

Documented Code For glossaries v4.31

Nicola L.C. Talbot

Dickimaw Books

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

2017-08-10

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 v4.31: 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	32
1.4	Xindy	42
1.5	Loops and conditionals	50
1.6	Defining new glossaries	57
1.7	Defining new entries	61
1.8	Resetting and unsetting entry flags	87
1.9	Keeping Track of How Many Times an Entry Has Been Unset	90
1.10	Loading files containing glossary entries	95
1.11	Using glossary entries in the text	95
1.12	Adding an entry to the glossary without generating text	154
1.13	Creating associated files	156
1.14	Writing information to associated files	175
1.15	Glossary Entry Cross-References	181
1.16	Displaying the glossary	183
1.17	Acronyms	212
1.18	Predefined acronym styles	217
1.19	Predefined Glossary Styles	248
1.20	Debugging Commands	249
1.21	Compatibility with version 2.07 and below	254
2	Prefix Support (glossaries-prefix Code)	256
3	Glossary Styles	263
3.1	Glossary hyper-navigation definitions (glossary-hypernav package)	263
3.2	In-line Style (glossary-inline.sty)	265
3.3	List Style (glossary-list.sty)	267
3.4	Glossary Styles using longtable (the glossary-long package)	271
3.5	Glossary Styles using longtable and booktabs (the glossary-longbooktabs) package	277
3.6	Glossary Styles using longtable (the glossary-longragged package)	282
3.7	Glossary Styles using multicol (glossary-mcols.sty)	287
3.8	Glossary Styles using supertabular environment (glossary-super package)	293
3.9	Glossary Styles using supertabular environment (glossary-superragged package)	300
3.10	Tree Styles (glossary-tree.sty)	306

4 Backwards Compatibility	316
4.1 glossaries-compatible-207	316
4.2 glossaries-compatible-307	322
5 Accessibility Support (glossaries-accsupp Code)	336
5.1 Defining Replacement Text	337
5.2 Accessing Replacement Text	340
5.3 Displaying the Glossary	356
5.4 Acronyms	357
5.5 Debugging Commands	372
6 Multi-Lingual Support	374
6.1 Polyglossia Captions	374
Glossary	376
Change History	377
Index	400

1 Main Package Code

1.1 Package Definition

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

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries}[2017/08/10 v4.31 (NLCT)]
```

Required packages:

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

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

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

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

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

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

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

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

Check if doc has been loaded.

```
f@gls@docloaded
```

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

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

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

org@theglossary

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

@endtheglossary

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

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

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

End of doc stuff.

```
31 \fi
```

1.2 Package Options

debug Switch on debug mode. This will also cancel the nowarn option.

```
32 \define@boolkey{glossaries.sty}{@gls@}{debug}[true]{%
33   \if@gls@debug
34     \renewcommand*{\GlossariesWarning}[1]{%
35       \PackageWarning{glossaries}{##1}%
36     }%
37     \renewcommand*{\GlossariesWarningNoLine}[1]{%
38       \PackageWarningNoLine{glossaries}{##1}%
39     }%
40     \PackageInfo{glossaries}{debug mode ON (nowarn option disabled)}%
41   \else
42     \PackageInfo{glossaries}{debug mode OFF}%
43   \fi
44 }
```

Determine what to do if the see key is used before \makeglossaries. The default is to produce an error.

gls@see@noindex

```
45 \newcommand*{\@gls@see@noindex}{%
46   \PackageError{glossaries}{%

```

```

47 {'\gls@xr@key' key may only be used after \string\makeglossaries\space
48 or \string\makenoidxglossaries}%
49 {You must use \string\makeglossaries\space
50 or \string\makenoidxglossaries\space before defining
51 any entries that have a '\gls@xr@key' key}%
52 }

```

seenoinindex

```

53 \define@choicekey{glossaries.sty}{seenoinindex}[\val\nr]{error,warn,ignore}{%
54   \ifcase\nr
55     \renewcommand*{\@gls@see@noindex}{%
56       \PackageError{glossaries}%
57       {'\gls@xr@key' key may only be used after \string\makeglossaries\space
58       or \string\makenoidxglossaries}%
59       {You must use \string\makeglossaries\space
60       or \string\makenoidxglossaries\space before defining
61       any entries that have a '\gls@xr@key' key}%
62     }%
63   \or
64     \renewcommand*{\@gls@see@noindex}{%
65       \GlossariesWarning{'\gls@xr@key' key ignored}%
66     }%
67   \or
68     \renewcommand*{\@gls@see@noindex}{}%
69   \fi
70 }

```

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

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

```

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

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

```

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

```

73 \ifcsundef{chapter}%
74   {\newcommand*{\@@glossarysec}{section}}%
75   {\newcommand*{\@@glossarysec}{chapter}}

```

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

```

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

```

Determine whether or not to use numbered sections.

`glossarysecstar`

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

`glossaryseclabel`

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

`\glsautoprefix`

Prefix to add before label if automatically generated:

```
81 \newcommand*{\glsautoprefix}{}
```

`numberedsection`

```
82 \define@choicekey{glossaries.sty}{numberedsection}[\val\nr]{%
83 false,nolabel,autolabel,nameref}[nolabel]{%
84   \ifcase\nr\relax
85     \renewcommand*{\@@glossarysecstar}{*}%
86     \renewcommand*{\@@glossaryseclabel}{}%
87   \or
88     \renewcommand*{\@@glossarysecstar}{}%
89     \renewcommand*{\@@glossaryseclabel}{}%
90   \or
91     \renewcommand*{\@@glossarysecstar}{}%
92     \renewcommand*{\@@glossaryseclabel}{%
93       \label{\glsautoprefix\@glo@type}}}%
94   \or
95     \renewcommand*{\@@glossarysecstar}{*}%
96     \renewcommand*{\@@glossaryseclabel}{%
97       \protected@edef\@currentlabelname{\glossarytoctitle}%
98       \label{\glsautoprefix\@glo@type}}}%
99   \fi
100 }
```

The default glossary style is stored in `\@glossary@default@style`. This is initialised to `list`. (The `list` style is defined in the accompanying package described in [section 1.19](#).) Note that the `list` style is incompatible with `classicthesis` so change the default to `index` if that package has been loaded.

`y@default@style`

