshort: added \do@gls@link@checkfirsthyper		removed \@sacrlong	138
.....	344	removed \@sACRshortpl	137
\@GLS@: moved \glsifhyper	118	removed \@sAcrshortpl	137
moved check for first use to		removed \@sacrshortpl	136
\@gls@link	118	removed \@sACRshort	135
\@GLSpl@: moved \glsifhyper	120	removed \@sAcrshort	135
moved check for first use to		removed \@sacrshort	134
\@gls@link	120	removed \@sgls@link	103
\@Gls@: moved \glsifhyper	117	removed \@sGLSdescplural	127
moved check for first use to		removed \@sGlsdescplural	127
\@gls@link	117	removed \@sglsdescplural	127
\@Glspl@: moved \glsifhyper	120	removed \@sGLSdesc	127
moved check for first use to		removed \@sGlsdesc	126
\@gls@link	120	removed \@sglsdesc	126
\@acrlong: added \do@gls@link@checkfirsthyper		removed \@sglsdisp	121
.....	345	removed \@sGLSfirstplural	125
\@acrshort: added \do@gls@link@checkfirsthyper		removed \@sGlsfirstplural	125
.....	343	removed \@sglsfirstplural	124
\@closegls: new	159	removed \@sGLSfirst	123
\@gls@: moved \glsifhyper	116	removed \@sGlsfirst	123
moved check for first use to		removed \@sglsfirst	123
\@gls@link	116	removed \@sGLSname	126
\@gls@doautomake: new	25	removed \@sGlsname	125

removed \@sglsname	125	\acrfull: removed \@sacrfull	206
removed \@sGLSplural	124	switched to using \@gls@hyp@opt ..	206
removed \@sglsplural	124	\ACRfullpl: removed \@sACRfullpl ..	208
removed \@sglsplural	124	switched to using \@gls@hyp@opt ..	208
removed \@sGLSpl	120	\Acrfullpl: removed \@sAcrfullpl ..	208
removed \@sGlspl	119	switched to using \@gls@hyp@opt ..	208
removed \@sglspl	118	\acrfullpl: removed \@sacrfullpl ..	207
removed \@sGLSsymbolplural	129	switched to using \@gls@hyp@opt ..	207
removed \@sGlsymbolplural	129	\ACRlong: switched to using \@gls@hyp@opt	
removed \@sglsymbolplural	128	139
removed \@sGLSsymbol	128	\Acrlong: switched to using \@gls@hyp@opt	
removed \@sGlsymbol	128	138
removed \@sglsymbol	128	\acrlong: switched to using \@gls@hyp@opt	
removed \@sGLStext	122	138
removed \@sGlstext	123	\ACRlongpl: switched to using	
removed \@sglstext	122	\@gls@hyp@opt	141
removed \@sGLSuseriii	131	\Acrlongpl: switched to using	
removed \@sGlsuseriii	131	\@gls@hyp@opt	140
removed \@sglsuseriii	131	\acrlongpl: switched to using	
removed \@sGLSuserii	131	\@gls@hyp@opt	140
removed \@sGlsuserii	130	\ACRshort: switched to using	
removed \@sglsuserii	130	\@gls@hyp@opt	135
removed \@sGLSuseriv	132	\Acrshort: switched to using	
removed \@sGlsuseriv	132	\@gls@hyp@opt	135
removed \@sglsuseriv	132	\acrshort: switched to using	
removed \@sGLSuseri	130	\@gls@hyp@opt	134
removed \@sGlsuseri	130	\ACRshortpl: switched to using	
removed \@sglsuseri	129	\@gls@hyp@opt	137
removed \@sGLSuservi	134	\Acrshortpl: switched to using	
removed \@sGlsuservi	134	\@gls@hyp@opt	136
removed \@sglsuservi	133	\acrshortpl: switched to using	
removed \@sGLSuserv	133	\@gls@hyp@opt	136
removed \@sGlsuserv	133	\forallacronyms: new	48
removed \@sglsuserv	133	\GLS: switched to using \@gls@hyp@opt	118
removed \@sGLS	118	\Gls: switched to using \@gls@hyp@opt	117
removed \@sGls	117	\gls: switched to using \@gls@hyp@opt	116
removed \@sgls	116	\gls@defglossaryentry: added check	
removed \@thirdofthree (defined in		for ignored glossary	76
kernel)	116	\gls@istfilebase: new	33
removed sPGLS	253	\glsaddkey: removed \@sGLS@user@<key>	
removed sPgls	251	71
removed spgls	250	removed \@sGls@user@<key>	70
removed sPGLSpl	253	removed \@sgls@user@<key>	70
removed sPglspl	252	switched to using \@gls@hyp@opt	70, 71
removed spglspl	251	\GLSdesc: switched to using \@gls@hyp@opt	
\ACRfull: removed \@sACRfull	207	126
switched to using \@gls@hyp@opt ..	207	\Glsdesc: switched to using \@gls@hyp@opt	
\Acrfull: removed \@sAcrfull	206	126
switched to using \@gls@hyp@opt ..	206		

\glsdesc: switched to using \@gls@hyp@opt 126	\glsplural: switched to using \@gls@hyp@opt 124
\GLSdescplural: switched to using \@gls@hyp@opt 127	\glsspace: new 206
\Glsdescplural: switched to using \@gls@hyp@opt 127	\GLSsymbol: switched to using \@gls@hyp@opt 128
\glsdescplural: switched to using \@gls@hyp@opt 127	\Glsymbol: switched to using \@gls@hyp@opt 128
\glsdisablehyper: added \KV@glslink@hyperfalse to definition 115	\glssymbol: switched to using \@gls@hyp@opt 128
\glsdisp: switched to using \@gls@hyp@opt 121	\GLSsymbolplural: switched to using \@gls@hyp@opt 129
\glsdohyperlink: new 115	\Glsymbolplural: switched to using \@gls@hyp@opt 129
\glsdohypertarget: new 115	\glssymbolplural: switched to using \@gls@hyp@opt 128
\glsenablehyper: added \KV@glslink@hypertrue to definition 115	\GLStext: switched to using \@gls@hyp@opt 122
\GLSfirst: switched to using \@gls@hyp@opt 123	\Glstext: switched to using \@gls@hyp@opt 122
\Glsfirst: switched to using \@gls@hyp@opt 123	\glstext: switched to using \@gls@hyp@opt 122
\glsfirst: switched to using \@gls@hyp@opt 123	\glstreenamfmt: new 296
\GLSfirstplural: switched to using \@gls@hyp@opt 125	\GLSuseri: switched to using \@gls@hyp@opt 130
\Glsfirstplural: switched to using \@gls@hyp@opt 125	\Glsuseri: switched to using \@gls@hyp@opt 130
\glsfirstplural: switched to using \@gls@hyp@opt 124	\glsuseri: switched to using \@gls@hyp@opt 129
\glsifhyper: deprecated 102	\GLSuserii: switched to using \@gls@hyp@opt 131
\glslink: switched to using \@gls@hyp@opt 103	\Glsuserii: switched to using \@gls@hyp@opt 130
\glslinkcheckfirsthyperhook: new 104	\glsuserii: switched to using \@gls@hyp@opt 130
\glslinkvar: new 102	\GLSuseriii: switched to using \@gls@hyp@opt 131
\GLSname: switched to using \@gls@hyp@opt 126	\Glsuseriii: switched to using \@gls@hyp@opt 131
\Glsname: switched to using \@gls@hyp@opt 125	\glsuseriii: switched to using \@gls@hyp@opt 131
\glsname: switched to using \@gls@hyp@opt 125	\GLSuseriv: switched to using \@gls@hyp@opt 132
\GLSpl: switched to using \@gls@hyp@opt 120	\Glsuseriv: switched to using \@gls@hyp@opt 132
\Glspl: switched to using \@gls@hyp@opt 119	\glsuseriv: switched to using \@gls@hyp@opt 132
\glspl: switched to using \@gls@hyp@opt 118	\GLSserv: switched to using \@gls@hyp@opt 133
\GLSplural: switched to using \@gls@hyp@opt 124	
\Glsplural: switched to using \@gls@hyp@opt 124	

\Glsuserv: switched to using	4.12 (2014-11-22)
\@gls@hyp@opt	133
\glsuserv: switched to using	
\@gls@hyp@opt	132
\GLSuservi: switched to using	
\@gls@hyp@opt	134
\Glsuservi: switched to using	
\@gls@hyp@opt	134
\glsuservi: switched to using	
\@gls@hyp@opt	133
\ifignoredglossary: new	58
altlongragged4col: fixed bug that displayed description instead of symbol	277
\newglossary: added starred version ..	55
\newignoredglossary: new	57
\ns@newglossary: added \@glotype@<name>@log	56
new	55
\p@gls@hyp@opt: new	103
\PGLS: changed to use \@gls@hyp@opt	253
\Pgls: changed to use \@gls@hyp@opt	251
\pgls: changed to use \@gls@hyp@opt	250
\PGLSpl: changed to use \@gls@hyp@opt	253
\Pglspl: changed to use \@gls@hyp@opt	252
\pglspl: changed to use \@gls@hyp@opt	251
\s@gls@hyp@opt: new	103
\s@newglossary: new	55
automake: new	25
4.09 (2014-08-12)	
\glsaddkey: fixed bug in user commands	70
4.10 (2014-08-27)	
\@Gls@acentryname: new	143
\@Gls@entryname: new	143
\@gls@glossary: Renamed \@glossary to \@gls@glossary	168
\glspercentchar: new	152
\glstildechar: new	152
alttree: moved space after symbol	302, 304
4.11 (2014-09-01)	
\@do@wrglossary: added hook	171
sanitize: none option	20
\gls@wrglossary: renamed from \@wrglossary to \gls@wrglossary	168
\glsaddprotectedpagefmt: new	170
\glsbackslash: new	152
\@gls@addpredefinedattributes: Added glsignore attribute	42
\@gls@adjustmode: new	151
\@gls@notranslatorhook: removed ...	21
\@gls@toc: added \protect to \numberline	39
\@gls@usetranslator: new	21
\glsacrpluralsuffix: new	30
\glsadd: added check for vertical mode	151
\glsaddallunused: replaced @gobble with glsignore	151
\glsifusedtranslatorordict: new	21
\glsignore: new	152
\glsupacrpluralsuffix: new	30
\ProvidesGlossariesLang: new	30
\RequireGlossariesLang: new	30
4.13 (2015-02-03)	
\indexspace: new	259, 279, 296
4.14 (2015-02-28)	
\@@glslocalreset: new	84
\@@glslocalunset: new	84
\@@glsreset: new	85
\@@glsunset: new	84
\@newglossaryentry@defcounters: new	86
\cGls: new	89
\cGls@: new	89
\cGlspl@: new	90
\cglsl: new	89
\cglsl@: new	89
\cglspl: new	90
\cglspl@: new	90
\@gls@entry@count: new	88
\@gls@increment@currcount: new ...	88
\@gls@local@increment@currcount: new	88
\@gls@write@entrycounts: new	88
\@glslocalreset: new	84
\@glslocalunset: new	84
\@glsreset: new	84
\@glsunset: new	84
\@newglossaryentry@defcounters: new	80
\cGls: new	89
\cglsl: new	88
\cGlsformat: new	89
\cglslformat: new	89
\cGlspl: new	90

