

Documented Code For glossaries v4.21

Nicola L.C. Talbot

Dickimaw Books

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

2016-01-24

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 v4.21: L^AT_EX2e Package to Assist Generating Glossaries”.

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

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

INSTALL Installation instructions.

CHANGES Change log.

README Package summary.

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

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

Contents

1 Main Package Code	4
1.1 Package Definition	4
1.2 Package Options	5
1.3 Predefined Text	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	208
1.19 Predefined Glossary Styles	240
1.20 Debugging Commands	240
1.21 Compatibility with version 2.07 and below	246
2 Prefix Support (glossaries-prefix Code)	247
3 Glossary Styles	254
3.1 Glossary hyper-navigation definitions (glossary-hypernav package)	254
3.2 In-line Style (glossary-inline.sty)	256
3.3 List Style (glossary-list.sty)	258
3.4 Glossary Styles using longtable (the glossary-long package)	261
3.5 Glossary Styles using longtable and booktabs (the glossary-longbooktabs) package	268
3.6 Glossary Styles using longtable (the glossary-longragged package)	272
3.7 Glossary Styles using multicol (glossary-mcols.sty)	277
3.8 Glossary Styles using supertabular environment (glossary-super package)	281
3.9 Glossary Styles using supertabular environment (glossary-superragged package)	287
3.10 Tree Styles (glossary-tree.sty)	293

4 Backwards Compatibility	302
4.1 glossaries-compatible-207	302
4.2 glossaries-compatible-307	308
5 Accessibility Support (glossaries-accsupp Code)	322
5.1 Defining Replacement Text	323
5.2 Accessing Replacement Text	326
5.3 Displaying the Glossary	342
5.4 Acronyms	343
5.5 Debugging Commands	358
6 Multi-Lingual Support	360
6.1 Polyglossia Captions	360
Glossary	362
Change History	363
Index	384

1 Main Package Code

1.1 Package Definition

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

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries}[2016/01/24 v4.21 (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 redefined \glossarysection.

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

Determine whether or not to use numbered sections.

`glossarysecstar`

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

`glossaryseclabel`

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

`\glsautoprefix`

Prefix to add before label if automatically generated:

```
42 \newcommand*\glsautoprefix}{}
```

`numberedsection`

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

The default glossary style is stored in `\@glossary@default@style`. This is initialised to list. (The list style is defined in the accompanying package described in [section 1.19](#).)

`y@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   \gls@savenumberlistfalse
```

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

```

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

```

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

```

`entrycounterwithin` 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 \glsesubentrycounterfalse

```

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

```

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

```

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

```

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

```

`gsprestandardsort`

```
\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

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

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

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

Internal command requires label and list to be expanded:

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

Not found

```
268   #4%
269   \else
```

Found

```
270   #3%
271   \fi
272 }
```

lsglsacronymlist Convenient boolean.

```
273 \newif\if@lsglsacronymlist
```

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

```
274 \newcommand*{\gls@cklsglsacronymlist}[1]{%
275   \glsIfListOfAcronyms{#1}%
276   {\@lsglsacronymlisttrue}{\@lsglsacronymlistfalse}%
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 }
```

`glossariesWarning` 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}
```

descplural@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 defined by loading the appropriate package.

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

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

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

`shortcuts` Define acronym shortcuts.
590 `\define@boolkey{glossaries.sty}[glsacr]{shortcuts}[true]{}`

`\glsorder` Stores the glossary ordering. This may either be “word” or “letter”. This passes the relevant information to `makeglossaries`. The default is word ordering.
591 `\newcommand*{\glsorder}{word}`

`\@glsorder` The ordering information is written to the auxiliary file for `makeglossaries`, so ignore the auxiliary information.
592 `\newcommand*{\@glsorder}[1]{}`

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

`\ifglsxindy` Provide boolean to determine whether `xindy` or `makeindex` will be used to sort the glossaries.
595 `\newif\ifglsxindy`
The default is `makeindex`:
596 `\glsxindyfalse`

`makeindex` Define package option to specify that `makeindex` will be used to sort the glossaries:
597 `\@gls@declareoption{makeindex}{\glsxindyfalse}`
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{\languagename}%`
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=glSnumbers=false` that can be passed via the document class options.

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

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

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

`@gls@doautomake`

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

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

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

Set default:

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

compatible-3.07

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

compatible-2.07

```
646 \define@boolkey{glossaries.sty}[gls]{compatible-2.07}[true]{}
  Also set 3.07 compatibility if this option is set.
647 \ifbool{glscompatible-2.07}%
648   {%
649     \booltrue{glscompatible-3.07}%
650   }%
651   {}%
652 }
653 \boolfalse{glscompatible-2.07}
```

symbols Create a “symbols” glossary type

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

Default is not to define the symbols glossary:

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

@gls@symbolsdef

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

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

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

numbers Create a “symbols” glossary type

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

Default is not to define the numbers glossary:

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

@gls@numbersdef

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

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

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

index Create an “index” glossary type

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

Default is not to define index glossary:

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

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

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

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

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

Now process options passed to the package:

```
698 \ProcessOptionsX
```

Load backward compatibility stuff:

```
699 \RequirePackage{glossaries-compatible-307}
```

setupglossaries Provide way to set options after package has been loaded. However, some options must be set before \ProcessOptionsX, so they have to be disabled:

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

Now define `\setupglossaries`:

```
703 \newcommand*{\setupglossaries}[1]{%
704   \renewcommand*{\@gls@setacrstyle}{}%
705   \ifglsacrshortcuts
706     \def\@gls@setupshortcuts{\glsacrshortcutstrue}%
707   \else
708     \def\@gls@setupshortcuts{%
709       \ifglsacrshortcuts
710         \DefineAcronymSynonyms
711       \fi
712     }%
713   \fi
714   \glsacrshortcutsfalse
715   \let\@gls@do@numbersdef\relax
716   \let\@gls@do@symbolssdef\relax
717   \let\@gls@do@indexdef\relax
718   \let\@gls@do@acronymsdef\relax
719   \setkeys{glossaries.sty}{#1}%
720   \@gls@setacrstyle
721   \@gls@setupshortcuts
722   \@gls@do@acronymsdef
723   \@gls@do@numbersdef
724   \@gls@do@symbolssdef
725   \@gls@do@indexdef
726 }
```

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

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

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

`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 @gls@#1@title\endcsname}}
```

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

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

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

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

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

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

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*\gl@istfilebase}{#1}%
  Just in case \istfilename has been modified.
860 \ifglsxindy
861 \def\istfilename{\gl@istfilebase.xdy}
862 \else
863 \def\istfilename{\gl@istfilebase.ist}
864 \fi
865 }
```

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

```

866 \@onlypremakeg\setStyleFile
```

The name of the `makeindex` or `xindy` style file is given by `\istfilename`. This file is created by `\writeist` (which is used by `\makeglossaries`) so 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{\gl@istfilebase.xdy}
869 \else
870 \def\istfilename{\gl@istfilebase.ist}
871 \fi
```

`gl@istfilebase`

```

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

The `makeglossaries` Perl script picks up this name from the auxiliary file. If the name ends with `.xdy` it calls `xindy` otherwise it calls `makeindex`. Since its not required by \LaTeX , `\@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 `\glsnumberformat`, otherwise it will simply display its argument "as is".

```
895 \ifcsundef{hyperlink}%  
896 {%  
897   \newcommand*{\glsnumberformat}[1]{#1}%  
898 }%  
899 {%  
900   \newcommand*{\glsnumberformat}[1]{\glsnumberformat{#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 `\glossary` environment. It is designed to allow the user to add information pertaining to the glossary (e.g. "page numbers in italic indicate the primary definition") therefore `\glossarypreamble` shouldn't be affected by the glossary style. (So if you define your own glossary style, don't have it change `\glossarypreamble`.) The preamble is empty by default. If you have multiple glossaries, and you want a different preamble for each glossary, you will need to use `\printglossary` for each glossary type, instead of `\printglossaries`, and redefine `\glossarypreamble` before each `\printglossary`.

`\glossarypreamble`

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

glossary preamble

```
\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 `theglossary` 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{}}
```

glossary postamble

```
916 \newcommand*{\glossarypostamble}{}
```

glossary section

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   \gls@glossarymark{\glossarytoctitle}%
927 }
```

gls glossary mark

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

```
928 \ifcsundef{glossarymark}%
929   {%
930     \newcommand{\gls@glossarymark}[1]{\glossarymark{#1}}
931   }%
932   {}%
```

```

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

```

`\glossarymark` Provided for backward compatibility:

```

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

```

The required sectional unit is given by `\@glossarysec` which was defined by the section package option. The starred form of the command is chosen. If you don't want any sectional command, you will need to redefine `\glossarysection`. The sectional unit can be changed, if different sectional units are required.

`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*\@pglossarysection [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@docclearpage` The `\gls@docclearpage` 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@docclearpage{%
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.

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

```

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

```

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

```

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

```

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

```

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

```

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

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

`\@gls@ifinlist`

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

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

```
1042 \ifglxindy
1043 \newcommand*\@xdycounters{\@glscounter}
1044 \newcommand*\GlsAddXdyCounters[1]{%
1045 \@for\@gls@ctr:=#1\do{%
```

Check if already in list before adding.

```
1046 \edef\@do@addcounter{%
1047 \noexpand\@gls@ifinlist{\@gls@ctr}{\@xdycounters}{}%
1048 {%
1049 \noexpand\edef\noexpand\@xdycounters{\@xdycounters,%
1050 \noexpand\@gls@ctr}%
1051 }%
1052 }%
1053 \@do@addcounter
1054 }
1055 }
```

Only has an effect before `\writeist`:

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

`\GlsAddXdyCounters` 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 \ifglxindy
```

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

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

Add to xindy attribute list

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

Add to xindy markup location.

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

```

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

```
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{hyperrm}
1118 \GlsAddXdyAttribute{hypersf}
1119 \GlsAddXdyAttribute{hypertt}
1120 \GlsAddXdyAttribute{hyperbf}
1121 \GlsAddXdyAttribute{hypermd}
1122 \GlsAddXdyAttribute{hyperit}
1123 \GlsAddXdyAttribute{hyperup}
1124 \GlsAddXdyAttribute{hypersl}
1125 \GlsAddXdyAttribute{hypersc}
1126 \GlsAddXdyAttribute{hyperemph}

1127 \GlsAddXdyAttribute{glsignore}
1128 }
1129 \else
1130 \let\@gls@addpredefinedattributes\relax
1131 \fi
```

`dyuseralphabets` List of additional alphabets

```
1132 \def\@xdyuseralphabets{}
```

`sAddXdyAlphabet` `\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

```

`\gls@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.

`\gls@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

```

`\gls@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   }%

```

```

ic-page-numbers  Arabic numbers (1, 2, ...).
1172  \expandafter\def\csname @gls@xdy@Lclass@arabic-page-numbers\endcsname{%
1173    \string"arabic-numbers\string"%
1174  }%

ha-page-numbers  Lower case alphabetical (a, b, ...).
1175  \expandafter\def\csname @gls@xdy@Lclass@alpha-page-numbers\endcsname{%
1176    \string"alpha\string"%
1177  }%

HA-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  }%

serlocationdefs  List of additional location definitions (separated by ^^J)
1191  \def\@xdyuserlocationdefs{}

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

      End of xindy-only block:
1193 \fi

sAddXdyLocation  \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][\@empty]{%
1196     \def\@gls@tmp{#1}%
1197     \ifx\@gls@tmp\@empty
1198       \edef\@xdyuserlocationdefs{%

```

```

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

```

Only has an effect before `\writeist`:

```

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

```

`\locationclassorder` Define location class order

```

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

```

Change the location order.

`\locationClassOrder`

```

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

```

```

1241 \glsnoxywarning\GlsSetXdyLocationClassOrder}
1242 \fi

```

`\@xdysortrules` Define sort rules

```

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

```

`\GlsAddSortRule` Add a sort rule

```

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

```

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

```

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

```

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

```

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

```

`\GlsSetXdyStyles` Reset the list of required styles

```

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

```

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

```

1273 \newcommand*\findrootlanguage{}

```

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

```
1274 \def\@xdylanguage#1#2{}
```

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

```
1275 \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{}
```

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

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

Suggested by egreg:

```
1292 \AtBeginDocument{%
1293   \ifx\@gls@codepage\@empty
1294     \@ifpackageloaded{fontspec}{\def\@gls@codepage{utf8}}{ }%
1295   \fi
1296 }
1297 \else
1298   \newcommand*\GlsSetXdyCodePage[1]{%
1299     \glsnoxywarning\GlsSetXdyCodePage}
1300 \fi
```

`\xdylettergroups` Store letter group definitions.

```
1301 \ifglxindy
1302   \ifglx@xindy@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

```

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

```

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

```

1.5 Loops and conditionals

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

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

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

```

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

```

`\forallacronyms`

```

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

```

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

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

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

```

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

```


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

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

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

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

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

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

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

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

Since the label is used to form the name of control sequences, by default UTF8 etc characters can't be used in the label. A possible workaround is to use `\scantokens`, but commands such as `\glsentrytext` will no longer be usable in sectioning, caption etc commands. If the user really wants to be able to construct a label with UTF8 characters, allow them the means to do so (but on their own head be it, if they then use entries in `\section` etc). This can be done via:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
1354 \newcommand{\glsdoifnoexists}[2]{%
1355   \ifglsentryexists{#1}{#2}{%
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   \ifglsentryexists{#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 }

```

glschaschildren `\ifglschaschildren{<label>}{<>true part>}{<>false part>}`

```

1381 \newcommand{\ifglschaschildren}[3]{%
1382   \glsdoifexists{#1}%
1383   {%
1384     \def\do@glshaschildren{#3}%
1385     \edef\@gls@thislabel{\glsdetoklabel{#1}}%
1386     \expandafter\forglsentries\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 }

```

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

```

1412 \newcommand*{\ifglsdescsuppressed}[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     \ifdequal\@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       \ifdequal\@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}{,}
```

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

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

Only need to do this once.

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

`\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}}
```

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

```
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 \LaTeX , `\@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\@gls@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}}
```

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

```
1697 \newcommand*\@glsnodesc{%  
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   \ifglsentryexists{#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\ifglsentryexists{#1}{}\glspercentchar^^J%
1838     \expandafter\@gobble\string\{\glspercentchar^^J%
1839     \string\gls@defglossaryentry{\glsdetoklabel{#1}}\glspercentchar^^J%
1840     \expandafter\@gobble\string\{\glspercentchar%
1841   }%
```

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 \gls@writedefhook
```

```

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 }

```

\glsaddkey	$\glsaddkey{\langle key \rangle}{\langle default\ value \rangle}{\langle no\ link\ cs \rangle}{\langle no\ link\ ucfirst\ cs \rangle}{\langle link\ cs \rangle}{\langle link\ ucfirst\ cs \rangle}{\langle link\ allcaps\ cs \rangle}$
------------	--

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 }

```

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

```

2053 \newcommand{\glsfielddef}[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 }
```

glswritedefhook

```
2119 \newcommand*{\glswritedefhook}{}%
```

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     \ifglsentryexists{\@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\@glsnoname
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\@glsnoname
2285   \@glsnoname
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\@glsnodesc
2300   \@glsnodesc
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\@gls@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}{}

```

`aryentryposthook` 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{\glsdetoklabel{#1}}%

```



```

2349 \edef\glo@type{\csname glo@\glo@thislabel @type\endcsname}%
2350 \def\glo@list{,}%
2351 \forlslentries[\glo@type]{\glo@label}%
2352   {%
2353     \ifdequal\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 }

```

`glossaryentryfield` 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

```

`glossarysubentryfield` 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 \ifglxindy

```

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     \@cgl's@{##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     \@cGls@{##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     \@cGlspl@{##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   \csdef{glo@glstoklabel{##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@glstoklabel{##1}@currcount}{%
2577     \number\numexpr\glsentrycurrcount{##1}+1}%
2578 }
```

ement@currcount

```
2579 \newcommand*{\@gls@local@increment@currcount}[1]{%
2580   \csedef{glo@glstoklabel{##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`s 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}s}{\@gls@hyp@opt\@cgl}s}
```



```

\@cgl's  Defined the un-starred form. Need to determine if there is a final optional argument
2595 \newcommand*{\@cgl's}[2][\%
2596 \new@ifnextchar[{\@cgl's@{#1}{#2}}{\@cgl's@{#1}{#2}[]}]%
2597 }

\@cgl's@  Read in the final optional argument. This defaults to same behaviour as \gls but issues a
warning.
2598 \def\@cgl's@#1#2[#3]{%
2599 \GlossariesWarning{\string\cgl's\space is defaulting to
2600 \string\gls\space since you haven't enabled entry counting}%
2601 \@gls@{#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*{\cgl'sformat}[2]{%
2604 \ifgls'haslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}#2%
2605 }

\cGls  Define command that works like \Gls but behaves differently if the entry count function is
enabled. (If not enabled, it behaves the same as \Gls but issues a warning.)
2606 \newrobustcmd*{\cGls}{\@gls@hyp@opt\@cGls}

\@cGls  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 }

\@cGls@  Read in the final optional argument. This defaults to same behaviour as \Gls 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 }

\cGlsformat  Format used by \cGls if entry only used once on previous run. The first argument is the label,
the second argument is the insert text.
2615 \newcommand*{\cGlsformat}[2]{%
2616 \ifgls'haslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}#2%
2617 }

\cgl'spl  Define command that works like \glspl but behaves differently if the entry count function
is enabled. (If not enabled, it behaves the same as \glspl but issues a warning.)
2618 \newrobustcmd*{\cgl'spl}{\@gls@hyp@opt\@cgl'spl}

```

`\@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}{\glentrylongpl{#1}}{\glentryfirstplural{#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}{\@gls@hyp@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}{\Glentrylongpl{#1}}{\Glentryfirstplural{#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     \glscapspace
2658     {%
```

Don't adjust case

```
2659     \ifglsused\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     \ifglsused\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 `\defglsentryfmt`, which avoids the issues caused by fragile commands.)

```
2674     \ifbool{glscompatible-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{glscompatible-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      \ifglsused\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      \ifglsused\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{\glsentryfirst{\glslabel}}}%
2775     {\glsentrydesc{\glslabel}}}%
2776     {\glsentrysymbol{\glslabel}}{\glsinsert}}}%
2777   }%
2778 }%
2779 }%
2780 }%
2781 {%

```

Custom text provided in `\glsdisp`

```

2782   \ifglsused{\glslabel}%
2783   {%

```

Subsequent use

```

2784     #2{\glscustomtext}%
2785     {\glsentrydesc{\glslabel}}}%
2786     {\glsentrysymbol{\glslabel}}{}%
2787   }%
2788   {%

```

First use

```

2789     #1{\glscustomtext}%
2790     {\glsentrydesc{\glslabel}}}%
2791     {\glsentrysymbol{\glslabel}}{}%
2792   }%
2793 }%
2794 }

```

`\glsentryfmt` 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*{\glsentryfmt}{%
2796   \ifdefempty\glscustomtext
2797   {%
2798     \glsifplural
2799     {%

```

Plural form

```

2800     \glsupcase
2801     {%

```

Don't adjust case

```

2802     \ifglsused\glslabel
2803     {%

```

Subsequent use

```

2804     \glsentryplural{\glslabel}\glsinsert
2805     }%
2806     {%

```

First use

```

2807     \glsentryfirstplural{\glslabel}\glsinsert
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     \ifglused\glslabel
2844     {%
```

Subsequent use

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

First use

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

Make all upper case

```
2852     \ifglused\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     \ifglused\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      \gls caps case
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      \gls caps case
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{<label>}{<insert>}`

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

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

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

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 }
```

nplacrfullformat `\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 }
```

nplacrfullformat `\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 L^AT_EX 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 \ifglsused{\glslabel}%
3059 {%
3060 }%
3061 {%
3062 \gls@checkisacronymlist\glstype
3063 \ifglshyperfirst
3064 \ifglsisacronymlist
3065 \ifglsacrfootnote
3066 \KV@glslink@hyperfalse
3067 \fi
3068 \fi
3069 \else
3070 \KV@glslink@hyperfalse
3071 \fi
3072 }%
3073 \glslinkcheckfirsthyperhook
3074 }
```

Allow user to hook into this

checkfirsthyperhook 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}

```

nohyperinlist Disable hyperlink if in the “nohyper” list.


```

3078 \newcommand*\do@gl:disablehyperinlist}{%
3079   \expandafter\DTLifinlist\expandafter{\glstype}{\@gl:nohyperlist}%
3080   {\KV@gl:link@hyperfalse}}%
3081 }

```

lt@gl:link@opts Hook to set default options for \@gl:link.

```

3082 \newcommand*\@gl:setdefault@gl:link@opts{}

```

\@gl:link

```

3083 \def\@gl:link[#1]#2#3{%

```

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

```

3084   \leavevmode
3085   \edef\gl:label{\gl:detoklabel{#2}}%

```

Save options in \@gl:link@opts and label in \@gl:link@label

```

3086   \def\@gl:link@opts{#1}%
3087   \let\@gl:link@label\gl:label
3088   \def\@gl:numberformat{gl:numberformat}%
3089   \edef\@gl:counter{\csname glo@\gl:label @counter\endcsname}%

```

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

```

3090   \edef\glstype{\csname glo@\gl:label @type\endcsname}%

```

Save original setting

```

3091   \let\org@ifKV@gl:link@hyper\ifKV@gl:link@hyper

```

Set defaults:

```

3092   \@gl:setdefault@gl:link@opts

```

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

```

3093   \do@gl:disablehyperinlist

```

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

```

3094   \do@gl:link@checkfirsthyper
3095   \setkeys{gl:link}{#1}%

```

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

```

3096   \gl:linkpostsetkeys

```

Store the entry’s counter in \thegl:entrycounter

```

3097   \@gl:saveentrycounter

```

Define sort key if necessary:

```

3098   \@gl:setsort{\gl:label}%

```

(De-tok’ing done by \@do@wrglossary)

```

3099   \@do@wrglossary{#2}%
3100   \ifKV@gl:link@hyper
3101     \@gl:link{\gl:linkprefix\gl:label}{\gl:textformat{#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 `\@currentenv` 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\theHglsentrycounter{\@gls@Hcounter}%

```

```

3134     \restorecounters@

```

```

3135     \fi

```

```

3136     }%

```

```

3137 }%

```

```

3138 {%

```

Not using equation counter so no special measures:

```
3139 \edef\theglentrycounter{\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\theHglentrycounter{\theglentrycounter}%
3146 }%
3147 {%
3148 \protected@edef\theHglentrycounter{\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   \ifglxindy
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@checkescbar Similarly for \|:

```

3305 \def\@gls@checkescbar#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@checkescbar{\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@checkescbar{\@gls@checkescbar#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@checkescbar{\@gls@checkescbar#2\|\|\null}%
3323     \else
3324       \def\@gls@checkescbar{\@gls@checkescbar#2\|#3\null}%
3325     \fi
3326     \fi
3327 \@@gls@checkescbar
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

```

glsdohypertarget

```
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}}
```

glsdonohyperlink

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

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

```
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 `\@glo@text`) Note that `\@gls@link` sets `\glstype`.

```
3526 \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.

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

Indicate that this entry has now been used

```
3528 \ifKV@glslink@local
3529 \glslocalunset{#2}%
3530 \else
3531 \glsunset{#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 \glsdoifexists{#2}%
3542 {%
3543 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3544 \let\glsifplural\@secondoftwo
3545 \let\gls caps case\@secondofthree
3546 \let\gls custom text\@empty
3547 \def\glsinsert{#3}%
```

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

```
3548 \def\@glo@text{\csname gls@\glstype @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.

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

Indicate that this entry has now been used

```
3550 \ifKV@glslink@local
3551 \glslocalunset{#2}%
3552 \else
3553 \glsunset{#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] [] {%
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\glsapscase\@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] [] {%
3582   \new@ifnextchar [\@glspl@{#1}{#2}]{\@glspl@{#1}{#2} []}%
3583 }
```

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

```
3584 \def\@glspl@#1#2[#3]{%
3585   \glsdoifexists{#2}%
3586   {%
3587     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3588     \let\glsifplural\@firstoftwo
3589     \let\glsapscase\@firstofthree
3590     \let\glscustomtext\@empty
3591     \def\glsinsert{#3}%
```

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

```
3592   \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.

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

Indicate that this entry has now been used

```
3594   \ifKV@gls@link@local
3595     \glslocalunset{#2}%
3596   \else
3597     \glsunset{#2}%
3598   \fi
3599 }%
```

```
3600 \glspostlinkhook
3601 }
```

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

`\Glspl`

```
3602 \newrobustcmd*{\Glspl}{\@gls@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   \glsdoifexists{#2}%
3608   {%
3609     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3610     \let\glsifplural\@firstoftwo
3611     \let\glsapscase\@secondofthree
3612     \let\glscustomtext\@empty
3613     \def\glsinsert{#3}%
```

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

```

3614 \def\@glo@text{\csname gls@\gls@type @entryfmt\endcsname}%
Call \@gls@link. If footnote package option has been used and the glossary type is
\acronymtype, suppress hyperlink for first use. Likewise if the hyperfirst=false package op-
tion is used.
3615 \@gls@link[#1]{#2}{\@glo@text}%
Indicate that this entry has now been used
3616 \ifKV@gls@link@local
3617 \glslocalunset{#2}%
3618 \else
3619 \glsunset{#2}%
3620 \fi
3621 }%
3622 \gls@postlinkhook
3623 }

```

`\GLSp1` behaves like `\glspl` 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\gls@capscase\@thirdofthree
3634 \let\gls@customtext\@empty
3635 \def\glsinsert{#3}%

```

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

```

3636 \def\@glo@text{\csname gls@\gls@type @entryfmt\endcsname}%
Call \@gls@link. If footnote package option has been used and the glossary type is
\acronymtype, suppress hyperlink for first use. Likewise if the hyperfirst=false package op-
tion is used.
3637 \@gls@link[#1]{#2}{\@glo@text}%

```


Indicate that this entry has now been used

```
3638 \ifKV@glslink@local
3639 \glslocalunset{#2}%
3640 \else
3641 \glsunset{#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 \glsdoifexists{#2}{%

3649 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3650 \let\glsifplural\@secondoftwo
3651 \let\glsapscase\@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 \glslocalunset{#2}%
3658 \else
3659 \glsunset{#2}%
3660 \fi
3661 }%

3662 \glspostlinkhook
3663 }
```

checkfirsthyper Instead of just setting `\do@gls@link@checkfirsthyper` to `\relax` in `\@gls@field@link`, set it to `\@gls@link@nocheckfirsthyper` in case some other action needs to take place.

```
3664 \newcommand*{\@gls@link@nocheckfirsthyper}{}
```

@gls@field@link

```
3665 \newcommand{\@gls@field@link}[3]{%
3666   \glsdoifexists{#2}%
3667   {%
3668     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
3669     \@gls@link[#1]{#2}{#3}%
3670   }%

3671   \glspostlinkhook
3672 }
```

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

`\gls@text`

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

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

```
3674 \newcommand*{\@gls@text}[2] [] {%
3675   \new@ifnextchar[{\@gls@text@{#1}{#2}}{\@gls@text@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3676 \def\@gls@text@#1#2[#3] {%
3677   \@gls@field@link{#1}{#2}{\gls@entrytext{#2}#3}%
3678 }
```

`\GLStext` behaves like `\gls@text` except the text is converted to uppercase.

`\GLStext`

```
3679 \newrobustcmd*{\GLStext}{\@gls@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   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\gls@entrytext{#2}#3}}%
3684 }
```

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

`\Glstext`

```
3685 \newrobustcmd*{\Glstext}{\@gls@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}{\Glsentrytext{#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}{\glsentryfirst{#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}{\Glsentryfirst{#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{\glsentryfirst{#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] [] {%
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] [] {%
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] [] {%
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}{\glstentrysymbol{#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}{\glstentrysymbol{#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{\glstentrysymbol{#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}{\glstentrysymbolplural{#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}{\glstentrysymbolplural{#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{\glstentrysymbolplural{#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}{\glstentryuseri{#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][\@Glsuseri]
```

```
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][\@GLSuseri]
```

```
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][\@glsuserii]
```

```
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][\@Glsuserii]
```

```
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][\@GLSuserii]
```

```
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][\@glsuseriii]
```

```
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][\@Glsuseriii]
```

```
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][\@GLSuseriii]
```

```
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*{\@GLSuservi}[2] [] {%
3921   \new@ifnextchar [{\@GLSuservi@{#1}{#2}}{\@GLSuservi@{#1}{#2} []}}
```

Read in the final optional argument:

```
3922 \def\@GLSuservi@#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}{\curname gls@\glstype @entryfmt\endcurname}%
```

```
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\glscapscase\@secondofthree
3972 \let\glsinsert\@empty
3973 \def\glscustomtext{%
3974 \acronymfont{\Glsentryshort{#2}}#3%
3975 }%
```

Call \@gls@link Note that \@gls@link sets \glsstyle.

```
3976 \@gls@link[#1]{#2}{\cename gls@\glsstyle @entryfmt\endcename}%
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\glsifcaps\@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\glsifcaps\@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@gls@link@checkfirsthyper\@gls@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@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
4034   }%
4035   \glspostlinkhook
4036 }
```

Call `\@gls@link` Note that `\@gls@link` sets `\glsstyle`.

```
4033   \@gls@link[#1]{#2}{\csname gls@\glsstyle @entryfmt\endcsname}%
4034   }%
4035   \glspostlinkhook
4036 }
```

`\ACRshortpl`

```
4037 \newrobustcmd*{\ACRshortpl}{\@gls@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@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
4045     \def\glslabel{#2}%
4046     \let\glsifplural\@firstoftwo
4047     \let\glsapspace\@thirdofthree
4048     \let\glsinsert\@empty
4049     \def\glscustomtext{%
4050       \mfirstucMakeUppercase{\acronymfont{\Glsentryshortpl{#2}}#3}%
4051     }%
4052     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
4053   }%
4054   \glspostlinkhook
4055 }
```

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\glscapscase\@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@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
4083 \def\glslabel{#2}%
4084 \let\glsifplural\@secondoftwo
4085 \let\glsapscase\@secondofthree
4086 \let\glsinsert\@empty

```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```

4087 \def\glscustomtext{%
4088 \Glsentrylong{#2}#3%
4089 }%

```

Call \@gls@link. Note that \@gls@link sets \glsstyle.

```

4090 \@gls@link[#1]{#2}{\csname gls@\glsstyle @entryfmt\endcsname}%
4091 }%
4092 \glspostlinkhook
4093 }

```

\ACRlong

```

4094 \newrobustcmd*{\ACRlong}{\@gls@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 \glsdoifexists{#2}%
4100 {%

```

```

4101 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
4102 \def\glslabel{#2}%
4103 \let\glsifplural\@secondoftwo
4104 \let\glsapscase\@thirdofthree
4105 \let\glsinsert\@empty

```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```

4106 \def\glscustomtext{%
4107 \mfirstucMakeUppercase{\glsentrylong{#2}#3}%
4108 }%

```

Call \@gls@link. Note that \@gls@link sets \glsstyle.

```

4109 \@gls@link[#1]{#2}{\csname gls@\glsstyle @entryfmt\endcsname}%
4110 }%
4111 \glspostlinkhook
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\glsifscaps\@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][\@ns@ACRlongpl]{%
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\glsifscaps\@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 assumption that the argument of `\acronymfont` is `\glsentryshort{<label>}`, as that's the behaviour of the predefined acronym styles.) This is scoped to localise the effect of the assignment.

```
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.

`\glsentrydesc`

```
4227 \newcommand*{\glsentrydesc}[1]{\@gls@entry@field{#1}{desc}}
```

`\Glsentrydesc`

```
4228 \newrobustcmd*{\Glsentrydesc}[1]{%
4229   \@Gls@entry@field{#1}{desc}%
4230 }
```

Plural form:

`entrydescplural`

```
4231 \newcommand*{\glsentrydescplural}[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:

`\glsentrytext`

```
4237 \newcommand*{\glsentrytext}[1]{\@gls@entry@field{#1}{text}}
```

`\Glsentrytext`

```
4238 \newrobustcmd*{\Glsentrytext}[1]{%
4239   \@Gls@entry@field{#1}{text}%
4240 }
```

Get the plural form:

`\glsentryplural`

```
4241 \newcommand*{\glsentryplural}[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).

`entryfirstplural`

```
4265 \newcommand*{\glsentryfirstplural}[1]{%
4266   \@gls@entry@field{#1}{firstpl}%
4267 }
```

`entryfirstplural`

```
4268 \newrobustcmd*{\Glsentryfirstplural}[1]{%
4269   \@Gls@entry@field{#1}{firstpl}%
4270 }
```

Display the glossary type with which this entry is associated (as specified by the type key used when the entry was defined)

`\glsentrytype`

```
4271 \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`

```
4272 \newcommand*\glsentrysort}[1]{%
4273   \@gls@entry@field{#1}{sort}%
4274 }
```

`\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.

```
4275 \newcommand*\glsentryuseri}[1]{%
4276   \@gls@entry@field{#1}{useri}%
4277 }
```

`\Glsentryuseri`

```
4278 \newrobustcmd*\Glsentryuseri}[1]{%
4279   \@Gls@entry@field{#1}{useri}%
4280 }
```

`\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.

```
4281 \newcommand*\glsentryuserii}[1]{%
4282   \@gls@entry@field{#1}{userii}%
4283 }
```

`\Glsentryuserii`

```
4284 \newrobustcmd*\Glsentryuserii}[1]{%
4285   \@Gls@entry@field{#1}{userii}%
4286 }
```

`\glsentryuseriii` Get the third user key (as specified by the user3 when the entry was defined). The argument is the label associated with the entry.

```
4287 \newcommand*\glsentryuseriii}[1]{%
4288   \@gls@entry@field{#1}{useriii}%
4289 }
```

`\Glsentryuseriii`

```
4290 \newrobustcmd*\Glsentryuseriii}[1]{%
4291   \@Gls@entry@field{#1}{useriii}%
4292 }
```

`\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.

```
4293 \newcommand*{\glentryuseriv}[1]{%
4294   \@gls@entry@field{#1}{useriv}%
4295 }
```

`\Glsentryuseriv`

```
4296 \newrobustcmd*{\Glsentryuseriv}[1]{%
4297   \@Gls@entry@field{#1}{useriv}%
4298 }
```

`\glentryuseriv` Get the fifth user key (as specified by the user5 when the entry was defined). The argument is the label associated with the entry.

```
4299 \newcommand*{\glentryuseriv}[1]{%
4300   \@gls@entry@field{#1}{useriv}%
4301 }
```

`\Glsentryuseriv`

```
4302 \newrobustcmd*{\Glsentryuseriv}[1]{%
4303   \@Gls@entry@field{#1}{useriv}%
4304 }
```

`\glentryuseriv` Get the sixth user key (as specified by the user6 when the entry was defined). The argument is the label associated with the entry.

```
4305 \newcommand*{\glentryuseriv}[1]{%
4306   \@gls@entry@field{#1}{useriv}%
4307 }
```

`\Glsentryuseriv`

```
4308 \newrobustcmd*{\Glsentryuseriv}[1]{%
4309   \@Gls@entry@field{#1}{useriv}%
4310 }
```

`\glentryshort` Get the short key (as specified by the short the entry was defined). The argument is the label associated with the entry.

```
4311 \newcommand*{\glentryshort}[1]{\@gls@entry@field{#1}{short}}
```

`\Glsentryshort`

```
4312 \newrobustcmd*{\Glsentryshort}[1]{%
4313   \@Gls@entry@field{#1}{short}%
4314 }
```

`glsentryshortpl` Get the short plural key (as specified by the shortplural the entry was defined). The argument is the label associated with the entry.

```
4315 \newcommand*{\glsentryshortpl}[1]{\@gls@entry@field{#1}{shortpl}}
```

`Glsentryshortpl`

```
4316 \newrobustcmd*{\Glsentryshortpl}[1]{%
4317   \@Gls@entry@field{#1}{shortpl}}%
4318 }
```

`\glsentrylong` Get the long key (as specified by the long the entry was defined). The argument is the label associated with the entry.

```
4319 \newcommand*{\glsentrylong}[1]{\@Gls@entry@field{#1}{long}}
```

`\Glsentrylong`

```
4320 \newrobustcmd*{\Glsentrylong}[1]{%
4321   \@Gls@entry@field{#1}{long}}%
4322 }
```

`\glsentrylongpl` Get the long plural key (as specified by the longplural the entry was defined). The argument is the label associated with the entry.

```
4323 \newcommand*{\glsentrylongpl}[1]{\@Gls@entry@field{#1}{longpl}}
```

`\Glsentrylongpl`

```
4324 \newrobustcmd*{\Glsentrylongpl}[1]{%
4325   \@Gls@entry@field{#1}{longpl}}%
4326 }
```

Short cut macros to access full form:

`\glsentryfull`

```
4327 \newcommand*{\glsentryfull}[1]{%
4328   \acrfullformat{\glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%
4329 }
```

`\Glsentryfull`

```
4330 \newrobustcmd*{\Glsentryfull}[1]{%
4331   \acrfullformat{\Glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%
4332 }
```

`\glsentryfullpl`

```
4333 \newcommand*{\glsentryfullpl}[1]{%
4334   \acrfullformat{\glsentrylongpl{#1}}{\acronymfont{\glsentryshortpl{#1}}}%
4335 }
```

`\Glsentryfullpl`

```
4336 \newrobustcmd*{\Glsentryfullpl}[1]{%
4337   \acrfullformat{\Glsentrylongpl{#1}}{\acronymfont{\glsentryshortpl{#1}}}%
4338 }
```

entrynumberlist Displays the number list as is.

```
4339 \newcommand*\glsentrynumberlist[1]{%
4340   \glsdoifexists{#1}%
4341   {%
4342     \gls@entry@field{#1}{numberlist}%
4343   }%
4344 }
```

splaynumberlist Formats the number list for the given entry label. Doesn't work with hyperref.

```
4345 \@ifpackageloaded{hyperref} {%
4346   \newcommand*\glsdisplaynumberlist[1]{%
4347     \GlossariesWarning
4348     {%
4349       \string\glsdisplaynumberlist\space
4350       doesn't work with hyperref.^^JUsing
4351       \string\glsentrynumberlist\space instead%
4352     }%
4353     \glsentrynumberlist{#1}%
4354   }%
4355 }%
4356 {%
4357   \newcommand*\glsdisplaynumberlist[1]{%
4358     \glsdoifexists{#1}%
4359     {%
4360       \bgroup
4361
4362       \edef\@glo@label{\glsdetoklabel{#1}}%
4363       \let\@org@glsnumberformat\glsnumberformat
4364       \def\glsnumberformat##1{##1}%
4365       \protected@edef\the@numberlist{%
4366         \csname glo@\@glo@label @numberlist\endcsname}%
4367       \def\@gls@numlist@sep{}%
4368       \def\@gls@numlist@nextsep{}%
4369       \def\@gls@numlist@lastsep{}%
4370       \def\@gls@thislist{}%
4371       \def\@gls@donext@def{}%
4372       \renewcommand\do[1]{%
4373         \protected@edef\@gls@thislist{%
4374           \@gls@thislist
4375           \noexpand\@gls@numlist@sep
4376           ##1%
4377         }%
4378         \let\@gls@numlist@sep\@gls@numlist@nextsep
4379         \def\@gls@numlist@nextsep{\glsnumlistsep}%
4380         \@gls@donext@def
4381         \def\@gls@donext@def{%
4382           \def\@gls@numlist@lastsep{\glsnumlistlastsep}%
4383         }%
4384       }%
4385     }%
4386   }%
4387 }
```

```

4384     \expandafter \glsnumlistparser \expandafter{\the@numberlist}%
4385     \let\@gls@numlist@sep\@gls@numlist@lastsep
4386     \@gls@thislist
4387     \egroup
4388 }%
4389 }
4390 }

```

`\glsnumlistsep`

```
4391 \newcommand*{\glsnumlistsep}{, }
```

`\glsnumlistlastsep`

```
4392 \newcommand*{\glsnumlistlastsep}{ \& }
```

`\gls hyperlink`

Provide a hyperlink to a glossary entry without adding information to the glossary file. The entry needs to be added using a command like `\gls link` or `\gls add` to ensure that the target is defined. The first (optional) argument specifies the link text. The entry name is used by default. The second argument is the entry label.

```

4393 \newcommand*{\gls hyperlink}[2][\gls entrytext{\@glo@label}]{%
4394 \def\@glo@label{#2}%
4395 \@gls link{\glo link prefix\gls detok label{#2}}{#1}}

```

1.12 Adding an entry to the glossary without generating text

The following keys are provided for `\gls add` and `\gls add all`:

```

4396 \define@key{gloss add}{counter}{\def\@gls@counter{#1}}
4397 \define@key{gloss add}{format}{\def\@gls number format{#1}}

```

This key is only used by `\gls add all`:

```
4398 \define@key{gloss add}{types}{\def\@glo@type{#1}}
```

```
\gls add[options]{label}
```

Add a term to the glossary without generating any link text. The optional argument indicates which counter to use, and how to format it (using a key-value list) the second argument is the entry label. Note that *options* only has two keys: counter and format (the types key will be ignored).

`\gls add`

```
4399 \newrobustcmd*{\gls add}[2][ ]{%
```

Need to move to horizontal mode if not already in it, but only if not in preamble.

```

4400 \@gls@adjustmode
4401 \gls do if exists{#2}%
4402 {%
4403 \def\@gls number format{gls number format}%

```

```

4404 \edef\@gls@counter{\csname glo@\glsdetoklabel{#2}@counter\endcsname}%
4405 \setkeys{glossadd}{#1}%

```

Store the entry's counter in `\theglsentrycounter`

```

4406 \@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.

```

4407 \@do@wrglossary{#2}%
4408 }%
4409 }

```

`@gls@adjustmode`

```

4410 \newcommand*{\@gls@adjustmode}{}
4411 \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`

```

4412 \newrobustcmd*{\glsaddall}[1] [] {%
4413 \edef\@glo@type{\@glo@types}%
4414 \setkeys{glossadd}{#1}%
4415 \forallglsentries[\@glo@type]{\@glo@entry}{%
4416 \glsadd[#1]{\@glo@entry}%
4417 }%
4418 }

```

`\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.

```

4419 \newrobustcmd*{\glsaddallunused}[1] [\@glo@types] {%
4420 \forallglsentries[#1]{\@glo@entry}%
4421 {%
4422 \ifglsused{\@glo@entry}{\glsadd[format=glsignore]{\@glo@entry}}%
4423 }%
4424 }

```

`\glsignore`

```

4425 \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.
4426 \edef\glsopenbrace{\expandafter\@gobble\string\{}}

\glsclosebrace  Define \glsclosebrace to make it easier to write an opening brace to a file.
4427 \edef\glsclosebrace{\expandafter\@gobble\string\}}

\glsbackslash  Define \glsbackslash to make it easier to write a backslash to a file.
4428 \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.
4429 \edef\glsquote#1{\string"#1\string"}

\glspercentchar  Define \glspercentchar to make it easier to write a percent character to a file.
4430 \edef\glspercentchar{\expandafter\@gobble\string\%}

\glstildechar   Define \glstildechar to make it easier to write a tilde character to a file.
4431 \edef\glstildechar{\string~}

\@glsfirstletter  Define the first letter to come after the digits 0,...,9. Only required for xindy.
4432 \ifglsxindy
4433   \newcommand*{\@glsfirstletter}{A}
4434 \fi

\GlsSetXdyFirstLetterAfterDigits  Sets the first letter to come after the digits 0,...,9.
4435 \ifglsxindy
4436   \newcommand*{\GlsSetXdyFirstLetterAfterDigits}[1]{%
4437     \renewcommand*{\@glsfirstletter}{#1}}
4438 \else
```



```

4439 \newcommand*\GlsSetXdyFirstLetterAfterDigits}[1]{%
4440   \glsnoxindywarning\GlsSetXdyFirstLetterAfterDigits}
4441 \fi

```

`\@glsminrange` Define the minimum number of successive location references to merge into a range.

```
4442 \newcommand*\@glsminrange}{2}
```

`yMinRangeLength` Set the minimum range length. The value must either be none or a positive integer. The glossaries package doesn't check if the argument is valid, that is left to xindy.

```

4443 \ifglsxindy
4444   \newcommand*\GlsSetXdyMinRangeLength}[1]{%
4445     \renewcommand*\@glsminrange}{#1}}
4446 \else
4447   \newcommand*\GlsSetXdyMinRangeLength}[1]{%
4448     \glsnoxindywarning\GlsSetXdyMinRangeLength}
4449 \fi

```

`\writeist`

```
4450 \ifglsxindy
```

Code to use if xindy is required.

```
4451 \def\writeist{%
```

Define write register if not already defined

```
4452   \ifundef{\glswrite}{\newwrite\glswrite}{}
```

Update attributes list

```
4453   \@gls@addpredefinedattributes
```

Open the file.

```
4454   \openout\glswrite=\istfilename
```

Write header comment at the start of the file

```
4455   \write\glswrite{;; xindy style file created by the glossaries
```

```
4456     package}%
```

```
4457   \write\glswrite{;; for document '\jobname' on
```

```
4458     \the\year-\the\month-\the\day}%
```

Specify the required styles

```
4459   \write\glswrite{^^J; required styles^^J}
```

```
4460   \@for\@xdystyle:=\@xdyrequiredstyles\do{%
```

```
4461     \ifx\@xdystyle\@empty
```

```
4462     \else
```

```
4463       \protected@write\glswrite}{(require
```

```
4464         \string"\@xdystyle.xdy\string")}%
```

```
4465     \fi
```

```
4466   }%
```

List the allowed attributes (possible values used by the format key)

```
4467   \write\glswrite{^^J%
```

```
4468     ; list of allowed attributes (number formats)^^J}%
```

```
4469   \write\glswrite{(define-attributes ((\@xdyattributes)))}%
```

Define any additional alphabets

```
4470 \write\glswrite{^^J; user defined alphabets^^J}%  
4471 \write\glswrite{\@xdyuseralphabets}%
```

Define location classes.

```
4472 \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.

```
4473 \@for\@gls@classI:=\@gls@xdy@locationlist\do{%
```

Case were $\langle Hprefix \rangle$ is empty:

```
4474 \protected@write\glswrite{}\{(define-location-class  
4475 \string"\@gls@classI\string"^^J\space\space\space  
4476 (  
4477 :sep "{}"  
4478 \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space  
4479 :sep "}"  
4480 )  
4481 ^^J\space\space\space  
4482 :min-range-length \@glsminrange^^J%  
4483 )  
4484 }%
```

Nested iteration over all classes:

```
4485 {%  
4486 \@for\@gls@classII:=\@gls@xdy@locationlist\do{%  
4487 \protected@write\glswrite{}\{(define-location-class  
4488 \string"\@gls@classII-\@gls@classI\string"  
4489 ^^J\space\space\space  
4490 (  
4491 :sep "{"  
4492 \csname @gls@xdy@Lclass@\@gls@classII\endcsname\space  
4493 :sep "}"  
4494 \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space  
4495 :sep "}"  
4496 )  
4497 ^^J\space\space\space  
4498 :min-range-length \@glsminrange^^J%  
4499 )  
4500 }%  
4501 }%  
4502 }%  
4503 }%
```

User defined location classes (needs checking for new location format).

```
4504 \write\glswrite{^^J; user defined location classes}%  
4505 \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.)

```

4506 \write\glswrite{^^J; define cross-reference class^^J}%
4507 \write\glswrite{(define-crossref-class \string"see\string"
4508 :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.)

```

4509 \write\glswrite{(markup-crossref-list
4510 :class \string"see\string"^^J\space\space\space
4511 :open \string"\string\glsseeformat\string"
4512 :close \string"{}\string")}%

```

List the order to sort the classes.

```

4513 \write\glswrite{^^J; define the order of the location classes}%
4514 \write\glswrite{(define-location-class-order
4515 (\@xdylocationclassorder))}%

```

Specify what to write to the start and end of the glossary file.

```

4516 \write\glswrite{^^J; define the glossary markup^^J}%

4517 \write\glswrite{(markup-index^^J\space\space\space
4518 :open \string"\string
4519 \glossarysection[\string\glossarytoctitle]{\string
4520 \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`)

```

4521 \@for\@this@ctr:=\@xdycounters\do{%
4522   {%
4523     \@for\@this@attr:=\@xdyattributelist\do{%
4524       \protected\write\glswrite{}{\string\providecommand*%
4525         \expandafter\string
4526         \csname glsX\@this@ctr X\@this@attr\endcsname[2]%
4527         {%
4528           \string\setentrycounter
4529           [\expandafter\@gobble\string\#1]{\@this@ctr}%
4530           \expandafter\string
4531           \csname\@this@attr\endcsname
4532           {\expandafter\@gobble\string\#2}%
4533         }%
4534       }%
4535     }%
4536   }%
4537 }%

```

Add the end part of the open tag and the rest of the markup-index information:

```

4538 \write\glswrite{%
4539   \string\begin
4540   {theglossary}\string\glossaryheader\glstildechar n\string" ^^J\space
4541   \space\space:close \string"\glspercentchar\glstildechar n\string

```

```

4542     \end{theglossary}\string\glossarypostamble
4543     \glstildechar n\string" ^^J\space\space\space
4544     :tree)}}%

```

Specify what to put between letter groups

```

4545     \write\glswrite{(markup-letter-group-list
4546     :sep \string"\string\glsgroupskip\glstildechar n\string")}%

```

Specify what to put between entries

```

4547     \write\glswrite{(markup-indexentry
4548     :open \string"\string\relax \string\glsresetentrylist
4549     \glstildechar n\string")}%

```

Specify how to format entries

```

4550     \write\glswrite{(markup-locclass-list :open
4551     \string"\glsopenbrace\string\glossaryentrynumbers
4552     \glsopenbrace\string\relax\space \string"^^J\space\space\space
4553     :sep \string", \string"
4554     :close \string"\glsclosebrace\glsclosebrace\string")}%

```

Specify how to separate location numbers

```

4555     \write\glswrite{(markup-locref-list
4556     :sep \string"\string\delimN\space\string")}%

```

Specify how to indicate location ranges

```

4557     \write\glswrite{(markup-range
4558     :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.

```

4559     \@onelevel@sanitize\gls@suffixF
4560     \@onelevel@sanitize\gls@suffixFF
4561     \ifx\gls@suffixF\@empty
4562     \else
4563     \write\glswrite{(markup-range
4564     :close "\gls@suffixF" :length 1 :ignore-end)}%
4565     \fi
4566     \ifx\gls@suffixFF\@empty
4567     \else
4568     \write\glswrite{(markup-range
4569     :close "\gls@suffixFF" :length 2 :ignore-end)}%
4570     \fi

```

Specify how to format locations.

```

4571     \write\glswrite{^^J; define format to use for locations^^J}%
4572     \write\glswrite{\@xdylocref}%

```

Specify how to separate letter groups.

```

4573     \write\glswrite{^^J; define letter group list format^^J}%
4574     \write\glswrite{(markup-letter-group-list
4575     :sep \string"\string\glsgroupskip\glstildechar n\string")}%

```

Define letter group headings.

```
4576 \write\glswrite{^^J; letter group headings^^J}%
4577 \write\glswrite{(markup-letter-group
4578 :open-head \string"\string\glsgroupheading
4579 \glsopenbrace\string"^^J\space\space\space
4580 :close-head \string"\glsclosebrace\string")}%
```

Define additional letter groups.

```
4581 \write\glswrite{^^J; additional letter groups^^J}%
4582 \write\glswrite{\@xdylettergroups}%
```

Define additional sort rules

```
4583 \write\glswrite{^^J; additional sort rules^^J}
4584 \write\glswrite{\@xdysortrules}%
```

Close the style file

```
4585 \closeout\glswrite
```

Suppress any further calls.

```
4586 \let\writeist\relax
4587 }
4588 \else
```

Code to use if makeindex is required.

```
4589 \edef\@gls@actualchar{\string?}
4590 \edef\@gls@encapchar{\string|}
4591 \edef\@gls@levelchar{\string!}
4592 \edef\@gls@quotechar{\string"}
4593 \def\writeist{\relax
4594 \ifundef{\glswrite}{\newwrite\glswrite}{\relax
4595 \openout\glswrite=\istfilename
4596 \write\glswrite{\glspercentchar\space makeindex style file
4597 created by the glossaries package}
4598 \write\glswrite{\glspercentchar\space for document
4599 '\jobname' on \the\year-\the\month-\the\day}
4600 \write\glswrite{actual '\@gls@actualchar'}
4601 \write\glswrite{encap '\@gls@encapchar'}
4602 \write\glswrite{level '\@gls@levelchar'}
4603 \write\glswrite{quote '\@gls@quotechar'}
4604 \write\glswrite{keyword \string"\string\glossaryentry\string"}
4605 \write\glswrite{preamble \string"\string\glossarysection[\string
4606 \glossarytoctitle]{\string\glossarytitle}\string
4607 \glossarypreamble\string\n\string\begin{theglossary}\string
4608 \glossaryheader\string\n\string"}
4609 \write\glswrite{postamble \string"\string%\string\n\string
4610 \end{theglossary}\string\glossarypostamble\string\n
4611 \string"}
4612 \write\glswrite{group_skip \string"\string\glsgroupskip\string\n
4613 \string"}
4614 \write\glswrite{item_0 \string"\string%\string\n\string"}
4615 \write\glswrite{item_1 \string"\string%\string\n\string"}
```

```

4616 \write\glswrite{item_2 \string"\string%\string\n\string"}
4617 \write\glswrite{item_01 \string"\string%\string\n\string"}
4618 \write\glswrite{item_x1
4619 \string"\string\relax \string\glsresetentrylist\string\n
4620 \string"}
4621 \write\glswrite{item_12 \string"\string%\string\n\string"}
4622 \write\glswrite{item_x2
4623 \string"\string\relax \string\glsresetentrylist\string\n
4624 \string"}

4625 \write\glswrite{delim_0 \string"\string{\string
4626 \glossaryentrynumbers\string{\string\relax \string"}
4627 \write\glswrite{delim_1 \string"\string{\string
4628 \glossaryentrynumbers\string{\string\relax \string"}
4629 \write\glswrite{delim_2 \string"\string{\string
4630 \glossaryentrynumbers\string{\string\relax \string"}
4631 \write\glswrite{delim_t \string"\string}\string}\string"}
4632 \write\glswrite{delim_n \string"\string\delimN \string"}
4633 \write\glswrite{delim_r \string"\string\delimR \string"}
4634 \write\glswrite{headings_flag 1}
4635 \write\glswrite{heading_prefix
4636 \string"\string\glsgroupheading\string{\string"}
4637 \write\glswrite{heading_suffix
4638 \string"\string}\string\relax
4639 \string\glsresetentrylist \string"}
4640 \write\glswrite{symhead_positive \string"glssymbols\string"}
4641 \write\glswrite{numhead_positive \string"glnumbers\string"}
4642 \write\glswrite{page_compositor \string"glscpositor\string"}
4643 \@gls@escbsdq\gls@suffixF
4644 \@gls@escbsdq\gls@suffixFF
4645 \ifx\gls@suffixF\@empty
4646 \else
4647 \write\glswrite{suffix_2p \string"\gls@suffixF\string"}
4648 \fi
4649 \ifx\gls@suffixFF\@empty
4650 \else
4651 \write\glswrite{suffix_3p \string"\gls@suffixFF\string"}
4652 \fi
4653 \closeout\glswrite
4654 \let\writeist\relax
4655 }
4656 \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
```

```
4657 \newcommand{\noist}{%
```

```
Update attributes list
```

```

4658 \@gls@addpredefinedattributes
4659 \let\writeist\relax
4660 }

```

\@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

```

4661 \newcommand*\@makeglossary}[1]{%
4662 \ifglossaryexists{#1}%
4663 {%

```

Only create a new write if `savewrites=false` otherwise create a token to collect the information.

```

4664 \ifglssavewrites
4665 \expandafter\newtoks\csname glo@#1@filetok\endcsname
4666 \else
4667 \expandafter\newwrite\csname glo@#1@file\endcsname
4668 \expandafter\@glsopenfile\csname glo@#1@file\endcsname{#1}%
4669 \fi
4670 \@gls@renewglossary
4671 \writeist
4672 }%
4673 {%
4674 \PackageError{glossaries}%
4675 {Glossary type ‘#1’ not defined}%
4676 {New glossaries must be defined before using \string\makeglossary}%
4677 }%
4678 }

```

\@glsopenfile Open write file associated with the given glossary.

```

4679 \newcommand*\@glsopenfile}[2]{%
4680 \immediate\openout#1=\jobname.\csname @glotype@#2@out\endcsname
4681 \PackageInfo{glossaries}{Writing glossary file
4682 \jobname.\csname @glotype@#2@out\endcsname}%
4683 }

```

\@closegls

```

4684 \newcommand*\@closegls}[1]{%
4685 \closeout\csname glo@#1@file\endcsname
4686 }
4687 % \end{macrocode}

```

```

4688 %\end{macro}
4689 %
4690 %\begin{macro}{\@gls@automake}
4691 %\changes{4.08}{2014-07-30}{new}
4692 %   \begin{macrocode}
4693 \ifglsxindy
4694   \newcommand*{\@gls@automake}[1]{%
4695     \ifglossaryexists{#1}
4696     {%
4697       \@closegls{#1}%
4698       \ifdefstring{\glsorder}{letter}%
4699       {\def\@gls@order{-M ord/letorder }}%
4700       {\let\@gls@order\@empty}%
4701       \ifcsundef{@xdy@#1@language}%
4702       {\let\@gls@langmod\@xdy@main@language}%
4703       {\letcs\@gls@langmod{@xdy@#1@language}}%
4704       \edef\@gls@dothiswrite{\noexpand\write18{xindy
4705         -I xindy
4706         \@gls@order
4707         -L \@gls@langmod\space
4708         -M \@gls@istfilebase\space
4709         -C \@gls@codepage\space
4710         -t \jobname.\csuse{@glotype@#1@log}
4711         -o \jobname.\csuse{@glotype@#1@in}
4712         \jobname.\csuse{@glotype@#1@out}}}%
4713       }%
4714       \@gls@dothiswrite
4715     }%
4716     {%
4717       \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
4718     }%
4719   }
4720 \else
4721   \newcommand*{\@gls@automake}[1]{%
4722     \ifglossaryexists{#1}
4723     {%
4724       \@closegls{#1}%
4725       \ifdefstring{\glsorder}{letter}%
4726       {\def\@gls@order{-l }}%
4727       {\let\@gls@order\@empty}%
4728       \edef\@gls@dothiswrite{\noexpand\write18{makeindex \@gls@order
4729         -s \istfilename\space
4730         -t \jobname.\csuse{@glotype@#1@log}
4731         -o \jobname.\csuse{@glotype@#1@in}
4732         \jobname.\csuse{@glotype@#1@out}}}%
4733       }%
4734       \@gls@dothiswrite
4735     }%
4736     {%

```



```

4737     \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
4738   }%
4739 }
4740 \fi

```

`\makeglossaries` Issue warning that `\makeglossaries` hasn't been used.

```

4741 \newcommand*{\@warn@nomakeglossaries}{}
    Only use this if warning if \printglossary has been used without \makeglossaries
4742 \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`

```

4743 \newcommand*{\makeglossaries}{%
    Define the write used for style file also used for all other output files if savewrites=true.
4744   \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.
4745   \protected@write\@auxout{}{\string\providecommand\string\@glsorder[1]{} }
4746   \protected@write\@auxout{}{\string\providecommand\string\@istfilename[1]{} }

```

Write the name of the style file to the aux file (needed by `makeglossaries`)

```

4747   \protected@write\@auxout{}{\string\@istfilename{\istfilename}}%
4748   \protected@write\@auxout{}{\string\@glsorder{\glsorder}}

```

Iterate through each glossary type and activate it.

```

4749   \@for\@glo@type:=\@glo@types\do{%
4750     \ifthenelse{\equal{\@glo@type}{}}{}{}%
4751     \@makeglossary{\@glo@type}}%
4752   }%

```

New glossaries must be created before `\makeglossaries` so disable `\newglossary`.

```

4753   \renewcommand*\newglossary[4] []{%
4754     \PackageError{glossaries}{New glossaries
4755     must be created before \string\makeglossaries}{You need
4756     to move \string\makeglossaries\space after all your
4757     \string\newglossary\space commands}}%

```

Any subsequent instances of this command should have no effect

```

4758   \let\@makeglossary\relax
4759   \let\makeglossary\relax
4760   \let\makeglossaries\relax

```

Disable all commands that have no effect after `\makeglossaries`

```

4761   \@disable@onlypremakeg

```

Allow see key:

```

4762   \let\gls@checkseeallowed\relax

```

```

Suppress warning about no \makeglossaries
4763 \let\warn@nomakeglossaries\relax

Activate warning about missing \printglossary
4764 \def\warn@noprintglossary{%
4765   \GlossariesWarningNoLine{No \string\printglossary\space
4766     or \string\printglossaries\space
4767     found.^^J(Remove \string\makeglossaries\space if you don't want
4768     any glossaries.)^^JThis document will not have a glossary}%
4769 }%

Declare list parser for \glsdisplaynumberlist
4770 \ifglssavenumberlist
4771   \edef\@gls@dodolistparser{\noexpand\DeclareListParser
4772     {\noexpand\glsnumlistparser}{\delimN}}%
4773   \@gls@dodolistparser
4774 \fi

Prevent user from also using \makenoidxglossaries
4775 \let\makenoidxglossaries\@no@makeglossaries

Prohibit sort key in printgloss family:
4776 \renewcommand*{\@printgloss@setsort}{%
4777   \let\@glo@assign@sortkey\@glo@no@assign@sortkey
4778 }%

Check the automake setting:
4779 \ifglsautomake
4780   \renewcommand*{\@gls@doautomake}{%
4781     \@for\@gls@type:=\@glo@types\do{%
4782       \ifdefempty{\@gls@type}{}%
4783       {\@gls@automake{\@gls@type}}%
4784     }%
4785   }%
4786 \fi
4787 }

Must occur in the preamble:
4788 \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`

```
4789 \let\makeglossary\makeglossaries
```

If `\makeglossaries` hasn't been used, issue a warning. Also issue a warning if neither `\printglossaries` nor `\printglossary` have been used.

```
4790 \AtEndDocument{%
4791   \warn@nomakeglossaries
4792   \warn@noprintglossary
4793 }
```

`noidxglossaries` Analogous to `\makeglossaries` this activates the commands needed for `\printnoidxglossary`

```
4794 \newcommand*{\makenoidxglossaries}{%
```

Redefine empty glossary warning:

```
4795   \renewcommand{\@gls@noref@warn}[1]{%
4796     \GlossariesWarning{Empty glossary for
4797     \string\printnoidxglossary[type={##1}].
4798     Rerun may be required (or you may have forgotten to use
4799     commands like \string\gls).}%
4800   }%
```

Don't escape makeindex/xindy characters

```
4801   \let\@gls@checkmkidxchars\@gobble
```

Write glossary information to aux instead of glossary files

```
4802   \let\@do@wrglossary\gls@noidxglossary
```

Switch on group headings that use the character code:

```
4803   \let\@gls@getgrouptitle\@gls@noidx@getgrouptitle
```

Allow see key:

```
4804   \let\gls@checkseeallowed\relax
```

Redefine cross-referencing macro:

```
4805   \renewcommand{\@do@seeglossary}[2]{%
4806     \edef\@gls@label{\glsdetoklabel{##1}}%
4807     \protected@write\@auxout{}{%
4808       \string\@gls@reference
4809       {\csname glo@\@gls@label @type\endcsname}%
4810       {\@gls@label}%
4811       {%
4812         \string\glsseeformat##2}%
4813       }%
4814     }%
4815   }%
```

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.

```
4816   \AtBeginDocument
4817   {%
4818     \write\@auxout{\string\providecommand\string\@gls@reference[3]{}}%
4819   }%
```

Change warning about no glossares

```
4820 \def\warn@noprntglossary{%
4821   \GlossariesWarningNoLine{No \string\printnoidxglossary\space
4822     or \string\printnoidxglossaries ^^J
4823     found. (Remove \string\makenoidxglossaries\space if you
4824     don't want any glossaries.)^^JThis document will not have a glossary}%
4825 }%
```

Suppress warning about no \makeglossaries

```
4826 \let\warn@nomakeglossaries\relax
```

Prevent user from also using \makeglossaries

```
4827 \let\makeglossaries\@no@makeglossaries
```

Allow sort key in printgloss family:

```
4828 \renewcommand*{\@printgloss@setsort}{%
4829   \let\@glo@assign@sortkey\@glo@assign@sortkey
```

Initialise default sort order:

```
4830   \def\@glo@sorttype{\@glo@default@sorttype}%
4831 }%
```

All entries must be defined in the preamble:

```
4832 \renewcommand*\new@glossaryentry[2]{%
4833   \PackageError{glossaries}{Glossary entries must be
4834     defined in the preamble^^Jwhen you use
4835     \string\makenoidxglossaries}%
4836   {Either move your definitions to the preamble or use
4837     \string\makeglossaries}%
4838 }%
```

Redefine \glsentrynumberlist

```
4839 \renewcommand*{\glsentrynumberlist}[1]{%
4840   \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
4841   \ifdef\@gls@loclist
4842     {%
4843       \glsnoidxloclist{\@gls@loclist}%
4844     }%
4845     {%
4846       ??\glsdoifexists{##1}%
4847       {%
4848         \GlossariesWarning{Missing location list for '##1'. Either
4849           a rerun is required or you haven't referenced the entry.}%
4850       }%
4851     }%
4852 }%
```

Redefine \glsdisplaynumberlist

```
4853 \renewcommand*{\glsdisplaynumberlist}[1]{%
4854   \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
4855   \ifdef\@gls@loclist
4856     {%
```

```

4857     \def\@gls@noidxloclist@sep{%
4858         \def\@gls@noidxloclist@sep{%
4859             \def\@gls@noidxloclist@sep{%
4860                 \glsnumlistsep
4861             }%
4862         \def\@gls@noidxloclist@finalsep{\glsnumlistlastsep}%
4863     }%
4864 }%
4865 \def\@gls@noidxloclist@finalsep{}%
4866 \def\@gls@noidxloclist@prev{}%
4867 \forlistloop{\glsnoidxdisplayloclisthandler}{\@gls@loclist}%
4868 \@gls@noidxloclist@finalsep
4869 \@gls@noidxloclist@prev
4870 }%
4871 {%
4872     ??\glsdoifexists{##1}%
4873     {%
4874         \GlossariesWarning{Missing location list for ‘##1’. Either
4875             a rerun is required or you haven’t referenced the entry.}%
4876     }%
4877 }%
4878 }%

```

Provide a generic way of iterating through the number list:

```

4879 \renewcommand*\glsnumberlistloop}[3]{%
4880     \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
4881     \let\@gls@org@glsnoidxdisplayloc\glsnoidxdisplayloc
4882     \let\@gls@org@glsseeformat\glsseeformat
4883     \let\glsnoidxdisplayloc##2\relax
4884     \let\glsseeformat##3\relax
4885     \ifdef\@gls@loclist
4886     {%
4887         \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
4888     }%
4889     {%
4890         ??\glsdoifexists{##1}%
4891         {%
4892             \GlossariesWarning{Missing location list for ‘##1’. Either
4893                 a rerun is required or you haven’t referenced the entry.}%
4894         }%
4895     }%
4896     \let\glsnoidxdisplayloc\@gls@org@glsnoidxdisplayloc
4897     \let\glsseeformat\@gls@org@glsseeformat
4898 }%

```

Modify sanitize sort function

```

4899 \let\@gls@sanitizesort\@gls@noidx@sanitizesort
4900 \let\@gls@nosanitizesort\@gls@noidx@nosanitizesort
4901 \@gls@noidx@setsanitizesort
4902 }

```

Preamble-only command:

```
4903 \@onlypreamble{\makenoidxglossaries}
```

```
glsnumberlistloop \glsnumberlistloop{<label>}{<handler>}
```

```
4904 \newcommand*{\glsnumberlistloop}[2]{%
4905   \PackageError{glossaries}{\string\glsnumberlistloop\space
4906     only works with \string\makenoidxglossaries}{}%
4907 }
```

listloophandler Handler macro for `\glsnumberlistloop`. (The argument should be in the form `\glsnoidxdisplayloc{<prefix>}`.)

```
4908 \newcommand*{\glsnoidxnumberlistloophandler}[1]{%
4909   #1%
4910 }
```

@makeglossaries Can't use both `\makeglossaries` and `\makenoidxglossaries`

```
4911 \newcommand*{\@no@makeglossaries}{%
4912   \PackageError{glossaries}{You can't use both
4913     \string\makeglossaries\space and \string\makenoidxglossaries}%
4914   {Either use one or other (or none) of those commands but not both
4915     together.}%
4916 }
```

@gls@noref@warn Warning when no instances of `\@gls@reference` found.

```
4917 \newcommand{\@gls@noref@warn}[1]{%
4918   \GlossariesWarning{\string\makenoidxglossaries\space
4919     is required to make \string\printnoidxglossary[type={#1}] work}%
4920 }
```

@noidxglossary Write the glossary information to the aux file:

```
4921 \newcommand*{\gls@noidxglossary}{%
4922   \protected@write\@auxout{}{%
4923     \string\@gls@reference
4924       {\csname glo@\@gls@label @type\endcsname}%
4925       {\@gls@label}%
4926       {\string\glsnoidxdisplayloc
4927         {\@glo@counterprefix}%
4928         {\@gls@counter}%
4929         {\@glsnumberformat}%
4930         {\@glslocref}%
4931       }%
4932   }%
4933 }
```

1.14 Writing information to associated files

`\istfile` Deprecated.

```
4934 \def\istfile{\glswrite}
```

At the end of the document, the files should be created if `savewrites=true`.

```
4935 \AtEndDocument{%
4936   \glswritefiles
4937 }
```

`\@glswritefiles` Only write the files if `savewrites=true`

```
4938 \newcommand*{\@glswritefiles}{%
```

Iterate through all the glossaries

```
4939 \forallglossaries{\@glo@type}{%
```

Check for empty glossaries (patch provided by Patrick Häcker)

```
4940   \ifcsundef{glo@\@glo@type @filetok}%
4941     {%
4942       \def\gls@tmp{}%
4943     }%
4944     {%
4945       \edef\gls@tmp{\expandafter\the
4946         \csname glo@\@glo@type @filetok\endcsname}%
4947     }%
4948     \ifx\gls@tmp\@empty
4949       \ifx\@glo@type\glsdefaulttype
4950         \GlossariesWarningNoLine{Glossary ‘\@glo@type’ has no
4951           entries.^^JRemember to use package option ‘nomain’ if
4952 you
4953           don’t want to^^Juse the main glossary}%
4954       \else
4955         \GlossariesWarningNoLine{Glossary ‘\@glo@type’ has no
4956           entries}%
4957       \fi
4958     \else
4959       \@glsopenfile{\glswrite}{\@glo@type}%
4960       \immediate\write\glswrite{%
4961         \expandafter\the
4962           \csname glo@\@glo@type @filetok\endcsname}%
4963       \immediate\closeout\glswrite
4964     \fi
4965   }%
4966 }
```

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`

```
4967 \if@gls@docloaded
4968 \else
4969   \renewcommand*{\glossary}[1][main]{\gls@glossary{#1}}
4970 \fi
```

The associated number should be stored in `\theglstrycounter` before using `\gls@glossary`.