```
101 \ifpackageloaded{classicthesis}
102 {\newcommand*{\@glossary@default@style}{index}}
103 {\newcommand*{\@glossary@default@style}{list}}
```

`style`

The default glossary style can be changed using the `style` package option. The value can be the name of any defined glossary style. The glossary style is set at the beginning of the document, so you can still use the `style` key to set a style that is defined in another package. This package comes with some predefined styles that are defined in [section 1.19](#). This now uses `\def` instead of `\renewcommand` as `\@glossary@default@style` may have been set to `\relax`.

```
104 \define@key{glossaries.sty}{style}{%
```

```

105 \def\@glossary@default@style{#1}%
106 }

```

Each `\DeclareOptionX` needs a corresponding `\DeclareOption` so that it can be passed as a document class option, so define a command that will implement both.

`s@declareoption`

```

107 \newcommand*{\@gls@declareoption}[2]{%
108   \DeclareOptionX{#1}{#2}%
109   \DeclareOption{#1}{#2}%
110 }

```

Each entry within a given glossary will have an associated number list. By default, this refers to the page numbers on which that entry has been used, but it can also refer to any counter used in the document (such as the section or equation counters). The default number list format displays the number list “as is”:

`aryentrynumbers`

```

111 \newcommand*{\glossaryentrynumbers}[1]{#1\gls@save@numberlist{#1}}

```

`nonumberlist` Note that the entire number list for a given entry will be passed to `\glossaryentrynumbers` so any font changes will also be applied to the delimiters. The `nonumberlist` package option suppresses the number lists (this simply redefines `\glossaryentrynumbers` to ignore its argument).

```

112 \@gls@declareoption{nonumberlist}{%
113   \renewcommand*{\glossaryentrynumbers}[1]{\gls@save@numberlist{#1}}%
114 }

```

`savenumberlist`

Provide means to store the number list for entries.

```

115 \define@boolkey{glossaries.sty}[gls]{savenumberlist}[true]{}
116 \glssavenumberlistfalse

```

`eautionumberlist`

```

117 \newcommand*{\@glo@seeautonumberlist}{}

```

`eautionumberlist` Automatically activates number list for entries containing the see key.

```

118 \@gls@declareoption{seeautonumberlist}{%
119   \renewcommand*{\@glo@seeautonumberlist}{%
120     \def\@glo@prefix{\glsnextpages}%
121   }%
122 }

```

`\@gls@loadlong`

```

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

```

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

```

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

```


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

```

125 \IfFileExists{supertabular.sty}{%
126   \newcommand*{\@gls@loadsuper}{\RequirePackage{glossary-super}}}%
127   \newcommand*{\@gls@loadsuper}{}

```

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

```

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

```

`\@gls@loadlist`

```

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

```

`nolist` This option prevents from being loaded (to reduce overheads if required). Naturally, the styles defined in will not be available if this option is used. If the style is still set to list, the default must be set to `\relax`.

```

130 \@gls@declareoption{nolist}{%
131   \renewcommand*{\@gls@loadlist}{%
132     \ifdefstring{\@glossary@default@style}{list}%
133     {\let\@glossary@default@style\relax}%
134     }%
135   }%
136 }

```

`\@gls@loadtree`

```

137 \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.

```

138 \@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).

```

139 \@gls@declareoption{nostyles}{%
140   \renewcommand*{\@gls@loadlong}{}%
141   \renewcommand*{\@gls@loadsuper}{}%
142   \renewcommand*{\@gls@loadlist}{}%
143   \renewcommand*{\@gls@loadtree}{}%
144   \let\@glossary@default@style\relax
145 }

```

`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.)

```

146 \newcommand*{\glspostdescription}{%
147   \ifglsnopostdot\else.\spacefactor\sfcode'\. \fi
148 }

```

nopostdot Boolean option to suppress post description dot

```

149 \define@boolkey{glossaries.sty}[gls]{nopostdot}[true]{}
150 \glsnopostdotfalse

```

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

```

151 \define@boolkey{glossaries.sty}[gls]{nogroupskip}[true]{}
152 \glsnogroupskipfalse

```

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

```

153 \define@boolkey{glossaries.sty}[gls]{ucmark}[true]{}
154 \@ifclassloaded{memoir}
155 {%
156   \glsucmarktrue
157 }%
158 {%
159   \glsucmarkfalse
160 }

```

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`.

```

161 \define@boolkey{glossaries.sty}[gls]{entrycounter}[true]{}
162 \glsentrycounterfalse

```

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

```

163 \define@key{glossaries.sty}{counterwithin}{%
164   \renewcommand*{\@gls@counterwithin}{#1}%
165   \glsentrycountertrue
166 }

```

entrycounterwithin The default value is no parent counter:

```

167 \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`.

```

168 \define@boolkey{glossaries.sty}[gls]{subentrycounter}[true]{}
169 \glssubentrycounterfalse

```

default@sorttype Initialise default sort for `\printnoidxglossary`

```

170 \newcommand*{\@gls@default@sorttype}{standard}

```

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

```

171 \define@choicekey{glossaries.sty}{sort}{standard,def,use,none}{%
172   \renewcommand*{\@gls@default@sorttype}{#1}%
173   \csname @gls@setupsort@#1\endcsname
174 }

```

glsprestandardsort

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

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

```
175 \newcommand*{\glsprestandardsort}[3]{%
176   \glsdosanitizesort
177 }
```

glscheck@sortallowed

```
178 \newcommand*{\@gls@check@sortallowed}[1]{}
```

glssetupsort@standard

Set up the macros for default sorting.