\cglspl:new	89	\glswriteentry:new	169
\cGlsplformat:new	90	\ifglsfieldcseq:new	74
\cglsplformat:new	90	\ifglsfielddefeq:new	73
\gls@defdocnewglossaryentry:new ..	65	\ifglsfieldeq:new	73
\glsenableentrycount:new	86	long-sp-short:new	212
\glslocalreset:switched to \@glslocalreset	83	\showglofield:new	245
4.18 (2015-09-09)			
\glslocalunset:switched to \@glslocalunset	84	General: split mfirstuc into separate bundle	4
\glsreset:switched to \@glsreset ...	83	4.19 (2015-10-31)	
\glsunset:switched to \@glsunset ...	83	\glstreenamibox:new	302
4.15 (2015-03-16)		4.19 (2015-11-22)	
General: bug fix replaced \@glo@type with \glstype	141	\@gls@link@nocheckfirsthyper:new	122
4.16 (2015-06-18)		\@gls@preglossaryhook:new	176
\glsaddstoragekey:new	68	\@printglossary:added \@gls@preglossaryhook	178
4.16 (2015-07-08)		\do@gl:disablehyperinlist:new ..	104
\@ACRlong:added \glspostlinkhook	346	\doifglossarynoexistsordo:new ...	51
\@ACRshort:added \glspostlinkhook	345	\gls@gobbleopt:new	55
\@Acrlong:added \glspostlinkhook	345	\glsdoifexistsordo:new	50
\@Acrshort:added \glspostlinkhook	344	4.20 (2015-11-30)	
\@GLS@:added \glspostlinkhook ...	118	\@gls@link:added \@gls@setdefault@glslink@opts	105
\@GLSpl:added \glspostlinkhook ..	121	added \glsdonohyperlink when hyperlink is suppressed	106
\@Gls@:added \glspostlinkhook ...	118	\@gls@setdefault@glslink@opts:new	105
\@Glspl@:added \glspostlinkhook ..	120	\gls@checkseeallowed@preambleonly:new	61
\@acrlong:added \glspostlinkhook	345	\glsdonohyperlink:new	115
\@acrshort:added \glspostlinkhook	344	4.21 (2016-01-24)	
\@gls@:added \glspostlinkhook ...	117	\@printglossary:warn if no style has been set	177
\@gls@@link:added \glspostlinkhook	104	General: changed checkfirsthyper assignment	134–141
\@gls@field@link:added \glspostlinkhook	122	\glossarystyle:set default style if not already set	201
\@gls@link: moved definition of \glsifhyperon outside of this macro	105	\glsLTpenaltycheck:new	272
\@glsdisp:added \glspostlinkhook	121	\glspatchLToutput:new	272
\@glspl@:added \glspostlinkhook ..	119	\glspenaltygroupskip:new	272
General:added \glspostlinkhook	135–141	altlong4col-booktabs:new	270
\glsacspace:new	213	altlongragged4col-booktabs:new ..	272
\glsadd:changed \@do@wrglossary to \@do@wrglossary	151	long-booktabs:new	269
\glsfielddef:new	72	long3col-booktabs:new	269
\glsfieldedef:new	71	long4col-booktabs:new	270
\glsfieldfetch:new	72	longragged-booktabs:new	271
\glsfieldgdef:new	72	longragged3col-booktabs:new	271
\glsfieldxdef:new	71	\setglossarystyle:set default style if not already set	200
\glsifhyperon: moved definition of \glsifhyperon	104		
\glslinkpostsetkeys:new	104		
\glspostlinkhook:new	104		

4.22 (2016-04-19)

\@@do@wrglossary: added check for	
\@arabic	171
added test to allow temporary primitive	
modifications and added arabic case	171
mcolalttreespannav: new	283
mcolindexspannav: new	280
mcoltreenonamespannav: new	282
mcoltreespannav: new	281
\gls@arabicpage: new	170
\gls@protected@pagefmts: added ara-	
bic to list	169
\glsentrytitlecase: new	146
\glsfindwidesttoplevelname: new .	301
\glslistgroupheaderfmt: new	260
\glslistnavigationitem: new	260
\glstreegroupheaderfmt: new	296
\glstreenavigationfmt: new	296
\ifglswrallowprimitivemods: new .	170
list: fixed missing space before descrip-	
tion	260
long: fixed typo in \glossentrydesc .	264
super4col: fixed bug in \glossentry .	288

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	
\!	111, 112
\"	18, 108, 110, 111, 113, 114
\#	155
\%	152, 157, 158, 310, 311
\&	30, 150
\'	18
\.	8, 18
\=	18
\?	109–111
\@@delimN	203
\@do@wrglossary	163, 172
\@do@wrglossary	151, 169
\@glo@assign@sortkey	164
\@glo@list	48
\@glo@sort	18
\@glo@type	176
\@glossarysec	5, 37, 38
\@glossaryseclabel	6, 38, 190
\@glossarysecstar	6, 37, 38, 190
\@gls@checkactual	113
\@gls@checkbar	112
\@gls@checkescactual	110, 111
\@gls@checkescbar	111
\@gls@checkesclevel	111, 112
\@gls@checkescquote	110
\@gls@checklevel	112, 113
\@gls@checkquote	109, 110
\@gls@default@entryfmt	91, 100, 101
\@gls@expand@field	16, 64, 68, 69, 224, 226–228, 231, 233, 236–238, 357–361
\@gls@fixbraces	174
\@gls@noexpand@field	17, 63, 64
\@gls@noidx@no@sanitizesort	18
\@gls@noidx@nosanitizesort	165
\@gls@nosanitizesort	17, 165
\@gls@sanitizesort	17, 165
\@@gls@xdycheckbackslash	114
\@@gls@xdycheckquote	113, 114
\@@gls@localreset	84, 87
\@@gls@localunset	84, 87
\@@gls@reset	84, 87
\@@gls@unset	84, 86
\@@newglossaryentry@defcounters	86
\@@this@glo@	49
\@ACRfull	207
\@ACRfullpl	208
\@ACRlong	139, 207
\@ACRlongpl	141, 208
\@ACRshort	135, 207
\@ACRshortpl	137, 208
\@Acrfull	206, 207
\@Acrfullpl	208
\@Acrlong	138, 207
\@Acrlongpl	140, 208
\@Acrshort	135
\@Acrshortpl	137
\@Alph	169, 171
\@GLS	118
\@GLS@	118, 253
\@GLSdesc	126, 127
\@GLSdesc@	127
\@GLSdescplural	127
\@GLSdescplural@	127
\@GLSfirst	123
\@GLSfirst@	123
\@GLSfirstplural	125
\@GLSfirstplural@	125
\@GLSname	126
\@GLSname@	126
\@GLSpl	120
\@GLSpl@	120, 254
\@GLSplural	124
\@GLSplural@	124

\@GLSsymbol	128	\@Glsuseriv	132
\@GLSsymbol@	128	\@Glsuseriv@	132
\@GLSsymbolplural	129	\@Glsuserv	133
\@GLSsymbolplural@	129	\@Glsuserv@	133
\@GLStext	122	\@Glsuservi	134
\@GLStext@	122	\@Glsuservi@	134
\@GLSuseri	130	\@Mi	272
\@GLSuseri@	130	\@PGLS	253
\@GLSuserii	131	\@PGLS@	253
\@GLSuserii@	131	\@PGLSpl	253
\@GLSuseriii	131	\@PGLSpl@	253
\@GLSuseriii@	131, 132	\@Pgls	251
\@GLSuseriv	132	\@Pgls@	251
\@GLSuseriv@	132	\@Pglspl	252
\@GLSuserv	133	\@Pglspl@	252
\@GLSuserv@	133	\@Roman	170, 171
\@GLSuservi	134	\@acrfull	206
\@GLSuservi@	134	\@acrfullpl	207
\@Gls	117	\@acrlong	138, 206
\@Gls@	87, 89, 117, 252	\@acrlongpl	140, 208
\@Gls@acentryname	209	\@acrshort	134, 206, 207
\@Gls@entry@field	69, 143–148	\@acrshortpl	136, 208
\@Gls@entryname	143, 209	\@addtoacronymlists	14
\@Glsdesc	126	\@after	14
\@Glsdesc@	126	\@afterheading	261, 314
\@Glsdescplural	127	\@alph	169, 171
\@Glsdescplural@	127	\@arabic	170, 171
\@Glsfirst	123	\@auxout	54, 56, 88, 161, 163, 166, 175, 179, 255
\@Glsfirst@	123	\@backslashchar	108, 114
\@Glsfirstplural	125	\@before	14
\@Glsfirstplural@	125	\@bsphack	168
\@Glsname	125	\@cGls	89
\@Glsname@	125, 126	\@cGls@	87, 89
\@Glspl	119	\@cGlspl	90
\@Glspl@	87, 90, 119, 252, 253	\@cGlspl@	87, 90
\@Glsplural	124	\@cclv	272, 273
\@Glsplural@	124	\@cgls	88
\@Glsymbol	128	\@cgls@	87, 89
\@Glsymbol@	128	\@cglspl	89
\@Glsymbolplural	129	\@cglspl@	87, 90
\@Glsymbolplural@	129	\@chapter	28
\@Glstext	122, 123	\@classoptionslist	27
\@Glstext@	123	\@colht	273
\@Glsuseri	130	\@colroom	273
\@Glsuseri@	130	\@currentlabelname	6, 190
\@Glsuserii	130	\@curroptions	27
\@Glsuserii@	130	\@declaredoptions	27
\@Glsuseriii	131	\@delimN	202
\@Glsuseriii@	131	\@delimR	202

\@disable@onlypremakeg	162	\@glo@descplural	59, 74, 75
\@disable@premakecs	29	\@glo@descpluralaccess	328–330
\@disabled@gl\$addxdycounters	41	\@glo@do@sortentries	180
\@do@addcounter	40	\@glo@entry	151
\@do@auxoutstuff	179	\@glo@entryprefix	248
\@do@glossentry	195, 196, 326	\@glo@entryprefixfirst	248
\@do@gl\$@getcounterprefix	172	\@glo@entryprefixfirstplural ..	248, 249
\@do@gl\$@islistofacronyms	14	\@glo@entryprefixplural	248
\@do@gl\$see	80	\@glo@esclabel	81, 82
\@do@ifinlist	40	\@glo@etext	92, 94
\@do@newglossaryentry	210,	\@glo@first	59, 75, 78, 79, 235, 361
224, 226–228, 230–233, 235–240,	357–361	\@glo@firstaccess	327, 329, 330
\@do@seeglossary	163, 174	\@glo@firstplural	59, 75, 78, 361
\@do@subglossentry	197, 327	\@glo@firstpluralaccess	328–330
\@do@wrglossary	105	\@glo@grabfirst	186
\@do@writeaux@info	175	\@glo@label	62, 68, 69,
\@ehc	272	71–80, 86, 149, 150, 248, 249,	301, 329, 330
\@empty	11, 14, 25–27, 29, 40, 41,	\@glo@list	80
44, 47, 48, 76, 77, 82, 106, 107,	116–120,	\@glo@long	52, 62, 76, 79
134–141, 153, 156, 158, 160, 167,	168,	\@glo@longaccess	328–330
173, 190, 193, 200, 225, 227, 229,	231–	\@glo@longpl	62,
234, 236, 238, 240, 308, 310, 312,	344–346	76, 79, 224, 226, 228, 230, 233,	235, 237, 357
\@end@fixbraces	174	\@glo@longpluralaccess	328–330
\@end@fortrue	22, 51, 68, 255	\@glo@name	10, 58, 63, 75, 77–79
\@esphack	169	\@glo@no@assign@sortkey	162
\@expandtwoargs	27	\@glo@numfmt	172, 308
\@first@ofone	18, 19	\@glo@parent	11, 61, 75, 77, 78, 82, 181
\@first@ofthree	103,	\@glo@plural	59, 75, 77, 78, 359
116, 119, 121, 134, 136, 138, 140,	344–346	\@glo@pluralaccess	327, 329, 330
\@first@oftwo	21,	\@glo@prefix	7, 61, 76, 81, 82, 107, 108,
22, 66, 67, 103, 119, 120, 136,	137, 140, 141	172, 307, 308	
\@for	22,	\@glo@range	172, 307, 308
27, 29, 40, 41, 48, 66, 67, 108,	153–155,	\@glo@see	60, 75, 80
161, 162, 169, 174, 180, 211, 225,	227,	\@glo@seeautonumberlist	7, 60
229, 231, 233, 236, 238, 240, 255,	256, 308	\@glo@short	53, 62, 76, 79, 360
\@glo@@desc	79	\@glo@shortaccess	328–330, 357–360
\@glo@@symbol	79	\@glo@shortpl	62, 76, 79,
\@glo@access	327, 329, 330, 332	224, 226, 228, 230, 232, 235,	237, 357, 360
\@glo@addchildren	180, 185	\@glo@shortpluralaccess	328–330
\@glo@assign@sortkey	162, 164, 192	\@glo@sort	10, 17, 18, 59, 75, 78, 81, 82
\@glo@check@mkidxrangechar	107, 172, 307, 308	\@glo@sortedinsert	181
\@glo@childlist	180	\@glo@sortentries	183, 184
\@glo@counter	60, 75, 78	\@glo@sorthandler@case	184
\@glo@counterprefix	166, 171–173, 200, 203	\@glo@sorthandler@letter	183
\@glo@default@sorttype	9, 164, 183, 184	\@glo@sorthandler@nocase	184
\@glo@defaultcounter	78	\@glo@sorthandler@word	183
\@glo@desc	59, 74–76, 79	\@glo@sortinghandler	180, 182
\@glo@descaccess	328–330	\@glo@sortinglist	180, 181, 184
		\@glo@sorttype	164, 185, 186, 192