`\gls@glossary`

```
4971 \newcommand*{\gls@glossary}[1]{%
4972   \@gls@glossary{#1}%
4973 }
```

`\@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.

```
4974 \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`

```
4975 \newcommand{\@gls@renewglossary}{%
4976   \gdef\@gls@glossary##1{\@bsphack\beginngroup\gls@wrglossary{##1}}%
4977   \let\@gls@renewglossary\@empty
4978 }
```

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`

```
4979 \newcommand*{\gls@wrglossary}[2]{%
4980   \ifglssavewrites
4981     \protected@edef\@gls@tmp{\the\csname glo@#1@filetok\endcsname#2}%
4982     \expandafter\global\expandafter\csname glo@#1@filetok\endcsname
4983       \expandafter{\@gls@tmp^^J}%
4984   \else
```



```

4985 \ifcsdef{glo@#1@file}%
4986 {%
4987 \expandafter\protected@write\csname glo@#1@file\endcsname{%
4988 \gls@disablepagerefexpansion}{#2}%
4989 }%
4990 {%
4991 \ifignoredglossary{#1}{}%
4992 {%
4993 \GlossariesWarning{No file defined for glossary '#1'}%
4994 }%
4995 }%
4996 \fi
4997 \endgroup\@esphack
4998 }

```

\do@wrglossary

```

4999 \newcommand*\do@wrglossary}[1]{%
5000 \glswriteentry{#1}{\do@wrglossary{#1}}%
5001 }

```

\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.

```

5002 \newcommand*\glswriteentry}[2]{%
5003 \ifglsindexonlyfirst
5004 \ifglsused{#1}{#2}%
5005 \else
5006 #2%
5007 \fi
5008 }

```

protected@pagefmts List of page formats to be protected against expansion.

```

5009 \newcommand{\gls@protected@pagefmts}{%
5010 \gls@numberpage,\gls@alphpage,\gls@Alphpage,\gls@romanpage,\gls@Romanpage%
5011 }

```

agerefexpansion

```

5012 \newcommand*\gls@disablepagerefexpansion){%
5013 \@for\@gls@this:=\gls@protected@pagefmts\do
5014 {%
5015 \expandafter\let\@gls@this\relax
5016 }%
5017 }

```

\gls@alphpage

```

5018 \newcommand*\gls@alphpage}{\@alph\c@page}

```

\gls@Alphpage

```

5019 \newcommand*\gls@Alphpage}{\@Alph\c@page}

```

`\gls@numberpage`

```
5020 \newcommand*{\gls@numberpage}{\number\c@page}
```

`\gls@romanpage`

```
5021 \newcommand*{\gls@romanpage}{\romannumeral\c@page}
```

`\gls@Romanpage`

```
5022 \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).

```
5023 \newcommand*{\glsaddprotectedpagefmt}[1]{%
5024   \eappto\gls@protected@pagefmts{\expandonce{\csname gls#1page\endcsname}}%
5025   \csedef{gls#1page}{\expandonce{\csname#1\endcsname}\noexpand\c@page}%
5026   \eappto\@wrglossarynumberhook{%
5027     \noexpand\let\expandonce{\csname org@gls#1\endcsname}%
5028     \expandonce{\csname#1\endcsname}%
5029     \noexpand\def\expandonce{\csname#1\endcsname}{%
5030       \noexpand\@wrglossary@pageformat
5031       \expandonce{\csname gls#1page\endcsname}%
5032       \expandonce{\csname org@gls#1\endcsname}%
5033     }%
5034   }%
5035 }
```

`ssarynumberhook` Hook used by `\@do@wrglossary`

```
5036 \newcommand*\@wrglossarynumberhook{}
```

`sary@pageformat`

```
5037 \newcommand{\@wrglossary@pageformat}[3]{%
5038   \ifx#3\c@page #1\else #2#3\fi
5039 }
```

`@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.

```
5040 \newcommand*{\@do@wrglossary}[1]{%
```

```
5041   \begingroup
```

First a bit of hackery to prevent premature expansion of `\c@page`. Store original definitions:

```
5042   \let\orgthe\the
5043   \let\orgnumber\number
5044   \let\orgromannumeral\romannumeral
5045   \let\orgalph\@alph
5046   \let\orgAlph\@Alph
5047   \let\orgRoman\@Roman
```

Redefine:

```
5048 \def\the##1{%
5049 \ifx##1\c@page \gls@numberpage\else\orgthe##1\fi}%
5050 \def\number##1{%
5051 \ifx##1\c@page \gls@numberpage\else\orgnumber##1\fi}%
5052 \def\romannumeral##1{%
5053 \ifx##1\c@page \gls@romanpage\else\orgromannumeral##1\fi}%
5054 \def\@Roman##1{%
5055 \ifx##1\c@page \gls@Romanpage\else\orgRoman##1\fi}%
5056 \def\@alph##1{%
5057 \ifx##1\c@page \gls@alphpage\else\orgalph##1\fi}%
5058 \def\@Alph##1{%
5059 \ifx##1\c@page \gls@Alphpage\else\orgAlph##1\fi}%
```

Add hook to allow for other number formats:

```
5060 \@wrglossarynumberhook
```

Prevent expansion:

```
5061 \gls@disablepagerefexpansion
```

Now store location in \@glslocref:

```
5062 \protected@xdef\@glslocref{\theglsentrycounter}%
5063 \endgroup
```

Escape any special characters

```
5064 \@gls@checkmkidxchars\@glslocref
```

Check if the hyper-location is the same as the location and set the hyper prefix.

```
5065 \expandafter\ifx\theHglentrycounter\theglsentrycounter\relax
5066 \def\@glo@counterprefix{%
5067 \else
5068 \protected@edef\@glsHlocref{\theHglentrycounter}%
5069 \@gls@checkmkidxchars\@glsHlocref
5070 \edef\@do@gls@getcounterprefix{\noexpand\@gls@getcounterprefix
5071 {\@glslocref}{\@glsHlocref}%
5072 }%
5073 \@do@gls@getcounterprefix
5074 \fi
```

De-tok label if required

```
5075 \edef\@gls@label{\glsdetoklabel{#1}}%
```

Write the information to file:

```
5076 \@do@@wrglossary
5077 }
```

```
@do@@wrglossary
```

```
5078 \newcommand*{\@do@@wrglossary}{%
```

Determine whether to use xindy or makeindex syntax

```
5079 \ifglsxindy
```

Need to determine if the formatting information starts with a (or) indicating a range.

```

5080 \expandafter\@glo@check@mkidxrangechar\@glsnumberformat\@nil
5081 \def\@glo@range{}%
5082 \expandafter\if\@glo@prefix(\relax
5083 \def\@glo@range{:open-range}%
5084 \else
5085 \expandafter\if\@glo@prefix)\relax
5086 \def\@glo@range{:close-range}%
5087 \fi
5088 \fi

```

Write to the glossary file using xindy syntax.

```

5089 \gls@glossary{\csname glo@\@gls@label @type\endcsname}{%
5090 (indexentry :key (\csname glo@\@gls@label @index\endcsname)
5091 :locref \string"\@glo@counterprefix}{\@gls@locref}\string" %
5092 :attr \string"\@gls@counter\@glo@suffix\string"
5093 \@glo@range
5094 )
5095 }%
5096 \else

```

Convert the format information into the format required for makeindex

```

5097 \@set@glo@numformat{\@glo@numfmt}{\@gls@counter}{\@glsnumberformat}%
5098 {\@glo@counterprefix}%

```

Write to the glossary file using makeindex syntax.

```

5099 \gls@glossary{\csname glo@\@gls@label @type\endcsname}{%
5100 \string\glossaryentry{\csname glo@\@gls@label @index\endcsname
5101 \@gls@encapchar\@glo@numfmt}{\@gls@locref}}%
5102 \fi
5103 }

```

`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.)

```

5104 \newcommand*\@gls@getcounterprefix[2]{%
5105 \edef\@gls@thisloc{#1}\edef\@gls@thisHloc{#2}%
5106 \ifx\@gls@thisloc\@gls@thisHloc
5107 \def\@glo@counterprefix{}%
5108 \else
5109 \def\@gls@get@counterprefix##1.#1##2\end@getprefix{%
5110 \def\@glo@tmp{##2}%
5111 \ifx\@glo@tmp\@empty
5112 \def\@glo@counterprefix{}%
5113 \else
5114 \def\@glo@counterprefix{##1}%
5115 \fi

```

```

5116 }%
5117 \@gls@get@counterprefix#2.#1\end@getprefix
Warn if no prefix can be formed.
5118 \ifx\@glo@counterprefix\@empty
5119 \GlossariesWarning{Hyper target ‘#2’ can’t be formed by
5120 prefixing^^Jlocation ‘#1’. You need to modify the
5121 definition of \string\theH\@gls@counter^^Jotherwise you
5122 will get the warning: “name{\@gls@counter.#1}’ has been^^J
5123 referenced but does not exist”}%
5124 \fi
5125 \fi
5126 }

```

1.15 Glossary Entry Cross-References

`\do@seeglossary` Write the glossary entry with a cross reference. The first argument is the entry’s label, the second must be in the form `[\langle tag \rangle]{\langle list \rangle}`, where `\langle tag \rangle` is a tag such as “see” and `\langle list \rangle` is a list of labels.

```

5127 \newcommand{\@do@seeglossary}[2]{%
5128 \def\@gls@xref{#2}%
5129 \@onelevel@sanitize\@gls@xref
5130 \@gls@checkmkidxchars\@gls@xref
5131 \ifglsxindy
5132 \gls@glossary{\csname glo@#1@type\endcsname}{%
5133 (indexentry
5134 :key (\csname glo@#1@index\endcsname)
5135 :xref (\string"\@gls@xref\string")
5136 :attr \string"see\string"
5137 )
5138 }%
5139 \else
5140 \gls@glossary{\csname glo@#1@type\endcsname}{%
5141 \string\glossaryentry{\csname glo@#1@index\endcsname
5142 \@gls@encapchar glsseeformat\@gls@xref}{Z}}%
5143 \fi
5144 }

```

`\@gls@fixbraces` If no optional argument is specified, list needs to be enclosed in a set of braces.

```

5145 \def\@gls@fixbraces#1#2#3\@nil{%
5146 \ifx#2[\relax
5147 \@gls@fixbraces#1#2#3\@end@fixbraces
5148 \else
5149 \def#1{{#2#3}}%
5150 \fi
5151 }

```

`@@gls@fixbraces`

```

5152 \def\@gls@fixbraces#1[#2]#3\@end@fixbraces{%
5153   \def#1{[#2]{#3}}%
5154 }

```

`\glssee` `\glssee{<label>}{<cross-ref list>}`

```

5155 \DeclareRobustCommand*\glssee}[3][\seename]{%
5156   \@do@seeglossary{#2}{#1}{#3}}
5157 \newcommand*\@glssee}[3][\seename]{%
5158   \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.

```

5159 \DeclareRobustCommand*\glsseeformat}[3][\seename]{%
5160   \emph{#1} \glsseelist{#2}}

```

`\glsseelist` `\glsseelist{<list>}` formats list of entry labels.

```

5161 \DeclareRobustCommand*\glsseelist}[1]{%

```

If there is only one item in the list, set the last separator to do nothing.

```

5162   \let\@gls@dolast\relax

```

Don’t display separator on the first iteration of the loop

```

5163   \let\@gls@donext\relax

```

Iterate through the labels

```

5164   \@for\@gls@thislabel:=#1\do{%

```

Check if on last iteration of loop

```

5165     \ifx\@xfor@nextelement\@nnil

```

```

5166       \@gls@dolast

```

```

5167     \else

```

```

5168       \@gls@donext

```

```

5169     \fi

```

Display the entry for this label. (Expanding label as it’s a temporary control sequence that’s used elsewhere.)

```

5170     \expandafter\glsseeitem\expandafter{\@gls@thislabel}%

```

Update separators

```

5171     \let\@gls@dolast\glsseelastsep

```

```

5172     \let\@gls@donext\glsseesep

```

```

5173   }%

```

```

5174 }

```

`\glsseelastsep` Separator to use between penultimate and ultimate entries in a cross-referencing list.

```

5175 \newcommand*\glsseelastsep{\space\andname\space}

```

`\glsseesep` Separator to use between entries in a cross-referencing list.

```

5176 \newcommand*\glsseesep}{, }

```

`\glsseeitem` `\glsseeitem{<label>}` formats individual entry in a cross-referencing list.

```
5177 \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.)

```
5178 \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.

```
5179 \newcommand*\gls@save@numberlist}[1]{%
5180   \ifglssavenumberlist
5181     \toks@{#1}%
5182     \edef\@do@writeaux@info{%
5183       \noexpand\csgdef{glo@\glscurrententrylabel @numberlist}{\the\toks@}%
5184     }%
5185     \@onelevel@sanitize\@do@writeaux@info
5186     \protected@write\@auxout{}\@do@writeaux@info}%
5187   \fi
5188 }
```

`\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.)

```
5189 \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.

```
5190 \ifcsundef{printglossary}{}%
5191 {%
  If \printglossary is already defined, issue a warning and undefine it.
5192   \@gls@warnonglossdefined
5193   \undef\printglossary
5194 }
```

`\printglossary` has an optional argument. The default value is to set the glossary type to the main glossary.

```
5195 \newcommand*\printglossary}[1][type=\glsdefaulttype]{%
5196   \@printglossary{#1}{\@print@glossary}%
5197 }
```

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`

```
5198 \newcommand*\printglossaries}{%
5199 \forallglossaries{\@glo@type}{\printglossary[type=\@glo@type]}%
5200 }
```

`\printnoidxglossary` Provide an alternative to `\printglossary` that doesn't require an external indexing application. Entries won't be sorted and the location list will be empty.

```
5201 \newcommand*\printnoidxglossary}[1][type=\glsdefaulttype]{%
5202 \@printglossary{#1}{\@printnoidxglossary}%
5203 }
```

`\printnoidxglossaries` Analogous to `\printglossaries`

```
5204 \newcommand*\printnoidxglossaries}{%
5205 \forallglossaries{\@glo@type}{\printnoidxglossary[type=\@glo@type]}%
5206 }
```

`\printgloss@setsort` Initialise to do nothing.

```
5207 \newcommand*\@printgloss@setsort}{}
```

`\preglossaryhook`

```
5208 \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.

```
5209 \newcommand{\@printglossary}[2]{%
```

Set up defaults.

```
5210 \def\@glo@type{\glsdefaulttype}%
5211 \def\glossarytitle{\csname @glo@type @title\endcsname}%
```

```
5212 \def\glossarytoctitle{\glossarytitle}%
```

```
5213 \let\org@glossarytitle\glossarytitle
```

```
5214 \def\@glossarystyle{%
```

```
5215 \ifx\@glossary@default@style\relax
```

```
5216 \GlossariesWarning{No default glossary style provided \MessageBreak
```

```
5217 for the glossary '\@glo@type'. \MessageBreak
```

```
5218 Using deprecated fallback. \MessageBreak
```

```
5219 To fix this set the style with \MessageBreak
```



```

5220     \string\setglossarystyle\space or use the \MessageBreak
5221     style key=value option}%
5222 \fi
5223 }%
5224 \def\gls@dotocitle{\glssettocitle{\@glo@type}}%

Store current value of \glossaryentrynumbers. (This may be changed via the optional ar-
gument)
5225 \let\org@glossaryentrynumbers\glossaryentrynumbers

Localise the effects of the optional argument
5226 \bgroup

Activate or deactivate sort key:
5227 \@printgloss@setsort

Determine settings specified in the optional argument.
5228 \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)
5229 \ifx\glossarytitle\org@glossarytitle
5230 \else
5231 \expandafter\let\csname @glo@type@\@glo@type @title\endcsname
5232 \glossarytitle
5233 \fi

Allow a high-level user command to indicate the current glossary
5234 \let\currentglossary\@glo@type

Enable individual number lists to be suppressed.
5235 \let\org@glossaryentrynumbers\glossaryentrynumbers
5236 \let\glsnonextpages\glsnonextpages

Enable individual number list to be activated:
5237 \let\glsnextpages\glsnextpages

Enable suppression of description terminators.
5238 \let\nopostdesc\@nopostdesc

Set up the entry for the TOC
5239 \gls@dotocitle

Set the glossary style
5240 \@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):
5241 \let\gls@org@glossaryentryfield\glossentry
5242 \let\gls@org@glossarysubentryfield\subglossentry
5243 \renewcommand{\glossentry}[1]{%
5244 \xdef\glscurrententrylabel{\glsdetoklabel{##1}}%
5245 \gls@org@glossaryentryfield{##1}%

```

```

5246 }%
5247 \renewcommand{\subglossentry}[2]{%
5248     \xdef\glscurrententrylabel{\glsdetoklabel{##2}}%
5249     \gls@org@glossarysubentryfield{##1}{##2}%
5250 }%

5251 \@gls@preglossaryhook

```

Now do the handler macro that deals with the actual glossary:

```

5252 #2%

    End the current scope
5253 \egroup

    Reset \glossaryentrynumbers
5254 \global\let\glossaryentrynumbers\@org@glossaryentrynumbers

    Suppress warning about no \printglossary
5255 \global\let\warn@noprntglossary\relax
5256 }

```

@print@glossary Internal workings of \printglossary dealing with reading the external file.

```

5257 \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.)
5258 \makeatletter

    Input the glossary file, if it exists.
5259 \@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 docu-
    ment isn't complete, so it shouldn't matter.
5260 \IfFileExists{\jobname.\csname @glo@type@\@glo@type @in\endcsname}%
5261 {}%
5262 {\null}%

    If xindy is being used, need to write the language dependent information to the .aux file for
    makeglossaries.
5263 \ifglxindy
5264     \ifcsundef{@xdy@\@glo@type @language}%
5265     {%
5266         \edef\@do@auxoutstuff{%
5267             \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.
5268                 \noexpand\immediate\noexpand\write\@auxout{%
5269                     \string\providecommand\string\@xdy@language[2]{}}%
5270                 \noexpand\immediate\noexpand\write\@auxout{%
5271                     \string\@xdy@language{\@glo@type}{\@xdy@main@language}}%

```

```

5272     }%
5273   }%
5274 }%
5275 {%
5276   \edef\@do@auxoutstuff{%
5277     \noexpand\AtEndDocument{%
5278       \noexpand\immediate\noexpand\write\@auxout{%
5279         \string\providecommand\string\@xdylanguage[2]{}%
5280       \noexpand\immediate\noexpand\write\@auxout{%
5281         \string\@xdylanguage{\@glo@type}{\csname @xdy@\@glo@type
5282           @language\endcsname}}%
5283     }%
5284   }%
5285 }%
5286 \do@auxoutstuff
5287 \edef\@do@auxoutstuff{%
5288   \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.

```

5289     \noexpand\immediate\noexpand\write\@auxout{%
5290       \string\providecommand\string\@gls@codepage[2]{}%
5291     \noexpand\immediate\noexpand\write\@auxout{%
5292       \string\@gls@codepage{\@glo@type}{\gls@codepage}}%
5293   }%
5294 }%
5295 \do@auxoutstuff
5296 \fi

```

Activate warning if `\makeglossaries` hasn't been used.

```

5297 \renewcommand*\@warn@nomakeglossaries{%
5298   \GlossariesWarningNoLine{\string\makeglossaries\space
5299     hasn't been used,^^Jthe glossaries will not be updated}%
5300 }%
5301 }

```

The sort macros all have the syntax:

```
\@glo@sortmacro@⟨order⟩{⟨type⟩}
```

where `⟨order⟩` is the sort order as specified by the sort key and `⟨type⟩` is the glossary type. (The referenced entry list is stored in `\@glsref@⟨type⟩`. The actual sorting is done by `\@glo@sortentries{⟨handler⟩}{⟨type⟩}`.)

`glo@sortentries`

```

5302 \newcommand*\@glo@sortentries[2]{%
5303   \def\@glo@sortinglist{}%
5304   \def\@glo@sortinghandler{#1}%
5305   \edef\@glo@type{#2}%

```

```

5306 \forlistcsloop{\@glo@do@sortentries}{@glsref@#2}%
5307 \csdef{@glsref@#2}{}%
5308 \@for\@this@label:=\@glo@sortinglist\do{%
  Has this entry already been added?
5309   \xifinlistcs{\@this@label}{@glsref@#2}%
5310   {}%
5311   {%
5312     \listcsxadd{@glsref@#2}{\@this@label}%
5313     }%
5314     \ifcsdef{@glo@sortingchildren@\@this@label}%
5315     {%
5316       \@glo@addchildren{#2}{\@this@label}%
5317       }%
5318     {}%
5319   }%
5320 }

```

```

@glo@addchildren \@glo@addchildren{<type>}{<parent>}

```

```

5321 \newcommand*{\@glo@addchildren}[2]{%
  Scope to allow nesting.
5322   \bgroup
5323   \letcs{\@glo@childlist}{@glo@sortingchildren@#2}%
5324   \@for\@this@childlabel:=\@glo@childlist\do
5325   {%
    Check this label hasn't already been added.
5326     \xifinlistcs{\@this@childlabel}{@glsref@#1}%
5327     {}%
5328     {%
5329       \listcsxadd{@glsref@#1}{\@this@childlabel}%
5330       }%
    Does this child have children?
5331     \ifcsdef{@glo@sortingchildren@\@this@childlabel}%
5332     {%
5333       \@glo@addchildren{#1}{\@this@childlabel}%
5334       }%
5335     {}%
5336     }%
5337   }%
5338 \egroup
5339 }

```

```

@do@sortentries
5340 \newcommand*{\@glo@do@sortentries}[1]{%
5341   \ifglshasparent{#1}%
5342   {%

```

This entry has a parent, so add it to the child list

```
5343 \edef\@glo@parent{\csuse{glo@glstdetoklabel{#1}@parent}}%
5344 \ifcsundef{glo@sortingchildren@\@glo@parent}%
5345 {%
5346 \csdef{glo@sortingchildren@\@glo@parent}{}%
5347 }%
5348 {}%
5349 \expandafter\@glo@sortedinsert
5350 \csname @glo@sortingchildren@\@glo@parent\endcsname{#1}%
```

Has the parent been added?

```
5351 \xifinlistcs{\@glo@parent}{@gl@sref@\@glo@type}%
5352 {%
```

Yes, it has so do nothing.

```
5353 }%
5354 {%
```

No, it hasn't so add it now.

```
5355 \expandafter\@glo@do@sortentries\expandafter{\@glo@parent}%
5356 }%
5357 }%
5358 {%
5359 \@glo@sortedinsert{\@glo@sortinglist}{#1}%
5360 }%
5361 }
```

glo@sortedinsert

```
\@glo@sortedinsert{<list>}{<entry label>}
```

Insert into list.

```
5362 \newcommand*{\@glo@sortedinsert}[2]{%
5363 \dtl@insertinto{#2}{#1}{\@glo@sortinghandler}%
5364 }%
```

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

```
5365 \newcommand*{\@glo@sorthandler@word}[2]{%
5366 \letcs\@gls@sort@A{glo@glstdetoklabel{#1}@sort}%
5367 \letcs\@gls@sort@B{glo@glstdetoklabel{#2}@sort}%
5368 \edef\@glo@do@compare{%
5369 \noexpand\dtlwordindexcompare{\noexpand\dtl@sortresult}%
5370 {\expandonce\@gls@sort@B}%
5371 {\expandonce\@gls@sort@A}%
5372 }%
5373 \@glo@do@compare
5374 }
```

thandler@letter

```
5375 \newcommand*{\@glo@sorthandler@letter}[2]{%
5376   \letcs\@gls@sort@A{glo\glsdetoklabel{#1}@sort}%
5377   \letcs\@gls@sort@B{glo\glsdetoklabel{#2}@sort}%
5378   \edef\glo@do@compare{%
5379     \noexpand\dtlletterindexcompare{\noexpand\dtl@sortresult}%
5380     {\expandonce\@gls@sort@B}%
5381     {\expandonce\@gls@sort@A}%
5382   }%
5383   \glo@do@compare
5384 }
```

orthandler@case Case-sensitive sort.

```
5385 \newcommand*{\@glo@sorthandler@case}[2]{%
5386   \letcs\@gls@sort@A{glo\glsdetoklabel{#1}@sort}%
5387   \letcs\@gls@sort@B{glo\glsdetoklabel{#2}@sort}%
5388   \edef\glo@do@compare{%
5389     \noexpand\dtlcompare{\noexpand\dtl@sortresult}%
5390     {\expandonce\@gls@sort@B}%
5391     {\expandonce\@gls@sort@A}%
5392   }%
5393   \glo@do@compare
5394 }
```

thandler@nocase Case-insensitive sort.

```
5395 \newcommand*{\@glo@sorthandler@nocase}[2]{%
5396   \letcs\@gls@sort@A{glo\glsdetoklabel{#1}@sort}%
5397   \letcs\@gls@sort@B{glo\glsdetoklabel{#2}@sort}%
5398   \edef\glo@do@compare{%
5399     \noexpand\dtlcompare{\noexpand\dtl@sortresult}%
5400     {\expandonce\@gls@sort@B}%
5401     {\expandonce\@gls@sort@A}%
5402   }%
5403   \glo@do@compare
5404 }
```

@sortmacro@word Sort macro for 'word'

```
5405 \newcommand*{\@glo@sortmacro@word}[1]{%
5406   \ifdefstring{\@glo@default@sorttype}{standard}%
5407   {%
5408     \@glo@sortentries{\@glo@sorthandler@word}{#1}%
5409   }%
5410   {%
5411     \PackageError{glossaries}{Conflicting sort options:^^J
5412       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5413       \string\printnoidxglossary[sort=word]}{}%
5414   }%
5415 }
```

ortmacro@letter Sort macro for 'letter'

```
5416 \newcommand*{\@glo@sortmacro@letter}[1]{%
5417   \ifdefstring{\@glo@default@sorttype}{standard}%
5418   {%
5419     \@glo@sortentries{\@glo@sorthandler@letter}{#1}%
5420   }%
5421   {%
5422     \PackageError{glossaries}{Conflicting sort options:^^J
5423       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5424       \string\printnoidxglossary[sort=letter]}{}}%
5425   }%
5426 }
```

tmacro@standard Sort macro for 'standard'. (Use either 'word' or 'letter' order.)

```
5427 \newcommand*{\@glo@sortmacro@standard}[1]{%
5428   \ifdefstring{\@glo@default@sorttype}{standard}%
5429   {%
5430     \ifcsdef{@glo@sorthandler@\glsorder}%
5431     {%
5432       \@glo@sortentries{\csuse{@glo@sorthandler@\glsorder}}{#1}%
5433     }%
5434     {%
5435       \PackageError{glossaries}{Unknown sort handler '\glsorder'}{}}%
5436     }%
5437   }%
5438   {%
5439     \PackageError{glossaries}{Conflicting sort options:^^J
5440       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5441       \string\printnoidxglossary[sort=standard]}{}}%
5442   }%
5443 }
```

@sortmacro@case Sort macro for 'case'

```
5444 \newcommand*{\@glo@sortmacro@case}[1]{%
5445   \ifdefstring{\@glo@default@sorttype}{standard}%
5446   {%
5447     \@glo@sortentries{\@glo@sorthandler@case}{#1}%
5448   }%
5449   {%
5450     \PackageError{glossaries}{Conflicting sort options:^^J
5451       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5452       \string\printnoidxglossary[sort=case]}{}}%
5453   }%
5454 }
```

ortmacro@nocase Sort macro for 'nocase'

```
5455 \newcommand*{\@glo@sortmacro@nocase}[1]{%
5456   \ifdefstring{\@glo@default@sorttype}{standard}%
5457   {%
```

```

5458   \@glo@sortentries{\@glo@sorthandler@nocase}{#1}%
5459 }%
5460 {%
5461   \PackageError{glossaries}{Conflicting sort options:^^J
5462     \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5463     \string\printnoidxglossary[sort=nocase]}{}}%
5464 }%
5465 }

```

`\sortmacro@def` Sort macro for ‘def’. The order of definition is given in `\glo@list@<type>`.

```

5466 \newcommand*\@glo@sortmacro@def}[1]{%
5467   \def\@glo@sortinglist{}%
5468   \for@gl@entries[#1]{\@gl@thislabel}%
5469   {%
5470     \xifinlistcs{\@gl@thislabel}{\@gl@sref@#1}%
5471     {%
5472       \listeadadd{\@glo@sortinglist}{\@gl@thislabel}%
5473     }%
5474     {%
5475       Hasn't been referenced.
5476     }%
5477     \cslet{\@gl@sref@#1}{\@glo@sortinglist}%
5478   }

```

`\sortmacro@def@do` This won't include parent entries that haven't been referenced.

```

5479 \newcommand*\@glo@sortmacro@def@do}[1]{%
5480   \ifinlistcs{#1}{\@gl@sref@\@glo@type}%
5481   {}%
5482   {%
5483     \listcsadd{\@gl@sref@\@glo@type}{#1}%
5484   }%
5485   \ifcsdef{\@glo@sortingchildren@#1}%
5486   {%
5487     \@glo@addchildren{\@glo@type}{#1}%
5488   }%
5489   {}%
5490 }

```

`\sortmacro@use` Sort macro for ‘use’. (No sorting is required, as the entries are already in order of use, so do nothing.)

```

5491 \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.


```

5492 \newcommand*{\@print@noidx@glossary}{%
5493   \ifcsdef{@glsref@\@glo@type}%
5494   {%
      Sort the entries:
5495     \ifcsdef{@glo@sortmacro@\@glo@sorttype}%
5496     {%
5497       \csuse{@glo@sortmacro@\@glo@sorttype}{\@glo@type}%
5498     }%
5499     {%
5500       \PackageError{glossaries}{Unknown sort handler ‘\@glo@sorttype’}{}%
5501     }%

```

Do the glossary heading and preamble

```

5502   \glossarysection[\glossarytoctitle]{\glossarytitle}%
5503   \glossarypreamble
5504   \begin{theglossary}%
5505   \glossaryheader
5506   \glsresetentrylist
5507   \def\@gls@currentlettergroup{}%

```

Iterate through the entries.

```

5508   \forlistcsloop{\@gls@noidx@do}{@glsref@\@glo@type}%

```

Finally end the glossary and do the postamble:

```

5509   \end{theglossary}%
5510   \glossarypostamble
5511 }%
5512 {%
5513   \@gls@noref@warn{\@glo@type}%
5514 }%
5515 }

```

\glo@grabfirst

```

5516 \def\glo@grabfirst#1#2\@nil{%
5517   \def\@gls@firsttok{#1}%
5518   \ifdefempty\@gls@firsttok
5519   {%
5520     \def\@glo@thislettergrp{0}%
5521   }%
5522   {%

```

Sanitize it:

```

5523   \@onelevel@sanitize\@gls@firsttok

```

Fetch the first letter:

```

5524   \expandafter\@glo@grabfirst\@gls@firsttok{}{}\@nil
5525 }%
5526 }

```

\@glo@grabfirst

```

5527 \def\@glo@grabfirst#1#2\@nil{%
5528   \ifdefempty\@glo@thislettergrp
5529   {%
5530     \def\@glo@thislettergrp{glssymbols}%
5531   }%
5532   {%
5533     \count@=\uccode'#1\relax
5534     \ifnum\count@=0\relax
5535       \def\@glo@thislettergrp{glssymbols}%
5536     \else
5537       \ifdefstring\@glo@sorttype{case}%
5538       {%
5539         \count@='#1\relax
5540       }%
5541     }%
5542   }%
5543   \edef\@glo@thislettergrp{\the\count@}%
5544 \fi
5545 }%
5546 }

```

`\@gls@noidx@do` Handler for list iteration used by `\@print@noidx@glossary`. The argument is the entry label. This only allows one sublevel.

```

5547 \newcommand{\@gls@noidx@do}[1]{%
  Get this entry's location list
5548   \global\letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
  Does this entry have a parent?
5549   \ifglshasparent{#1}%
5550   {%
  Has a parent.
5551     \gls@level=\csuse{glo@\glsdetoklabel{#1}@level}\relax
5552     \ifdefvoid{\@gls@loclist}
5553     {%
5554       \subglossentry{\@gls@level}{#1}{%
5555       }%
5556     }%
5557     \subglossentry{\@gls@level}{#1}%
5558     {%
5559       \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
5560     }%
5561   }%
5562 }%
5563 }%

```

Doesn't have a parent Get this entry's sort key

```

5564   \letcs{\@gls@sort}{glo@\glsdetoklabel{#1}@sort}%

```

Fetch the first letter:

```

5565 \expandafter\glo@grabfirst\@gls@sort{}{}\@nil
5566 \ifdefequal{\@glo@thislettergrp}{\@gls@currentlettergroup}%
5567 {}%
5568 {%

```

Do the group header:

```

5569 \ifdefempty{\@gls@currentlettergroup}{\@gls@currentlettergroup}%
5570 \gls@groupheading{\@glo@thislettergrp}%
5571 }%
5572 \let\@gls@currentlettergroup\@glo@thislettergrp

```

Do this entry:

```

5573 \ifdefvoid{\@gls@loclist}
5574 {%
5575 \glossentry{#1}{}%
5576 }%
5577 {%
5578 \glossentry{#1}%
5579 {%
5580 \glossaryentrynumbers{\@gls@loclist}%
5581 }%
5582 }%
5583 }%
5584 }

```

`\glsnoidxloclist` `\glsnoidxloclist{<list cs>}`

Display location list.

```

5585 \newcommand*{\glsnoidxloclist}[1]{%
5586 \def\@gls@noidxloclist@sep{}%
5587 \def\@gls@noidxloclist@prev{}%
5588 \forlistloop{\glsnoidxloclist@handler}{#1}%
5589 }

```

`xloclisthandler` Handler for location list iterator.

```

5590 \newcommand*{\glsnoidxloclist@handler}[1]{%
5591 \ifdefstring{\@gls@noidxloclist@prev}{#1}%
5592 {%

```

Same as previous location so skip.

```

5593 }%
5594 {%
5595 \@gls@noidxloclist@sep
5596 #1%
5597 \def\@gls@noidxloclist@sep{\delimN}%
5598 \def\@gls@noidxloclist@prev{#1}%
5599 }%
5600 }

```

`gloclisthandler` Handler for location list iterator when used with `\glsdisplaynumberlist`.

```
5601 \newcommand*\glsnoidxdisplaygloclisthandler}[1]{%
5602   \ifdefstring{\@gls@noidxgloclist@prev}{#1}%
5603   {%
      Same as previous location so skip.
5604   }%
5605   {%
5606     \@gls@noidxgloclist@sep
5607     \@gls@noidxgloclist@prev
5608     \def\@gls@noidxgloclist@prev{#1}%
5609   }%
5610 }
```

`glsnoidxdisplayloc` `\glsnoidxdisplayloc{<prefix>}{<counter>}{<format>}{<location>}`

Display a location in the location list.

```
5611 \newcommand*\glsnoidxdisplayloc[4]{%
5612   \setentrycounter[#1]{#2}%
5613   \csuse{#3}{#4}%
5614 }
```

`@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.

```
5615 \newcommand*\@gls@reference}[3]{%
```

Add to label list

```
5616   \glsdoifexistsorwarn{#2}%
5617   {%
5618     \ifcsundef{@glsref@#1}{\csgdef{@glsref@#1}{}}{}%
5619     \ifinlistcs{#2}{@glsref@#1}%
5620     {}%
5621     {\listcsgadd{@glsref@#1}{#2}}%
5622 }
```

Add to location list

```
5622   \ifcsundef{glo@glsdetoklabel{#2}@loclist}%
5623   {\csgdef{glo@glsdetoklabel{#2}@loclist}{}}%
5624   {}%
5625   \listcsgadd{glo@glsdetoklabel{#2}@loclist}{#3}%
5626   }%
5627 }
```

The keys that can be used in the optional argument to `\printglossary` or `\printnoidxglossary` are as follows: The type key sets the glossary type.

```
5628 \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`.

```
5629 \define@key{printgloss}{title}{%
5630 \def\glossarytitle{#1}%
5631 \let\gls@dotoc\title\relax
5632 }
```

The toctitle sets the text used for the relevant entry in the table of contents.

```
5633 \define@key{printgloss}{toctitle}{%
5634 \def\glossarytoctitle{#1}%
5635 \let\gls@dotoc\title\relax
5636 }
```

The style key sets the glossary style (but only for the given glossary).

```
5637 \define@key{printgloss}{style}{%
5638 \ifcsundef{@glsstyle@#1}%
5639 {%
5640 \PackageError{glossaries}%
5641 {Glossary style ‘#1’ undefined}{}%
5642 }%
5643 {%
5644 \def\@glossarystyle{\setglossentrycompatibility
5645 \csname @glsstyle@#1\endcsname}%
5646 }%
5647 }
```

The numberedsection key determines if this glossary should be in a numbered section.

```
5648 \define@choicekey{printgloss}{numberedsection}[\val\nr]{%
5649 false,nolabel,autolabel,nameref}[nolabel]{%
5650 \ifcase\nr\relax
5651 \renewcommand*{\@glossarysecstar}{*}%
5652 \renewcommand*{\@glossaryseclabel}{}%
5653 \or
5654 \renewcommand*{\@glossarysecstar}{}%
5655 \renewcommand*{\@glossaryseclabel}{}%
5656 \or
5657 \renewcommand*{\@glossarysecstar}{}%
5658 \renewcommand*{\@glossaryseclabel}{\label{\glsautoprefix\@glo@type}}%
5659 \or
5660 \renewcommand*{\@glossarysecstar}{*}%
5661 \renewcommand*{\@glossaryseclabel}{%
5662 \protected@edef\@currentlabelname{\glossarytoctitle}%
5663 \label{\glsautoprefix\@glo@type}}%
5664 \fi
5665 }
```

The nogroupskip key determines whether or not there should be a vertical gap between glossary groups.

```
5666 \define@choicekey{printgloss}{nogroupskip}{true,false}[true]{%
5667 \csuse{glsnogroupskip#1}%
5668 }
```

The nopostdot key has the same effect as the package option of the same name.

```
5669 \define@choicekey{printgloss}{nopostdot}{true,false}[true]{%
5670   \csuse{glsnopostdot#1}%
5671 }
```

The entrycounter key is the same as the package option but localised to the current glossary.

```
5672 \define@choicekey{printgloss}{entrycounter}{true,false}[true]{%
5673   \csuse{glentrycounter#1}%
5674   \ifglentrycounter
5675     \ifx\@gls@counterwithin\@empty
5676       \newcounter{glossaryentry}%
5677     \else
5678       \newcounter{glossaryentry}[\@gls@counterwithin]%
5679     \fi
5680     \def\theHglossaryentry{\currentglossary.\theglossaryentry}%
5681     \renewcommand*\glsresetentrycounter{%
5682       \setcounter{glossaryentry}{0}%
5683     }%
5684     \renewcommand*\glsstepentry}[1]{%
5685       \refstepcounter{glossaryentry}%
5686       \label{glsentry-\glsdetoklabel{##1}}%
5687     }%
5688     \renewcommand*\glsentrycounterlabel{\theglossaryentry.\space}%
5689     \renewcommand*\glsentryitem}[1]{%
5690       \glsstepentry{##1}\glsentrycounterlabel
5691     }%
5692   \else
5693     \renewcommand*\glsresetentrycounter{}%
5694     \renewcommand*\glsstepentry}[1]{}%
5695     \renewcommand*\glsentrycounterlabel{}%
5696     \renewcommand*\glsentryitem}[1]{\glsresetsubentrycounter}
5697   \fi
5698 }
```

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.

```
5699 \define@choicekey{printgloss}{subentrycounter}{true,false}[true]{%
5700   \csuse{glssubentrycounter#1}%
5701   \ifglssubentrycounter
5702     \ifundef\c@glossarysubentry
5703     {%
5704       \ifglentrycounter
5705         \newcounter{glossarysubentry}[glossaryentry]%
5706       \else
5707         \newcounter{glossarysubentry}
5708       \fi
5709     }{}%
5710     \renewcommand*\glsstepsubentry}[1]{%
5711       \edef\currentglssubentry{\glsdetoklabel{##1}}%
```

```

5712     \refstepcounter{glossarysubentry}%
5713     \label{glsentry-\currentglssubentry}%
5714 }%
5715 \renewcommand*{\glsresetsubentrycounter}{%
5716     \setcounter{glossarysubentry}{0}%
5717 }%
5718 \renewcommand*{\glssubentryitem}[1]{%
5719     \glsstepsubentry{##1}\glssubentrycounterlabel
5720 }%
5721 \renewcommand*{\glssubentrycounterlabel}{\theglossarysubentry\space}%
5722 \def\theHglossarysubentry{\currentglssubentry.\theglossarysubentry}
5723 \else
5724 \renewcommand*{\glssubentryitem}[1]{}%
5725 \renewcommand*{\glsstepsubentry}[1]{}%
5726 \renewcommand*{\glsresetsubentrycounter}{}%
5727 \renewcommand*{\glssubentrycounterlabel}{}%
5728 \fi
5729 }

```

The nonumberlist key determines if this glossary should have a number list.

```

5730 \define@boolkey{printgloss}[gls]{nonumberlist}[true]{%
5731 \ifglsnonumberlist
5732     \def\glossaryentrynumbers##1{}%
5733 \else
5734     \def\glossaryentrynumbers##1{##1}%
5735 \fi}

```

The sort key sets the glossary sort handler (`\printnoidxglossary` only).

```

5736 \define@key{printgloss}{sort}{\@glo@assign@sortkey{##1}}

```

`@assign@sortkey` Issue error if used with `\printglossary`

```

5737 \newcommand*{\@glo@no@assign@sortkey}[1]{%
5738     \PackageError{glossaries}{‘sort’ key not permitted with
5739     \string\printglossary}%
5740     {The ‘sort’ key may only be used with \string\printnoidxglossary}%
5741 }

```

`@assign@sortkey` For use with `\printnoidxglossary`

```

5742 \newcommand*{\@glo@assign@sortkey}[1]{%
5743     \def\@glo@sorttype{##1}%
5744 }

```

`@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.

```

5745 \newcommand*{\@glsnonextpages}{%
5746     \gdef\glossaryentrynumbers##1{

```

```

5747   \glsresetentrylist
5748 }%
5749 }

```

`\@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.

```

5750 \newcommand*\@glsnextpages}{%
5751   \gdef\glossaryentrynumbers##1{%
5752     ##1\glsresetentrylist}}

```

`\glsresetentrylist` Resets `\glossaryentrynumbers`

```

5753 \newcommand*\glsresetentrylist}{%
5754   \global\let\glossaryentrynumbers\org@glossaryentrynumbers}

```

`\glsnonextpages` Outside of `\printglossary` this does nothing.

```

5755 \newcommand*\glsnonextpages}{}

```

`\glsnextpages` Outside of `\printglossary` this does nothing.

```

5756 \newcommand*\glsnextpages}{}

```

`\glossaryentry` If the `entrycounter` package option has been used, define a counter to number each level 0 entry.

```

5757 \ifgl Sentrycounter
5758   \ifx\@gls@counterwithin\@empty
5759     \newcounter{glossaryentry}
5760   \else
5761     \newcounter{glossaryentry}[\@gls@counterwithin]
5762   \fi
5763   \def\theHglossaryentry{\currentglossary.\theglossaryentry}
5764 \fi

```

`\glossarysubentry` If the `subentrycounter` package option has been used, define a counter to number each level 1 entry.

```

5765 \ifgl ssubentrycounter
5766   \ifgl Sentrycounter
5767     \newcounter{glossarysubentry}[glossaryentry]
5768   \else
5769     \newcounter{glossarysubentry}
5770   \fi
5771   \def\theHglossarysubentry{\currentgl ssubentry.\theglossarysubentry}
5772 \fi

```

`\glossarysubentrycounter` Resets the `\glossarysubentry` counter.

```

5773 \ifgl ssubentrycounter

```



```

5774 \newcommand*\glsresetsubentrycounter}{%
5775   \setcounter{glossarysubentry}{0}%
5776 }
5777 \else
5778 \newcommand*\glsresetsubentrycounter}{%
5779 \fi

```

`subentrycounter` Resets the glossaryentry counter.

```

5780 \ifglsentrycounter
5781 \newcommand*\glsresetentrycounter}{%
5782   \setcounter{glossaryentry}{0}%
5783 }
5784 \else
5785 \newcommand*\glsresetentrycounter}{%
5786 \fi

```

`\glsstepentry` Advance the glossaryentry counter if in use. The argument is the label associated with the entry.

```

5787 \ifglsentrycounter
5788 \newcommand*\glsstepentry}[1]{%
5789   \refstepcounter{glossaryentry}%
5790   \label{glsentry-\glsdetoklabel{#1}}%
5791 }
5792 \else
5793 \newcommand*\glsstepentry}[1]{%
5794 \fi

```

`glsstepsubentry` Advance the glossarysubentry counter if in use. The argument is the label associated with the subentry.

```

5795 \ifglsentrycounter
5796 \newcommand*\glsstepsubentry}[1]{%
5797   \edef\currentglsstepsubentry{\glsdetoklabel{#1}}%
5798   \refstepcounter{glossarysubentry}%
5799   \label{glsentry-\currentglsstepsubentry}%
5800 }
5801 \else
5802 \newcommand*\glsstepsubentry}[1]{%
5803 \fi

```

`\glsrefentry` Reference the entry or sub-entry counter if in use, otherwise just do `\gls`.

```

5804 \ifglsentrycounter
5805 \newcommand*\glsrefentry}[1]{\ref{glsentry-\glsdetoklabel{#1}}}
5806 \else
5807 \ifglsstepsubentrycounter
5808 \newcommand*\glsrefentry}[1]{\ref{glsentry-\glsdetoklabel{#1}}}
5809 \else
5810 \newcommand*\glsrefentry}[1]{\gls{#1}}
5811 \fi
5812 \fi

```

trycounterlabel Defines how to display the glossaryentry counter.

```
5813 \ifglentrycounter
5814 \newcommand*{\glentrycounterlabel}{\theglossaryentry.\space}
5815 \else
5816 \newcommand*{\glentrycounterlabel}{}
5817 \fi
```

trycounterlabel Defines how to display the glossarysubentry counter.

```
5818 \ifglssubentrycounter
5819 \newcommand*{\glssubentrycounterlabel}{\theglossarysubentry}\space}
5820 \else
5821 \newcommand*{\glssubentrycounterlabel}{}
5822 \fi
```

\glentryitem Step and display glossaryentry counter, if appropriate.

```
5823 \ifglentrycounter
5824 \newcommand*{\glentryitem}[1]{%
5825 \glstepentry{#1}\glentrycounterlabel
5826 }
5827 \else
5828 \newcommand*{\glentryitem}[1]{\glresetsubentrycounter}
5829 \fi
```

glssubentryitem Step and display glossarysubentry counter, if appropriate.

```
5830 \ifglssubentrycounter
5831 \newcommand*{\glssubentryitem}[1]{%
5832 \glstepsubentry{#1}\glssubentrycounterlabel
5833 }
5834 \else
5835 \newcommand*{\glssubentryitem}[1]{}
5836 \fi
```

theglossary If the theglossary environment has already been defined, a warning will be issued. This environment should be redefined by glossary styles.

```
5837 \ifcsundef{theglossary}%
5838 {%
5839 \newenvironment{theglossary}{}{}%
5840 }%
5841 {%
5842 \@gls@warnontheglossdefined
5843 \renewenvironment{theglossary}{}{}%
5844 }
```

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`

```
5845 \newcommand*\glossaryheader{}
```

`\glstarget` `\glstarget{<label>}{<name>}`

Provide user interface to `\glstarget` to make it easier to modify the glossary style in the document.

```
5846 \newcommand*\glstarget[2]{\@glstarget{\glo@linkprefix#1}{#2}}
```

As from version 3.08, glossary information is now written to the external files using `\glossentry` and `\subglossentry` instead of `\glossaryentryfield` and `\glossarysubentryfield`. The default definition provides backward compatibility for glossary styles that use the old forms.

`\compatibleglossentry`

```
\glossentry{<label>}{<page-list>}
```

```
5847 \providecommand*\compatibleglossentry[2]{%
5848   \toks@{#2}%
5849   \protected@edef\@do@glossentry{\noexpand\glossaryentryfield{#1}%
5850     {\noexpand\glsnamefont
5851       {\expandafter\expandonce\csname glo@#1@name\endcsname}}}%
5852     {\expandafter\expandonce\csname glo@#1@desc\endcsname}%
5853     {\expandafter\expandonce\csname glo@#1@symbol\endcsname}%
5854     {\the\toks@}}%
5855   }%
5856   \@do@glossentry
5857 }
```

`\glossentryname`

```
5858 \newcommand*\glossentryname[1]{%
5859   \glsdoifexistsorwarn{#1}%
5860   {%
5861     \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
5862     \expandafter\glsnamefont\expandafter{\glo@name}%
5863   }%
5864 }
```

`\Glossentryname`

```
5865 \newcommand*\Glossentryname[1]{%
5866   \glsdoifexistsorwarn{#1}%
5867   {%
5868     \glsnamefont{\Glsentryname{#1}}%
5869   }%
5870 }
```

`\glossentrydesc`

```
5871 \newcommand*\glossentrydesc[1]{%
```

```

5872 \glsdoifexistsorwarn{#1}%
5873 {%
5874     \glsentrydesc{#1}%
5875 }%
5876 }

```

\Glossentrydesc

```

5877 \newcommand*{\Glossentrydesc}[1]{%
5878     \glsdoifexistsorwarn{#1}%
5879     {%
5880         \Glsentrydesc{#1}%
5881     }%
5882 }

```

lossentrysymbol

```

5883 \newcommand*{\glossentrysymbol}[1]{%
5884     \glsdoifexistsorwarn{#1}%
5885     {%
5886         \glsentrysymbol{#1}%
5887     }%
5888 }

```

lossentrysymbol

```

5889 \newcommand*{\Glossentrysymbol}[1]{%
5890     \glsdoifexistsorwarn{#1}%
5891     {%
5892         \Glsentrysymbol{#1}%
5893     }%
5894 }

```

blesubglossentry

```
\subglossentry{<level>}{<label>}{<page-list>}
```

```

5895 \providecommand*{\compatiblesubglossentry}[3]{%
5896     \toks@{#3}%
5897     \protected@edef\@do@subglossentry{\noexpand\glossarysubentryfield{\number#1}%
5898         {#2}%
5899         {\noexpand\glsnamefont
5900             {\expandafter\expandonce\csname glo@#2@name\endcsname}}%
5901         {\expandafter\expandonce\csname glo@#2@desc\endcsname}%
5902         {\expandafter\expandonce\csname glo@#2@symbol\endcsname}%
5903         {\the\toks@}}%
5904     }%
5905     \@do@subglossentry
5906 }

```

rycompatibility

```
5907 \newcommand*{\setglossentrycompatibility}{%
```

```

5908 \let\glossentry\compatibleglossentry
5909 \let\subglossentry\compatiblesubglossentry
5910 }
5911 \setglossentrycompatibility

```

glossaryentryfield

```
\glossaryentryfield{<label>}{<name>}{<description>}{<symbol>}{<page-list>}
```

This command formerly governed how each entry row should be formatted in the glossary. Now deprecated.

```

5912 \newcommand{\glossaryentryfield}[5]{%
5913   \GlossariesWarning
5914   {Deprecated use of \string\glossaryentryfield.^^J
5915     I recommend you change to \string\glossentry.^^J
5916     If you've just upgraded, try removing your gls auxiliary
5917     files^^J and recompile}%
5918   \noindent\textbf{\glstarget{#1}{#2}} #4 #3. #5\par}

```

glossarysubentryfield

```
\glossarysubentryfield{<level>}{<label>}{<name>}{<description>}{<symbol>}{<page-list>}
```

This command governs how each subentry should be formatted in the glossary. Glossary styles need to redefine this command. Most of the predefined styles ignore *<symbol>*. The first argument is a number indicating the level. (The level should be greater than or equal to 1.)

```

5919 \newcommand*{\glossarysubentryfield}[6]{%
5920   \GlossariesWarning
5921   {Deprecated use of \string\glossarysubentryfield.^^J
5922     I recommend you change to \string\subglossentry.^^J
5923     If you've just upgraded, try removing your gls auxiliary
5924     files^^J and recompile}%
5925   \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`

```
5926 \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`

```
5927 \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 an `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 `\glssetgrouplabel` 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.

`glsgetgrouptitle`

```
5928 \newcommand*{\glsgetgrouptitle}[1]{%
5929   \@gls@getgrouptitle{#1}{\@gls@grptitle}%
5930   \@gls@grptitle
5931 }
```

`gls@getgrouptitle`

Gets the group title specified by the label (first argument) and stores in the second argument, which must be a control sequence.

```
5932 \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.

```
5933   \dtl@ifsingle{#1}%
5934   {%
5935     \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
5936   }%
5937   {%
5938     \ifboolexpr{test{\ifstrequal{#1}{glssymbols}}
5939               or test{\ifstrequal{#1}{glsnumbers}}}%
5940     {%
5941       \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
5942     }%
5943     {%
```

```

5944     \def#2{#1}%
5945   }%
5946 }%
5947 }

```

othergrouptitle Version for the no-indexing app option:

```

5948 \newcommand*{\@gls@noidx@getgrouptitle}[2]{%
5949   \DTLifint{#1}%
5950   {\edef#2{\char#1\relax}}%
5951   {%
5952     \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
5953   }%
5954 }

```

`\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

```

5955 \newcommand*{\glsgetgrouplabel}[1]{%
5956 \ifthenelse{\equal{#1}{\glssymbolsgroupname}}{\glssymbols}{%
5957 \ifthenelse{\equal{#1}{\glsnumbersgroupname}}{\glsnumbers}{#1}}

```

The command `\setentrycounter` sets the entry's associated counter (required by `\glsnumber` etc.) `\glslink` and `\glsadd` encode the `\glossary` argument so that the relevant counter is set prior to the formatting command.

setentrycounter

```

5958 \newcommand*{\setentrycounter}[2] [] {%
5959   \def\@glo@counterprefix{#1}%
5960   \ifx\@glo@counterprefix\@empty
5961     \def\@glo@counterprefix{.}%
5962   \else
5963     \def\@glo@counterprefix{.#1.}%
5964   \fi
5965   \def\glsentrycounter{#2}%
5966 }

```

The current glossary style can be set using `\setglossarystyle{<style>}`.

setglossarystyle

```

5967 \newcommand*{\setglossarystyle}[1]{%
5968   \ifcsundef{@glsstyle@#1}%
5969   {%
5970     \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
5971   }%

```

```

5972 {%
5973   \csname @glsstyle@#1\endcsname
5974 }%

```

Set the default style if it's not already set.

```

5975 \ifx\@glossary@default@style\relax
5976   \protected@edef\@glossary@default@style{#1}%
5977 \fi
5978 }

```

`\glossarystyle`

```

5979 \newcommand*{\glossarystyle}[1]{%
5980   \ifcsundef{@glsstyle@#1}%
5981   {%
5982     \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
5983   }%
5984   {%
5985     \GlossariesWarning
5986     {Deprecated command \string\glossarystyle.^^J
5987      I recommend you switch to \string\setglossarystyle\space unless
5988      you want to maintain backward compatibility}%
5989     \setglossentrycompatibility
5990     \csname @glsstyle@#1\endcsname

5991     \ifcsdef{@glscompstyle@#1}%
5992     {\setglossentrycompatibility\cuse{@glscompstyle@#1}}%
5993     {}%
5994   }%

```

Set the default style if it isn't already set so that `\printglossary` can warn if the fallback style is in use.

```

5995 \ifx\@glossary@default@style\relax
5996   \protected@edef\@glossary@default@style{#1}%
5997 \fi
5998 }

```

`\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 `\glossary preamble` and `\glossary postamble`, as the user should be able to switch between styles without affecting the pre- and postambles.

```

5999 \newcommand{\newglossarystyle}[2]{%
6000   \ifcsundef{@glsstyle@#1}%
6001   {%
6002     \expandafter\def\csname @glsstyle@#1\endcsname{#2}%

```



```

6003 }%
6004 {%
6005   \PackageError{glossaries}{Glossary style ‘#1’ is already defined}{}%
6006 }%
6007 }

```

`\renewglossarystyle` Code for this macro supplied by Marco Daniel.

```

6008 \newcommand{\renewglossarystyle}[2]{%
6009   \ifcsundef{@glsstyle@#1}%
6010   {%
6011     \PackageError{glossaries}{Glossary style ‘#1’ isn’t already defined}{}%
6012   }%
6013   {%
6014     \csdef{@glsstyle@#1}{#2}%
6015   }%
6016 }

```

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`

```

6017 \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 `\glsnumber`. 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.

`\glsnumber`

```

6018 \ifcsundef{hyperlink}%
6019 {%
6020   \def\glsnumber#1{#1}%
6021 }%
6022 {%
6023   \def\glsnumber#1{\@glsnumber#1\nohyperpage{}}\@nil}
6024 }

```

`@glshypernumber` This code was provided by Heiko Oberdiek to allow material to be attached to the location.

```
6025 \def\@glshypernumber#1\nohyperpage#2#3\@nil{%
6026   \ifx\#1\%
6027   \else
6028     \@delimR#1\delimR\delimR\%
6029   \fi
6030   \ifx\#2\%
6031   \else
6032     #2%
6033   \fi
6034   \ifx\#3\%
6035   \else
6036     \@glshypernumber#3\@nil
6037   \fi
6038 }
```

`\@delimR` displays a range of numbers for the counter whose name is given by `\@gls@counter` (which must be set prior to using `\glshypernumber`).

`\@delimN`

```
6039 \def\@delimR#1\delimR #2\delimR #3\{%
6040 \ifx\#2\%
6041   \@delimN{#1}%
6042 \else
6043   \@gls@numberlink{#1}\delimR\@gls@numberlink{#2}%
6044 \fi}
```

`\@delimN` displays a list of individual numbers, instead of a range:

`\@delimN`

```
6045 \def\@delimN#1{\@delimN#1\delimN \delimN\}
6046 \def\@@delimN#1\delimN #2\delimN#3\{%
6047 \ifx\#3\%
6048   \@gls@numberlink{#1}%
6049 \else
6050   \@gls@numberlink{#1}\delimN\@gls@numberlink{#2}%
6051 \fi
6052 }
```

The following code is modified from `hyperref's \HyInd@pagelink` where the name of the counter being used is given by `\@gls@counter`.

```
6053 \def\@gls@numberlink#1{%
6054 \begingroup
6055 \toks@={}%
6056 \@gls@removespaces#1 \@nil
6057 \endgroup}

6058 \def\@gls@removespaces#1 #2\@nil{%
6059 \toks@=\expandafter{\the\toks@#1}%
6060 \ifx\#2\%
```

```

6061 \edef\x{\the\toks@}%
6062 \ifx\x\empty
6063 \else

6064 \hyperlink{\glsentrycounter\glo@counterprefix\the\toks@}%
6065           {\the\toks@}%
6066 \fi
6067 \else
6068 \@gls@ReturnAfterFi{%
6069 \@gls@removespaces#2\@nil
6070 }%
6071 \fi
6072 }
6073 \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`

```
6074 \newcommand*\hyperrm[1]{\textrm{\glshypernumber{#1}}}
```

`\hypersf`

```
6075 \newcommand*\hypersf[1]{\textsf{\glshypernumber{#1}}}
```

`\hypertt`

```
6076 \newcommand*\hypertt[1]{\texttt{\glshypernumber{#1}}}
```

`\hyperbf`

```
6077 \newcommand*\hyperbf[1]{\textbf{\glshypernumber{#1}}}
```

`\hypermd`

```
6078 \newcommand*\hypermd[1]{\textmd{\glshypernumber{#1}}}
```

`\hyperit`

```
6079 \newcommand*\hyperit[1]{\textit{\glshypernumber{#1}}}
```

`\hypersl`

```
6080 \newcommand*\hypersl[1]{\textsl{\glshypernumber{#1}}}
```

`\hyperup`

```
6081 \newcommand*\hyperup[1]{\textup{\glshypernumber{#1}}}
```

`\hypersc`

```
6082 \newcommand*\hypersc[1]{\textsc{\glshypernumber{#1}}}
```

`\hyperemph`

```
6083 \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.

```
6084 \newcommand{\oldacronym}[4][\gls@label]{%
6085   \def\gls@label{#2}%
6086   \newacronym[#4]{#1}{#2}{#3}%
6087   \ifcsundef{xspace}%
6088   {%
6089     \expandafter\edef\csname#1\endcsname{%
6090       \noexpand\@ifstar{\noexpand\Gls{#1}}{\noexpand\gls{#1}}}%
6091   }%
6092 }%
6093 {%
6094   \expandafter\edef\csname#1\endcsname{%
6095     \noexpand\@ifstar{\noexpand\Gls{#1}\noexpand\xspace}{%
6096       \noexpand\gls{#1}\noexpand\xspace}%
6097   }%
6098 }%
6099 }
```

```
\newacronym[⟨key-val list⟩]{⟨label⟩}{⟨abbrv⟩}{⟨long⟩}
```

This is a quick way of defining acronyms, using `\newglossaryentry` with the appropriate values. It sets the glossary type to `\acronymtype` which will be `acronym` if the package option `acronym` has been used, otherwise it will be the default glossary. Since `\newacronym` merely calls `\newglossaryentry`, the acronym is treated like any other glossary entry.

If you prefer a different format, you can redefine `\newacronym` as required. The optional argument can be used to override any of the settings.

This is just a stub. It's redefined by commands like `\SetDefaultAcronymStyle`.

```
\newacronym
6100 \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.

```
6101 \newcommand*\acrpluralsuffix{\glsacrpluralsuffix}
```

If `garamondx` has been loaded, need to use `\textulc` instead of `\textup`.

`\glstextup`

```
6102 \newrobustcmd*\glstextup[1]{\ifdef\textulc{\textulc{#1}}{\textup{#1}}}
```

The following are defined for compatibility with version 2.07 and earlier.

`\glsshortkey`

```
6103 \newcommand*\glsshortkey{short}
```

`\glsshortpluralkey`

```
6104 \newcommand*\glsshortpluralkey{shortplural}
```

`\glslongkey`

```
6105 \newcommand*\glslongkey{long}
```

`\glslongpluralkey`

```
6106 \newcommand*\glslongpluralkey{longplural}
```

`\acrfull` Full form of the acronym.

```
6107 \newrobustcmd*\acrfull{\@gls@hyp@opt\ns@acrfull}
```

```
6108 \newcommand*\ns@acrfull[2][ ]{%
```

```
6109 \new@ifnextchar[{\@acrfull{#1}{#2}}%
```

```
6110 \@acrfull{#1}{#2}[ ]%
```

```
6111 }
```

`\@acrfull` Low-level macro:

```
6112 \def\@acrfull#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6113 \acrfullfmt{#1}{#2}{#3}%
```

```
6114 }
```

Using `\acrlinkfullformat` and `\acrfullformat` is now deprecated as it can cause complications with the first letter upper case variants, but the package needs to provide backward compatibility support.

`\acrfullfmt` No case change full format.

```
6115 \newcommand*\acrfullfmt}[3]{%
6116   \acrlinkfullformat{\@acrlong}{\@acrshort}{#1}{#2}{#3}%
6117 }
```

`\acrlinkfullformat` Format for full links like `\acrfull`. Syntax: `\acrlinkfullformat{<long cs>}{<short cs>}{<options>}{<label>}{<in`

```
6118 \newcommand{\acrlinkfullformat}[5]{%
6119   \acrfullformat{#1}{#3}{#4}[#5]{#2}{#3}{#4}[]}%
6120 }
```

`\acrfullformat` Default full form is *<long>* (*<short>*).

```
6121 \newcommand{\acrfullformat}[2]{#1\glsspace(#2)}
```

`\glsspace` Robust space to ensure it's written to the `.glsdefs` file.

```
6122 \newrobustcmd{\glsspace}{\space}
```

Default format for full acronym

`\Acrfull`

```
6123 \newrobustcmd*\Acrfull{\@gls@hyp@opt\ns@Acrfull}
6124 \newcommand*\ns@Acrfull[2][]{%
6125   \new@ifnextchar[{\@Acrfull{#1}{#2}}%
6126     {\@Acrfull{#1}{#2}[]}%
6127 }
```

Low-level macro:

```
6128 \def\@Acrfull#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6129   \Acrfullfmt{#1}{#2}{#3}%
6130 }
```

`\Acrfullfmt` First letter upper case full format.

```
6131 \newcommand*\Acrfullfmt}[3]{%
6132   \acrlinkfullformat{\@Acrlong}{\@acrshort}{#1}{#2}{#3}%
6133 }
```

`\ACRfull`

```
6134 \newrobustcmd*\ACRfull{\@gls@hyp@opt\ns@ACRfull}
6135 \newcommand*\ns@ACRfull[2][]{%
6136   \new@ifnextchar[{\@ACRfull{#1}{#2}}%
6137     {\@ACRfull{#1}{#2}[]}%
6138 }
```

Low-level macro:

```
6139 \def\@ACRfull#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6140 \ACRfullfmt{#1}{#2}{#3}%  
6141 }
```

\ACRfullfmt All upper case full format.

```
6142 \newcommand*\ACRfullfmt [3] {%  
6143 \acrlinkfullformat{\@ACRlong}{\@ACRshort}{#1}{#2}{#3}%  
6144 }
```

Plural:

\acrfullpl

```
6145 \newrobustcmd*\acrfullpl{\@gls@hyp@opt\ns@acrfullpl}  
  
6146 \newcommand*\ns@acrfullpl [2] [] {%  
6147 \new@ifnextchar [{\@acrfullpl{#1}{#2}}%  
6148 {\@acrfullpl{#1}{#2} []}%  
6149 }
```

Low-level macro:

```
6150 \def\@acrfullpl#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6151 \acrfullplfmt{#1}{#2}{#3}%  
6152 }
```

\acrfullplfmt No case change plural full format.

```
6153 \newcommand*\acrfullplfmt [3] {%  
6154 \acrlinkfullformat{\@acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%  
6155 }
```

\Acrfullpl

```
6156 \newrobustcmd*\Acrfullpl{\@gls@hyp@opt\ns@Acrfullpl}  
  
6157 \newcommand*\ns@Acrfullpl [2] [] {%  
6158 \new@ifnextchar [{\@Acrfullpl{#1}{#2}}%  
6159 {\@Acrfullpl{#1}{#2} []}%  
6160 }
```

Low-level macro:

```
6161 \def\@Acrfullpl#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6162 \Acrfullplfmt{#1}{#2}{#3}%  
6163 }
```

\Acrfullplfmt First letter upper case plural full format.

```
6164 \newcommand*\Acrfullplfmt [3] {%  
6165 \acrlinkfullformat{\@Acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%  
6166 }
```

```

\ACRfullpl
6167 \newrobustcmd*{\ACRfullpl}{\@gls@hyp@opt\ns@ACRfullpl}
6168 \newcommand*\ns@ACRfullpl [2] [] {%
6169   \new@ifnextchar [{\@ACRfullpl{#1}{#2}}%
6170   {\@ACRfullpl{#1}{#2} []}%
6171 }

```

Low-level macro:

```

6172 \def\@ACRfullpl#1#2[#3]{%
  Make it easier for acronym styles to change this:
6173   \ACRfullplfmt{#1}{#2}{#3}%
6174 }

```

`\ACRfullplfmt` All upper case plural full format.

```

6175 \newcommand*{\ACRfullplfmt} [3] {%
6176   \acrlinkfullformat{\@ACRlongpl}{\@ACRshortpl}{#1}{#2}{#3}%
6177 }

```

1.18 Predefined acronym styles

`\acronymfont` This is only used with the additional acronym styles:

```
6178 \newcommand{\acronymfont} [1] {#1}
```

`\firstacronymfont` This is only used with the additional acronym styles:

```
6179 \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.

```
6180 \newcommand*{\acrnameformat} [2] {\acronymfont{#1}}
```

Define some tokens used by `\newacronym`:

`\glskeylisttok`

```
6181 \newtoks\glskeylisttok
```

`\glslabeltok`

```
6182 \newtoks\glslabeltok
```

`\glsshorttok`

```
6183 \newtoks\glsshorttok
```

`\gslongtok`

```
6184 \newtoks\gslongtok
```

`\newacronymhook` Provide a hook for `\newacronym`:

```
6185 \newcommand*{\newacronymhook}{}
```


GenericNewAcronym New improved version of setting the acronym style.