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

Store entry information when it's defined.

```
180   \def\do@gls@storeentry{\@gls@storeentry}%
```

No count register required for standard sort.

```
181   \def\@gls@defsortcount##1{}%
```

Sort according to sort key (*\@gls@sort*) if provided otherwise sort according to the entry's name (*\@gls@name*). (First argument glossary type, second argument entry label.)

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

```
183     \ifx\@gls@sort\@gls@defaultsort
```

```
184       \let\@gls@sort\@gls@name
```

```
185     \fi
```

```
186     \let\glsdosanitizesort\@gls@sanitizesort
```

```
187     \glsprestandardsort{\@gls@sort}{##1}{##2}%
```

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

```
189   }%
```

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

```
190   \def\@gls@setsort##1{}%
```

This sort option is allowed with *\makeglossaries* and *\makenoidxglossaries*.

```
191   \let\@gls@check@sortallowed\@gobble
```

```
192 }
```

Set standard sort as the default:

```
193 \@gls@setupsort@standard
```

glsnumberfmt

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

```
194 \newcommand*{\glsnumberfmt}[1]{%
```

```
195   \ifnum#1<100000 0\fi
```

```
196   \ifnum#1<10000 0\fi
```

```
197   \ifnum#1<1000 0\fi
```

```
198   \ifnum#1<100 0\fi
```

```

199 \ifnum#1<10 0\fi
200 \number#1%
201 }

s@setupsort@def Set up the macros for order of definition sorting.
202 \newcommand*{\@gls@setupsort@def}{%
  Store entry information when it's defined.
203 \def\do@glo@storeentry{\@glo@storeentry}%
  Defined count register associated with the glossary.
204 \def\@gls@defsortcount##1{%
205 \expandafter\global
206 \expandafter\newcount\csname glossary@##1@sortcount\endcsname
207 }%
  Increment count register associated with the glossary and use as the sort key.
208 \def\@gls@defsort##1##2{%
  It may be that the sort order was changed after the glossary was defined, so check if the count
  register has been defined.
209 \ifcsundef{glossary@##1@sortcount}%
210 {\@gls@defsortcount{##1}}%
211 {}%
212 \expandafter\global\expandafter
213 \advance\csname glossary@##1@sortcount\endcsname by 1\relax
214 \expandafter\protected@xdef\csname glo@##2@sort\endcsname{%
215 \expandafter\glssortnumberfmt
216 {\csname glossary@##1@sortcount\endcsname}}%
217 }%
  Don't need to do anything when the entry is used.
218 \def\@gls@setsort##1{%
  This sort option is allowed with \makeglossaries and \makenoidxglossaries.
219 \let\@glo@check@sortallowed\@gobble
220 }

s@setupsort@use Set up the macros for order of use sorting.
221 \newcommand*{\@gls@setupsort@use}{%
  Don't store entry information when it's defined.
222 \let\do@glo@storeentry\@gobble
  Defined count register associated with the glossary.
223 \def\@gls@defsortcount##1{%
224 \expandafter\global
225 \expandafter\newcount\csname glossary@##1@sortcount\endcsname
226 }%
  Initialise the sort key to empty.
227 \def\@gls@defsort##1##2{%
228 \expandafter\gdef\csname glo@##2@sort\endcsname{%
229 }%

```

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

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

Get the parent, if one exists

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

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

```
232 \ifx\@glo@parent\@empty
```

```
233 \else
```

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

```
235 \fi
```

Set index information for this entry

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

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

```
238 \ifx\@gls@tmp\@empty
```

```
239 \expandafter\global\expandafter
```

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

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

```
242 \expandafter\glssortnumberfmt
```

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

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

```
245 \fi
```

```
246 }%
```

This sort option is allowed with `\makeglossaries` and `\makenoidxglossaries`.

```
247 \let\@glo@check@sortallowed\@gobble
```

```
248 }
```

`@setupsort@none` Slightly improves efficiency in the event that no indexing is required.

```
249 \newcommand*{\@gls@setupsort@none}{%
```

Don't store entry index information.

```
250 \def\do@glo@storeentry##1{}%
```

No count register required for standard sort.

```
251 \def\@gls@defsortcount##1{}%
```

Don't modify sort value.

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

```
253 \expandafter\global\expandafter\let\csname glo@##2@sort\endcsname\@glo@sort
```

```
254 }%
```

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

```
255 \def\@gls@setsort##1{}%
```

This sort option isn't allowed with `\makeglossaries` or `\makenoidxglossaries`.

```
256 \renewcommand\@glo@check@sortallowed[1]{\PackageError{glossaries}
```

```
257 {Option sort=none not allowed with \string##1}%
```

```
258 {(Use sort=def instead)}}%
```

```
259 }
```

`\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.)

```
260 \newcommand*{\glsdefmain}{%
261   \if@gls@docloaded
262     \newglossary[glg2]{main}{gls2}{glo2}{\glossaryname}%
263   \else
264     \newglossary{main}{gls}{glo}{\glossaryname}%
265   \fi
```

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

```
266 \newcommand*{\gls@tr@set@main@toctitle}{%
267   \translatelet{\glossarytoctitle}{Glossary}%
268 }%
269 }
```

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`

```
270 \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`

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

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

```
272 \@gls@declareoption{nomain}{%
273   \let\glsdefaulttype\relax
274   \renewcommand*{\glsdefmain}{}%
275 }
```

`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.