\@glo@storeentry	10, 12	\@gls@checkescactual	109
\@glo@suffiix	107, 108, 172, 308	\@gls@checkescbar	109
\@glo@symbol	52, 60, 75, 79, 230, 235, 329	\@gls@checkescquote	108
\@glo@symbolaccess	328–330, 360	\@gls@checklevel	109
\@glo@symbolplural	60, 75, 79	\@gls@checkmkidxchars	
\@glo@symbolpluralaccess	328–330	81, 107, 163, 171, 173, 307, 308
\@glo@text	59,	\@gls@checkquote	108
75, 78, 79, 116–121, 142, 143, 230, 249, 359		\@gls@classI	154
\@glo@textaccess ...	327, 329, 330, 357–360	\@gls@classII	154
\@glo@thislabel	80, 81	\@gls@codepage	179
\@glo@thislettergrp	186, 187	\@gls@counter	
\@glo@thisvalue	53, 54	102, 105–107, 150, 151, 166, 172, 173, 308	
\@glo@tmp	68, 69, 173	\@gls@counterwithin	9, 190, 193
\@glo@type		\@gls@ctr	40
... 6, 11, 60, 75–80, 150, 151, 161, 167,		\@gls@currentlettergroup	185, 187
177–181, 185, 186, 189, 190, 209, 225,		\@gls@declareoption	
227, 229, 231, 233, 236, 238, 240, 255, 256		7, 8, 12, 13, 16, 21, 24–27
\@glo@types		\@gls@default	91
.... 48, 49, 55, 85, 151, 161, 162, 246, 301		\@gls@default@value	
\@glo@useri	61, 76, 78	52–54, 63, 64, 75, 77–79, 234, 248
\@glo@useri.. ..	61, 76, 78	\@gls@deffile	66, 67
\@glo@useri.. ..	61, 76, 79	\@gls@defsort	10, 11, 79
\@glo@useriv	61, 76, 79	\@gls@defsortcount	10, 11, 56
\@glo@userv	61, 76, 79	\@gls@do@acronymsdef	13, 28, 57
\@glo@uservi	62, 76, 79	\@gls@do@indexdef	27, 28, 57
\@glodesc	79	\@gls@do@numbersdef	26, 28, 57
\@glolist@	77	\@gls@do@symbolsdef	26, 57
\@gloname	79	\@gls@do@symbolssdef	28
\@glossary@default@style		\@gls@doautomake	25, 162
..... 6, 8, 177, 200, 201, 241		\@gls@docloadedfalse	4
\@glossaryentryfield	82	\@gls@docloadedtrue	4
\@glossarysection	36	\@gls@dodeflistparser	162
\@glossarystyle	177, 178, 189	\@gls@doentrydef	100, 101
\@glossarysubentryfield	82	\@gls@dolast	174, 175
\@gls	116	\@gls@donext	174, 175
\@gls@	87, 89, 116, 251, 252	\@gls@donext@def	149, 150
\@gls@@link	103	\@gls@dothiswrite	160, 161
\@gls@Hcounter	106, 107	\@gls@elem	255
\@gls@ReturnAfterFi	203	\@gls@encapchar	
\@gls@access@display	332, 333	111, 112, 157, 172, 174, 308, 311
\@gls@actualchar	82, 110, 113, 157, 311	\@gls@entry@count	88
\@gls@addpredefinedattributes .	153, 159	\@gls@entry@field .	69, 86, 143–149, 330–332
\@gls@adjustmode	151	\@gls@escbsdq	108, 158, 312
\@gls@automake	160, 162	\@gls@expand@fields	64, 65
\@gls@between	256, 257	\@gls@expandonce	64
\@gls@body	143	\@gls@fetchfield	53
\@gls@checkactual	109	\@gls@field@link	70, 71, 122–134
\@gls@checkbar	109	\@gls@firsttok	186
\@gls@checkedmkidx	108–114	\@gls@fixbraces	80

<code>\@gls@forbidtexext</code>	56	<code>\@gls@numberlink</code>	203
<code>\@gls@get@counterprefix</code>	173	<code>\@gls@numbersdef</code>	26
<code>\@gls@getbody</code>	143	<code>\@gls@numlist@lastsep</code>	149, 150
<code>\@gls@getcounterprefix</code>	172	<code>\@gls@numlist@nextsep</code>	149, 150
<code>\@gls@getgrouptitle</code>	163, 199, 257	<code>\@gls@numlist@sep</code>	149, 150
<code>\@gls@glossary</code>	168	<code>\@gls@old@chapter</code>	28
<code>\@gls@gobbleopt</code>	55	<code>\@gls@oldnewglossaryentryposthook</code> ..	329
<code>\@gls@grptitle</code>	199, 255, 257	<code>\@gls@oldnewglossaryentryprehook</code> ..	329
<code>\@gls@hyp@opt</code>	70,	<code>\@gls@onlypremakeg</code>	29
71, 88–90, 103, 116–141, 206–208, 250–253		<code>\@gls@order</code>	160
<code>\@gls@hyp@opt@cs</code>	103	<code>\@gls@org@LT@output</code>	272
<code>\@gls@hypergroup</code>	255	<code>\@gls@org@glsnoidxdisplayloc</code>	165
<code>\@gls@ifinlist</code>	40	<code>\@gls@org@glssseeformat</code>	165
<code>\@gls@ifnotmeasuring</code>	83	<code>\@gls@preglossaryhook</code>	178
<code>\@gls@igtype</code>	58	<code>\@gls@prevlevel</code> ..	282–284, 302–304, 320, 321
<code>\@gls@increment@currcount</code>	86	<code>\@gls@provide@newglossary</code>	56
<code>\@gls@indexdef</code>	27	<code>\@gls@quotechar</code>	109–113, 157, 311
<code>\@gls@islistofacronyms</code>	14	<code>\@gls@reference</code>	163, 166
<code>\@gls@keylist</code>	356	<code>\@gls@removespaces</code>	203
<code>\@gls@keymap</code>	66–69, 248, 329	<code>\@gls@renewglossary</code>	159
<code>\@gls@label</code>	163, 166, 172	<code>\@gls@replacementtext</code>	332
<code>\@gls@langmod</code>	160	<code>\@gls@rest</code>	143
<code>\@gls@levelchar</code>	82, 111–113, 157, 311	<code>\@gls@roman</code>	43, 308, 309
<code>\@gls@link</code> ..	103, 116–122, 134–141, 344–346	<code>\@gls@sanitized@tmp</code>	108
<code>\@gls@link@checkfirsthyper</code> ..	104, 116–121	<code>\@gls@sanitizedesc</code>	23
<code>\@gls@link@label</code>	105, 226, 232	<code>\@gls@sanitizesort</code>	10
<code>\@gls@link@nocheckfirsthyper</code> ..	122, 134–141	<code>\@gls@sanitizesymbol</code>	23
<code>\@gls@link@opts</code>	105, 226, 232	<code>\@gls@saveentrycounter</code>	105, 151
<code>\@gls@list</code>	255, 256	<code>\@gls@setacrstyle</code>	23, 28
<code>\@gls@listsuffix</code>	40	<code>\@gls@setcounter</code>	56
<code>\@gls@loadlist</code>	8, 240	<code>\@gls@setdefault@glslink@opts</code>	105
<code>\@gls@loadlong</code>	7, 8, 241	<code>\@gls@setsort</code>	10, 11, 105
<code>\@gls@loadsuper</code>	8, 241	<code>\@gls@setupshortcuts</code>	28
<code>\@gls@loadtree</code>	8, 241	<code>\@gls@sort</code>	187
<code>\@gls@local@increment@currcount</code> ..	87	<code>\@gls@sort@A</code>	182, 183
<code>\@gls@loclist</code>	164, 165, 186, 187	<code>\@gls@sort@B</code>	182, 183
<code>\@gls@map</code>	66–68	<code>\@gls@startswithexpandonce</code>	64
<code>\@gls@missingnumberlist</code>	79	<code>\@gls@symbolsdef</code>	26
<code>\@gls@noaccess</code>	332	<code>\@gls@this</code>	169
<code>\@gls@noexpand@fields</code>	65	<code>\@gls@thisHloc</code>	173
<code>\@gls@nohyperlist</code>	15, 58, 105	<code>\@gls@thisfield</code>	53
<code>\@gls@noidx@do</code>	185	<code>\@gls@thislabel</code>	51, 174, 175, 184
<code>\@gls@noidx@getgrouptitle</code>	163, 199	<code>\@gls@thislist</code>	149, 150
<code>\@gls@noidx@sanitizesort</code>	18, 165	<code>\@gls@thisloc</code>	173
<code>\@gls@noidx@setsanitizesort</code>	20, 165	<code>\@gls@thisval</code>	67
<code>\@gls@noidx@loclist@finalsep</code>	165	<code>\@gls@title</code>	36
<code>\@gls@noidx@loclist@prev</code>	165, 188	<code>\@gls@tmp</code> ...	11, 31, 44, 64, 108, 168, 256, 257
<code>\@gls@noidx@loclist@sep</code>	165, 188	<code>\@gls@tmpb</code>	109–114
<code>\@gls@noref@warn</code>	163, 186	<code>\@gls@toc</code>	38

\@gls@type	162, 211,	\@glsopenfile	159, 167
225, 227, 229, 231, 233, 236, 238, 240, 301		\@glsorder	161
\@gls@updatechecked	108, 109	\@glspl	118
\@gls@usetranslator	21, 22, 31	\@glspl@	87, 90, 118, 251–253
\@gls@value	63, 64, 146	\@glsplural	124
\@gls@warnonglossdefined	16, 176	\@glsplural@	124
\@gls@warnontheGLOSSdefined	16, 195	\@glsreset	83, 87
\@gls@write@entrycounts	88	\@glssee	80, 174
\@gls@writedef	66	\@glsymbol	128
\@gls@xdy@locationlist	154	\@glsymbol@	128
\@gls@xdycheckbackslash	108	\@glsymbolplural	128
\@gls@xdycheckquote	108	\@glsymbolplural@	128, 129
\@gls@xref	173, 174	\@glstarget	115, 116, 195, 255
\@gls@Alphacompositor	34, 44, 309	\@glstext	122
\@gls@Hlocref	171, 172	\@glstext@	122
\@gls@acronymlists ..	14, 15, 48, 209, 211,	\@glsunset	84, 86
225, 227, 229, 231, 233, 236, 238, 240, 245		\@glsuseri	129
\@gls@addkey	69	\@glsuseri@	129
\@gls@addstoragekey	68	\@glsuserii	130
\@gls@addxdyattribute	41	\@glsuserii@	130
\@gls@defaultsort	10	\@glsuseriii	131
\@gls@desc	126	\@glsuseriii@	131
\@gls@desc@	126	\@glsuseriv	132
\@gls@descplural	127	\@glsuseriv@	132
\@gls@descplural@	127	\@glsuserv	132, 133
\@gls@disp	121	\@glsuserv@	133
\@gls@entry	85, 88	\@glsuservi	133
\@gls@entrytitlecase	146	\@glsuservi@	133
\@gls@first	123	\@gls@widestname	301–303, 320
\@gls@first@	123	\@gls@writefiles	25
\@gls@firstletter	48, 153	\@gobble	11,
\@gls@firstplural	124	66, 67, 108, 152, 155, 163, 306, 310, 311	
\@gls@firstplural@	124	\@idxitem	279, 280, 297
\@gls@hypernumber	202	\@ifclassloaded	4, 9, 37
\@gls@isacronymlistfalse	14	\@ifnextchar	56, 103
\@gls@isacronymlisttrue	14	\@ifpackageloaded	4, 21, 22, 31, 47, 83, 149, 326
\@gls@link	105, 115, 150, 255	\@ifstar	55, 68, 69, 103, 205
\@gls@localreset	83, 87	\@ifundefined	
\@gls@localunset	84, 87	30, 256, 263, 273, 284, 291, 303, 320, 334	
\@gls@locref	166, 171, 172, 307, 308	\@ignored@glossaries	57, 58
\@gls@minrange	153, 154, 309	\@input@	178
\@gls@name	125	\@istfilename	161
\@gls@name@	125	\@makecol	272, 273
\@gls@nextpages	178	\@makeglossary	161
\@gls@nodesc	75, 76, 79	\@minus	259, 279, 296
\@gls@noname	75, 77, 79	\@mkboth	37
\@gls@nonextpages	177	\@newglossary	54, 56
\@gls@numberformat		\@newglossaryentry@defcounters ..	80, 86
102, 105, 150, 151, 166, 172, 307, 308			