```
6186 \newcommand*{\SetGenericNewAcronym}{%
```

Change the behaviour of \Glsentryname to workaround expansion issues that cause a problem for \makefirstuc

```
6187 \let\@Gls@entryname\@Gls@acentryname
```

Change the way acronyms are defined:

```
6188 \renewcommand{\newacronym}[4][]{%
```

```
6189 \ifdefempty{\@glsacronymlists}%
```

```
6190 {%
```

```
6191 \def\@glo@type{\acronymtype}%
```

```
6192 \setkeys{glossentry}{##1}%
```

```
6193 \DeclareAcronymList{\@glo@type}%
```

```
6194 }%
```

```
6195 }%
```

```
6196 \glskeylisttok{##1}%
```

```
6197 \glslabeltok{##2}%
```

```
6198 \glsshorttok{##3}%
```

```
6199 \glslongtok{##4}%
```

```
6200 \newacronymhook
```

```
6201 \protected@edef\@do@newglossaryentry{%
```

```
6202 \noexpand\newglossaryentry{\the\glslabeltok}%
```

```
6203 {%
```

```
6204 type=\acronymtype,%
```

```
6205 name={\expandonce{\acronymentry{##2}}},%
```

```
6206 sort={\acronymsort{\the\glsshorttok}{\the\glslongtok}},%
```

```
6207 text={\the\glsshorttok},%
```

```
6208 short={\the\glsshorttok},%
```

```
6209 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
```

```
6210 long={\the\glslongtok},%
```

```
6211 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
```

```
6212 \GenericAcronymFields,%
```

```
6213 \the\glskeylisttok
```

```
6214 }%
```

```
6215 }%
```

```
6216 \@do@newglossaryentry
```

```
6217 }%
```

Make sure that \acrfull etc reflects the new style:

```
6218 \renewcommand*{\acrfullfmt}[3]{%
```

```
6219 \glslink[##1]{##2}{\genacrfullformat{##2}{##3}}}%
```

```
6220 \renewcommand*{\Acrfullfmt}[3]{%
```

```
6221 \glslink[##1]{##2}{\Genacrfullformat{##2}{##3}}}%
```

```
6222 \renewcommand*{\ACRfullfmt}[3]{%
```

```
6223 \glslink[##1]{##2}{%
```

```
6224 \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}}%
```

```
6225 \renewcommand*{\acrfullplfmt}[3]{%
```

```
6226 \glslink[##1]{##2}{\genplacrfullformat{##2}{##3}}}%
```

```
6227 \renewcommand*{\Acrfullplfmt}[3]{%
```

```

6228 \glslink{##1}{##2}{\Genplacrfullformat{##2}{##3}}%
6229 \renewcommand*\ACRfullplfmt}[3]{%
6230 \glslink{##1}{##2}{%
6231 \mfirstucMakeUppercase{\genplacrfullformat{##2}{##3}}}%

```

Make sure that `\glsentryfull` etc reflects the new style:

```

6232 \renewcommand*\glsentryfull}[1]{\genacrfullformat{##1}{}}%
6233 \renewcommand*\Glsentryfull}[1]{\Genacrfullformat{##1}{}}%
6234 \renewcommand*\glsentryfullpl}[1]{\genplacrfullformat{##1}{}}%
6235 \renewcommand*\Glsentryfullpl}[1]{\Genplacrfullformat{##1}{}}%
6236 }

```

`\GenericAcronymFields` Fields used by `\SetGenericNewAcronym` that can be changed by the acronym style.

```

6237 \newcommand*\GenericAcronymFields{description={\the\glslongtok}}

```

`\acronymentry` `\acronymentry{<label>}`

Display style for the name field in the list of acronyms.

```

6238 \newcommand*\acronymentry}[1]{\acronymfont{\glsentryshort{#1}}}

```

`\acronymsort` `\acronymsort{<short>}{<long>}`

Default sort format for acronyms.

```

6239 \newcommand*\acronymsort}[2]{#1}

```

`\setacronymstyle` `\setacronymstyle{<style name>}`

```

6240 \newcommand*\setacronymstyle}[1]{%
6241 \ifcsundef{@glsacr@dispstyle@#1}
6242 {%
6243 \PackageError{glossaries}{Undefined acronym style ‘#1’}{%
6244 }%
6245 {%
6246 \ifdefempty{\@glsacronymlists}%
6247 {%
6248 \DeclareAcronymList{\acronymtype}%
6249 }%
6250 }%
6251 \SetGenericNewAcronym
6252 \GlsUseAcrStyleDefs{#1}%
6253 \@for\@gls@type:=\@glsacronymlists\do{%
6254 \defglsentryfmt[\@gls@type]{\GlsUseAcrEntryDispStyle{#1}}%
6255 }%
6256 }%
6257 }

```

```
\newacronymstyle \newacronymstyle{<style name>}{<entry format definition>}{<display
definitions>}
```

Defines a new acronym style called *<style name>*.

```
6258 \newcommand*{\newacronymstyle}[3]{%
6259   \ifcsdef{@glsacr@dispstyle@#1}%
6260   {%
6261     \PackageError{glossaries}{Acronym style ‘#1’ already exists}{}%
6262   }%
6263   {%
6264     \csdef{@glsacr@dispstyle@#1}{#2}%
6265     \csdef{@glsacr@styledefs@#1}{#3}%
6266   }%
6267 }
```

`\renewacronymstyle` Redefines the given acronym style.

```
6268 \newcommand*{\renewacronymstyle}[3]{%
6269   \ifcsdef{@glsacr@dispstyle@#1}%
6270   {%
6271     \csdef{@glsacr@dispstyle@#1}{#2}%
6272     \csdef{@glsacr@styledefs@#1}{#3}%
6273   }%
6274   {%
6275     \PackageError{glossaries}{Acronym style ‘#1’ doesn’t exist}{}%
6276   }%
6277 }
```

`\rEntryDispStyle`

```
6278 \newcommand*{\GlsUseAcrEntryDispStyle}[1]{\csuse{@glsacr@dispstyle@#1}}
```

`\UseAcrStyleDefs`

```
6279 \newcommand*{\GlsUseAcrStyleDefs}[1]{\csuse{@glsacr@styledefs@#1}}
```

Predefined acronym styles:

`long-short` *<long>* (*<short>*) acronym style.

```
6280 \newacronymstyle{long-short}%
6281 {%
```

Check for long form in case this is a mixed glossary.

```
6282   \ifglshaslong{glslabel}{glsngenacfmt}{glsngenentryfmt}%
6283 }%
6284 {%
6285   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6286   \renewcommand*{\genacrfullformat}[2]{%
6287     \glsentrylong{##1}##2\space
6288     (\protect\firstacronymfont{\glsentryshort{##1}})%
6289   }%
6290   \renewcommand*{\Genacrfullformat}[2]{%
```

```

6291 \Glsentrylong{##1}##2\space
6292 (\protect\firstacronymfont{\glsentryshort{##1}})%
6293 }%
6294 \renewcommand*\genplacrfullformat}[2]{%
6295 \glsentrylongpl{##1}##2\space
6296 (\protect\firstacronymfont{\glsentryshortpl{##1}})%
6297 }%
6298 \renewcommand*\Genplacrfullformat}[2]{%
6299 \Glsentrylongpl{##1}##2\space
6300 (\protect\firstacronymfont{\glsentryshortpl{##1}})%
6301 }%
6302 \renewcommand*\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6303 \renewcommand*\acronymsort}[2]{##1}%
6304 \renewcommand*\acronymfont}[1]{##1}%
6305 \renewcommand*\firstacronymfont}[1]{\acronymfont{##1}}%
6306 \renewcommand*\acrpluralsuffix{\glspluralsuffix}%
6307 }

```

long-sp-short Similar to the previous style but allows the space between the long and short form to be customized.

```

6308 \newacronymstyle{long-sp-short}%
6309 {%

```

Check for long form in case this is a mixed glossary.

```

6310 \ifglshaslong{\glslabel}{\glsacspace}{\glsacspace}%
6311 }%
6312 {%
6313 \renewcommand*\GenericAcronymFields{description={\the\glslongtok}}%
6314 \renewcommand*\genacrfullformat}[2]{%
6315 \glsentrylong{##1}##2\glsacspace{##1}%
6316 (\protect\firstacronymfont{\glsentryshort{##1}})%
6317 }%
6318 \renewcommand*\Genacrfullformat}[2]{%
6319 \Glsentrylong{##1}##2\glsacspace{##1}%
6320 (\protect\firstacronymfont{\glsentryshort{##1}})%
6321 }%
6322 \renewcommand*\genplacrfullformat}[2]{%
6323 \glsentrylongpl{##1}##2\glsacspace{##1}%
6324 (\protect\firstacronymfont{\glsentryshortpl{##1}})%
6325 }%
6326 \renewcommand*\Genplacrfullformat}[2]{%
6327 \Glsentrylongpl{##1}##2\glsacspace{##1}%
6328 (\protect\firstacronymfont{\glsentryshortpl{##1}})%
6329 }%
6330 \renewcommand*\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6331 \renewcommand*\acronymsort}[2]{##1}%
6332 \renewcommand*\acronymfont}[1]{##1}%
6333 \renewcommand*\firstacronymfont}[1]{\acronymfont{##1}}%
6334 \renewcommand*\acrpluralsuffix{\glspluralsuffix}%
6335 }

```

`\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.

```
6336 \newcommand*{\glsacspace}[1]{%
6337   \settowidth{\dimen@}{(\firstacronymfont{\glsentryshort{#1}})}%
6338   \ifdim\dimen@<3em~\else\space\fi
6339 }
```

`short-long` (*short*) (*long*) acronym style.

```
6340 \newacronymstyle{short-long}%
6341 {%
```

Check for long form in case this is a mixed glossary.

```
6342   \ifglshaslong{\glslabel}{\glsngenacfmt}{\glsngenentryfmt}%
6343 }%
6344 {%
6345   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6346   \renewcommand*{\genacrfullformat}[2]{%
6347     \protect\firstacronymfont{\glsentryshort{##1}}##2\space
6348     (\glsentrylong{##1})%
6349   }%
6350   \renewcommand*{\Genacrfullformat}[2]{%
6351     \protect\firstacronymfont{\Glsentryshort{##1}}##2\space
6352     (\glsentrylong{##1})%
6353   }%
6354   \renewcommand*{\genplacrfullformat}[2]{%
6355     \protect\firstacronymfont{\glsentryshortpl{##1}}##2\space
6356     (\glsentrylongpl{##1})%
6357   }%
6358   \renewcommand*{\Genplacrfullformat}[2]{%
6359     \protect\firstacronymfont{\Glsentryshortpl{##1}}##2\space
6360     (\glsentrylongpl{##1})%
6361   }%
6362   \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6363   \renewcommand*{\acronymsort}[2]{##1}%
6364   \renewcommand*{\acronymfont}[1]{##1}%
6365   \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
6366   \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6367 }
```

`long-sc-short` (*long*) (`\textsc{short}`) acronym style.

```
6368 \newacronymstyle{long-sc-short}%
6369 {%
6370   \GlsUseAcrEntryDispStyle{long-short}%
6371 }%
6372 {%
6373   \GlsUseAcrStyleDefs{long-short}%
6374   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6375   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6376 }
```

long-sm-short *<long>* (*\textsmaller{<short>}*) acronym style.

```
6377 \newacronymstyle{long-sm-short}%
6378 {%
6379   \GlsUseAcrEntryDispStyle{long-short}%
6380 }%
6381 {%
6382   \GlsUseAcrStyleDefs{long-short}%
6383   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6384   \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6385 }
```

sc-short-long *<short>* (*\textsc{<long>}*) acronym style.

```
6386 \newacronymstyle{sc-short-long}%
6387 {%
6388   \GlsUseAcrEntryDispStyle{short-long}%
6389 }%
6390 {%
6391   \GlsUseAcrStyleDefs{short-long}%
6392   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6393   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6394 }
```

sm-short-long *<short>* (*\textsmaller{<long>}*) acronym style.

```
6395 \newacronymstyle{sm-short-long}%
6396 {%
6397   \GlsUseAcrEntryDispStyle{short-long}%
6398 }%
6399 {%
6400   \GlsUseAcrStyleDefs{short-long}%
6401   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6402   \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6403 }
```

long-short-desc *<long>* (*{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```
6404 \newacronymstyle{long-short-desc}%
6405 {%
6406   \GlsUseAcrEntryDispStyle{long-short}%
6407 }%
6408 {%
6409   \GlsUseAcrStyleDefs{long-short}%
6410   \renewcommand*{\GenericAcronymFields}{}%
6411   \renewcommand*{\acronymsort}[2]{##2}%
6412   \renewcommand*{\acronymentry}[1]{%
6413     \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6414 }
```

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*.

```

6415 \newacronymstyle{long-sp-short-desc}%
6416 {%
6417   \GlsUseAcrEntryDispStyle{long-sp-short}%
6418 }%
6419 {%
6420   \GlsUseAcrStyleDefs{long-sp-short}%
6421   \renewcommand*{\GenericAcronymFields}{}%
6422   \renewcommand*{\acronymsort}[2]{##2}%
6423   \renewcommand*{\acronymentry}[1]{%
6424     \glentrylong{##1}\glsacspace{##1}(\acronymfont{\glentryshort{##1}})}%
6425 }

```

`g-sc-short-desc` *<long>* (`\textsc{<short>}`) acronym style that has an accompanying description (which the user needs to supply).

```

6426 \newacronymstyle{long-sc-short-desc}%
6427 {%
6428   \GlsUseAcrEntryDispStyle{long-sc-short}%
6429 }%
6430 {%
6431   \GlsUseAcrStyleDefs{long-sc-short}%
6432   \renewcommand*{\GenericAcronymFields}{}%
6433   \renewcommand*{\acronymsort}[2]{##2}%
6434   \renewcommand*{\acronymentry}[1]{%
6435     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6436 }

```

`g-sm-short-desc` *<long>* (`\textsmaller{<short>}`) acronym style that has an accompanying description (which the user needs to supply).

```

6437 \newacronymstyle{long-sm-short-desc}%
6438 {%
6439   \GlsUseAcrEntryDispStyle{long-sm-short}%
6440 }%
6441 {%
6442   \GlsUseAcrStyleDefs{long-sm-short}%
6443   \renewcommand*{\GenericAcronymFields}{}%
6444   \renewcommand*{\acronymsort}[2]{##2}%
6445   \renewcommand*{\acronymentry}[1]{%
6446     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6447 }

```

`short-long-desc` *<short>* (`{<long>}`) acronym style that has an accompanying description (which the user needs to supply).

```

6448 \newacronymstyle{short-long-desc}%
6449 {%
6450   \GlsUseAcrEntryDispStyle{short-long}%
6451 }%
6452 {%
6453   \GlsUseAcrStyleDefs{short-long}%
6454   \renewcommand*{\GenericAcronymFields}{}%

```

```

6455 \renewcommand*\acronymsort}[2]{##2}%
6456 \renewcommand*\acronymentry}[1]{%
6457   \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6458 }

```

short-long-desc *<long>* (\textsc{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

6459 \newacronymstyle{sc-short-long-desc}%
6460 {%
6461   \GlsUseAcrEntryDispStyle{sc-short-long}%
6462 }%
6463 {%
6464   \GlsUseAcrStyleDefs{sc-short-long}%
6465   \renewcommand*\GenericAcronymFields{}%
6466   \renewcommand*\acronymsort}[2]{##2}%
6467   \renewcommand*\acronymentry}[1]{%
6468     \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6469 }

```

short-long-desc *<long>* (\textsmaller{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

6470 \newacronymstyle{sm-short-long-desc}%
6471 {%
6472   \GlsUseAcrEntryDispStyle{sm-short-long}%
6473 }%
6474 {%
6475   \GlsUseAcrStyleDefs{sm-short-long}%
6476   \renewcommand*\GenericAcronymFields{}%
6477   \renewcommand*\acronymsort}[2]{##2}%
6478   \renewcommand*\acronymentry}[1]{%
6479     \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6480 }

```

dua *<long>* only acronym style.

```

6481 \newacronymstyle{dua}%
6482 {%

```

Check for long form in case this is a mixed glossary.

```

6483 \ifdefempty\glscustomtext
6484   {%
6485     \ifglshaslong{\glslabel}%
6486     {%
6487       \glsifplural
6488       {%

```

Plural form:

```

6489     \glscapscase
6490     {%

```


Plural form, don't adjust case:

```
6491      \glentrylongpl{\glslabel}\glsinsert
6492      }%
6493      {%
```

Plural form, make first letter upper case:

```
6494      \Glsentrylongpl{\glslabel}\glsinsert
6495      }%
6496      {%
```

Plural form, all caps:

```
6497      \mfirstucMakeUppercase
6498      {\glentrylongpl{\glslabel}\glsinsert}%
6499      }%
6500      }%
6501      {%
```

Singular form

```
6502      \glscapscase
6503      {%
```

Singular form, don't adjust case:

```
6504      \glentrylong{\glslabel}\glsinsert
6505      }%
6506      {%
```

Subsequent singular form, make first letter upper case:

```
6507      \Glsentrylong{\glslabel}\glsinsert
6508      }%
6509      {%
```

Subsequent singular form, all caps:

```
6510      \mfirstucMakeUppercase
6511      {\glentrylong{\glslabel}\glsinsert}%
6512      }%
6513      }%
6514      }%
6515      {%
```

Not an acronym:

```
6516      \glsgenentryfmt
6517      }%
6518      }%
6519      {\glscustomtext\glsinsert}%
6520      }%
6521      {%
6522      \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6523      \renewcommand*{\acrfullfmt}[3]{%
6524      \glslink[##1]{##2}{\glentrylong{##2}##3\space
6525      (\acronymfont{\glentryshort{##2}})}}%
6526      \renewcommand*{\Acrfullfmt}[3]{%
```

```

6527 \glslink{##1}{##2}{\Glsentrylong{##2}##3\space
6528 (\acronymfont{\glsentryshort{##2}})}%
6529 \renewcommand*{\ACRfullfmt}[3]{%
6530 \glslink{##1}{##2}{%
6531 \mfirstucMakeUppercase{\glsentrylong{##2}##3\space
6532 (\acronymfont{\glsentryshort{##2}})}%

6533 \renewcommand*{\acrfullplfmt}[3]{%
6534 \glslink{##1}{##2}{\Glsentrylongpl{##2}##3\space
6535 (\acronymfont{\glsentryshortpl{##2}})}%

6536 \renewcommand*{\Acrfullplfmt}[3]{%
6537 \glslink{##1}{##2}{\Glsentrylongpl{##2}##3\space
6538 (\acronymfont{\glsentryshortpl{##2}})}%
6539 \renewcommand*{\ACRfullplfmt}[3]{%
6540 \glslink{##1}{##2}{%
6541 \mfirstucMakeUppercase{\glsentrylongpl{##2}##3\space
6542 (\acronymfont{\glsentryshortpl{##2}})}%
6543 \renewcommand*{\glsentryfull}[1]{%
6544 \glsentrylong{##1}\space(\acronymfont{\glsentryshort{##1}})%
6545 }%
6546 \renewcommand*{\Glsentryfull}[1]{%
6547 \Glsentrylong{##1}\space(\acronymfont{\glsentryshort{##1}})%
6548 }%
6549 \renewcommand*{\glsentryfullpl}[1]{%
6550 \glsentrylongpl{##1}\space(\acronymfont{\glsentryshortpl{##1}})%
6551 }%
6552 \renewcommand*{\Glsentryfullpl}[1]{%
6553 \Glsentrylongpl{##1}\space(\acronymfont{\glsentryshortpl{##1}})%
6554 }%
6555 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}%
6556 \renewcommand*{\acronymsort}[2]{##1}%
6557 \renewcommand*{\acronymfont}[1]{##1}%
6558 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6559 }

```

dua-desc <long> only acronym style with user-supplied description.

```

6560 \newacronymstyle{dua-desc}%
6561 {%
6562 \GlsUseAcrEntryDispStyle{dua}%
6563 }%
6564 {%
6565 \GlsUseAcrStyleDefs{dua}%
6566 \renewcommand*{\GenericAcronymFields}{}%

6567 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentrylong{##1}}%
6568 \renewcommand*{\acronymsort}[2]{##2}%
6569 }%

```

footnote *(short)*\footnote{*(long)*} acronym style.

```
6570 \newacronymstyle{footnote}%  
6571 {%
```

Check for long form in case this is a mixed glossary.

```
6572 \ifglshaslong{\glslabel}{\glsngenacfmt}{\glsngenentryfmt}%  
6573 }%  
6574 {%  
6575 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
```

Need to ensure hyperlinks are switched off on first use:

```
6576 \glshyperfirstfalse  
6577 \renewcommand*{\genacrfullformat}[2]{%  
6578 \protect\firstacronymfont{\glsentryshort{##1}}##2%  
6579 \protect\footnote{\glsentrylong{##1}}%  
6580 }%  
6581 \renewcommand*{\Genacrfullformat}[2]{%  
6582 \firstacronymfont{\Glsentryshort{##1}}##2%  
6583 \protect\footnote{\glsentrylong{##1}}%  
6584 }%  
6585 \renewcommand*{\genplacrfullformat}[2]{%  
6586 \protect\firstacronymfont{\glsentryshortpl{##1}}##2%  
6587 \protect\footnote{\glsentrylongpl{##1}}%  
6588 }%  
6589 \renewcommand*{\Genplacrfullformat}[2]{%  
6590 \protect\firstacronymfont{\Glsentryshortpl{##1}}##2%  
6591 \protect\footnote{\glsentrylongpl{##1}}%  
6592 }%  
6593 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%  
6594 \renewcommand*{\acronymsort}[2]{##1}%  
6595 \renewcommand*{\acronymfont}[1]{##1}%  
6596 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
```

Don't use footnotes for \acrfull:

```
6597 \renewcommand*{\acrfullfmt}[3]{%  
6598 \glslink[##1]{##2}{\acronymfont{\glsentryshort{##2}}##3\space  
6599 (\glsentrylong{##2})}%  
6600 \renewcommand*{\Acrfullfmt}[3]{%  
6601 \glslink[##1]{##2}{\acronymfont{\Glsentryshort{##2}}##3\space  
6602 (\glsentrylong{##2})}%  
6603 \renewcommand*{\ACRfullfmt}[3]{%  
6604 \glslink[##1]{##2}{%  
6605 \mfirstucMakeUppercase{\acronymfont{\glsentryshort{##2}}##3\space  
6606 (\glsentrylong{##2})}}}%  
6607 \renewcommand*{\acrfullplfmt}[3]{%  
6608 \glslink[##1]{##2}{\acronymfont{\glsentryshortpl{##2}}##3\space  
6609 (\glsentrylongpl{##2})}%  
6610 \renewcommand*{\Acrfullplfmt}[3]{%  
6611 \glslink[##1]{##2}{\acronymfont{\Glsentryshortpl{##2}}##3\space  
6612 (\glsentrylongpl{##2})}%
```

```

6613 \renewcommand*{\ACRfullplfmt}[3]{%
6614   \glslink[##1]{##2}{%
6615     \mfirstucMakeUppercase{\acronymfont{\glsentryshortpl{##2}}##3\space
6616     (\glsentrylongpl{##2})}}}%

```

Similarly for \glsentryfull etc:

```

6617 \renewcommand*{\glsentryfull}[1]{%
6618   \acronymfont{\glsentryshort{##1}}\space(\glsentrylong{##1})}%
6619 \renewcommand*{\Glsentryfull}[1]{%
6620   \acronymfont{\Glsentryshort{##1}}\space(\glsentrylong{##1})}%
6621 \renewcommand*{\glsentryfullpl}[1]{%
6622   \acronymfont{\glsentryshortpl{##1}}\space(\glsentrylongpl{##1})}%
6623 \renewcommand*{\Glsentryfullpl}[1]{%
6624   \acronymfont{\Glsentryshortpl{##1}}\space(\glsentrylongpl{##1})}%
6625 }

```

footnote-sc \textsc{<short>}\footnote{<long>} acronym style.

```

6626 \newacronymstyle{footnote-sc}%
6627 {%
6628   \GlsUseAcrEntryDisplayStyle{footnote}%
6629 }%
6630 {%
6631   \GlsUseAcrStyleDefs{footnote}%
6632   \renewcommand{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}
6633   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6634   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6635 }%

```

footnote-sm \textsmaller{<short>}\footnote{<long>} acronym style.

```

6636 \newacronymstyle{footnote-sm}%
6637 {%
6638   \GlsUseAcrEntryDisplayStyle{footnote}%
6639 }%
6640 {%
6641   \GlsUseAcrStyleDefs{footnote}%
6642   \renewcommand{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}
6643   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6644   \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6645 }%

```

footnote-desc <short>\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

6646 \newacronymstyle{footnote-desc}%
6647 {%
6648   \GlsUseAcrEntryDisplayStyle{footnote}%
6649 }%
6650 {%
6651   \GlsUseAcrStyleDefs{footnote}%
6652   \renewcommand*{\GenericAcronymFields}{}%

```

```

6653 \renewcommand*\acronymsort}[2]{##2}%
6654 \renewcommand*\acronymentry}[1]{%
6655   \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6656 }

```

ootnote-sc-desc \textsc{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

6657 \newacronymstyle{footnote-sc-desc}%
6658 {%
6659   \GlsUseAcrEntryDispStyle{footnote-sc}%
6660 }%
6661 {%
6662   \GlsUseAcrStyleDefs{footnote-sc}%
6663   \renewcommand*\GenericAcronymFields{}%
6664   \renewcommand*\acronymsort}[2]{##2}%
6665   \renewcommand*\acronymentry}[1]{%
6666     \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6667 }

```

ootnote-sm-desc \textsmaller{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

6668 \newacronymstyle{footnote-sm-desc}%
6669 {%
6670   \GlsUseAcrEntryDispStyle{footnote-sm}%
6671 }%
6672 {%
6673   \GlsUseAcrStyleDefs{footnote-sm}%
6674   \renewcommand*\GenericAcronymFields{}%
6675   \renewcommand*\acronymsort}[2]{##2}%
6676   \renewcommand*\acronymentry}[1]{%
6677     \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6678 }

```

AcronymSynonyms

```
6679 \newcommand*\DefineAcronymSynonyms}{%
```

Short form

\acs

```
6680 \let\acs\acrshort
```

First letter uppercase short form

\Acs

```
6681 \let\Acs\Acrshort
```

Plural short form

\acsp

```
6682 \let\acsp\acrshortpl
```

First letter uppercase plural short form

`\Acsp`

6683 `\let\Acsp\Acrshortpl`

Long form

`\acl`

6684 `\let\acl\acrlong`

Plural long form

`\aclp`

6685 `\let\aclp\acrlongpl`

First letter upper case long form

`\Acl`

6686 `\let\Acl\Acrlong`

First letter upper case plural long form

`\Aclp`

6687 `\let\Aclp\Acrlongpl`

Full form

`\acf`

6688 `\let\acf\acrfull`

Plural full form

`\acfp`

6689 `\let\acfp\acrfullpl`

First letter upper case full form

`\Acf`

6690 `\let\Acf\Acrfull`

First letter upper case plural full form

`\Acfp`

6691 `\let\Acfp\Acrfullpl`

Standard form

`\ac`

6692 `\let\ac\gls`

First upper case standard form

`\Ac`

6693 `\let\Ac\Gls`

Standard plural form

`\acp`

```
6694 \let\acp\glspl
```

Standard first letter upper case plural form

`\Acp`

```
6695 \let\Acp\Glspl
```

```
6696 }
```

Define synonyms if required

```
6697 \ifglsacrshortcuts
```

```
6698 \DefineAcronymSynonyms
```

```
6699 \fi
```

These commands for setting the style are now deprecated but are kept for backward compatibility.

`\glsAcronymDisplayStyle` Sets the default acronym display style for given glossary.

```
6700 \newcommand*{\SetDefaultAcronymDisplayStyle}[1]{%
```

```
6701 \defglsentryfmt[#1]{\glsentryfmt}%
```

```
6702 }
```

`\glsNewAcronymDef` Sets up the acronym definition for the default style. The information is provided by the tokens

`\glslabeltok`, `\glsshorttok`, `\glslongtok` and `\glskeylisttok`.

```
6703 \newcommand*{\DefaultNewAcronymDef}{%
```

```
6704 \edef\@do@newglossaryentry{%
```

```
6705 \noexpand\newglossaryentry{\the\glslabeltok}%
```

```
6706 {%
```

```
6707 type=\acronymtype,%
```

```
6708 name={\the\glsshorttok},%
```

```
6709 sort={\the\glsshorttok},%
```

```
6710 text={\the\glsshorttok},%
```

```
6711 first={\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
```

```
6712 plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
```

```
6713 firstplural={\acrfullformat{\noexpand\expandonce\noexpand\@glo@longpl}%
```

```
6714 {\noexpand\expandonce\noexpand\@glo@shortpl}},%
```

```
6715 short={\the\glsshorttok},%
```

```
6716 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
```

```
6717 long={\the\glslongtok},%
```

```
6718 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
```

```
6719 description={\the\glslongtok},%
```

```
6720 descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
```

Remaining options specified by the user:

```
6721 \the\glskeylisttok
```

```
6722 }%
```

```
6723 }%
```

```
6724 \let\@org@gls@assign@firstpl\gls@assign@firstpl
```

```

6725 \let\@org@gls@assign@plural\gls@assign@plural
6726 \let\@org@gls@assign@descplural\gls@assign@descplural
6727 \def\gls@assign@firstpl##1##2{%
6728   \@gls@expand@field{##1}{firstpl}{##2}%
6729 }%
6730 \def\gls@assign@plural##1##2{%
6731   \@gls@expand@field{##1}{plural}{##2}%
6732 }%
6733 \def\gls@assign@descplural##1##2{%
6734   \@gls@expand@field{##1}{descplural}{##2}%
6735 }%
6736 \@do@newglossaryentry
6737 \let\gls@assign@firstpl\@org@gls@assign@firstpl
6738 \let\gls@assign@plural\@org@gls@assign@plural
6739 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
6740 }

```

`\ultAcronymStyle` Set up the default acronym style:

```
6741 \newcommand*{\SetDefaultAcronymStyle}{%
```

Set the display style:

```

6742   \@for\@gls@type:=\@gls@acronymlists\do{%
6743     \SetDefaultAcronymDisplayStyle{\@gls@type}%
6744   }%

```

Set up the definition of `\newacronym`:

```
6745 \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).

```

6746   \ifx\@gls@acronymlists\empty
6747     \def\@glo@type{\acronymtype}%
6748     \setkeys{glossentry}{##1}%
6749     \DeclareAcronymList{\@glo@type}%
6750     \SetDefaultAcronymDisplayStyle{\@glo@type}%
6751   \fi
6752   \glskeylisttok{##1}%
6753   \glslabeltok{##2}%
6754   \glsshorttok{##3}%
6755   \glslongtok{##4}%
6756   \newacronymhook
6757   \DefaultNewAcronymDef
6758 }%
6759 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6760 }

```

`\acrfootnote` Used by the footnote acronym styles.

```
6761 \newcommand*{\acrfootnote}[3]{\acrlinkfootnote{#1}{#2}{#3}}
```

`\acrlinkfootnote`


```

6762 \newcommand*{\acrlinkfootnote}[3]{%
6763   \footnote{\glslink[#1]{#2}{#3}}%
6764 }

```

acrnolinkfootnote

```

6765 \newcommand*{\acrnolinkfootnote}[3]{%
6766   \footnote{#3}%
6767 }

```

acronymDisplayStyle Sets the acronym display style for given glossary for the description and footnote combination.

```

6768 \newcommand*{\SetDescriptionFootnoteAcronymDisplayStyle}[1]{%
6769   \defglsentryfmt[#1]{%
6770     \ifdefempty\glscustomtext
6771     {%
6772       \ifglsused{\glslabel}%
6773       {%
6774         \acronymfont{\glsentryfmt}%
6775       }%
6776       {%
6777         \firstacronymfont{\glsentryfmt}%
6778         \ifgls hassymbol{\glslabel}%
6779         {%
6780           \expandafter\protect\expandafter\acrfootnote\expandafter
6781             {\@gls@link@opts}{\@gls@link@label}%
6782           {%
6783             \glsifplural
6784               {\glsentrysymbolplural{\glslabel}}%
6785               {\glsentrysymbol{\glslabel}}%
6786             }%
6787           }%
6788         }%
6789       }%
6790     {\glscustomtext\glsinsert}%
6791   }%
6792 }

```

acronymDef

```

6793 \newcommand*{\DescriptionFootnoteNewAcronymDef}{%
6794   \edef\@do@newglossaryentry{%
6795     \noexpand\newglossaryentry{\the\glslabeltok}%
6796     {%
6797       type=\acronymtype,%
6798       name={\noexpand\acronymfont{\the\glsshorttok}},%
6799       sort={\the\glsshorttok},%
6800       first={\the\glsshorttok},%
6801       firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6802       text={\the\glsshorttok},%

```

```

6803 plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6804 short={\the\glsshorttok},%
6805 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6806 long={\the\glslongtok},%
6807 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6808 symbol={\the\glslongtok},%
6809 symbolplural={\noexpand\expandonce\noexpand\@glo@longpl},%
6810 \the\glskeylisttok
6811 }%
6812 }%
6813 \let\@org@gls@assign@firstpl\gls@assign@firstpl
6814 \let\@org@gls@assign@plural\gls@assign@plural
6815 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
6816 \def\gls@assign@firstpl##1##2{%
6817 \@@gls@expand@field{##1}{firstpl}{##2}%
6818 }%
6819 \def\gls@assign@plural##1##2{%
6820 \@@gls@expand@field{##1}{plural}{##2}%
6821 }%
6822 \def\gls@assign@symbolplural##1##2{%
6823 \@@gls@expand@field{##1}{symbolplural}{##2}%
6824 }%
6825 \do@newglossaryentry
6826 \let\gls@assign@plural\@org@gls@assign@plural
6827 \let\gls@assign@firstpl\@org@gls@assign@firstpl
6828 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
6829 }

```

`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.

```

6830 \newcommand*{\SetDescriptionFootnoteAcronymStyle}{%
6831 \renewcommand{\newacronym}[4][[]]{%
6832 \ifx\@glsacronymlists\@empty
6833 \def\@glo@type{\acronymtype}%
6834 \setkeys{glossentry}{##1}%
6835 \DeclareAcronymList{\@glo@type}%
6836 \SetDescriptionFootnoteAcronymDisplayStyle{\@glo@type}%
6837 \fi
6838 \glskeylisttok{##1}%
6839 \glslabeltok{##2}%
6840 \glsshorttok{##3}%
6841 \glslongtok{##4}%
6842 \newacronymhook
6843 \DescriptionFootnoteNewAcronymDef
6844 }%

```

If footnote package option is specified, set the first use to append the long form (stored in

symbol) as a footnote.

```
6845 \@for\@gls@type:=\@glsacronymlists\do{%
6846   \SetDescriptionFootnoteAcronymDisplayStyle{\@gls@type}%
6847 }%
```

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.

```
6848 \ifglsacrsmallcaps
6849   \renewcommand*\acronymfont}[1]{\textsc{##1}}%
6850   \renewcommand*\acrpluralsuffix{\glsupacrpluralsuffix}%
6851 \else
6852   \ifglsacrsmaller
6853     \renewcommand*\acronymfont}[1]{\textsmaller{##1}}%
6854   \fi
6855 \fi
```

Check for package option clash

```
6856 \ifglsacrdua
6857   \PackageError{glossaries}{Option clash: 'footnote' and 'dua'
6858   can't both be set}{}%
6859 \fi
6860 }%
```

`AcronymDisplayStyle` Sets the acronym display style for given glossary with description and dua combination.

```
6861 \newcommand*\SetDescriptionDUAAcronymDisplayStyle}[1]{%
6862   \defglsentryfmt[#1]{\glsgenentryfmt}%
6863 }
```

`UANewAcronymDef`

```
6864 \newcommand*\DescriptionDUANewAcronymDef}{%
6865   \edef\@do@newglossaryentry{%
6866     \noexpand\newglossaryentry{\the\glslabeltok}%
6867     {%
6868       type=\acronymtype,%
6869       name={\the\glslongtok},%
6870       sort={\the\glslongtok},
6871       text={\the\glslongtok},%
6872       first={\the\glslongtok},%
6873       plural={\noexpand\expandonce\noexpand\@glo@longpl},%
6874       firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
6875       short={\the\glsshorttok},%
6876       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6877       long={\the\glslongtok},%
6878       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6879       symbol={\the\glsshorttok},%
6880       symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6881       \the\glskeylisttok
6882     }%
6883   }%
```

```

6884 \let\@org@gl@s@assign@firstpl\gl@s@assign@firstpl
6885 \let\@org@gl@s@assign@plural\gl@s@assign@plural
6886 \let\@org@gl@s@assign@symbolplural\gl@s@assign@symbolplural
6887 \def\gl@s@assign@firstpl##1##2{%
6888   \@@gl@s@expand@field{##1}{firstpl}{##2}%
6889 }%
6890 \def\gl@s@assign@plural##1##2{%
6891   \@@gl@s@expand@field{##1}{plural}{##2}%
6892 }%
6893 \def\gl@s@assign@symbolplural##1##2{%
6894   \@@gl@s@expand@field{##1}{symbolplural}{##2}%
6895 }%
6896 \@do@newglossaryentry
6897 \let\gl@s@assign@firstpl\@org@gl@s@assign@firstpl
6898 \let\gl@s@assign@plural\@org@gl@s@assign@plural
6899 \let\gl@s@assign@symbolplural\@org@gl@s@assign@symbolplural
6900 }

```

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.

```

6901 \newcommand*\SetDescriptionDUAAcronymStyle{%
6902   \ifgl@sacrsmallcaps
6903     \PackageError{glossaries}{Option clash: 'smallcaps' and 'dua'
6904       can't both be set}{}%
6905   \else
6906     \ifgl@sacrsmaller
6907       \PackageError{glossaries}{Option clash: 'smaller' and 'dua'
6908         can't both be set}{}%
6909     \fi
6910   \fi
6911   \renewcommand{\newacronym}[4] []{%
6912     \ifx\@gl@sacronymlists\@empty
6913       \def\@glo@type{\acronymtype}%
6914       \setkeys{glossentry}{##1}%
6915       \DeclareAcronymList{\@glo@type}%
6916       \SetDescriptionDUAAcronymDisplayStyle{\@glo@type}%
6917     \fi
6918     \gl@skeylisttok{##1}%
6919     \gl@slabeltok{##2}%
6920     \gl@sshorttok{##3}%
6921     \gl@slongtok{##4}%
6922     \newacronymhook
6923     \DescriptionDUANewAcronymDef
6924   }%

```

Set display.

```

6925 \@for\@gl@s@type:=\@gl@sacronymlists\do{%
6926   \SetDescriptionDUAAcronymDisplayStyle{\@gl@s@type}%

```

```
6927 }%
6928 }%
```

`\acronymDisplayStyle` Sets the acronym display style for given glossary using the description setting (but not footnote or dua).

```
6929 \newcommand*{\SetDescriptionAcronymDisplayStyle}[1]{%
6930   \def\glsentryfmt[#1]{%
6931     \ifdefempty\glscustomtext
6932     {%
6933       \ifglsused{\glslabel}%
6934       {%
```

Move the inserted text outside of `\acronymfont`

```
6935       \let\gls@org@insert\glsinsert
6936       \let\glsinsert\@empty
6937       \acronymfont{\glsgenentryfmt}\gls@org@insert
6938     }%
6939   {%
6940     \glsgenentryfmt
6941     \ifgls hassymbol{\glslabel}%
6942     {%
6943       \glsifplural
6944       {%
6945         \def\@glo@symbol{\glsentrysymbolplural{\glslabel}}%
6946       }%
6947     }%
6948     \def\@glo@symbol{\glsentrysymbol{\glslabel}}%
6949   }%
6950   \space(\protect\firstacronymfont
6951   {\glscapscase
6952   {\@glo@symbol}
6953   {\@glo@symbol}
6954   {\mfirstucMakeUppercase{\@glo@symbol}}})%
6955   }%
6956   }%
6957 }%
6958 }%
6959 {\glscustomtext\glsinsert}%
6960 }%
6961 }
```

`\newAcronymDef`

```
6962 \newcommand*{\DescriptionNewAcronymDef}{%
6963   \edef\@do@newglossaryentry{%
6964     \noexpand\newglossaryentry{\the\glslabeltok}%
6965     {%
6966       type=\acronymtype,%
6967       name={\noexpand
```

```

6968     \acronymformat{\the\glsshorttok}{\the\glslongtok}},%
6969     sort={\the\glsshorttok},%
6970     first={\the\glslongtok},%
6971     firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
6972     text={\the\glsshorttok},%
6973     plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6974     short={\the\glsshorttok},%
6975     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6976     long={\the\glslongtok},%
6977     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6978     symbol={\noexpand\@glo@text},%
6979     symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6980     \the\glskeylisttok}%
6981 }%
6982 \let\@org@gls@assign@firstpl\gls@assign@firstpl
6983 \let\@org@gls@assign@plural\gls@assign@plural
6984 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
6985 \def\gls@assign@firstpl##1##2{%
6986   \@@gls@expand@field{##1}{firstpl}{##2}%
6987 }%
6988 \def\gls@assign@plural##1##2{%
6989   \@@gls@expand@field{##1}{plural}{##2}%
6990 }%
6991 \def\gls@assign@symbolplural##1##2{%
6992   \@@gls@expand@field{##1}{symbolplural}{##2}%
6993 }%
6994 \@do@newglossaryentry
6995 \let\gls@assign@firstpl\@org@gls@assign@firstpl
6996 \let\gls@assign@plural\@org@gls@assign@plural
6997 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
6998 }

```

ionAcronymStyle Option description is used, but not dua or footnote. Store long form in first key and short form in text and symbol key. The name is stored using \acronymformat to allow the user to override the way the name is displayed in the list of acronyms.

```

6999 \newcommand*{\SetDescriptionAcronymStyle}{%
7000   \renewcommand{\newacronym}[4] []{%
7001     \ifx\@glsacronymlists\@empty
7002       \def\@glo@type{\acronymtype}%
7003       \setkeys{glossentry}{##1}%
7004       \DeclareAcronymList{\@glo@type}%
7005       \SetDescriptionAcronymDisplayStyle{\@glo@type}%
7006     \fi
7007     \glskeylisttok{##1}%
7008     \glslabeltok{##2}%
7009     \glsshorttok{##3}%
7010     \glslongtok{##4}%
7011     \newacronymhook
7012     \DescriptionNewAcronymDef

```

7013 }%

Set display.

```
7014 \@for\@gls@type:=\@gls@acronymlists\do{%
7015   \SetDescriptionAcronymDisplayStyle{\@gls@type}%
7016 }%
```

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.

```
7017 \ifglsacrsmallcaps
7018   \renewcommand{\acronymfont}[1]{\textsc{##1}}
7019   \renewcommand*\acrpluralsuffix{\glsupacrpluralsuffix}%
7020 \else
7021   \ifglsacrsmaller
7022     \renewcommand*\acronymfont}[1]{\textsmaller{##1}}%
7023   \fi
7024 \fi
7025 }%
```

`\gls@acronymfont` Sets the acronym display style for given glossary with footnote setting (but not description or dua).

```
7026 \newcommand*\SetFootnoteAcronymDisplayStyle}[1]{%
7027   \defglsentryfmt[#1]{%
```

```
7028     \ifdefempty\gls@customtext
7029     {%
```

Move the inserted text outside of `\acronymfont`

```
7030     \let\gls@org@insert\glsinsert
7031     \let\glsinsert\@empty
7032     \ifglsused{\glslabel}%
7033     {%
7034       \acronymfont{\gls@genentryfmt}\gls@org@insert
7035     }%
7036     {%
7037       \firstacronymfont{\gls@genentryfmt}\gls@org@insert
7038       \ifgls@haslong{\glslabel}%
7039       {%
7040         \expandafter\protect\expandafter\acrfootnote\expandafter
7041         {\@gls@link@opts}{\@gls@link@label}%
7042         {%
7043           \glsifplural
7044             {\glsentrylongpl{\glslabel}}%
7045             {\glsentrylong{\glslabel}}%
7046           }%
7047         }%
7048       }%
7049     }%
7050 }%
```

```

7051   {\glscustomtext\glsinsert}%
7052 }%
7053 }

```

teNewAcronymDef

```

7054 \newcommand*{\FootnoteNewAcronymDef}{%
7055   \edef\@do@newglossaryentry{%
7056     \noexpand\newglossaryentry{\the\glslabeltok}%
7057     {%
7058       type=\acronymtype,%
7059       name={\noexpand\acronymfont{\the\glsshorttok}},%
7060       sort={\the\glsshorttok},%
7061       text={\the\glsshorttok},%
7062       plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7063       first={\the\glsshorttok},%
7064       firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7065       short={\the\glsshorttok},%
7066       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7067       long={\the\glslongtok},%
7068       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7069       description={\the\glslongtok},%
7070       descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7071       \the\glskeylisttok
7072     }%
7073   }%
7074   \let\@org@gls@assign@plural\gls@assign@plural
7075   \let\@org@gls@assign@firstpl\gls@assign@firstpl
7076   \let\@org@gls@assign@descplural\gls@assign@descplural
7077   \def\gls@assign@firstpl##1##2{%
7078     \@gls@expand@field{##1}{firstpl}{##2}%
7079   }%
7080   \def\gls@assign@plural##1##2{%
7081     \@gls@expand@field{##1}{plural}{##2}%
7082   }%
7083   \def\gls@assign@descplural##1##2{%
7084     \@gls@expand@field{##1}{descplural}{##2}%
7085   }%
7086   \@do@newglossaryentry
7087   \let\gls@assign@plural\@org@gls@assign@plural
7088   \let\gls@assign@firstpl\@org@gls@assign@firstpl
7089   \let\gls@assign@descplural\@org@gls@assign@descplural
7090 }

```

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.

```

7091 \newcommand*{\SetFootnoteAcronymStyle}{%
7092   \renewcommand{\newacronym}[4][]{%
7093     \ifx\@glsacronymlists\@empty
7094       \def\@glo@type{\acronymtype}%

```



```

7095     \setkeys{glossentry}{##1}%
7096     \DeclareAcronymList{\@glo@type}%
7097     \SetFootnoteAcronymDisplayStyle{\@glo@type}%
7098     \fi
7099     \glskeylisttok{##1}%
7100     \glslabeltok{##2}%
7101     \glsshorttok{##3}%
7102     \glslongtok{##4}%
7103     \newacronymhook
7104     \FootnoteNewAcronymDef
7105 }%

```

Set display

```

7106 \@for\@gls@type:=\@gls@acronymlists\do{%
7107   \SetFootnoteAcronymDisplayStyle{\@gls@type}%
7108 }%

```

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.

```

7109 \ifglsacrsmallcaps
7110   \renewcommand*\acronymfont}[1]{\textsc{##1}}%
7111   \renewcommand*\acrpluralsuffix{\glsupacrpluralsuffix}%
7112 \else
7113   \ifglsacrsmaller
7114     \renewcommand*\acronymfont}[1]{\textsmaller{##1}}%
7115   \fi
7116 \fi

```

Check for option clash

```

7117 \ifglsacrdua
7118   \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’
7119     can’t both be set}{}%
7120 \fi
7121 }%

```

`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.

```

7122 \DeclareRobustCommand*\glsdoparenifnotempty}[2]{%
7123   \protected@edef\gls@tmp{##1}%
7124   \ifdefempty\gls@tmp
7125   }%
7126   {%
7127     \ifx\gls@tmp\@gls@default@value
7128     \else
7129       \space (#2{##1})%
7130     \fi
7131   }%
7132 }

```

`nymDisplayStyle` Sets the acronym display style for given glossary where neither footnote nor description is required, but smallcaps or smaller specified.

```
7133 \newcommand*{\SetSmallAcronymDisplayStyle}[1]{%
7134   \defglentryfmt[#1]{%
```

```
7135     \ifdefempty\glscustomtext
7136     {%
```

Move the inserted text outside of `\acronymfont`

```
7137     \let\gls@org@insert\glsinsert
7138     \let\glsinsert\@empty
7139     \ifglused{\glslabel}%
7140     {%
7141       \acronymfont{\glsentryfmt}\gls@org@insert
7142     }%
7143     {%
7144       \glsentryfmt
7145       \ifglshassymbol{\glslabel}%
7146       {%
7147         \glsifplural
7148         {%
7149           \def\@glo@symbol{\glsentrysymbolplural{\glslabel}}%
7150           }%
7151           {%
7152             \def\@glo@symbol{\glsentrysymbol{\glslabel}}%
7153             }%
7154             \space
7155             (\glscapscase
7156             {\firstacronymfont{\@glo@symbol}}%
7157             {\firstacronymfont{\@glo@symbol}}%
7158             {\firstacronymfont{\mfirstucMakeUppercase{\@glo@symbol}}})%
7159           }%
7160           {}%
7161         }%
7162       }%
7163     {\glscustomtext\glsinsert}%
7164   }%
7165 }
```

`llNewAcronymDef`

```
7166 \newcommand*{\SmallNewAcronymDef}{%
7167   \edef\@do@newglossaryentry{%
7168     \noexpand\newglossaryentry{\the\glslabeltok}%
7169     {%
7170       type=\acronymtype,%
7171       name={\noexpand\acronymfont{\the\glsshorttok}},%
7172       sort={\the\glsshorttok},%
7173       text={\the\glsshorttok},%
```

Default to the short plural.

```

7174 plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7175 first={\the\glslongtok},%
    Default to the long plural.
7176 firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7177 short={\the\glsshorttok},%
7178 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7179 long={\the\glslongtok},%
7180 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7181 description={\noexpand\@glo@first},%
7182 descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7183 symbol={\the\glsshorttok},%
    Default to the short plural.
7184 symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7185 \the\glskeylisttok
7186 }%
7187 }%
7188 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7189 \let\@org@gls@assign@plural\gls@assign@plural
7190 \let\@org@gls@assign@descplural\gls@assign@descplural
7191 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7192 \def\gls@assign@firstpl##1##2{%
7193   \@gls@expand@field{##1}{firstpl}{##2}%
7194 }%
7195 \def\gls@assign@plural##1##2{%
7196   \@gls@expand@field{##1}{plural}{##2}%
7197 }%
7198 \def\gls@assign@descplural##1##2{%
7199   \@gls@expand@field{##1}{descplural}{##2}%
7200 }%
7201 \def\gls@assign@symbolplural##1##2{%
7202   \@gls@expand@field{##1}{symbolplural}{##2}%
7203 }%
7204 \do@newglossaryentry
7205 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7206 \let\gls@assign@plural\@org@gls@assign@plural
7207 \let\gls@assign@descplural\@org@gls@assign@descplural
7208 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7209 }

```

`\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.

```

7210 \newcommand*\SetSmallAcronymStyle{%
7211   \renewcommand{\newacronym}[4][ ]{%
7212     \ifx\@glsacronymlists\@empty
7213       \def\@glo@type{\acronymtype}%
7214       \setkeys{glossentry}{##1}%
7215       \DeclareAcronymList{\@glo@type}%
7216       \SetSmallAcronymDisplayStyle{\@glo@type}%

```

```

7217 \fi
7218 \glskeylisttok{##1}%
7219 \glslabeltok{##2}%
7220 \glsshorttok{##3}%
7221 \glslongtok{##4}%
7222 \newacronymhook
7223 \SmallNewAcronymDef
7224 }%

```

Change the display since first only contains long form.

```

7225 \@for\@gls@type:=\@gls@acronymlists\do{%
7226 \SetSmallAcronymDisplayStyle{\@gls@type}%
7227 }%

```

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.

```

7228 \ifglsacrsmallcaps
7229 \renewcommand*\acronymfont[1]{\textsc{##1}}
7230 \renewcommand*\acrpluralsuffix{\glsupacrpluralsuffix}%
7231 \else
7232 \renewcommand*\acronymfont[1]{\textsmaller{##1}}
7233 \fi

```

check for option clash

```

7234 \ifglsacrdua
7235 \ifglsacrsmallcaps
7236 \PackageError{glossaries}{Option clash: 'smallcaps' and 'dua'
7237 can't both be set}{}%
7238 \else
7239 \PackageError{glossaries}{Option clash: 'smaller' and 'dua'
7240 can't both be set}{}%
7241 \fi
7242 \fi
7243 }%

```

`DUADisplayStyle` Sets the acronym display style for given glossary with dua setting.

```

7244 \newcommand*\SetDUADisplayStyle[1]{%
7245 \defglsentryfmt[##1]{\glsentryfmt}%
7246 }

```

`UANewAcronymDef`

```

7247 \newcommand*\DUANewAcronymDef{%
7248 \edef\@do@newglossaryentry{%
7249 \noexpand\newglossaryentry{\the\glslabeltok}%
7250 {%
7251 type=\acronymtype,%
7252 name={\the\glsshorttok},%
7253 text={\the\glslongtok},%
7254 first={\the\glslongtok},%
7255 plural={\noexpand\expandonce\noexpand\@glo@longpl},%

```

```

7256     firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7257     short={\the\glsshorttok},%
7258     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7259     long={\the\glslongtok},%
7260     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7261     description={\the\glslongtok},%
7262     descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7263     symbol={\the\glsshorttok},%
7264     symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7265     \the\glskeylisttok
7266   }%
7267 }%
7268 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7269 \let\@org@gls@assign@plural\gls@assign@plural
7270 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7271 \let\@org@gls@assign@descplural\gls@assign@descplural
7272 \def\gls@assign@firstpl##1##2{%
7273   \@@gls@expand@field{##1}{firstpl}{##2}%
7274 }%
7275 \def\gls@assign@plural##1##2{%
7276   \@@gls@expand@field{##1}{plural}{##2}%
7277 }%
7278 \def\gls@assign@symbolplural##1##2{%
7279   \@@gls@expand@field{##1}{symbolplural}{##2}%
7280 }%
7281 \def\gls@assign@descplural##1##2{%
7282   \@@gls@expand@field{##1}{descplural}{##2}%
7283 }%
7284 \@do@newglossaryentry
7285 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7286 \let\gls@assign@plural\@org@gls@assign@plural
7287 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7288 \let\gls@assign@descplural\@org@gls@assign@descplural
7289 }

```

\SetDUASyle Always expand acronyms.

```

7290 \newcommand*{\SetDUASyle}{%
7291   \renewcommand{\newacronym}[4][]{%
7292     \ifx\@glsacronymlists\@empty
7293       \def\@glo@type{\acronymtype}%
7294       \setkeys{glossentry}{##1}%
7295       \DeclareAcronymList{\@glo@type}%
7296       \SetDUADisplayStyle{\@glo@type}%
7297     \fi
7298     \glskeylisttok{##1}%
7299     \glslabeltok{##2}%
7300     \glsshorttok{##3}%
7301     \glslongtok{##4}%
7302     \newacronymhook

```

```

7303   \DUANewAcronymDef
7304 }%

  Set the display
7305   \@for\@gls@type:=\@glsacronymlists\do{%
7306     \SetDUADisplayStyle{\@gls@type}%
7307   }%
7308 }

```

SetAcronymStyle

```

7309 \newcommand*\SetAcronymStyle{%
7310   \SetDefaultAcronymStyle
7311   \ifglsacrdescription
7312     \ifglsacrfootnote
7313       \SetDescriptionFootnoteAcronymStyle
7314     \else
7315       \ifglsacrdua
7316         \SetDescriptionDUAAcronymStyle
7317       \else
7318         \SetDescriptionAcronymStyle
7319       \fi
7320     \fi
7321   \else
7322     \ifglsacrfootnote
7323       \SetFootnoteAcronymStyle
7324     \else
7325       \ifthenelse{\boolean{glsacrsmalldescription}\OR
7326         \boolean{glsacrsmalldescription}}%
7327         {%
7328           \SetSmallAcronymStyle
7329         }%
7330       {%
7331         \ifglsacrdua
7332           \SetDUASyle
7333         \fi
7334       }%
7335     \fi
7336   \fi
7337 }

```

Set the acronym style according to the package options

```
7338 \SetAcronymStyle
```

Allow user to define their own custom acronyms. (For compatibility with versions before v3.0, the short form is stored in the user1 key, the plural short form is stored in the user2 key, the long form is stored in the user3 key and the plural long form is stored in the user4 key.) Defaults to displaying only the acronym with the long form as the description.

CustomDisplayStyle Sets the acronym display style.

```
7339 \newcommand*\SetCustomDisplayStyle}[1]{%
```

```

7340 \defglentryfmt[#1]{\glsgenentryfmt}%
7341 }

```

omAcronymFields

```

7342 \newcommand*{\CustomAcronymFields}{%
7343   name={\the\glsshorttok},%
7344   description={\the\glslongtok},%
7345   first={\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
7346   firstplural={\acrfullformat
7347     {\noexpand\glsentrylongpl{\the\glslabeltok}}%
7348     {\noexpand\glsentryshortpl{\the\glslabeltok}}},%
7349   text={\the\glsshorttok},%
7350   plural={\the\glsshorttok\noexpand\acrpluralsuffix}%
7351 }

```

omNewAcronymDef

```

7352 \newcommand*{\CustomNewAcronymDef}{%
7353   \protected@edef\do@newglossaryentry{%
7354     \noexpand\newglossaryentry{\the\glslabeltok}%
7355     {%
7356       type=\acronymtype,%
7357       short={\the\glsshorttok},%
7358       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7359       long={\the\glslongtok},%
7360       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7361       user1={\the\glsshorttok},%
7362       user2={\the\glsshorttok\noexpand\acrpluralsuffix},%
7363       user3={\the\glslongtok},%
7364       user4={\the\glslongtok\noexpand\acrpluralsuffix},%
7365       \CustomAcronymFields,%
7366       \the\glskeylisttok
7367     }%
7368   }%
7369   \do@newglossaryentry
7370 }

```

\SetCustomStyle

```

7371 \newcommand*{\SetCustomStyle}{%
7372   \renewcommand{\newacronym}[4][ ]{%
7373     \ifx\@glsacronymlists\@empty
7374       \def\@glo@type{\acronymtype}%
7375       \setkeys{glossentry}{##1}%
7376       \DeclareAcronymList{\@glo@type}%
7377       \SetCustomDisplayStyle{\@glo@type}%
7378     \fi
7379     \glskeylisttok{##1}%
7380     \glslabeltok{##2}%
7381     \glsshorttok{##3}%

```

```

7382   \glslongtok{##4}%
7383   \newacronymhook
7384   \CustomNewAcronymDef
7385 }%

Set the display
7386 \@for\@gls@type:=\@glsacronymlists\do{%
7387   \SetCustomDisplayStyle{\@gls@type}%
7388 }%
7389 }

```

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.

```
7390 \RequirePackage{glossary-hypernav}
```

The styles that use list-like environments. These are not loaded if the `nolist` option is used:

```
7391 \@gls@loadlist
```

The styles that use the `longtable` environment. These are not loaded if the `nolong` package option is used.

```
7392 \@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.

```
7393 \@gls@loadsuper
```

The tree-like styles. These are not loaded if the `notree` package option is used.

```
7394 \@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.

```

7395 \ifx\@glossary@default@style\relax
7396 \else
7397   \setglossarystyle{\@glossary@default@style}
7398 \fi

```

1.20 Debugging Commands

```
\showgloparent
```

```
\showgloparent{<label>}
```

```

7399 \newcommand*{\showgloparent}[1]{%
7400   \expandafter\show\csname glo@\glsdetoklabel{#1}@parent\endcsname
7401 }

```


`\showglolevel` `\showglolevel{<label>}`

```
7402 \newcommand*{\showglolevel}[1]{%
7403   \expandafter\show\csname glo@glstetoklabel{#1}@level\endcsname
7404 }
```

`\showglotext` `\showglotext{<label>}`

```
7405 \newcommand*{\showglotext}[1]{%
7406   \expandafter\show\csname glo@glstetoklabel{#1}@text\endcsname
7407 }
```

`\showgloplural` `\showgloplural{<label>}`

```
7408 \newcommand*{\showgloplural}[1]{%
7409   \expandafter\show\csname glo@glstetoklabel{#1}@plural\endcsname
7410 }
```

`\showglofirst` `\showglofirst{<label>}`

```
7411 \newcommand*{\showglofirst}[1]{%
7412   \expandafter\show\csname glo@glstetoklabel{#1}@first\endcsname
7413 }
```

`\showglofirstpl` `\showglofirstpl{<label>}`

```
7414 \newcommand*{\showglofirstpl}[1]{%
7415   \expandafter\show\csname glo@glstetoklabel{#1}@firstpl\endcsname
7416 }
```

`\showglotype` `\showglotype{<label>}`

```
7417 \newcommand*{\showglotype}[1]{%
7418   \expandafter\show\csname glo@glstetoklabel{#1}@type\endcsname
7419 }
```

`\showglocounter` `\showglocounter{<label>}`

```
7420 \newcommand*{\showglocounter}[1]{%  
7421 \expandafter\show\csname glo@glstdetoklabel{#1}@counter\endcsname  
7422 }
```

`\showglouserii` `\showglouserii{<label>}`

```
7423 \newcommand*{\showglouserii}[1]{%  
7424 \expandafter\show\csname glo@glstdetoklabel{#1}@userii\endcsname  
7425 }
```

`\showglouseriii` `\showglouseriii{<label>}`

```
7426 \newcommand*{\showglouseriii}[1]{%  
7427 \expandafter\show\csname glo@glstdetoklabel{#1}@useriii\endcsname  
7428 }
```

`\showglouseriiii` `\showglouseriiii{<label>}`

```
7429 \newcommand*{\showglouseriiii}[1]{%  
7430 \expandafter\show\csname glo@glstdetoklabel{#1}@useriiii\endcsname  
7431 }
```

`\showglouseriv` `\showglouseriv{<label>}`

```
7432 \newcommand*{\showglouseriv}[1]{%  
7433 \expandafter\show\csname glo@glstdetoklabel{#1}@useriv\endcsname  
7434 }
```

`\showglouserv` `\showglouserv{<label>}`

```
7435 \newcommand*{\showglouserv}[1]{%  
7436 \expandafter\show\csname glo@glstdetoklabel{#1}@userv\endcsname  
7437 }
```

`\showglouservi` `\showglouservi{<label>}`

```
7438 \newcommand*{\showglouservi}[1]{%
7439   \expandafter\show\csname glo@glstdetoklabel{#1}@uservi\endcsname
7440 }
```

`\showgloname` `\showgloname{<label>}`

```
7441 \newcommand*{\showgloname}[1]{%
7442   \expandafter\show\csname glo@glstdetoklabel{#1}@name\endcsname
7443 }
```

`\showglodesc` `\showglodesc{<label>}`

```
7444 \newcommand*{\showglodesc}[1]{%
7445   \expandafter\show\csname glo@glstdetoklabel{#1}@desc\endcsname
7446 }
```

`showglodescplural` `\showglodescplural{<label>}`

```
7447 \newcommand*{\showglodescplural}[1]{%
7448   \expandafter\show\csname glo@glstdetoklabel{#1}@descplural\endcsname
7449 }
```

`\showglosort` `\showglosort{<label>}`

```
7450 \newcommand*{\showglosort}[1]{%
7451   \expandafter\show\csname glo@glstdetoklabel{#1}@sort\endcsname
7452 }
```

`\showglosymbol` `\showglosymbol{<label>}`

```
7453 \newcommand*{\showglosymbol}[1]{%
7454   \expandafter\show\csname glo@glstdetoklabel{#1}@symbol\endcsname
7455 }
```

glosymbolplural `\showglosymbolplural{<label>}`

```
7456 \newcommand*{\showglosymbolplural}[1]{%
7457   \expandafter\show\csname glo@glstdetoklabel{#1}@symbolplural\endcsname
7458 }
```

\showgloshort `\showgloshort{<label>}`

```
7459 \newcommand*{\showgloshort}[1]{%
7460   \expandafter\show\csname glo@glstdetoklabel{#1}@short\endcsname
7461 }
```

\showglolong `\showglolong{<label>}`

```
7462 \newcommand*{\showglolong}[1]{%
7463   \expandafter\show\csname glo@glstdetoklabel{#1}@long\endcsname
7464 }
```

\showgloindex `\showgloindex{<label>}`

```
7465 \newcommand*{\showgloindex}[1]{%
7466   \expandafter\show\csname glo@glstdetoklabel{#1}@index\endcsname
7467 }
```

\showgloflag `\showgloflag{<label>}`

```
7468 \newcommand*{\showgloflag}[1]{%
7469   \expandafter\show\csname ifglo@glstdetoklabel{#1}@flag\endcsname
7470 }
```

\showgloloclist `\showgloloclist{<label>}`

```
7471 \newcommand*{\showgloloclist}[1]{%
7472   \expandafter\show\csname glo@glstdetoklabel{#1}@loclist\endcsname
7473 }
```

`\showglofield` `\showglofield{<label>}{<field>}`

```
7474 \newcommand*{\showglofield}[2]{%
7475   \csshow{glo@glstetoklabel{#1}@#2}%
7476 }
```

`showacronymlists` `\showacronymlists`

Show list of glossaries that have been flagged as a list of acronyms.

```
7477 \newcommand*{\showacronymlists}{%
7478   \show\@glsacronymlists
7479 }
```

`\showglossaries` `\showglossaries`

Show list of defined glossaries.

```
7480 \newcommand*{\showglossaries}{%
7481   \show\@glo@types
7482 }
```

`\showglossaryin` `\showglossaryin{<glossary-label>}`

Show the ‘in’ extension for the given glossary.

```
7483 \newcommand*{\showglossaryin}[1]{%
7484   \expandafter\show\csname @glo@type@#1@in\endcsname
7485 }
```

`\showglossaryout` `\showglossaryout{<glossary-label>}`

Show the ‘out’ extension for the given glossary.

```
7486 \newcommand*{\showglossaryout}[1]{%
7487   \expandafter\show\csname @glo@type@#1@out\endcsname
7488 }
```

`showglossarytitle` `\showglossarytitle{<glossary-label>}`

Show the title for the given glossary.

```
7489 \newcommand*{\showglossarytitle}[1]{%
7490   \expandafter\show\csname @glo@type@#1@title\endcsname
7491 }
```

wglossarycounter `\showglossarycounter{<glossary-label>}`

Show the counter for the given glossary.

```
7492 \newcommand*{\showglossarycounter}[1]{%
7493   \expandafter\show\csname @glotype@#1@counter\endcsname
7494 }
```

wglossaryentries `\showglossaryentries{<glossary-label>}`

Show the list of entry labels for the given glossary.

```
7495 \newcommand*{\showglossaryentries}[1]{%
7496   \expandafter\show\csname glolist@#1\endcsname
7497 }
```

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.

```
7498 \csname ifglcompatible-2.07\endcsname
7499   \RequirePackage{glossaries-compatible-207}
7500 \fi
```

2 Prefix Support (glossaries-prefix Code)

This package provides a means of adding prefixes to your glossary entries. For example, you may want to use “`a \gls{<label>}`” on first use but use “`an \gls{<label>}`” on subsequent use.

```
7501 \NeedsTeXFormat{LaTeX2e}
```

```
7502 \ProvidesPackage{glossaries-prefix}[2016/01/24 v4.21 (NLCT)]
```

Pass all options to glossaries:

```
7503 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
7504 \ProcessOptions
```

Load glossaries:

```
7505 \RequirePackage{glossaries}
```

Add the new keys:

```
7506 \define@key{glossentry}{prefixfirst}{\def\@glo@entryprefixfirst{#1}}%
```

```
7507 \define@key{glossentry}{prefixfirstplural}{\def\@glo@entryprefixfirstplural{#1}}%
```

```
7508 \define@key{glossentry}{prefix}{\def\@glo@entryprefix{#1}}%
```

```
7509 \define@key{glossentry}{prefixplural}{\def\@glo@entryprefixplural{#1}}%
```

Add them to `\@gls@keymap`:

```
7510 \appto\@gls@keymap{,%
```

```
7511   {prefixfirst}{prefixfirst},%
```

```
7512   {prefixfirstplural}{prefixfirstplural},%
```

```
7513   {prefix}{prefix},%
```

```
7514   {prefixplural}{prefixplural}}%
```

```
7515 }
```

Set the default values:

```
7516 \appto\@newglossaryentryprehook{%
```

```
7517   \def\@glo@entryprefix{}}%
```

```
7518   \def\@glo@entryprefixplural{}}%
```

```
7519   \let\@glo@entryprefixfirst\@gls@default@value
```

```
7520   \let\@glo@entryprefixfirstplural\@gls@default@value
```

```
7521 }
```

Set the assignment code:

```
7522 \appto\@newglossaryentryposthook{%
```

```
7523   \gls@assign@field{ }\@glo@label}{prefix}{\@glo@entryprefix}}%
```

```
7524   \gls@assign@field{ }\@glo@label}{prefixplural}{\@glo@entryprefixplural}}%
```

If `prefixfirst` has not been supplied, make it the same as `prefix`.

```
7525 \expandafter\gls@assign@field\expandafter
```

```
7526   {\csname glo@\@glo@label @prefix\endcsname}{\@glo@label}{prefixfirst}}%
```

```
7527   {\@glo@entryprefixfirst}}%
```

If prefixfirstplural has not been supplied, make it the same as prefixplural.

```
7528 \expandafter\gls@assign@field\expandafter
7529   {\csname glo@\glo@label @prefixplural\endcsname}{\@glo@label}%
7530   {prefixfirstplural}{\@glo@entryprefixfirstplural}%
7531 }
```

Define commands to access these fields:

entryprefixfirst

```
7532 \newcommand*\glsentryprefixfirst[1]{\csuse{glo@#1@prefixfirst}}
```

entryprefixfirstplural

```
7533 \newcommand*\glsentryprefixfirstplural[1]{\csuse{glo@#1@prefixfirstplural}}
```

\glsentryprefix

```
7534 \newcommand*\glsentryprefix[1]{\csuse{glo@#1@prefix}}
```

entryprefixplural

```
7535 \newcommand*\glsentryprefixplural[1]{\csuse{glo@#1@prefixplural}}
```

Now for the initial upper case variants:

entryprefixfirst

```
7536 \newrobustcmd*\Glsentryprefixfirst[1]{%
7537   \protected@edef\@glo@text{\csname glo@#1@prefixfirst\endcsname}%
7538   \xmakefirstuc\@glo@text
7539 }
```

entryprefixfirstplural

```
7540 \newrobustcmd*\Glsentryprefixfirstplural[1]{%
7541   \protected@edef\@glo@text{\csname glo@#1@prefixfirstplural\endcsname}%
7542   \xmakefirstuc\@glo@text
7543 }
```

\Glsentryprefix

```
7544 \newrobustcmd*\Glsentryprefix[1]{%
7545   \protected@edef\@glo@text{\csname glo@#1@prefix\endcsname}%
7546   \xmakefirstuc\@glo@text
7547 }
```

entryprefixplural

```
7548 \newrobustcmd*\Glsentryprefixplural[1]{%
7549   \protected@edef\@glo@text{\csname glo@#1@prefixplural\endcsname}%
7550   \xmakefirstuc\@glo@text
7551 }
```