```
276 \define@boolkey{glossaries.sty}[gls]{acronym}[true]{%
277   \ifglsacronym
278     \renewcommand{\@gls@do@acronymsdef}{%
279       \DeclareAcronymList{acronym}%
280       \newglossary[alg]{acronym}{acr}{acn}{\acronymname}%
281       \renewcommand*{\acronymtype}{acronym}%
282     }
```

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

```

282     \newcommand*{\gls@tr@set@acronym@toctitle}{%
283         \translatelet{\glossarytoctitle}{Acronyms}%
284     }%
285 }%
286 \else
287     \let\@gls@do@acronymsdef\relax
288 \fi
289 }

```

`\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.

```

290 \AtBeginDocument{%
291     \ifglsacronym
292     \ifbool{glscompatible-3.07}%
293     {}%
294     {%
295         \providecommand*{\printacronyms}[1][1]{%
296             \printglossary[type=\acronymtype,#1]}%
297     }%
298 \fi
299 }

```

`@do@acronymsdef` Set default value

```

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

```

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

```

301 \@gls@declareoption{acronyms}{%
302     \glsacronymtrue
303     \renewcommand{\@gls@do@acronymsdef}{%
304         \DeclareAcronymList{acronym}%
305         \newglossary[alg]{acronym}{acr}{acn}{\acronymname}%
306         \renewcommand*{\acronymtype}{acronym}%

```

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

```

307     \newcommand*{\gls@tr@set@acronym@toctitle}{%
308         \translatelet{\glossarytoctitle}{Acronyms}%
309     }%
310 }%
311 }

```

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

```

312 \newcommand*{\@glsacronymlists}{}

```

`dtoacronymlists`

```

313 \newcommand*{\@addtoacronymlists}[1]{%
314     \ifx\@glsacronymlists\@empty

```

```

315   \protected@xdef\@glsacronymlists{#1}%
316   \else
317     \protected@xdef\@glsacronymlists{\@glsacronymlists,#1}%
318   \fi
319 }

```

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

```

320 \newcommand*{\DeclareAcronymList}[1]{%
321   \glsIfListOfAcronyms{#1}{\@addtoacronymlists{#1}}%
322 }

```

`\IfListOfAcronyms`

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

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

```

323 \newcommand{\glsIfListOfAcronyms}[1]{%
324   \edef\@do@gls@islistofacronyms{%
325     \noexpand\@gls@islistofacronyms{#1}{\@glsacronymlists}}%
326   \@do@gls@islistofacronyms
327 }

```

Internal command requires label and list to be expanded:

```

328 \newcommand{\@gls@islistofacronyms}[4]{%
329   \def\gls@islistofacronyms##1,#1,##2\end@gls@islistofacronyms{%
330     \def\@before{##1}\def\@after{##2}}%
331   \gls@islistofacronyms,#2,#1,\@nil\end@gls@islistofacronyms
332   \ifx\@after\@nnil

```

Not found

```

333     #4%
334   \else

```

Found

```

335     #3%
336   \fi
337 }

```

`\glsisacronymlist` Convenient boolean.

```
338 \newif\if@glsisacronymlist
```

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

```

339 \newcommand*{\chkisacronymlist}[1]{%
340   \glsIfListOfAcronyms{#1}%
341   {\@glsisacronymlisttrue}{\@glsisacronymlistfalse}%
342 }

```


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.)

```
343 \newcommand*{\SetAcronymLists}[1]{%
344   \renewcommand*{\@glsacronymlists}{#1}%
345 }
```

acronymlists

```
346 \define@key{glossaries.sty}{acronymlists}{%
347   \DeclareAcronymList{#1}%
348 }
```

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

```
349 \newcommand{\glscounter}{page}
```

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

```
350 \define@key{glossaries.sty}{counter}{%
351   \renewcommand*{\glscounter}{#1}%
352 }
```

gls@nohyperlist

```
353 \newcommand*{\@gls@nohyperlist}{}
```

lareNoHyperList

```
354 \newcommand*{\GlsDeclareNoHyperList}[1]{%
355   \ifdefempty\@gls@nohyperlist
356   {%
357     \renewcommand*{\@gls@nohyperlist}{#1}%
358   }%
359   {%
360     \appto\@gls@nohyperlist{,#1}%
361   }%
362 }
```

nohypertypes

```
363 \define@key{glossaries.sty}{nohypertypes}{%
364   \GlsDeclareNoHyperList{#1}%
365 }
```

ossariesWarning Prints a warning message.

```
366 \newcommand*{\GlossariesWarning}[1]{%
367   \PackageWarning{glossaries}{#1}%
368 }
```

esWarningNoLine Prints a warning message without the line number.

```
369 \newcommand*\GlossariesWarningNoLine}[1]{%
370   \PackageWarningNoLine{glossaries}{#1}%
371 }
```

tentrieswarning Warn user that sorting may take a long time. This is actually an informational message rather than a warning so just use \typeout.

```
372 \newcommand{\glosortentrieswarning}{%
373   \typeout{Using TeX to sort glossary entries---this may
374   take a while}%
375 }
```

nowarn Define package option to suppress warnings

```
376 \@gls@declareoption{nowarn}{%
377   \if@gls@debug
378     \GlossariesWarning{Warnings can't be suppressed in debug mode}%
379   \else
380     \renewcommand*\GlossariesWarning}[1]{}%
381     \renewcommand*\GlossariesWarningNoLine}[1]{}%
382     \renewcommand*\glosortentrieswarning}{}%
383   \fi
384 }
```

nonglossdefined Issue a warning if overriding \printglossary

```
385 \newcommand*\@gls@warnnonglossdefined){%
386   \GlossariesWarning{Overriding \string\printglossary}%
387 }
```

theglossdefined Issue a warning if overriding theglossary

```
388 \newcommand*\@gls@warnontheglossdefined){%
389   \GlossariesWarning{Overriding 'theglossary' environment}%
390 }
```

noredefwarn Suppress warning on redefinition of \printglossary

```
391 \@gls@declareoption{noredefwarn}{%
392   \renewcommand*\@gls@warnnonglossdefined){}%
393   \renewcommand*\@gls@warnontheglossdefined){}%
394 }
```

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

```
395 \newcommand*\@gls@sanitizedesc){%
396 }
```

lssetexpandfield `\glsetexpandfield{<field>}`

Sets field to always expand.