\@newglossaryentryposthook	\@this@ctr	155
..... 68, 69, 80, 248, 329	\@this@key	67
\@newglossaryentryprehook	\@this@label	180
..... 68, 69, 74, 76, 248, 329	\@thiscs	29
\@nil	\@tmp	43, 309
143, 172, 174, 186, 187, 202, 203, 307, 308	\@use@ption	27
\@nnil	\@warn@nomakeglossaries	161, 180
14, 174	\@wrglossary@pageformat	170
\@no@makeglossaries	\@wrglossarynumberhook	170, 171
162, 164	\@xdy@main@language	24, 160, 179
\@no@post@desc	\@xdy@attributelist	41, 155
313	\@xdy@attributes	41, 154, 306, 308
\@nopostdesc	\@xdy@counters	40, 41, 155
178	\@xdy@language	179
\@onelevel@sanitize	\@xdy@lettergroups	48, 157, 311
17, 18,	\@xdy@locationclassorder	45, 155, 310
43, 66, 82, 108, 156, 173, 175, 186, 309, 310	\@xdy@locref	41, 156, 306, 310
\@onlypreamble ...	\@xdy@requiredstyles	46, 153, 308
56, 65, 75, 88, 91, 162, 166	\@xdy@sortrules	46, 157, 311
\@onlypremakeg	\@xdystyle	153, 308
33, 34, 40, 42, 45, 56	\@xdy@useralphabets	42, 154, 308
\@org@glossaryentrynumbers	\@xdy@userlocationdefs	44, 45, 155, 307, 309
177, 178	\@xdy@userlocationnames	45, 307
\@org@gls@assign@descplural	\@xfor@nextelement	174
.....	\\	81, 108, 152, 157,
224, 233, 235–238, 357, 360, 361	158, 202, 203, 311, 312, 314–316, 324, 325
\@org@gls@assign@firstpl	\{	66, 152, 158, 306, 311, 312
224,	\}	66, 67, 152, 158, 306, 312
226–228, 230, 231, 233, 235–238, 357–361	\^	18
\@org@gls@assign@plural	\‘	18
224,	\ 	109, 111
226–228, 230, 231, 233, 235–238, 357–361	\~	18
\@org@gls@assign@symbolplural ..		
224,		
226–228, 230, 231, 236–238, 358, 359, 361		
\@org@gls@numberformat		
149		
\@org@newglossaryentryprehook		
74		
\@outputpage		
272, 273		
\@p@glossarysection		
36		
\@pgls		
250		
\@pgls@		
250		
\@pglspl		
251		
\@pglspl@		
251		
\@plus		
259, 279, 296		
\@print@glossary		
176		
\@print@noidx@glossary		
176		
\@printgloss@setsort		
162, 164, 177		
\@printglossary		
176		
\@roman		
43, 308		
\@secondofthree		
103, 116, 117, 119, 135, 137, 139, 141, 344		
\@secondoftwo		
21, 22, 31, 66, 68, 115–		
118, 121, 134–136, 138, 139, 344–346, 364		
\@set@glo@numformat		
172, 308		
\@sglsaddkey		
69		
\@sglsaddstoragekey		
68		
\@thirdofthree		
103, 118, 120, 136, 137, 139, 141, 344		
\@this@attr		
155		
\@this@childlabel		
180, 181		
\@this@counter		
41		

A

\AA	19
\aa	19
accsupp package	326
\accsuppglossaryentryfield	326
\accsuppglossarysubentryfield	327
\acrfootnote	226, 232
\Acrfull	223
\acrfull	223
\ACRfullfmt	207, 210, 218, 220, 352, 354
\Acrfullfmt	207, 210, 218, 220, 352, 354
\acrfullfmt	206, 210, 218, 220, 352, 354
\acrfullformat	148, 149, 206, 224, 239
\Acrfullpl	223
\acrfullpl	223
\ACRfullplfmt ...	208, 210, 218, 220, 352, 354

\acrfullplfmt ...	208, 210, 218, 220, 352, 354	\begingroup	5, 168, 171, 203
\acrfullplfmt ...	207, 210, 218, 220, 352, 354	\bfseries	264,
\acrlinkfootnote	225	266–270, 275–278, 285–290, 292, 294–296	
\acrlinkfullformat	206–208	\bgroup	18, 74, 149, 177, 180
\Acrlong	223	booktabs package	269–272
\acrlong	222	\boolean	239
\Acrlongpl	223	\boolfalse	26
\acrlongpl	222	\booltrue	26
\acrnameformat	230, 359	\bottomrule	269, 270
\acronymentry		\box	273
210, 212–217, 219–222, 348–350, 353–356			
acronymfont	98, 134–	C	
137, 143, 148, 149, 209, 210, 212–222,		\c	18
226, 227, 229, 231, 232, 234–236, 341,		\c@equation	106
342, 344–346, 348–350, 352–356, 358–360		\c@glossarysubentry	191
\acronymname	13, 32	\c@page	169–171
\acronymsort	210,	\cGls	89
212–217, 219–222, 348–350, 353, 354, 356		\cgls	89
\acronymtype	13,	\cGlsformat	87
209–211, 224–233, 235–238, 240, 357–360		\cglsformat	87
\acrpluralsuffix		\cGlSpl	90
210, 212–215, 219–221, 224–228, 230–		\cglspl	90
237, 239, 240, 348, 349, 353–355, 357–361		\cGlSplformat	87
\Acrshort	222	\cglsplformat	87
\acrshort	222	\changes	104, 160, 260
Acrshortpl	222	\char	199
\acrshortpl	222	\cleardoublepage	38
\addcontentsline	39	\clearpage	38
\addglossarytocaptions	31	\closeout	66, 157, 158, 160, 167
\adddtolength	303, 320	\compatglossarystyle	313–325
\advance	11, 77, 106, 272	\compatibleglossentry	197
\AE	19	\compatiblesubglossentry	197
\ae	19	\copy	272, 273
amsgen package	4, 101	\count@	186
amsmath package	83	\cs	104
\andname	175	\csdef	16, 17, 68–71,
\AnyTrackedLanguages	31, 364	80, 81, 86, 87, 180, 181, 201, 211, 212, 312	
\appto	15, 68, 69, 248, 329	\csedef	88, 170
array package	269, 273, 291	\csgdef	36, 54, 58, 87, 88, 175, 189
article class	173	\cslet	74, 81, 184
\AtBeginDocument	13, 47, 66, 83, 151, 163	\csname .	9–11, 27, 29, 31, 32, 37, 38, 41, 43,
\AtEndDocument .	25, 66, 88, 163, 167, 179, 256	44, 47, 48, 51, 56, 57, 63, 64, 66, 68–73,	
B		77–82, 84, 85, 100, 101, 105–107, 116–	
\b	19	121, 134–142, 149, 151, 154, 155, 159,	
babel package	21, 29, 31, 46	160, 163, 166–170, 172–174, 177–179,	
\begin	104, 156, 160,	181, 189, 195–197, 200, 201, 205, 241–	
185, 260, 263–269, 271, 272, 274–296, 310		249, 255, 256, 302, 303, 306–308, 320,	
\BeginAccSupp	332	326, 327, 329–331, 334, 344–346, 362, 363	
		\csshow	245

<code>\csuse</code>	32, 35, 54, 63, 64, 70, 71, 100, 101, 160, 181, 183, 185, 187, 188, 190, 191, 201, 212, 249, 313–325
<code>\csxdef</code>	79, 88
<code>\currentglossary</code>	35, 177, 190, 193
<code>\currentglssubentry</code>	191, 193, 194
<code>\CurrentOption</code>	27, 248, 326
<code>\CurrentTrackedLanguage</code>	32, 364, 365
<code>\CurrentTrackedTag</code>	32, 364, 365
<code>\CustomAcronymFields</code>	240
<code>\CustomNewAcronymDef</code>	240
D	
<code>\d</code>	18
<code>datatool package</code>	182
<code>\day</code>	153, 157, 308, 311
<code>\DeclareAcronymList</code>	13, 15, 209, 211, 225, 227, 229, 231, 233, 236, 238, 240
<code>\DeclareListParser</code>	162
<code>\DeclareOption</code>	7, 248, 326
<code>\DeclareOptionX</code>	7
<code>\DeclareRobustCommand</code>	32, 174, 175, 234, 332–334
<code>\def</code> .. 7, 10, 11, 14, 18, 19, 24, 28, 29, 32, 33, 36, 40, 42–48, 51, 54–62, 64, 72, 74–79, 81, 87, 89–91, 100–102, 105–114, 116– 141, 149–151, 153, 157, 160, 162, 164, 165, 167, 170–174, 177, 180, 184–186, 188–193, 199–204, 206–209, 224–231, 233, 235–238, 240, 248, 250–254, 256– 259, 282–284, 302–304, 307, 308, 311, 313, 320, 321, 326–329, 343–346, 357–361	
<code>\def@gl@xdycheckbackslash</code>	114
<code>\DefaultNewAcronymDef</code>	225
<code>\defgl@entryfmt</code>	56, 58, 100, 101, 211, 224, 225, 228, 229, 232, 234, 237, 239
<code>\define@boolkey</code>	5, 7–9, 12, 19, 20, 23–26, 102, 191
<code>\define@choicekey</code>	5, 6, 9, 20, 21, 24, 61, 190, 191
<code>\define@key</code> 6, 9, 15, 20, 24, 25, 58–62, 68, 69, 101, 102, 150, 189, 192, 248, 327, 328	
<code>\DefineAcronymSynonyms</code>	28, 223
<code>\delimN</code>	156, 162, 188, 203, 310
<code>\delimR</code>	156, 202, 203, 310
<code>\DescriptionDUANewAcronymDef</code>	229
<code>\DescriptionFootnoteNewAcronymDef</code> ..	227
<code>\descriptionname</code>	32, 264, 266, 267, 269, 270, 275–278, 285–290, 292, 294–296
<code>\DescriptionNewAcronymDef</code>	231
<code>\dimen@</code>	213, 272, 301
<code>\disable@keys</code>	27
<code>\do</code>	22, 27, 29, 40, 41, 48, 66, 67, 108, 149, 153–155, 161, 162, 169, 174, 180, 211, 225, 227, 229, 231, 233, 236, 238, 240, 255, 256, 308
<code>\do@gl@storeentry</code>	10, 11, 80
<code>\do@gl@link@checkfirsthyper</code>	103, 105, 116–122, 134–141, 344, 345
<code>\do@gl@xdycheckbackslash</code>	108
<code>\do@gl@disablehyperinlist</code>	105
<code>\do@gl@shaschildren</code>	51
<code>doc package</code>	4, 5, 12
<code>\doifglossarynoexistsordo</code>	55
<code>\dtl@ifsingle</code>	199
<code>\dtl@insertinto</code>	182
<code>\dtl@sortresult</code>	182, 183
<code>\dtlcompare</code>	182
<code>\dtlicompare</code>	183
<code>\DTLifinlist</code>	58, 105
<code>\DTLifint</code>	199
<code>\dtlletterindexcompare</code>	182
<code>\DTLsubstituteall</code>	108
<code>\dtlwordindexcompare</code>	182
<code>\DUANewAcronymDef</code>	238
E	
<code>\eappto</code>	57, 58, 81, 170
<code>\edef</code>	11, 14, 29, 31, 39–46, 48, 51, 55–58, 63, 64, 66–68, 71–76, 80–82, 100, 101, 105–114, 149, 151, 152, 157, 160, 162, 163, 167, 172, 173, 175, 179–183, 186, 191, 194, 199, 203, 205, 224, 226, 228, 230, 232, 235, 237, 255, 306, 307, 309, 311, 356–360
<code>\egroup</code>	18, 74, 150, 178, 181
<code>\else</code>	8, 11–14, 17, 19, 20, 25, 27–29, 33, 34, 37–43, 45–48, 61, 63, 77, 78, 81–83, 87, 104–114, 117–121, 143, 153, 156–160, 167–174, 177, 186, 190, 191, 193–195, 200, 202, 203, 213, 227, 229, 231, 234, 236–239, 241, 256, 260, 264, 265, 267, 269, 270, 272–274, 276, 277, 285, 286, 288, 292, 293, 295, 297–300, 302–304, 307–313, 318–321, 332
<code>\emph</code>	174, 204
<code>\empty</code>	203
<code>\end</code>	104, 156, 160, 185, 260, 263–269, 271, 272, 274–296, 310