Define commands to determine if the prefix keys have been set:

`\ifglshasprefix`

```
7552 \newcommand*{\ifglshasprefix}[3]{%
7553   \ifcseempty{glo@#1@prefix}%
7554   {#3}%
7555   {#2}%
7556 }
```

`hasprefixplural`

```
7557 \newcommand*{\ifglshasprefixplural}[3]{%
7558   \ifcseempty{glo@#1@prefixplural}%
7559   {#3}%
7560   {#2}%
7561 }
```

`shasprefixfirst`

```
7562 \newcommand*{\ifglshasprefixfirst}[3]{%
7563   \ifcseempty{glo@#1@prefixfirst}%
7564   {#3}%
7565   {#2}%
7566 }
```

`efixfirstplural`

```
7567 \newcommand*{\ifglshasprefixfirstplural}[3]{%
7568   \ifcseempty{glo@#1@prefixfirstplural}%
7569   {#3}%
7570   {#2}%
7571 }
```

Define commands that insert the prefix before commands like `\gls`:

`\pgls`

```
7572 \newrobustcmd{\pgls}{\@gls@hyp@opt\@pgls}
```

`\@pgls` Unstarred version.

```
7573 \newcommand*{\@pgls}[2][ ]{%
7574   \new@ifnextchar[%
7575     {\@pgls@{#1}{#2}}%
7576     {\@pgls@{#1}{#2}[ ]}%
7577 }
```

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

```
7578 \def\@pgls@#1#2[#3]{%
7579   \glsdoifexists{#2}%
7580   {%
7581     \ifglsused{#2}%
7582     {%
7583       \glsentryprefix{#2}%
7584     }%

```

```

7585   {%
7586     \glsentryprefixfirst{#2}%
7587   }%
7588   \@gls@{#1}{#2}[#3]%
7589 }%
7590 }

```

Similarly for the plural version:

```

\pglsp1
7591 \newrobustcmd{\pglsp1}{\@gls@hyp@opt\@pglsp1}

```

\@pglsp1 Unstarred version.

```

7592 \newcommand*{\@pglsp1}[2][ ]{%
7593   \new@ifnextchar[%
7594     {\@pglsp1@{#1}{#2}}%
7595     {\@pglsp1@{#1}{#2}[]}%
7596 }

```

\@pglsp1@ Read in the final optional argument:

```

7597 \def\@pglsp1@#1#2[#3]{%
7598   \glsdoifexists{#2}%
7599   {%
7600     \ifglsused{#2}%
7601     {%
7602       \glsentryprefixplural{#2}%
7603     }%
7604     {%
7605       \glsentryprefixfirstplural{#2}%
7606     }%
7607     \@glspl@{#1}{#2}[#3]%
7608   }%
7609 }

```

Now for the first letter upper case versions:

```

\Pgls
7610 \newrobustcmd{\Pgls}{\@gls@hyp@opt\@Pgls}

```

\@Pgls Unstarred version.

```

7611 \newcommand*{\@Pgls}[2][ ]{%
7612   \new@ifnextchar[%
7613     {\@Pgls@{#1}{#2}}%
7614     {\@Pgls@{#1}{#2}[]}%
7615 }

```

\@Pgls@ Read in the final optional argument:

```

7616 \def\@Pgls@#1#2[#3]{%

```

```

7617 \glsdoifexists{#2}%
7618 {%
7619   \ifglsused{#2}%
7620   {%
7621     \ifglshasprefix{#2}%
7622     {%
7623       \Glsentryprefix{#2}%
7624       \@gls@{#1}{#2}[#3]%
7625     }%
7626     {\@Gls@{#1}{#2}[#3]}%
7627   }%
7628   {%
7629     \ifglshasprefixfirst{#2}%
7630     {%
7631       \Glsentryprefixfirst{#2}%
7632       \@gls@{#1}{#2}[#3]%
7633     }%
7634     {\@Gls@{#1}{#2}[#3]}%
7635   }%
7636 }%
7637 }

```

Similarly for the plural version:

```

\Pglspl
7638 \newrobustcmd{\Pglspl}{\@gls@hyp@opt\@Pglspl}

```

\@Pglspl Unstarred version.

```

7639 \newcommand*{\@Pglspl}[2] [] {%
7640   \new@ifnextchar [%
7641     {\@Pglspl@{#1}{#2}}%
7642     {\@Pglspl@{#1}{#2} []}%
7643 }

```

\@Pglspl@ Read in the final optional argument:

```

7644 \def\@Pglspl@#1#2[#3] {%
7645   \glsdoifexists{#2}%
7646   {%
7647     \ifglsused{#2}%
7648     {%
7649       \ifglshasprefixplural{#2}%
7650       {%
7651         \Glsentryprefixplural{#2}%
7652         \@glspl@{#1}{#2}[#3]%
7653       }%
7654       {\@Glspl@{#1}{#2}[#3]}%
7655     }%
7656     {%
7657       \ifglshasprefixfirstplural{#2}%

```

```

7658     {%
7659         \Glsentryprefixfirstplural{#2}%
7660         \@glspl@{#1}{#2}[#3]%
7661     }%
7662     {\@Glspl@{#1}{#2}[#3]}%
7663 }%
7664 }%
7665 }

```

Finally the all upper case versions:

\PGLS

```
7666 \newrobustcmd{\PGLS}{\@gls@hyp@opt\PGLS}
```

\@PGLS Unstarred version.

```

7667 \newcommand*{\@PGLS}[2][ ]{%
7668     \new@ifnextchar[%
7669     {\@PGLS@{#1}{#2}}%
7670     {\@PGLS@{#1}{#2}[ ]}%
7671 }

```

\@PGLS@ Read in the final optional argument:

```

7672 \def\@PGLS@#1#2[#3]{%
7673     \glsdoifexists{#2}%
7674     {%
7675         \ifglsused{#2}%
7676         {%
7677             \mfirstucMakeUppercase{\glsentryprefix{#2}}%
7678         }%
7679         {%
7680             \mfirstucMakeUppercase{\glsentryprefixfirst{#2}}%
7681         }%
7682         \@GLS@{#1}{#2}[#3]%
7683     }%
7684 }

```

Plural version:

\PGLSp1

```
7685 \newrobustcmd{\PGLSp1}{\@gls@hyp@opt\PGLSp1}
```

\@PGLSp1 Unstarred version.

```

7686 \newcommand*{\@PGLSp1}[2][ ]{%
7687     \new@ifnextchar[%
7688     {\@PGLSp1@{#1}{#2}}%
7689     {\@PGLSp1@{#1}{#2}[ ]}%
7690 }

```

\@PGLSp1@ Read in the final optional argument:

```
7691 \def\@PGLSp1@#1#2[#3]{%  
7692   \glsdoifexists{#2}%  
7693   {%  
7694     \ifglsused{#2}%  
7695     {%  
7696       \mfirstucMakeUppercase{\glsentryprefixplural{#2}}%  
7697     }%  
7698     {%  
7699       \mfirstucMakeUppercase{\glsentryprefixfirstplural{#2}}%  
7700     }%  
7701     \@GLSp1@{#1}{#2}[#3]%  
7702   }%  
7703 }
```

3 Glossary Styles

3.1 Glossary hyper-navigation definitions (glossary-hypernav package)

Package Definition:

```
7704 \ProvidesPackage{glossary-hypernav}[2016/01/24 v4.21 (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`

```
7705 \newcommand*{\glsnavhyperlink}[3][\@glo@type]{%
7706   \edef\gls@grplabel{#2}\protected@edef\gls@grptitle{#3}%
7707   \@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`

```
7708 \newcommand*{\glsnavhypertarget}[3][\@glo@type]{%
  Add this group to the aux file for re-run check.
7709   \protected@write\@auxout{}{\string\@gls@hypergroup{#1}{#2}}%
  Add the target.
7710   \@glstarget{glsn:#1@#2}{#3}%
  Check list of know groups to determine if a re-run is required.
7711   \expandafter\let
7712     \expandafter\@gls@list\csname @gls@hypergroup@list@#1\endcsname
  Iterate through list and terminate loop if this group is found.
7713   \@for\@gls@elem:=\@gls@list\do{%
7714     \ifthenelse{equal{\@gls@elem}{#2}}{\@endfortrue}{}}}
```

Check if list terminated prematurely.

```
7715 \if@endfor
7716 \else
```

This group was not included in the list, so issue a warning.

```
7717 \GlossariesWarningNoLine{Navigation panel
7718     for glossary type ‘#1’^^Jmissing group ‘#2’}%
7719 \gdef\gls@hypergrouprerun{%
7720     \GlossariesWarningNoLine{Navigation panel
7721     has changed. Rerun LaTeX}}%
7722 \fi
7723 }
```

hypergrouprerun Give a warning at the end if re-run required

```
7724 \let\gls@hypergrouprerun\relax
7725 \AtEndDocument{\gls@hypergrouprerun}
```

@gls@hypergroup This adds to (or creates) the command `\@gls@hypergrouplist@<glossary type>` which lists all groups for a given glossary, so that the navigation bar only contains those groups that are present. However it requires at least 2 runs to ensure the information is up-to-date.

```
7726 \newcommand*{\@gls@hypergroup}[2]{%
7727 \@ifundefined{\@gls@hypergrouplist@#1}{%
7728     \expandafter\xdef\csname @gls@hypergrouplist@#1\endcsname{#2}%
7729 }{%
7730     \expandafter\let\expandafter\@gls@tmp
7731     \csname @gls@hypergrouplist@#1\endcsname
7732     \expandafter\xdef\csname @gls@hypergrouplist@#1\endcsname{%
7733         \@gls@tmp,#2}%
7734 }%
7735 }
```

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`

```
7736 \newcommand*{\glsnavigation}{%
7737 \def\@gls@between{%
7738 \@ifundefined{\@gls@hypergrouplist@\@glo@type}{%
7739     \def\@gls@list{%
7740 }{%
7741     \expandafter\let\expandafter\@gls@list
7742     \csname @gls@hypergrouplist@\@glo@type\endcsname
7743 }%
7744 \@for\@gls@tmp:=\@gls@list\do{%
7745     \@gls@between
```

```

7746 \@gls@getgrouptitle{\@gls@tmp}{\@gls@grptitle}%
7747 \glsnavhyperlink{\@gls@tmp}{\@gls@grptitle}%
7748 \let\@gls@between\glshypernavsep%
7749 }%
7750 }

```

`\glshypernavsep` Separator for the hyper navigation bar.

```

7751 \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`

```

7752 \newcommand*{\glssymbolnav}{%
7753 \glsnavhyperlink{glssymbols}{\glsgetgrouptitle{glssymbols}}%
7754 \glshypernavsep
7755 \glsnavhyperlink{glsnumbers}{\glsgetgrouptitle{glsnumbers}}%
7756 \glshypernavsep
7757 }

```

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.

```

7758 \ProvidesPackage{glossary-inline}[2016/01/24 v4.21 (NLCT)]

```

`inline` Define the inline style.

```

7759 \newglossarystyle{inline}{%

```

Start of glossary sets up first empty separator between entries. (This is then changed by `\glossentry`)

```

7760 \renewenvironment{theglossary}%
7761   {%
7762     \def\gls@inlinesep{}%
7763     \def\gls@inlinesubsep{}%
7764     \def\gls@inlinepostchild{}%
7765   }%
7766   {\glspostinline}%

```

No header:

```

7767 \renewcommand*{\glossaryheader}{}%

```

No group headings (if heading is required, add `\glsinlinedopostchild` to start definition in case heading follows a child entry):

```

7768 \renewcommand*{\glsgroupheading}[1]{}%

```


Just display separator followed by name and description:

```
7769 \renewcommand{\glossentry}[2]{%
7770   \glsinlinedopostchild
7771   \gls@inlinesep
7772   \glsentryitem{##1}%
7773   \glsinlinenameformat{##1}{%
7774     \glossentryname{##1}%
7775   }%
7776   \ifglsdescsuppressed{##1}%
7777   {%
7778     \glsinlineemptydescformat
7779     {%
7780       \glossentrysymbol{##1}%
7781     }%
7782     {%
7783       ##2%
7784     }%
7785   }%
7786   {%
7787     \ifglshasdesc{##1}%
7788     {\glsinlinedescformat{\glossentrydesc{##1}}{\glossentrysymbol{##1}}{##2}}%
7789     {\glsinlineemptydescformat{\glossentrysymbol{##1}}{##2}}%
7790   }%
7791   \ifglshaschildren{##1}%
7792   {%
7793     \glsresetsubentrycounter
7794     \glsinlineparentchildseparator
7795     \def\gls@inlinesubsep{}%
7796     \def\gls@inlinepostchild{\glsinlinepostchild}%
7797   }%
7798   {}%
7799   \def\gls@inlinesep{\glsinlineseparator}%
7800 }%
```

Sub-entries display description:

```
7801 \renewcommand{\subglossentry}[3]{%
7802   \gls@inlinesubsep%
7803   \glsinlinesubnameformat{##2}{%
7804     \glossentryname{##2}}%
7805   \glssubentryitem{##2}%
7806   \glsinlinesubdescformat{\glossentrydesc{##2}}{\glossentrysymbol{##2}}{##3}%
7807   \def\gls@inlinesubsep{\glsinlinesubseparator}%
7808 }%
```

Nothing special between groups:

```
7809 \renewcommand*{\glsgroupskip}{}%
7810 }
```

linedopostchild

```
7811 \newcommand*{\glsinlinedopostchild}{%
```

```

7812 \gls@inlinepostchild
7813 \def\gls@inlinepostchild{}%
7814 }

```

`inlineseparator` Separator to use between entries.

```
7815 \newcommand*\glsinlineseparator}{;\space}
```

`inlinesubseparator` Separator to use between sub-entries.

```
7816 \newcommand*\glsinlinesubseparator}{,\space}
```

`parentchildseparator` Separator to use between parent and children.

```
7817 \newcommand*\glsinlineparentchildseparator}{:\space}
```

`inlinepostchild` Hook to use between child and next entry

```
7818 \newcommand*\glsinlinepostchild}{}
```

`\glspostinline` Terminator for inline glossary.

```
7819 \newcommand*\glspostinline}{\glspostdescription\space}
```

`inlinenameformat` Formats the name of the entry (first argument label, second argument name):

```
7820 \newcommand*\glsinlinenameformat}[2]{\glstarget{#1}{#2}}
```

`inlinedescformat` Formats the entry's description, symbol and location list:

```
7821 \newcommand*\glsinlinedescformat}[3]{\space#1}
```

`emptydescformat` Formats the entry's symbol and location list when the description is empty:

```
7822 \newcommand*\glsinlineemptydescformat}[2]{}
```

`inlinesubnameformat` Formats the name of the subentry (first argument label, second argument name):

```
7823 \newcommand*\glsinlinesubnameformat}[2]{\glstarget{#1}{}}
```

`inlinesubdescformat` Formats the subentry's description, symbol and location list:

```
7824 \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.

```
7825 \ProvidesPackage{glossary-list}[2016/01/24 v4.21 (NLCT)]
```

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```

7826 \providecommand{\indexspace}{%
7827 \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
7828 }

```

`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.

```
7829 \newglossarystyle{list}{%
```

Use description environment:

```
7830 \renewenvironment{theglossary}%
```

```
7831   {\begin{description}}{\end{description}}%
```

No header at the start of the environment:

```
7832 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
7833 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries start a new item in the list:

```
7834 \renewcommand*{\glossentry}[2]{%
```

```
7835   \item[\glsentryitem{##1}%
```

```
7836     \glstarget{##1}{\glossentryname{##1}}]
```

```
7837     \glossentrydesc{##1}\glspostdescription\space ##2}%
```

Sub-entries continue on the same line:

```
7838 \renewcommand*{\subglossentry}[3]{%
```

```
7839   \glssubentryitem{##2}%
```

```
7840   \glstarget{##2}{\strut}%
```

```
7841   \glossentrydesc{##2}\glspostdescription\space ##3.}%
```

```
7842 % \end{macrocode}
```

```
7843 % Add vertical space between groups:
```

```
7844 %\changes{3.03}{2012/09/21}{added check for glsnogroupskip}
```

```
7845 % \begin{macrocode}
```

```
7846 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
```

```
7847 }
```

`listgroup` The listgroup style is like the list style, but the glossary groups have headings.

```
7848 \newglossarystyle{listgroup}{%
```

Base it on the list style:

```
7849 \setglossarystyle{list}%
```

Each group has a heading:

```
7850 \renewcommand*{\glsgroupheading}[1]{\item[\glsgetgrouptitle{##1}]}
```

`listhypergroup` The listhypergroup style is like the listgroup style, but has a set of links to the groups at the start of the glossary.

```
7851 \newglossarystyle{listhypergroup}{%
```

Base it on the list style:

```
7852 \setglossarystyle{list}%
```

Add navigation links at the start of the environment:

```
7853 \renewcommand*{\glossaryheader}{%
7854   \item[\glsnavigation]}%
```

Each group has a heading with a hypertarget:

```
7855 \renewcommand*{\glsgroupheading}[1]{%
7856   \item[\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}]}
```

altlist The altlist glossary style is like the list style, but places the description on a new line. Sub-entries follow in separate paragraphs without the sub-entry name. This style does not use the entry's symbol.

```
7857 \newglossarystyle{altlist}{%
```

Base it on the list style:

```
7858 \setglossarystyle{list}%
```

Main (level 0) entries start a new item in the list with a line break after the entry name:

```
7859 \renewcommand*{\glossentry}[2]{%
7860   \item[\glsentryitem{##1}%
7861     \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.

```
7862   \mbox{}\par\nobreak\@afterheading
7863   \glossentrydesc{##1}\glspostdescription\space ##2}%
```

Sub-entries start a new paragraph:

```
7864 \renewcommand{\subglossentry}[3]{%
7865   \par
7866   \glsentryitem{##2}%
7867   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space ##3}%
7868 }
```

altlistgroup The altlistgroup glossary style is like the altlist style, but the glossary groups have headings.

```
7869 \newglossarystyle{altlistgroup}{%
```

Base it on the altlist style:

```
7870 \setglossarystyle{altlist}%
```

Each group has a heading:

```
7871 \renewcommand*{\glsgroupheading}[1]{\item[\glsgetgrouptitle{##1}]}
```

altlisthypergroup The altlisthypergroup glossary style is like the altlistgroup style, but has a set of links to the groups at the start of the glossary.

```
7872 \newglossarystyle{altlisthypergroup}{%
```

Base it on the altlist style:

```
7873 \setglossarystyle{altlist}%
```

Add navigation links at the start of the environment:

```
7874 \renewcommand*{\glossaryheader}{%
7875   \item[\glsnavigation]}%
```

Each group has a heading with a hypertarget:

```
7876 \renewcommand*{\glsgroupheading}[1]{%
7877   \item[\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}]}
```

`listdotted` The `listdotted` glossary style was supplied by Axel Menzel. I've modified it slightly so that the distance from the start of the name to the end of the dotted line is specified by `\glslistdottedwidth`. Note that this style ignores the page numbers as well as the symbol. Sub-entries are displayed in the same way as top-level entries.

```
7878 \newglossarystyle{listdotted}{%
```

Base it on the list style:

```
7879 \setglossarystyle{list}%
```

Each main (level 0) entry starts a new item:

```
7880 \renewcommand*{\glossentry}[2]{%
7881   \item[]\makebox[\glslistdottedwidth][l]{%
7882     \glsentryitem{##1}%
7883     \glstarget{##1}{\glossentryname{##1}}%
7884     \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##1}}%
```

Sub entries have the same format as main entries:

```
7885 \renewcommand*{\subglossentry}[3]{%
7886   \item[]\makebox[\glslistdottedwidth][l]{%
7887     \glssubentryitem{##2}%
7888     \glstarget{##2}{\glossentryname{##2}}%
7889     \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##2}}%
7890 }
```

`listdottedwidth`

```
7891 \newlength\glslistdottedwidth
7892 \setlength{\glslistdottedwidth}{.5\hsize}
```

`sublistdotted` This style is similar to the `glostylelistdotted` style, except that the main entries just have the name displayed.

```
7893 \newglossarystyle{sublistdotted}{%
```

Base it on the `listdotted` style:

```
7894 \setglossarystyle{listdotted}%
```

Main (level 0) entries just display the name:

```
7895 \renewcommand*{\glossentry}[2]{%
7896   \item[\glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}}}%
7897 }
```

3.4 Glossary Styles using `longtable` (the `glossary-long` package)

The glossary styles defined in the package used the `longtable` environment in the glossary.

```
7898 \ProvidesPackage{glossary-long}[2016/01/24 v4.21 (NLCT)]
```

Requires the package:

```
7899 \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`.)

```
7900 \@ifundefined{glsdescwidth}{%
7901   \newlength\glsdescwidth
7902   \setlength{\glsdescwidth}{0.6\hsize}
7903 }{}
```

`lspagelistwidth` This is a length that governs the width of the page list column.

```
7904 \@ifundefined{glspagelistwidth}{%
7905   \newlength\glspagelistwidth
7906   \setlength{\glspagelistwidth}{0.1\hsize}
7907 }{}
```

`long` The long glossary style command which uses the `longtable` environment:

```
7908 \newglossarystyle{long}{%
```

Use `longtable` with two columns:

```
7909   \renewenvironment{theglossary}%
7910     {\begin{longtable}{lp{\glsdescwidth}}%
7911     {\end{longtable}}%
```

Do nothing at the start of the environment:

```
7912   \renewcommand*{\glossaryheader}{}%
```

No heading between groups:

```
7913   \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries displayed in a row:

```
7914   \renewcommand{\glossentry}[2]{%
7915     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
7916     \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline
7917   }%
```

Sub entries displayed on the following row without the name:

```
7918   \renewcommand{\subglossentry}[3]{%
7919     &
7920     \glssubentryitem{##2}%
7921     \glstarget{##2}{\strut}\glosentrydesc{##2}\glspostdescription\space
7922     ##3\tabularnewline
7923   }%
```

Blank row between groups:

```
7924   \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else &
7925   \tabularnewline\fi}%
7926 }
```

`longborder` The `longborder` style is like the above, but with horizontal and vertical lines:

```
7927 \newglossarystyle{longborder}{%
```

Base it on the `glostylelong` style:

```
7928 \setglossarystyle{long}%
```

Use `longtable` with two columns with vertical lines between each column:

```
7929 \renewenvironment{theglossary}{%
```

```
7930 \begin{longtable}{|l|p{\glsdescwidth}|}{\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
7931 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
```

```
7932 }
```

`longheader` The `longheader` style is like the `long` style but with a header:

```
7933 \newglossarystyle{longheader}{%
```

Base it on the `glostylelong` style:

```
7934 \setglossarystyle{long}%
```

Set the table's header:

```
7935 \renewcommand*{\glossaryheader}{%
```

```
7936 \bfseries \entryname & \bfseries \descriptionname\tabularnewline\endhead}%
```

```
7937 }
```

`longheaderborder` The `longheaderborder` style is like the `long` style but with a header and border:

```
7938 \newglossarystyle{longheaderborder}{%
```

Base it on the `glostylelongborder` style:

```
7939 \setglossarystyle{longborder}%
```

Set the table's header and add horizontal line to table's foot:

```
7940 \renewcommand*{\glossaryheader}{%
```

```
7941 \hline\bfseries \entryname & \bfseries
```

```
7942 \descriptionname\tabularnewline\hline
```

```
7943 \endhead
```

```
7944 \hline\endfoot}%
```

```
7945 }
```

`long3col` The `long3col` style is like `long` but with 3 columns

```
7946 \newglossarystyle{long3col}{%
```

Use a `longtable` with 3 columns:

```
7947 \renewenvironment{theglossary}{%
```

```
7948 {\begin{longtable}{lp{\glsdescwidth}p{\glspagelistwidth}}}%
```

```
7949 {\end{longtable}}%
```

No table header:

```
7950 \renewcommand*{\glossaryheader}{}%
```

No headings between groups:

```
7951 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
7952 \renewcommand{\glossentry}[2]{%
7953   \glentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
7954   \glossentrydesc{##1} & ##2\tabularnewline
7955 }
```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```
7956 \renewcommand{\subglossentry}[3]{%
7957   &
7958   \glssubentryitem{##2}%
7959   \glstarget{##2}{\strut}\glossentrydesc{##2} &
7960   ##3\tabularnewline
7961 }
```

Blank row between groups:

```
7962 \renewcommand*{\glsgroupskip}{%
7963   \ifglsnogroupskip\else & \tabularnewline\fi}%
7964 }
```

`long3colborder` The `long3colborder` style is like the `long3col` style but with a border:

```
7965 \newglossarystyle{long3colborder}{%
```

Base it on the `glostylelong3col` style:

```
7966 \setglossarystyle{long3col}{%
```

Use a `longtable` with 3 columns with vertical lines around them:

```
7967 \renewenvironment{theglossary}{%
7968   {\begin{longtable}{|l|p{\glstdescwidth}|p{\glspagelistwidth}|}}%
7969   {\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
7970 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
7971 }
```

`long3colheader` The `long3colheader` style is like `long3col` but with a header row:

```
7972 \newglossarystyle{long3colheader}{%
```

Base it on the `glostylelong3col` style:

```
7973 \setglossarystyle{long3col}{%
```

Set the table's header:

```
7974 \renewcommand*{\glossaryheader}{%
7975   \bfseries\entryname&\bfseries\descriptionname&
7976   \bfseries\pagelistname\tabularnewline\endhead}%
7977 }
```

`colheaderborder` The `long3colheaderborder` style is like the above but with a border

```
7978 \newglossarystyle{long3colheaderborder}{%
```


Base it on the `glostylelong3colborder` style:

```
7979 \setglossarystyle{long3colborder}%
```

Set the table's header and add horizontal line at table's foot:

```
7980 \renewcommand*{\glossaryheader}{%
7981   \hline
7982   \bfseries\entryname&\bfseries\descriptionname&
7983   \bfseries\pagelistname\tabularnewline\hline\endhead
7984   \hline\endfoot}%
7985 }
```

`long4col` The `long4col` style has four columns where the third column contains the value of the associated symbol key.

```
7986 \newglossarystyle{long4col}{%
```

Use a `longtable` with 4 columns:

```
7987 \renewenvironment{theglossary}%
7988   {\begin{longtable}{l111}}%
7989   {\end{longtable}}%
```

No table header:

```
7990 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
7991 \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):

```
7992 \renewcommand{\glossentry}[2]{%
7993   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
7994   \glossentrydesc{##1} &
7995   \glossentrysymbol{##1} &
7996   ##2\tabularnewline
7997 }%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
7998 \renewcommand{\subglossentry}[3]{%
7999   &
8000   \glssubentryitem{##2}%
8001   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8002   \glossentrysymbol{##2} & ##3\tabularnewline
8003 }%
```

Blank row between groups:

```
8004 \renewcommand*{\glsgroupskip}{%
8005   \ifglsnogroupskip\else & & \tabularnewline\fi}%
8006 }
```

`long4colheader` The `long4colheader` style is like `long4col` but with a header row.

```
8007 \newglossarystyle{long4colheader}{%
```

Base it on the `glostylelong4col` style:

```
8008 \setglossarystyle{long4col}%
```

Table has a header:

```
8009 \renewcommand*{\glossaryheader}{%
8010   \bfseries\entryname&\bfseries\descriptionname&
8011   \bfseries \symbolname&
8012   \bfseries\pagelistname\tabularnewline\endhead}%
8013 }
```

`long4colborder` The `long4colborder` style is like `long4col` but with a border.

```
8014 \newglossarystyle{long4colborder}{%
```

Base it on the `glostylelong4col` style:

```
8015 \setglossarystyle{long4col}%
```

Use a longtable with 4 columns surrounded by vertical lines:

```
8016 \renewenvironment{theglossary}%
8017   {\begin{longtable}{|l|l|l|l|}}%
8018   {\end{longtable}}%
```

Add horizontal lines to the head and foot of the table:

```
8019 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8020 }
```

`colheaderborder` The `long4colheaderborder` style is like the above but with a border.

```
8021 \newglossarystyle{long4colheaderborder}{%
```

Base it on the `glostylelong4col` style:

```
8022 \setglossarystyle{long4col}%
```

Use a longtable with 4 columns surrounded by vertical lines:

```
8023 \renewenvironment{theglossary}%
8024   {\begin{longtable}{|l|l|l|l|}}%
8025   {\end{longtable}}%
```

Add table header and horizontal line at the table's foot:

```
8026 \renewcommand*{\glossaryheader}{%
8027   \hline\bfseries\entryname&\bfseries\descriptionname&
8028   \bfseries \symbolname&
8029   \bfseries\pagelistname\tabularnewline\hline\endhead
8030   \hline\endfoot}%
8031 }
```

`altlong4col` The `altlong4col` style is like the `long4col` style but can have multiline descriptions and page lists.

```
8032 \newglossarystyle{altlong4col}{%
```

Base it on the `glostylelong4col` style:

```
8033 \setglossarystyle{long4col}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8034 \renewenvironment{theglossary}%
8035   {\begin{longtable}{lp{\glsdescwidth}lp{\glspagelistwidth}}}%
8036   {\end{longtable}}%
8037 }
```

`altlong4colheader` The `altlong4colheader` style is like `altlong4col` but with a header row.

```
8038 \newglossarystyle{altlong4colheader}{%
```

Base it on the `glostylelong4colheader` style:

```
8039 \setglossarystyle{long4colheader}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8040 \renewenvironment{theglossary}%
8041   {\begin{longtable}{lp{\glsdescwidth}lp{\glspagelistwidth}}}%
8042   {\end{longtable}}%
8043 }
```

`altlong4colborder` The `altlong4colborder` style is like `altlong4col` but with a border.

```
8044 \newglossarystyle{altlong4colborder}{%
```

Base it on the `glostylelong4colborder` style:

```
8045 \setglossarystyle{long4colborder}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8046 \renewenvironment{theglossary}%
8047   {\begin{longtable}{|lp{\glsdescwidth}|lp{\glspagelistwidth}|}}%
8048   {\end{longtable}}%
8049 }
```

`altlong4colheaderborder` The `altlong4colheaderborder` style is like the above but with a header as well as a border.

```
8050 \newglossarystyle{altlong4colheaderborder}{%
```

Base it on the `glostylelong4colheaderborder` style:

```
8051 \setglossarystyle{long4colheaderborder}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8052 \renewenvironment{theglossary}%
8053   {\begin{longtable}{|lp{\glsdescwidth}|lp{\glspagelistwidth}|}}%
8054   {\end{longtable}}%
8055 }
```

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>

```
8056 \ProvidesPackage{glossary-longbooktabs}[2016/01/24 v4.21 (NLCT)]
```

Requires booktabs package:

```
8057 \RequirePackage{booktabs}
```

and the base packages for long styles:

```
8058 \RequirePackage{glossary-long}
```

```
8059 \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.

```
8060 \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.

```
8061 \glspatchLToutput
```

As with the longheader style, use the long style as a base.

```
8062 \setglossarystyle{long}{%
```

Add a header with rules.

```
8063 \renewcommand*{\glossaryheader}{%
```

```
8064 \toprule \bfseries \entryname & \bfseries
```

```
8065 \descriptionname\tabularnewline\midrule\endhead
```

```
8066 \bottomrule\endfoot}{%
```

Check for the nogroupskip package option. If there should be a gap between groups, insert the penalty and the vertical space.

```
8067 \renewcommand*{\glsgroupskip}{%
```

```
8068 \ifglsgroupskip \else \glspenaltygroupskip\fi}{%
```

```
8069 }
```

long3col-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.

```
8070 \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.

```
8071 \glspatchLToutput
```

Use the long3col style as a base.

```
8072 \setglossarystyle{long3col}{%
```

Add a header with rules.

```
8073 \renewcommand*{\glossaryheader}{%
8074   \toprule \bfseries \entryname &
8075   \bfseries \descriptionname &
8076   \bfseries \pagelistname
8077   \tabularnewline\midrule\endhead
8078   \bottomrule\endfoot}%
```

Check for the `nogroupskip` package option. If there should be a gap between groups, insert the penalty and the vertical space.

```
8079 \renewcommand*{\glsgroupskip}{%
8080   \ifglsnogroupskip \else \glspenaltygroupskip\fi}%
8081 }
```

`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.

```
8082 \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.

```
8083 \glspatchLToutput
```

Use the `long4col` style as a base.

```
8084 \setglossarystyle{long4col}%
```

Add a header with rules.

```
8085 \renewcommand*{\glossaryheader}{%
8086   \toprule \bfseries \entryname &
8087   \bfseries \descriptionname &
8088   \bfseries \symbolname &
8089   \bfseries \pagelistname
8090   \tabularnewline\midrule\endhead
8091   \bottomrule\endfoot}%
```

Check for the `nogroupskip` package option. If there should be a gap between groups, insert the penalty and the vertical space.

```
8092 \renewcommand*{\glsgroupskip}{%
8093   \ifglsnogroupskip \else \glspenaltygroupskip\fi}%
8094 }
```

`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.

```
8095 \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.

```
8096 \glspatchLToutput
```

Use the `long4col-booktabs` style as a base.

```
8097 \setglossarystyle{long4col-booktabs}%
```

Change the column specifications:

```
8098 \renewenvironment{theglossary}%  
8099   {\begin{longtable}{lp{\glsdescwidth}lp{\glspagelistwidth}}}%  
8100   {\end{longtable}}%  
8101 }
```

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.

```
8102 \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.

```
8103 \glspatchLToutput
```

Use the long-booktabs style as a base.

```
8104 \setglossarystyle{long-booktabs}%
```

Adjust the column specification.

```
8105 \renewenvironment{theglossary}%  
8106   {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}}}%  
8107   {\end{longtable}}%  
8108 }
```

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.

```
8109 \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.

```
8110 \glspatchLToutput
```

Use the long3col-booktabs style as a base.

```
8111 \setglossarystyle{long3col-booktabs}%
```

Adjust the column specification.

```
8112 \renewenvironment{theglossary}%  
8113   {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}%  
8114     >{\raggedright}p{\glspagelistwidth}}}%  
8115   {\end{longtable}}%  
8116 }
```

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.

```
8117 \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.

```
8118 \glspatchLToutput
```

Use the `altlong4col-booktabs` style as a base.

```
8119 \setglossarystyle{altlong4col-booktabs}%
```

Adjust the column specification.

```
8120 \renewenvironment{theglossary}%  
8121   {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}l%  
8122     >{\raggedright}p{\glspagelistwidth}}}%  
8123   {\end{longtable}}%  
8124 }
```

`sLTpenaltycheck`

```
8125 \newcommand*{\glsLTpenaltycheck}{%  
8126   \ifnum\outputpenalty=-50\vskip-\normalbaselineskip\relax\fi  
8127 }
```

`enaltygroupskip`

```
8128 \newcommand{\glspenaltygroupskip}{%  
8129   \noalign{\penalty-50\vskip\normalbaselineskip}}
```

`restoreLToutput` Provide a way of restoring `\LT@output` for the user.

```
8130 \let\@gls@org@LT@output\LT@output  
8131 \newcommand*{\glsrestoreLToutput}{\let\LT@output\@gls@org@LT@output}
```

This is David's patch, but I've replaced the hard-coded values with `\glsLTpenaltycheck` to make it easier to adjust.

`lspatchLToutput`

```
8132 \newcommand*{\glspatchLToutput}{%  
8133   \renewcommand*{\LT@output}{%  
8134     \ifnum\outputpenalty <-\@Mi  
8135       \ifnum\outputpenalty > -\LT@end@pen  
8136         \LT@err{floats and marginpars not allowed in a longtable}\@ehc  
8137       \else  
8138         \setbox\z@\vbox{\unvbox\@cclv}%  
8139         \ifdim \ht\LT@lastfoot>\ht\LT@foot  
8140           \dimen@\pagegoal  
8141           \advance\dimen@-\ht\LT@lastfoot  
8142           \ifdim\dimen@<\ht\z@  
8143             \setbox\@cclv\vbox{\unvbox\z@\copy\LT@foot\vss}%  
8144             \@makecol  
8145             \@outputpage  
8146             \setbox\z@\vbox{\box\LT@head\glsLTpenaltycheck}%  
8147           \fi  
8148         \fi  
8149         \global\@colroom\@colht  
8150         \global\@vsize\@colht  
8151         {\unvbox\z@\box\ifvoid\LT@lastfoot\LT@foot\else\LT@lastfoot\fi}%  
8152       \fi  
8153     \else
```

```

8154     \setbox\@cclv\vbox{\unvbox\@cclv\copy\LT@foot\vss}%
8155     \@makecol
8156     \@outputpage
8157     \global\ysize\@colroom
8158     \copy\LT@head
8159     \glsLTpenaltycheck
8160     \nobreak
8161     \fi
8162 }%
8163 }

```

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.

```
8164 \ProvidesPackage{glossary-longragged}[2016/01/24 v4.21 (NLCT)]
```

Requires the package:

```
8165 \RequirePackage{array}
```

Requires the package:

```
8166 \RequirePackage{longtable}
```

`\glsdescwidth` This is a length that governs the width of the description column. This may have already been defined.

```

8167 \@ifundefined{glsdescwidth}{%
8168   \newlength\glsdescwidth
8169   \setlength{\glsdescwidth}{0.6\hsize}
8170 }{}

```

`glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```

8171 \@ifundefined{glspagelistwidth}{%
8172   \newlength\glspagelistwidth
8173   \setlength{\glspagelistwidth}{0.1\hsize}
8174 }{}

```

`longragged` The longragged glossary style is like the long but uses ragged right formatting for the description column.

```
8175 \newglossarystyle{longragged}{%
```

Use longtable with two columns:

```

8176   \renewenvironment{theglossary}%
8177     {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}}}%
8178     {\end{longtable}}%

```

Do nothing at the start of the environment:

```
8179   \renewcommand*{\glossaryheader}{}%
```


No heading between groups:

```
8180 \renewcommand*\glsgroupheading}[1]{}
```

Main (level 0) entries displayed in a row:

```
8181 \renewcommand{\glossentry}[2]{%
8182   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8183   \glossentrydesc{##1}\glspostdescription\space ##2%
8184   \tabularnewline
8185 }
```

Sub entries displayed on the following row without the name:

```
8186 \renewcommand{\subglossentry}[3]{%
8187   &
8188   \glssubentryitem{##2}%
8189   \glstarget{##2}{\strut}\glossentrydesc{##2}%
8190   \glspostdescription\space ##3%
8191   \tabularnewline
8192 }
```

Blank row between groups:

```
8193 \renewcommand*\glsgroupskip{\ifglsnogroupskip\else & \tabularnewline\fi}%
8194 }
```

`longraggedborder` The `longraggedborder` style is like the above, but with horizontal and vertical lines:

```
8195 \newglossarystyle{longraggedborder}{%
```

Base it on the `glostylelongragged` style:

```
8196 \setglossarystyle{longragged}%
```

Use `longtable` with two columns with vertical lines between each column:

```
8197 \renewenvironment{theglossary}{%
8198   \begin{longtable}{|l|>{\raggedright}p{\glsdescwidth}|}%
8199   {\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
8200 \renewcommand*\glossaryheader{\hline\endhead\hline\endfoot}%
8201 }
```

`longraggedheader` The `longraggedheader` style is like the `longragged` style but with a header:

```
8202 \newglossarystyle{longraggedheader}{%
```

Base it on the `glostylelongragged` style:

```
8203 \setglossarystyle{longragged}%
```

Set the table's header:

```
8204 \renewcommand*\glossaryheader{%
8205   \bfseries \entryname & \bfseries \descriptionname
8206   \tabularnewline\endhead}%
8207 }
```

`longraggedheaderborder` The `longraggedheaderborder` style is like the `longragged` style but with a header and border:

```
8208 \newglossarystyle{longraggedheaderborder}{%
```

Base it on the `glostylelongraggedborder` style:

```
8209 \setglossarystyle{longraggedborder}%
```

Set the table's header and add horizontal line to table's foot:

```
8210 \renewcommand*{\glossaryheader}{%
8211   \hline\bfseries \entryname & \bfseries \descriptionname
8212   \tabularnewline\hline
8213   \endhead
8214   \hline\endfoot}%
8215 }
```

`longragged3col` The `longragged3col` style is like `longragged` but with 3 columns

```
8216 \newglossarystyle{longragged3col}{%
```

Use a longtable with 3 columns:

```
8217 \renewenvironment{theglossary}%
8218   {\begin{longtable}{1>{\raggedright}p{\glsdescwidth}%
8219     >{\raggedright}p{\glspagelistwidth}}}%
8220   {\end{longtable}}%
```

No table header:

```
8221 \renewcommand*{\glossaryheader}{}%
```

No headings between groups:

```
8222 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
8223 \renewcommand{\glossentry}[2]{%
8224   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8225   \glossentrydesc{##1} & ##2\tabularnewline
8226   }%
```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```
8227 \renewcommand{\subglossentry}[3]{%
8228   &
8229   \glsesubentryitem{##2}%
8230   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8231   ##3\tabularnewline
8232   }%
```

Blank row between groups:

```
8233 \renewcommand*{\glsgroupskip}{%
8234   \ifglsnogroupskip\else & &\tabularnewline\fi}%
8235 }
```

`ragged3colborder` The `longragged3colborder` style is like the `longragged3col` style but with a border:

```
8236 \newglossarystyle{longragged3colborder}{%
```

Base it on the `glostylelongragged3col` style:

```
8237 \setglossarystyle{longragged3col}%
```

Use a longtable with 3 columns with vertical lines around them:

```
8238 \renewenvironment{theglossary}%
8239   {\begin{longtable}{|l|>{\raggedright}p{\glsdescwidth}|}%
8240    >{\raggedright}p{\glspagelistwidth}|}%
8241   {\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
8242 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8243 }
```

`agged3colheader` The `longragged3colheader` style is like `longragged3col` but with a header row:

```
8244 \newglossarystyle{longragged3colheader}{%
```

Base it on the `glostylelongragged3col` style:

```
8245 \setglossarystyle{longragged3col}%
```

Set the table's header:

```
8246 \renewcommand*{\glossaryheader}{%
8247   \bfseries\entryname&\bfseries\descriptionname&
8248   \bfseries\pagelistname\tabularnewline\endhead}%
8249 }
```

`colheaderborder` The `longragged3colheaderborder` style is like the above but with a border

```
8250 \newglossarystyle{longragged3colheaderborder}{%
```

Base it on the `glostylelongragged3colborder` style:

```
8251 \setglossarystyle{longragged3colborder}%
```

Set the table's header and add horizontal line at table's foot:

```
8252 \renewcommand*{\glossaryheader}{%
8253   \hline
8254   \bfseries\entryname&\bfseries\descriptionname&
8255   \bfseries\pagelistname\tabularnewline\hline\endhead
8256   \hline\endfoot}%
8257 }
```

`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.

```
8258 \newglossarystyle{altlongragged4col}{%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8259 \renewenvironment{theglossary}%
8260   {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}l%
8261    >{\raggedright}p{\glspagelistwidth}}}%
8262   {\end{longtable}}%
```

No table header:

```
8263 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8264 \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):

```
8265 \renewcommand{\glossentry}[2]{%
8266   \glstarget{##1}{\glossentryname{##1}} &
8267   \glossentrydesc{##1} & \glossentrysymbol{##1} &
8268   ##2\tabularnewline
8269 }%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
8270 \renewcommand{\subglossentry}[3]{%
8271   &
8272   \glssubentryitem{##2}%
8273   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8274   \glossentrysymbol{##2} & ##3\tabularnewline
8275 }%
```

Blank row between groups:

```
8276 \renewcommand*{\glsgroupskip}{%
8277   \ifglsnogroupskip\else & & \tabularnewline\fi}%
8278 }
```

ragged4colheader The altlongragged4colheader style is like altlongragged4col but with a header row.

```
8279 \newglossarystyle{altlongragged4colheader}{%
```

Base it on the glostylealtlongragged4col style:

```
8280 \setglossarystyle{altlongragged4col}{%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8281 \renewenvironment{theglossary}%
8282   {\begin{longtable}[1>{\raggedright}p{\glsdescwidth}l%
8283     >{\raggedright}p{\glspagelistwidth}}}%
8284   {\end{longtable}}%
```

Table has a header:

```
8285 \renewcommand*{\glossaryheader}{%
8286   \bfseries\entryname&\bfseries\descriptionname&
8287   \bfseries \symbolname&
8288   \bfseries\pagelistname\tabularnewline\endhead}%
8289 }
```

ragged4colborder The altlongragged4colborder style is like altlongragged4col but with a border.

```
8290 \newglossarystyle{altlongragged4colborder}{%
```

Base it on the glostylealtlongragged4col style:

```
8291 \setglossarystyle{altlongragged4col}{%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8292 \renewenvironment{theglossary}%
```

```

8293   {\begin{longtable}{|l|>{\raggedright}p{\glsdescwidth}|l|}
8294     >{\raggedright}p{\glspagelistwidth}|}}%
8295   {\end{longtable}}%

```

Add horizontal lines to the head and foot of the table:

```

8296   \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8297 }

```

`colheaderborder` The `altlongragged4colheaderborder` style is like the above but with a header as well as a border.

```

8298 \newglossarystyle{altlongragged4colheaderborder}{%

```

Base it on the `glostylealtlongragged4col` style:

```

8299   \setglossarystyle{altlongragged4col}%

```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```

8300   \renewenvironment{theglossary}%
8301     {\begin{longtable}{|l|>{\raggedright}p{\glsdescwidth}|l|}
8302       >{\raggedright}p{\glspagelistwidth}|}}%
8303     {\end{longtable}}%

```

Add table header and horizontal line at the table's foot:

```

8304   \renewcommand*{\glossaryheader}{%
8305     \hline\bfseries\entryname&\bfseries\descriptionname&
8306     \bfseries \symbolname&
8307     \bfseries\pagelistname\@tabularnewline\hline\endhead
8308     \hline\endfoot}%
8309 }

```

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.

```

8310 \ProvidesPackage{glossary-mcols}[2016/01/24 v4.21 (NLCT)]

```

Required packages:

```

8311 \RequirePackage{multicol}
8312 \RequirePackage{glossary-tree}

```

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```

8313 \providecommand{\indexspace}{%
8314   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
8315 }

```

`\glscols` Define macro in which to store the number of columns. (Defaults to 2.)

```

8316 \newcommand*{\glscols}{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.)

```
8317 \newglossarystyle{mcolindex}{%
8318   \setglossarystyle{index}%
8319   \renewenvironment{theglossary}%
8320     {%
8321       \begin{multicols}{\glsmcols}
8322       \setlength{\parindent}{0pt}%
8323       \setlength{\parskip}{0pt plus 0.3pt}%
8324       \let\item\@idxitem}%
8325     {\end{multicols}}%
8326 }
```

`mcolindexgroup` As `mcolindex` but has headings:

```
8327 \newglossarystyle{mcolindexgroup}{%
8328   \setglossarystyle{mcolindex}%
8329   \renewcommand*{\glsgroupheading}[1]{%
8330     \item\textbf{\glsgetgrouptitle{##1}}\indexspace}%
8331 }
```

`indexhypergroup` The `mcolindexhypergroup` style is like the `mcolindexgroup` style but has hyper navigation.

```
8332 \newglossarystyle{mcolindexhypergroup}{%
  Base it on the glostylemcolindex style:
8333   \setglossarystyle{mcolindex}%
  Put navigation links to the groups at the start of the glossary:
8334   \renewcommand*{\glossaryheader}{%
8335     \item\textbf{\glsnavigation}\indexspace}%
  Add a heading for each group (with a target). The group's title is in bold followed by a vertical
  gap.
8336   \renewcommand*{\glsgroupheading}[1]{%
8337     \item\textbf{\glsnavhypertarget{##1}}{\glsgetgrouptitle{##1}}}%
8338     \indexspace}%
8339 }
```

`mcoltree` Multi-column index style. Same as the `tree`, but puts the glossary in multiple columns.

```
8340 \newglossarystyle{mcoltree}{%
8341   \setglossarystyle{tree}%
8342   \renewenvironment{theglossary}%
8343     {%
8344       \begin{multicols}{\glsmcols}
8345       \setlength{\parindent}{0pt}%
8346       \setlength{\parskip}{0pt plus 0.3pt}%
8347     }%
8348     {\end{multicols}}%
8349 }
```

`mcoltreegroup` Like the `mcoltree` style but the glossary groups have headings.

```

8350 \newglossarystyle{mcoltreegroup}{%
      Base it on the glostylemcoltree style:
8351 \setglossarystyle{mcoltree}%
      Each group has a heading (in bold) followed by a vertical gap):
8352 \renewcommand{\glsgroupheading}[1]{\par
8353 \noindent\textbf{\glsgetgrouptitle{##1}}\par\indexspace}%
8354 }

```

`treehypergroup` The `mcoltreehypergroup` style is like the `treegroup` style, but has a set of links to the groups at the start of the glossary.

```

8355 \newglossarystyle{mcoltreehypergroup}{%
      Base it on the glostylemcoltree style:
8356 \setglossarystyle{mcoltree}%
      Put navigation links to the groups at the start of the theglossary environment:
8357 \renewcommand*{\glossaryheader}{%
8358 \par\noindent\textbf{\glsnavigation}\par\indexspace}%
      Each group has a heading (in bold with a target) followed by a vertical gap):
8359 \renewcommand*{\glsgroupheading}[1]{%
8360 \par\noindent
8361 \textbf{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8362 \indexspace}%
8363 }

```

`mcoltreename` Multi-column index style. Same as the `treename`, but puts the glossary in multiple columns.

```

8364 \newglossarystyle{mcoltreename}{%
8365 \setglossarystyle{treename}%
8366 \renewenvironment{theglossary}%
8367 {%
8368 \begin{multicols}{\glscols}
8369 \setlength{\parindent}{0pt}%
8370 \setlength{\parskip}{0pt plus 0.3pt}%
8371 }%
8372 {\end{multicols}}%
8373 }

```

`treenamegroup` Like the `mcoltreename` style but the glossary groups have headings.

```

8374 \newglossarystyle{mcoltreenamegroup}{%
      Base it on the glostylemcoltreename style:
8375 \setglossarystyle{mcoltreename}%
      Give each group a heading:
8376 \renewcommand{\glsgroupheading}[1]{\par
8377 \noindent\textbf{\glsgetgrouptitle{##1}}\par\indexspace}%
8378 }

```

`onamehypergroup` The `mcoltreenonamehypergroup` style is like the `mcoltreenonamegroup` style, but has a set of links to the groups at the start of the glossary.

```
8379 \newglossarystyle{mcoltreenonamehypergroup}{%
```

Base it on the `glostylemcoltreenoname` style:

```
8380 \setglossarystyle{mcoltreenoname}%
```

Put navigation links to the groups at the start of the `theglossary` environment:

```
8381 \renewcommand*{\glossaryheader}{%
```

```
8382 \par\noindent\textbf{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
8383 \renewcommand*{\glsgroupheading}[1]{%
```

```
8384 \par\noindent
```

```
8385 \textbf{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
```

```
8386 \indexspace}%
```

```
8387 }
```

`mcolalmtree` Multi-column index style. Same as the `almtree`, but puts the glossary in multiple columns.

```
8388 \newglossarystyle{mcolalmtree}{%
```

```
8389 \setglossarystyle{almtree}%
```

```
8390 \renewenvironment{theglossary}%
```

```
8391 {%
```

```
8392 \begin{multicols}{\glscols}
```

```
8393 \def\@gls@prevlevel{-1}%
```

```
8394 \mbox{}\par
```

```
8395 }%
```

```
8396 {\par\end{multicols}}%
```

```
8397 }
```

`mcolalmtreegroup` Like the `mcolalmtree` style but the glossary groups have headings.

```
8398 \newglossarystyle{mcolalmtreegroup}{%
```

Base it on the `glostylemcolalmtree` style:

```
8399 \setglossarystyle{mcolalmtree}%
```

Give each group a heading.

```
8400 \renewcommand{\glsgroupheading}[1]{\par
```

```
8401 \def\@gls@prevlevel{-1}%
```

```
8402 \hangindent0pt\relax
```

```
8403 \parindent0pt\relax
```

```
8404 \textbf{\glsgetgrouptitle{##1}}\par\indexspace}%
```

```
8405 }
```

`almtreehypergroup` The `mcolalmtreehypergroup` style is like the `mcolalmtreegroup` style, but has a set of links to the groups at the start of the glossary.

```
8406 \newglossarystyle{mcolalmtreehypergroup}{%
```

Base it on the `glostylemcolalmtree` style:

```
8407 \setglossarystyle{mcolalmtree}%
```


Put the navigation links in the header

```
8408 \renewcommand*{\glossaryheader}{%
8409   \par
8410   \def\@gls@prevlevel{-1}%
8411   \hangindent0pt\relax
8412   \parindent0pt\relax
8413   \textbf{\glsnavigation}\par\indexspace}%
```

Put a hypertarget at the start of each group

```
8414 \renewcommand*{\glsgroupheading}[1]{%
8415   \par
8416   \def\@gls@prevlevel{-1}%
8417   \hangindent0pt\relax
8418   \parindent0pt\relax
8419   \textbf{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8420   \indexspace}}
```

3.8 Glossary Styles using supertabular environment (glossary-super package)

The glossary styles defined in the package use the supertabular environment.

```
8421 \ProvidesPackage{glossary-super}[2016/01/24 v4.21 (NLCT)]
```

Requires the package:

```
8422 \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.

```
8423 \@ifundefined{glsdescwidth}{%
8424   \newlength\glsdescwidth
8425   \setlength{\glsdescwidth}{0.6\hsize}
8426 }{}
```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined if has been loaded.

```
8427 \@ifundefined{glspagelistwidth}{%
8428   \newlength\glspagelistwidth
8429   \setlength{\glspagelistwidth}{0.1\hsize}
8430 }{}
```

`super` The super glossary style uses the supertabular environment (it uses lengths defined in the package.)

```
8431 \newglossarystyle{super}{%
```

Put the glossary in a supertabular environment with two columns and no head or tail:

```
8432   \renewenvironment{theglossary}%
8433     {\tablehead{}\tabletail{}}%
8434     \begin{supertabular}{lp{\glsdescwidth}}%
8435     {\end{supertabular}}%
```

Do nothing at the start of the table:

```
8436 \renewcommand*\glossaryheader{}%
```

No group headings:

```
8437 \renewcommand*\glsgroupheading}[1]{}%
```

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```
8438 \renewcommand\glossentry}[2]{%
8439   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8440   \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline
8441 }%
```

Sub entries put in a row (no name, description and page list in second column):

```
8442 \renewcommand\subglossentry}[3]{%
8443   &
8444   \glssubentryitem{##2}%
8445   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
8446   ##3\tabularnewline
8447 }%
```

Blank row between groups:

```
8448 \renewcommand*\glsgroupskip{}%
8449   \ifglsnogroupskip\else & \tabularnewline\fi}%
8450 }
```

superborder The superborder style is like the above, but with horizontal and vertical lines:

```
8451 \newglossarystyle{superborder}{}%
```

Base it on the `glostylesuper` style:

```
8452 \setglossarystyle{super}{}%
```

Put the glossary in a `supertabular` environment with two columns and a horizontal line in the head and tail:

```
8453 \renewenvironment{theglossary}%
8454   {\tablehead{\hline}\tabletail{\hline}%
8455   \begin{supertabular}{|l|p{\glsdescwidth}|}}%
8456   {\end{supertabular}}%
8457 }
```

superheader The superheader style is like the super style, but with a header:

```
8458 \newglossarystyle{superheader}{}%
```

Base it on the `glostylesuper` style:

```
8459 \setglossarystyle{super}{}%
```

Put the glossary in a `supertabular` environment with two columns, a header and no tail:

```
8460 \renewenvironment{theglossary}%
8461   {\tablehead{\bfseries \entryname &
8462   \bfseries\descriptionname\tabularnewline}%
8463   \tabletail{}}%
8464   \begin{supertabular}{lp{\glsdescwidth}}%
```

```
8465 {\end{supertabular}}}%
8466 }
```

perheaderborder The superheaderborder style is like the super style but with a header and border:

```
8467 \newglossarystyle{superheaderborder}{%
```

Base it on the glostylesuper style:

```
8468 \setglossarystyle{super}%
```

Put the glossary in a supertabular environment with two columns, a header and horizontal lines above and below the table:

```
8469 \renewenvironment{theglossary}%
8470 {\tablehead{\hline\bfseries \entryname &
8471 \bfseries \descriptionname\tabularnewline\hline}%
8472 \tabletail{\hline}
8473 \begin{supertabular}{|l|p{\glsdescwidth}|}%
8474 {\end{supertabular}}}%
8475 }
```

super3col The super3col style is like the super style, but with 3 columns:

```
8476 \newglossarystyle{super3col}{%
```

Put the glossary in a supertabular environment with three columns and no head or tail:

```
8477 \renewenvironment{theglossary}%
8478 {\tablehead{}\tabletail{}}%
8479 \begin{supertabular}{|lp{\glsdescwidth}p{\glspagelistwidth}|}%
8480 {\end{supertabular}}%
```

Do nothing at the start of the table:

```
8481 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8482 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
8483 \renewcommand{\glossentry}[2]{%
8484 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8485 \glossentrydesc{##1} & ##2\tabularnewline
8486 }%
```

Sub entries on a row (no name, description in second column, page list in last column):

```
8487 \renewcommand{\subglossentry}[3]{%
8488 &
8489 \glssubentryitem{##2}%
8490 \glstarget{##2}{\strut}\glossentrydesc{##2} &
8491 ##3\tabularnewline
8492 }%
```

Blank row between groups:

```
8493 \renewcommand*{\glsgroupskip}{%
8494 \ifglsnogroupskip\else & \tabularnewline\fi}%
8495 }
```

`super3colborder` The `super3colborder` style is like the `super3col` style, but with a border:

```
8496 \newglossarystyle{super3colborder}{%
```

Base it on the `glostylesuper3col` style:

```
8497 \setglossarystyle{super3col}{%
```

Put the glossary in a `supertabular` environment with three columns and a horizontal line in the head and tail:

```
8498 \renewenvironment{theglossary}{%
```

```
8499 {\tablehead{\hline}\tabletail{\hline}%
```

```
8500 \begin{supertabular}{|l|p{\glsdescwidth}|p{\glspagelistwidth}|}%
```

```
8501 {\end{supertabular}}%
```

```
8502 }
```

`super3colheader` The `super3colheader` style is like the `super3col` style but with a header row:

```
8503 \newglossarystyle{super3colheader}{%
```

Base it on the `glostylesuper3col` style:

```
8504 \setglossarystyle{super3col}{%
```

Put the glossary in a `supertabular` environment with three columns, a header and no tail:

```
8505 \renewenvironment{theglossary}{%
```

```
8506 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
```

```
8507 \bfseries\pagelistname\tabularnewline}\tabletail{}}%
```

```
8508 \begin{supertabular}{lp{\glsdescwidth}p{\glspagelistwidth}}%
```

```
8509 {\end{supertabular}}%
```

```
8510 }
```

`colheaderborder` The `super3colheaderborder` style is like the `super3col` style but with a header and border:

```
8511 \newglossarystyle{super3colheaderborder}{%
```

Base it on the `glostylesuper3colborder` style:

```
8512 \setglossarystyle{super3colborder}{%
```

Put the glossary in a `supertabular` environment with three columns, a header with horizontal lines and a horizontal line in the tail:

```
8513 \renewenvironment{theglossary}{%
```

```
8514 {\tablehead{\hline
```

```
8515 \bfseries\entryname&\bfseries\descriptionname&
```

```
8516 \bfseries\pagelistname\tabularnewline\hline}%
```

```
8517 \tabletail{\hline}%
```

```
8518 \begin{supertabular}{|l|p{\glsdescwidth}|p{\glspagelistwidth}|}%
```

```
8519 {\end{supertabular}}%
```

```
8520 }
```

`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.

```
8521 \newglossarystyle{super4col}{%
```

Put the glossary in a supertabular environment with four columns and no head or tail:

```
8522 \renewenvironment{theglossary}%  
8523   {\tablehead{\tabletail}{}%  
8524     \begin{supertabular}{1111}}{%  
8525     \end{supertabular}}%
```

Do nothing at the start of the table:

```
8526 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8527 \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:

```
8528 \renewcommand{\glossentry}[2]{%  
8529   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &  
8530   \glossentrydesc{##1} &  
8531   \glossentrysymbol{##1} & ##3\tabularnewline  
8532 }%
```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```
8533 \renewcommand{\subglossentry}[3]{%  
8534   &  
8535   \glssubentryitem{##2}%  
8536   \glstarget{##2}{\strut}\glossentrydesc{##2} &  
8537   \glossentrysymbol{##2} & ##3\tabularnewline  
8538 }%
```

Blank row between groups:

```
8539 \renewcommand*{\glsgroupskip}{%  
8540   \ifglsnogroupskip\else & & \tabularnewline\fi}%  
8541 }
```

super4colheader The `super4colheader` style is like the `super4col` but with a header row.

```
8542 \newglossarystyle{super4colheader}{%
```

Base it on the `glostylesuper4col` style:

```
8543 \setglossarystyle{super4col}%
```

Put the glossary in a supertabular environment with four columns, a header and no tail:

```
8544 \renewenvironment{theglossary}%  
8545   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&  
8546     \bfseries\symbolname &  
8547     \bfseries\pagelistname\tabularnewline}%  
8548   \tabletail}{%  
8549   \begin{supertabular}{1111}}{%  
8550   \end{supertabular}}%  
8551 }
```

super4colborder The `super4colborder` style is like the `super4col` but with a border.

```
8552 \newglossarystyle{super4colborder}{%
```

Base it on the `glostylesuper4col` style:

```
8553 \setglossarystyle{super4col}%
```

Put the glossary in a `supertabular` environment with four columns and a horizontal line in the head and tail:

```
8554 \renewenvironment{theglossary}%
8555   {\tablehead{\hline}\tabletail{\hline}%
8556   \begin{supertabular}{|l|l|l|l|}}%
8557   {\end{supertabular}}%
8558 }
```

`colheaderborder` The `super4colheaderborder` style is like the `super4col` but with a header and border.

```
8559 \newglossarystyle{super4colheaderborder}{%
```

Base it on the `glostylesuper4col` style:

```
8560 \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:

```
8561 \renewenvironment{theglossary}%
8562   {\tablehead{\hline\bfseries\entryname&\bfseries\descriptionname&
8563   \bfseries\symbolname &
8564   \bfseries\pagelistname\tabularnewline\hline}%
8565   \tabletail{\hline}%
8566   \begin{supertabular}{|l|l|l|l|}}%
8567   {\end{supertabular}}%
8568 }
```

`altsuper4col` The `altsuper4col` glossary style is like `super4col` but has provision for multiline descriptions.

```
8569 \newglossarystyle{altsuper4col}{%
```

Base it on the `glostylesuper4col` style:

```
8570 \setglossarystyle{super4col}%
```

Put the glossary in a `supertabular` environment with four columns and no head or tail:

```
8571 \renewenvironment{theglossary}%
8572   {\tablehead{}\tabletail{}}%
8573   \begin{supertabular}{lp{\glsdescwidth}lp{\glspagelistwidth}}}%
8574   {\end{supertabular}}%
8575 }
```

`super4colheader` The `altsuper4colheader` style is like the `altsuper4col` but with a header row.

```
8576 \newglossarystyle{altsuper4colheader}{%
```

Base it on the `glostylesuper4colheader` style:

```
8577 \setglossarystyle{super4colheader}%
```

Put the glossary in a `supertabular` environment with four columns, a header and no tail:

```
8578 \renewenvironment{theglossary}%
8579   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8580   \bfseries\symbolname &
```

```

8581     \bfseries\pagelistname\tabularnewline\tabletail{}}%
8582     \begin{supertabular}{lp{\glsdescwidth}lp{\glspagelistwidth}}}%
8583     {\end{supertabular}}}%
8584 }

```

`super4colborder` The `altsuper4colborder` style is like the `altsuper4col` but with a border.

```

8585 \newglossarystyle{altsuper4colborder}{%
      Base it on the glostylesuper4colborder style:
8586   \setglossarystyle{super4colborder}%
      Put the glossary in a supertabular environment with four columns and a horizontal line in the
      head and tail:
8587   \renewenvironment{theglossary}%
8588     {\tablehead{\hline}\tabletail{\hline}%
8589     \begin{supertabular}%
8590       {lllp{\glsdescwidth}lllp{\glspagelistwidth}}}%
8591     {\end{supertabular}}}%
8592 }

```

`colheaderborder` The `altsuper4colheaderborder` style is like the `altsuper4col` but with a header and border.

```

8593 \newglossarystyle{altsuper4colheaderborder}{%
      Base it on the glostylesuper4colheaderborder style:
8594   \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:
8595   \renewenvironment{theglossary}%
8596     {\tablehead{\hline
8597       \bfseries\entryname &
8598       \bfseries\descriptionname &
8599       \bfseries\symbolname &
8600       \bfseries\pagelistname\tabularnewline\hline}%
8601     \tabletail{\hline}%
8602     \begin{supertabular}%
8603       {lllp{\glsdescwidth}lllp{\glspagelistwidth}}}%
8604     {\end{supertabular}}}%
8605 }

```

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.

```

8606 \ProvidesPackage{glossary-superragged}[2016/01/24 v4.21 (NLCT)]

```

Requires the package:

```
8607 \RequirePackage{array}
```

Requires the package:

```
8608 \RequirePackage{supertabular}
```

`\glsdescwidth` This is a length that governs the width of the description column. This may already have been defined.

```
8609 \@ifundefined{glsdescwidth}{%
8610   \newlength\glsdescwidth
8611   \setlength{\glsdescwidth}{0.6\hsize}
8612 }{}
```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```
8613 \@ifundefined{glspagelistwidth}{%
8614   \newlength\glspagelistwidth
8615   \setlength{\glspagelistwidth}{0.1\hsize}
8616 }{}
```

`superragged` The superragged glossary style uses the supertabular environment.

```
8617 \newglossarystyle{superragged}{%
```

Put the glossary in a supertabular environment with two columns and no head or tail:

```
8618   \renewenvironment{theglossary}{%
8619     {\tablehead{}\tabletail{}}%
8620     \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}}}%
8621     {\end{supertabular}}%
```

Do nothing at the start of the table:

```
8622   \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8623   \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```
8624   \renewcommand{\glossentry}[2]{%
8625     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8626     \glossentrydesc{##1}\glspostdescription\space ##2%
8627     \tabularnewline
8628   }%
```

Sub entries put in a row (no name, description and page list in second column):

```
8629   \renewcommand{\subglossentry}[3]{%
8630     &
8631     \glssubentryitem{##2}%
8632     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
8633     ##3%
8634     \tabularnewline
8635   }%
```


Blank row between groups:

```
8636 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else & \tabularnewline\fi}%  
8637 }
```

`superraggedborder` The `superraggedborder` style is like the above, but with horizontal and vertical lines:

```
8638 \newglossarystyle{superraggedborder}{%
```

Base it on the `glostylesuperragged` style:

```
8639 \setglossarystyle{superragged}%
```

Put the glossary in a `supertabular` environment with two columns and a horizontal line in the head and tail:

```
8640 \renewenvironment{theglossary}%  
8641 {\tablehead{\hline}\tabletail{\hline}%  
8642 \begin{supertabular}{|l|>{\raggedright}p{\glsdescwidth}|}}%  
8643 {\end{supertabular}}%  
8644 }
```

`superraggedheader` The `superraggedheader` style is like the `super` style, but with a header:

```
8645 \newglossarystyle{superraggedheader}{%
```

Base it on the `glostylesuperragged` style:

```
8646 \setglossarystyle{superragged}%
```

Put the glossary in a `supertabular` environment with two columns, a header and no tail:

```
8647 \renewenvironment{theglossary}%  
8648 {\tablehead{\bfseries \entryname & \bfseries \descriptionname  
8649 \tabularnewline}%  
8650 \tabletail{}}%  
8651 \begin{supertabular}{|l>{\raggedright}p{\glsdescwidth}|}}%  
8652 {\end{supertabular}}%  
8653 }
```

`superraggedheaderborder` The `superraggedheaderborder` style is like the `superragged` style but with a header and border:

```
8654 \newglossarystyle{superraggedheaderborder}{%
```

Base it on the `glostylesuper` style:

```
8655 \setglossarystyle{superragged}%
```

Put the glossary in a `supertabular` environment with two columns, a header and horizontal lines above and below the table:

```
8656 \renewenvironment{theglossary}%  
8657 {\tablehead{\hline\bfseries \entryname &  
8658 \bfseries \descriptionname\hline}\tabletail{\hline}%  
8659 \begin{supertabular}{|l|>{\raggedright}p{\glsdescwidth}|}}%  
8660 {\end{supertabular}}%  
8661 {\end{supertabular}}%  
8662 }
```

`superragged3col` The `superragged3col` style is like the `superragged` style, but with 3 columns:

```
8663 \newglossarystyle{superragged3col}{%
```

Put the glossary in a supertabular environment with three columns and no head or tail:

```
8664 \renewenvironment{theglossary}%  
8665   {\tablehead{}\tabletail{}}%  
8666   \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}%  
8667     >{\raggedright}p{\glspagelistwidth}}}%  
8668   {\end{supertabular}}%
```

Do nothing at the start of the table:

```
8669 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8670 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
8671 \renewcommand{\glossentry}[2]{%  
8672   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &  
8673   \glossentrydesc{##1} &  
8674   ##2\tabularnewline  
8675 }%
```

Sub entries on a row (no name, description in second column, page list in last column):

```
8676 \renewcommand{\subglossentry}[3]{%  
8677   &  
8678   \glssubentryitem{##2}%  
8679   \glstarget{##2}{\strut}\glossentrydesc{##2} &  
8680   ##3\tabularnewline  
8681 }%
```

Blank row between groups:

```
8682 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else & &\tabularnewline\fi}%  
8683 }
```

agged3colborder The superragged3colborder style is like the superragged3col style, but with a border:

```
8684 \newglossarystyle{superragged3colborder}{%
```

Base it on the glostylesuperragged3col style:

```
8685 \setglossarystyle{superragged3col}%
```

Put the glossary in a supertabular environment with three columns and a horizontal line in the head and tail:

```
8686 \renewenvironment{theglossary}%  
8687   {\tablehead{\hline}\tabletail{\hline}%  
8688   \begin{supertabular}{|l|>{\raggedright}p{\glsdescwidth}|%  
8689     >{\raggedright}p{\glspagelistwidth}|}}%  
8690   {\end{supertabular}}%  
8691 }
```

agged3colheader The superragged3colheader style is like the superragged3col style but with a header row:

```
8692 \newglossarystyle{superragged3colheader}{%
```

Base it on the `glostylesuperragged3col` style:

```
8693 \setglossarystyle{superragged3col}%
```

Put the glossary in a `supertabular` environment with three columns, a header and no tail:

```
8694 \renewenvironment{theglossary}%
8695   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8696     \bfseries\pagelistname\tabularnewline}\tabletail{}}%
8697   \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}%
8698     >{\raggedright}p{\glspagelistwidth}}%
8699   {\end{supertabular}}%
8700 }
```

`colheaderborder` The `superragged3colheaderborder` style is like the `superragged3col` style but with a header and border:

```
8701 \newglossarystyle{superragged3colheaderborder}{%
```

Base it on the `glostylesuperragged3colborder` style:

```
8702 \setglossarystyle{superragged3colborder}%
```

Put the glossary in a `supertabular` environment with three columns, a header with horizontal lines and a horizontal line in the tail:

```
8703 \renewenvironment{theglossary}%
8704   {\tablehead{\hline
8705     \bfseries\entryname&\bfseries\descriptionname&
8706     \bfseries\pagelistname\tabularnewline\hline}%
8707   \tabletail{\hline}%
8708   \begin{supertabular}{|l|>{\raggedright}p{\glsdescwidth}|%
8709     >{\raggedright}p{\glspagelistwidth}|}%
8710   {\end{supertabular}}%
8711 }
```

`superragged4col` The `altsuperragged4col` glossary style is like `altsuper4col` style in the package but uses ragged right formatting in the description and page list columns.