```
397 \newcommand*{\glsetexpandfield}[1]{%
398   \csdef{gls@assign@#1@field}##1##2{%
399     \@@gls@expand@field{##1}{#1}{##2}%
400   }%
401 }
```

setnoexpandfield `\glsetnoexpandfield{<field>}`

Sets field to never expand.

```
402 \newcommand*{\glsetnoexpandfield}[1]{%
403   \csdef{gls@assign@#1@field}##1##2{%
404     \@@gls@noexpand@field{##1}{#1}{##2}%
405   }%
406 }
```

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

```
407 \glsetexpandfield{type}
```

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

```
408 \glsetnoexpandfield{desc}
```

escplural@field

```
409 \glsetnoexpandfield{descplural}
```

ls@sanitizename

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

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

```
411 \glsetnoexpandfield{name}
```

@sanitizesymbol

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

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

```
413 \glsetnoexpandfield{symbol}
```

bolplural@field

```
414 \glsetnoexpandfield{symbolplural}
```

Sanitizing stuff:

ls@sanitizesort

```
415 \newcommand*{\@gls@sanitizesort}{%
416   \ifglssanitizesort
417     \@gls@sanitizesort
418   \else
419     \@gls@nosanitizesort
420   \fi
421 }
```

ls@sanitizesort

```
422 \newcommand*{\@gls@sanitizesort}{%
423   \@onelevel@sanitize\@glo@sort
424 }
```

@nosanitizesort

```
425 \newcommand*{\@gls@nosanitizesort}{}%
```

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

```
426 \newcommand*{\@gls@noidx@sanitizesort}{%
427   \ifdefvoid\@glo@sort
428   {}%
429   {%
430     \expandafter\@gls@noidx@sanitizesort\@glo@sort\gls@end@sanitizesort
431   }%
432 }
433 \def\@gls@noidx@sanitizesort#1#2\gls@end@sanitizesort{%
434   \def\@glo@sort{#1#2}%
435   \@onelevel@sanitize\@glo@sort
436 }
```

@nosanitizesort

```
437 \newcommand*{\@gls@noidx@nosanitizesort}{%
438   \ifdefvoid\@glo@sort
439   {}%
440   {%
441     \expandafter\@gls@noidx@no@sanitizesort\@glo@sort\gls@end@sanitizesort
442   }%
443 }
444 \def\@gls@noidx@no@sanitizesort#1#2\gls@end@sanitizesort{%
445   \bgroup
446     \glsnoidxstripaccents
447     \protected@xdef\@glo@sort{#1#2}%
448   \egroup
449   \let\@glo@sort\@glo@sort
450 }
```

idxstripaccents This strips accents by redefining the standard accent commands to just do their argument. (This will be localised since \glsnoidxstripaccents is used within a group.) Anything outside this standard set really shouldn't be using \makenoidxglossaries.

```

451 \newcommand*\glsnoidxstripaccents{%
452   \let\IeC\@firstofone
453   \let\''\@firstofone
454   \let\'\@firstofone
455   \let\~\@firstofone
456   \let\""\@firstofone
457   \let\u\@firstofone
458   \let\t\@firstofone
459   \let\d\@firstofone
460   \let\r\@firstofone
461   \let\=\@firstofone
462   \let\.\@firstofone
463   \let\~\@firstofone
464   \let\v\@firstofone
465   \let\H\@firstofone
466   \let\c\@firstofone
467   \let\b\@firstofone

468   \let\a\@secondoftwo
469   \def\AE{AE}%
470   \def\ae{ae}%
471   \def\OE{OE}%
472   \def\oe{oe}%
473   \def\AA{AA}%
474   \def\aa{aa}%
475   \def\L{L}%
476   \def\l{l}%
477   \def\O{O}%
478   \def\o{o}%
479   \def\SS{SS}%
480   \def\ss{ss}%
481   \def\th{th}%

482   \def\TH{TH}%
483   \def\dh{dh}%
484   \def\DH{DH}%
485 }

```

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.

```

486 \define@boolkey[gls]{sanitize}{description}[true]{%
487   \GlossariesWarning{sanitize={description} package option deprecated}%
488   \ifgls@sanitize@description
489     \glssetnoexpandfield{desc}%
490     \glssetnoexpandfield{descplural}%
491   \else
492     \glssetexpandfield{desc}%
493     \glssetexpandfield{descplural}%
494   \fi
495 }

```

```

496 \define@boolkey[glS]{sanitize}{name}[true]{%
497   \GlossariesWarning{sanitize={name} package option deprecated}%
498   \ifglS@sanitize@name
499     \glSsetnoexpandfield{name}%
500   \else
501     \glSsetexpandfield{name}%
502   \fi
503 }

504 \define@boolkey[glS]{sanitize}{symbol}[true]{%
505   \GlossariesWarning{sanitize={symbol} package option deprecated}%
506   \ifglS@sanitize@symbol
507     \glSsetnoexpandfield{symbol}%
508     \glSsetnoexpandfield{symbolplural}%
509   \else
510     \glSsetexpandfield{symbol}%
511     \glSsetexpandfield{symbolplural}%
512   \fi
513 }

```

sanitizesort