\glossary	307, 308	longborder	264, 315
glossary package	1, 204	longheader	264, 269, 315
glossary styles:		longheaderborder	264, 315
altlist	261, 262, 314	longragged	271, 273–275
altlistgroup	261, 262, 314	longragged-booktabs	271
altlisthypergroup	262, 314	longragged3col	271, 275, 276, 317
altlong4col	268, 276, 316	longragged3col-booktabs	271
altlong4col-booktabs	270, 272	longragged3colborder	276, 317
altlong4colborder	268, 316	longragged3colheader	276, 317
altlong4colheader	268, 270, 316	longragged3colheaderborder	276, 317
altlong4colheaderborder	268, 316	longraggedborder	274, 316
altlongragged4col ...	272, 276–278, 317	longraggedheader	274, 317
altlongragged4col-booktabs	272	longraggedheaderborder	275, 317
altlongragged4colborder	278, 317	mcolalttree	283, 322
altlongragged4colheader	277, 317	mcolalttreegroup	283, 322
altlongragged4colheaderborder	278, 318	mcolalttreehypergroup	283, 322
altsuper4col	289, 290, 294, 325	mcolindex	279, 321
altsuper4colborder	290, 325	mcolindexgroup	279, 321
altsuper4colheader	289, 325	mcolindexhypergroup	279, 280, 321
altsuper4colheaderborder	290, 325	mcoltree	280, 321
altsuperragged4col	294–296, 323	mcoltreegroup	321
altsuperragged4colborder	295, 323	mcoltreehypergroup	280, 281, 321
altsuperragged4colheader	295, 323	mcoltreenoname	281, 322
altsuperragged4colheaderborder	296, 323	mcoltreenonamegroup	282, 322
alttree	282, 302, 304, 320	mcoltreenonamehypergroup	282, 322
alttreegroup	304, 321	sublistdotted	314
alttreehypergroup	304, 321	super	284–286, 292, 324
index	279, 297, 298, 318	super3col	286, 287, 324
indexgroup	298, 318	super3colborder	287, 324
indexhypergroup	298, 318	super3colheader	287, 324
inline	313	super3colheaderborder	287, 324
list	6, 260–262, 313	super4col	287–289, 325
listdotted	262, 263, 314	super4colborder	288, 325
listgroup	260, 261, 313	super4colheader	288, 325
listhypergroup	261, 314	super4colheaderborder	289, 325
long	263–265, 269, 273, 314, 316	superborder	285, 324
long-booktabs	269, 271	superheader	285, 324
long3col	265, 270, 315	superheaderborder	286, 324
long3col-booktabs	269, 271	superragged	291, 292, 322
long3colborder	265, 315	superragged3col	292–294, 323
long3colheader	265, 269, 315	superragged3colborder	293, 323
long3colheaderborder	266, 315	superragged3colheader	293, 323
long4col	266–268, 270, 315	superragged3colheaderborder	294, 323
long4col-booktabs	270, 271	superraggedborder	292, 322
long4colborder	267, 316	superraggedheader	292, 322
long4colheader	267, 270, 316	superraggedheaderborder	292, 323
long4colheaderborder	267, 316	tree	280, 298–300, 302, 318
		treegroup	280, 299, 319
		treehypergroup	299, 319

treeoname	281, 300, 301, 319	\gls@assign@desc	74, 79
treeonamegroup	301, 320	\gls@assign@descplural	
treeonamehypergroup	301, 320	224, 233, 235–238, 357, 360, 361
glossary-hypernav package	152	\gls@assign@field	
glossary-list package	6, 8, 259	65, 68, 69, 74, 76, 78, 79, 248, 249
glossary-long package	7, 263, 276, 284	\gls@assign@firstpl	224,
glossary-longragged package	273	226–228, 230, 231, 233, 235–238, 357–361
glossary-mcols package	278	\gls@assign@plural	224,
glossary-super package ..	7, 8, 263, 284, 290, 294	226–228, 230, 231, 233, 235–238, 357–361
glossary-superragged package	290	\gls@assign@symbolplural	224,
glossary-tree package	8, 296	226–228, 230, 231, 236–238, 358, 359, 361
\glossaryentry	172, 174, 308	\gls@checkisacronymlist	104
glossaryentry (counter)	9, 193, 194	\gls@checkseeallowed	60, 65, 162, 163
\glossaryentryfield		\gls@checkseeallowed@preambleonly ..	65
.....	195, 313–320, 322–325, 347	\gls@codepage	47, 160, 179
\glossaryentrynumbers		\gls@defdocnewglossaryentry	66, 86
.....	7, 156, 177, 178, 187, 191, 192, 310	\gls@defglossaryentry	65, 66, 74
\glossaryheader	156, 185, 257, 260–	\gls@disablepagerefexpansion ..	169, 171
	267, 269, 270, 274–279, 281–283, 285,	\gls@do@addxdyattribute	41
	286, 288, 291, 293, 294, 297–302, 304, 310	\gls@doclearpage	39
\glossarymark	36	\gls@dosubst	108
\glossaryname	12, 31, 32	\gls@dotocitle	177, 178, 189
\glossarypostamble	156, 185, 310	\gls@end@sanitizesort	18
\glossarypreamble	155, 185, 310	\gls@endcheck	64
\glossarysection	155, 185, 310	\gls@glossary	168, 172–174
glossarysubentry (counter)	9, 193–195	\gls@gobbleopt	56
\glossarysubentryfield		\gls@grplabel	255
.....	197, 313–320, 322–325, 347	\gls@hypergroup prerun	256
\glossarytitle	155, 177, 185, 189, 310	\gls@ifnotmeasuring	83, 84
\glossarytoctitle	6, 12, 13, 26,	\gls@inlinepostchild	257–259, 313
	27, 29, 32, 36, 155, 177, 185, 189, 190, 310	\gls@inlinesep	257, 258, 313
\glossentry	81, 178, 187, 197, 258,	\gls@inlinesubsep	257, 258, 313
	260–263, 265, 266, 274, 275, 277, 285,	\gls@islistofacronyms	14
	286, 288, 291, 293, 294, 297, 299, 300, 302	\gls@istfilebase	33, 160
\Glossentrydesc	346	\gls@label	204
\glossentrydesc	258,	\gls@level	77, 78, 187
	260–267, 274, 275, 277, 285, 286, 288,	\gls@noidxglossary	163
	291, 293, 295, 297, 299, 300, 302, 304, 346	\gls@numberpage	169, 171
\glossentryname	258, 260–263,	\gls@org@glossaryentryfield	178
	265, 266, 274, 275, 277, 285, 286, 288,	\gls@org@glossarysubentryfield	178
	291, 293, 295, 297, 299, 300, 302, 304, 346	\gls@org@insert	229, 232, 234
\Glossentrysymbol	347	\gls@protected@pagefmts	108, 169, 170
\glossentrysymbol	258, 266, 267,	\gls@Romanpage	169, 171
	277, 288, 295, 297, 299, 300, 302, 304, 347	\gls@romanpage	169, 171
\Gls	89, 205, 223	\gls@save@numberlist	7
\gls	89, 163, 194, 205, 223	\gls@suffiX	34, 156, 158, 310, 312
\gls@Alphpage	169, 171	\gls@suffiFF	35, 156, 158, 310, 312
\gls@alphpage	169, 171	\gls@text	100
\gls@arabicpage	169, 171	\gls@thissty	22

<code>\gls@tmp</code>	167, 234	<code>\glsdetoklabel</code> 49–53, 66, 71–75, 80, 84–88,	
<code>\gls@tmpen</code>	115, 301, 303, 304, 320, 321	105, 142, 143, 149–151, 163–165, 172,	
<code>\gls@tr@set@acronym@toctitle</code>	13	178, 181–183, 186, 187, 189–191, 193,	
<code>\gls@tr@set@main@toctitle</code>	12	194, 196, 241–245, 301, 326, 327, 362, 363	
<code>\gls@tr@set@numbers@toctitle</code>	27	<code>\glsdisplay</code>	91, 101
<code>\gls@tr@set@symbols@toctitle</code>	26	<code>\glsdisplayfirst</code>	91, 100
<code>\gls@wrglossary</code>	168	<code>\glsdisplaynumberlist</code>	164
<code>\gls@xdyststring</code>	108	<code>\glsdohyperlink</code>	115
<code>\gls@xindy@glsnumbersfalse</code>	25	<code>\glsdohypertarget</code>	115, 116
<code>\gls@xindy@glsnumberstrue</code>	24	<code>\glsdoifexists</code>	
<code>\glsaccsupp</code>	332	.. 51–53, 71–74, 83, 84, 116–122, 134–	
<code>\glsacronymtrue</code>	13	141, 149, 151, 164, 165, 250–254, 343–347	
<code>\glsacrpluralsuffix</code>		<code>\glsdoifexistsordo</code>	103, 142
..... 30, 205, 214, 215, 219–221, 225		<code>\glsdoifexistsorwarn</code>	189, 196, 197
<code>\glsacrshortcutsfalse</code>	28	<code>\glsdoifnoexists</code>	65, 74
<code>\glsacrshortcutstrue</code>	28	<code>\glsdonohyperlink</code>	106, 115
<code>\glsacspace</code>	213, 215	<code>\glsdosanitizesort</code>	9, 10
<code>\glsadd</code>	151	<code>\glsentryaccess</code>	332
<code>\glsadd options</code>		<code>\glsentrycounter</code>	200, 203
counter	150	<code>\glsentrycounterfalse</code>	9
format	150, 202	<code>\glsentrycounterlabel</code>	190, 191, 194
<code>\glsaddall options</code>		<code>\glsentrycountertrue</code>	9
types	150, 151	<code>\glsentrycurrcount</code>	86, 88
<code>\GlsAddXdyAttribute</code>	40–42, 306, 307	<code>\Glsentrydesc</code>	126, 196, 346
<code>\GlsAddXdyCounters</code>	40, 41, 57	<code>\glsentrydesc</code>	
<code>\glsautomakefalse</code>	25 93–95, 126, 127, 196, 336–338, 346	
<code>\glsautoprefix</code>	6, 190	<code>\glsentrydescaccess</code>	333
<code>\glscapscase</code>	91, 93, 95–99,	<code>\Glsentrydescplural</code>	127
116–121, 134–141, 217, 218, 230, 235,		<code>\glsentrydescplural</code> .. 92, 93, 127, 334, 335	
334, 336, 338, 339, 341, 342, 344–346, 351		<code>\glsentrydescpluralaccess</code>	333
<code>\glsclearpage</code>	38	<code>\Glsentryfirst</code>	89, 94, 97, 123, 337, 340
<code>\glsclsebrace</code>	45, 156, 157, 310, 311	<code>\glsentryfirst</code> . 89, 93–97, 123, 336, 337, 340	
<code>\glscompositor</code>	34, 44, 158, 309, 312	<code>\glsentryfirstaccess</code>	333
<code>\glscounter</code>	15, 28, 40, 56, 78, 106, 306	<code>\Glsentryfirstplural</code> 90, 93, 96, 125, 335, 339	
<code>\glscurrententrylabel</code>	175, 178	<code>\glsentryfirstplural</code>	90,
<code>\glscustomtext</code>		92, 93, 95, 96, 124, 125, 334, 335, 338, 339	
. 91, 95, 97, 99, 116–121, 134–141, 217,		<code>\glsentryfirstpluralaccess</code>	333
218, 225, 226, 229, 230, 232, 234, 235,		<code>\glsentryfmt</code>	56, 58
334, 337, 338, 340, 341, 343–346, 351, 352		<code>\Glsentryfull</code>	210, 219, 220, 353, 355
<code>\GlsDeclareNoHyperList</code>	15	<code>\glsentryfull</code>	210, 219, 220, 352, 355
<code>\glsdefaulttype</code>	12, 36,	<code>\Glsentryfullpl</code> 210, 219, 220, 353, 355	
47, 48, 54, 55, 76, 91, 100, 101, 167, 176, 177		<code>\glsentryfullpl</code> 210, 219, 220, 353, 355	
<code>\glsdefmain</code>	12, 57	<code>\glsentryitem</code>	
<code>\glsdescriptionaccessdisplay</code> 190, 191, 258, 260–263, 265, 266,	
..... 336–338, 346, 347		274, 275, 277, 285, 286, 288, 291, 293,	
<code>\glsdescriptionpluralaccessdisplay</code>		295, 297, 299, 300, 302, 313–320, 322–325	
..... 334, 335		<code>\Glsentrylong</code>	89, 139, 143, 148,
<code>\glsdescwidth</code>	263–265, 268,	212, 213, 218, 219, 343, 345, 348, 351–353	
269, 271, 272, 274–278, 284–287, 289–296			