```
8712 \newglossarystyle{altsuperragged4col}{%
```

Put the glossary in a `supertabular` environment with four columns and no head or tail:

```
8713 \renewenvironment{theglossary}%
8714   {\tablehead{} \tabletail{}}%
8715   \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}l%
8716     >{\raggedright}p{\glspagelistwidth}}%
8717   {\end{supertabular}}%
```

Do nothing at the start of the table:

```
8718 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8719 \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:

```
8720 \renewcommand{\glosseentry}[2]{}%
```

```

8721 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8722 \glossentrydesc{##1} &
8723 \glossentrysymbol{##1} & ##2\tabularnewline
8724 }%

```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```

8725 \renewcommand{\subglossentry}[3]{%
8726   &
8727   \glssubentryitem{##2}%
8728   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8729   \glossentrysymbol{##2} & ##3\tabularnewline
8730 }%

```

Blank row between groups:

```

8731 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else & & &\tabularnewline\fi}%
8732 }

```

`ragged4colheader` The `altsuperragged4colheader` style is like the `altsuperragged4col` style but with a header row.

```

8733 \newglossarystyle{altsuperragged4colheader}{%

```

Base it on the `glostylealtsuperragged4col` style:

```

8734 \setglossarystyle{altsuperragged4col}%

```

Put the glossary in a `supertabular` environment with four columns, a header and no tail:

```

8735 \renewenvironment{theglossary}%
8736   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8737     \bfseries\symbolname &
8738     \bfseries\pagelistname\tabularnewline}\tabletail{}}%
8739   \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}l%
8740     >{\raggedright}p{\glspagelistwidth}}}%
8741   {\end{supertabular}}%
8742 }

```

`ragged4colborder` The `altsuperragged4colborder` style is like the `altsuperragged4col` style but with a border.

```

8743 \newglossarystyle{altsuperragged4colborder}{%

```

Base it on the `glostylealtsuperragged4col` style:

```

8744 \setglossarystyle{altsuper4col}%

```

Put the glossary in a `supertabular` environment with four columns and a horizontal line in the head and tail:

```

8745 \renewenvironment{theglossary}%
8746   {\tablehead{\hline}\tabletail{\hline}%
8747   \begin{supertabular}%
8748     {l|>{\raggedright}p{\glsdescwidth}l|}%
8749     >{\raggedright}p{\glspagelistwidth}|}}%
8750   {\end{supertabular}}%
8751 }

```

`colheaderborder` The `altsuperragged4colheaderborder` style is like the `altsuperragged4col` style but with a header and border.

```
8752 \newglossarystyle{altsuperragged4colheaderborder}{%
```

Base it on the `glostylealtsuperragged4col` style:

```
8753 \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:

```
8754 \renewenvironment{theglossary}{%
8755   {\tablehead{\hline
8756     \bfseries\entryname &
8757     \bfseries\descriptionname &
8758     \bfseries\symbolname &
8759     \bfseries\pagelistname\tabularnewline\hline}%
8760   \tabletail{\hline}%
8761   \begin{supertabular}%
8762     {|l|>{\raggedright}p{\glsdescwidth}|l|}%
8763     >{\raggedright}p{\glspagelistwidth}|}}%
8764   {\end{supertabular}}%
8765 }
```

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.

```
8766 \ProvidesPackage{glossary-tree}[2016/01/24 v4.21 (NLCT)]
```

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```
8767 \providecommand{\indexspace}{%
8768   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
8769 }
```

`\glstreenamefmt` Format used to display the name in the tree styles. (This may be counteracted by `\glstnamefont`.) This command is also used to format the group headings.

```
8770 \newcommand*{\glstreenamefmt}[1]{\textbf{#1}}
```

`index` The index glossary style is similar in style to the way indices are usually typeset using `\item`, `\subitem` and `\subsubitem`. The entry name is set in bold. If an entry has a symbol, it is placed in brackets after the name. Then the description is displayed, followed by the number list. This style allows up to three levels.

```
8771 \newglossarystyle{index}{%
```

Set the paragraph indentation and skip and define `\item` to be the same as that used by `theindex`:

```
8772 \renewenvironment{theglossary}{%
```

```

8773   {\setlength{\parindent}{0pt}}%
8774   \setlength{\parskip}{0pt plus 0.3pt}}%
8775   \let\item\@idxitem}}%

8776   {\par}}%

```

Do nothing at the start of the environment:

```
8777 \renewcommand*\glossaryheader}{}}%
```

No group headers:

```
8778 \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.

```

8779 \renewcommand*\glossentry}[2]{%
8780   \item\glsentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}}%
8781   \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}}%
8782   \space \glossentrydesc{##1}\glspostdescription\space ##2%
8783 }%

```

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.

```

8784 \renewcommand*\subglossentry}[3]{%
8785   \ifcase##1\relax
8786     % level 0
8787     \item
8788   \or
8789     % level 1
8790     \subitem
8791     \glssubentryitem{##2}}%
8792   \else
8793     % all other levels
8794     \subsubitem
8795   \fi
8796   \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}}%
8797   \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}}%
8798   \space\glossentrydesc{##2}\glspostdescription\space ##3%
8799 }%

```

Vertical gap between groups is the same as that used by indices:

```
8800 \renewcommand*\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}}
```

`indexgroup` The `indexgroup` style is like the `index` style but has headings.

```
8801 \newglossarystyle{indexgroup}{%
```

Base it on the `glostyleindex` style:

```
8802 \setglossarystyle{index}}%
```

Add a heading for each group. This puts the group's title in bold followed by a vertical gap.

```
8803 \renewcommand*{\glsgroupheading}[1]{%
8804   \item\glstreenamefmt{\glsgetgrouptitle{##1}}\indexspace}%
8805 }
```

`indexhypergroup` The `indexhypergroup` style is like the `indexgroup` style but has hyper navigation.

```
8806 \newglossarystyle{indexhypergroup}{%
```

Base it on the `glostyleindex` style:

```
8807 \setglossarystyle{index}%
```

Put navigation links to the groups at the start of the glossary:

```
8808 \renewcommand*{\glossaryheader}{%
8809   \item\glstreenamefmt{\glsnavigation}\indexspace}%
```

Add a heading for each group (with a target). The group's title is in bold followed by a vertical gap.

```
8810 \renewcommand*{\glsgroupheading}[1]{%
8811   \item\glstreenamefmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
8812   \indexspace}%
8813 }
```

`tree` The tree glossary style is similar in style to the `index` style, but can have arbitrary levels.

```
8814 \newglossarystyle{tree}{%
```

Set the paragraph indentation and skip:

```
8815 \renewenvironment{theglossary}%
8816   {\setlength{\parindent}{0pt}%
8817   \setlength{\parskip}{0pt plus 0.3pt}}%
8818   {}%
```

Do nothing at the start of the `theglossary` environment:

```
8819 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8820 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries: name in bold, followed by symbol in brackets (if it exists), the description and the page list:

```
8821 \renewcommand{\glossentry}[2]{%
8822   \hangindent0pt\relax
8823   \parindent0pt\relax
8824   \glsentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
8825   \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
8826   \space\glossentrydesc{##1}\glspostdescription\space##2\par
8827   }%
```

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.

```
8828 \renewcommand{\subglossentry}[3]{%
8829   \hangindent##1\glstreeindent\relax
```

```

8830 \parindent##1\glstreeindent\relax
8831 \ifnum##1=1\relax
8832 \glssubentryitem{##2}%
8833 \fi
8834 \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
8835 \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
8836 \space\glossentrydesc{##2}\glspostdescription\space ##3\par
8837 }%

```

Vertical gap between groups is the same as that used by indices:

```

8838 \renewcommand*\glsgroupskip{\ifglsnogroupskip\else\indexspace\fi}

```

treegroup Like the tree style but the glossary groups have headings.

```

8839 \newglossarystyle{treegroup}{%

```

Base it on the `glostyletree` style:

```

8840 \setglossarystyle{tree}%

```

Each group has a heading (in bold) followed by a vertical gap):

```

8841 \renewcommand{\glsgroupheading}[1]{\par
8842 \noindent\glstreenamefmt{\glsgrouptitle{##1}}\par\indexspace}%
8843 }

```

treehypergroup The `treehypergroup` style is like the `treegroup` style, but has a set of links to the groups at the start of the glossary.

```

8844 \newglossarystyle{treehypergroup}{%

```

Base it on the `glostyletree` style:

```

8845 \setglossarystyle{tree}%

```

Put navigation links to the groups at the start of the `theglossary` environment:

```

8846 \renewcommand*\glossaryheader{%
8847 \par\noindent\glstreenamefmt{\glsnavigation}\par\indexspace}%

```

Each group has a heading (in bold with a target) followed by a vertical gap):

```

8848 \renewcommand*\glsgroupheading[1]{%
8849 \par\noindent
8850 \glstreenamefmt{\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}\par
8851 \indexspace}%
8852 }

```

\glstreeindent Length governing left indent for each level of the tree style.

```

8853 \newlength\glstreeindent
8854 \setlength{\glstreeindent}{10pt}

```

treenoname The `treenoname` glossary style is like the tree style, but doesn't print the name or symbol for sub-levels.

```

8855 \newglossarystyle{treenoname}{%

```


Set the paragraph indentation and skip:

```
8856 \renewenvironment{theglossary}%  
8857   {\setlength{\parindent}{0pt}%  
8858    \setlength{\parskip}{0pt plus 0.3pt}}%  
8859   {}%
```

No header:

```
8860 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8861 \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.

```
8862 \renewcommand{\glossentry}[2]{%  
8863   \hangindent0pt\relax  
8864   \parindent0pt\relax  
8865   \glsentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%  
8866   \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%  
8867   \space\glossentrydesc{##1}\glspostdescription\space##2\par  
8868 }%
```

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.

```
8869 \renewcommand{\subglossentry}[3]{%  
8870   \hangindent##1\glstreeindent\relax  
8871   \parindent##1\glstreeindent\relax  
8872   \ifnum##1=1\relax  
8873     \glssubentryitem{##2}%  
8874     \fi  
8875     \glstarget{##2}{\strut}%  
8876     \glossentrydesc{##2}\glspostdescription\space##3\par  
8877 }%
```

Vertical gap between groups is the same as that used by indices:

```
8878 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%  
8879 }
```

`treenonamegroup` Like the `treenoname` style but the glossary groups have headings.

```
8880 \newglossarystyle{treenonamegroup}{%
```

Base it on the `glostyletreenoname` style:

```
8881 \setglossarystyle{treenoname}%
```

Give each group a heading:

```
8882 \renewcommand{\glsgroupheading}[1]{\par  
8883   \noindent\glstreenamefmt{\glsgetgrouptitle{##1}}\par\indexspace}%  
8884 }
```

`onamehypergroup` The `treenonamehypergroup` style is like the `treenonamegroup` style, but has a set of links to the groups at the start of the glossary.

```
8885 \newglossarystyle{treenonamehypergroup}{%
```

Base it on the `glostytreename` style:

```
8886 \setglossarystyle{treename}%
```

Put navigation links to the groups at the start of the `theglossary` environment:

```
8887 \renewcommand*{\glossaryheader}{%
```

```
8888   \par\noindent\glstreenamefmt{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
8889 \renewcommand*{\glsgroupheading}[1]{%
```

```
8890   \par\noindent
```

```
8891   \glstreenamefmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
```

```
8892   \indexspace}%
```

```
8893 }
```

`\glssetwidest` `\glssetwidest[⟨level⟩]{⟨text⟩}` sets the widest text for the given level. It is used by the `alttree` glossary styles to determine the indentation of each level.

```
8894 \newcommand*{\glssetwidest}[2][0]{%
```

```
8895   \expandafter\def\csname @glswidestname\romannumeral#1\endcsname{%
```

```
8896     #2}%
```

```
8897 }
```

`\@glswidestname` Initialise `\@glswidestname`.

```
8898 \newcommand*{\@glswidestname}{}%
```

`\glstreenamebox` Used by the `alttree` style to create the box for the name and associated information.

```
8899 \newcommand*{\glstreenamebox}[2]{%
```

```
8900   \makebox[#1][l]{#2}%
```

```
8901 }
```

`alttree` The `alttree` glossary style is similar in style to the `tree` style, but the indentation is obtained from the width of `\@glswidestname` which is set using `\glssetwidest`.

```
8902 \newglossarystyle{alttree}{%
```

Redefine the `theglossary` environment.

```
8903 \renewenvironment{theglossary}{%
```

```
8904   {\def\@gls@prevlevel{-1}%
```

```
8905   \mbox{}\par}%
```

```
8906   {\par}%
```

Set the header and group headers to nothing.

```
8907 \renewcommand*{\glossaryheader}{}%
```

```
8908 \renewcommand*{\glsgroupheading}[1]{}%
```

Redefine the way that the level 0 entries are displayed.

```
8909 \renewcommand{\glossentry}[2]{%
```

```
8910   \ifnum\@gls@prevlevel=0\relax
```

```
8911   \else
```

Find out how big the indentation should be by measuring the widest entry.

```
8912   \settowidth{\glstreeindent}{\glstreenamefmt{\@glswidestname\space}}%
```

```
8913   \fi
```

Set the hangindent and paragraph indent.

```
8914     \hangindent\glstreeindent
8915     \parindent\glstreeindent
```

Put the name to the left of the paragraph block.

```
8916     \makebox[0pt][r]{\glstreenamebox{\glstreeindent}{%
8917         \glstreeentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}}}%
```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
8918     \ifglshassymbol{##1}{(\glossentrysymbol{##1})\space}{}%
```

Do the description followed by the description terminator and location list.

```
8919     \glossentrydesc{##1}\glspostdescription \space ##2\par
```

Set the previous level to 0.

```
8920     \def\@gls@prevlevel{0}%
8921     }%
```

Redefine the way sub-entries are displayed.

```
8922     \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.

```
8923     \ifnum##1=1\relax
8924         \glssubentryitem{##2}%
8925     \fi
```

If the level hasn't changed, keep the same settings, otherwise adjust `\glstreeindent` accordingly.

```
8926     \ifnum\@gls@prevlevel=##1\relax
8927     \else
```

Compute the widest entry for this level, or for level 0 if not defined for this level. Store in `\gls@tmplen`

```
8928         \@ifundefined{@glswidestname\romannumeral##1}{%
8929             \settowidth{\gls@tmplen}{\glstreenamefmt{\@glswidestname\space}}{%
8930             \settowidth{\gls@tmplen}{\glstreenamefmt{%
8931                 \csname @glswidestname\romannumeral##1\endcsname\space}}}%
```

Determine if going up or down a level

```
8932         \ifnum\@gls@prevlevel<##1\relax
```

Depth has increased, so add the width of the widest entry to `\glstreeindent`.

```
8933             \setlength\glstreeindent\gls@tmplen
8934             \addtolength\glstreeindent\parindent
8935             \parindent\glstreeindent
8936         \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`.

```
8937             \@ifundefined{@glswidestname\romannumeral\@gls@prevlevel}{%
8938             \settowidth{\glstreeindent}{\glstreenamefmt{%
```

```

8939         \@glswidestname\space}}}{%
8940         \settowidth{\glstreeindent}{\glstreenamefmt{%
8941         \csname @glswidestname\romannumeral\@gls@prevlevel
8942         \endcsname\space}}}{%

```

Subtract this length from the previous level's paragraph indent and set to \glstreeindent.

```

8943         \addtolength\parindent{-\glstreeindent}%
8944         \setlength\glstreeindent\parindent
8945     \fi
8946 \fi

```

Set the hanging indentation.

```
8947 \hangindent\glstreeindent
```

Put the name to the left of the paragraph block

```

8948 \makebox[0pt][r]{\glstreenamebox{\gls@tmplen}{%
8949 \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}}}{%

```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
8950 \ifglshassymbol{##2}{(\glossentrysymbol{##2})\space}{}%

```

Do the description followed by the description terminator and location list.

```
8951 \glossentrydesc{##2}\glspostdescription\space ##3\par
```

Set the previous level macro to the current level.

```

8952 \def\@gls@prevlevel{##1}%
8953 }%

```

Vertical gap between groups is the same as that used by indices:

```

8954 \renewcommand*\glsgroupskip{\ifglsnogroupskip\else\indexspace\fi}%
8955 }

```

almtreegroup Like the almtree style but the glossary groups have headings.

```
8956 \newglossarystyle{almtreegroup}{%
```

Base it on the glostylealmtree style:

```
8957 \setglossarystyle{almtree}%

```

Give each group a heading.

```

8958 \renewcommand{\glsgroupheading}[1]{\par
8959 \def\@gls@prevlevel{-1}%
8960 \hangindent0pt\relax
8961 \parindent0pt\relax
8962 \glstreenamefmt{\glsgetgrouptitle{##1}}\par\indexspace}%
8963 }

```

almtreehypergroup The almtreehypergroup style is like the almtreegroup style, but has a set of links to the groups at the start of the glossary.

```
8964 \newglossarystyle{almtreehypergroup}{%
```

Base it on the glostylealmtree style:

```
8965 \setglossarystyle{almtree}%

```

Put the navigation links in the header

```
8966 \renewcommand*{\glossaryheader}{%
8967   \par
8968   \def\@gls@prevlevel{-1}%
8969   \hangindent0pt\relax
8970   \parindent0pt\relax
8971   \glstreenamefmt{\glsnavigation}\par\indexspace}%
```

Put a hypertarget at the start of each group

```
8972 \renewcommand*{\glsgroupheading}[1]{%
8973   \par
8974   \def\@gls@prevlevel{-1}%
8975   \hangindent0pt\relax
8976   \parindent0pt\relax
8977   \glstreenamefmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8978   \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.

```
8979 \NeedsTeXFormat{LaTeX2e}
8980 \ProvidesPackage{glossaries-compatible-207}[2016/01/24 v4.21 (NLCT)]
```

AddXdyAttribute Adds an attribute in old format.

```
8981 \ifglxsindy
8982 \renewcommand*\GlsAddXdyAttribute[1]{%
8983 \edef\xdyattributes{\xdyattributes ^^J \string"#1\string"}%
8984 \expandafter\toks@\expandafter{\xdylocref}%
8985 \edef\xdylocref{\the\toks@ ^^J%
8986 (markup-locref
8987 :open \string"\string~n\string\setentrycounter
8988 {\noexpand\glscounter}%
8989 \expandafter\string\csname#1\endcsname
8990 \expandafter@gobble\string\{\string" ^^J
8991 :close \string"\expandafter@gobble\string}\string" ^^J
8992 :attr \string"#1\string")}}
```

Only has an effect before `\writeist`:

```
8993 \fi
```

sAddXdyCounters

```
8994 \renewcommand*\GlsAddXdyCounters[1]{%
8995 \GlossariesWarning{\string\GlsAddXdyCounters\space not available
8996 in compatibility mode.}%
8997 }
```

Add predefined attributes

```
8998 \GlsAddXdyAttribute{glsnumberformat}
8999 \GlsAddXdyAttribute{textrm}
9000 \GlsAddXdyAttribute{textsf}
9001 \GlsAddXdyAttribute{texttt}
9002 \GlsAddXdyAttribute{textbf}
9003 \GlsAddXdyAttribute{textmd}
9004 \GlsAddXdyAttribute{textit}
9005 \GlsAddXdyAttribute{textup}
9006 \GlsAddXdyAttribute{textsl}
```

```

9007 \GlsAddXdyAttribute{textsc}
9008 \GlsAddXdyAttribute{emph}
9009 \GlsAddXdyAttribute{glshypernumber}
9010 \GlsAddXdyAttribute{hyperrm}
9011 \GlsAddXdyAttribute{hypersf}
9012 \GlsAddXdyAttribute{hypertt}
9013 \GlsAddXdyAttribute{hyperbf}
9014 \GlsAddXdyAttribute{hypermd}
9015 \GlsAddXdyAttribute{hyperit}
9016 \GlsAddXdyAttribute{hyperup}
9017 \GlsAddXdyAttribute{hypersl}
9018 \GlsAddXdyAttribute{hypersc}
9019 \GlsAddXdyAttribute{hyperemph}

```

sAddXdyLocation Restore v2.07 definition:

```

9020 \ifglxindy
9021 \renewcommand*\GlsAddXdyLocation}[2]{%
9022   \edef\xdyuserlocationdefs{%
9023     \@xdyuserlocationdefs ^^J%
9024     (define-location-class \string"#1\string"^^J\space\space
9025     \space(#2))
9026   }%
9027   \edef\xdyuserlocationnames{%
9028     \@xdyuserlocationnames^^J\space\space\space
9029     \string"#1\string"}%
9030 }
9031 \fi

```

\@do@wrglossary

```

9032 \renewcommand{\@do@wrglossary}[1]{%
  Determine whether to use xindy or makeindex syntax
9033 \ifglxindy
  Need to determine if the formatting information starts with a ( or ) indicating a range.
9034 \expandafter\@glo@check@mkidxrangechar\@glsnumberformat\@nil
9035 \def\@glo@range{}%
9036 \expandafter\if\@glo@prefix(\relax
9037   \def\@glo@range{:open-range}%
9038   \else
9039     \expandafter\if\@glo@prefix)\relax
9040     \def\@glo@range{:close-range}%
9041   \fi
9042 \fi

  Get the location and escape any special characters
9043 \protected@edef\@glslocref{\theglsentrycounter}%
9044 \@gls@checkmkidxchars\@glslocref

  Write to the glossary file using xindy syntax.
9045 \glossary[\csname glo@#1@type\endcsname]{%

```

```

9046 (indexentry :tkey (\csname glo@#1@index\endcsname)
9047   :locref \string"\@glslocref\string" %
9048   :attr \string"\@glo@suffix\string" \@glo@range
9049 )
9050 }%
9051 \else

```

Convert the format information into the format required for makeindex

```

9052 \@set@glo@numformat\@glo@numfmt\@gls@counter\@glsnumberformat

```

Write to the glossary file using makeindex syntax.

```

9053 \glossary[\csname glo@#1@type\endcsname]{%
9054 \string\glossaryentry{\csname glo@#1@index\endcsname
9055   \@gls@encapchar\@glo@numfmt}{\theglsentrycounter}}%
9056 \fi
9057 }

```

t@glo@numformat Only had 3 arguments in v2.07

```

9058 \def\@set@glo@numformat#1#2#3{%
9059   \expandafter\@glo@check@mkidxrangechar#3\@nil
9060   \protected@edef#1{%
9061     \@glo@prefix setentrycounter[] {#2}%
9062     \expandafter\string\csname\@glo@suffix\endcsname
9063   }%
9064   \@gls@checkmkidxchars#1%
9065 }

```

\writeist Redefine \writeist back to the way it was in v2.07, but change \istfile to \glswrite.

```

9066 \ifglxindy
9067   \def\writeist{%
9068     \openout\glswrite=\istfilename
9069     \write\glswrite{;; xindy style file created by the glossaries
9070       package in compatible-2.07 mode}%
9071     \write\glswrite{;; for document '\jobname' on
9072       \the\year-\the\month-\the\day}%
9073     \write\glswrite{^^J; required styles^^J}
9074     \@for\@xdystyle:=\@xdyrequiredstyles\do{%
9075       \ifx\@xdystyle\@empty
9076       \else
9077         \protected@write\glswrite{{(require
9078           \string"\@xdystyle.xdy\string")}}%
9079       \fi
9080     }%
9081     \write\glswrite{^^J%
9082       ; list of allowed attributes (number formats)^^J}%
9083     \write\glswrite{(define-attributes ((\@xdyattributes)))}%
9084     \write\glswrite{^^J; user defined alphabets^^J}%
9085     \write\glswrite{\@xdyuseralphabets}%
9086     \write\glswrite{^^J; location class definitions^^J}%
9087     \protected@edef\@gls@roman{\@roman{0}\string"

```



```

9088     \string"roman-numbers-lowercase\string" :sep \string"}}%
9089 \@onelevel@sanitize\@gls@roman
9090 \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
9091     :sep \string"}%
9092 \@onelevel@sanitize\@tmp
9093 \ifx\@tmp\@gls@roman
9094     \write\glswrite{(define-location-class
9095         \string"roman-page-numbers\string"^^J\space\space\space
9096         (\string"roman-numbers-lowercase\string")
9097         :min-range-length \@glsminrange)}%
9098 \else
9099     \write\glswrite{(define-location-class
9100         \string"roman-page-numbers\string"^^J\space\space\space
9101         (:sep "\@gls@roman")
9102         :min-range-length \@glsminrange)}%
9103 \fi
9104 \write\glswrite{(define-location-class
9105     \string"Roman-page-numbers\string"^^J\space\space\space
9106     (\string"roman-numbers-uppercase\string")
9107     :min-range-length \@glsminrange)}%
9108 \write\glswrite{(define-location-class
9109     \string"arabic-page-numbers\string"^^J\space\space\space
9110     (\string"arabic-numbers\string")
9111     :min-range-length \@glsminrange)}%
9112 \write\glswrite{(define-location-class
9113     \string"alpha-page-numbers\string"^^J\space\space\space
9114     (\string"alpha\string")
9115     :min-range-length \@glsminrange)}%
9116 \write\glswrite{(define-location-class
9117     \string"Alpha-page-numbers\string"^^J\space\space\space
9118     (\string"ALPHA\string")
9119     :min-range-length \@glsminrange)}%
9120 \write\glswrite{(define-location-class
9121     \string"Appendix-page-numbers\string"^^J\space\space\space
9122     (\string"ALPHA\string"
9123     :sep \string"\@glsAlphacompositor\string"
9124     \string"arabic-numbers\string")
9125     :min-range-length \@glsminrange)}%
9126 \write\glswrite{(define-location-class
9127     \string"arabic-section-numbers\string"^^J\space\space\space
9128     (\string"arabic-numbers\string"
9129     :sep \string"\glscompositor\string"
9130     \string"arabic-numbers\string")
9131     :min-range-length \@glsminrange)}%
9132 \write\glswrite{^^J; user defined location classes}%
9133 \write\glswrite{\@xdyuserlocationdefs}%
9134 \write\glswrite{^^J; define cross-reference class^^J}%
9135 \write\glswrite{(define-crossref-class \string"see\string"
9136     :unverified )}%

```

```

9137 \write\glswrite{(markup-crossref-list
9138   :class \string"see\string"^^J\space\space\space
9139   :open \string"\string\glsseeformat\string"
9140   :close \string"{}\string")}%
9141 \write\glswrite{^^J; define the order of the location classes}%
9142 \write\glswrite{(define-location-class-order
9143   (\@xdylocationclassorder))}%
9144 \write\glswrite{^^J; define the glossary markup^^J}%
9145 \write\glswrite{(markup-index^^J\space\space\space
9146   :open \string"\string
9147     \glossarysection[\string\glossarytoctitle]{\string
9148     \glossarytitle}\string\glossarypreamble\string~n\string\begin
9149     {theglossary}\string\glossaryheader\string~n\string" ^^J\space
9150     \space\space:close \string"\expandafter\@gobble
9151     \string%\string~n\string
9152     \end{theglossary}\string\glossarypostamble
9153     \string~n\string" ^^J\space\space\space
9154   :tree)}}%
9155 \write\glswrite{(markup-letter-group-list
9156   :sep \string"\string\glsgroupskip\string~n\string")}%
9157 \write\glswrite{(markup-indexentry
9158   :open \string"\string\relax \string\glsresetentrylist
9159     \string~n\string")}%
9160 \write\glswrite{(markup-locclass-list :open
9161   \string"\glsopenbrace\string\glossaryentrynumbers
9162     \glsopenbrace\string\relax\space \string"^^J\space\space\space
9163   :sep \string", \string"
9164   :close \string"\glsclosebrace\glsclosebrace\string")}%
9165 \write\glswrite{(markup-locref-list
9166   :sep \string"\string\delimN\space\string")}%
9167 \write\glswrite{(markup-range
9168   :sep \string"\string\delimR\space\string")}%
9169 \@onelevel@sanitize\gls@suffixF
9170 \@onelevel@sanitize\gls@suffixFF
9171 \ifx\gls@suffixF\@empty
9172 \else
9173   \write\glswrite{(markup-range
9174     :close "\gls@suffixF" :length 1 :ignore-end)}%
9175 \fi
9176 \ifx\gls@suffixFF\@empty
9177 \else
9178   \write\glswrite{(markup-range
9179     :close "\gls@suffixFF" :length 2 :ignore-end)}%
9180 \fi
9181 \write\glswrite{^^J; define format to use for locations^^J}%
9182 \write\glswrite{\@xdylocref}%
9183 \write\glswrite{^^J; define letter group list format^^J}%
9184 \write\glswrite{(markup-letter-group-list
9185   :sep \string"\string\glsgroupskip\string~n\string")}%

```

```

9186 \write\glswrite{^^J; letter group headings^^J}%
9187 \write\glswrite{(markup-letter-group
9188   :open-head \string"\string\glsgroupheading
9189   \glsopenbrace\string"^^J\space\space\space
9190   :close-head \string"\glsclosebrace\string")}%
9191 \write\glswrite{^^J; additional letter groups^^J}%
9192 \write\glswrite{\@xdylettergroups}%
9193 \write\glswrite{^^J; additional sort rules^^J}
9194 \write\glswrite{\@xdysortrules}%
9195 \noist}
9196 \else
9197 \edef\@gls@actualchar{\string?}
9198 \edef\@gls@encapchar{\string|}
9199 \edef\@gls@levelchar{\string!}
9200 \edef\@gls@quotechar{\string"}
9201 \def\writeist{\relax
9202   \openout\glswrite=\istfilename
9203   \write\glswrite{\expandafter\@gobble\string\% makeindex style file
9204     created by the glossaries package}
9205   \write\glswrite{\expandafter\@gobble\string\% for document
9206     'jobname' on \the\year-\the\month-\the\day}
9207   \write\glswrite{actual '@gls@actualchar'}
9208   \write\glswrite{encap '@gls@encapchar'}
9209   \write\glswrite{level '@gls@levelchar'}
9210   \write\glswrite{quote '@gls@quotechar'}
9211   \write\glswrite{keyword \string"\string\glossaryentry\string"}
9212   \write\glswrite{preamble \string"\string\glossarysection[\string
9213     \glossarytoctitle]{\string\glossarytitle}\string
9214     \glossarypreamble\string\n\string\begin{theglossary}\string
9215     \glossaryheader\string\n\string"}
9216   \write\glswrite{postamble \string"\string%\string\n\string
9217     \end{theglossary}\string\glossarypostamble\string\n
9218     \string"}
9219   \write\glswrite{group_skip \string"\string\glsgroupskip\string\n
9220     \string"}
9221   \write\glswrite{item_0 \string"\string%\string\n\string"}
9222   \write\glswrite{item_1 \string"\string%\string\n\string"}
9223   \write\glswrite{item_2 \string"\string%\string\n\string"}
9224   \write\glswrite{item_01 \string"\string%\string\n\string"}
9225   \write\glswrite{item_x1
9226     \string"\string\relax \string\glsresetentrylist\string\n
9227     \string"}
9228   \write\glswrite{item_12 \string"\string%\string\n\string"}
9229   \write\glswrite{item_x2
9230     \string"\string\relax \string\glsresetentrylist\string\n
9231     \string"}
9232   \write\glswrite{delim_0 \string"\string{\string
9233     \glossaryentrynumbers\string\{\string\relax \string"}
9234   \write\glswrite{delim_1 \string"\string{\string

```

```

9235     \glossaryentrynumbers\string\{\string\relax \string}
9236 \write\glswrite{delim_2 \string"\string\{\string
9237     \glossaryentrynumbers\string\{\string\relax \string}
9238 \write\glswrite{delim_t \string"\string}\string}\string}
9239 \write\glswrite{delim_n \string"\string\delimN \string}
9240 \write\glswrite{delim_r \string"\string\delimR \string}
9241 \write\glswrite{headings_flag 1}
9242 \write\glswrite{heading_prefix
9243     \string"\string\glsgroupheading\string\{\string}
9244 \write\glswrite{heading_suffix
9245     \string"\string}\string\relax
9246     \string\glsresetentrylist \string}
9247 \write\glswrite{symhead_positive \string"glssymbols\string}
9248 \write\glswrite{numhead_positive \string"glnumbers\string}
9249 \write\glswrite{page_compositor \string"glsc compositor\string}
9250 \@gls@escbsdq\gls@suffixF
9251 \@gls@escbsdq\gls@suffixFF
9252 \ifx\gls@suffixF\@empty
9253 \else
9254     \write\glswrite{suffix_2p \string"\gls@suffixF\string}
9255 \fi
9256 \ifx\gls@suffixFF\@empty
9257 \else
9258     \write\glswrite{suffix_3p \string"\gls@suffixFF\string}
9259 \fi
9260 \noist
9261 }
9262 \fi

```

\noist

```
9263 \renewcommand*{\noist}{\let\writeist\relax}
```

4.2 glossaries-compatible-307

```

9264 \NeedsTeXFormat{LaTeX2e}
9265 \ProvidesPackage{glossaries-compatible-307}[2016/01/24 v4.21 (NLCT)]

```

Compatibility macros for predefined glossary styles:

`atglossarystyle` Defines a compatibility glossary style.

```

9266 \newcommand{\compatglossarystyle}[2]{%
9267   \ifcsundef{@glscompstyle@#1}%
9268   {%
9269     \csdef{@glscompstyle@#1}{#2}%
9270   }%
9271   {%
9272     \PackageError{glossaries}{Glossary compatibility style ‘#1’ is already defined}{%
9273   }%
9274 }

```

Backward compatible inline style.

```
9275 \compatglossarystyle{inline}{%
9276   \renewcommand{\glossaryentryfield}[5]{%
9277     \glsinlinedopostchild
9278     \gls@inlinesep
9279     \def\glo@desc{##3}%
9280     \def\@no@post@desc{\nopostdesc}%
9281     \glsentryitem{##1}\glsinlinenameformat{##1}{##2}%
9282     \ifx\glo@desc\@no@post@desc
9283       \glsinlineemptydescformat{##4}{##5}%
9284     \else
9285       \ifstrempy{##3}%
9286         {\glsinlineemptydescformat{##4}{##5}}%
9287         {\glsinlinedescformat{##3}{##4}{##5}}%
9288     \fi
9289     \ifglshaschildren{##1}%
9290     {%
9291       \glsresetsubentrycounter
9292       \glsinlineparentchildseparator
9293       \def\gls@inlinesubsep{}%
9294       \def\gls@inlinepostchild{\glsinlinepostchild}%
9295     }%
9296     {}%
9297     \def\gls@inlinesep{\glsinlineseparator}%
9298   }%
```

Sub-entries display description:

```
9299 \renewcommand{\glossarysubentryfield}[6]{%
9300   \gls@inlinesubsep%
9301   \glsinlinesubnameformat{##2}{##3}%
9302   \glssubentryitem{##2}\glsinlinesubdescformat{##4}{##5}{##6}%
9303   \def\gls@inlinesubsep{\glsinlinesubseparator}%
9304 }%
9305 }
```

Backward compatible list style.

```
9306 \compatglossarystyle{list}{%
9307   \renewcommand*{\glossaryentryfield}[5]{%
9308     \item[\glsentryitem{##1}\glstarget{##1}{##2}]
9309     ##3\glspostdescription\space ##5}%
9310 }
```

Sub-entries continue on the same line:

```
9310 \renewcommand*{\glossarysubentryfield}[6]{%
9311   \glssubentryitem{##2}%
9312   \glstarget{##2}{\strut}##4\glspostdescription\space ##6.}%
9313 }
```

Backward compatible listgroup style.

```
9314 \compatglossarystyle{listgroup}{%
9315   \csuse{@glscompstyle@list}%
9316 }%
```

Backward compatible listhypergroup style.

```
9317 \compatglossarystyle{listhypergroup}{%
9318 \csuse{@glscompstyle@list}%
9319 }%
```

Backward compatible altlist style.

```
9320 \compatglossarystyle{altlist}{%
9321 \renewcommand*{\glossaryentryfield}[5]{%
9322 \item[\glsentryitem{##1}\glstarget{##1}{##2}]%
9323 \mbox{}\par\nobreak\@afterheading
9324 ##3\glspostdescription\space ##5}%
9325 \renewcommand{\glossarysubentryfield}[6]{%
9326 \par
9327 \glssubentryitem{##2}%
9328 \glstarget{##2}{\strut}##4\glspostdescription\space ##6}%
9329 }%
```

Backward compatible altlistgroup style.

```
9330 \compatglossarystyle{altlistgroup}{%
9331 \csuse{@glscompstyle@altlist}%
9332 }%
```

Backward compatible altlisthypergroup style.

```
9333 \compatglossarystyle{altlisthypergroup}{%
9334 \csuse{@glscompstyle@altlist}%
9335 }%
```

Backward compatible listdotted style.

```
9336 \compatglossarystyle{listdotted}{%
9337 \renewcommand*{\glossaryentryfield}[5]{%
9338 \item[]\makebox[\glslistdottedwidth][l]{%
9339 \glsentryitem{##1}\glstarget{##1}{##2}%
9340 \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##3}%
9341 \renewcommand*{\glossarysubentryfield}[6]{%
9342 \item[]\makebox[\glslistdottedwidth][l]{%
9343 \glssubentryitem{##2}%
9344 \glstarget{##2}{##3}%
9345 \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##4}%
9346 }%
```

Backward compatible sublistdotted style.

```
9347 \compatglossarystyle{sublistdotted}{%
9348 \csuse{@glscompstyle@listdotted}%
9349 \renewcommand*{\glossaryentryfield}[5]{%
9350 \item[\glsentryitem{##1}\glstarget{##1}{##2}]}%
9351 }%
```

Backward compatible long style.

```
9352 \compatglossarystyle{long}{%
9353 \renewcommand*{\glossaryentryfield}[5]{%
9354 \glsentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\}%
9355 \renewcommand*{\glossarysubentryfield}[6]{%

```

```

9356      &
9357      \glssubentryitem{##2}%
9358      \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
9359 }%

```

Backward compatible longborder style.

```

9360 \compatglossarystyle{longborder}{%
9361 \csuse{@glscompstyle@long}%
9362 }%

```

Backward compatible longheader style.

```

9363 \compatglossarystyle{longheader}{%
9364 \csuse{@glscompstyle@long}%
9365 }%

```

Backward compatible longheaderborder style.

```

9366 \compatglossarystyle{longheaderborder}{%
9367 \csuse{@glscompstyle@long}%
9368 }%

```

Backward compatible long3col style.

```

9369 \compatglossarystyle{long3col}{%
9370 \renewcommand*{\glossaryentryfield}[5]{%
9371 \glstarget{##1}{\strut}##4 & ##3 & ##5\\}%
9372 \renewcommand*{\glossarysubentryfield}[6]{%
9373 &
9374 \glssubentryitem{##2}%
9375 \glstarget{##2}{\strut}##4 & ##6\\}%
9376 }%

```

Backward compatible long3colborder style.

```

9377 \compatglossarystyle{long3colborder}{%
9378 \csuse{@glscompstyle@long3col}%
9379 }%

```

Backward compatible long3colheader style.

```

9380 \compatglossarystyle{long3colheader}{%
9381 \csuse{@glscompstyle@long3col}%
9382 }%

```

Backward compatible long3colheaderborder style.

```

9383 \compatglossarystyle{long3colheaderborder}{%
9384 \csuse{@glscompstyle@long3col}%
9385 }%

```

Backward compatible long4col style.

```

9386 \compatglossarystyle{long4col}{%
9387 \renewcommand*{\glossaryentryfield}[5]{%
9388 \glstarget{##1}{\strut}##4 & ##3 & ##4 & ##5\\}%
9389 \renewcommand*{\glossarysubentryfield}[6]{%
9390 &
9391 \glssubentryitem{##2}%

```

```

9392 \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%
9393 }%

Backward compatible long4colheader style.
9394 \compatglossarystyle{long4colheader}{%
9395 \csuse{@glscompstyle@long4col}%
9396 }%

Backward compatible long4colborder style.
9397 \compatglossarystyle{long4colborder}{%
9398 \csuse{@glscompstyle@long4col}%
9399 }%

Backward compatible long4colheaderborder style.
9400 \compatglossarystyle{long4colheaderborder}{%
9401 \csuse{@glscompstyle@long4col}%
9402 }%

Backward compatible altlong4col style.
9403 \compatglossarystyle{altlong4col}{%
9404 \csuse{@glscompstyle@long4col}%
9405 }%

Backward compatible altlong4colheader style.
9406 \compatglossarystyle{altlong4colheader}{%
9407 \csuse{@glscompstyle@long4col}%
9408 }%

Backward compatible altlong4colborder style.
9409 \compatglossarystyle{altlong4colborder}{%
9410 \csuse{@glscompstyle@long4col}%
9411 }%

Backward compatible altlong4colheaderborder style.
9412 \compatglossarystyle{altlong4colheaderborder}{%
9413 \csuse{@glscompstyle@long4col}%
9414 }%

Backward compatible long style.
9415 \compatglossarystyle{longragged}{%
9416 \renewcommand*{\glossaryentryfield}[5]{%
9417 \glstarget{##1}{\strut}##4 & ##3\glspostdescription\space ##5%
9418 \tabularnewline}%
9419 \renewcommand*{\glossarysubentryfield}[6]{%
9420 &
9421 \glssubentryitem{##2}%
9422 \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
9423 \tabularnewline}%
9424 }%

Backward compatible longraggedborder style.
9425 \compatglossarystyle{longraggedborder}{%
9426 \csuse{@glscompstyle@longragged}%
9427 }%

```


Backward compatible longraggedheader style.

```
9428 \compatglossarystyle{longraggedheader}{%
9429 \csuse{@glscompstyle@longragged}%
9430 }%
```

Backward compatible longraggedheaderborder style.

```
9431 \compatglossarystyle{longraggedheaderborder}{%
9432 \csuse{@glscompstyle@longragged}%
9433 }%
```

Backward compatible longragged3col style.

```
9434 \compatglossarystyle{longragged3col}{%
9435 \renewcommand*{\glossaryentryfield}[5]{%
9436 \glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%
9437 \renewcommand*{\glossarysubentryfield}[6]{%
9438 &
9439 \glssubentryitem{##2}%
9440 \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%
9441 }%
```

Backward compatible longragged3colborder style.

```
9442 \compatglossarystyle{longragged3colborder}{%
9443 \csuse{@glscompstyle@longragged3col}%
9444 }%
```

Backward compatible longragged3colheader style.

```
9445 \compatglossarystyle{longragged3colheader}{%
9446 \csuse{@glscompstyle@longragged3col}%
9447 }%
```

Backward compatible longragged3colheaderborder style.

```
9448 \compatglossarystyle{longragged3colheaderborder}{%
9449 \csuse{@glscompstyle@longragged3col}%
9450 }%
```

Backward compatible altlongragged4col style.

```
9451 \compatglossarystyle{altlongragged4col}{%
9452 \renewcommand*{\glossaryentryfield}[5]{%
9453 \glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%
9454 \renewcommand*{\glossarysubentryfield}[6]{%
9455 &
9456 \glssubentryitem{##2}%
9457 \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%
9458 }%
```

Backward compatible altlongragged4colheader style.

```
9459 \compatglossarystyle{altlongragged4colheader}{%
9460 \csuse{@glscompstyle@altlong4col}%
9461 }%
```

Backward compatible altlongragged4colborder style.

```
9462 \compatglossarystyle{altlongragged4colborder}{%
```

```
9463 \csuse{@glscompstyle@altlong4col}%
9464 }%
```

Backward compatible altlongragged4colheaderborder style.

```
9465 \compatglossarystyle{altlongragged4colheaderborder}{%
9466 \csuse{@glscompstyle@altlong4col}%
9467 }%
```

Backward compatible index style.

```
9468 \compatglossarystyle{index}{%
9469 \renewcommand*{\glossaryentryfield}[5]{%
9470 \item\glsentryitem{##1}\textbf{\glstarget{##1}{##2}}%
9471 \ifx\relax##4\relax
9472 \else
9473 \space{##4}%
9474 \fi
9475 \space ##3\glspostdescription \space ##5}%
9476 \renewcommand*{\glossarysubentryfield}[6]{%
9477 \ifcase##1\relax
9478 % level 0
9479 \item
9480 \or
9481 % level 1
9482 \subitem
9483 \glssubentryitem{##2}%
9484 \else
9485 % all other levels
9486 \subsubitem
9487 \fi
9488 \textbf{\glstarget{##2}{##3}}%
9489 \ifx\relax##5\relax
9490 \else
9491 \space{##5}%
9492 \fi
9493 \space##4\glspostdescription\space ##6}%
9494 }%
```

Backward compatible indexgroup style.

```
9495 \compatglossarystyle{indexgroup}{%
9496 \csuse{@glscompstyle@index}%
9497 }%
```

Backward compatible indexhypergroup style.

```
9498 \compatglossarystyle{indexhypergroup}{%
9499 \csuse{@glscompstyle@index}%
9500 }%
```

Backward compatible tree style.

```
9501 \compatglossarystyle{tree}{%
9502 \renewcommand{\glossaryentryfield}[5]{%
9503 \hangindent0pt\relax
```

```

9504 \parindent0pt\relax
9505 \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
9506 \ifx\relax##4\relax
9507 \else
9508 \space{##4}%
9509 \fi
9510 \space ##3\glspostdescription \space ##5\par}%
9511 \renewcommand{\glossarysubentryfield}[6]{%
9512 \hangindent##1\glstreeindent\relax
9513 \parindent##1\glstreeindent\relax
9514 \ifnum##1=1\relax
9515 \glssubentryitem{##2}%
9516 \fi
9517 \textbf{\glstarget{##2}{##3}}%
9518 \ifx\relax##5\relax
9519 \else
9520 \space{##5}%
9521 \fi
9522 \space##4\glspostdescription\space ##6\par}%
9523 }%

```

Backward compatible treegroup style.

```

9524 \compatglossarystyle{treegroup}{%
9525 \csuse{@glscompstyle@tree}%
9526 }%

```

Backward compatible treehypergroup style.

```

9527 \compatglossarystyle{treehypergroup}{%
9528 \csuse{@glscompstyle@tree}%
9529 }%

```

Backward compatible treenoname style.

```

9530 \compatglossarystyle{treenoname}{%
9531 \renewcommand{\glossaryentryfield}[5]{%
9532 \hangindent0pt\relax
9533 \parindent0pt\relax
9534 \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
9535 \ifx\relax##4\relax
9536 \else
9537 \space{##4}%
9538 \fi
9539 \space ##3\glspostdescription \space ##5\par}%
9540 \renewcommand{\glossarysubentryfield}[6]{%
9541 \hangindent##1\glstreeindent\relax
9542 \parindent##1\glstreeindent\relax
9543 \ifnum##1=1\relax
9544 \glssubentryitem{##2}%
9545 \fi
9546 \glstarget{##2}{\strut}%
9547 ##4\glspostdescription\space ##6\par}%
9548 }%

```

Backward compatible treenonamegroup style.

```
9549 \compatglossarystyle{treenonamegroup}{%
9550 \csuse{@glscompstyle@treenoname}%
9551 }%
```

Backward compatible treenonamehypergroup style.

```
9552 \compatglossarystyle{treenonamehypergroup}{%
9553 \csuse{@glscompstyle@treenoname}%
9554 }%
```

Backward compatible almtree style.

```
9555 \compatglossarystyle{almtree}{%
9556 \renewcommand{\glossaryentryfield}[5]{%
9557 \ifnum\@gls@prevlevel=0\relax
9558 \else
9559 \settowidth{\glstreeindent}{\textbf{\@glswidestname\space}}%
9560 \hangindent\glstreeindent
9561 \parindent\glstreeindent
9562 \fi
9563 \makebox[0pt][r]{\makebox[\glstreeindent][l]{%
9564 \glsentryitem{##1}\textbf{\glstarget{##1}{##2}}}}%
9565 \ifx\relax##4\relax
9566 \else
9567 (##4)\space
9568 \fi
9569 ##3\glspostdescription \space ##5\par
9570 \def\@gls@prevlevel{0}%
9571 }%
9572 \renewcommand{\glossarysubentryfield}[6]{%
9573 \ifnum##1=1\relax
9574 \glsesubentryitem{##2}%
9575 \fi
9576 \ifnum\@gls@prevlevel=##1\relax
9577 \else
9578 \@ifundefined{@glswidestname\romannumeral##1}{%
9579 \settowidth{\gls@tmplen}{\textbf{\@glswidestname\space}}{%
9580 \settowidth{\gls@tmplen}{\textbf{%
9581 \csname @glswidestname\romannumeral##1\endcsname\space}}}%
9582 \ifnum\@gls@prevlevel<##1\relax
9583 \setlength\glstreeindent\gls@tmplen
9584 \addtolength\glstreeindent\parindent
9585 \parindent\glstreeindent
9586 \else
9587 \@ifundefined{@glswidestname\romannumeral\@gls@prevlevel}{%
9588 \settowidth{\glstreeindent}{\textbf{%
9589 \@glswidestname\space}}{%
9590 \settowidth{\glstreeindent}{\textbf{%
9591 \csname @glswidestname\romannumeral\@gls@prevlevel
9592 \endcsname\space}}}%
9593 \addtolength\parindent{-\glstreeindent}}%
```

```

9594     \setlength\glstreeindent\parindent
9595     \fi
9596     \fi
9597     \hangindent\glstreeindent
9598     \makebox[0pt][r]{\makebox[\gls@tmplen][l]{%
9599       \textbf{\glstarget{##2}{##3}}}}%
9600     \ifx##5\relax\relax
9601     \else
9602       (##5)\space
9603     \fi
9604     ##4\glspostdescription\space ##6\par
9605     \def\@gls@prevlevel{##1}%
9606   }%
9607 }%

```

Backward compatible alltreegroup style.

```

9608 \compatglossarystyle{alttreegroup}{%
9609   \csuse{@glscompstyle@alttree}%
9610 }%

```

Backward compatible alltreehypergroup style.

```

9611 \compatglossarystyle{alttreehypergroup}{%
9612   \csuse{@glscompstyle@alttree}%
9613 }%

```

Backward compatible mcolindex style.

```

9614 \compatglossarystyle{mcolindex}{%
9615   \csuse{@glscompstyle@index}%
9616 }%

```

Backward compatible mcolindexgroup style.

```

9617 \compatglossarystyle{mcolindexgroup}{%
9618   \csuse{@glscompstyle@index}%
9619 }%

```

Backward compatible mcolindexhypergroup style.

```

9620 \compatglossarystyle{mcolindexhypergroup}{%
9621   \csuse{@glscompstyle@index}%
9622 }%

```

Backward compatible mcoltree style.

```

9623 \compatglossarystyle{mcoltree}{%
9624   \csuse{@glscompstyle@tree}%
9625 }%

```

Backward compatible mcoltreegroup style.

```

9626 \compatglossarystyle{mcolindextreegroup}{%
9627   \csuse{@glscompstyle@tree}%
9628 }%

```

Backward compatible mcoltreehypergroup style.

```

9629 \compatglossarystyle{mcolindextreehypergroup}{%

```

```
9630 \csuse{@glscompstyle@tree}%
9631 }%
```

Backward compatible mcoltreenoname style.

```
9632 \compatglossarystyle{mcoltreenoname}{%
9633 \csuse{@glscompstyle@tree}%
9634 }%
```

Backward compatible mcoltreenonamegroup style.

```
9635 \compatglossarystyle{mcoltreenonamegroup}{%
9636 \csuse{@glscompstyle@tree}%
9637 }%
```

Backward compatible mcoltreenonamehypergroup style.

```
9638 \compatglossarystyle{mcoltreenonamehypergroup}{%
9639 \csuse{@glscompstyle@tree}%
9640 }%
```

Backward compatible mcolalmtree style.

```
9641 \compatglossarystyle{mcolalmtree}{%
9642 \csuse{@glscompstyle@almtree}%
9643 }%
```

Backward compatible mcolalmtreegroup style.

```
9644 \compatglossarystyle{mcolalmtreegroup}{%
9645 \csuse{@glscompstyle@almtree}%
9646 }%
```

Backward compatible mcolalmtreehypergroup style.

```
9647 \compatglossarystyle{mcolalmtreehypergroup}{%
9648 \csuse{@glscompstyle@almtree}%
9649 }%
```

Backward compatible superragged style.

```
9650 \compatglossarystyle{superragged}{%
9651 \renewcommand*{\glossaryentryfield}[5]{%
9652 \glssubentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
9653 \tabularnewline}%
9654 \renewcommand*{\glossarysubentryfield}[6]{%
9655 &
9656 \glssubentryitem{##2}%
9657 \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
9658 \tabularnewline}%
9659 }%
```

Backward compatible superraggedborder style.

```
9660 \compatglossarystyle{superraggedborder}{%
9661 \csuse{@glscompstyle@superragged}%
9662 }%
```

Backward compatible superraggedheader style.

```
9663 \compatglossarystyle{superraggedheader}{%
9664 \csuse{@glscompstyle@superragged}%
9665 }%
```

Backward compatible superraggedheaderborder style.

```
9666 \compatglossarystyle{superraggedheaderborder}{%
9667 \csuse{@glscompstyle@superragged}%
9668 }%
```

Backward compatible superragged3col style.

```
9669 \compatglossarystyle{superragged3col}{%
9670 \renewcommand*{\glossaryentryfield}[5]{%
9671 \glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%
9672 \renewcommand*{\glossarysubentryfield}[6]{%
9673 &
9674 \glssubentryitem{##2}%
9675 \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%
9676 }%
```

Backward compatible superragged3colborder style.

```
9677 \compatglossarystyle{superragged3colborder}{%
9678 \csuse{@glscompstyle@superragged3col}%
9679 }%
```

Backward compatible superragged3colheader style.

```
9680 \compatglossarystyle{superragged3colheader}{%
9681 \csuse{@glscompstyle@superragged3col}%
9682 }%
```

Backward compatible superragged3colheaderborder style.

```
9683 \compatglossarystyle{superragged3colheaderborder}{%
9684 \csuse{@glscompstyle@superragged3col}%
9685 }%
```

Backward compatible altsuperragged4col style.

```
9686 \compatglossarystyle{altsuperragged4col}{%
9687 \renewcommand*{\glossaryentryfield}[5]{%
9688 \glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%
9689 \renewcommand*{\glossarysubentryfield}[6]{%
9690 &
9691 \glssubentryitem{##2}%
9692 \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%
9693 }%
```

Backward compatible altsuperragged4colheader style.

```
9694 \compatglossarystyle{altsuperragged4colheader}{%
9695 \csuse{@glscompstyle@altsuperragged4col}%
9696 }%
```

Backward compatible altsuperragged4colborder style.

```
9697 \compatglossarystyle{altsuperragged4colborder}{%
9698 \csuse{@glscompstyle@altsuperragged4col}%
9699 }%
```

Backward compatible altsuperragged4colheaderborder style.

```
9700 \compatglossarystyle{altsuperragged4colheaderborder}{%

```

```
9701 \csuse{@glscompstyle@altsuperragged4col}%
9702 }%
```

Backward compatible super style.

```
9703 \compatglossarystyle{super}{%
9704 \renewcommand*{\glossaryentryfield}[5]{%
9705 \glentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\}%
9706 \renewcommand*{\glossarysubentryfield}[6]{%
9707 &
9708 \glssubentryitem{##2}%
9709 \glstarget{##2}{\strut}##4\glspostdescription\space ##6\}%
9710 }%
```

Backward compatible superborder style.

```
9711 \compatglossarystyle{superborder}{%
9712 \csuse{@glscompstyle@super}%
9713 }%
```

Backward compatible superheader style.

```
9714 \compatglossarystyle{superheader}{%
9715 \csuse{@glscompstyle@super}%
9716 }%
```

Backward compatible superheaderborder style.

```
9717 \compatglossarystyle{superheaderborder}{%
9718 \csuse{@glscompstyle@super}%
9719 }%
```

Backward compatible super3col style.

```
9720 \compatglossarystyle{super3col}{%
9721 \renewcommand*{\glossaryentryfield}[5]{%
9722 \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\}%
9723 \renewcommand*{\glossarysubentryfield}[6]{%
9724 &
9725 \glssubentryitem{##2}%
9726 \glstarget{##2}{\strut}##4 & ##6\}%
9727 }%
```

Backward compatible super3colborder style.

```
9728 \compatglossarystyle{super3colborder}{%
9729 \csuse{@glscompstyle@super3col}%
9730 }%
```

Backward compatible super3colheader style.

```
9731 \compatglossarystyle{super3colheader}{%
9732 \csuse{@glscompstyle@super3col}%
9733 }%
```

Backward compatible super3colheaderborder style.

```
9734 \compatglossarystyle{super3colheaderborder}{%
9735 \csuse{@glscompstyle@super3col}%
9736 }%
```


Backward compatible super4col style.

```
9737 \compatglossarystyle{super4col}{%
9738   \renewcommand*\glossaryentryfield}[5]{%
9739     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\}%
9740   \renewcommand*\glossarysubentryfield}[6]{%
9741     &
9742     \glssubentryitem{##2}%
9743     \glstarget{##2}{\strut}##4 & ##5 & ##6\}%
9744 }%
```

Backward compatible super4colheader style.

```
9745 \compatglossarystyle{super4colheader}{%
9746   \csuse{@glscompstyle@super4col}%
9747 }%
```

Backward compatible super4colborder style.

```
9748 \compatglossarystyle{super4colborder}{%
9749   \csuse{@glscompstyle@super4col}%
9750 }%
```

Backward compatible super4colheaderborder style.

```
9751 \compatglossarystyle{super4colheaderborder}{%
9752   \csuse{@glscompstyle@super4col}%
9753 }%
```

Backward compatible altsuper4col style.

```
9754 \compatglossarystyle{altsuper4col}{%
9755   \csuse{@glscompstyle@super4col}%
9756 }%
```

Backward compatible altsuper4colheader style.

```
9757 \compatglossarystyle{altsuper4colheader}{%
9758   \csuse{@glscompstyle@super4col}%
9759 }%
```

Backward compatible altsuper4colborder style.

```
9760 \compatglossarystyle{altsuper4colborder}{%
9761   \csuse{@glscompstyle@super4col}%
9762 }%
```

Backward compatible altsuper4colheaderborder style.

```
9763 \compatglossarystyle{altsuper4colheaderborder}{%
9764   \csuse{@glscompstyle@super4col}%
9765 }%
```

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.

```
9766 \NeedsTeXFormat{LaTeX2e}
```

Package version number now in line with main glossaries package number.

```
9767 \ProvidesPackage{glossaries-accsupp}[2016/01/24 v4.21 (NLCT)
```

```
9768 Experimental glossaries accessibility]
```

Pass all options to glossaries:

```
9769 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
9770 \ProcessOptions
```

This package should be loaded before glossaries-extra, so complain if that has already been loaded.

```
9771 \@ifpackageloaded{glossaries-extra}
```

```
9772 {%
```

```
9773 \PackageWarning{glossaries-accsupp}{The ‘glossaries-accsupp’
```

```
9774 package has been loaded after the ‘glossaries-extra’
```

```
9775 package. This can cause a failure to integrate both
```

```
9776 packages. Either use the ‘accsupp’ option when you
```

```
9777 load ‘glossaries-extra’ or load ‘glossaries-accsupp’
```

```
9778 before loading ‘glossaries-extra’}}%
```

```
9779 }
```

```
9780 {}
```

tibleglossentry Override style compatibility macros:

```
9781 \def\compatibleglossentry#1#2{%
```

```
9782 \toks@{#2}%
```

```
9783 \protected@edef\do@glossentry{%
```

```
9784 \noexpand\accsuppglossaryentryfield{#1}%
```

```
9785 {\noexpand\glsnamefont
```

```
9786 {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@name\endcsname}}%
```

```
9787 {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@desc\endcsname}%
```

```
9788 {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@symbol\endcsname}%
```

```
9789 {\the\toks@}%
```

```
9790 }%
```

```
9791 \do@glossentry
```

```
9792 }
```

lesubglossentry

```
9793 \def\compatiblesubglossentry#1#2#3{%
9794   \toks@{#3}%
9795   \protected@edef\@do@subglossentry{%
9796     \noexpand\accsuppglossarysubentryfield{\number#1}%
9797     {#2}%
9798     {\noexpand\glsnamefont
9799       {\expandafter\expandonce\csname glo@\glsdetoklabel{#2}@name\endcsname}}%
9800     {\expandafter\expandonce\csname glo@\glsdetoklabel{#2}@desc\endcsname}%
9801     {\expandafter\expandonce\csname glo@\glsdetoklabel{#2}@symbol\endcsname}%
9802     {\the\toks@}%
9803   }%
9804   \@do@subglossentry
9805 }
```

Required packages:

```
9806 \RequirePackage{glossaries}
9807 \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:

```
9808 \define@key{glossentry}{access}{%
9809   \def\@glo@access{#1}%
9810 }
```

textaccess The replacement text corresponding to the text key:

```
9811 \define@key{glossentry}{textaccess}{%
9812   \def\@glo@textaccess{#1}%
9813 }
```

firstaccess The replacement text corresponding to the first key:

```
9814 \define@key{glossentry}{firstaccess}{%
9815   \def\@glo@firstaccess{#1}%
9816 }
```

pluralaccess The replacement text corresponding to the plural key:

```
9817 \define@key{glossentry}{pluralaccess}{%
9818   \def\@glo@pluralaccess{#1}%
9819 }
```

firstpluralaccess The replacement text corresponding to the firstplural key:

```
9820 \define@key{glossentry}{firstpluralaccess}{%
9821   \def\@glo@firstpluralaccess{#1}%
9822 }
```

symbolaccess The replacement text corresponding to the symbol key:

```
9823 \define@key{glossentry}{symbolaccess}{%
9824   \def\@glo@symbolaccess{#1}%
9825 }
```

symbolpluralaccess The replacement text corresponding to the symbolplural key:

```
9826 \define@key{glossentry}{symbolpluralaccess}{%
9827   \def\@glo@symbolpluralaccess{#1}%
9828 }
```

descriptionaccess The replacement text corresponding to the description key:

```
9829 \define@key{glossentry}{descriptionaccess}{%
9830   \def\@glo@descaccess{#1}%
9831 }
```

descriptionpluralaccess The replacement text corresponding to the descriptionplural key:

```
9832 \define@key{glossentry}{descriptionpluralaccess}{%
9833   \def\@glo@descpluralaccess{#1}%
9834 }
```

shortaccess The replacement text corresponding to the short key:

```
9835 \define@key{glossentry}{shortaccess}{%
9836   \def\@glo@shortaccess{#1}%
9837 }
```

shortpluralaccess The replacement text corresponding to the shortplural key:

```
9838 \define@key{glossentry}{shortpluralaccess}{%
9839   \def\@glo@shortpluralaccess{#1}%
9840 }
```

longaccess The replacement text corresponding to the long key:

```
9841 \define@key{glossentry}{longaccess}{%
9842   \def\@glo@longaccess{#1}%
9843 }
```

longpluralaccess The replacement text corresponding to the longplural key:

```
9844 \define@key{glossentry}{longpluralaccess}{%
9845   \def\@glo@longpluralaccess{#1}%
9846 }
```

There are no equivalent keys for the user1...user6 keys. The replacement text would have to be explicitly put in the value, e.g., user1={\glsaccsupp{inches}{in}}.

Append these new keys to \@gls@keymap:

```
9847 \appto\@gls@keymap{,%  
9848   {access}{access},%  
9849   {textaccess}{textaccess},%  
9850   {firstaccess}{firstaccess},%  
9851   {pluralaccess}{pluralaccess},%  
9852   {firstpluralaccess}{firstpluralaccess},%  
9853   {symbolaccess}{symbolaccess},%  
9854   {symbolpluralaccess}{symbolpluralaccess},%  
9855   {descaccess}{descaccess},%  
9856   {descpluralaccess}{descpluralaccess},%  
9857   {shortaccess}{shortaccess},%  
9858   {shortpluralaccess}{shortpluralaccess},%  
9859   {longaccess}{longaccess},%  
9860   {longpluralaccess}{longpluralaccess}}%  
9861 }
```

\@gls@noaccess Indicates that no replacement text has been provided.

```
9862 \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):

```
9863 \let\@gls@oldnewglossaryentryprehook\@newglossaryentryprehook  
9864 \renewcommand*{\@newglossaryentryprehook}{%  
9865   \@gls@oldnewglossaryentryprehook  
9866   \def\@glo@access{\@glo@symbol}}%
```

Initialise the other keys:

```
9867 \def\@glo@textaccess{\@glo@access}%  
9868 \def\@glo@firstaccess{\@glo@access}%  
9869 \def\@glo@pluralaccess{\@glo@textaccess}%  
9870 \def\@glo@firstpluralaccess{\@glo@pluralaccess}%  
9871 \def\@glo@symbolaccess{\relax}%  
9872 \def\@glo@symbolpluralaccess{\@glo@symbolaccess}%  
9873 \def\@glo@descaccess{\relax}%  
9874 \def\@glo@descpluralaccess{\@glo@descaccess}%  
9875 \def\@glo@shortaccess{\relax}%  
9876 \def\@glo@shortpluralaccess{\@glo@shortaccess}%  
9877 \def\@glo@longaccess{\relax}%  
9878 \def\@glo@longpluralaccess{\@glo@longaccess}%  
9879 }
```

Add to the end hook:

```
9880 \let\@gls@oldnewglossaryentryposthook\@newglossaryentryposthook  
9881 \renewcommand*{\@newglossaryentryposthook}{%  
9882   \@gls@oldnewglossaryentryposthook
```

Store the access information:

```
9883 \expandafter  
9884   \protected@xdef\csname glo@\@glo@label @access\endcsname{%
```

```

9885     \@glo@access}%
9886 \expandafter
9887   \protected@xdef\csname glo@\@glo@label @textaccess\endcsname{%
9888     \@glo@textaccess}%
9889 \expandafter
9890   \protected@xdef\csname glo@\@glo@label @firstaccess\endcsname{%
9891     \@glo@firstaccess}%
9892 \expandafter
9893   \protected@xdef\csname glo@\@glo@label @pluralaccess\endcsname{%
9894     \@glo@pluralaccess}%
9895 \expandafter
9896   \protected@xdef\csname glo@\@glo@label @firstpluralaccess\endcsname{%
9897     \@glo@firstpluralaccess}%
9898 \expandafter
9899   \protected@xdef\csname glo@\@glo@label @symbolaccess\endcsname{%
9900     \@glo@symbolaccess}%
9901 \expandafter
9902   \protected@xdef\csname glo@\@glo@label @symbolpluralaccess\endcsname{%
9903     \@glo@symbolpluralaccess}%
9904 \expandafter
9905   \protected@xdef\csname glo@\@glo@label @descaccess\endcsname{%
9906     \@glo@descaccess}%
9907 \expandafter
9908   \protected@xdef\csname glo@\@glo@label @descpluralaccess\endcsname{%
9909     \@glo@descpluralaccess}%
9910 \expandafter
9911   \protected@xdef\csname glo@\@glo@label @shortaccess\endcsname{%
9912     \@glo@shortaccess}%
9913 \expandafter
9914   \protected@xdef\csname glo@\@glo@label @shortpluralaccess\endcsname{%
9915     \@glo@shortpluralaccess}%
9916 \expandafter
9917   \protected@xdef\csname glo@\@glo@label @longaccess\endcsname{%
9918     \@glo@longaccess}%
9919 \expandafter
9920   \protected@xdef\csname glo@\@glo@label @longpluralaccess\endcsname{%
9921     \@glo@longpluralaccess}%
9922 }

```

5.2 Accessing Replacement Text

`\glsentryaccess` Get the value of the access key for the entry with the given label:

```

9923 \newcommand*\glsentryaccess}[1]{%
9924   \@gls@entry@field{#1}{access}%
9925 }

```

`\glsentrytextaccess` Get the value of the textaccess key for the entry with the given label:

```

9926 \newcommand*\glsentrytextaccess}[1]{%

```

```
9927 \@gls@entry@field{#1}{textaccess}%  
9928 }
```

entryfirstaccess Get the value of the firstaccess key for the entry with the given label:

```
9929 \newcommand*{\glsentryfirstaccess}[1]{%  
9930 \@gls@entry@field{#1}{firstaccess}%  
9931 }
```

entrypluralaccess Get the value of the pluralaccess key for the entry with the given label:

```
9932 \newcommand*{\glsentrypluralaccess}[1]{%  
9933 \@gls@entry@field{#1}{pluralaccess}%  
9934 }
```

entryfirstpluralaccess Get the value of the firstpluralaccess key for the entry with the given label:

```
9935 \newcommand*{\glsentryfirstpluralaccess}[1]{%  
9936 \csname glo#1@firstpluralaccess\endcsname  
9937 }
```

entrysymbolaccess Get the value of the symbolaccess key for the entry with the given label:

```
9938 \newcommand*{\glsentrysymbolaccess}[1]{%  
9939 \@gls@entry@field{#1}{symbolaccess}%  
9940 }
```

entrysymbolpluralaccess Get the value of the symbolpluralaccess key for the entry with the given label:

```
9941 \newcommand*{\glsentrysymbolpluralaccess}[1]{%  
9942 \@gls@entry@field{#1}{symbolpluralaccess}%  
9943 }
```

entrydescaccess Get the value of the descriptionaccess key for the entry with the given label:

```
9944 \newcommand*{\glsentrydescaccess}[1]{%  
9945 \@gls@entry@field{#1}{descaccess}%  
9946 }
```

entrydescpluralaccess Get the value of the descriptionpluralaccess key for the entry with the given label:

```
9947 \newcommand*{\glsentrydescpluralaccess}[1]{%  
9948 \@gls@entry@field{#1}{descaccess}%  
9949 }
```

entryshortaccess Get the value of the shortaccess key for the entry with the given label:

```
9950 \newcommand*{\glsentryshortaccess}[1]{%  
9951 \@gls@entry@field{#1}{shortaccess}%  
9952 }
```

entryshortpluralaccess Get the value of the shortpluralaccess key for the entry with the given label:

```
9953 \newcommand*{\glsentryshortpluralaccess}[1]{%  
9954 \@gls@entry@field{#1}{shortpluralaccess}%  
9955 }
```

entrylongaccess Get the value of the longaccess key for the entry with the given label:

```
9956 \newcommand*{\glsentrylongaccess}[1]{%
9957   \@gls@entry@field{#1}{longaccess}%
9958 }
```

ongpluralaccess Get the value of the longpluralaccess key for the entry with the given label:

```
9959 \newcommand*{\glsentrylongpluralaccess}[1]{%
9960   \@gls@entry@field{#1}{longpluralaccess}%
9961 }
```

`\glsaccsupp` `\glsaccsupp{<replacement text>}{<text>}`

This can be redefined to use E or Alt instead of ActualText. (I don't have the software to test the E or Alt options.)