```

514 \define@boolkey{glossaries.sty}[glS]{sanitizesort}[true]{%
515   \ifglssanitizesort
516     \glSsetnoexpandfield{sortvalue}%
517     \renewcommand*{\@glS@noidx@setsanitizesort}{%
518       \glssanitizesorttrue
519       \glSsetnoexpandfield{sortvalue}%
520     }%
521   \else
522     \glSsetexpandfield{sortvalue}%
523     \renewcommand*{\@glS@noidx@setsanitizesort}{%
524       \glssanitizesortfalse
525       \glSsetexpandfield{sortvalue}%
526     }%
527   \fi
528 }

```

Default setting:

```

529 \glssanitizesorttrue
530 \glSsetnoexpandfield{sortvalue}%

```

setsanitizesort Default behaviour for \makenoidxglossaries is sanitizesort=false.

```

531 \newcommand*{\@glS@noidx@setsanitizesort}{%
532   \glssanitizesortfalse
533   \glSsetexpandfield{sortvalue}%
534 }

535 \define@choicekey[glS]{sanitize}{sort}{true,false}[true]{%
536   \setbool{glssanitizesort}{#1}%
537   \ifglssanitizesort

```

```

538   \glsssetnoexpandfield{sortvalue}%
539 \else
540   \glsssetexpandfield{sortvalue}%
541 \fi
542 \GlossariesWarning{sanitize={sort} package option
543   deprecated. Use sanitizesort instead}%
544 }

```

sanitize

```

545 \define@key{glossaries.sty}{sanitize}[description=true,symbol=true,name=true]{%
546   \ifthenelse{\equal{#1}{none}}{%
547     {%
548       \GlossariesWarning{sanitize package option deprecated}%
549       \glsssetexpandfield{name}%
550       \glsssetexpandfield{symbol}%
551       \glsssetexpandfield{symbolplural}%
552       \glsssetexpandfield{desc}%
553       \glsssetexpandfield{descplural}%
554     }%
555   }%
556   \setkeys{gls}{sanitize}{#1}%
557 }%
558 }

```

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

```

559 \newif\ifglstranslate

```

otranslatorhook \@gls@notranslatorhook has been removed.

s@usetranslator

```

560 \newcommand*\@gls@usetranslator{%
  polyglossia tricks \@ifpackageloaded into thinking that babel has been loaded, so check for
  polyglossia as well.
561   \@ifpackageloaded{polyglossia}%
562   {%
563     \let\glsifusetranslator\@secondoftwo
564   }%
565   {%
566     \@ifpackageloaded{babel}%
567     {%
568       \IfFileExists{translator.sty}%
569       {%
570         \RequirePackage{translator}%
571         \let\glsifusetranslator\@firstoftwo
572       }%
573     }%
574   }%

```

```

575     {}%
576   }%
577 }

```

dtranslatordict Checks if given translator dictionary has been loaded.

```

578 \newcommand{\glsifusedtranslator}{3}{%
579   \glsifusetranslator
580   {\ifcsdef{ver@glossaries-dictionary-#1.dict}{#2}{#3}}%
581   {#3}%
582 }

```

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

```

583 \@gls@declareoption{notranslate}{%
584   \glstranslatefalse
585   \let\@gls@usetranslator\relax
586   \let\glsifusetranslator\@secondoftwo
587 }

```

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

```

588 \define@choicekey{glossaries.sty}{translate}[\val\nr]%
589 {true,false,babel}[true]%
590 {%
591   \ifcase\nr\relax
592     \glstranslatetrue
593     \renewcommand*\@gls@usetranslator{%
594       \@ifpackageloaded{polyglossia}%
595       {%
596         \let\glsifusetranslator\@secondoftwo
597       }%
598       {%
599         \@ifpackageloaded{babel}%
600         {%
601           \IfFileExists{translator.sty}%
602           {%
603             \RequirePackage{translator}%
604             \let\glsifusetranslator\@firstoftwo
605           }%
606           {}%
607         }%
608         {}%
609       }%
610     }%
611   \or
612     \glstranslatefalse
613     \let\@gls@usetranslator\relax
614     \let\glsifusetranslator\@secondoftwo
615   \or
616     \glstranslatetrue
617     \let\@gls@usetranslator\relax

```



```

618     \let\glsifusetranslator\@secondoftwo
619     \fi
620 }

```

Set the default value:

```

621 \glstranslatefalse
622 \let\glsifusetranslator\@secondoftwo
623 \@ifpackageloaded{translator}%
624 {%
625     \glstranslatetrue
626     \let\glsifusetranslator\@firstoftwo
627 }%
628 {%
629     \@for\gls@thissty:=tracklang,babel,ngerman,polyglossia\do
630     {
631         \@ifpackageloaded{\gls@thissty}%
632         {%
633             \glstranslatetrue
634             \@endfortrue
635         }%
636     }%
637 }
638 }

```

indexonlyfirst Set whether to only index on first use.

```

639 \define@boolkey{glossaries.sty}[gls]{indexonlyfirst}[true]{}
640 \glsindexonlyfirstfalse

```

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

```

641 \define@boolkey{glossaries.sty}[gls]{hyperfirst}[true]{}
642 \glshyperfirsttrue

```

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

```

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

```

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

```

644 \define@boolkey{glossaries.sty}[glsacr]{footnote}[true]{}
645 \ifbool{glsacrdescription}%
646 {}%
647 {%
648     \renewcommand*{\@gls@sanitizedesc}{}%
649 }%
650 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
651 }

```

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

```

652 \define@boolkey{glossaries.sty}[glsacr]{description}[true]{}

```

```

653 \renewcommand*{\@gls@sanitizesymbol}{}%
654 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
655 }

```

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

```

656 \define@boolkey{glossaries.sty}[glsacr]{smallcaps}[true]{%
657 \renewcommand*{\@gls@sanitizesymbol}{}%
658 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
659 }

```

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

```

660 \define@boolkey{glossaries.sty}[glsacr]{smaller}[true]{%
661 \renewcommand*{\@gls@sanitizesymbol}{}%
662 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
663 }

```

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

```

664 \define@boolkey{glossaries.sty}[glsacr]{dua}[true]{%
665 \renewcommand*{\@gls@sanitizesymbol}{}%
666 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
667 }

```

shortcuts Define acronym shortcuts.