<code>\glsentrylong</code>	89, 99, 138, 139, 143, 148, 212–222, 232, 343, 345–356
<code>\glsentrylongaccess</code>	333
<code>\Glsentrylongpl</code>	90, 141, 149, 212, 213, 217–219, 343, 348, 351–353
<code>\glsentrylongpl</code> 90, 100, 140, 141, 149, 212–214, 217–220, 232, 239, 343, 348, 349, 351–355
<code>\glsentrylongpluralaccess</code>	333
<code>\Glsentryname</code>	126, 196, 346
<code>\glsentryname</code>	125, 126, 301, 346
<code>\glsentrynumberlist</code>	149, 164
<code>\Glsentryplural</code>	92, 96, 124, 335, 339
<code>\glsentryplural</code> 92, 93, 95, 96, 124, 334, 335, 338, 339
<code>\glsentrypluralaccess</code>	332
<code>\Glsentryprefix</code>	252
<code>\glsentryprefix</code>	250, 253
<code>\Glsentryprefixfirst</code>	252
<code>\glsentryprefixfirst</code>	251, 253
<code>\Glsentryprefixfirstplural</code>	253
<code>\glsentryprefixfirstplural</code>	251, 254
<code>\Glsentryprefixplural</code>	252
<code>\glsentryprefixplural</code>	251, 254
<code>\glsentryprevcount</code>	86, 87
<code>\Glsentryshort</code>	98, 135, 143, 214, 220, 342–344, 348, 354, 355
<code>\glsentryshort</code> 98, 99, 134, 136, 143, 148, 210, 212–222, 341–344, 347–350, 352–356	
<code>\glsentryshortaccess</code>	333
<code>\Glsentryshorttpl</code> 98, 137, 214, 220, 341, 348, 354, 355
<code>\glsentryshorttpl</code> 98, 100, 136, 137, 149, 212– 214, 218–220, 239, 341, 343, 348, 352–355
<code>\glsentryshortpluralaccess</code>	333
<code>\Glsentrysymbol</code>	128, 197, 347
<code>\glsentrysymbol</code>	93– 95, 128, 196, 226, 230, 235, 336–338, 347
<code>\glsentrysymbolaccess</code>	333
<code>\Glsentrysymbolplural</code>	129
<code>\glsentrysymbolplural</code> 92, 93, 129, 226, 230, 235, 334, 335
<code>\glsentrysymbolpluralaccess</code>	333
<code>\Glsentrytext</code>	94, 97, 123, 336, 340
<code>\glsentrytext</code>	93, 94, 96, 97, 122, 150, 175, 336, 337, 339, 340
<code>\glsentrytextaccess</code>	332
<code>\glsentrytype</code>	76
<code>\Glsentryuseri</code>	130
<code>\glsentryuseri</code>	129, 130
<code>\Glsentryuserii</code>	130
<code>\glsentryuserii</code>	130, 131
<code>\Glsentryuseriii</code>	131
<code>\glsentryuseriii</code>	131, 132
<code>\Glsentryuseriv</code>	132
<code>\glsentryuseriv</code>	132
<code>\Glsentryuserv</code>	133
<code>\glsentryuserv</code>	133
<code>\Glsentryuservi</code>	134
<code>\glsentryuservi</code>	133, 134
<code>\glsfieldfetch</code>	146
<code>\glsfirstaccessdisplay</code>	336, 337, 340
<code>\glsfirstpluralaccessdisplay</code> 334, 335, 338, 339
<code>\glsfirstpluralaccessdisplay</code>	339
<code>\glsгенacfmt</code>	212, 213, 219, 347, 348, 353
<code>\glsгенentryfmt</code> 212, 213, 218, 219, 224, 226, 228– 230, 232, 234, 237, 239, 347, 348, 352, 353
<code>\glsgetgrouptitle</code> 257, 261, 262, 279–284, 298–301, 304, 305
<code>\gls glossarymark</code>	36
<code>\gls groupheading</code>	157, 187, 257, 260–263, 265, 266, 274, 275, 277, 279– 286, 288, 291, 293, 294, 297–302, 304, 311
<code>\gls groupskip</code> 156, 157, 187, 258, 260, 264, 265, 267, 269, 270, 274, 276, 277, 285, 286, 288, 292, 293, 295, 298–300, 304, 310	
<code>\glshyperfirstfalse</code>	219, 353
<code>\glshyperfirsttrue</code>	23
<code>\glshyperlink</code>	175
<code>\glshypernavsep</code>	257
<code>\glshypernumber</code>	35, 203, 204
<code>\glsifhyperon</code>	102
<code>\glsIfListOfAcronyms</code>	14
<code>\glsifplural</code>	91, 95, 97, 98, 116–121, 134–141, 217, 226, 230, 232, 234, 334, 338, 341, 342, 344–346, 351
<code>\glsifusetranslator</code>	21, 22, 31, 32, 364
<code>\glsindexonlyfirstfalse</code>	23
<code>\glsinlinedescformat</code>	258, 313
<code>\glsinlinedopostchild</code>	258, 313
<code>\glsinlineemptydescformat</code>	258, 313
<code>\glsinlinenameformat</code>	258, 313
<code>\glsinlineparentchildseparator</code>	258, 313
<code>\glsinlinepostchild</code>	258, 313
<code>\glsinlineseparator</code>	258, 313

<code>\glsinlinesubdescformat</code>	258, 313	<code>\glsnoidxloclisthandler</code>	188
<code>\glsinlinesubnameformat</code>	258, 313	<code>\glsnoidxnumberlistloophandler</code>	165
<code>\glsinlinesubseparator</code>	258, 313	<code>\glsnoidxstripaccents</code>	18
<code>\glsinsert</code>	92–	<code>\glsnnextpages</code>	61, 177
99, 116–121, 134–141, 217, 218, 226,		<code>\glsnopostdotfalse</code>	8
229, 230, 232, 234, 235, 334–346, 351, 352		<code>\glsnoxindywarning</code> 34, 40, 42, 43, 45–47, 153	
<code>\glskeylisttok</code>	209,	<code>\glsnumberformat</code>	149
210, 224–231, 233, 235–238, 240, 356–361		<code>\glsnumberlistloop</code>	165
<code>\glslabel</code> ... 75, 92–99, 104–106, 135–141,		<code>\glsnumbersgroupname</code>	26, 32, 200
212, 213, 217–219, 225, 226, 229, 230,		<code>\glsnumlistlastsep</code>	150, 165
232, 234, 235, 334–343, 347, 348, 351–353		<code>\glsnumlistparser</code>	150, 162
<code>\glslabeltok</code>		<code>\glsnumlistsep</code>	150, 165
.... 209, 210, 224–233, 235–240, 357–360		<code>\glsopenbrace</code>	45, 156, 157, 310, 311
<code>\glslink</code>	210, 218, 220, 225, 352, 354	<code>\glsorder</code>	24, 160, 161, 183
<code>\glslink options</code>		<code>\glsorg@endtheglossary</code>	5
counter	101, 116, 247	<code>\glsorg@PrintChanges</code>	5
format	102, 116, 202	<code>\glsorg@theglossary</code>	5
hyper	102, 104, 116	<code>\glspagelistwidth</code> ... 265, 268, 269, 271,	
local	102	272, 275–278, 286, 287, 289, 290, 293–296	
<code>\glslinkcheckfirsthyperhook</code>	104	<code>\glspatchLTOutput</code>	269–272
<code>\glslinkpostsetkeys</code>	105	<code>\glspenaltygroupskip</code>	269, 270
<code>\glslinkvar</code>	102, 103	<code>\glsperscentchar</code>	66, 67, 156, 157
<code>\glslistdottedwidth</code>	262, 314	<code>\Glspl</code>	90, 223
<code>\glslistgroupheaderfmt</code>	261, 262	<code>\glspl</code>	90, 223
<code>\glslistnavigationitem</code>	261, 262	<code>\glspluralaccessdisplay</code> 334, 335, 338, 339	
<code>\glslocalreset</code>	85	<code>\glspluralsuffix</code>	
<code>\glslocalunset</code>	85, 117–121 30, 78, 212–214, 348, 349, 353–355	
<code>\glslongaccessdisplay</code>	343, 345–357	<code>\glspostdescription</code>	33,
<code>\glslongkey</code>	361	259–261, 263, 264, 274, 285, 291, 297,	
<code>\glslongpluralaccessdisplay</code>		299, 300, 302, 304, 313–316, 318–322, 324	
.... 343, 348, 349, 351–355, 357		<code>\glspostinline</code>	257
<code>\glslongpluralkey</code>	361	<code>\glspostlinkhook</code>	
<code>\glslongtok</code> 104, 117–122, 135–141, 344–346	
.... 209, 210, 212, 213, 218, 219, 224–		<code>\glsprestandardsort</code>	10
233, 235–240, 347, 348, 352, 353, 356–361		<code>\glsreset</code>	85
<code>\glsLTpenaltycheck</code>	273	<code>\glsresetentrycounter</code>	190, 191, 193
<code>\glsmcols</code>	279–283	<code>\glsresetentrylist</code>	156, 185, 192, 310
<code>\glsnameaccessdisplay</code>	346, 347	<code>\glsresetsubentrycounter</code> 191, 194, 258, 313	
<code>\glsnamefont</code>	195–197, 326, 327, 346	<code>\glssanitizesortfalse</code>	20
<code>\glsnavhyperlink</code>	257	<code>\glssanitizesorttrue</code>	20
<code>\glsnavhypertarget</code>		<code>\glssavenumberlistfalse</code>	7
.... 261, 262, 279–284, 298, 300, 301, 305		<code>\glssavewritesfalse</code>	26
<code>\glsnavigation</code>		<code>\glseeformat</code>	155, 163, 165, 310
.... 261, 262, 279–283, 298, 299, 301, 304		<code>\glseeitem</code>	175
<code>\glsnextpages</code>	7, 61, 178	<code>\glseeitemformat</code>	175
<code>\glsnogroupskipfalse</code>	8	<code>\glseeelastsep</code>	175
<code>\glsnoidxdisplayloc</code>	165, 166	<code>\glseeelist</code>	174
<code>\glsnoidxdisplayloclisthandler</code>	165	<code>\glseeesep</code>	175
<code>\glsnoidxloclist</code>	164, 187	<code>\glsssetexpandfield</code>	17, 19, 20