```
9962 \newcommand*{\glsaccsupp}[2]{%
9963   \BeginAccSupp{ActualText=#1}#2\EndAccSupp{}%
9964 }
```

`\xglsaccsupp` Fully expands replacement text before calling `\glsaccsupp`

```
9965 \newcommand*{\xglsaccsupp}[2]{%
9966   \protected@edef\@gls@replacementtext{#1}%
9967   \expandafter\glsaccsupp\expandafter{\@gls@replacementtext}{#2}%
9968 }
```

@access@display

```
9969 \newcommand*{\@gls@access@display}[2]{%
9970   \protected@edef\@glo@access{#2}%
9971   \ifx\@glo@access\@gls@noaccess
9972     #1%
9973   \else
9974     \xglsaccsupp{\@glo@access}{#1}%
9975   \fi
9976 }
```

meaccessdisplay Displays the first argument with the accessibility text for the entry with the label given by the second argument (if set).

```
9977 \DeclareRobustCommand*{\glsnameaccessdisplay}[2]{%
9978   \@gls@access@display{#1}{\glsentryaccess{#2}}%
9979 }
```

xtaccessdisplay As above but for the textaccess replacement text.

```
9980 \DeclareRobustCommand*{\glsstextaccessdisplay}[2]{%
9981   \@gls@access@display{#1}{\glsentrytextaccess{#2}}%
9982 }
```

alaccessdisplay As above but for the pluralaccess replacement text.

```
9983 \DeclareRobustCommand*{\glspluralaccessdisplay}[2]{%
9984   \@gls@access@display{#1}{\glsentrypluralaccess{#2}}%
9985 }
```



```

staccessdisplay  As above but for the firstaccess replacement text.
9986 \DeclareRobustCommand*\glsfirstaccessdisplay}[2]{%
9987  \@gls@access@display{#1}{\glsentryfirstaccess{#2}}%
9988 }

alaccessdisplay  As above but for the firstpluralaccess replacement text.
9989 \DeclareRobustCommand*\glsfirstpluralaccessdisplay}[2]{%
9990  \@gls@access@display{#1}{\glsentryfirstpluralaccess{#2}}%
9991 }

olaccessdisplay  As above but for the symbolaccess replacement text.
9992 \DeclareRobustCommand*\glsymbolaccessdisplay}[2]{%
9993  \@gls@access@display{#1}{\glsentrysymbolaccess{#2}}%
9994 }

alaccessdisplay  As above but for the symbolpluralaccess replacement text.
9995 \DeclareRobustCommand*\glsymbolpluralaccessdisplay}[2]{%
9996  \@gls@access@display{#1}{\glsentrysymbolpluralaccess{#2}}%
9997 }

onaccessdisplay  As above but for the descriptionaccess replacement text.
9998 \DeclareRobustCommand*\glsdescriptionaccessdisplay}[2]{%
9999  \@gls@access@display{#1}{\glsentrydescaccess{#2}}%
10000 }

alaccessdisplay  As above but for the descriptionpluralaccess replacement text.
10001 \DeclareRobustCommand*\glsdescriptionpluralaccessdisplay}[2]{%
10002  \@gls@access@display{#1}{\glsentrydescpluralaccess{#2}}%
10003 }

rtaccessdisplay  As above but for the shortaccess replacement text.
10004 \DeclareRobustCommand*\glsshortaccessdisplay}[2]{%
10005  \@gls@access@display{#1}{\glsentryshortaccess{#2}}%
10006 }

alaccessdisplay  As above but for the shortpluralaccess replacement text.
10007 \DeclareRobustCommand*\glsshortpluralaccessdisplay}[2]{%
10008  \@gls@access@display{#1}{\glsentryshortpluralaccess{#2}}%
10009 }

ngaccessdisplay  As above but for the longaccess replacement text.
10010 \DeclareRobustCommand*\glslongaccessdisplay}[2]{%
10011  \@gls@access@display{#1}{\glsentrylongaccess{#2}}%
10012 }

alaccessdisplay  As above but for the longpluralaccess replacement text.
10013 \DeclareRobustCommand*\glslongpluralaccessdisplay}[2]{%
10014  \@gls@access@display{#1}{\glsentrylongpluralaccess{#2}}%
10015 }

```

`glsaccessdisplay` Gets the replacement text corresponding to the named key given by the first argument and calls the appropriate command defined above.

```
10016 \DeclareRobustCommand*{\glsaccessdisplay}[3]{%
10017   \@ifundefined{gls#1accessdisplay}%
10018   {%
10019     \PackageError{glossaries-accsupp}{No accessibility support
10020       for key ‘#1’}{%
10021   }%
10022   {%
10023     \csname gls#1accessdisplay\endcsname{#2}{#3}%
10024   }%
10025 }
```

`default@entryfmt` Redefine the default entry format to use accessibility information

```
10026 \renewcommand*{\@@gls@default@entryfmt}[2]{%
10027   \ifdefempty\glscustomtext
10028   {%
10029     \glsifplural
10030     {%
10031       Plural form
10032       \glscapscase
10033       \ifglsused\glslabel
10034       {%
10035         Subsequent use
10036         #2{\glspluralaccessdisplay
10037           {\glsentryplural{\glslabel}}{\glslabel}}%
10038         {\glsdescriptionpluralaccessdisplay
10039           {\glsentrydescplural{\glslabel}}{\glslabel}}%
10040         {\glsymbolpluralaccessdisplay
10041           {\glsentrysymbolplural{\glslabel}}{\glslabel}}
10042         {\glsinsert}%
10043       }%
10044       First use
10045       #1{\glsfirstpluralaccessdisplay
10046         {\glsentryfirstplural{\glslabel}}{\glslabel}}%
10047         {\glsdescriptionpluralaccessdisplay
10048         {\glsentrydescplural{\glslabel}}{\glslabel}}%
10049         {\glsymbolpluralaccessdisplay
10050         {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10051         {\glsinsert}%
10052       }%
10053     }%
10054   }
```

Make first letter upper case

10054 \ifglsused\glslabel
10055 {%

Subsequent use.

10056 #2{\glspluralaccessdisplay
10057 {\Glsentryplural{\glslabel}}{\glslabel}}%
10058 {\glsdescriptionpluralaccessdisplay
10059 {\glsentrydescplural{\glslabel}}{\glslabel}}%
10060 {\glsymbolpluralaccessdisplay
10061 {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10062 {\glsinsert}%
10063 }%
10064 {%

First use

10065 #1{\glsfirstpluralaccessdisplay
10066 {\Glsentryfirstplural{\glslabel}}{\glslabel}}%
10067 {\glsdescriptionpluralaccessdisplay
10068 {\glsentrydescplural{\glslabel}}{\glslabel}}%
10069 {\glsymbolpluralaccessdisplay
10070 {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10071 {\glsinsert}%
10072 }%
10073 }%
10074 {%

Make all upper case

10075 \ifglsused\glslabel
10076 {%

Subsequent use

10077 \MakeUppercase{%
10078 #2{\glspluralaccessdisplay
10079 {\glsentryplural{\glslabel}}{\glslabel}}%
10080 {\glsdescriptionpluralaccessdisplay
10081 {\glsentrydescplural{\glslabel}}{\glslabel}}%
10082 {\glsymbolpluralaccessdisplay
10083 {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10084 {\glsinsert}}%
10085 }%
10086 {%

First use

10087 \MakeUppercase{%
10088 #1{\glsfirstpluralaccessdisplay
10089 {\glsentryfirstplural{\glslabel}}{\glslabel}}%
10090 {\glsdescriptionpluralaccessdisplay
10091 {\glsentrydescplural{\glslabel}}{\glslabel}}%
10092 {\glsymbolpluralaccessdisplay
10093 {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10093 {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10093 {\glsentrysymbolplural{\glslabel}}{\glslabel}}%

```

10094         {\glsinsert}}%
10095     }%
10096 }%
10097 }%
10098 {%
```

Singular form

```

10099     \gls caps case
10100     {%
```

Don't adjust case

```

10101     \ifglsused\glslabel
10102     {%
```

Subsequent use

```

10103     #2{\gls text access display
10104         {\gls entry text{\glslabel}}{\glslabel}}%
10105     {\gls description access display
10106         {\gls entry desc{\glslabel}}{\glslabel}}%
10107     {\gls symbol access display
10108         {\gls entry symbol{\glslabel}}{\glslabel}}%
10109     {\glsinsert}%
10110 }%
10111 {%
```

First use

```

10112     #1{\gls first access display
10113         {\gls entry first{\glslabel}}{\glslabel}}%
10114     {\gls description access display
10115         {\gls entry desc{\glslabel}}{\glslabel}}%
10116     {\gls symbol access display
10117         {\gls entry symbol{\glslabel}}{\glslabel}}%
10118     {\glsinsert}%
10119 }%
10120 }%
10121 {%
```

Make first letter upper case

```

10122     \ifglsused\glslabel
10123     {%
```

Subsequent use

```

10124     #2{\gls text access display
10125         {\Gls entry text{\glslabel}}{\glslabel}}%
10126     {\gls description access display
10127         {\Gls entry desc{\glslabel}}{\glslabel}}%
10128     {\gls symbol access display
10129         {\Gls entry symbol{\glslabel}}{\glslabel}}%
10130     {\glsinsert}%
10131 }%
10132 {%
```

First use

```
10133      #1{\glsfirstaccessdisplay
10134          {\Glsentryfirst{\glslabel}}{\glslabel}}%
10135          {\glsdescriptionaccessdisplay
10136           {\glsentrydesc{\glslabel}}{\glslabel}}%
10137          {\glsymbolaccessdisplay
10138           {\glsentrysymbol{\glslabel}}{\glslabel}}%
10139          {\glsinsert}}%
10140      }%
10141  }%
10142  {%
```

Make all upper case

```
10143      \ifglsused\glslabel
10144      {%
```

Subsequent use

```
10145      \MakeUppercase{%
10146          #2{\glsfirstaccessdisplay
10147             {\glsentrytext{\glslabel}}{\glslabel}}%
10148             {\glsdescriptionaccessdisplay
10149              {\glsentrydesc{\glslabel}}{\glslabel}}%
10150             {\glsymbolaccessdisplay
10151              {\glsentrysymbol{\glslabel}}{\glslabel}}%
10152             {\glsinsert}}%
10153      }%
10154  {%
```

First use

```
10155      \MakeUppercase{%
10156          #1{\glsfirstaccessdisplay
10157             {\glsentryfirst{\glslabel}}{\glslabel}}%
10158             {\glsdescriptionaccessdisplay
10159              {\glsentrydesc{\glslabel}}{\glslabel}}%
10160             {\glsymbolaccessdisplay
10161              {\glsentrysymbol{\glslabel}}{\glslabel}}%
10162             {\glsinsert}}%
10163      }%
10164  }%
10165 }%
10166 }%
10167 {%
```

Custom text provided in \glsdisp

```
10168      \ifglsused{\glslabel}%
10169      {%
```

Subsequent use

```
10170      #2{\glscustomtext}%
10171      {\glsdescriptionaccessdisplay
10172       {\glsentrydesc{\glslabel}}{\glslabel}}%
```

```

10173      {\glssymbolaccessdisplay
10174      {\glentrysymbol{\glslabel}}{\glslabel}}%
10175      {\glsinsert}}%
10176  }%
10177  {%

```

First use

```

10178      #1{\glscustomtext}}%
10179      {\glsdescriptionaccessdisplay
10180      {\glentrydesc{\glslabel}}{\glslabel}}%
10181      {\glssymbolaccessdisplay
10182      {\glentrysymbol{\glslabel}}{\glslabel}}%
10183      {\glsinsert}}%
10184  }%
10185  }%
10186 }

```

`\glsentryfmt` Redefine to use accessibility information.

```

10187 \renewcommand*{\glsentryfmt}{%
10188   \ifdefempty\glscustomtext
10189   {%
10190     \glsifplural
10191     {%

```

Plural form

```

10192     \glscapscase
10193     {%

```

Don't adjust case

```

10194     \ifglsused\glslabel
10195     {%

```

Subsequent use

```

10196     \glspluralaccessdisplay
10197     {\glentryplural{\glslabel}}{\glslabel}}%
10198     {\glsinsert
10199     }%
10200     {%

```

First use

```

10201     \glsfirstpluralaccessdisplay
10202     {\glentryfirstplural{\glslabel}}{\glslabel}}%
10203     {\glsinsert
10204     }%
10205     }%
10206     {%

```

Make first letter upper case

```

10207     \ifglsused\glslabel
10208     {%

```

Subsequent use.

10209 \backslash glspluralaccessdisplay
10210 $\{\backslash$ Gsentryplural $\{\backslash$ glslabel $\}\{\backslash$ glslabel $\}$ %
10211 \backslash glsinsert
10212 $\}$ %
10213 $\{$ %

First use

10214 \backslash glsfirstpluralaccessdisplay
10215 $\{\backslash$ Gsentryfirstplural $\{\backslash$ glslabel $\}\{\backslash$ glslabel $\}$ %
10216 \backslash glsinsert
10217 $\}$ %
10218 $\}$ %
10219 $\{$ %

Make all upper case

10220 \backslash ifglsused \backslash glslabel
10221 $\{$ %

Subsequent use

10222 \backslash glspluralaccessdisplay
10223 $\{\backslash$ mfirstucMakeUppercase $\{\backslash$ gsentryplural $\{\backslash$ glslabel $\}\}\}$ %
10224 $\{\backslash$ glslabel $\}$ %
10225 \backslash mfirstucMakeUppercase $\{\backslash$ glsinsert $\}$ %
10226 $\}$ %
10227 $\{$ %

First use

10228 \backslash glsfirstpluralaccessdisplay
10229 $\{\backslash$ mfirstucMakeUppercase $\{\backslash$ gsentryfirstplural $\{\backslash$ glslabel $\}\}\}$ %
10230 $\{\backslash$ glslabel $\}$ %
10231 \backslash mfirstucMakeUppercase $\{\backslash$ glsinsert $\}$ %
10232 $\}$ %
10233 $\}$ %
10234 $\}$ %
10235 $\{$ %

Singular form

10236 \backslash glscapscase
10237 $\{$ %

Don't adjust case

10238 \backslash ifglsused \backslash glslabel
10239 $\{$ %

Subsequent use

10240 \backslash glstextaccessdisplay $\{\backslash$ gsentrytext $\{\backslash$ glslabel $\}\{\backslash$ glslabel $\}$ %
10241 \backslash glsinsert
10242 $\}$ %
10243 $\{$ %

First use

```
10244      \glsfirstaccessdisplay{\glsentryfirst{\glslabel}}{\glslabel}%
10245      \glsinsert
10246      }%
10247      }%
10248      {%
```

Make first letter upper case

```
10249      \ifglsused\glslabel
10250      {%
```

Subsequent use

```
10251      \glstextaccessdisplay{\Glsentrytext{\glslabel}}{\glslabel}%
10252      \glsinsert
10253      }%
10254      {%
```

First use

```
10255      \glsfirstaccessdisplay{\Glsentryfirst{\glslabel}}{\glslabel}%
10256      \glsinsert
10257      }%
10258      }%
10259      {%
```

Make all upper case

```
10260      \ifglsused\glslabel
10261      {%
```

Subsequent use

```
10262      \glstextaccessdisplay
10263      {\mfirstucMakeUppercase{\glsentrytext{\glslabel}}}{\glslabel}%
10264      \mfirstucMakeUppercase{\glsinsert}%
10265      }%
10266      {%
```

First use

```
10267      \glsfirstaccessdisplay
10268      {\mfirstucMakeUppercase{\glsentryfirst{\glslabel}}}{\glslabel}%
10269      \mfirstucMakeUppercase{\glsinsert}%
10270      }%
10271      }%
10272      }%
10273      }%
10274      {%
```

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.

```
10275      \glscustomtext\glsinsert
10276      }%
10277      }
```


`\glsgenacfmt` Redefine to include accessibility information.

```
10278 \renewcommand*{\glsgenacfmt}{%
10279   \ifdefempty\glscustomtext
10280   {%
10281     \ifglused\glslabel
10282     {%
```

Subsequent use:

```
10283     \glsifplural
10284     {%
```

Subsequent plural form:

```
10285     \glscapscase
10286     {%
```

Subsequent plural form, don't adjust case:

```
10287     \acronymfont
10288     {\glsshortpluralaccessdisplay
10289      {\glentryshortpl{\glslabel}}{\glslabel}}%
10290     \glsinsert
10291     }%
10292     {%
```

Subsequent plural form, make first letter upper case:

```
10293     \acronymfont
10294     {\glsshortpluralaccessdisplay
10295      {\Glsentryshortpl{\glslabel}}{\glslabel}}%
10296     \glsinsert
10297     }%
10298     {%
```

Subsequent plural form, all caps:

```
10299     \mfirstucMakeUppercase
10300     {\acronymfont
10301      {\glsshortpluralaccessdisplay
10302       {\glentryshortpl{\glslabel}}{\glslabel}}%
10303      \glsinsert}%
10304     }%
10305     }%
10306     {%
```

Subsequent singular form

```
10307     \glscapscase
10308     {%
```

Subsequent singular form, don't adjust case:

```
10309     \acronymfont
10310     {\glsshortaccessdisplay{\glentryshort{\glslabel}}{\glslabel}}%
10311     \glsinsert
10312     }%
10313     {%
```

Subsequent singular form, make first letter upper case:

```
10314      \acronymfont
10315      {\glsshortaccessdisplay{\Glsentryshort{\glslabel}}{\glslabel}}%
10316      \glsinsert
10317      }%
10318      {%
```

Subsequent singular form, all caps:

```
10319      \mfirstucMakeUppercase
10320      {\acronymfont{%
10321      \glsshortaccessdisplay{\glsentryshort{\glslabel}}{\glslabel}}%
10322      \glsinsert}%
10323      }%
10324      }%
10325      }%
10326      {%
```

First use:

```
10327      \glsifplural
10328      {%
```

First use plural form:

```
10329      \glscapscase
10330      {%
```

First use plural form, don't adjust case:

```
10331      \genplacrfullformat{\glslabel}{\glsinsert}%
10332      }%
10333      {%
```

First use plural form, make first letter upper case:

```
10334      \Genplacrfullformat{\glslabel}{\glsinsert}%
10335      }%
10336      {%
```

First use plural form, all caps:

```
10337      \mfirstucMakeUppercase
10338      {\genplacrfullformat{\glslabel}{\glsinsert}}%
10339      }%
10340      }%
10341      {%
```

First use singular form

```
10342      \glscapscase
10343      {%
```

First use singular form, don't adjust case:

```
10344      \genacrfullformat{\glslabel}{\glsinsert}%
10345      }%
10346      {%
```

First use singular form, make first letter upper case:

```
10347      \Genacrfullformat{\glslabel}{\glsinsert}%
10348      }%
10349      {%
```

First use singular form, all caps:

```
10350      \mfirstucMakeUppercase
10351      {\genacrfullformat{\glslabel}{\glsinsert}}%
10352      }%
10353      }%
10354      }%
10355      }%
10356      {%
```

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.

```
10357      \glscustomtext
10358      }%
10359      }
```

enacrfullformat Redefine to include accessibility information.

```
10360 \renewcommand*{\genacrfullformat}[2]{%
10361   \glslongaccessdisplay{\glsentrylong{#1}}{#1}#2\space
10362   (\glsshortaccessdisplay{\protect\firstacronymfont{\glsentryshort{#1}}}{#1})%
10363 }
```

enacrfullformat Redefine to include accessibility information.

```
10364 \renewcommand*{\Genacrfullformat}[2]{%
10365   \glslongaccessdisplay{\Glsentrylong{#1}}{#1}#2\space
10366   (\glsshortaccessdisplay{\protect\firstacronymfont{\Glsentryshort{#1}}}{#1})%
10367 }
```

placrfullformat Redefine to include accessibility information.

```
10368 \renewcommand*{\genplacrfullformat}[2]{%
10369   \glslongpluralaccessdisplay{\glsentrylongpl{#1}}{#1}#2\space
10370   (\glsshortpluralaccessdisplay
10371     {\protect\firstacronymfont{\glsentryshortpl{#1}}}{#1})%
10372 }
```

placrfullformat Redefine to include accessibility information.

```
10373 \renewcommand*{\Genplacrfullformat}[2]{%
10374   \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}#2\space
10375   (\glsshortpluralaccessdisplay
10376     {\protect\firstacronymfont{\glsentryshortpl{#1}}}{#1})%
10377 }
```

\@acrshort

```
10378 \def\@acrshort#1#2[#3]{%
10379   \glsdoifexists{#2}%
```

```

10380 {%
10381   \let\do@gls@link@checkfirsthyper\relax

10382   \let\glsifplural\@secondoftwo
10383   \let\glsapscase\@firstofthree
10384   \let\glsinsert\@empty
10385   \def\glscustomtext{%
10386     \acronymfont{\glsshortaccessdisplay{\glsentryshort{#2}}{#2}}#3%
10387   }%

   Call \@gls@link
10388   \@gls@link[#1]{#2}{\cename gls@\glstype @entryfmt\endcename}%
10389 }%

10390 \glspostlinkhook
10391 }

```

\@Acrshort

```

10392 \def\@Acrshort#1#2[#3]{%
10393   \glsdoifexists{#2}%
10394   {%
10395     \let\do@gls@link@checkfirsthyper\relax

10396     \let\glsifplural\@secondoftwo
10397     \let\glsapscase\@secondofthree
10398     \let\glsinsert\@empty
10399     \def\glscustomtext{%
10400       \acronymfont{\glsshortaccessdisplay{\Glsentryshort{#2}}{#2}}#3%
10401     }%

     Call \@gls@link
10402     \@gls@link[#1]{#2}{\cename gls@\glstype @entryfmt\endcename}%
10403   }%

10404   \glspostlinkhook
10405 }

```

\@ACRshort

```

10406 \def\@ACRshort#1#2[#3]{%
10407   \glsdoifexists{#2}%
10408   {%
10409     \let\do@gls@link@checkfirsthyper\relax

10410     \let\glsifplural\@secondoftwo
10411     \let\glsapscase\@thirdofthree
10412     \let\glsinsert\@empty
10413     \def\glscustomtext{%
10414       \acronymfont{\glsshortaccessdisplay
10415         {\MakeUppercase{\glsentryshort{#2}}}{#2}}#3%
10416     }%

```

```

Call \@gls@link
10417   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10418   }%

10419   \glspostlinkhook
10420 }

```

\@acrlong

```

10421 \def\@acrlong#1#2[#3]{%
10422   \glsdoifexists{#2}%
10423   {%
10424     \let\do@gls@link@checkfirsthyper\relax

10425     \let\glsifplural\@secondoftwo
10426     \let\glscapscase\@firstofthree
10427     \let\glsinsert\@empty
10428     \def\glscustomtext{%
10429       \acronymfont{\glslongaccessdisplay{\glsentrylong{#2}}{#2}}#3%
10430     }%

```

Call \@gls@link

```

10431   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10432   }%

10433   \glspostlinkhook
10434 }

```

\@Acrlong

```

10435 \def\@Acrlong#1#2[#3]{%
10436   \glsdoifexists{#2}%
10437   {%
10438     \let\do@gls@link@checkfirsthyper\relax

10439     \let\glsifplural\@secondoftwo
10440     \let\glscapscase\@firstofthree
10441     \let\glsinsert\@empty
10442     \def\glscustomtext{%
10443       \acronymfont{\glslongaccessdisplay{\Glsentrylong{#2}}{#2}}#3%
10444     }%

```

Call \@gls@link

```

10445   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10446   }%

10447   \glspostlinkhook
10448 }

```

\@ACRlong

```

10449 \def\@ACRlong#1#2[#3]{%
10450   \glsdoifexists{#2}%
10451   {%
10452     \let\do@gls@link@checkfirsthyper\relax

```

```

10453 \let\glsifplural\@secondoftwo
10454 \let\glsifscaps\@firstofthree
10455 \let\glsinsert\@empty
10456 \def\glscustomtext{%
10457 \acronymfont{\glslongaccessdisplay{%
10458 \MakeUppercase{\glsentrylong{#2}}}{#2}#3}%
10459 }%

Call \@gls@link
10460 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10461 }%

10462 \glspostlinkhook
10463 }

```

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.

```

10464 \renewcommand*\glossentryname}[1]{%
10465 \glsdoifexists{#1}%
10466 {%
10467 \glsnamefont{\glsnameaccessdisplay{\glsentryname{#1}}{#1}}%
10468 }%
10469 }

10470 \renewcommand*\glossentrydesc}[1]{%
10471 \glsdoifexists{#1}%
10472 {%
10473 \glsnamefont{\glsnameaccessdisplay{\Glsentryname{#1}}{#1}}%
10474 }%
10475 }

10476 \renewcommand*\glossentrydesc}[1]{%
10477 \glsdoifexists{#1}%
10478 {%
10479 \glsdescriptionaccessdisplay{\glsentrydesc{#1}}{#1}%
10480 }%
10481 }

10482 \renewcommand*\Glossentrydesc}[1]{%
10483 \glsdoifexists{#1}%
10484 {%
10485 \glsdescriptionaccessdisplay{\Glsentrydesc{#1}}{#1}%
10486 }%
10487 }

```

```

10488 \renewcommand*{\glossentrysymbol}[1]{%
10489   \glsdoifexists{#1}%
10490   {%
10491     \glssymbolaccessdisplay{\glsentrysymbol{#1}}{#1}%
10492   }%
10493 }

10494 \renewcommand*{\Glossentrysymbol}[1]{%
10495   \glsdoifexists{#1}%
10496   {%
10497     \glssymbolaccessdisplay{\Glsentrysymbol{#1}}{#1}%
10498   }%
10499 }

```

ssaryentryfield

```

10500 \newcommand*{\accsuppglossaryentryfield}[5]{%
10501   \glossaryentryfield{#1}%
10502   {\glsnameaccessdisplay{#2}{#1}}%
10503   {\glsdescriptionaccessdisplay{#3}{#1}}%
10504   {\glssymbolaccessdisplay{#4}{#1}}{#5}%
10505 }

```

rysubentryfield

```

10506 \newcommand*{\accsuppglossarysubentryfield}[6]{%
10507   \glossarysubentryfield{#1}{#2}%
10508   {\glsnameaccessdisplay{#3}{#2}}%
10509   {\glsdescriptionaccessdisplay{#4}{#2}}%
10510   {\glssymbolaccessdisplay{#5}{#2}}{#6}%
10511 }

```

5.4 Acronyms

Redefine acronym styles provided by glossaries:

long-short *<long>* (*<short>*) acronym style.

```

10512 \renewacronymstyle{long-short}%
10513 {%

```

Check for long form in case this is a mixed glossary.

```

10514   \ifglshaslong{\glslabel}{\glsngenacfmt}{\glsngenentryfmt}%
10515 }%
10516 {%
10517   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
10518   \renewcommand*{\genacrfullformat}[2]{%
10519     \glslongaccessdisplay{\glsentrylong{##1}}{##1}##2\space
10520     (\glsshortaccessdisplay
10521       {\protect\firstacronymfont{\glsentryshort{##1}}}{##1})%
10522   }%
10523   \renewcommand*{\Genacrfullformat}[2]{%

```

```

10524 \glslongaccessdisplay{\Glsentrylong{##1}}{##1}##2\space
10525 (\glsshortaccessdisplay
10526   {\protect\firstacronymfont{\glsentryshort{##1}}}{##1})%
10527 }%
10528 \renewcommand*{\genplacrformat}[2]{%
10529   \glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}##2\space
10530   (\glsshortpluralaccessdisplay
10531     {\protect\firstacronymfont{\glsentryshortpl{##1}}}{##1})%
10532   }%
10533 \renewcommand*{\Genplacrformat}[2]{%
10534   \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}##2\space
10535   (\glsshortpluralaccessdisplay
10536     {\protect\firstacronymfont{\glsentryshortpl{##1}}}{##1})%
10537   }%
10538 \renewcommand*{\acronymentry}[1]{%
10539   \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}
10540 \renewcommand*{\acronymsort}[2]{##1}%
10541 \renewcommand*{\acronymfont}[1]{##1}%
10542 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
10543 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
10544 }

```

short-long (*short*) (*long*) acronym style.

```

10545 \renewacronymstyle{short-long}%
10546 {%

```

Check for long form in case this is a mixed glossary.

```

10547 \ifglshaslong{\glslabel}{\glsngenacfmt}{\glsngenentryfmt}%
10548 }%
10549 {%
10550 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
10551 \renewcommand*{\genacrformat}[2]{%
10552   \glsshortaccessdisplay
10553     {\protect\firstacronymfont{\glsentryshort{##1}}}{##1}##2\space
10554   (\glslongaccessdisplay{\glsentrylong{##1}}{##1})%
10555   }%
10556 \renewcommand*{\Genacrformat}[2]{%
10557   \glsshortaccessdisplay
10558     {\protect\firstacronymfont{\Glsentryshort{##1}}}{##1}##2\space
10559   (\glslongaccessdisplay{\Glsentrylong{##1}}{##1})%
10560   }%
10561 \renewcommand*{\genplacrformat}[2]{%
10562   \glsshortpluralaccessdisplay
10563     {\protect\firstacronymfont{\glsentryshortpl{##1}}}{##1}##2\space
10564   (\glslongpluralaccessdisplay
10565     {\glsentrylongpl{##1}}{##1})%
10566   }%
10567 \renewcommand*{\Genplacrformat}[2]{%
10568   \glsshortpluralaccessdisplay
10569     {\protect\firstacronymfont{\Glsentryshortpl{##1}}}{##1}##2\space

```



```

10570 (\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1})%
10571 }%
10572 \renewcommand*{\acronymentry}[1]{%
10573   \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}{##1}}%
10574 \renewcommand*{\acronymsort}[2]{##1}%
10575 \renewcommand*{\acronymfont}[1]{##1}%
10576 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
10577 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
10578 }

```

long-short-desc *long* (*short*) acronym style that has an accompanying description (which the user needs to supply).

```

10579 \renewacronymstyle{long-short-desc}%
10580 {%
10581   \GlsUseAcrEntryDispStyle{long-short}%
10582 }%
10583 {%
10584   \GlsUseAcrStyleDefs{long-short}%
10585 \renewcommand*{\GenericAcronymFields}{}%
10586 \renewcommand*{\acronymsort}[2]{##2}%
10587 \renewcommand*{\acronymentry}[1]{%
10588   \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10589   (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}{##1})}%
10590 }

```

g-sc-short-desc *long* (\textsc{short}) acronym style that has an accompanying description (which the user needs to supply).

```

10591 \renewacronymstyle{long-sc-short-desc}%
10592 {%
10593   \GlsUseAcrEntryDispStyle{long-sc-short}%
10594 }%
10595 {%
10596   \GlsUseAcrStyleDefs{long-sc-short}%
10597 \renewcommand*{\GenericAcronymFields}{}%
10598 \renewcommand*{\acronymsort}[2]{##2}%
10599 \renewcommand*{\acronymentry}[1]{%
10600   \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10601   (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}{##1})}%
10602 }

```

g-sm-short-desc *long* (\textsmaller{short}) acronym style that has an accompanying description (which the user needs to supply).

```

10603 \renewacronymstyle{long-sm-short-desc}%
10604 {%
10605   \GlsUseAcrEntryDispStyle{long-sm-short}%
10606 }%
10607 {%
10608   \GlsUseAcrStyleDefs{long-sm-short}%
10609 \renewcommand*{\GenericAcronymFields}{}%

```

```

10610 \renewcommand*\acronymsort}[2]{##2}%
10611 \renewcommand*\acronymentry}[1]{%
10612   \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10613   (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10614 }

```

short-long-desc *<short>* (*<long>*) acronym style that has an accompanying description (which the user needs to supply).

```

10615 \renewacronymstyle{short-long-desc}%
10616 {%
10617   \GlsUseAcrEntryDispStyle{short-long}%
10618 }%
10619 {%
10620   \GlsUseAcrStyleDefs{short-long}%
10621   \renewcommand*\GenericAcronymFields{}%
10622   \renewcommand*\acronymsort}[2]{##2}%
10623   \renewcommand*\acronymentry}[1]{%
10624     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10625     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10626 }

```

short-long-desc *<long>* (`\textsc{<short>}`) acronym style that has an accompanying description (which the user needs to supply).

```

10627 \renewacronymstyle{sc-short-long-desc}%
10628 {%
10629   \GlsUseAcrEntryDispStyle{sc-short-long}%
10630 }%
10631 {%
10632   \GlsUseAcrStyleDefs{sc-short-long}%
10633   \renewcommand*\GenericAcronymFields{}%
10634   \renewcommand*\acronymsort}[2]{##2}%
10635   \renewcommand*\acronymentry}[1]{%
10636     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10637     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10638 }

```

short-long-desc *<long>* (`\textsmaller{<short>}`) acronym style that has an accompanying description (which the user needs to supply).

```

10639 \renewacronymstyle{sm-short-long-desc}%
10640 {%
10641   \GlsUseAcrEntryDispStyle{sm-short-long}%
10642 }%
10643 {%
10644   \GlsUseAcrStyleDefs{sm-short-long}%
10645   \renewcommand*\GenericAcronymFields{}%
10646   \renewcommand*\acronymsort}[2]{##2}%
10647   \renewcommand*\acronymentry}[1]{%
10648     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10649     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%

```

10650 }

dua *<long>* only acronym style.

10651 \renewacronymstyle{dua}%
10652 {%

Check for long form in case this is a mixed glossary.

10653 \ifdefempty\glscustomtext
10654 {%
10655 \ifglshaslong{\glslabel}%
10656 {%
10657 \glsifplural
10658 {%

Plural form:

10659 \glscapscase
10660 {%

Plural form, don't adjust case:

10661 \glslongpluralaccessdisplay{\glsentrylongpl{\glslabel}}{\glslabel}%
10662 \glsinsert
10663 }%
10664 {%

Plural form, make first letter upper case:

10665 \glslongpluralaccessdisplay{\Glsentrylongpl{\glslabel}}{\glslabel}%
10666 \glsinsert
10667 }%
10668 {%

Plural form, all caps:

10669 \glslongpluralaccessdisplay
10670 {\mfirstucMakeUppercase{\glsentrylongpl{\glslabel}}}{\glslabel}%
10671 \mfirstucMakeUppercase{\glsinsert}%
10672 }%
10673 }%
10674 {%

Singular form

10675 \glscapscase
10676 {%

Singular form, don't adjust case:

10677 \glslongaccessdisplay{\glsentrylong{\glslabel}}{\glslabel}\glsinsert
10678 }%
10679 {%

Subsequent singular form, make first letter upper case:

10680 \glslongaccessdisplay{\Glsentrylong{\glslabel}}{\glslabel}\glsinsert
10681 }%
10682 {%

Subsequent singular form, all caps:

```

10683     \glslongaccessdisplay
10684     {\mfirstucMakeUppercase
10685       {\glsentrylong{\glslabel}\glsinsert}}{\glslabel}%
10686     \mfirstucMakeUppercase{\glsinsert}%
10687   }%
10688 }%
10689 }%
10690 {%

```

Not an acronym:

```

10691     \glsgenentryfmt
10692   }%
10693 }%
10694 {\glscustomtext\glsinsert}%
10695}%
10696{%
10697 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
10698 \renewcommand*{\acrfullfmt}[3]{%
10699   \glslink[##1]{##2}{%
10700     \glslongaccessdisplay{\glsentrylong{##2}}{##2}##3\space
10701     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2})}}%
10702 \renewcommand*{\Acrfullfmt}[3]{%
10703   \glslink[##1]{##2}{%
10704     \glslongaccessdisplay{\Glsentrylong{##2}}{##2}##3\space
10705     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2})}}%
10706 \renewcommand*{\ACRfullfmt}[3]{%
10707   \glslink[##1]{##2}{%
10708     \glslongaccessdisplay
10709     {\mfirstucMakeUppercase{\glsentrylong{##2}}{##2}##3\space
10710     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2})}}}%
10711 \renewcommand*{\acrfullplfmt}[3]{%
10712   \glslink[##1]{##2}{%
10713     \glslongpluralaccessdisplay
10714     {\glsentrylongpl{##2}}{##2}##3\space
10715     (\glsshortpluralaccessdisplay
10716     {\acronymfont{\glsentryshortpl{##2}}}{##2})}}%
10717 \renewcommand*{\ACRfullplfmt}[3]{%
10718   \glslink[##1]{##2}{%
10719     \glslongpluralaccessdisplay
10720     {\Glsentrylongpl{##2}}{##2}##3\space
10721     (\glsshortpluralaccessdisplay
10722     {\acronymfont{\glsentryshortpl{##2}}}{##2})}}%
10723 \renewcommand*{\ACRfullplfmt}[3]{%
10724   \glslink[##1]{##2}{%
10725     \glslongpluralaccessdisplay
10726     {\mfirstucMakeUppercase{\glsentrylongpl{##2}}{##2}##3\space
10727     (\glsshortpluralaccessdisplay
10728     {\acronymfont{\glsentryshortpl{##2}}}{##2})}}}%
10729 \renewcommand*{\glsentryfull}[1]{%

```

```

10730 \glslongaccessdisplay{\glsentrylong{##1}}\space
10731 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})%
10732 }%
10733 \renewcommand*\Glsentryfull}[1]{%
10734 \glslongaccessdisplay{\Glsentrylong{##1}}{##1}\space
10735 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})%
10736 }%
10737 \renewcommand*\glsentryfullpl}[1]{%
10738 \glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}\space
10739 (\glsshortpluralaccessdisplay{\acronymfont{\glsentryshortpl{##1}}}{##1})%
10740 }%
10741 \renewcommand*\Glsentryfullpl}[1]{%
10742 \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}\space
10743 (\glsshortpluralaccessdisplay{\acronymfont{\glsentryshortpl{##1}}}{##1})%
10744 }%
10745 \renewcommand*\acronymentry}[1]{%
10746 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}%
10747 \renewcommand*\acronymsort}[2]{##1}%
10748 \renewcommand*\acronymfont}[1]{##1}%
10749 \renewcommand*\acrpluralsuffix{\glspluralsuffix}%
10750 }

```

dua-desc *<long>* only acronym style with user-supplied description.

```

10751 \renewacronymstyle{dua-desc}%
10752 {%
10753 \GlsUseAcrEntryDispStyle{dua}%
10754 }%
10755 {%
10756 \GlsUseAcrStyleDefs{dua}%
10757 \renewcommand*\GenericAcronymFields{}%
10758 \renewcommand*\acronymentry}[1]{%
10759 \glslongaccessdisplay{\acronymfont{\glsentrylong{##1}}}{##1}}%
10760 \renewcommand*\acronymsort}[2]{##2}%
10761 }%

```

footnote *<short>*\footnote{*<long>*} acronym style.

```

10762 \renewacronymstyle{footnote}%
10763 {%
    Check for long form in case this is a mixed glossary.
10764 \ifglsahaslong{\glslabel}{\glsngenacfmt}{\glsngenentryfmt}%
10765 }%
10766 {%
10767 \renewcommand*\GenericAcronymFields{description={\the\glslongtok}}%

```

Need to ensure hyperlinks are switched off on first use:

```

10768 \glshyperfirstfalse
10769 \renewcommand*\genacrfullformat}[2]{%
10770 \glsshortaccessdisplay
10771 {\protect\firstacronymfont{\glsentryshort{##1}}}{##1}##2%

```

```

10772 \protect\footnote{\glslongaccessdisplay{\glsentrylong{##1}}{##1}}%
10773 }%
10774 \renewcommand*{\Genacrfullformat}[2]{%
10775 \glsshortaccessdisplay
10776   {\firstacronymfont{\Glsentryshort{##1}}{##1}##2%
10777 \protect\footnote{\glslongaccessdisplay{\glsentrylong{##1}}{##1}}%
10778 }%
10779 \renewcommand*{\genplacrfullformat}[2]{%
10780 \glsshortpluralaccessdisplay
10781   {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}##2%
10782 \protect\footnote{\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}}%
10783 }%
10784 \renewcommand*{\Genplacrfullformat}[2]{%
10785 \glsshortpluralaccessdisplay
10786   {\protect\firstacronymfont{\Glsentryshortpl{##1}}{##1}##2%
10787 \protect\footnote{\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}}%
10788 }%
10789 \renewcommand*{\acronymentry}[1]{%
10790 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}{##1}}%
10791 \renewcommand*{\acronymsort}[2]{##1}%
10792 \renewcommand*{\acronymfont}[1]{##1}%
10793 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%

```

Don't use footnotes for \acrfull:

```

10794 \renewcommand*{\acrfullfmt}[3]{%
10795 \glslink[##1]{##2}{%
10796 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}{##2}##3\space
10797 (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}%
10798 \renewcommand*{\Acrfullfmt}[3]{%
10799 \glslink[##1]{##2}{%
10800 \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##2}}{##2}##3\space
10801 (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}%
10802 \renewcommand*{\ACRfullfmt}[3]{%
10803 \glslink[##1]{##2}{%
10804 \glsshortaccessdisplay
10805   {\mfirstucMakeUppercase
10806   {\acronymfont{\glsentryshort{##2}}{##2}##3\space
10807   (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}}%
10808 \renewcommand*{\acrfullplfmt}[3]{%
10809 \glslink[##1]{##2}{%
10810 \glsshortpluralaccessdisplay
10811   {\acronymfont{\glsentryshortpl{##2}}{##2}##3\space
10812   (\glslongpluralaccessdisplay{\glsentrylongpl{##2}}{##2})}}}%
10813 \renewcommand*{\Acrfullplfmt}[3]{%
10814 \glslink[##1]{##2}{%
10815 \glsshortpluralaccessdisplay
10816   {\acronymfont{\Glsentryshortpl{##2}}{##2}##3\space
10817   (\glslongpluralaccessdisplay{\glsentrylongpl{##2}}{##2})}}}%
10818 \renewcommand*{\ACRfullplfmt}[3]{%
10819 \glslink[##1]{##2}{%

```

```

10820 \glsshortpluralaccessdisplay
10821   {\mfirstucMakeUppercase
10822     {\acronymfont{\glentryshortpl{##2}}}{##2}##3\space
10823   (\glslongpluralaccessdisplay{\glentrylongpl{##2}}{##2})}}}%

```

Similarly for \glentryfull etc:

```

10824 \renewcommand*{\glentryfull}[1]{%
10825   \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}}{##1}\space
10826   (\glslongaccessdisplay{\glentrylong{##1}}{##1})}%
10827 \renewcommand*{\Glsentryfull}[1]{%
10828   \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##1}}}{##1}\space
10829   (\glslongaccessdisplay{\glentrylong{##1}}{##1})}%
10830 \renewcommand*{\glentryfullpl}[1]{%
10831   \glsshortpluralaccessdisplay
10832     {\acronymfont{\glentryshortpl{##1}}}{##1}\space
10833     (\glslongpluralaccessdisplay{\glentrylongpl{##1}}{##1})}%
10834 \renewcommand*{\Glsentryfullpl}[1]{%
10835   \glsshortpluralaccessdisplay
10836     {\acronymfont{\Glsentryshortpl{##1}}}{##1}\space
10837     (\glslongpluralaccessdisplay{\glentrylongpl{##1}}{##1})}%
10838 }

```

footnote-sc \textsc{<short>}\footnote{<long>} acronym style.

```

10839 \renewacronymstyle{footnote-sc}%
10840 {%
10841   \GlsUseAcrEntryDispStyle{footnote}%
10842 }%
10843 {%
10844   \GlsUseAcrStyleDefs{footnote}%
10845   \renewcommand{\acronymentry}[1]{%
10846     \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}}{##1}}
10847   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
10848   \renewcommand*{\acrpluralsuffix}{\glstextup{\glspluralsuffix}}%
10849 }%

```

footnote-sm \textsmaller{<short>}\footnote{<long>} acronym style.

```

10850 \renewacronymstyle{footnote-sm}%
10851 {%
10852   \GlsUseAcrEntryDispStyle{footnote}%
10853 }%
10854 {%
10855   \GlsUseAcrStyleDefs{footnote}%
10856   \renewcommand{\acronymentry}[1]{%
10857     \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}}{##1}}
10858   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
10859   \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
10860 }%

```

footnote-desc <short>\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

10861 \renewacronymstyle{footnote-desc}%
10862 {%
10863   \GlsUseAcrEntryDispStyle{footnote}%
10864 }%
10865 {%
10866   \GlsUseAcrStyleDefs{footnote}%
10867   \renewcommand*{\GenericAcronymFields}{}%
10868   \renewcommand*{\acronymsort}[2]{##2}%
10869   \renewcommand*{\acronymentry}[1]{%
10870     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10871     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10872 }

```

ootnote-sc-desc \textsc{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

10873 \renewacronymstyle{footnote-sc-desc}%
10874 {%
10875   \GlsUseAcrEntryDispStyle{footnote-sc}%
10876 }%
10877 {%
10878   \GlsUseAcrStyleDefs{footnote-sc}%
10879   \renewcommand*{\GenericAcronymFields}{}%
10880   \renewcommand*{\acronymsort}[2]{##2}%
10881   \renewcommand*{\acronymentry}[1]{%
10882     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10883     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10884 }

```

ootnote-sm-desc \textsmaller{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

10885 \renewacronymstyle{footnote-sm-desc}%
10886 {%
10887   \GlsUseAcrEntryDispStyle{footnote-sm}%
10888 }%
10889 {%
10890   \GlsUseAcrStyleDefs{footnote-sm}%
10891   \renewcommand*{\GenericAcronymFields}{}%
10892   \renewcommand*{\acronymsort}[2]{##2}%
10893   \renewcommand*{\acronymentry}[1]{%
10894     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10895     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10896 }

```

Use \newacronymhook to modify the key list to set the access text to the long version by default.

```

10897 \renewcommand*{\newacronymhook}{%
10898   \edef\@gls@keylist{shortaccess=\the\gls\longtok,%
10899     \the\glskeylisttok}%
10900   \expandafter\glskeylisttok\expandafter{\@gls@keylist}%

```


10901 }

ltNewAcronymDef Modify default style to use access text:

```
10902 \renewcommand*{\DefaultNewAcronymDef}{%
10903   \edef\@do@newglossaryentry{%
10904     \noexpand\newglossaryentry{\the\glslabeltok}%
10905     {%
10906       type=\acronymtype,%
10907       name={\the\glsshorttok},%
10908       description={\the\glslongtok},%
10909       descriptionaccess=\relax,
10910       text={\the\glsshorttok},%
10911       access={\noexpand\@glo@textaccess},%
10912       sort={\the\glsshorttok},%
10913       short={\the\glsshorttok},%
10914       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10915       shortaccess={\the\glslongtok},%
10916       long={\the\glslongtok},%
10917       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10918       descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10919       first={\noexpand\glslongaccessdisplay
10920         {\the\glslongtok}{\the\glslabeltok}\space
10921         {\noexpand\glsshortaccessdisplay
10922           {\the\glsshorttok}{\the\glslabeltok}}},%
10923       plural={\the\glsshorttok\acrpluralsuffix},%
10924       firstplural={\noexpand\glslongpluralaccessdisplay
10925         {\noexpand\@glo@longpl}{\the\glslabeltok}\space
10926         {\noexpand\glsshortpluralaccessdisplay
10927           {\noexpand\@glo@shortpl}{\the\glslabeltok}}},%
10928       firstaccess=\relax,
10929       firstpluralaccess=\relax,
10930       textaccess={\noexpand\@glo@shortaccess},%
10931       \the\glskeylisttok
10932     }%
10933   }%
10934   \let\@org@gls@assign@firstpl\gls@assign@firstpl
10935   \let\@org@gls@assign@plural\gls@assign@plural
10936   \let\@org@gls@assign@descplural\gls@assign@descplural
10937   \def\gls@assign@firstpl##1##2{%
10938     \@gls@expand@field{##1}{firstpl}{##2}%
10939   }%
10940   \def\gls@assign@plural##1##2{%
10941     \@gls@expand@field{##1}{plural}{##2}%
10942   }%
10943   \def\gls@assign@descplural##1##2{%
10944     \@gls@expand@field{##1}{descplural}{##2}%
10945   }%
10946   \@do@newglossaryentry
10947   \let\gls@assign@firstpl\@org@gls@assign@firstpl
```

```

10948 \let\gls@assign@plural\@org@gls@assign@plural
10949 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
10950 }

```

teNewAcronymDef

```

10951 \renewcommand*{\DescriptionFootnoteNewAcronymDef}{%
10952 \edef\@do@newglossaryentry{%
10953 \noexpand\newglossaryentry{\the\glslabeltok}%
10954 {%
10955 type=\acronymtype,%
10956 name={\noexpand\acronymfont{\the\glsshorttok}},%
10957 sort={\the\glsshorttok},%
10958 text={\the\glsshorttok},%
10959 short={\the\glsshorttok},%
10960 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10961 shortaccess={\the\glslongtok},%
10962 long={\the\glslongtok},%
10963 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10964 access={\noexpand\@glo@textaccess},%
10965 plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10966 symbol={\the\glslongtok},%
10967 symbolplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10968 firstpluralaccess=\relax,
10969 textaccess={\noexpand\@glo@shortaccess},%
10970 \the\glskeylisttok
10971 }%
10972 }%
10973 \let\@org@gls@assign@firstpl\gls@assign@firstpl
10974 \let\@org@gls@assign@plural\gls@assign@plural
10975 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
10976 \def\gls@assign@firstpl##1##2{%
10977 \@@gls@expand@field{##1}{firstpl}{##2}%
10978 }%
10979 \def\gls@assign@plural##1##2{%
10980 \@@gls@expand@field{##1}{plural}{##2}%
10981 }%
10982 \def\gls@assign@symbolplural##1##2{%
10983 \@@gls@expand@field{##1}{symbolplural}{##2}%
10984 }%
10985 \@do@newglossaryentry
10986 \let\gls@assign@plural\@org@gls@assign@plural
10987 \let\gls@assign@firstpl\@org@gls@assign@firstpl
10988 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
10989 }

```

onNewAcronymDef

```

10990 \renewcommand*{\DescriptionNewAcronymDef}{%
10991 \edef\@do@newglossaryentry{%
10992 \noexpand\newglossaryentry{\the\glslabeltok}%

```

```

10993   {%
10994     type=\acronymtype,%
10995     name={\noexpand
10996       \acronymformat{\the\glssshorttok}{\the\glslongtok}},%
10997     access={\noexpand\@glo@textaccess},%
10998     sort={\the\glssshorttok},%
10999     short={\the\glssshorttok},%
11000     shortplural={\the\glssshorttok\noexpand\acrpluralsuffix},%
11001     shortaccess={\the\glslongtok},%
11002     long={\the\glslongtok},%
11003     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11004     first={\the\glslongtok},%
11005     firstaccess=\relax,
11006     firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11007     text={\the\glssshorttok},%
11008     textaccess={\the\glslongtok},%
11009     plural={\the\glssshorttok\noexpand\acrpluralsuffix},%
11010     symbol={\noexpand\@glo@text},%
11011     symbolaccess={\noexpand\@glo@textaccess},%
11012     symbolplural={\noexpand\@glo@plural},%
11013     firstpluralaccess=\relax,
11014     textaccess={\noexpand\@glo@shortaccess},%
11015     \the\glskeylisttok}%
11016   }%
11017   \let\@org@gls@assign@firstpl\gls@assign@firstpl
11018   \let\@org@gls@assign@plural\gls@assign@plural
11019   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
11020   \def\gls@assign@firstpl##1##2{%
11021     \@gls@expand@field{##1}{firstpl}{##2}%
11022   }%
11023   \def\gls@assign@plural##1##2{%
11024     \@gls@expand@field{##1}{plural}{##2}%
11025   }%
11026   \def\gls@assign@symbolplural##1##2{%
11027     \@gls@expand@field{##1}{symbolplural}{##2}%
11028   }%
11029   \@do@newglossaryentry
11030   \let\gls@assign@firstpl\@org@gls@assign@firstpl
11031   \let\gls@assign@plural\@org@gls@assign@plural
11032   \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11033 }

```

teNewAcronymDef

```

11034 \renewcommand*{\FootnoteNewAcronymDef}{%
11035   \edef\@do@newglossaryentry{%
11036     \noexpand\newglossaryentry{\the\glslabeltok}%
11037     {%
11038       type=\acronymtype,%
11039       name={\noexpand\acronymfont{\the\glssshorttok}},%

```

```

11040     sort={\the\glsshorttok},%
11041     text={\the\glsshorttok},%
11042     textaccess={\the\glslongtok},%
11043     access={\noexpand\@glo@textaccess},%
11044     plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11045     short={\the\glsshorttok},%
11046     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11047     long={\the\glslongtok},%
11048     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11049     description={\the\glslongtok},%
11050     descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11051     \the\glskeylisttok
11052   }%
11053 }%
11054 \let\@org@gls@assign@plural\gls@assign@plural
11055 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11056 \let\@org@gls@assign@descplural\gls@assign@descplural
11057 \def\gls@assign@firstpl##1##2{%
11058   \@@gls@expand@field{##1}{firstpl}{##2}%
11059 }%
11060 \def\gls@assign@plural##1##2{%
11061   \@@gls@expand@field{##1}{plural}{##2}%
11062 }%
11063 \def\gls@assign@descplural##1##2{%
11064   \@@gls@expand@field{##1}{descplural}{##2}%
11065 }%
11066 \do@newglossaryentry
11067 \let\gls@assign@plural\@org@gls@assign@plural
11068 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11069 \let\gls@assign@descplural\@org@gls@assign@descplural
11070 }

```

11NewAcronymDef

```

11071 \renewcommand*{\SmallNewAcronymDef}{%
11072   \edef\@do@newglossaryentry{%
11073     \noexpand\newglossaryentry{\the\glslabeltok}%
11074     {%
11075       type=\acronymtype,%
11076       name={\noexpand\acronymfont{\the\glsshorttok}},%
11077       access={\noexpand\@glo@symbolaccess},%
11078       sort={\the\glsshorttok},%
11079       short={\the\glsshorttok},%
11080       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11081       shortaccess={\the\glslongtok},%
11082       long={\the\glslongtok},%
11083       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11084       text={\noexpand\@glo@short},%
11085       textaccess={\noexpand\@glo@shortaccess},%
11086       plural={\noexpand\@glo@shortpl},%

```

```

11087     first={\the\glslongtok},%
11088     firstaccess=\relax,
11089     firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11090     description={\noexpand\@glo@first},%
11091     descriptionplural={\noexpand\@glo@firstplural},%
11092     symbol={\the\glsshorttok},%
11093     symbolaccess={\the\glslongtok},%
11094     symbolplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11095     \the\glskeylisttok
11096   }%
11097 }%
11098 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11099 \let\@org@gls@assign@plural\gls@assign@plural
11100 \let\@org@gls@assign@descplural\gls@assign@descplural
11101 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
11102 \def\gls@assign@firstpl##1##2{%
11103   \@@gls@expand@field{##1}{firstpl}{##2}%
11104 }%
11105 \def\gls@assign@plural##1##2{%
11106   \@@gls@expand@field{##1}{plural}{##2}%
11107 }%
11108 \def\gls@assign@descplural##1##2{%
11109   \@@gls@expand@field{##1}{descplural}{##2}%
11110 }%
11111 \def\gls@assign@symbolplural##1##2{%
11112   \@@gls@expand@field{##1}{symbolplural}{##2}%
11113 }%
11114 \@do@newglossaryentry
11115 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11116 \let\gls@assign@plural\@org@gls@assign@plural
11117 \let\gls@assign@descplural\@org@gls@assign@descplural
11118 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11119 }

```

The following are kept for compatibility with versions before 3.0:

sshortaccesskey

```
11120 \newcommand*{\glsshortaccesskey}{\glsshortkey access}%
```

pluralaccesskey

```
11121 \newcommand*{\glsshortpluralaccesskey}{\glsshortpluralkey access}%
```

lslongaccesskey

```
11122 \newcommand*{\glslongaccesskey}{\glslongkey access}%
```

pluralaccesskey

```
11123 \newcommand*{\glslongpluralaccesskey}{\glslongpluralkey access}%
```

5.5 Debugging Commands

owglonameaccess

```
11124 \newcommand*{\showglonameaccess}[1]{%
11125   \expandafter\show\csname glo@\glsdetoklabel{#1}@textaccess\endcsname
11126 }
```

owglotextaccess

```
11127 \newcommand*{\showglotextaccess}[1]{%
11128   \expandafter\show\csname glo@\glsdetoklabel{#1}@textaccess\endcsname
11129 }
```

glopluralaccess

```
11130 \newcommand*{\showglopluralaccess}[1]{%
11131   \expandafter\show\csname glo@\glsdetoklabel{#1}@pluralaccess\endcsname
11132 }
```

wglofirstaccess

```
11133 \newcommand*{\showglofirstaccess}[1]{%
11134   \expandafter\show\csname glo@\glsdetoklabel{#1}@firstaccess\endcsname
11135 }
```

rstpluralaccess

```
11136 \newcommand*{\showglofirstpluralaccess}[1]{%
11137   \expandafter\show\csname glo@\glsdetoklabel{#1}@firstpluralaccess\endcsname
11138 }
```

glosymbolaccess

```
11139 \newcommand*{\showglosymbolaccess}[1]{%
11140   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolaccess\endcsname
11141 }
```

bolpluralaccess

```
11142 \newcommand*{\showglosymbolpluralaccess}[1]{%
11143   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolpluralaccess\endcsname
11144 }
```

owglodescaccess

```
11145 \newcommand*{\showglodescaccess}[1]{%
11146   \expandafter\show\csname glo@\glsdetoklabel{#1}@descaccess\endcsname
11147 }
```

escpluralaccess

```
11148 \newcommand*{\showglodescpluralaccess}[1]{%
11149   \expandafter\show\csname glo@\glsdetoklabel{#1}@descpluralaccess\endcsname
11150 }
```

wgloshortaccess

```
11151 \newcommand*{\showgloshortaccess}[1]{%  
11152   \expandafter\show\csname glo@\glsdetoklabel{#1}@shortaccess\endcsname  
11153 }
```

ortpluralaccess

```
11154 \newcommand*{\showgloshortpluralaccess}[1]{%  
11155   \expandafter\show\csname glo@\glsdetoklabel{#1}@shortpluralaccess\endcsname  
11156 }
```

owglolongaccess

```
11157 \newcommand*{\showglolongaccess}[1]{%  
11158   \expandafter\show\csname glo@\glsdetoklabel{#1}@longaccess\endcsname  
11159 }
```

ongpluralaccess

```
11160 \newcommand*{\showglolongpluralaccess}[1]{%  
11161   \expandafter\show\csname glo@\glsdetoklabel{#1}@longpluralaccess\endcsname  
11162 }
```

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.

```
11163 \NeedsTeXFormat{LaTeX2e}
11164 \ProvidesPackage{glossaries-babel}[2016/01/24 v4.21 (NLCT)]
```

Load tracklang to obtain language settings.

```
11165 \RequirePackage{tracklang}
11166 \let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
11167 \AnyTrackedLanguages
11168 {%
11169   \ForEachTrackedDialect{\this@dialect}{%
11170     \IfTrackedLanguageFileExists{\this@dialect}%
11171       {glossaries-}% prefix
11172       {.ldf}%
11173       {%
11174         \RequireGlossariesLang{\CurrentTrackedTag}%
11175       }%
11176       {%
11177         \PackageWarningNoLine{glossaries}%
11178           {No language module detected for ‘\this@dialect’.\MessageBreak
11179             Language modules need to be installed separately.\MessageBreak
11180             Please check on CTAN for a bundle called\MessageBreak
11181             ‘glossaries-\CurrentTrackedLanguage’ or similar}%
11182       }%
11183     }%
11184   }%
11185 }
```

6.1 Polyglossia Captions

Language support has now been split off into independent language modules.

```
11186 \NeedsTeXFormat{LaTeX2e}
11187 \ProvidesPackage{glossaries-polyglossia}[2016/01/24 v4.21 (NLCT)]
```

Load tracklang to obtain language settings.

```
11188 \RequirePackage{tracklang}
11189 \let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
11190 \AnyTrackedLanguages
```



```

11191 {%
11192   \ForEachTrackedDialect{\this@dialect}{%
11193     \IfTrackedLanguageFileExists{\this@dialect}%
11194     {glossaries-}% prefix
11195     {.ldf}%
11196     {%
11197       \RequireGlossariesLang{\CurrentTrackedTag}%
11198     }%
11199     {%
11200       \PackageWarningNoLine{glossaries}%
11201       {No language module detected for ‘\this@dialect’.\MessageBreak
11202       Language modules need to be installed separately.\MessageBreak
11203       Please check on CTAN for a bundle called\MessageBreak
11204       ‘glossaries-\CurrentTrackedLanguage’ or similar}%
11205     }%
11206   }%
11207 }%
11208 {}%

```

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)	
General: Added range facility in format key	107
\writeist: Added spaces after \delimN and \delimR in ist file	153
1.04 (2007-08-03)	
General: Added \glstextformat	91
1.05 (2007-08-10)	
\glossarysection: added \@mkboth to \glossarysection	36
\gls@defglossaryentry: Changed the default value of the sort key to just the value of the name key	75
1.07 (2007-09-13)	
\@gls@link: fixed bug caused by \theglsentrycounter setting the page number too soon	105
\glsadd: fixed bug caused by \theglsentrycounter setting the page number too soon	150
1.08 (2007-10-13)	
General: Added babel support	30
listgroup: changed listgroup style to use \glsgetgrouptitle	259
alllistgroup: changed alllistgroup style to use \glsgetgrouptitle	260
1.1 (2008-02-22)	
\@glossarysection: numbered sections and auto label added	37
\@gls@tmpb: changed \toksdef to \newtoks	109
\@gls@toc: numberline added	39
\@p@glossarysection: numbered sections and auto label added	38
General: amsgen now loaded (\new@ifnextchar needed)	4
translate: translate option added	21
\setglossarysection: new	37
numberedsection: numberedsection package option added	6
numberline: numberline option added ..	5
1.12 (2008-03-08)	
\@GLSpl: now uses \glsentrydescplural and \glsentrysymbolplural instead of \glsentrydesc and \glsentrysymbol	120
\@Glspl@: now uses \glsentrydescplural and \glsentrysymbolplural instead of \glsentrydesc and \glsentrysymbol	120
\@glspl@: now uses \glsentrydescplural and \glsentrysymbolplural instead of \glsentrydesc and \glsentrysymbol	119
General: added check for \hypertarget separate to \hyperlink (memoir defines \hyperlink but not \hypertarget)	115
descriptionplural: new	59
\gls@defglossaryentry: Changed default first plural to be first key with s appended (was text key with s appended)	75
descriptionplural support added	75
symbolplural support added	75
\Glsentrydescplural: New	144
\glsentrydescplural: New	144
\Glsentrysymbolplural: New	145
\glsentrysymbolplural: New	145
\SetDescriptionFootnoteAcronymStyle: Added \protect before \footnote and \glslink	227
\SetFootnoteAcronymStyle: Added \protect before \footnote and \glslink	233
symbolplural: new	60
1.13 (2008-05-10)	
General: fixed bug that ignored 3rd parameter	122–129
\ACRfullpl: new	208

<code>\Acrfullpl</code> : new	207	<code>\@gls@</code> : Test glossary type is <code>\acronymtype</code>	
<code>\acrfullpl</code> : new	207	in addition to checking if footnote op-	
<code>\acrpluralsuffix</code> : New	205	tion has been used	116
<code>\gls@defglossaryentry</code> : Changed de-		<code>\@glsdisp</code> : Test glossary type is	
fault first value	75	<code>\acronymtype</code> in addition to check-	
Changed default firstplural value	75	ing if footnote option has been used	121
Removed restriction on only using		<code>\@glspl@</code> : Test glossary type is	
<code>\newglossaryentry</code> in the preamble	80	<code>\acronymtype</code> in addition to check-	
<code>\newacronym</code> : Removed restriction on		ing if footnote option has been used	119
only using <code>\newacronym</code> in the preamble	205	<code>\@glstarget</code> : raised the hypertarget so	
1.14 (2008-06-17)		the target text doesn't scroll off the top	
<code>\@gls@hypergroup</code> : new	255	of the page	115
General: added nonnumberlist key to		<code>\gls@defglossaryentry</code> : Changed def	
<code>\printglossary</code>	191	to let	75
added numberedsection key to		1.17 (2008-12-26)	
<code>\printglossary</code>	189	<code>\@do@wrglossary</code> : new	170
<code>\firstacronymfont</code> : new	208	<code>\@do@seeglossary</code> : new	173
<code>\glsautoprefix</code> : new	6	<code>\@glo@storeentry</code> : new	81
<code>\glsnavhyperlink</code> : changed <code>\edef</code> to		<code>\@gls@glossary</code> : changed definition to	
<code>\protected@edef</code>	254	use <code>\index</code> instead of <code>\@index</code> ...	168
<code>\glsnavhypertarget</code> : added write to aux		<code>\@glsdefaultplural</code> : new	62
file	254	<code>\@glsdefaultsort</code> : new	63
<code>\glsnavigation</code> : changed to only use la-		<code>\@gls@hypernumber</code> : new	202
bel for groups that are present ...	255	<code>\@gls@name</code> : new	62
1.15 (2008-08-15)		<code>\@gls@nonextpages</code> : new	191
<code>\@gls@link</code> : added <code>\glslabel</code>	105	General: added xindy support	24
<code>\gls@defglossaryentry</code> : check for		parent: new	61
<code>\@glo@first</code> in description	79	see: new	60
check for <code>\@glo@text</code> in symbol	79	<code>\gls@defglossaryentry</code> : added non-	
<code>\gls@hypergroup@rerun</code> : new	255	umberlist key	76
<code>\glsnavhypertarget</code> : added check if re-		added parent key	75
run required	254	added see key	75
<code>\glssettoctitle</code> : new	29	Stored main part of entry format when	
<code>\printglossary</code> : changed the way the		entry is defined	80
TOC title is set	175	<code>\gls@suffixF</code> : new	34
1.16 (2008-08-27)		<code>\gls@suffixFF</code> : new	35
<code>\@GLS@</code> : Test glossary type is <code>\acronymtype</code>		<code>\gls@wrglossary</code> : modified to allow for	
in addition to checking if footnote op-		xindy support	168
tion has been used	118	<code>\gls@hyperlink</code> : new	150
<code>\@GLSpl</code> : Test glossary type is		<code>\gls@hypernumber</code> : modified to allow ma-	
<code>\acronymtype</code> in addition to check-		terial to be attached to location ...	201
ing if footnote option has been used	120	<code>\glsnavhyperlink</code> : replaced <code>\hyperlink</code>	
<code>\@Gls@</code> : Test glossary type is <code>\acronymtype</code>		to <code>\@glslink</code>	254
in addition to checking if footnote op-		<code>\glsnavhypertarget</code> :	
tion has been used	117	replaced	
<code>\@Glspl@</code> : Test glossary type is		<code>\hypertarget</code> to <code>\@glstarget</code> ...	254
<code>\acronymtype</code> in addition to check-		<code>\glssee</code> : new	174
ing if footnote option has been used	120	<code>\glsseeformat</code> : new	174
		<code>\glsSetSuffixF</code> : new	34
		<code>\glsSetSuffixFF</code> : new	35

<code>\ifglxindy</code> : new	24	before term is displayed to prevent un-	
<code>\istfilename</code> : added xindy support ...	33	wanted whatsit	105
<code>\newglossarystyle</code> : made <code>\newglossarystyle</code>		<code>\forallglossaries</code> : replaced <code>\ifthenelse</code>	
<code>long</code>	200	with <code>\ifx</code>	48
<code>\nopostdesc</code> : new	32	<code>\forallglsentries</code> : replaced <code>\ifthenelse</code>	
<code>nonumberlist</code> : new	61	with <code>\ifx</code>	48
<code>\printglossary</code> : added check to deter-		<code>\glsdefmain</code> : new	12
mine if <code>\printglossary</code> is already		<code>\glsdescwidth</code> : changed <code>\linewidth</code> to	
defined	175	<code>\hsize</code>	262, 281
added print language to aux file	175	<code>\glslistdottedwidth</code> : changed	
<code>order</code> : order package option added	24	<code>\linewidth</code> to <code>\hsize</code>	261
<code>\writeist</code> : added xindy support	153	<code>\glspagelistwidth</code> : changed <code>\linewidth</code>	
1.18 (2009-01-14)		to <code>\hsize</code>	262, 281
<code>\@gls@loadlist</code> : new	8	<code>nomain</code> : added <code>nomain</code> package option .	12
<code>\@gls@loadlong</code> : new	7	<code>\writeist</code> : removed <code>item_02</code> - no such	
<code>\@gls@loadsuper</code> : new	7	<code>makeindex</code> key	157
<code>\@gls@loadtree</code> : new	8	2.02 (2007-07-13)	
<code>\gls@defglossaryentry</code> : Changed de-		<code>\@printglossary</code> : suppressed warning	
fault value of <code>sort</code> to <code>\@glsdefaultsort</code>		globally rather than locally	178
.....	75	2.02 (2009-07-13)	
moved <code>sort</code> sanitization to <code>\newglossaryentry</code>		<code>\glossarysection</code> : changed <code>\@mkboth</code>	
.....	79	to <code>\glossarymark</code>	36
<code>\glstarget</code> : new	195	<code>\gls glossarymark</code> : New	36
<code>\oldacronym</code> : new	204	2.03 (2009-09-23)	
<code>nolist</code> : new	8	<code>\@GLS@</code> : Added check for <code>hyperfirst</code>	118
<code>nolong</code> : new	7	<code>\@GLSpl</code> : Added check for <code>hyperfirst</code> ...	120
<code>sort</code> : moved sanitization to <code>\newglossaryentry</code>		<code>\@Gls@</code> : Added check for <code>hyperfirst</code>	117
.....	59	<code>\@Glspl@</code> : Added check for <code>hyperfirst</code> ..	120
<code>nostyles</code> : new	8	<code>\@gls@</code> : Added check for <code>hyperfirst</code>	116
<code>nosuper</code> : new	8	<code>\@gls@link</code> : new	103
<code>notree</code> : new	8	<code>\@gls@link</code> : added <code>\leavevmode</code>	105
1.19 (2009-03-02)		Moved entry existence check to avoid	
<code>\gls clearpage</code> : new	39	duplicate code	105
<code>\glsdisp</code> : new	121	<code>\@glsdisp</code> : Added check for <code>hyperfirst</code> .	121
<code>\SetDescriptionAcronymStyle</code> :		<code>\@glspl@</code> : Added check for <code>hyperfirst</code> ..	119
changed <code>\acronymfont</code> to use		<code>\gls glossarymark</code> : Added check to see if	
<code>\textsmaller</code> instead of <code>\smaller</code> 231		it's already defined	36
<code>\SetDescriptionFootnoteAcronymStyle</code> :		<code>hyperfirst</code> : new	23
changed <code>\acronymfont</code> to use		2.04 (2009-11-10)	
<code>\textsmaller</code> instead of <code>\smaller</code> 227		<code>\@GLS@</code> : Changed test to check if glossary	
<code>\SetFootnoteAcronymStyle</code> : changed		type has been identified as a list of	
<code>\acronymfont</code> to use <code>\textsmaller</code>		acronyms	118
instead of <code>\smaller</code>	233	<code>\@GLSpl</code> : Changed test to check if glos-	
<code>\SetSmallAcronymStyle</code> : changed		sary type has been identified as a list	
<code>\acronymfont</code> to use <code>\textsmaller</code>		of acronyms	120
instead of <code>\smaller</code>	236	<code>\@Gls@</code> : Changed test to check if glossary	
2.01 (2009 May 30)		type has been identified as a list of	
<code>\@gls@link</code> : moved <code>\@do@wrglossary</code>		acronyms	117

\@Glspl@: Changed test to check if glossary type has been identified as a list of acronyms	120	\SetDUADisplayStyle: new	236
\@glossaryentryfield: new	81	\SetFootnoteAcronymDisplayStyle: new	231
\@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	221	\SetCustomDisplayStyle: new	238
\gls@defglossaryentry: added user1-6 keys	76	\SetCustomStyle: new	239
\glsadd: fixed bug that ignored counter	150	2.07 (2010-07-10)	
\Glsentryuseri: new	146	General: glsadd format key stored in \@glsnumberformat (was mistakenly stored in \@glo@format)	150
\Glsentryuserii: new	146	3.0 (2010-07-12)	
\Glsentryuseriii: new	146	\@makeglossary: Added check for savewrites	159
\Glsentryuseriiii: new	146	\gls@wrglossary: modified to take into account savewrites	168
\Glsentryuseriv: new	147	3.0 (2010/03/31)	
\Glsentryuseriv: 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
\Glsentryuseriv: new	147	modified to use new format	170
\Glsentryuseriv: new	147	\@glossarysec: replaced \@ifundefined with \@ifcsundef	5
\ns@newglossary: added check to determine if \gls@{type}@display and \gls@{type}@displayfirst have been defined.	56	\@do@seeglossary: Sanitize and escape cross-referencing information	173
\SetAcronymLists: new	15	\@gls@counterwithin: new	9
\SetDefaultAcronymDisplayStyle: new	223	\@gls@ifinlist: new	40
\SetDefaultAcronymStyle: new	224	\@gls@link: added \@gls@saveentrycounter	105
\SetDescriptionAcronymDisplayStyle: new	229	added \@gls@setsort	105
\SetDescriptionDUAAcronymDisplayStyle: new	227	\@gls@saveentrycounter: new	106
\SetDescriptionFootnoteAcronymDisplayStyle: new	225	\@gls@setupsort@def: new	10
		\@gls@setupsort@standard: new	10
		\@gls@setupsort@use: new	11
		\@gls@xdy@locationlist: new	43

<code>\@glslink:</code> replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	115	<code>\glsentryshort:</code> new	147
<code>\@glsnextpages:</code> new	192	<code>\Glsentryshortpl:</code> new	148
<code>\@print@glossary:</code> replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	178	<code>\glsentryshortpl:</code> new	147
<code>\@printglossary:</code> added <code>\currentglossary</code>	177	<code>\glsgetgrouptitle:</code> replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	198
added <code>\glsnextpages</code>	177	<code>\gls glossarymark:</code> replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	36
make toctitle default to title	177	<code>\gls hyperlink:</code> changed default from <code>\glsentryname</code> to <code>\glsentrytext</code>	150
<code>\@xdyattributelist:</code> new	39	<code>\gls hypernumber:</code> replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	201
General: added prefix to hyperlink	203	<code>\gls numberformat:</code> replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	35
etoolbox now loaded	4	<code>\gls reformat:</code> replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	35
replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	28, 31, 101, 189	<code>\gls reformat:</code> new	193
<code>\acrfootnote:</code> new	224	<code>\gls resetsubentrycounter:</code> new ...	192
<code>\ACRfull:</code> added starred version	206	<code>\gls seeitem:</code> hyperlink uses <code>\glsseeitemformat</code> instead of <code>\glsentryname</code>	175
<code>\Acrfull:</code> added starred version	206	<code>\gls seeitemformat:</code> new	175
<code>\acrfull:</code> added starred version	205	<code>\gls sortnumberfmt:</code> new	10
<code>\ACRfullpl:</code> added starred version ...	208	<code>\gls stepentry:</code> new	193
<code>\Acrfullpl:</code> added starred version ...	207	<code>\gls stepsubentry:</code> new	193
<code>\acrfullpl:</code> added starred version ...	207	<code>\gls subentrycounterlabel:</code> new ...	194
<code>\acrlinkfootnote:</code> new	224	<code>\gls subentryitem:</code> new	194
<code>\acrno linkfootnote:</code> new	225	<code>theglossary:</code> replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	194
<code>\savewrites:</code> new	25	<code>short:</code> new	62
see: added <code>\@glo@seeautonumberlist</code>	60	<code>shortplural:</code> new	62
<code>\seeautonumberlist:</code> new	7	<code>\ifglossaryexists:</code> replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	49
<code>\glossarysection:</code> replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	36	<code>\ifglsentryexists:</code> replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	49
<code>\glossarystyle:</code> replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	200	<code>\istfile:</code> deprecated	167
<code>\gls@codepage:</code> replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	24	<code>glossaryentry:</code> new	192
<code>\gls@def glossaryentry:</code> added <code>\@gls@defsort</code>	79	<code>glossarysubentry:</code> new	192
added short and long keys	76	<code>\newglossaryentry:</code> replaced <code>\DeclareRobustCommand</code> with <code>\newrobustcmd</code>	65
replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	76	<code>\newglossarystyle:</code> replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	200
<code>\gls@docclearpage:</code> replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	38	<code>\ns@newglossary:</code> added <code>\@gls@defsortcount</code>	56
<code>\glsadd:</code> added <code>\@gls@saveentrycounter</code>	151	replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	56
<code>\GlsAddXdyCounters:</code> new	40	<code>entrycounter:</code> new	9
<code>\glsentrycounterlabel:</code> new	194	<code>entrycounterwithin:</code> new	9
<code>\glsentryitem:</code> new	194	<code>\oldacronym:</code> replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	204
<code>\Glsentrylong:</code> new	148	<code>compatible-2.07:</code> compatible-2.07 op- tion added	26
<code>\glsentrylong:</code> new	148	<code>long:</code> new	62
<code>\Glsentrylongpl:</code> new	148		
<code>\glsentrylongpl:</code> new	148		
<code>\Glsentryshort:</code> new	147		

longplural: new	62	\Acrfull: made robust	206
nonumberlist: now boolean	61	\acrfull: made robust	205
sort: new	9	\acrfullformat: removed \acronymfont as it should already be set in the sec- ond argument.	206
counter: replaced \@ifundefined with \ifcsundef	60	\ACRfullpl: made robust	208
\printglossary: replaced \@ifundefined with \ifcsundef	175	\Acrfullpl: made robust	207
\SetDescriptionFootnoteAcronymDisplayStyle expanded options link options	225	\acrfullpl: made robust	207
\setentrycounter: added optional ar- gument	199	\ACRlong: made robust	139
\showacronymlists: new	245	\Acrlong: made robust	138
\showglocounter: new	242	\acrlong: made robust	138
\showgloDESC: new	243	\ACRlongpl: made robust	141
\showgloDESCplural: new	243	\Acrlongpl: made robust	140
\showglofirst: new	241	\acrlongpl: made robust	140
\showglofirstpl: new	241	\ACRshort: made robust	135
\showgloflag: new	244	\Acrshort: made robust	135
\showgloindex: new	244	\acrshort: made robust	134
\showglolevel: new	241	\ACRshortpl: made robust	137
\showgloNAME: new	243	\Acrshortpl: made robust	136
\showgloparent: new	240	\acrshortpl: made robust	136
\showgloplural: new	241	\Gls: made robust	117
\showglosort: new	243	\glsadd: made robust	150
\showglossaries: new	245	\glsaddall: made robust	151
\showglossarycounter: new	246	\GLSdesc: made robust	126
\showglossaryentries: new	246	\Glsdesc: made robust	126
\showglossaryin: new	245	\glsdesc: made robust	126
\showglossaryout: new	245	\GLSdescplural: made robust	127
\showglossarytitle: new	245	\Glsdescplural: made robust	127
\showglosymbol: new	243	\glsdescplural: made robust	127
\showglosymbolplural: new	244	\glsfirst: made robust	123
\showglotext: new	241	\GLSfirstplural: made robust	125
\showglotype: new	241	\Glsfirstplural: made robust	125
\showglouserI: new	242	\glsfirstplural: made robust	124
\showglouserII: new	242	\glslink: made robust	103
\showglouserIII: new	242	\GLSname: made robust	126
\showglouserIV: new	242	\Glsname: made robust	125
\showglouserV: new	242	\glsname: made robust	125
\showglouserVI: new	243	\GLSpl: made robust	120
subentrycounter: new	9	\Glspl: made robust	119
\writeist: added xindy-only macro defi- nitions to glossary open tag	155	\glspl: made robust	118
modified to support new format	153	\GLSplural: made robust	124
3.01 (2011-04-12)		\GLSsymbol: made robust	128
\@glswritefiles: added check for empty glossaries	167	\Glsymbol: made robust	128
General: made robust	118	\glssymbol: made robust	128
\ACRfull: made robust	206	\GLSsymbolplural: made robust	129
		\Glsymbolplural: made robust	129
		\glssymbolplural: made robust	128
		\Glstext: made robust	122
		\glstext: made robust	122

<code>\GLSuseri</code> : made robust	130	<code>\glsentryfullpl</code> : fixed bug (re-	
<code>\Glsuseri</code> : made robust	130	placed <code>\glsentryshort</code> with	
<code>\glsuseri</code> : made robust	129	<code>\glsentryshortpl</code>)	148
<code>\GLSuserii</code> : made robust	131	<code>\glsentrynumberlist</code> : new	149
<code>\Glsuserii</code> : made robust	130	<code>\glsmoveentry</code> : new	80
<code>\glsuserii</code> : made robust	130	<code>\glsresetsubentrycounter</code> : new ...	193
<code>\GLSuseriii</code> : made robust	131	<code>\ifglshaschildren</code> : new	51
<code>\Glsuseriii</code> : made robust	131	<code>\ifglshasparent</code> : new	51
<code>\glsuseriii</code> : made robust	131	<code>\makeglossaries</code> : added list parser ..	162
<code>\GLSuseriv</code> : made robust	132	<code>indexonlyfirst</code> : new	23
<code>\Glsuseriv</code> : made robust	132	<code>\renewglossarystyle</code> : new	201
<code>\glsuseriv</code> : made robust	132	<code>\showglossaryentries</code> : fixed misspelt	
<code>\GLSuserv</code> : made robust	133	command	246
<code>\Glsuserv</code> : made robust	133	<code>\SmallNewAcronymDef</code> : fixed broken	
<code>\glsuserv</code> : made robust	132	short and long plural	234
<code>\GLSuservi</code> : made robust	134	3.03 (2012/09/21)	
<code>\Glsuservi</code> : made robust	134	<code>\@gls@sanitizesort</code> : new	17
<code>\glsuservi</code> : made robust	133	<code>\@gls@setupsort@standard</code> : used	
3.02 (2012-05-19)		<code>\@gls@sanitizesort</code>	10
<code>\glsnumlistlastsep</code> : new	150	<code>\@printglossary</code> : allow title to override	
<code>\glsnumlistsep</code> : new	150	default toctitle	176
3.02 (2012-05-21)		General: allow title to set toctitle	189
<code>\@do@wrglossary</code> : changed <code>\@gls@locref</code>		<code>\glsinlinedescformat</code> : new	258
to <code>\theglentrycounter</code>	172	<code>\glsinlineemptydescformat</code> : new ..	258
<code>\@do@wrglossary</code> : changed <code>\@do@wr@glossary</code>		<code>\glsinlinenameformat</code> : new	258
to test for <code>indexonlyfirst</code> option; put		<code>\glsinlinepostchild</code> : new	258
old <code>\@do@wr@glossary</code> code into		<code>\glsinlinesubdescformat</code> : new	258
<code>\@do@wrglossary</code>	169	<code>\glsinlinesubnameformat</code> : new	258
<code>\@gls@missingnumberlist</code> : new	63	<code>\glspostinline</code> : replaced “.” with	
<code>\@gls@writefiles</code> : added check		<code>\glspostdescription</code>	258
for existence of token in case		<code>altlongragged4col</code> : added check for	
<code>\makeglossaries</code> has been		<code>glsnogroupskip</code>	276
omitted	167	<code>altsuperragged4col</code> : added check for	
<code>\@printglossary</code> : add a way to fetch		<code>glsnogroupskip</code>	292
current entry label	177	<code>alttree</code> : added check for <code>glsnogroupskip</code>	
<code>savenumberlist</code> : new	7	300
<code>ucmark</code> : new	8	index: added check for <code>glsnogroupskip</code>	294
<code>\gls@defglossaryentry</code> : added num-		<code>nogroupskip</code> : new	8
berlist element	79	long: added check for <code>glsnogroupskip</code> .	262
<code>\gls@save@numberlist</code> : new	175	<code>long3col</code> : added check for <code>glsnogroup-</code>	
<code>\gls@wrglossary</code> : added check for glos-		<code>skip</code>	264
sary file defined	169	<code>long4col</code> : added check for <code>glsnogroup-</code>	
<code>\glsdisplaynumberlist</code> : new	149	<code>skip</code>	265
<code>\glsentrycounter</code> : set default value ..	106	<code>longragged</code> : added check for <code>glsnogroup-</code>	
<code>\Glsentryfull</code> : fixed bug (re-		<code>skip</code>	273
placed <code>\glsentryshortpl</code> with		<code>longragged3col</code> : added check for	
<code>\glsentryshort</code>)	148	<code>glsnogroupskip</code>	274
		<code>nopostdot</code> : new	8
		<code>tree</code> : added check for <code>glsnogroupskip</code> .	296

treenoname: added check for glsnogroup-skip	297	mcoltree: replaced ‘2’ with \glsmcols	278
super: added check for glsnogroupskip	282	mcoltreenoname: replaced ‘2’ with \glsmcols	279
super3col: added check for glsnogroup-skip	283	\gls@protected@pagefmts: added Roman to list	169
super4col: added check for glsnogroup-skip	285	\gls@Romanpage: new	170
superragged: added check for glsnogroupskip	289	\glsgetgrouplabel: fixed bug (typo in \equal)	199
superragged3col: added check for glsnogroupskip	290	\nopostdesc: made robust	32
3.04 (2012-11-11)		3.05 (2013/04/21)	
altlist: replaced \newline with paragraph break	260	\@gls@nohyperlist: new	15
3.04 (2012-11-18)		\GlsDeclareNoHyperList: new	15
\@do@wrglossary: changed \theglsentrycounter back to \@glslocref	172	nohypertypes: new	15
\@do@wrglossary: modified to compensate for possible incorrect page number	170	3.06 (2013/06/17)	
\@gls@escbsdq: unsanitize \gls@numberpage, \gls@alphpage, \gls@Alphpage and \gls@romanpage	108	\@xdy@main@language: Changed back to using \languagenam	24
\@print@glossary: Moved aux write to end of document to prevent unwanted whatsit occurring here.	178	\findrootlanguage: Obsoleted	46
General: Added check for doc package ...	4	3.07 (2013-07-05)	
added datatool-base as a required package	4	\@gls@link: fixed bug that failed to find entry in list	105
added local key	102	\glossary preamble: modified to work with \setglossary preamble	35
\gls@Alphpage: new	169	\gls@docclearpage: added check for openright	38
\gls@alphpage: new	169	\glspostdescription: Added spacefactor code	8
\gls@disablepagerefexpansion: new	169	\GlsSetXdyCodePage: Added check for fontspec	47
\gls@numberpage: new	170	\SetDescriptionAcronymDisplayStyle: now using \glsdoparenifnotempty	229
\gls@protected@pagefmts: new	169	\setglossary preamble: new	36
\gls@romanpage: new	170	3.08a (2013-08-30)	
\glsdefmain: added check for doc package	12	list: updated list style to use \glossentry and \subglossentry	259
\glsorg@endtheglossary: new	5	listdotted: updated listdotted style to use \glossentry and \subglossentry	261
\glsorg@theglossary: new	5	altlist: updated altlist style to use \glossentry and \subglossentry	260
\PrintChanges: new	5	inline: updated inline style to use \glossentry and \subglossentry	257
3.05 (2013-04-21)		3.08a (2013-09-28)	
\@do@wrglossary: add Roman case. Fixed bugs in the else statements ..	171	\@glo@storeentry: no longer need to check for special characters in any of the fields other than sort	81
\@gls@link: added check for “nohypertypes”	105	updated for \glossentry	82
mcolalttree: replaced ‘2’ with \glsmcols	280	\@glossaryentryfield: switched to \glossentry	81
mcolindex: replaced ‘2’ with \glsmcols	278		

\@glossarysubentryfield: switched to	superragged: updated to use
\subglossentry 81	\glossentry and \subglossentry 288
General: added nogroupskip key to	3.09a (2013-10-09)
\printglossary 189	\@gls@assign@symbolplural@field:
removed definition of \@glossaryentryfield	new 17
..... 342	\@gls@default@value: new 59
removed definition of \@glossarysubentryfield	\Glsentrydesc: made robust 144
..... 342	\Glsentrydescplural: made robust .. 144
\compatibleglossentry: new 195	\Glsentryfirst: made robust 145
\compatiblesubglossentry: new ... 196	\Glsentryfirstplural: made robust . 145
\glossaryentryfield: deprecated ... 197	\Glsentryfull: made robust 148
\Glossentrydesc: new 196	\Glsentryfullpl: made robust 148
\glossentrydesc: new 195	\Glsentrylong: made robust 148
\Glossentryname: new 195	\Glsentrylongpl: made robust 148
\glossentryname: new 195	\Glsentryname: made robust 143
\Glossentrysymbol: new 196	\Glsentryplural: made robust 144
\glossentrysymbol: new 196	\Glsentryshort: made robust 147
\gls@assign@desc@field: new 17	\Glsentryshortpl: made robust 148
\gls@assign@descplural@field: new 17	\Glsentrysymbol: made robust 145
\gls@assign@field: new 65	\Glsentrysymbolplural: made robust 145
\gls@ifnotmeasuring: new 83	\Glsentrytext: made robust 144
\glsaddallunused: new 151	\Glsentryuseri: made robust 146
\glsexpandfields: new 65	\Glsentryuserii: made robust 146
\glsnoexpandfields: new 65	\Glsentryuseriii: made robust 146
\glssee: made robust 174	\Glsentryuseriv: made robust 147
\glsseeformat: made robust 174	\Glsentryuserv: made robust 147
\glsseeitem: made robust 175	\Glsentryuservi: made robust 147
\glsseelist: made robust 174	\glstextup: new 205
\ifglshdescsuppressed: new 52	\ifglshassymbol: changed test to check
\ifglshasdesc: new 52	for \@gls@default@symbol 52
\ifglshassymbol: new 52	3.10a (2013-09-28)
altlongragged4col: updated to use	\gls@assign@type@field: new 17
\glossentry and \subglossentry 276	3.10a (2013-10-13)
alttree: updated to use \glossentry	\@gls@keymap: new 67
and \subglossentry 298	\@gls@provide@newglossary: new ... 54
index: added paragraph break at end of	\@gls@writedef: new 66
environment 294	\@gls@defaultplural: Obsolete 62
updated to use \glossentry and	\@glsnodesc: new 62
\subglossentry 294	\@print@glossary: Added providecom-
long: updated to use \glossentry and	mand code to aux file 178, 179
\subglossentry 262	\gls@defglossaryentry: Changed to
longragged: updated to use \glossentry	using \@gls@default@value 75
and \subglossentry 273	new 75
longragged3col: updated to use	\gls@writedefhook: new 74
\glossentry and \subglossentry 274	\makeglossaries: Added providecom-
tree: updated to use \glossentry and	mand code to aux file 161
\subglossentry 295	\new@glossaryentry: new 66
\setglossarystyle: new 199	\ns@newglossary: added \@gls@provide@newglossary
\setglossentrycompatibility: new 196 56

3.11a (2013-10-15)

<code>\@ACRlong:</code> added <code>\glslabel,</code> <code>\glsifplural,</code> <code>\glscapscase,</code> <code>\glsinsert</code> and <code>\glscustomtext</code>	342	change to using <code>\glsentryfmt</code> style commands	116
<code>\@ACRshort:</code> added <code>\glslabel,</code> <code>\glsifplural,</code> <code>\glscapscase,</code> <code>\glsinsert</code> and <code>\glscustomtext</code>	340	<code>\@gls@noexpand@fields:</code> Fixed bug expand replaced with noexpand	63
<code>\@Acrlong:</code> added <code>\glslabel,</code> <code>\glsifplural,</code> <code>\glscapscase,</code> <code>\glsinsert</code> and <code>\glscustomtext</code>	341	<code>\@glsdisp:</code> add <code>\glslabel,</code> <code>\glsifplural,</code> <code>\glscapscase,</code> <code>\glscustomtext</code> and <code>\glsinsert</code>	121
<code>\@Acrshort:</code> added <code>\glslabel,</code> <code>\glsifplural,</code> <code>\glscapscase,</code> <code>\glsinsert</code> and <code>\glscustomtext</code>	340	change to using <code>\glsentryfmt</code> style commands	121
<code>\@GLS@:</code> add <code>\glslabel,</code> <code>\glsifplural,</code> <code>\glscapscase,</code> <code>\glscustomtext</code> and <code>\glsinsert</code>	118	<code>\@glspl@:</code> add <code>\glslabel,</code> <code>\glsifplural,</code> <code>\glscapscase,</code> <code>\glscustomtext</code> and <code>\glsinsert</code>	119
change to using <code>\glsentryfmt</code> style commands	118	change to using <code>\glsentryfmt</code> style commands	119
removed <code>\MakeUppercase</code> (now moved to <code>\glsentryfmt</code>)	118	General: added <code>\glslabel,</code> <code>\glsifplural,</code> <code>\glscapscase,</code> <code>\glsinsert</code> and <code>\glscustomtext</code>	134–141
<code>\@GLSpl:</code> add <code>\glslabel,</code> <code>\glsifplural,</code> <code>\glscapscase,</code> <code>\glscustomtext</code> and <code>\glsinsert</code>	120	changed to just use <code>\Glsentrydescplural</code>	127
change to using <code>\glsentryfmt</code> style commands	120	changed to just use <code>\glsentrydescplural</code>	127
removed <code>\MakeUppercase</code> as now dealt with in <code>\glsentryfmt</code>	120	changed to just use <code>\Glsentrydesc</code> .	126
<code>\@Gls@:</code> add <code>\glsifplural,</code> <code>\glscapscase,</code> <code>\glscustomtext</code> and <code>\glsinsert</code>	117	126, 127
change to using <code>\glsentryfmt</code> style commands	117	changed to just use <code>\Glsentryfirstplural</code>	125
removed <code>\makefirstuc</code> (now dealt with in <code>\glsentryfmt</code>)	117	changed to just use <code>\glsentryfirstplural</code>	124, 125
<code>\@Glspl@:</code> add <code>\glsifplural,</code> <code>\glscapscase,</code> <code>\glscustomtext</code> and <code>\glsinsert</code>	119	changed to just use <code>\Glsentryfirst</code>	123
change to using <code>\glsentryfmt</code> style commands	120	changed to just use <code>\glsentryfirst</code>	123
removed <code>\makefirstuc</code> (now dealt with in <code>\glsentryfmt</code>)	120	changed to just use <code>\Glsentryname</code> .	126
<code>\@acrlong:</code> added <code>\glslabel,</code> <code>\glsifplural,</code> <code>\glscapscase,</code> <code>\glsinsert</code> and <code>\glscustomtext</code>	341	125, 126
<code>\@acrshort:</code> added <code>\glslabel,</code> <code>\glsifplural,</code> <code>\glscapscase,</code> <code>\glsinsert</code> and <code>\glscustomtext</code>	340	changed to just use <code>\Glsentryplural</code>	124
<code>\@gls@:</code> add <code>\glslabel,</code> <code>\glsifplural,</code> <code>\glscapscase,</code> <code>\glscustomtext</code> and <code>\glsinsert</code>	116	changed to just use <code>\glsentryplural</code>	124
		changed to just use <code>\Glsentrysymbolplural</code>	129
		changed to just use <code>\glsentrysymbolplural</code>	129
		changed to just use <code>\Glsentrysymbol</code>	128
		changed to just use <code>\glsentrysymbol</code>	128
		Changed to just use <code>\Glsentrytext</code> .	123
		changed to just use <code>\glsentrytext</code> .	122
		changed to just use <code>\Glsentryuseriii</code>	131
		changed to just use <code>\glsentryuseriii</code>	131, 132
		changed to just use <code>\Glsentryuserii</code>	130

changed to just use <code>\glsentryuserii</code>	
.....	130, 131
changed to just use <code>\Glsentryuseriv</code>	132
changed to just use <code>\glsentryuseriv</code>	132
changed to just use <code>\Glsentryuseri</code>	130
changed to just use <code>\glsentryuseri</code>	
.....	129, 130
changed to just use <code>\Glsentryuservi</code>	134
changed to just use <code>\glsentryuservi</code>	
.....	133, 134
changed to just use <code>\Glsentryuserv</code>	133
changed to just use <code>\glsentryuserv</code>	133
Now requires textcase	4
acronymlists: replaced <code>\@addtoacronymlists</code>	
with <code>\DeclareAcronymList</code>	15
<code>\defglsgdisplay</code> : obsoleted	100
<code>\defglsgdisplayfirst</code> : obsoleted	101
<code>\defglsgentryfmt</code> : new	54
<code>\forglsgentries</code> : replaced <code>\ifx</code> with	
<code>\ifdefempty</code>	48
<code>\gls@assign@desc</code> : new	74
<code>\gls@defglossaryentry</code> : Fixed default	
counter if none supplied	78
<code>\gls@doentryfmt</code> : new	54
<code>\glsdisplay</code> : obsoleted	100
<code>\glsdisplayfirst</code> : obsoleted	100
<code>\glsgenentryfmt</code> : new	95
<code>\glsgetgrouptitle</code> : Added check in	
case non-Latin alphabet in use	198
<code>\gls glossarymark</code> : replaced <code>\MakeUppercase</code>	
with <code>\mfirstucMakeUppercase</code>	36
<code>\glsnavigation</code> : switched to using	
<code>\@gls@getgrouptitle</code>	256
<code>\ifglshasdesc</code> : replaced <code>\ifdefempty</code>	
with <code>\ifcsemt</code>	52
<code>\ifglshaslong</code> : new	52
<code>\ifglshasshort</code> : new	52
<code>\ifglshassymbol</code> : replaced <code>\ifdefempty</code>	
with <code>\ifcsemt</code>	52
<code>\ifglsgused</code> : replaced <code>\ifthenelse</code> with	
<code>\ifbool</code>	50
<code>\longnewglossaryentry</code> : new	74
<code>\ns@newglossary</code> : replaced <code>\glsdisplay</code>	
and <code>\glsdisplayfirst</code> with	
<code>\glsentryfmt</code>	56
compatible-3.07: cnew	26
<code>\SetCustomDisplayStyle</code> : updated to	
use <code>\defglsgentryfmt</code>	238
<code>\SetDefaultAcronymDisplayStyle</code> :	
changed to use <code>\defglsgentryfmt</code>	223
<code>\SetDescriptionAcronymDisplayStyle</code> :	
updated to use <code>\defglsgentryfmt</code>	229
<code>\SetDescriptionDUAAcronymDisplayStyle</code> :	
updated to use <code>\defglsgentryfmt</code>	227
<code>\SetDescriptionFootnoteAcronymDisplayStyle</code> :	
updated to use <code>\defglsgentryfmt</code>	225
<code>\SetDUADisplayStyle</code> : updated to use	
<code>\defglsgentryfmt</code>	236
<code>\SetFootnoteAcronymDisplayStyle</code> :	
updated to use <code>\defglsgentryfmt</code>	231
<code>\SetSmallAcronymDisplayStyle</code> : up-	
dated to use <code>\defglsgentryfmt</code>	234
<code>\setupglossaries</code> : new	27
<code>\showglo long</code> : new	244
<code>\showglo short</code> : new	244
numbers: new	26
symbols: new	26
3.12a (2013-10-16)	
<code>\gls@defglossaryentry</code> : added	
<code>\glslabel</code>	75
<code>\glsaddkey</code> : new	69
3.13a (2013-11-05)	
<code>\@gls@assign@symbol@field</code> : changed	
to use <code>\glssetnoexpandfield</code>	17
<code>\@gls@assign@symbolplural@field</code> :	
changed to use <code>\glssetnoexpandfield</code>	17
<code>\@gls@link</code> : removed <code>\relax</code>	106
<code>\@gls@notranslatorhook</code> : new	21
<code>\@gls@setupsort@standard</code> : moved	
<code>\@gls@santizesort</code> to <code>\glsprestandardsort</code>	10
ucmark: added check for memoir	9
see: added <code>\gls@checkseeallowed</code>	60
<code>\glossarysection</code> : changed <code>\glossarymark</code>	
to <code>\gls glossarymark</code>	36
<code>\glossarystyle</code> : fixed bug caused by us-	
ing <code>\ifdef</code> instead of <code>\ifcsdef</code>	200
<code>\gls@assign@desc@field</code> : changed to	
use <code>\glssetnoexpandfield</code>	17
<code>\gls@assign@descplural@field</code> :	
changed to use <code>\glssetnoexpandfield</code>	17
<code>\gls@assign@name@field</code> : changed to	
use <code>\glssetnoexpandfield</code>	17
<code>\gls@assign@type@field</code> : changed to	
use <code>\glssetexpandfield</code>	17

<code>\gls@checkseeallowed</code> : new	60	<code>superheader</code> : switched to <code>\tabularnewline</code>	282
<code>\glsaddallunused</code> : set default to <code>\@glo@types</code>	151	<code>superheaderborder</code> : switched to <code>\tabularnewline</code>	283
<code>\Glsentryfull</code> : changed to use <code>\acrfullformat</code>	148	3.14a (2013-11-12)	
<code>\glsentryfull</code> : changed to use <code>\acrfullformat</code>	148	<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>	148	General: new	247
<code>\glsentryfullpl</code> : changed to use <code>\acrfullformat</code>	148	acronyms: new	13
<code>\gls glossarymark</code> : renamed <code>\glossarymark</code> to <code>\gls glossarymark</code> to avoid con- flict with memoir	36	<code>\gls@def glossaryentry</code> : added check for existence of default glossary	76
<code>\glsprestandardsort</code> : new	9	set the default for firstplural to be the value of plural	78
<code>\glssetexpandfield</code> : new	16	<code>xindygloss</code> : new	25
<code>\glssetnoexpandfield</code> : new	16	<code>\longprovideglossaryentry</code> : new ...	75
<code>altsuper4colheader</code> : switched to <code>\tabularnewline</code>	286	compatible-2.07: added check for 2.07 before setting 3.07 compatibility	26
<code>altsuper4colheaderborder</code> : switched to <code>\tabularnewline</code>	287	<code>notranslate</code> : new	21
<code>long</code> : switched to <code>\tabularnewline</code> ..	262	<code>\provideglossaryentry</code> : new	65
<code>long3col</code> : switched to <code>\tabularnewline</code>	264	4.0 (2013-11-14)	
<code>long3colheader</code> : switched to <code>\tabularnewline</code>	264	<code>\gls@def glossaryentry</code> : added check for first key	78
<code>long3colheaderborder</code> : switched to <code>\tabularnewline</code>	265	<code>super</code> : fixed typo in <code>\subglossentry</code> (<code>\glossentrydesc</code>)	282
<code>long4col</code> : switched to <code>\tabularnewline</code>	265	4.01 (2013-11-16)	
<code>long4colheader</code> : switched to <code>\tabularnewline</code>	266	General: fixed non-value options so that they can be passed to document class .	6
<code>longheader</code> : switched to <code>\tabularnewline</code>	263	<code>\CustomAcronymFields</code> : inserted miss- ing comma	239
<code>longheaderborder</code> : switched to <code>\tabularnewline</code>	263	4.02 (2013-12-05)	
<code>\SetFootnoteAcronymDisplayStyle</code> : fixed missing argument bug	231	<code>\@acrfull</code> : now using <code>\acrfullfmt</code> ..	205
<code>super</code> : switched to <code>\tabularnewline</code> ..	282	<code>\@gls@indexdef</code> : new	27
<code>super3col</code> : switched to <code>\tabularnewline</code>	283	<code>\@gls@numbersdef</code> : new	26
<code>super3colheader</code> : switched to <code>\tabularnewline</code>	284	<code>\@gls@symbolsdef</code> : new	26
<code>super4col</code> : switched to <code>\tabularnewline</code>	285	General: Removed <code>\acronymfont</code> .	138–141
<code>super4colheader</code> : switched to <code>\tabularnewline</code>	285	<code>\ACRfullfmt</code> : new	207
<code>super4colheaderborder</code> : switched to <code>\tabularnewline</code>	286	<code>\Acrfullfmt</code> : new	206
		<code>\acrfullfmt</code> : new	206
		<code>\ACRfullplfmt</code> : new	208
		<code>\Acrfullplfmt</code> : new	207
		<code>\acrfullplfmt</code> : new	207
		<code>\acronymentry</code> : new	210
		<code>sanitize</code> : fixed bug that caused an error here	20
		<code>sc-short-long</code> : new	214
		<code>sc-short-long-desc</code> : new	216
		<code>\Genacrfullformat</code> : new	99

<code>\genacrfullformat:new</code>	99	<code>\SetSmallAcronymDisplayStyle:</code>	
<code>\GenericAcronymFields:new</code>	210	Moved check for empty custom text to	
<code>\Genplacrfullformat:new</code>	100	prevent unwanted parenthetical ma-	
<code>\genplacrfullformat:new</code>	100	terial	234
<code>\Glsentryfull: bug fix: added missing</code>		<code>dua:new</code>	216
<code>\acronymfont</code>	148	<code>dua-desc:new</code>	218
<code>\glsentryfull: bug fix: added missing</code>		<code>numberedsection: added nameref op-</code>	
<code>\acronymfont</code>	148	tion	6
<code>\Glsentryfullpl: bug fix: added miss-</code>		4.02 (2013-13-05)	
ing <code>\acronymfont</code>	148	<code>\makeglossaries: made preamble only</code>	162
<code>\glsentryfullpl: bug fix: added miss-</code>		4.03 (2014-01-17)	
ing <code>\acronymfont</code>	148	General: changed default to <code>\@empty</code> in-	
<code>\glsgenacfmt:new</code>	97	stead of <code>\relax</code>	26
<code>\GlsUseAcrEntryDispStyle:new</code> ...	211	4.03 (2014-01-20)	
<code>\GlsUseAcrStyleDefs:new</code>	211	<code>\@do@wrglossary: added \glsdetoklabel</code>	
<code>short-long:new</code>	213	171
<code>short-long-desc:new</code>	215	<code>\@ACRlong: removed \glslabel (defined</code>	
<code>xindynoglsnumbers:new</code>	25	in <code>\@gls@link</code>)	342
<code>sm-short-long:new</code>	214	<code>\@ACRshort: removed \glslabel (de-</code>	
<code>sm-short-long-desc:new</code>	216	fined in <code>\@gls@link</code>)	340
<code>index:new</code>	27	<code>\@Acrlong: removed \glslabel (defined</code>	
<code>\newacronymstyle:new</code>	211	in <code>\@gls@link</code>)	341
<code>long-sc-short:new</code>	213	<code>\@Acrshort: removed \glslabel (de-</code>	
<code>long-sc-short-desc:new</code>	215	fined in <code>\@gls@link</code>)	340
<code>long-short:new</code>	211	<code>\@GLS@: removed \glslabel (defined in</code>	
<code>long-short-desc:new</code>	214	<code>\@gls@link</code>)	118
<code>long-sm-short:new</code>	214	<code>\@GLSpl: removed \glslabel (defined in</code>	
<code>long-sm-short-desc:new</code>	215	<code>\@gls@link</code>)	120
<code>long-sp-short-desc:new</code>	214	<code>\@Gls@: removed \glslabel (defined in</code>	
<code>footnote:new</code>	218	<code>\@gls@link</code>)	117
<code>footnote-desc:new</code>	220	<code>\@Gls@entry@field:new</code>	142
<code>footnote-sc:new</code>	220	<code>\@Glspl@: removed \glslabel (defined</code>	
<code>footnote-sc-desc:new</code>	221	in <code>\@gls@link</code>)	119
<code>footnote-sm:new</code>	220	<code>\@acrlong: removed \glslabel (defined</code>	
<code>footnote-sm-desc:new</code>	221	in <code>\@gls@link</code>)	341
<code>\setacronymstyle:new</code>	210	<code>\@acrshort: removed \glslabel (de-</code>	
<code>\SetDescriptionAcronymDisplayStyle:</code>		fined in <code>\@gls@link</code>)	340
Moved check for empty custom text to		<code>\@gls@: removed \glslabel (defined in</code>	
prevent unwanted parenthetical ma-		<code>\@gls@link</code>)	116
terial	229	<code>\@gls@access@display:new</code>	328
<code>\SetDescriptionFootnoteAcronymDisplayStyle:</code>		<code>\@gls@entry@field:new</code>	142
Moved check for empty custom text to		<code>\@gls@fetchfield:new</code>	67
prevent unwanted parenthetical ma-		<code>\@gls@field@link:new</code>	122
terial	225	<code>\@gls@link: added \glsdetoklabel</code>	
<code>\SetFootnoteAcronymDisplayStyle:</code>		moved <code>\@gls@link@opts</code> and	
Moved check for empty custom text to		<code>\@gls@link@label</code> to <code>\@gls@link</code>	105
prevent unwanted parenthetical ma-		<code>\@gls@writedef: added \glsdetoklabel</code>	
terial	231	66
<code>\SetGenericNewAcronym:new</code>	209		

<code>\@glsdisp: removed \glslabel</code> (defined in <code>\@gls@link</code>)	121	<code>\glsentrylongpluralaccess: switched to using \@gls@entry@field</code>	328
<code>\@glspl@: removed \glslabel</code> (defined in <code>\@gls@link</code>)	119	<code>\glsentrypluralaccess: switched to using \@gls@entry@field</code>	327
<code>\@printglossary: added \glsdetoklabel</code>	177	<code>\glsentryshortaccess: switched to using \@gls@entry@field</code>	327
General: removed <code>\glslabel</code> (defined in <code>\@gls@link</code>)	134	<code>\glsentryshortpluralaccess: switched to using \@gls@entry@field</code>	327
<code>sc-short-long-desc: redefined to use accessibility information</code>	346	<code>\glsentrysymbolaccess: switched to using \@gls@entry@field</code>	327
<code>\compatibleglossentry: added \glsdetoklabel</code>	322	<code>\glsentrysymbolpluralaccess: switched to using \@gls@entry@field</code>	327
<code>\compatiblesubglossentry: added \glsdetoklabel</code>	323	<code>\glsentrytextaccess: switched to using \@gls@entry@field</code>	326
<code>\Genacrfullformat: redefined to use accessibility information</code>	339	<code>\glsngenacfmt: redefined to use accessibility information</code>	337
<code>\genacrfullformat: redefined to use accessibility information</code>	339	<code>\glsgenentryfmt: redefined to use accessibility information</code>	334
<code>\Genplacrfullformat: redefined to use accessibility information</code>	339	<code>\gls hyperlink: added \glsdetoklabel</code>	150
<code>\genplacrfullformat: redefined to use accessibility information</code>	339	<code>\glslocalreset: added \glsdetoklabel</code>	83
<code>\glossentryname: added \glsdetoklabel</code>	195	<code>\glslocalunset: added \glsdetoklabel</code>	84
<code>\gls@defglossaryentry: added \glsdetoklabel</code>	75	<code>\glsmoveentry: added \glsdetoklabel</code>	80
replaced <code>#1</code> with <code>\@glo@label</code>	76	replaced <code>\ifthenelse</code> with <code>\ifdefequal</code>	81
replaced <code>\ifthenelse</code> with <code>\ifdefequal</code>	77	<code>\glsrefentry: added \glsdetoklabel</code>	193
<code>\glsadd: added \glsdetoklabel</code>	150	<code>\glsreset: added \glsdetoklabel</code> ...	83
<code>\glsaddkey: switched to using \@gls@field@link</code>	70	<code>\glsseelist: added \expandafter commands</code>	174
<code>\glsdetoklabel: new</code>	49	<code>\glsstepentry: added \glsdetoklabel</code>	193
<code>\glsdisplaynumberlist: added \glsdetoklabel</code>	149	<code>\glsstepsubentry: added \glsdetoklabel</code>	193
<code>\glsdoifexistsorwarn: new</code>	50	<code>\glsunset: added \glsdetoklabel</code> ...	83
<code>\glsentryaccess: switched to using \@gls@entry@field</code>	326	<code>short-long: commented spurious EOL</code>	213
<code>\glsentrydescaccess: switched to using \@gls@entry@field</code>	327	redefined to use accessibility information	344
<code>\glsentrydescpluralaccess: switched to using \@gls@entry@field</code>	327	<code>short-long-desc: redefined to use accessibility information</code>	346
<code>\glsentryfirstaccess: switched to using \@gls@entry@field</code>	327	<code>\ifglsdescsuppressed: added \glsdetoklabel</code>	52
<code>\glsentryfirstplural: added \glsdetoklabel</code>	145	fixed typo	52
<code>\glsentrylongaccess: switched to using \@gls@entry@field</code>	328	<code>\ifglsentryexists: added \glsdetoklabel</code>	49

<code>\ifglshaschildren:added\glsdetoklabel</code>	51	<code>\showglofirst:added\glsdetoklabel</code>	241
<code>\ifglshasdesc:added\glsdetoklabel</code>	52	<code>\showglofirstaccess:added\glsdetoklabel</code>	358
<code>\ifglshasfield:new</code>	53	<code>\showglofirstpl:added\glsdetoklabel</code>	241
<code>\ifglshaslong:added\glsdetoklabel</code>	52	<code>\showglofirstpluralaccess: added</code> <code>\glsdetoklabel</code>	358
<code>\ifglshasparent:added\glsdetoklabel</code>	51	<code>\showgloflag:added\glsdetoklabel</code>	244
<code>\ifglshasshort:added\glsdetoklabel</code>	52	<code>\showgloindex:added\glsdetoklabel</code>	244
<code>\ifglshassymbol:added\glsdetoklabel</code>	52	<code>\showglolevel:added\glsdetoklabel</code>	241
replaced <code>\ifcsemtyp</code> with <code>\ifdefempty</code> and replaced <code>\ifx</code> with <code>\ifdefequal</code>	52	<code>\showglolong:added\glsdetoklabel</code>	244
<code>\ifglsused:added\glsdetoklabel</code> ..	50	<code>\showglolongaccess:added\glsdetoklabel</code>	359
<code>sm-short-long-desc:redefined to use</code> <code>accessibility information</code>	346	<code>\showglolongpluralaccess: added</code> <code>\glsdetoklabel</code>	359
<code>long-sc-short-desc:redefined to use</code> <code>accessibility information</code>	345	<code>\showglongname:added\glsdetoklabel</code>	243
<code>long-short:redefined to use accessibil-</code> <code>ity information</code>	343	<code>\showglongnameaccess:added\glsdetoklabel</code>	358
<code>long-short-desc:redefined to use ac-</code> <code>cessibility information</code>	345	<code>\showgloparent:added\glsdetoklabel</code>	240
<code>long-sm-short-desc:redefined to use</code> <code>accessibility information</code>	345	<code>\showgloplural:added\glsdetoklabel</code>	241
<code>footnote:redefined to use accessibility</code> <code>information</code>	349	<code>\showglopluralaccess: added</code> <code>\glsdetoklabel</code>	358
<code>footnote-desc:redefined to use accessi-</code> <code>bility information</code>	351	<code>\showgloshort:added\glsdetoklabel</code>	244
<code>footnote-sc:redefined to use accessibil-</code> <code>ity information</code>	351	<code>\showgloshortaccess:added\glsdetoklabel</code>	359
<code>footnote-sc-desc:redefined to use ac-</code> <code>cessibility information</code>	352	<code>\showgloshortpluralaccess: added</code> <code>\glsdetoklabel</code>	359
<code>footnote-sm:redefined to use accessibil-</code> <code>ity information</code>	351	<code>\showglosort:added\glsdetoklabel</code>	243
<code>footnote-sm-desc:redefined to use ac-</code> <code>cessibility information</code>	352	<code>\showglosymbol:added\glsdetoklabel</code>	243
<code>\renewacronymstyle:new</code>	211	<code>\showglosymbolaccess: added</code> <code>\glsdetoklabel</code>	358
<code>\showglocounter:added\glsdetoklabel</code>	242	<code>\showglosymbolplural: added</code> <code>\glsdetoklabel</code>	244
<code>\showglodesc:added\glsdetoklabel</code>	243	<code>\showglosymbolpluralaccess: added</code> <code>\glsdetoklabel</code>	358
<code>\showglodescaccess:added\glsdetoklabel</code>	358	<code>\showgлотext:added\glsdetoklabel</code>	241
<code>\showglodescplural:added\glsdetoklabel</code>	243	<code>\showgлотextaccess:added\glsdetoklabel</code>	358
<code>\showglodescpluralaccess: added</code> <code>\glsdetoklabel</code>	358	<code>\showgлотype:added\glsdetoklabel</code>	241
		<code>\showglouseri:added\glsdetoklabel</code>	242

<code>\showglouserii:added\glsdetoklabel</code>	242	<code>\@no@makeglossaries:new</code>	166
<code>\showglouseriii:added\glsdetoklabel</code>	242	<code>\@print@glossary:new</code>	178
<code>\showglouseriv:added\glsdetoklabel</code>	242	<code>\@print@noidx@glossary:new</code>	184
<code>\showglouserv:added\glsdetoklabel</code>	242	<code>\@printgloss@setsort:new</code>	176
<code>\showglouservi:added\glsdetoklabel</code>	243	<code>\@printglossary:new</code>	176
<code>dua:fixed bug in \acrfullfmt</code>	217	General: added sort key to printgloss group	191
fixed bug in <code>\Acrfullplfmt</code>	218	<code>\compatibleglossentry:</code> changed <code>\newcommand to \def as is may or may</code> not be defined	322
fixed bug in <code>\acrfullplfmt</code>	218	<code>\compatiblesubglossentry:</code> changed <code>\newcommand to \def as is may or may</code> not be defined	323
redefined to use accessibility informa- tion	347	<code>\defglsdisplayfirst:fixed unwanted</code> space	101
<code>dua-desc:commented spurious EOL</code> ..	218	<code>\glo@grabfirst:new</code>	185
redefined to use accessibility informa- tion	349	<code>\gls@defglossaryentry:replaced \ifx</code> with <code>\ifdefvoid</code>	80
4.04 (2014-03-04)		<code>\glsnoidxdisplayloc:new</code>	188
<code>\@gls@getcounterprefix:added warn-</code> ing if no prefix can be formed	173	<code>\glsnoidxdisplayloclisthandler:</code> new	188
4.04 (2014-03-06)		<code>\glsnoidxloclist:new</code>	187
<code>\@@gls@noidx@nosanitizesort:new</code> .	18	<code>\glsnoidxloclisthandler:new</code>	187
<code>\@@gls@noidx@sanitizesort:new</code> ...	18	<code>\glsnoidxstripaccents:new</code>	18
<code>\@@gls@nosanitizesort:new</code>	18	alttree: moved hangindent and parindent assignments outside level test	299
<code>\@@gls@sanitizesort:new</code>	17	<code>\makeglossaries: Moved definition of</code> <code>\glswrite to \makeglossaries</code> ..	161
<code>\@glo@addchildren:new</code>	180	<code>\makenoidxglossaries:new</code>	163
<code>\@glo@do@sortentries:new</code>	180	<code>\printglossary: changed to use new</code> <code>\@printglossary</code>	175
<code>\@glo@grabfirst:new</code>	185	<code>\printnoidxglossaries:new</code>	176
<code>\@glo@sortedinsert:new</code>	181	<code>\printnoidxglossary:new</code>	176
<code>\@glo@sortentries:new</code>	179	<code>\showgloclist:new</code>	244
<code>\@glo@sorthandler@case:new</code>	182	<code>\warn@noprintglossary: Activate warn-</code> ing in <code>\makeglossaries</code>	175
<code>\@glo@sorthandler@letter:new</code> ...	182	<code>\writeist: checked for definition of</code> <code>\glswrite</code>	153, 157
<code>\@glo@sorthandler@nocase:new</code> ...	182	4.06 (2014-03-12)	
<code>\@glo@sorthandler@word:new</code>	181	<code>\@GLS@:added \glsifhyper</code>	118
<code>\@glo@sortmacro@case:new</code>	183	<code>\@GLSpl:added \glsifhyper</code>	120
<code>\@glo@sortmacro@def:new</code>	184	<code>\@Gls@:added \glsifhyper</code>	117
<code>\@glo@sortmacro@def@do:new</code>	184	<code>\@Glspl@:added \glsifhyper</code>	120
<code>\@glo@sortmacro@letter:new</code>	183	<code>\@gls@:added \glsifhyper</code>	116
<code>\@glo@sortmacro@nocase:new</code>	183	<code>\@gls@numbersdef: added hook to set</code> toc title	27
<code>\@glo@sortmacro@standard:new</code> ...	183	<code>\@gls@symbolsdef: added hook to set</code> toc title	26
<code>\@glo@sortmacro@use:new</code>	184		
<code>\@glo@sortmacro@word:new</code>	182		
<code>\@gls@getothergrouptitle:new</code> ...	199		
<code>\@gls@noidx@do:new</code>	186		
<code>\@gls@noref@warn:new</code>	166		
<code>\@gls@reference:new</code>	188		
<code>\@gls@warnonglossdefined:new</code>	16		
<code>\@gls@warnontheglossdefined:new</code> .	16		

<code>\@glsdisp</code> : added <code>\glsifhyper</code>	121	renamed <code>\gls@type</code> to <code>\gls@type</code>	105
<code>\@glspl@</code> : added <code>\glsifhyper</code>	119	<code>\@glsdisp</code> : moved <code>\glsifhyper</code>	121
General: added <code>\glsifhyper</code>	134–141	moved check for first use to	
acronym: added hook to set toc title	13	<code>\@gls@link</code>	121
acronyms: added hook to set toc title	13	<code>\@glspl@</code> : moved <code>\glsifhyper</code>	119
<code>\glsdefmain</code> : added hook to set toc title	12	moved check for first use to	
4.07 (2014-04-04)		<code>\@gls@link</code>	119
<code>\@glossarysection</code> : added optional argument when using unstarred version	37	<code>\@ignored@glossaries</code> : new	58
<code>\@gls@noidx@do</code> : added <code>\global</code> in case it's used in a tabular-like style	186	General: added <code>entrycounter</code> option to <code>printgloss</code> family	190
<code>\Acrfullplfmt</code> : fixed no case change bug	207	added <code>nopostdot</code> option to <code>printgloss</code> family	190
<code>\glsletentryfield</code> : new	142	added <code>subentrycounter</code> option to <code>printgloss</code> family	190
4.08 (2014-07-30)		explicitly initialise hyper key	102
<code>\@ACRlong</code> : added <code>\do@gls@link@checkfirsthyper</code>	341	removed <code>\glsifhyper</code>	134–141
<code>\@ACRshort</code> : added <code>\do@gls@link@checkfirsthyper</code>	340	removed <code>\@sACRlongpl</code>	141
<code>\@Acrlong</code> : added <code>\do@gls@link@checkfirsthyper</code>	341	removed <code>\@sAcrlongpl</code>	140
<code>\@Acrshort</code> : added <code>\do@gls@link@checkfirsthyper</code>	340	removed <code>\@sacrlongpl</code>	140
<code>\@GLS@</code> : moved <code>\glsifhyper</code>	118	removed <code>\@sACRlong</code>	139
moved check for first use to		removed <code>\@sAcrlong</code>	138
<code>\@gls@link</code>	118	removed <code>\@sacrlong</code>	138
<code>\@GLSpl</code> : moved <code>\glsifhyper</code>	120	removed <code>\@sACRshortpl</code>	137
moved check for first use to		removed <code>\@sAcrshortpl</code>	137
<code>\@gls@link</code>	120	removed <code>\@sacrshortpl</code>	136
<code>\@Gls@</code> : moved <code>\glsifhyper</code>	117	removed <code>\@sACRshort</code>	135
moved check for first use to		removed <code>\@sAcrshort</code>	135
<code>\@gls@link</code>	117	removed <code>\@sacrshort</code>	134
<code>\@Glspl@</code> : moved <code>\glsifhyper</code>	120	removed <code>\@sgls@link</code>	103
moved check for first use to		removed <code>\@sGLSdescplural</code>	127
<code>\@acrlong</code> : added <code>\do@gls@link@checkfirsthyper</code>	341	removed <code>\@sGlsdescplural</code>	127
<code>\@acrshort</code> : added <code>\do@gls@link@checkfirsthyper</code>	339	removed <code>\@sGlsdesc</code>	127
<code>\@closegls</code> : new	159	removed <code>\@sGlsdesc</code>	126
<code>\@gls@</code> : moved <code>\glsifhyper</code>	116	removed <code>\@sglsdesc</code>	126
moved check for first use to		removed <code>\@glsdisp</code>	121
<code>\@gls@link</code>	116	removed <code>\@sGLSfirstplural</code>	125
<code>\@gls@doautomake</code> : new	25	removed <code>\@sGlsfirstplural</code>	124
<code>\@gls@field@link</code> : added assignment of <code>\do@gls@link@checkfirsthyper</code>	122	removed <code>\@sGLSfirst</code>	123
<code>\@gls@forbidtextext</code> : new	54	removed <code>\@sGlsfirst</code>	123
<code>\@gls@hyp@opt</code> : new	102	removed <code>\@sglsfirst</code>	123
<code>\@gls@link</code> : removed redundancy	105	removed <code>\@sGLSname</code>	126
		removed <code>\@sGlsname</code>	125
		removed <code>\@sglsname</code>	125
		removed <code>\@sGLSplural</code>	124
		removed <code>\@sGlsplural</code>	124
		removed <code>\@sglsplural</code>	124
		removed <code>\@sGLSpl</code>	120

removed \@sGlspl	119	switched to using \@gls@hyp@opt ..	207
removed \@sglsp1	118	\acrfullpl: removed \s@acrfullpl ..	207
removed \@sGLSsymbolplural	129	switched to using \@gls@hyp@opt ..	207
removed \@sGlsymbolplural	129	\ACRlong: switched to using \@gls@hyp@opt	
removed \@sglssymbolplural	128	139
removed \@sGLSsymbol	128	\Acrlong: switched to using \@gls@hyp@opt	
removed \@sGlsymbol	128	138
removed \@sglssymbol	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 \@sgluseriii	131	\acrlongpl: switched to using	
removed \@sGLSuserii	131	\@gls@hyp@opt	140
removed \@sGlsuserii	130	\ACRshort: switched to using	
removed \@sgluserii	130	\@gls@hyp@opt	135
removed \@sGLSuseriv	132	\Acrshort: switched to using	
removed \@sGlsuseriv	132	\@gls@hyp@opt	135
removed \@sgluseriv	132	\acrshort: switched to using	
removed \@sGLSuseri	130	\@gls@hyp@opt	134
removed \@sGlsuseri	130	\ACRshortpl: switched to using	
removed \@sgluseri	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 \@sGLS	118	\GLS: switched to using \@gls@hyp@opt	118
removed \@sGls	117	\Gls: switched to using \@gls@hyp@opt	117
removed \@sgls	116	\gls: switched to using \@gls@hyp@opt	116
removed \@thirdofthree (defined in		\gls@defglossaryentry: added check	
kernel)	116	for ignored glossary	76
removed sPGLS	252	\gls@istfilebase: new	33
removed sPgls	250	\glsaddkey: removed \@sGLS@user@<key>	
removed spgls	249	71
removed sPGLSpl	252	removed \@sGls@user@<key>	70
removed sPglsp1	251	removed \@sgls@user@<key>	70
removed spglsp1	250	switched to using \@gls@hyp@opt	70, 71
\ACRfull: removed \s@ACRfull	206	\GLSdesc: switched to using \@gls@hyp@opt	
switched to using \@gls@hyp@opt ..	206	126
\Acrfull: removed \@sAcrfull	206	\Glsdesc: switched to using \@gls@hyp@opt	
switched to using \@gls@hyp@opt ..	206	126
\acrfull: removed \@sacrfull	205	\glsdesc: switched to using \@gls@hyp@opt	
switched to using \@gls@hyp@opt ..	205	126
\ACRfullpl: removed \s@ACRfullpl ..	208	\GLSdescplural: switched to using	
switched to using \@gls@hyp@opt ..	208	\@gls@hyp@opt	127
\Acrfullpl: removed \s@Acrfullpl ..	207	\Glsdescplural: switched to using	
		\@gls@hyp@opt	127

<code>\glsdescplural:</code> switched to using <code>\@gls@hyp@opt</code>	127	<code>\Glsymbol:</code> switched to using <code>\@gls@hyp@opt</code>	128
<code>\glsdisablehyper:</code> added <code>\KV@glslink@hyperfalse</code> to definition	115	<code>\Glsymbol:</code> switched to using <code>\@gls@hyp@opt</code>	128
<code>\glsdisp:</code> switched to using <code>\@gls@hyp@opt</code>	121	<code>\GLSsymbolplural:</code> switched to using <code>\@gls@hyp@opt</code>	129
<code>\glsdohyperlink:</code> new	115	<code>\Glsymbolplural:</code> switched to using <code>\@gls@hyp@opt</code>	129
<code>\glsdohypertarget:</code> new	115	<code>\glsymbolplural:</code> switched to using <code>\@gls@hyp@opt</code>	128
<code>\glsenablehyper:</code> added <code>\KV@glslink@hypertrue</code> to definition	115	<code>\GLStext:</code> switched to using <code>\@gls@hyp@opt</code>	122
<code>\GLSfirst:</code> switched to using <code>\@gls@hyp@opt</code>	123	<code>\Glstext:</code> switched to using <code>\@gls@hyp@opt</code>	122
<code>\Glsfirst:</code> switched to using <code>\@gls@hyp@opt</code>	123	<code>\glstext:</code> switched to using <code>\@gls@hyp@opt</code>	122
<code>\glsfirst:</code> switched to using <code>\@gls@hyp@opt</code>	123	<code>\glstreenamfmt:</code> new	293
<code>\GLSfirstplural:</code> switched to using <code>\@gls@hyp@opt</code>	125	<code>\GLSuseri:</code> switched to using <code>\@gls@hyp@opt</code>	130
<code>\Glsfirstplural:</code> switched to using <code>\@gls@hyp@opt</code>	125	<code>\Glsuseri:</code> switched to using <code>\@gls@hyp@opt</code>	130
<code>\glsfirstplural:</code> switched to using <code>\@gls@hyp@opt</code>	124	<code>\glsuseri:</code> switched to using <code>\@gls@hyp@opt</code>	129
<code>\glsifhyper:</code> deprecated	102	<code>\GLSuserii:</code> switched to using <code>\@gls@hyp@opt</code>	131
<code>\glslink:</code> switched to using <code>\@gls@hyp@opt</code>	103	<code>\Glsuserii:</code> switched to using <code>\@gls@hyp@opt</code>	130
<code>\glslinkcheckfirsthyperhook:</code> new	104	<code>\glsuserii:</code> switched to using <code>\@gls@hyp@opt</code>	130
<code>\glslinkvar:</code> new	102	<code>\GLSuseriii:</code> switched to using <code>\@gls@hyp@opt</code>	131
<code>\GLSname:</code> switched to using <code>\@gls@hyp@opt</code>	126	<code>\Glsuseriii:</code> switched to using <code>\@gls@hyp@opt</code>	131
<code>\Glsname:</code> switched to using <code>\@gls@hyp@opt</code>	125	<code>\glsuseriii:</code> switched to using <code>\@gls@hyp@opt</code>	131
<code>\glsname:</code> switched to using <code>\@gls@hyp@opt</code>	125	<code>\GLSuseriv:</code> switched to using <code>\@gls@hyp@opt</code>	132
<code>\GLSpl:</code> switched to using <code>\@gls@hyp@opt</code>	120	<code>\Glsuseriv:</code> switched to using <code>\@gls@hyp@opt</code>	132
<code>\Glspl:</code> switched to using <code>\@gls@hyp@opt</code>	119	<code>\glsuseriv:</code> switched to using <code>\@gls@hyp@opt</code>	132
<code>\glspl:</code> switched to using <code>\@gls@hyp@opt</code>	118	<code>\GLSuserv:</code> switched to using <code>\@gls@hyp@opt</code>	133
<code>\GLSplural:</code> switched to using <code>\@gls@hyp@opt</code>	124	<code>\Glsuserv:</code> switched to using <code>\@gls@hyp@opt</code>	133
<code>\Glsplural:</code> switched to using <code>\@gls@hyp@opt</code>	124	<code>\glsuserv:</code> switched to using <code>\@gls@hyp@opt</code>	132
<code>\glsplural:</code> switched to using <code>\@gls@hyp@opt</code>	124	<code>\GLSuservi:</code> switched to using <code>\@gls@hyp@opt</code>	134
<code>\glsspace:</code> new	206		
<code>\GLSsymbol:</code> switched to using <code>\@gls@hyp@opt</code>	128		

\Glsuservi: switched to using		\@gls@usetranslator:new	21
\@gls@hyp@opt	134	\glsacrpluralsuffix:new	30
\glsuservi: switched to using		\glsadd: added check for vertical mode	150
\@gls@hyp@opt	133	\glsaddallunused: replaced @gobble	
\ifignoredglossary:new	58	with glsignore	151
altlongragged4col: fixed bug that displayed description instead of symbol	276	\glsifusedtranslatordict:new	21
\newglossary: added starred version	55	\glsignore:new	151
\newignoredglossary:new	57	\glsupacrpluralsuffix:new	30
\ns@newglossary: added \@glotype@<name>@log		\ProvidesGlossariesLang:new	30
.....	56	4.13 (2015-02-03)	
new	55	\indexspace:new	258, 277, 293
\p@gls@hyp@opt:new	103	4.14 (2015-02-28)	
\PGLS: changed to use \@gls@hyp@opt	252	\@@glslocalreset:new	84
\PglS: changed to use \@gls@hyp@opt	250	\@@glslocalunset:new	84
\pglS: changed to use \@gls@hyp@opt	249	\@@glsreset:new	85
\PGLSpl: changed to use \@gls@hyp@opt		\@@glsunset:new	84
.....	252	\@newglossaryentry@defcounters:	
\PglSpl: changed to use \@gls@hyp@opt		new	86
.....	251	\cGls:new	89
\pglSpl: changed to use \@gls@hyp@opt		\cGls@:new	89
.....	250	\cGlspl@:new	90
\s@gls@hyp@opt:new	103	\cglS:new	89
\s@newglossary:new	55	\cglS@:new	89
automake:new	25	\cglSpl:new	90
4.09 (2014-08-12)		\cglSpl@:new	90
\glsaddkey: fixed bug in user commands	70	\@gls@entry@count:new	88
4.10 (2014-08-27)		\@gls@increment@currcount:new	88
\@Gls@acentryname:new	143	\@gls@local@increment@currcount:	
\@Gls@entryname:new	143	new	88
\@gls@glossary: Renamed \@glossary to \@gls@glossary	168	\@gls@write@entrycounts:new	88
\glspercentchar:new	152	\@glslocalreset:new	84
\glstildechar:new	152	\@glslocalunset:new	84
alttree: moved space after symbol	299, 300	\@glsreset:new	84
4.11 (2014-09-01)		\@glsunset:new	84
\@@do@wrglossary: added hook	171	\@newglossaryentry@defcounters:	
sanitize: none option	20	new	80
\gls@wrglossary: renamed from \@wrglossary to \gls@wrglossary	168	\cGls:new	89
\glsaddprotectedpagefmt:new	170	\cglS:new	88
\glsbackslash:new	152	\cGlsformat:new	89
4.12 (2014-11-22)		\cglSformat:new	89
\@gls@addpredefinedattributes:		\cGlspl:new	90
Added glsignore attribute	42	\cglSpl:new	89
\@gls@adjustmode:new	151	\cGlsplformat:new	90
\@gls@notranslatorhook: removed	21	\cglSplformat:new	90
\@gls@toc: added \protect to \numberline	39	\gls@defdocnewglossaryentry:new	65
		\glsenableentrycount:new	86
		\glslocalreset: switched to \@glslocalreset	
		83

\glslocalunset: switched to \@glslocalunset	long-sp-short: new	212
..... 84	\showglofield: new	245
\glsreset: switched to \@glsreset	4.18 (2015-09-09)	
... 83	General: split mfirstuc into separate bundle	4
4.15 (2015-03-16) 4	
General: bug fix replaced \@glo@type	4.19 (2015-10-31)	
with \glstype	\glstreenamibox: new	298
..... 141	4.19 (2015-11-22)	
4.16 (2015-06-18)	\@gls@link@nocheckfirsthyper: new	122
\glsaddstoragekey: new	\@gls@preglossaryhook: new	176
..... 68	\@printglossary: added \@gls@preglossaryhook	
4.16 (2015-07-08) 178	
\@ACRlong: added \glspostlinkhook	\do@glsdisablehyperinlist: new	104
342	\doifglossarynoexistsordo: new	51
\@ACRshort: added \glspostlinkhook	\gls@gobbleopt: new	55
341	\glsdoifexistsordo: new	50
\@Acrlong: added \glspostlinkhook	4.20 (2015-11-30)	
341	\@gls@link: added \@gls@setdefault@glslink@opts	
\@Acrshort: added \glspostlinkhook 105	
340	added \glsdonohyperlink when hyperlink is suppressed	106
\@GLS@: added \glspostlinkhook	\@gls@setdefault@glslink@opts:	
... 118	new	105
\@GLSpl: added \glspostlinkhook	\gls@checkseeallowed@preambleonly:	
.. 121	new	61
\@Gls@: added \glspostlinkhook	\glsdonohyperlink: new	115
... 118	4.21 (2016-01-24)	
\@Glspl@: added \glspostlinkhook	\@printglossary: warn if no style has been set	176
. 120	General: changed checkfirsthyper assignment	134-141
\@acrlong: added \glspostlinkhook	\glossarystyle: set default style if not already set	200
341	\glsLTpenaltycheck: new	271
\@acrshort: added \glspostlinkhook	\glspatchLToutput: new	271
340	\glspenaltygroupskip: new	271
\@gls@: added \glspostlinkhook	altlong4col-booktabs: new	269
... 117	altlongragged4col-booktabs: new	270
\@gls@@link: added \glspostlinkhook	long-booktabs: new	268
..... 104	long3col-booktabs: new	268
\@gls@field@link: added \glspostlinkhook	long4col-booktabs: new	269
..... 122	longragged-booktabs: new	270
\@gls@link: moved definition of	longragged3col-booktabs: new	270
\glsifhyperon outside of this macro	\setglossarystyle: set default style if not already set	200
105		
\@glsdisp: added \glspostlinkhook		
121		
\@glspl@: added \glspostlinkhook		
. 119		
General: added \glspostlinkhook		
135-141		
\glsacspace: new		
..... 213		
\glsadd: changed \@do@wrglossary to		
\@do@wrglossary		
..... 151		
\glsfielddef: new		
..... 72		
\glsfieldedef: new		
..... 71		
\glsfieldfetch: new		
..... 72		
\glsfieldgdef: new		
..... 72		
\glsfieldxdef: new		
..... 71		
\glsifhyperon: moved definition of		
\glsifhyperon		
..... 104		
\glslinkpostsetkeys: new		
..... 104		
\glspostlinkhook: new		
..... 104		
\glswriteentry: new		
..... 169		
\ifglsfieldcseq: new		
..... 74		
\ifglsfielddefeq: new		
..... 73		
\ifglsfielddeq: new		
..... 73		