<code>\ifdefstring</code> ..	31, 54, 160, 183, 184, 186, 188
<code>\ifdefvoid</code>	18, 80, 187
<code>\ifdim</code>	213, 272, 301
<code>\iffalse</code>	80, 84
<code>\IfFileExists</code>	7, 21, 22, 179
<code>\ifglossaryexists</code>	36, 47, 51, 159, 160
<code>\ifgl\$@sanitize@description</code>	19
<code>\ifgl\$@sanitize@name</code>	19
<code>\ifgl\$@sanitize@symbol</code>	19
<code>\ifgl\$@xindy@gl\$numbers</code>	47
<code>\ifgl\$acrdescription</code>	238
<code>\ifgl\$acrdua</code>	228, 234, 237–239
<code>\ifgl\$acrfootnote</code>	104, 238, 239
<code>\ifgl\$acronym</code>	12, 13
<code>\ifgl\$acrshortcuts</code>	28, 223
<code>\ifgl\$acrsmallcaps</code>	227, 229, 231, 233, 236, 237
<code>\ifgl\$acrsmaller</code>	227, 229, 231, 234
<code>\ifgl\$automake</code>	25, 162
<code>\ifgl\$descsuppressed</code>	258
<code>\ifgl\$entrycounter</code>	190, 191, 193, 194
<code>\ifgl\$entryexists</code>	50, 51, 65, 66, 75, 77
<code>\ifgl\$haschildren</code>	258, 313
<code>\ifgl\$hasdesc</code>	258
<code>\ifgl\$haslong</code>	89, 90, 143, 212, 213, 217, 219, 232, 347, 348, 351, 353
<code>\ifgl\$hasparent</code>	181, 187, 301
<code>\ifgl\$hasprefix</code>	252
<code>\ifgl\$hasprefixfirst</code>	252
<code>\ifgl\$hasprefixfirstplural</code>	252
<code>\ifgl\$hasprefixplural</code>	252
<code>\ifgl\$has\$ymbol</code>	226, 230, 234, 297, 299, 300, 302, 304
<code>\ifgl\$hyperfirst</code>	104
<code>\ifgl\$indexonlyfirst</code>	169
<code>\ifgl\$nogroupskip</code>	260, 264, 265, 267, 269, 270, 274, 276, 277, 285, 286, 288, 292, 293, 295, 298–300, 304
<code>\ifgl\$nonumberlist</code>	191
<code>\ifgl\$nopostdot</code>	8
<code>\ifgl\$numberline</code>	39
<code>\ifgl\$sanitizesort</code>	17, 20
<code>\ifgl\$save\$umberlist</code>	63, 162, 175
<code>\ifgl\$save\$writes</code>	25, 159, 168
<code>\ifgl\$subentrycounter</code>	191, 193–195
<code>\ifgl\$stoc</code>	39
<code>\ifgl\$stranslate</code>	31
<code>\ifgl\$sucmark</code>	37
<code>\ifgl\$used</code>	88, 92–97, 104, 151, 169, 225, 229, 232, 234, 250–254, 334–341
<code>\ifgl\$swallowprimitivemods</code>	171
<code>\ifgl\$xindy</code>	33, 34, 39–47, 57, 81, 82, 108, 152, 153, 160, 172, 173, 179, 306–308
<code>\ifignoredglossary</code>	76, 80, 169
<code>\ifin@</code>	27
<code>\ifinlistcs</code>	185, 189
<code>\ifKV@gl\$link@hyper</code>	104–106
<code>\ifKV@gl\$link@local</code>	117–121
<code>\ifmeasuring@</code>	83
<code>\ifnum</code>	10, 87, 186, 272, 299, 300, 302, 303, 319, 320
<code>\ifstre\$empty</code>	313
<code>\ifstre\$equal</code>	199
<code>\ifthenelse</code>	20, 28, 38, 106, 161, 200, 239, 255
<code>\IfTrackedLanguageFileExists</code>	32, 364, 365
<code>\iftrue</code>	80, 84
<code>\ifundef</code>	55, 66, 76, 153, 157, 161, 191
<code>\ifvmode</code>	151
<code>\ifvoid</code>	273
<code>\ifx</code>	10, 11, 14, 27, 29, 40, 41, 43, 44, 47, 48, 76–79, 82, 106, 107, 109–114, 143, 153, 156, 158, 167, 170, 171, 173, 174, 177, 190, 193, 200–203, 225, 227, 229, 231, 233, 234, 236, 238, 240, 241, 308–310, 312, 313, 318–321, 332
<code>\immediate</code>	66, 67, 88, 159, 167, 179
<code>\in@</code>	27
<code>\index</code>	168
<code>\indexname</code>	27
<code>\indexspace</code>	260, 279–284, 298–301, 304, 305
<code>\input</code>	30, 91
<code>\inputencodingname</code>	24
<code>\InputIfFileExists</code>	66
<code>\istfilename</code>	33, 153, 157, 160, 161, 308, 311
<code>\item</code>	260–263, 279, 280, 297, 298, 313, 314, 318
J	
<code>\jobname</code>	33, 66, 153, 157, 159, 160, 178, 179, 308, 311
K	
<code>\key@ifundefined</code>	68, 69
<code>\KV@gl\$link@hyperfalse</code>	102, 104, 105, 115
<code>\KV@gl\$link@hypertrue</code>	102, 115
L	
<code>\L</code>	19
<code>\l</code>	19

\label	6, 190, 191, 193, 194	\MakeUppercase	335, 337, 344, 346
\language	24	\markboth	37
\leaders	262, 314	\mbox	151, 261, 283, 284, 302, 314
\leavevmode	74, 105	memoir class	168
\let	5, 8, 10–13, 18, 19, 21, 22, 25–28, 31, 33, 41, 42, 54, 63, 65, 74–77, 79, 80, 83, 84, 86, 88, 91, 103, 105, 106, 108, 115–122, 134–141, 143, 149, 150, 157–165, 168–171, 174, 175, 177, 178, 187, 189, 192, 197, 209, 222– 224, 226–238, 248, 255–257, 272, 279, 280, 297, 312, 329, 344–346, 357–361, 364	\memUchead	37
\letcs	51– 53, 68, 69, 73, 78, 142, 143, 160, 164, 165, 180, 182, 183, 186, 187, 196, 199, 301	\MessageBreak	32, 55, 177, 364, 365
link text	91	mfirstuc package	1
\listcsadd	185	\mfirstucMakeUppercase	4, 37, 71, 93–99, 122–134, 136, 137, 139, 141, 210, 217–220, 230, 235, 253, 254, 339–343, 351, 352, 354, 355
\listcsgadd	189	\midrule	269, 270
\listcsxadd	180, 181	\month	153, 157, 308, 311
\listead	184	multicol package	278
\loadglsentries	91	N	
\long	74, 203	\n	157, 158, 311
\longnewglossaryentry	75	\NeedsTeXFormat	4, 248, 306, 312, 326, 364
longtable package	263, 269, 273	\new@glossaryentry	65, 164
\LT@end@open	272	\new@ifnextchar	55, 70, 71, 89, 90, 116–120, 122–141, 206–208, 250–253
\LT@err	272	\newacronym	204, 209, 225, 227, 229, 231, 233, 236, 238, 240
\LT@foot	272, 273	\newacronymhook	210, 225, 227, 229, 231, 233, 236, 238, 240, 356
\LT@head	273	\newacronymstyle	212–217, 219, 221, 222
\LT@lastfoot	272, 273	\newcommand	5–18, 20, 21, 23–31, 33–75, 80, 81, 83–86, 88–91, 95, 97, 99–106, 108, 115–153, 159–161, 163, 166–178, 180–186, 188, 189, 192–213, 222, 224–246, 249–253, 255–260, 272, 279, 296, 302, 312, 330–332, 347, 361–363
\LT@output	272	\newcount	10, 11, 63
M		\newcounter	190, 191, 193
\makeatletter	66, 178	\newenvironment	195
\makeatother	66	\newglossary	12, 13, 26, 27, 56, 161
\makebox	262, 302, 304, 314, 320, 321	\newglossaryentry	27, 62, 65, 86, 210, 224, 226, 228, 230, 232, 235, 237, 239, 357–360
makeglossaries	24, 33, 47, 55, 161, 179	\newglossaryentry options	
\makeglossaries	25, 29, 60, 61, 163, 164, 166, 180	access	329, 330
\makeglossary	159, 161	counter	60
makeindex	366	description	23, 58, 62, 65, 75, 126, 144, 205, 233, 328
makeindex	9, 24, 25, 30, 33–35, 55, 57, 59, 82, 107, 110, 152, 155, 157, 159, 167, 172, 198, 307, 308	descriptionaccess	331, 333
delim_n	35	descriptionplural	127, 328
delim_r	35	descriptionpluralaccess	331, 333
page_compositor	34	first	59, 78, 116, 123, 145, 231, 236, 327
special characters	108, 109, 152	firstaccess	331, 333
\makenoidxglossaries	60, 61, 162, 166	firstplural	59, 124, 145, 328
\MakeTextUppercase	4		

firstpluralaccess	331, 333	205, 210, 224, 226, 228, 230, 232, 233,
format	154	235, 237, 239, 240, 306, 326, 327, 357–361
long	97, 148, 328	\nohyperpage 202
longaccess	332, 333	\noindent 197, 280–283, 299–301
longplural	148, 328	\noist 311, 312
longpluralaccess	332, 333	\nopostdesc 27, 33, 74, 178, 313
name	58, 59, 62, 65, 75, 125, 142, 175, 327	\normalbaselineskip 272
nonumberlist	61	\nr 6, 21, 22, 61, 190
parent	61, 65	\ns@ACRfull 207
plural	59, 78, 123, 327	\ns@Acrfull 206
pluralaccess	331, 332	\ns@acrfull 206
prefix	248	\ns@ACRfullpl 208
prefixfirst	248	\ns@Acrfullpl 208
prefixfirstplural	249	\ns@acrfullpl 207
prefixplural	249	\ns@ACRlong 139
see	7, 60, 65, 162, 163	\ns@Acrlong 138
short	97, 148, 328	\ns@acrlong 138
shortaccess	331, 333	\ns@ACRlongpl 141
shortplural	148, 328	\ns@Acrlongpl 140
shortpluralaccess	331, 333	\ns@acrlongpl 140
sort	59, 146, 198	\ns@ACRshort 135
symbol	58, 59,	\ns@Acrshort 135
	128, 227, 229, 231, 236, 266, 287, 327–329	\ns@acrshort 134
symbolaccess	331, 333	\ns@ACRshortpl 137
symbolplural	128, 328	\ns@Acrshortpl 136, 137
symbolpluralaccess	331, 333	\ns@acrshortpl 136
text	59, 116, 122, 144, 227, 231, 327	\ns@newglossary 55
textaccess	330, 332	\null 108–114, 179
type	12, 60, 91, 146	\number 10, 78, 88, 170, 171, 197, 327
user1	129, 146, 328	\numberline 39
user2	130, 146	\numexpr 88
user3	131, 147	
user4	132, 147	
user5	132, 147	
user6	133, 147, 328	
\newglossarystyle	257, 260–272, 274–302, 304	
\newif	4, 14, 21, 24, 170	
\newlength	115, 262, 263, 273, 284, 291, 300	
\newrobustcmd		
	65, 66, 70, 71, 88–90, 103, 116–	
	141, 143–149, 151, 205–208, 249–253, 301	
\newterm	27	
\newtoks	109, 159, 209	
\newwrite	66, 153, 157, 159, 161	
\noalign	272	
\nobreak	261, 273, 314	
\noexpand	14, 29, 40, 41, 79, 80,	
	100, 101, 106–108, 114, 150, 160, 162,	
	170, 172, 175, 179, 182, 183, 195, 197,	

O

\O	19
\o	19
\OE	19
\oe	19
\openout	66, 153, 157, 159, 308, 311
\OR	239
\or	6, 22, 190, 297, 318
\org@glossaryentrynumbers	177, 192
\org@glossarytitle	177
\org@glspostdescription	33
\org@ifKV@glslink@hyper	105, 106
\orgAlpha	171
\orgalph	171
\orgarabic	171
\orgnumber	171
\orgRoman	171

O

\O	19
\o	19
\OE	19
\oe	19
\openout	66, 153, 157, 159, 308, 311
\OR	239
\or	6, 22, 190, 297, 318
\org@glossaryentrynumbers	177, 192
\org@glossarytitle	177
\org@glspostdescription	33
\org@ifKV@glslink@hyper	105, 106
\orgAlph	171
\orgalph	171
\orgarabic	171
\orgnumber	171
\orgRoman	171