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, 306, 307
\&	30, 150
\'	18
\.	8, 18
\=	18
\?	109–111
\@@delimN	202
\@do@wrglossary	163, 171
\@do@wrglossary	151, 169
\@glo@assign@sortkey	164
\@glo@list	48
\@glo@sort	18
\@glo@type	176
\@glossarysec	5, 37, 38
\@glossaryseclabel	6, 38, 189
\@glossarysecstar	6, 37, 38, 189
\@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, 230, 232, 235, 237, 353–357
\@gls@fixbraces	173
\@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	206
\@ACRfullpl	208
\@ACRlong	139, 207
\@ACRlongpl	141, 208
\@ACRshort	135, 207
\@ACRshortpl	137, 208
\@Acrfull	206
\@Acrfullpl	207
\@Acrlong	138, 206
\@Acrlongpl	140, 207
\@Acrshort	135
\@Acrshortpl	137
\@Alph	169–171
\@GLS	118
\@GLS@	118, 252
\@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, 253
\@GLSplural	124
\@GLSplural@	124

<code>\@GLSsymbol</code>	128	<code>\@Glsuseriv</code>	132
<code>\@GLSsymbol@</code>	128	<code>\@Glsuseriv@</code>	132
<code>\@GLSsymbolplural</code>	129	<code>\@Glsuseriv</code>	133
<code>\@GLSsymbolplural@</code>	129	<code>\@Glsuseriv@</code>	133
<code>\@GLStext</code>	122	<code>\@Glsuserivi</code>	134
<code>\@GLStext@</code>	122	<code>\@Glsuserivi@</code>	134
<code>\@GLSuseri</code>	130	<code>\@Mi</code>	271
<code>\@GLSuseri@</code>	130	<code>\@PGLS</code>	252
<code>\@GLSuserii</code>	131	<code>\@PGLS@</code>	252
<code>\@GLSuserii@</code>	131	<code>\@PGLSpl</code>	252
<code>\@GLSuseriii</code>	131	<code>\@PGLSpl@</code>	252
<code>\@GLSuseriii@</code>	131, 132	<code>\@Pgls</code>	250
<code>\@GLSuseriv</code>	132	<code>\@Pgls@</code>	250
<code>\@GLSuseriv@</code>	132	<code>\@Pglspl</code>	251
<code>\@GLSuseriv</code>	133	<code>\@Pglspl@</code>	251
<code>\@GLSuseriv@</code>	133	<code>\@Roman</code>	170, 171
<code>\@GLSuserivi</code>	134	<code>\@acrfull</code>	205
<code>\@GLSuserivi@</code>	134	<code>\@acrfullpl</code>	207
<code>\@Gls</code>	117	<code>\@acrlong</code>	138, 206
<code>\@Gls@</code>	87, 89, 117, 251	<code>\@acrlongpl</code>	140, 207
<code>\@Gls@acentryname</code>	209	<code>\@acrshort</code>	134, 206
<code>\@Gls@entry@field</code>	69, 143–148	<code>\@acrshortpl</code>	136, 207
<code>\@Gls@entryname</code>	143, 209	<code>\@addtoacronymlists</code>	14
<code>\@Glsdesc</code>	126	<code>\@after</code>	14
<code>\@Glsdesc@</code>	126	<code>\@afterheading</code>	260, 310
<code>\@Glsdescplural</code>	127	<code>\@alph</code>	169–171
<code>\@Glsdescplural@</code>	127	<code>\@auxout</code>	54,
<code>\@Glsfirst</code>	123		56, 88, 161, 163, 166, 175, 178, 179, 254
<code>\@Glsfirst@</code>	123	<code>\@backslashchar</code>	108, 114
<code>\@Glsfirstplural</code>	125	<code>\@before</code>	14
<code>\@Glsfirstplural@</code>	125	<code>\@bsphack</code>	168
<code>\@Glsname</code>	125	<code>\@cGls</code>	89
<code>\@Glsname@</code>	125, 126	<code>\@cGls@</code>	87, 89
<code>\@Glspl</code>	119	<code>\@cGlspl</code>	90
<code>\@Glspl@</code>	87, 90, 119, 251, 252	<code>\@cGlspl@</code>	87, 90
<code>\@Glsplural</code>	124	<code>\@cclv</code>	271, 272
<code>\@Glsplural@</code>	124	<code>\@cgls</code>	88
<code>\@Glsymbol</code>	128	<code>\@cgls@</code>	87, 89
<code>\@Glsymbol@</code>	128	<code>\@cglspl</code>	89
<code>\@Glsymbolplural</code>	129	<code>\@cglspl@</code>	87, 90
<code>\@Glsymbolplural@</code>	129	<code>\@chapter</code>	28
<code>\@Glstext</code>	122, 123	<code>\@classoptionslist</code>	27
<code>\@Glstext@</code>	123	<code>\@colht</code>	271
<code>\@Glsuseri</code>	130	<code>\@colroom</code>	271, 272
<code>\@Glsuseri@</code>	130	<code>\@currentlabelname</code>	6, 189
<code>\@Glsuserii</code>	130	<code>\@curroptions</code>	27
<code>\@Glsuserii@</code>	130	<code>\@declaredoptions</code>	27
<code>\@Glsuseriii</code>	131	<code>\@delimN</code>	202
<code>\@Glsuseriii@</code>	131	<code>\@delimR</code>	202

<code>\@disable@onlypremakeg</code>	161	<code>\@glo@descplural</code>	59, 74, 75
<code>\@disable@premakecs</code>	29	<code>\@glo@descpluralaccess</code>	324–326
<code>\@disabled@glssaddxdycounters</code>	41	<code>\@glo@do@sortentries</code>	180
<code>\@do@addcounter</code>	40	<code>\@glo@entry</code>	151
<code>\@do@auxoutstuff</code>	178, 179	<code>\@glo@entryprefix</code>	247
<code>\@do@glossentry</code>	195, 322	<code>\@glo@entryprefixfirst</code>	247
<code>\@do@gls@getcounterprefix</code>	171	<code>\@glo@entryprefixfirstplural</code>	247, 248
<code>\@do@gls@islistofacronyms</code>	14	<code>\@glo@entryprefixplural</code>	247
<code>\@do@glssee</code>	80	<code>\@glo@esclabel</code>	81, 82
<code>\@do@ifinlist</code>	40	<code>\@glo@etext</code>	92, 94
<code>\@do@newglossaryentry</code>		<code>\@glo@first</code>	59, 75, 78, 79, 235, 357
	209, 223–230, 232, 234–237, 239, 353–357	<code>\@glo@firstaccess</code>	323, 325, 326
<code>\@do@seeglossary</code>	163, 174	<code>\@glo@firstplural</code>	59, 75, 78, 357
<code>\@do@subglossentry</code>	196, 323	<code>\@glo@firstpluralaccess</code>	324–326
<code>\@do@wrglossary</code>	105	<code>\@glo@grabfirst</code>	185
<code>\@do@writeaux@info</code>	175	<code>\@glo@label</code>	62, 68,
<code>\@ehc</code>	271		69, 71–80, 86, 149, 150, 247, 248, 325, 326
<code>\@empty</code>	11, 14, 25–27, 29, 40, 41, 44, 47,	<code>\@glo@list</code>	80
	48, 76, 77, 82, 106, 107, 116–120, 134–	<code>\@glo@long</code>	52, 62, 76, 79
	141, 153, 156, 158, 160, 167, 168, 172,	<code>\@glo@longaccess</code>	324–326
	173, 190, 192, 199, 224, 226, 228–232,	<code>\@glo@longpl</code>	62,
	234, 235, 237, 239, 304, 306, 308, 340–342		76, 79, 223, 226, 227, 230, 232, 235–237, 353
<code>\@end@fixbraces</code>	173, 174	<code>\@glo@longpluralaccess</code>	324–326
<code>\@end@fortrue</code>	22, 51, 68, 254	<code>\@glo@name</code>	10, 58, 63, 75, 77–79
<code>\@esphack</code>	169	<code>\@glo@no@assign@sortkey</code>	162
<code>\@expandtwoargs</code>	27	<code>\@glo@numfmt</code>	172, 304
<code>\@firstofone</code>	18, 19	<code>\@glo@parent</code>	11, 61, 75, 77, 78, 82, 181
<code>\@firstofthree</code>	103,	<code>\@glo@plural</code>	59, 75, 77, 78, 355
	116, 119, 121, 134, 136, 138, 140, 340–342	<code>\@glo@pluralaccess</code>	323, 325, 326
<code>\@firstoftwo</code>	21,	<code>\@glo@prefix</code>	
	22, 66, 67, 103, 119, 120, 136, 137, 140, 141		7, 61, 76, 81, 82, 107, 108, 172, 303, 304
<code>\@for</code>	22,	<code>\@glo@range</code>	172, 303, 304
	27, 29, 40, 41, 48, 66, 67, 108, 153–155,	<code>\@glo@see</code>	60, 75, 80
	161, 162, 169, 174, 180, 210, 224, 227,	<code>\@glo@seeautonumberlist</code>	7, 60
	228, 231, 233, 236, 238, 240, 254, 255, 304	<code>\@glo@short</code>	53, 62, 76, 79, 356
<code>\@glo@@desc</code>	79	<code>\@glo@shortaccess</code>	324–326, 353–356
<code>\@glo@@symbol</code>	79	<code>\@glo@shortpl</code>	62, 76, 79,
<code>\@glo@access</code>	323, 325, 326, 328		223, 225–227, 230, 232, 235, 237, 353, 356
<code>\@glo@addchildren</code>	180, 184	<code>\@glo@shortpluralaccess</code>	324–326
<code>\@glo@assign@sortkey</code>	162, 164, 191	<code>\@glo@sort</code>	10, 17, 18, 59, 75, 78, 81, 82
<code>\@glo@check@mkidxrangechar</code>		<code>\@glo@sortedinsert</code>	181
	107, 172, 303, 304	<code>\@glo@sortentries</code>	182–184
<code>\@glo@childlist</code>	180	<code>\@glo@sorthandler@case</code>	183
<code>\@glo@counter</code>	60, 75, 78	<code>\@glo@sorthandler@letter</code>	183
<code>\@glo@counterprefix</code>	166, 171–173, 199, 203	<code>\@glo@sorthandler@nocase</code>	184
<code>\@glo@default@sorttype</code>	9, 164, 182–184	<code>\@glo@sorthandler@word</code>	182
<code>\@glo@defaultcounter</code>	78	<code>\@glo@sortinghandler</code>	179, 181
<code>\@glo@desc</code>	59, 74–76, 79	<code>\@glo@sortinglist</code>	179–181, 184
<code>\@glo@descaccess</code>	324–326	<code>\@glo@sorttype</code>	164, 185, 186, 191

<code>\@glo@storeentry</code>	10, 12	<code>\@gls@checkescquote</code>	108
<code>\@glo@suffi</code>	107, 108, 172, 304	<code>\@gls@checklevel</code>	109
<code>\@glo@symbol</code>	52, 60, 75, 79, 229, 234, 325	<code>\@gls@checkmkidxchars</code>	81, 107, 163, 171, 173, 303, 304
<code>\@glo@symbolaccess</code>	324–326, 356	<code>\@gls@checkquote</code>	108
<code>\@glo@symbolplural</code>	60, 75, 79	<code>\@gls@classI</code>	154
<code>\@glo@symbolpluralaccess</code>	324–326	<code>\@gls@classII</code>	154
<code>\@glo@text</code>	59, 75, 78, 79, 116–121, 142, 143, 230, 248, 355	<code>\@gls@codepage</code>	179
<code>\@glo@textaccess</code>	323, 325, 326, 353–356	<code>\@gls@counter</code>	102, 105–107, 150, 151, 166, 172, 173, 304
<code>\@glo@thislabel</code>	80, 81	<code>\@gls@counterwithin</code>	9, 190, 192
<code>\@glo@thislettergrp</code>	185–187	<code>\@gls@ctr</code>	40
<code>\@glo@thisvalue</code>	53, 54	<code>\@gls@currentlettergroup</code>	185, 187
<code>\@glo@tmp</code>	68, 69, 172	<code>\@gls@declareoption</code>	7, 8, 12, 13, 16, 21, 24–27
<code>\@glo@type</code>	6, 11, 60, 75–80, 150, 151, 161, 167, 176–179, 181, 184, 185, 188, 189, 209, 224, 226, 228, 230, 232, 233, 235, 237, 239, 254, 255	<code>\@gls@default</code>	91
<code>\@glo@types</code>	48, 49, 55, 85, 151, 161, 162, 245	<code>\@gls@default@value</code>	52–54, 63, 64, 75, 77–79, 233, 247
<code>\@glo@useri</code>	61, 76, 78	<code>\@gls@deffile</code>	66, 67
<code>\@glo@userii</code>	61, 76, 78	<code>\@gls@defsort</code>	10, 11, 79
<code>\@glo@useriii</code>	61, 76, 79	<code>\@gls@defsortcount</code>	10, 11, 56
<code>\@glo@useriv</code>	61, 76, 79	<code>\@gls@do@acronymsdef</code>	13, 28, 57
<code>\@glo@userv</code>	61, 76, 79	<code>\@gls@do@indexdef</code>	27, 28, 57
<code>\@glo@uservi</code>	62, 76, 79	<code>\@gls@do@numbersdef</code>	26, 28, 57
<code>\@glodesc</code>	79	<code>\@gls@do@symbolsdef</code>	26, 57
<code>\@glolist@</code>	77	<code>\@gls@do@symbolssdef</code>	28
<code>\@gloname</code>	79	<code>\@gls@doautomake</code>	25, 162
<code>\@glossary@default@style</code>	6, 8, 176, 200, 240	<code>\@gls@docloadedfalse</code>	4
<code>\@glossaryentryfield</code>	82	<code>\@gls@docloadedtrue</code>	4
<code>\@glossarysection</code>	36	<code>\@gls@dodeflistparser</code>	162
<code>\@glossarystyle</code>	176, 177, 189	<code>\@gls@doentrydef</code>	100, 101
<code>\@glossarysubentryfield</code>	82	<code>\@gls@dolast</code>	174
<code>\@gls</code>	116	<code>\@gls@donext</code>	174
<code>\@gls@</code>	87, 89, 116, 250, 251	<code>\@gls@donext@def</code>	149
<code>\@gls@@link</code>	103	<code>\@gls@dothiswrite</code>	160
<code>\@gls@Hcounter</code>	106, 107	<code>\@gls@elem</code>	254
<code>\@gls@ReturnAfterFi</code>	203	<code>\@gls@encapchar</code>	111, 112, 157, 172, 173, 304, 307
<code>\@gls@access@display</code>	328, 329	<code>\@gls@entry@count</code>	88
<code>\@gls@actualchar</code>	82, 110, 113, 157, 307	<code>\@gls@entry@field</code>	69, 86, 143–149, 326–328
<code>\@gls@addpredefinedattributes</code>	153, 159	<code>\@gls@escbsdq</code>	108, 158, 308
<code>\@gls@adjustmode</code>	150	<code>\@gls@expand@fields</code>	64, 65
<code>\@gls@automake</code>	160, 162	<code>\@gls@expandonce</code>	64
<code>\@gls@between</code>	255, 256	<code>\@gls@fetchfield</code>	53
<code>\@gls@body</code>	143	<code>\@gls@field@link</code>	70, 71, 122–134
<code>\@gls@checkactual</code>	109	<code>\@gls@firsttok</code>	185
<code>\@gls@checkbar</code>	109	<code>\@gls@fixbraces</code>	80
<code>\@gls@checkedmkidx</code>	108–114	<code>\@gls@forbidtexext</code>	56
<code>\@gls@checkescactual</code>	109	<code>\@gls@get@counterprefix</code>	172, 173
<code>\@gls@checkescbar</code>	109		

<code>\@gls@getbody</code>	143	<code>\@gls@numlist@lastsep</code>	149, 150
<code>\@gls@getcounterprefix</code>	171	<code>\@gls@numlist@nextsep</code>	149
<code>\@gls@getgrouptitle</code>	163, 198, 256	<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>	325
<code>\@gls@grptitle</code>	198, 254, 256	<code>\@gls@oldnewglossaryentryprehook</code>	325
<code>\@gls@hyp@opt</code>	70,	<code>\@gls@onlypremakeg</code>	29
	71, 88–90, 103, 116–141, 205–208, 249–252	<code>\@gls@order</code>	160
<code>\@gls@hyp@opt@cs</code>	103	<code>\@gls@org@LT@output</code>	271
<code>\@gls@hypergroup</code>	254	<code>\@gls@org@glsnoidxdisplayloc</code>	165
<code>\@gls@ifinlist</code>	40	<code>\@gls@org@glsseeformat</code>	165
<code>\@gls@ifnotmeasuring</code>	83	<code>\@gls@preglossaryhook</code>	178
<code>\@gls@igtype</code>	58	<code>\@gls@prevlevel</code>	280, 281, 298–301, 316, 317
<code>\@gls@increment@currcount</code>	86	<code>\@gls@provide@newglossary</code>	56
<code>\@gls@indexdef</code>	27	<code>\@gls@quotechar</code>	109–113, 157, 307
<code>\@gls@islistofacronyms</code>	14	<code>\@gls@reference</code>	163, 166
<code>\@gls@keylist</code>	352	<code>\@gls@removespaces</code>	202, 203
<code>\@gls@keymap</code>	66–69, 247, 325	<code>\@gls@renewglossary</code>	159
<code>\@gls@label</code>	163, 166, 171, 172	<code>\@gls@replacementtext</code>	328
<code>\@gls@langmod</code>	160	<code>\@gls@rest</code>	143
<code>\@gls@levelchar</code>	82, 111–113, 157, 307	<code>\@gls@roman</code>	43, 304, 305
<code>\@gls@link</code>	103, 116–122, 134–141, 340–342	<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, 225, 231	<code>\@gls@sanitizesort</code>	10
<code>\@gls@link@nocheckfirsthyper</code>	122, 134–141	<code>\@gls@sanitizesymbol</code>	23
<code>\@gls@link@opts</code>	105, 225, 231	<code>\@gls@saveentrycounter</code>	105, 151
<code>\@gls@list</code>	254, 255	<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, 240	<code>\@gls@setsort</code>	10, 11, 105
<code>\@gls@loadsuper</code>	8, 240	<code>\@gls@setupshortcuts</code>	28
<code>\@gls@loadtree</code>	8, 240	<code>\@gls@sort</code>	186, 187
<code>\@gls@local@increment@currcount</code>	87	<code>\@gls@sort@A</code>	181, 182
<code>\@gls@loclist</code>	164, 165, 186, 187	<code>\@gls@sort@B</code>	181, 182
<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>	328	<code>\@gls@this</code>	169
<code>\@gls@noexpand@fields</code>	65	<code>\@gls@thisHloc</code>	172
<code>\@gls@nohyperlist</code>	15, 58, 105	<code>\@gls@thisfield</code>	53
<code>\@gls@noidx@do</code>	185	<code>\@gls@thislabel</code>	51, 174, 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>	172
<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, 187, 188	<code>\@gls@tmp</code>	11, 31, 44, 64, 108, 168, 255, 256
<code>\@gls@noidx@loclist@sep</code>	165, 187, 188	<code>\@gls@tmpb</code>	109–114
<code>\@gls@noref@warn</code>	163, 185	<code>\@gls@toc</code>	38
<code>\@gls@numberlink</code>	202	<code>\@gls@type</code>	162,
<code>\@gls@numbersdef</code>	26		210, 224, 227, 228, 231, 233, 236, 238, 240

<code>\@gls@updatechecked</code>	108, 109	<code>\@glspl@</code>	87, 90, 118, 250–252
<code>\@gls@usetranslator</code>	21, 22, 31	<code>\@glsplural</code>	124
<code>\@gls@value</code>	63, 64	<code>\@glsplural@</code>	124
<code>\@gls@warnonglossdefined</code>	16, 175	<code>\@glsreset</code>	83, 87
<code>\@gls@warnontheglossdefined</code>	16, 194	<code>\@glssee</code>	80, 174
<code>\@gls@write@entrycounts</code>	88	<code>\@glsymbol</code>	128
<code>\@gls@writedef</code>	66	<code>\@glsymbol@</code>	128
<code>\@gls@xdy@locationlist</code>	154	<code>\@glsymbolplural</code>	128
<code>\@gls@xdycheckbackslash</code>	108	<code>\@glsymbolplural@</code>	128, 129
<code>\@gls@xdycheckquote</code>	108	<code>\@gls@target</code>	115, 116, 195, 254
<code>\@gls@xref</code>	173	<code>\@gls@text</code>	122
<code>\@gls@Alphacompositor</code>	34, 44, 305	<code>\@gls@text@</code>	122
<code>\@gls@Hlocref</code>	171	<code>\@gls@unset</code>	84, 86
<code>\@gls@acronymlists</code>	14, 15, 48, 209, 210, 224, 226–228, 230–233, 235–240, 245	<code>\@gls@useri</code>	129
<code>\@gls@addkey</code>	69	<code>\@gls@useri@</code>	129
<code>\@gls@addstoragekey</code>	68	<code>\@gls@userii</code>	130
<code>\@gls@addxdyattribute</code>	41	<code>\@gls@userii@</code>	130
<code>\@gls@defaultsort</code>	10	<code>\@gls@useriii</code>	131
<code>\@gls@desc</code>	126	<code>\@gls@useriii@</code>	131
<code>\@gls@desc@</code>	126	<code>\@gls@useriv</code>	132
<code>\@gls@descplural</code>	127	<code>\@gls@useriv@</code>	132
<code>\@gls@descplural@</code>	127	<code>\@gls@userv</code>	132, 133
<code>\@gls@disp</code>	121	<code>\@gls@userv@</code>	133
<code>\@gls@entry</code>	85, 88	<code>\@gls@uservi</code>	133
<code>\@gls@first</code>	123	<code>\@gls@uservi@</code>	133
<code>\@gls@first@</code>	123	<code>\@gls@widestname</code>	298–300, 316
<code>\@gls@firstletter</code>	48, 152	<code>\@gls@writefiles</code>	25
<code>\@gls@firstplural</code>	124	<code>\@gobble</code>	11, 66, 67, 108, 152, 155, 163, 302, 306, 307
<code>\@gls@firstplural@</code>	124	<code>\@idxitem</code>	278, 294
<code>\@gls@hypernumber</code>	201	<code>\@ifclassloaded</code>	4, 9, 37
<code>\@gls@isacronymlistfalse</code>	14	<code>\@ifnextchar</code>	56, 103
<code>\@gls@isacronymlisttrue</code>	14	<code>\@ifpackageloaded</code>	4, 21, 22, 31, 47, 83, 149, 322
<code>\@gls@link</code>	105, 115, 150, 254	<code>\@ifstar</code>	55, 68, 69, 103, 204
<code>\@gls@localreset</code>	83, 87	<code>\@ifundefined</code>	30, 255, 262, 272, 281, 288, 299, 316, 330
<code>\@gls@localunset</code>	84, 87	<code>\@ignored@glossaries</code>	57, 58
<code>\@gls@locref</code>	166, 171, 172, 303, 304	<code>\@input@</code>	178
<code>\@gls@minrange</code>	153, 154, 305	<code>\@istfilename</code>	161
<code>\@gls@name</code>	125	<code>\@makecol</code>	271, 272
<code>\@gls@name@</code>	125	<code>\@makeglossary</code>	161
<code>\@gls@nextpages</code>	177	<code>\@minus</code>	258, 277, 293
<code>\@gls@nodesc</code>	75, 76, 79	<code>\@mkboth</code>	37
<code>\@gls@noname</code>	75, 77, 79	<code>\@newglossary</code>	54, 56
<code>\@gls@nonextpages</code>	177	<code>\@newglossary@entry@defcounters</code>	80, 86
<code>\@gls@numberformat</code>	102, 105, 150, 166, 172, 303, 304	<code>\@newglossary@entry@posthook</code>	68, 69, 80, 247, 325
<code>\@gls@openfile</code>	159, 167	<code>\@newglossary@entry@prehook</code>	68, 69, 74, 76, 247, 325
<code>\@gls@order</code>	161		
<code>\@gls@pl</code>	118		

<code>\Acrlong</code>	222	<code>\bgroup</code>	18, 74, 149, 177, 180		
<code>\acrlong</code>	222	booktabs package	268–270		
<code>\Acrlongpl</code>	222	<code>\boolean</code>	238		
<code>\acrlongpl</code>	222	<code>\boolfalse</code>	26		
<code>\acrnameformat</code>	230, 355	<code>\booltrue</code>	26		
<code>\acronymentry</code>	209, 212–216, 218–221, 344–346, 349–352	<code>\bottomrule</code>	268, 269		
<code>\acronymfont</code>	98, 134–137, 143, 148, 208, 210, 212–221, 225, 227, 229, 231–234, 236, 337, 338, 340–342, 344–346, 348–352, 354–356	<code>\box</code>	271		
<code>\acronymname</code>	13, 32	C			
<code>\acronymsort</code>	209, 212–216, 218, 219, 221, 344–346, 349, 350, 352	<code>\c</code>	18		
<code>\acronymtype</code>	13, 209, 210, 223–230, 232, 234–237, 239, 353–356	<code>\c@equation</code>	106		
<code>\acrpluralsuffix</code>	209, 212–214, 218–220, 223, 224, 226, 227, 230–233, 235–237, 239, 344, 345, 349–351, 353–357	<code>\c@glossarysubentry</code>	190		
<code>\Acrshort</code>	221	<code>\c@page</code>	169–171		
<code>\acrshort</code>	221	<code>\cGls</code>	89		
<code>\Acrshorttpl</code>	222	<code>\cGls</code>	89		
<code>\acrshorttpl</code>	221	<code>\cGlsformat</code>	87		
<code>\addcontentsline</code>	39	<code>\cGlsformat</code>	87		
<code>\addglossarytocaptions</code>	31	<code>\cGlspl</code>	90		
<code>\addtolength</code>	299, 300, 316	<code>\cGlspl</code>	90		
<code>\advance</code>	11, 77, 106, 271	<code>\cGlsplformat</code>	87		
<code>\AE</code>	19	<code>\cGlsplformat</code>	87		
<code>\ae</code>	19	<code>\changes</code>	104, 160, 259		
amsgen package	4, 101	<code>\char</code>	199		
amsmath package	83	<code>\cleardoublepage</code>	38		
<code>\andname</code>	174	<code>\clearpage</code>	38		
<code>\AnyTrackedLanguages</code>	31, 360	<code>\closeout</code>	66, 157–159, 167		
<code>\appto</code>	15, 68, 69, 247, 325	<code>\compatglossarystyle</code>	309–321		
array package	268, 272, 288	<code>\compatibleglossentry</code>	197		
article class	172	<code>\compatiblesubglossentry</code>	197		
<code>\AtBeginDocument</code>	13, 47, 66, 83, 151, 163	<code>\copy</code>	271, 272		
<code>\AtEndDocument</code>	25, 66, 88, 163, 167, 178, 179, 255	<code>\count@</code>	186		
B				<code>\cs</code>	104
<code>\b</code>	19	<code>\csdef</code>	16, 17, 68–71, 80, 81, 86, 87, 180, 181, 201, 211, 308		
babel package	21, 29, 31, 46	<code>\csedef</code>	88, 170		
<code>\begin</code>	104, 155, 160, 185, 259, 262–267, 270–293, 306	<code>\csgdef</code>	36, 54, 58, 87, 88, 175, 188		
<code>\BeginAccSupp</code>	328	<code>\cslet</code>	74, 81, 184		
<code>\begingroup</code>	5, 168, 170, 202	<code>\csname</code>	9–11, 27, 29, 31, 32, 37, 38, 41, 43, 44, 47, 48, 51, 56, 57, 63, 64, 66, 68–73, 77–82, 84, 85, 100, 101, 105–107, 116–121, 134–142, 149, 151, 154, 155, 159, 163, 166–170, 172, 173, 176–179, 181, 189, 195, 196, 200, 204, 240–248, 254, 255, 298–300, 302–304, 316, 322, 323, 325–327, 330, 340–342, 358, 359		
<code>\bfseries</code>	263–266, 268, 269, 273–277, 282–287, 289, 291–293	<code>\csshow</code>	245		
		<code>\csuse</code>	32, 35, 54, 63, 64, 70, 71, 100, 101, 160, 181, 183, 185, 186, 188–190, 200, 211, 248, 309–321		

<code>\csxdef</code>	79, 88	<code>\do</code>	22, 27, 29, 40, 41, 48, 66, 67, 108, 149, 153–155, 161, 162, 169, 174, 180, 210, 224, 227, 228, 231, 233, 236, 238, 240, 254, 255, 304
<code>\currentglossary</code>	35, 177, 190, 192	<code>\do@glo@storeentry</code>	10, 11, 80
<code>\currentglssubentry</code>	190–193	<code>\do@gls@link@checkfirsthyper</code> 103, 105, 116–122, 134–141, 340, 341
<code>\CurrentOption</code>	27, 247, 322	<code>\do@gls@xdycheckbackslash</code>	108
<code>\CurrentTrackedLanguage</code>	32, 360, 361	<code>\do@gls@disablehyperinlist</code>	105
<code>\CurrentTrackedTag</code>	32, 360, 361	<code>\do@glshaschildren</code>	51
<code>\CustomAcronymFields</code>	239	doc package	4, 5, 12
<code>\CustomNewAcronymDef</code>	240	<code>\doifglossarynoexistsordo</code>	55
D			
<code>\d</code>	18	<code>\dtl@ifsingle</code>	198
datatool package	181	<code>\dtl@insertinto</code>	181
<code>\day</code>	153, 157, 304, 307	<code>\dtl@sortresult</code>	181, 182
<code>\DeclareAcronymList</code>	13, 15, 209, 210, 224, 226, 228, 230, 233, 235, 237, 239	<code>\dtlcompare</code>	182
<code>\DeclareListParser</code>	162	<code>\dtlicompare</code>	182
<code>\DeclareOption</code>	7, 247, 322	<code>\DTLifinlist</code>	58, 105
<code>\DeclareOptionX</code>	7	<code>\DTLifint</code>	199
<code>\DeclareRobustCommand</code> 32, 174, 175, 233, 328–330	<code>\dtlletterindexcompare</code>	182
<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, 150, 153, 157, 160, 162, 164, 165, 167, 170–174, 176, 177, 179, 184– 192, 198–209, 224, 226, 228–230, 232, 234, 235, 237, 239, 247, 249–253, 255– 258, 280, 281, 298–301, 303, 304, 307, 309, 316, 317, 322–325, 339–342, 353–357		<code>\DTLsubstituteall</code>	108
<code>\def@gls@xdycheckbackslash</code>	114	<code>\dtlwordindexcompare</code>	181
<code>\DefaultNewAcronymDef</code>	224	<code>\DUANewAcronymDef</code>	238
<code>\defglsentryfmt</code>	56, 58, 100, 101, 210, 223, 225, 227, 229, 231, 234, 236, 239	E	
<code>\define@boolkey</code> 5, 7–9, 12, 19, 20, 23–26, 102, 191	<code>\eappto</code>	57, 58, 81, 170
<code>\define@choicekey</code> 5, 6, 9, 20, 21, 24, 61, 189, 190	<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, 171, 172, 175, 178, 179, 181, 182, 186, 190, 193, 199, 203, 204, 223, 225, 227, 229, 232, 234, 236, 254, 302, 303, 305, 307, 352–356
<code>\define@key</code> 6, 9, 15, 20, 24, 25, 58–62, 68, 69, 101, 102, 150, 188, 189, 191, 247, 323, 324		<code>\egroup</code>	18, 74, 150, 178, 180
<code>\DefineAcronymSynonyms</code>	28, 223	<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, 152, 153, 156–160, 167– 174, 177, 186, 190–194, 199, 202, 203, 213, 227, 228, 231, 233, 236, 238, 240, 255, 259, 262, 264, 265, 268, 269, 271, 273, 274, 276, 282, 283, 285, 289, 290, 292, 294, 296–300, 303–309, 314–317, 328
<code>\delimN</code>	156, 162, 187, 202, 306	<code>\emph</code>	174, 203
<code>\delimR</code>	156, 202, 306	<code>\empty</code>	203
<code>\DescriptionDUANewAcronymDef</code>	228	<code>\end</code>	104, 156, 159, 160, 185, 259, 262–267, 270–293, 306
<code>\DescriptionFootnoteNewAcronymDef</code> ..	226	<code>\end@doifinlist</code>	40
<code>\descriptionname</code>	32, 263–266, 268, 269, 273–277, 282–287, 289, 291–293	<code>\end@getprefix</code>	172, 173
<code>\DescriptionNewAcronymDef</code>	230		
<code>\dimen@</code>	213, 271		
<code>\disable@keys</code>	27		

glossary package	1, 204	longheader	263, 268, 311
glossary styles:		longheaderborder	263, 311
altlist	260, 310	longragged	270, 272–274
altlistgroup	260, 310	longragged-booktabs	270
altlisthypergroup	260, 310	longragged3col	270, 274, 275, 313
altlong4col	266, 267, 275, 312	longragged3col-booktabs	270
altlong4col-booktabs	269, 271	longragged3colborder	274, 313
altlong4colborder	267, 312	longragged3colheader	275, 313
altlong4colheader	267, 269, 312	longragged3colheaderborder .	275, 313
altlong4colheaderborder	267, 312	longraggedborder	273, 312
altlongragged4col ...	270, 275, 276, 313	longraggedheader	273, 313
altlongragged4col-booktabs	270	longraggedheaderborder	273, 313
altlongragged4colborder	276, 313	mcolalttree	280, 318
altlongragged4colheader	276, 313	mcolalttreegroup	280, 318
altlongragged4colheaderborder	277, 314	mcolalttreehypergroup	280, 318
altsuper4col	286, 287, 291, 321	mcolindex	278, 317
altsuper4colborder	287, 321	mcolindexgroup	278, 317
altsuper4colheader	286, 321	mcolindexhypergroup	278, 317
altsuper4colheaderborder ...	287, 321	mcoltree	279, 317
altsuperragged4col	291–293, 319	mcoltreegroup	317
altsuperragged4colborder ...	292, 319	mcoltreehypergroup	279, 317
altsuperragged4colheader ...	292, 319	mcoltreenoname	279, 318
altsuperragged4colheaderborder .		mcoltreenonamegroup	280, 318
.....	293, 319	mcoltreenonamehypergroup ...	280, 318
alttree	280, 298, 300, 316	sublistdotted	310
alttreegroup	300, 317	super	281–283, 289, 320
alttreehypergroup	300, 317	super3col	283, 284, 320
index	278, 293–295, 314	super3colborder	284, 320
indexgroup	294, 295, 314	super3colheader	284, 320
indexhypergroup	295, 314	super3colheaderborder	284, 320
inline	309	super4col	284–286, 321
list	6, 259–261, 309	super4colborder	285, 321
listdotted	261, 310	super4colheader	285, 321
listgroup	259, 309	super4colheaderborder	286, 321
listhypergroup	259, 310	superborder	282, 320
long	262, 263, 268, 272, 310, 312	superheader	282, 320
long-booktabs	268, 270	superheaderborder	283, 320
long3col	263, 264, 268, 311	superragged	288, 289, 318
long3col-booktabs	268, 270	superragged3col	289–291, 319
long3colborder	264, 311	superragged3colborder	290, 319
long3colheader	264, 268, 311	superragged3colheader	290, 319
long3colheaderborder	264, 311	superragged3colheaderborder	291, 319
long4col	265, 266, 269, 311	superraggedborder	289, 318
long4col-booktabs	269	superraggedheader	289, 318
long4colborder	266, 312	superraggedheaderborder	289, 319
long4colheader	265, 269, 312	tree	278, 295, 296, 298, 314
long4colheaderborder	266, 312	treegroup	279, 296, 315
longborder	263, 311	treehypergroup	296, 315
		treenoname	279, 296, 297, 315

treenonamegroup	297, 316	\gls@assign@descplural	224, 232, 235, 237, 353, 356, 357
treenonamehypergroup	297, 316	\gls@assign@field	65, 68, 69, 74, 76, 78, 79, 247, 248
glossary-hypernav package	152	\gls@assign@firstpl	223, 224, 226, 228, 230, 232, 235, 237, 353–357
glossary-list package	6, 8, 258	\gls@assign@plural	224, 226, 228, 230, 232, 235, 237, 353–357
glossary-long package	7, 261, 262, 275, 281	\gls@assign@symbolplural	224, 226, 228, 230, 235, 237, 354, 355, 357
glossary-longragged package	272	\gls@checkisacronymlist	104
glossary-mcols package	277	\gls@checkseeallowed	60, 65, 161, 163
glossary-super package .. 7, 8, 262, 281, 287, 291		\gls@checkseeallowed@preambleonly ..	65
glossary-superragged package	287	\gls@codepage	47, 160, 179
glossary-tree package	8, 293	\gls@defdocnewglossaryentry	66, 86
\glossaryentry	172, 173, 304	\gls@defglossaryentry	65, 66, 74
glossaryentry (counter)	9, 193, 194	\gls@disablepagerefexpansion ..	169, 171
\glossaryentryfield	195, 309–316, 318–321, 343	\gls@do@addxdyattribute	41
\glossaryentrynumbers	7, 156, 177, 178, 186, 187, 191, 192, 306	\gls@doclearpage	39
\glossaryheader	155, 185, 256, 259, 260, 262–266, 268, 269, 272–283, 285, 288, 290, 291, 294–298, 301, 306	\gls@dosubst	108
\glossarymark	36	\gls@dotocitle	177, 189
\glossaryname	12, 31, 32	\gls@end@sanitizesort	18
\glossarypostamble	156, 185, 306	\gls@endcheck	64
\glossarypreamble	155, 185, 306	\gls@glossary	168, 172, 173
\glossarysection	155, 185, 306	\gls@gobbleopt	56
glossarysubentry (counter)	9, 192–194	\gls@grplabel	254
\glossarysubentryfield	196, 309–316, 318–321, 343	\gls@hypergrouprerun	255
\glossarytitle .. 155, 176, 177, 185, 189, 306		\gls@ifnotmeasuring	83, 84
\glossarytocitle	6, 12, 13, 26, 27, 29, 32, 36, 155, 176, 185, 189, 306	\gls@inlinepostchild	256–258, 309
\glossentry	81, 177, 187, 197, 257, 259–262, 264, 265, 273, 274, 276, 282, 283, 285, 288, 289, 290, 291, 294, 295, 297, 298	\gls@inlinesep	256, 257, 309
\Glossentrydesc	342	\gls@inlinesubsep	256, 257, 309
\glossentrydesc	257, 259–262, 264, 265, 273, 274, 276, 282, 283, 285, 288, 290, 292, 294–297, 299, 300, 342	\gls@islistofacronyms	14
\glossentryname	257, 259–262, 264, 265, 273, 274, 276, 282, 283, 285, 288, 290, 292, 294–297, 299, 300, 342	\gls@istfilebase	33, 160
\Glossentrysymbol	343	\gls@label	204
\glossentrysymbol	257, 265, 276, 285, 292, 294–297, 299, 300, 343	\gls@level	77, 78, 186
\Gls	89, 204, 222	\gls@noidxglossary	163
\gls	89, 163, 193, 204, 222	\gls@numberpage	169, 171
\gls@Alphpage	169, 171	\gls@org@glossaryentryfield	177
\gls@alphpage	169, 171	\gls@org@glossarysubentryfield ..	177, 178
\gls@assign@desc	74, 79	\gls@org@insert	229, 231, 234
\gls@assign@descplural	224, 232, 235, 237, 353, 356, 357	\gls@protected@pagefmts	108, 169, 170
\gls@assign@field	65, 68, 69, 74, 76, 78, 79, 247, 248	\gls@Romanpage	169, 171
\gls@assign@firstpl	223, 224, 226, 228, 230, 232, 235, 237, 353–357	\gls@romanpage	169, 171
\gls@assign@plural	224, 226, 228, 230, 232, 235, 237, 353–357	\gls@save@numberlist	7
\gls@assign@symbolplural	224, 226, 228, 230, 235, 237, 354, 355, 357	\gls@suffiX	34, 156, 158, 306, 308
\gls@checkisacronymlist	104	\gls@suffiXFF	35, 156, 158, 306, 308
\gls@checkseeallowed	60, 65, 161, 163	\gls@text	100
\gls@checkseeallowed@preambleonly ..	65	\gls@thissty	22
\gls@codepage	47, 160, 179	\gls@tmp	167, 233
\gls@defdocnewglossaryentry	66, 86		
\gls@defglossaryentry	65, 66, 74		
\gls@disablepagerefexpansion ..	169, 171		
\gls@do@addxdyattribute	41		
\gls@doclearpage	39		
\gls@dosubst	108		
\gls@dotocitle	177, 189		
\gls@end@sanitizesort	18		
\gls@endcheck	64		
\gls@glossary	168, 172, 173		
\gls@gobbleopt	56		
\gls@grplabel	254		
\gls@hypergrouprerun	255		
\gls@ifnotmeasuring	83, 84		
\gls@inlinepostchild	256–258, 309		
\gls@inlinesep	256, 257, 309		
\gls@inlinesubsep	256, 257, 309		
\gls@islistofacronyms	14		
\gls@istfilebase	33, 160		
\gls@label	204		
\gls@level	77, 78, 186		
\gls@noidxglossary	163		
\gls@numberpage	169, 171		
\gls@org@glossaryentryfield	177		
\gls@org@glossarysubentryfield ..	177, 178		
\gls@org@insert	229, 231, 234		
\gls@protected@pagefmts	108, 169, 170		
\gls@Romanpage	169, 171		
\gls@romanpage	169, 171		
\gls@save@numberlist	7		
\gls@suffiX	34, 156, 158, 306, 308		
\gls@suffiXFF	35, 156, 158, 306, 308		
\gls@text	100		
\gls@thissty	22		
\gls@tmp	167, 233		

<code>\gls@tmplen</code>	115, 299, 300, 316, 317	<code>\glsdetoklabel</code>	49–53, 66, 71–75, 80, 84–88, 105, 142, 143, 149–151, 163– 165, 171, 177, 178, 181, 182, 186, 188, 190, 193, 195, 240–245, 322, 323, 358, 359
<code>\gls@tr@set@acronym@toctitle</code>	13	<code>\glsdisplay</code>	91, 101
<code>\gls@tr@set@main@toctitle</code>	12	<code>\glsdisplayfirst</code>	91, 100
<code>\gls@tr@set@numbers@toctitle</code>	27	<code>\glsdisplaynumberlist</code>	164
<code>\gls@tr@set@symbols@toctitle</code>	26	<code>\glsdohyperlink</code>	115
<code>\gls@wrglossary</code>	168	<code>\glsdohypertarget</code>	115, 116
<code>\gls@xdystring</code>	108	<code>\glsdoifexists</code>	.. 51–53, 71–74, 83, 84, 116–122, 134– 141, 149, 150, 164, 165, 249–253, 339–343
<code>\gls@xindy@glsnumbersfalse</code>	25	<code>\glsdoifexistsordo</code>	103, 142
<code>\gls@xindy@glsnumberstrue</code>	24	<code>\glsdoifexistsorwarn</code>	188, 195, 196
<code>\glsaccsupp</code>	328	<code>\glsdoifnoexists</code>	65, 74
<code>\glsacronymtrue</code>	13	<code>\glsdonohyperlink</code>	106, 115
<code>\glsacrpluralsuffix</code>	30, 205, 214, 218–220, 224	<code>\glsdosanitizesort</code>	9, 10
<code>\glsacrshortcutsfalse</code>	28	<code>\glsentryaccess</code>	328
<code>\glsacrshortcutstrue</code>	28	<code>\glsentrycounter</code>	199, 203
<code>\glsacspace</code>	212, 215	<code>\glsentrycounterfalse</code>	9
<code>\glsadd</code>	151	<code>\glsentrycounterlabel</code>	190, 194
<code>\glsadd options</code>		<code>\glsentrycountertrue</code>	9
<code>counter</code>	150	<code>\glsentrycurrcount</code>	86, 88
<code>format</code>	150, 201	<code>\glsentrydesc</code>	126, 196, 342
<code>\glsaddall options</code>		<code>\glsentrydesc</code>	.. 93–95, 126, 127, 196, 332–334, 342
<code>types</code>	150, 151	<code>\glsentrydescaccess</code>	329
<code>\GlsAddXdyAttribute</code>	40–42, 302, 303	<code>\Glsentrydescplural</code>	127
<code>\GlsAddXdyCounters</code>	40, 41, 57	<code>\glsentrydescplural</code>	.. 92, 93, 127, 330, 331
<code>\glsautomakefalse</code>	25	<code>\glsentrydescpluralaccess</code>	329
<code>\glsautoprefix</code>	6, 189	<code>\Glsentryfirst</code>	.. 89, 94, 97, 123, 333, 336
<code>\glscapscase</code>	91, 93, 95–99, 116–121, 134–141, 216, 217, 229, 234, 330, 332, 334, 335, 337, 338, 340–342, 347	<code>\glsentryfirst</code>	.. 89, 93–97, 123, 332, 333, 336
<code>\glsclearpage</code>	38	<code>\glsentryfirstaccess</code>	329
<code>\glsclosebrace</code>	45, 156, 157, 306, 307	<code>\Glsentryfirstplural</code>	90, 93, 96, 125, 331, 335
<code>\glscompositor</code>	34, 44, 158, 305, 308	<code>\glsentryfirstplural</code>	.. 90, 92, 93, 95, 96, 124, 125, 330, 331, 334, 335
<code>\glscounter</code>	15, 28, 40, 56, 78, 106, 302	<code>\glsentryfirstpluralaccess</code>	329
<code>\glscurrententrylabel</code>	175, 177, 178	<code>\glsentryfmt</code>	56, 58
<code>\glscustomtext</code>	91, 95, 97, 99, 116–121, 134– 141, 216, 217, 225, 229, 231, 232, 234, 330, 333, 334, 336, 337, 339–342, 347, 348	<code>\Glsentryfull</code>	.. 210, 218, 220, 349, 351
<code>\GlsDeclareNoHyperList</code>	15	<code>\glsentryfull</code>	.. 210, 218, 220, 348, 351
<code>\glsdefaulttype</code>	12, 36, 47, 48, 54, 55, 76, 91, 100, 101, 167, 175, 176	<code>\Glsentryfullpl</code>	.. 210, 218, 220, 349, 351
<code>\glsdefmain</code>	12, 57	<code>\glsentryfullpl</code>	.. 210, 218, 220, 349, 351
<code>\glsdescriptionaccessdisplay</code>	332–334, 342, 343	<code>\glsentryitem</code>	190, 257, 259–262, 264, 265, 273, 274, 276, 282, 283, 285, 288, 290, 292, 294, 295, 297, 299, 309–316, 318–321
<code>\glsdescriptionpluralaccessdisplay</code>	330, 331	<code>\Glsentrylong</code>	.. 89, 139, 143, 148, 212, 217, 218, 339, 341, 344, 347–349
<code>\glsdescwidth</code>	262–264, 267, 270–277, 281–284, 286–293	<code>\glsentrylong</code>	.. 89, 99, 138, 139, 143, 148, 211–221, 231, 339, 341–352

<code>\glsentrylongaccess</code>	329	<code>\Glsentryuserii</code>	130
<code>\Glsentrylongpl</code>	90,	<code>\glsentryuserii</code>	130, 131
141, 148, 212, 217, 218, 339, 344, 347–349		<code>\Glsentryuseriii</code>	131
<code>\glsentrylongpl</code>		<code>\glsentryuseriii</code>	131, 132
..... 90, 100, 140, 141, 148, 212, 213,		<code>\Glsentryuseriv</code>	132
217–220, 231, 239, 339, 344, 345, 347–351		<code>\glsentryuseriv</code>	132
<code>\glsentrylongpluralaccess</code>	329	<code>\Glsentryuserv</code>	133
<code>\Glsentryname</code>	126, 195, 342	<code>\glsentryuserv</code>	133
<code>\glsentryname</code>	125, 126, 342	<code>\Glsentryuservi</code>	134
<code>\glsentrynumberlist</code>	149, 164	<code>\glsentryuservi</code>	133, 134
<code>\Glsentryplural</code>	92, 96, 124, 331, 335	<code>\glsfirstaccessdisplay</code>	332, 333, 336
<code>\glsentryplural</code>		<code>\glsfirstpluralaccessdisplay</code>	
..... 92, 93, 95, 96, 124, 330, 331, 334, 335		330, 331, 334, 335
<code>\glsentrypluralaccess</code>	328	<code>\glsfirstpluralacesdisplay</code>	335
<code>\Glsentryprefix</code>	251	<code>\glsgenacfmt</code>	211–213, 219, 343, 344, 349
<code>\glsentryprefix</code>	249, 252	<code>\glsgenentryfmt</code>	
<code>\Glsentryprefixfirst</code>	251 211–213, 217, 219, 223, 225, 227,	
<code>\glsentryprefixfirst</code>	250, 252	229, 231, 234, 236, 239, 343, 344, 348, 349	
<code>\Glsentryprefixfirstplural</code>	252	<code>\glsgetgroupitle</code>	
<code>\glsentryprefixfirstplural</code>	250, 253	256, 259–261, 278–281, 295–298, 300, 301	
<code>\Glsentryprefixplural</code>	251	<code>\gls glossarymark</code>	36
<code>\glsentryprefixplural</code>	250, 253	<code>\gls groupheading</code>	157, 187,
<code>\glsentryprevcount</code>	86, 87	256, 259–263, 265, 273–275, 278–283,	
<code>\Glsentryshort</code>	98, 135,	285, 288, 290, 291, 294–298, 300, 301, 307	
143, 213, 219, 220, 338–340, 344, 350, 351		<code>\gls groupskip</code> 156, 187, 257, 259, 262, 264,	
<code>\glsentryshort</code>	98, 99, 134, 136, 143,	265, 268, 269, 273, 274, 276, 282, 283,	
148, 210–221, 337–340, 343–346, 348–352		285, 289, 290, 292, 294, 296, 297, 300, 306	
<code>\glsentryshortaccess</code>	329	<code>\gls hyperfirstfalse</code>	219, 349
<code>\Glsentryshortpl</code>		<code>\gls hyperfirsttrue</code>	23
..... 98, 137, 213, 219, 220, 337, 344, 350, 351		<code>\gls hyperlink</code>	175
<code>\glsentryshortpl</code>		<code>\gls hypernavsep</code>	256
..... 98, 100, 136, 137, 148, 212,		<code>\gls hypernumber</code>	35, 203
213, 218–220, 239, 337, 339, 344, 348–351		<code>\gls ifhyperon</code>	102
<code>\glsentryshortpluralaccess</code>	329	<code>\gls ifListOfAcronyms</code>	14
<code>\Glsentrysymbol</code>	128, 196, 343	<code>\gls ifplural</code>	91, 95,
<code>\glsentrysymbol</code>	93–	97, 98, 116–121, 134–141, 216, 225, 229,	
95, 128, 196, 225, 229, 234, 332–334, 343		231, 234, 330, 334, 337, 338, 340–342, 347	
<code>\glsentrysymbolaccess</code>	329	<code>\gls ifusetranslator</code>	21, 22, 31, 32, 360
<code>\Glsentrysymbolplural</code>	129	<code>\gls indexonlyfirstfalse</code>	23
<code>\glsentrysymbolplural</code>		<code>\gls inlinedescformat</code>	257, 309
..... 92, 93, 129, 225, 229, 234, 330, 331		<code>\gls inlinedopostchild</code>	257, 309
<code>\glsentrysymbolpluralaccess</code>	329	<code>\gls inlineemptydescformat</code>	257, 309
<code>\Glsentrytext</code>	94, 97, 123, 332, 336	<code>\gls inlinenameformat</code>	257, 309
<code>\glsentrytext</code>	93,	<code>\gls inlineparentchildseparator</code> 257, 309	
94, 96, 97, 122, 150, 175, 332, 333, 335, 336		<code>\gls inlinepostchild</code>	257, 309
<code>\glsentrytextaccess</code>	328	<code>\gls inlineseparator</code>	257, 309
<code>\glsentrytype</code>	76	<code>\gls inlinesubdescformat</code>	257, 309
<code>\Glsentryuseri</code>	130	<code>\gls inlinesubnameformat</code>	257, 309
<code>\glsentryuseri</code>	129, 130	<code>\gls inlinesubseparator</code>	257, 309

<code>\glsinsert</code>	92–99, 116–121, 134–141, 217, 225, 229, 231, 232, 234, 330–342, 347, 348	<code>\glsnumberformat</code>	149
<code>\glskeylisttok</code>	209, 223, 224, 226–228, 230, 232, 233, 235–237, 239, 352–357	<code>\glsnumberlistloop</code>	165
<code>\glslabel</code>	75, 92–99, 104–106, 135–141, 211–213, 216, 217, 219, 225, 229, 231, 234, 330–339, 343, 344, 347–349	<code>\glsnumbersgroupname</code>	26, 32, 199
<code>\glslabeltok</code>	209, 223–230, 232–234, 236, 237, 239, 353–356	<code>\glsnumlistlastsep</code>	149, 165
<code>\glslink</code>	209, 210, 217–220, 225, 348, 350	<code>\glsnumlistparser</code>	150, 162
<code>\glslink options</code>		<code>\glsnumlistsep</code>	149, 165
<code>counter</code>	101, 116, 246	<code>\glsopenbrace</code>	45, 156, 157, 306, 307
<code>format</code>	102, 116, 201	<code>\glsorder</code>	24, 160, 161, 183
<code>hyper</code>	102, 104, 116	<code>\glsorg@endtheglossary</code>	5
<code>local</code>	102	<code>\glsorg@PrintChanges</code>	5
<code>\glslinkcheckfirsthyperhook</code>	104	<code>\glsorg@theglossary</code>	5
<code>\glslinkpostsetkeys</code>	105	<code>\glspagelistwidth</code>	263, 264, 267, 270, 271, 274–277, 283, 284, 286, 287, 290–293
<code>\glslinkvar</code>	102, 103	<code>\glspatchLTOutput</code>	268–270
<code>\glslistdottedwidth</code>	261, 310	<code>\glspenaltygroupskip</code>	268, 269
<code>\glslocalreset</code>	85	<code>\glspercentchar</code>	66, 67, 155, 157
<code>\glslocalunset</code>	85, 117–121	<code>\Glspl</code>	90, 223
<code>\glslongaccessdisplay</code>	339, 341–353	<code>\glspl</code>	90, 223
<code>\glslongkey</code>	357	<code>\glspluralaccessdisplay</code>	330, 331, 334, 335
<code>\glslongpluralaccessdisplay</code>	339, 344, 345, 347–351, 353	<code>\glspluralsuffix</code>	30, 78, 212, 213, 344, 345, 349–351
<code>\glslongpluralkey</code>	357	<code>\glspostdescription</code>	33, 258–260, 262, 273, 282, 288, 294–297, 299, 300, 309–312, 314–318, 320
<code>\glslongtok</code>	209–213, 217, 219, 223, 224, 226–228, 230, 232, 233, 235–237, 239, 240, 343, 344, 348, 349, 352–357	<code>\glspostinline</code>	256
<code>\glsLTpenaltycheck</code>	271, 272	<code>\glspostlinkhook</code>	104, 117–122, 135–141, 340–342
<code>\glsncols</code>	278–280	<code>\glsprestandardsort</code>	10
<code>\glsnameaccessdisplay</code>	342, 343	<code>\glsreset</code>	85
<code>\glsnamefont</code>	195, 196, 322, 323, 342	<code>\glsresetentrycounter</code>	190, 193
<code>\glsnavhyperlink</code>	256	<code>\glsresetentrylist</code>	156, 185, 192, 306
<code>\glsnavhypertarget</code>	260, 261, 278–281, 295, 296, 298, 301	<code>\glsresetsubentrycounter</code>	190, 191, 194, 257, 309
<code>\glsnavigation</code>	260, 278–281, 295, 296, 298, 301	<code>\glsanitizesortfalse</code>	20
<code>\glsnextpages</code>	7, 61, 177	<code>\glsanitizesorttrue</code>	20
<code>\glsnogroupskipfalse</code>	8	<code>\glsavenumberlistfalse</code>	7
<code>\glsnoidxdisplayloc</code>	165, 166	<code>\glsavewritesfalse</code>	26
<code>\glsnoidxdisplayloclisthandler</code>	165	<code>\glsseeformat</code>	155, 163, 165, 306
<code>\glsnoidxloclist</code>	164, 186, 187	<code>\glsseeitem</code>	174
<code>\glsnoidxloclisthandler</code>	187	<code>\glsseeitemformat</code>	175
<code>\glsnoidxnumberlistloophandler</code>	165	<code>\glsseeelastsep</code>	174
<code>\glsnoidxstripaccents</code>	18	<code>\glsseeelist</code>	174
<code>\glsnonnextpages</code>	61, 177	<code>\glsseeesep</code>	174
<code>\glsnopostdotfalse</code>	8	<code>\glssetexpandfield</code>	17, 19, 20
<code>\glsnoxindywarning</code>	34, 40, 42, 43, 45–47, 153	<code>\glssetnoexpandfield</code>	17, 19, 20
		<code>\glssettoctitle</code>	32, 177
		<code>\glsshortaccessdisplay</code>	337–340, 343–346, 348–353
		<code>\glsshortkey</code>	357

link text [91](#)
\listcsadd [184](#)
\listcsgadd [188](#)
\listcsxadd [180](#)
\listead [184](#)
\loadglsentries [91](#)
\long [74](#), [203](#)
\longnewglossaryentry [75](#)
longtable package [262](#), [268](#), [272](#)
\LT@end@pen [271](#)
\LT@err [271](#)
\LT@foot [271](#), [272](#)
\LT@head [271](#), [272](#)
\LT@lastfoot [271](#)
\LT@output [271](#)

M

\makeatletter [66](#), [178](#)
\makeatother [66](#)
\makebox [261](#), [298–300](#), [310](#), [316](#), [317](#)
makeglossaries [24](#), [33](#), [47](#), [55](#), [161](#), [178](#)
\makeglossaries
..... [25](#), [29](#), [60](#), [61](#), [162](#), [164](#), [166](#), [179](#)
\makeglossary [159](#), [161](#)
makeindex [362](#)
makeindex [9](#), [24](#), [25](#), [30](#),
[33–35](#), [55](#), [57](#), [59](#), [82](#), [107](#), [110](#), [152](#), [155](#),
[157](#), [159](#), [167](#), [171](#), [172](#), [197](#), [198](#), [303](#), [304](#)
delim_n [35](#)
delim_r [35](#)
page_compositor [34](#)
special characters [108](#), [109](#), [152](#)
\makenoidxglossaries [60](#), [61](#), [162](#), [166](#)
\MakeTextUppercase [4](#)
\MakeUppercase [331](#), [333](#), [340](#), [342](#)
\markboth [37](#)
\mbox [151](#), [260](#), [280](#), [298](#), [310](#)
memoir class [168](#)
\memUchead [37](#)
\MessageBreak [32](#), [55](#), [176](#), [177](#), [360](#), [361](#)
mfirstuc package [1](#)
\mfirstucMakeUppercase
..... [4](#), [37](#), [71](#), [93–99](#), [122–134](#), [136](#),
[137](#), [139](#), [141](#), [209](#), [210](#), [217–220](#), [229](#),
[234](#), [252](#), [253](#), [335–339](#), [347](#), [348](#), [350](#), [351](#)
\midrule [268](#), [269](#)
\month [153](#), [157](#), [304](#), [307](#)
multicol package [277](#)

N

\n [157](#), [158](#), [307](#)
\NeedsTeXFormat ... [4](#), [247](#), [302](#), [308](#), [322](#), [360](#)
\new@glossaryentry [65](#), [164](#)
\new@ifnextchar [55](#), [70](#), [71](#), [89](#),
[90](#), [116–120](#), [122–141](#), [205–208](#), [249–252](#)
\newacronym [204](#),
[209](#), [224](#), [226](#), [228](#), [230](#), [232](#), [235](#), [237](#), [239](#)
\newacronymhook [209](#),
[224](#), [226](#), [228](#), [230](#), [233](#), [236](#), [237](#), [240](#), [352](#)
\newacronymstyle [211–216](#), [218–221](#)
\newcommand 5–18, [20](#), [21](#), [23–31](#), [33–75](#), [80](#),
[81](#), [83–86](#), [88–91](#), [95](#), [97](#), [99–106](#), [108](#),
[115–153](#), [158–161](#), [163](#), [166–176](#), [178–](#)
[188](#), [191–201](#), [203–211](#), [213](#), [221](#), [223–](#)
[232](#), [234–246](#), [248–252](#), [254–258](#), [271](#),
[277](#), [293](#), [298](#), [308](#), [326–328](#), [343](#), [357–359](#)
\newcount [10](#), [11](#), [63](#)
\newcounter [190](#), [192](#)
\newenvironment [194](#)
\newglossary [12](#), [13](#), [26](#), [27](#), [56](#), [161](#)
\newglossaryentry [27](#), [62](#), [65](#), [86](#), [209](#), [223](#),
[225](#), [227](#), [229](#), [232](#), [234](#), [236](#), [239](#), [353–356](#)
\newglossaryentry options
access [325](#), [326](#)
counter [60](#)
description
. [23](#), [58](#), [62](#), [65](#), [75](#), [126](#), [144](#), [205](#), [232](#), [324](#)
descriptionaccess [327](#), [329](#)
descriptionplural [127](#), [324](#)
descriptionpluralaccess [327](#), [329](#)
first . [59](#), [78](#), [116](#), [123](#), [145](#), [230](#), [235](#), [236](#), [323](#)
firstaccess [327](#), [329](#)
firstplural [59](#), [124](#), [145](#), [324](#)
firstpluralaccess [327](#), [329](#)
format [153](#)
long [97](#), [148](#), [324](#)
longaccess [328](#), [329](#)
longplural [148](#), [324](#)
longpluralaccess [328](#), [329](#)
name ... [58](#), [59](#), [62](#), [65](#), [75](#), [125](#), [142](#), [175](#), [323](#)
nonumberlist [61](#)
parent [61](#), [65](#)
plural [59](#), [78](#), [123](#), [323](#)
pluralaccess [327](#), [328](#)
prefix [247](#)
prefixfirst [247](#)
prefixfirstplural [248](#)
prefixplural [248](#)

hyperfirst	
false	116–121
indexonlyfirst	369
makeindex	155, 246
nogroupskip	268, 269
nolist	240
nolong	240, 262
nomain	12
nonumberlist	7
nosuper	240
notree	240
numberline	5
sanitize	19, 58, 142, 144
sanitizesort	16
savewrites	25, 366
false	159
true	161, 167
section	5, 37
sort	
def	9, 10
standard	9
use	9, 10
style	6, 240
subentrycounter	190, 192
toc	5
true	5
translate	21
false	21
translator	21
xindy	24, 25, 155, 246
<code>\PackageError</code>	25, 29, 40, 47, 50, 51, 55, 60, 62, 63, 69–74, 76–78, 86, 101, 142, 159, 161, 164, 166, 182–185, 189, 191, 199–201, 210, 211, 227, 228, 233, 236, 308, 330
<code>\PackageInfo</code>	159
<code>\PackageWarning</code>	15, 322
<code>\PackageWarningNoLine</code>	16, 32, 360, 361
<code>\pagegoal</code>	271
<code>\pagelistname</code>	32, 264–266, 269, 275–277, 284–287, 291–293
<code>\par</code>	33, 197, 258, 260, 277, 279–281, 293–301, 310, 315–317
<code>\parindent</code>	278–281, 294–297, 299–301, 315–317
<code>\parskip</code>	278, 279, 294, 295, 297
<code>\PassOptionsToPackage</code>	247, 322
<code>\penalty</code>	271
<code>\phantomsection</code>	38
polyglossia package	21, 31
<code>\printglossaries</code>	162
<code>\printglossary</code>	13, 16, 26, 27, 162, 176, 191
<code>\printglossary options</code>	
entrycounter	190
nogroupskip	189
nonumberlist	191
nopostdot	190
numberedsection	189
style	189
subentrycounter	190
title	189
toctitle	189
type	12, 175, 188
<code>\printindex</code>	27
<code>\printnoidxglossaries</code>	164
<code>\printnoidxglossary</code>	163, 164, 166, 176, 182–184, 191
<code>\printnoidxglossary options</code>	
sort	191
<code>\printnumbers</code>	26
<code>\printsymbols</code>	26
<code>\ProcessOptions</code>	247, 322
<code>\ProcessOptionsX</code>	27
<code>\protect</code>	39, 99, 100, 211–213, 219, 225, 229, 231, 339, 343, 344, 349, 350
<code>\protected@edef</code>	6, 41, 43, 46, 48, 77, 80, 81, 92, 94, 100, 106, 107, 149, 168, 171, 189, 195, 196, 200, 209, 233, 239, 248, 254, 303, 304, 322, 323, 328
<code>\protected@write</code>	54, 56, 153–155, 161, 163, 166, 169, 175, 254, 304
<code>\protected@xdef</code>	10, 11, 14, 18, 64, 82, 171, 325, 326
<code>\providecommand</code>	13, 29, 30, 37, 54, 88, 116, 155, 161, 163, 178, 179, 195, 196, 258, 277, 293
<code>\ProvidesFile</code>	30
<code>\ProvidesPackage</code>	4, 247, 254, 256, 258, 261, 268, 272, 277, 281, 287, 293, 302, 308, 322, 360
R	
<code>\r</code>	18
<code>\raggedright</code>	270–277, 288–293
<code>\raisebox</code>	115
<code>\ref</code>	193
<code>\refstepcounter</code>	190, 191, 193
<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, 171–174, 176,

178, 186, 189, 199, 200, 240, 255, 258,
271, 277, 280, 281, 293–301, 303, 306–
308, 314–317, 325, 340, 341, 353–355, 357

`\renewacronymstyle` . 343–347, 349, 351, 352

`\renewcommand` 4–9, 12,
13, 15, 16, 20, 22, 23, 25, 28, 31–35, 47,
58, 74, 86–88, 149, 151–153, 161–165,
168, 177–179, 189–191, 209–221, 224,
226–228, 230–233, 235–237, 239, 256,
257, 259–266, 268, 269, 271–283, 285,
288–292, 294–303, 308–316, 318–321,
325, 330, 334, 337, 339, 342–346, 348–356

`\renewenvironment` 194,
256, 259, 262–267, 270–293, 295, 297, 298

`\RequireGlossariesLang` 32, 360, 361

`\RequirePackage`
..... 4, 7, 8, 21, 22, 27, 31, 240, 246,
247, 262, 268, 272, 277, 281, 288, 323, 360

`\restorecounters@` 106

`\romannumeral` 170, 171, 298–300, 316

S

`\s@glshyp@opt` 103

`\s@newglossary` 55

`\savecounters@` 106

`\seename` 174

`\SetAcronymStyle` 23

`\setbool` 20

`\setbox` 271, 272

`\setcounter` 190, 191, 193

`\SetCustomDisplayStyle` 239, 240

`\SetDefaultAcronymDisplayStyle` 224

`\SetDefaultAcronymStyle` 238

`\SetDescriptionAcronymDisplayStyle`
..... 230, 231

`\SetDescriptionAcronymStyle` 238

`\SetDescriptionDUAAcronymDisplayStyle`
..... 228

`\SetDescriptionDUAAcronymStyle` 238

`\SetDescriptionFootnoteAcronymDisplayStyle`
..... 226, 227

`\SetDescriptionFootnoteAcronymStyle` 238

`\SetDUADisplayStyle` 237, 238

`\SetDUAStyle` 238

`\setentrycounter` 41, 155, 188, 302

`\SetFootnoteAcronymDisplayStyle` ... 233

`\SetFootnoteAcronymStyle` 238

`\SetGenericNewAcronym` 210

`\setglossarystyle` 177, 200, 240, 259–261,
263–271, 273–280, 282–287, 289–298, 300

`\setglossentrycompatibility` ... 189, 200

`\setkeys` .. 20, 25, 28, 37, 76, 105, 151, 177,
209, 224, 226, 228, 230, 233, 235, 237, 239

`\setlength` 261, 262, 272, 278,
279, 281, 288, 294–297, 299, 300, 316, 317

`\SetSmallAcronymDisplayStyle` .. 235, 236

`\SetSmallAcronymStyle` 238

`\settoheight` 115

`\settowidth` 213, 298–300, 316

`\sfcode` 8

`\show` 240–246, 358, 359

`\SmallNewAcronymDef` 236

`\space` 25, 29, 40, 41, 45, 47,
48, 60–62, 86, 89, 90, 99–102, 149, 154–
157, 160–162, 164, 166, 174, 177, 179,
190, 191, 194, 200, 206, 211–221, 229,
233, 234, 256, 258–260, 262, 273, 282,
288, 294–300, 302, 303, 305–307, 309–
312, 314–318, 320, 339, 343–346, 348–353

`\spacefactor` 8

`\SS` 19

`\ss` 19

`\string` ... 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, 173, 177–
179, 182–184, 191, 197, 200, 254, 302–308

`\strut` 197, 259–262,
264, 265, 273, 274, 276, 282, 283, 285,
288, 290, 292, 297, 309–313, 315, 318–321

`\subglossentry` 81, 177, 178, 186, 197, 257,
259–262, 264, 265, 273, 274, 276, 282,
283, 285, 288, 290, 292, 294, 295, 297, 299

`\subitem` 294, 314

`\subsubitem` 294, 314

supertabular package 7, 240, 281, 288

`\symbolname`
. 32, 266, 269, 276, 277, 285–287, 292, 293

T

`\t` 18

`\tablehead` 281–293

`\tabletail` 281–293

`\tabularnewline` 262–266, 268,
269, 273–277, 282–293, 312, 313, 318, 319

`\textbar` 256

`\textbf` 197, 203, 278–281, 293, 314–317

textcase package 4

`\textit` 203