<code>\orgromannumeral</code>	171	161, 164, 166, 183–185, 189, 192, 200,
<code>\orgthe</code>	171	201, 211, 212, 228, 229, 234, 237, 312, 334
<code>\outputpenalty</code>	272	
P		
<code>\p@</code>	259, 279, 296	
<code>\p@glshyp@opt</code>	103	
package options:		
acronym	12, 13, 29, 176, 205	
true	13	
counter	15	
description	231, 232	
dua	229, 231, 232	
entrycounter	191, 193	
true	9	
footnote	116–121, 227, 229, 231, 233	
hyperfirst		
false	116–121	
indexonlyfirst	373	
makeindex	155, 247	
nogroupskip	269, 270	
nolist	240	
nolong	241, 263	
nomain	12	
nonumberlist	7	
nosuper	241	
notree	241	
numberline	5	
sanitize	19, 58, 142, 144	
sanitizesort	16	
savewrites	25, 370	
false	159	
true	161, 167	
section	5, 37	
sort		
def	9, 10	
standard	9	
use	9, 10	
style	6, 240, 241	
subentrycounter	191, 193	
toc	5	
true	5	
translate	21	
false	21	
translator	21	
xindy	24, 25, 155, 247	
<code>\PackageError</code> 25, 29, 40, 47, 50, 51, 55, 60,		
62, 63, 69–74, 76–78, 86, 101, 142, 159,		
161, 164, 166, 183–185, 189, 192, 200,		
201, 211, 212, 228, 229, 234, 237, 312, 334		
<code>\PackageInfo</code>	159	
<code>\PackageWarning</code>	15, 326	
<code>\PackageWarningNoLine</code>	16, 32, 364, 365	
<code>\pagegoal</code>	272	
<code>\pagelistname</code>	32,	
266–268, 270, 276, 278, 287–290, 294–296		
<code>\par</code>	33, 197, 198, 259, 261, 279–284,	
296, 297, 299–302, 304, 305, 314, 319–321		
<code>\parindent</code> 279–284, 297–300, 302–305, 319–321		
<code>\parskip</code>	279–282, 297, 298, 300	
<code>\PassOptionsToPackage</code>	248, 326	
<code>\penalty</code>	272	
<code>\phantomsection</code>	38	
polyglossia package	21, 31	
<code>\printglossaries</code>	162	
<code>\printglossary</code> ..	13, 16, 26, 27, 162, 176, 192	
<code>\printglossary options</code>		
entrycounter	190	
nogroupskip	190	
nonumberlist	191	
nopostdot	190	
numberedsection	190	
style	189	
subentrycounter	191	
title	189	
toctitle	189	
type	12, 175, 189	
<code>\printindex</code>	27	
<code>\printnoidxglossaries</code>	164	
<code>\printnoidxglossary</code>		
163, 164, 166, 176, 183, 184, 192		
<code>\printnoidxglossary options</code>		
sort	192	
<code>\printnumbers</code>	26	
<code>\printsymbols</code>	26	
<code>\ProcessOptions</code>	248, 326	
<code>\ProcessOptionsX</code>	27	
<code>\protect</code>	39, 99, 100, 212–214, 219,	
220, 226, 230, 232, 343, 347, 348, 353, 354		
<code>\protected@edef</code>	6, 41, 43, 46,	
48, 77, 80, 81, 92, 94, 100, 106, 107, 149,		
168, 171, 190, 195, 197, 200, 201, 210,		
234, 239, 249, 255, 307, 308, 326, 327, 332		
<code>\protected@write</code>	54, 56,	
153–155, 161, 163, 166, 169, 175, 255, 308		
<code>\protected@xdef</code>		
10, 11, 14, 18, 64, 82, 171, 329, 330		

<code>\providecommand</code>	13, 29, 30, 37, 54, 88, 116, 155, 161, 163, 179, 195, 197, 259, 279, 296
<code>\ProvidesFile</code>	30
<code>\ProvidesPackage</code>	4, 248, 255, 257, 259, 263, 269, 273, 278, 284, 290, 296, 306, 312, 326, 364
R	
<code>\r</code>	18
<code>\raggedright</code>	271, 272, 274–278, 291–296
<code>\raisebox</code>	115
<code>\ref</code>	194
<code>\refstepcounter</code>	190, 191, 193, 194
<code>\relax</code>	6, 8, 11–13, 21, 22, 28, 42, 54, 59, 61, 64, 77, 79, 87, 88, 103, 106–114, 143, 156–159, 161–165, 169, 171, 172, 174, 177, 178, 186, 187, 189, 190, 199–201, 241, 256, 259, 272, 279, 283, 284, 296, 297, 299–305, 307, 310– 312, 318–321, 329, 344, 345, 357–359, 361
<code>\renewacronymstyle</code>	347–351, 353, 355, 356
<code>\renewcommand</code>	4– 9, 12, 13, 15, 16, 20, 22, 23, 25, 28, 31– 35, 47, 58, 74, 86–88, 149, 151, 153, 161–165, 168, 178, 180, 190, 191, 209, 210, 212–222, 225, 227, 229, 231, 233, 234, 236, 238, 240, 257, 258, 260–267, 269, 270, 272, 274–286, 288, 291–295, 297–304, 306, 307, 312–320, 322–325, 329, 334, 338, 341, 343, 346–350, 352–360
<code>\renewenvironment</code>	195, 257, 260, 263–269, 271, 272, 274–298, 300, 302
<code>\RequireGlossariesLang</code>	32, 364, 365
<code>\RequirePackage</code>	4, 7, 8, 21, 22, 27, 31, 240, 247, 248, 263, 269, 273, 278, 284, 291, 327, 364
<code>\restorecounters@</code>	106
<code>\romannumeral</code>	170, 171, 302, 303, 320
S	
<code>\s@glshyp@opt</code>	103
<code>\s@newglossary</code>	55
<code>\savecounters@</code>	106
<code>\seename</code>	174
<code>\SetAcronymStyle</code>	23
<code>\setbool</code>	20
<code>\setbox</code>	272, 273
<code>\setcounter</code>	190, 191, 193
<code>\SetCustomDisplayStyle</code>	240
<code>\SetDefaultAcronymDisplayStyle</code>	225
<code>\SetDefaultAcronymStyle</code>	238
<code>\SetDescriptionAcronymDisplayStyle</code>	231
<code>\SetDescriptionAcronymStyle</code>	238
<code>\SetDescriptionDUAAcronymDisplayStyle</code>	229
<code>\SetDescriptionDUAAcronymStyle</code>	238
<code>\SetDescriptionFootnoteAcronymDisplayStyle</code>	227
<code>\SetDescriptionFootnoteAcronymStyle</code>	238
<code>\SetDUADisplayStyle</code>	238
<code>\SetDUASStyle</code>	239
<code>\setentrycounter</code>	41, 155, 188, 306
<code>\SetFootnoteAcronymDisplayStyle</code>	233
<code>\SetFootnoteAcronymStyle</code>	239
<code>\SetGenericNewAcronym</code>	211
<code>\setglossarystyle</code>	177, 200, 241, 260–272, 274– 283, 285–290, 292–296, 298, 299, 301, 304
<code>\setglossentrycompatibility</code>	189, 200, 201
<code>\setkeys</code>	20, 25, 28, 37, 76, 105, 151, 177, 209, 225, 227, 229, 231, 233, 236, 238, 240
<code>\setlength</code>	262, 263, 273, 279– 282, 284, 291, 297, 298, 300, 303, 320, 321
<code>\SetSmallAcronymDisplayStyle</code>	236
<code>\SetSmallAcronymStyle</code>	239
<code>\settoheight</code>	115
<code>\settowidth</code>	213, 301–303, 320
<code>\sfcode</code>	8
<code>\show</code>	241–246, 362, 363
<code>\SmallNewAcronymDef</code>	236
<code>\space</code>	25, 29, 40, 41, 45, 47, 48, 60–62, 86, 89, 90, 99–102, 149, 154–157, 160–162, 164, 166, 175, 177, 180, 190, 191, 194, 200, 206, 212–222, 230, 234, 235, 257, 259– 261, 263, 264, 274, 285, 291, 297, 299, 300, 302–304, 306, 307, 309–311, 313– 316, 318–322, 324, 343, 347–350, 352–357
<code>\spacefactor</code>	8
<code>\SS</code>	19
<code>\ss</code>	19
<code>\string</code>	16, 25, 29, 39–41, 43–48, 54, 56, 60–62, 66, 67, 70, 81, 82, 86, 88–90, 100–102, 107, 108, 110, 111, 113, 114, 149, 152–159, 161–164, 166, 172–174, 177, 179, 180, 183, 184, 192, 197, 198, 200, 255, 306–312
<code>\strut</code>	198, 260–262, 264, 265, 267, 274, 275, 277, 285, 286, 288, 291, 293, 295, 300, 313–317, 319, 322–325

<code>\subglossentry</code>	81, 178, 187, 197, 198, 258, 260–262, 264–266, 274, 275, 277, 285, 286, 288, 291, 293, 295, 297, 299, 300, 303
<code>\subitem</code>	297, 318
<code>\subsubitem</code>	297, 318
<code>supertabular</code> package	7, 241, 284, 291
<code>\symbolname</code>	32, 267, 268, 270, 277, 278, 288–290, 295, 296
T	
<code>\t</code>	18
<code>\tablehead</code>	284–296
<code>\tabletail</code>	284–296
<code>\tabularnewline</code>	263– 270, 274–278, 285–296, 316, 317, 322, 323
<code>\texorpdfstring</code>	146
<code>\textbar</code>	257
<code>\textbf</code>	197, 204, 296, 318–321
<code>textcase</code> package	4
<code>\textit</code>	204
<code>\textmd</code>	204
<code>\textrm</code>	203
<code>\textsc</code>	204, 214, 215, 221, 227, 231, 234, 236, 355
<code>\textsf</code>	203
<code>\textsl</code>	204
<code>\textsmaller</code>	214, 215, 221, 227, 231, 234, 236, 355
<code>\texttt</code>	204
<code>\textulc</code>	205
<code>\textup</code>	204, 205
<code>\th</code>	19
<code>\the</code>	29, 31, 41, 46, 48, 55, 109–114, 153, 157, 167, 168, 171, 175, 186, 196, 197, 203, 210, 212, 213, 218, 219, 224, 226, 228, 230, 232, 233, 235, 237, 239, 240, 306, 308, 311, 326, 327, 347, 348, 352, 353, 356–361
<code>\the@numberlist</code>	149, 150
<code>\theglossary</code>	5
<code>\theglossaryentry</code>	190, 193, 194
<code>\theglossarysubentry</code>	191, 193, 194
<code>\theglentrycounter</code>	106, 107, 171, 307, 308
<code>\theH</code>	173
<code>\theHglossaryentry</code>	190, 193
<code>\theHglossarysubentry</code>	191, 193
<code>\theHglentrycounter</code>	106, 107, 171
<code>\thesection</code>	28
<code>\this@dialect</code>	32, 364, 365
<code>\toks@</code>	29, 31, 41, 46, 48, 55, 109–114, 175, 195–197, 203, 306, 326, 327
<code>\toprule</code>	269, 270
<code>tracklang</code> package	31, 364
<code>\trans@languages</code>	31
<code>\translate</code>	31, 32
<code>\translatelet</code>	12, 13, 26, 27
<code>translator</code> package	12, 13, 21, 26, 27, 31, 32, 176
U	
<code>\u</code>	18
<code>\uccode</code>	186
<code>\undef</code>	176
<code>\unskip</code>	74, 262, 314
<code>\unvbox</code>	272, 273
<code>\usedictionary</code>	31
<code>\usepackage</code>	183, 184
V	
<code>\v</code>	18
<code>\val</code>	6, 21, 61, 190
<code>\vbox</code>	272, 273
<code>\vsize</code>	273
<code>\vskip</code>	259, 272, 279, 296
<code>\vss</code>	272, 273
W	
<code>\warn@nomakeglossaries</code>	162–164
<code>\warn@noprintglossary</code>	162–164, 178
<code>\write</code>	66, 67, 88, 153–158, 160, 163, 167, 179, 308–312
<code>\writeist</code>	159, 312
X	
<code>\x</code>	203
<code>\xatlevel@</code>	106
<code>\xcapitalisewords</code>	146
<code>\xdef</code>	71, 77–80, 178, 256
<code>\xglaccsupp</code>	332
<code>\xifinlistcs</code>	180, 181, 184
<code>xindy</code>	366
<code>xindy</code>	9, 24, 25, 33, 34, 39, 42, 44, 46–48, 82, 113, 114, 152, 153, 155, 167, 172, 179, 198, 247, 307
<code>\xmakefirstuc</code>	92, 94, 100, 142, 143, 249
<code>\xspace</code>	205
<code>xspace</code> package	4, 204
Y	
<code>\year</code>	153, 157, 308, 311
Z	
<code>\z@</code>	272, 273