```

668 \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.

```

669 \newcommand*{\glsorder}{word}

```

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

```

670 \newcommand*{\@glsorder}[1]{%

```

order

```

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

```

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

```

673 \newif\ifglxsindy

```

The default is `makeindex`:

```

674 \glxsindyfalse

```

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

```

675 \@gls@declareoption{makeindex}{\glxsindyfalse}

```

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.

```
676 \define@boolkey[gls]{xindy}{glsnumbers}[true]{}
677 \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.)

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

Define key to set the language

```
679 \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.

```
680 \ifcsundef{inputencodingname}{%
681   \def\gls@codepage{}}{%
682   \def\gls@codepage{\inputencodingname}
683 }
```

Define a key to set the code page.

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

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

```
685 \define@key{glossaries.sty}{xindy}[]{%
686   \glsxindytrue
687   \setkeys[gls]{xindy}{#1}%
688 }
```

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

```
689 \@gls@declareoption{xindygloss}{%
690   \glsxindytrue
691 }
```

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

```
692 \@gls@declareoption{xindynoglsnumbers}{%
693   \glsxindytrue
694   \gls@xindy@glsnumbersfalse
695 }
```

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

```
696 \define@boolkey{glossaries.sty}[gls]{automake}[true]{%
697   \ifglsautomake
698     \renewcommand*{\@gls@doautomake}{%
699       \PackageError{glossaries}{You must use
```

```

700     \string\makeglossaries\space with automake=true}
701     {%
702         Either remove the automake=true setting or
703         add \string\makeglossaries\space to your document preamble.%
704     }%
705     }%
706 \else
707     \renewcommand*{\@gls@doautomake}{}%
708 \fi
709 }
710 \glsautomakefalse

```

@gls@doautomake

```

711 \newcommand*{\@gls@doautomake}{}
712 \AtEndDocument{\@gls@doautomake}

```

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

```

713 \define@boolkey{glossaries.sty}[gls]{savewrites}[true]{%
714     \ifglssavewrites
715         \renewcommand*{\glswritefiles}{\@glswritefiles}%
716     \else
717         \let\glswritefiles\@empty
718     \fi
719 }

```

Set default:

```

720 \glssavewritesfalse
721 \let\glswritefiles\@empty

```

compatible-3.07

```

722 \define@boolkey{glossaries.sty}[gls]{compatible-3.07}[true]{%
723 \boolfalse{glscompatible-3.07}

```

compatible-2.07

```

724 \define@boolkey{glossaries.sty}[gls]{compatible-2.07}[true]{%
    Also set 3.07 compatibility if this option is set.
725     \ifbool{glscompatible-2.07}%
726     {%
727         \booltrue{glscompatible-3.07}%
728     }%
729     {}%
730 }
731 \boolfalse{glscompatible-2.07}

```

symbols Create a “symbols” glossary type

```

732 \@gls@declareoption{symbols}{}%
733 \let\@gls@do@symbolsdef\@gls@symbolsdef
734 }

```

Default is not to define the symbols glossary:

```
735 \newcommand*{\@gls@do@symbolsdef}{}
```

@gls@symbolsdef

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

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

```
739   \newcommand*{\gls@tr@set@symbols@toctitle}{%
740     \translatelet{\glossarytoctitle}{Symbols (glossaries)}%
741   }%
742 }%
```

numbers Create a “symbols” glossary type

```
743 \@gls@declareoption{numbers}{%
744   \let\@gls@do@numbersdef\@gls@numbersdef
745 }
```

Default is not to define the numbers glossary:

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

@gls@numbersdef

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

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

```
750   \newcommand*{\gls@tr@set@numbers@toctitle}{%
751     \translatelet{\glossarytoctitle}{Numbers (glossaries)}%
752   }%
753 }%
```

index Create an “index” glossary type

```
754 \@gls@declareoption{index}{%
755   \let\@gls@do@indexdef\@gls@indexdef
756 }
```

Default is not to define index glossary:

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

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

```
758 \newcommand*{\@gls@indexdef}{%
759   \newglossary[ilg]{index}{ind}{idx}{\indexname}%
760   \newcommand*{\printindex}[1][\printglossary[type=index,##1]]%
761   \newcommand*{\newterm}[2][\%
762     \newglossaryentry{##2}%
763     {type={index},name={##2},description={\nopostdesc},##1}}
764 }%
```

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

```

765 \@for\CurrentOption := \@declaredoptions\do{%
766   \ifx\CurrentOption\@empty
767   \else
768     \@expandtwoargs
769     \in@ {,\CurrentOption ,}{,\@classoptionslist,\@curroptions,}%
770     \ifin@
771     \@use@option
772     \expandafter \let\csname ds@\CurrentOption\endcsname\@empty
773   \fi
774 \fi
775 }

```

Now process options passed to the package:

```

776 \ProcessOptionsX

```

Load backward compatibility stuff:

```

777 \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:

```

778 \disable@keys{glossaries.sty}{compatible-2.07,%
779 xindy,xindygloss,xindynoglsnumbers,makeindex,%
780 acronym,translate,notranslate,nolong,nosuper,notree,nostyles,nomain}

```

Now define `\setupglossaries`:

```

781 \newcommand*{\setupglossaries}[1]{%
782   \renewcommand*{\@gls@setacrstyle}{}%
783   \ifglsacrshortcuts
784     \def\@gls@setupshortcuts{\glsacrshortcutstrue}%
785   \else
786     \def\@gls@setupshortcuts{%
787       \ifglsacrshortcuts
788         \DefineAcronymSynonyms
789       \fi
790     }%
791   \fi
792   \glsacrshortcutsfalse
793   \let\@gls@do@numbersdef\relax
794   \let\@gls@do@symbolssdef\relax
795   \let\@gls@do@indexdef\relax
796   \let\@gls@do@acronymsdef\relax
797   \setkeys{glossaries.sty}{#1}%
798   \@gls@setacrstyle
799   \@gls@setupshortcuts
800   \@gls@do@acronymsdef
801   \@gls@do@numbersdef
802   \@gls@do@symbolssdef
803   \@gls@do@indexdef

```

804 }

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

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

```
805 \ifthenelse{\equal{\glscounter}{section}}{%
806 {%
807   \ifcsundef{chapter}{}%
808   {%
809     \let\@gls@old@chapter\@chapter
810     \def\@chapter[#1]#2{\@gls@old@chapter[#1]{#2}%
811       \ifcsundef{hyperdef}{\hyperdef{section}{\thesection}}}%
812   }%
813 }%
814 }
```

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

```
815 \newcommand*{\@gls@onlypremakeg}{}
```

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

```
816 \newcommand*{\@onlypremakeg}[1]{%
817   \ifx\@gls@onlypremakeg\@empty
818     \def\@gls@onlypremakeg{#1}%
819   \else
820     \expandafter\toks@\expandafter{\@gls@onlypremakeg}%
821     \edef\@gls@onlypremakeg{\the\toks@,\noexpand#1}%
822   \fi
823 }
```

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

```
824 \newcommand*{\@disable@onlypremakeg}{%
825   \for\@thiscs:=\@gls@onlypremakeg\do{%
826     \expandafter\@disable@premakecs\@thiscs%
827   }}
```

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

```
828 \newcommand*{\@disable@premakecs}[1]{%
829   \def#1{\PackageError{glossaries}{\string#1\space may only be
830     used before \string\makeglossaries}{You can't use
831     \string#1\space after \string\makeglossaries}}%
832 }
```

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

```
833 \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

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

\glstocctitle Sets the TOC title for the given glossary.

```
835 \newcommand*\glstocctitle[1]{%
```

```
836 \def\glossarytocctitle{\csname @glotype@#1@title\endcsname}}
```

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

\entryname

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

descriptionname

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

\symbolname

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

\pagelistname

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

Labels for makeindex's symbol and number groups:

ymbolsgroupname

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

umbersgroupname

```
842 \providecommand*\glsnumbersgroupname{Numbers}
```

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

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

acrpluralsuffix Default plural suffix for acronyms

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


acrpluralsuffix

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

\seename

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

\andname

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

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

eGlossariesLang

```
848 \newcommand*{\RequireGlossariesLang}[1]{%
849   \@ifundefined{ver@glossaries-#1.ldf}{\input{glossaries-#1.ldf}}{}%
850 }
```

sGlossariesLang

```
851 \newcommand*{\ProvidesGlossariesLang}[1]{%
852   \ProvidesFile{glossaries-#1.ldf}%
853 }
```

ssarytocaptions

Does nothing if translator hasn't been loaded.

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

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

```
855 \ifglstranslate
```

Load tracklang

```
856 \RequirePackage{tracklang}
```

Load translator if required.

```
857 \@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.)

```
858 \@ifpackageloaded{translator}
859 {%
```

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.

```
860   \ifboolexpr
861   {
862     test {\ifdefstring{\trans@languages}{English}}
863     and not
```

```

864     test {\ifdefstring{bbl@loaded}{english}}
865   }
866   {%
867     \let\glsifusetranslator\@secondoftwo
868   }%
869   {%
870     \usedictionary{glossaries-dictionary}%
871     \renewcommand*{\addglossarytocaptions}[1]{%
872       \ifcsundef{captions#1}{}%
873       {%
874         \expandafter\let\expandafter\@gls@tmp\csname captions#1\endcsname
875         \expandafter\toks@\expandafter{\@gls@tmp
876           \renewcommand*{\glossaryname}{\translate{Glossary}}}%
877       }%
878       \expandafter\edef\csname captions#1\endcsname{\the\toks@}%
879     }%
880   }%
881 }%
882 }%
883 {}%

```

Check for tracked languages

```

884 \AnyTrackedLanguages
885 {%
886   \ForEachTrackedDialect{\this@dialect}{%
887     \IfTrackedLanguageFileExists{\this@dialect}%
888     {glossaries-}% prefix
889     {.ldf}%
890     {%
891       \RequireGlossariesLang{\CurrentTrackedTag}%
892     }%
893     {%
894       \PackageWarningNoLine{glossaries}%
895       {No language module detected for ‘\this@dialect’.\MessageBreak
896       Language modules need to be installed separately.\MessageBreak
897       Please check on CTAN for a bundle called\MessageBreak
898       ‘glossaries-\CurrentTrackedLanguage’ or similar}%
899     }%
900   }%
901 }%
902 {}%

```

if using translator use translator interface.

```

903 \glsifusetranslator
904 {%
905   \renewcommand*{\glssettoctitle}[1]{%
906     \ifcsdef{gls@tr@set@#1@toctitle}%
907     {%
908       \csuse{gls@tr@set@#1@toctitle}%
909     }%

```

```

910      {%
911      \def\glossarytoctitle{\csname @glotype@#1@title\endcsname}%
912      }%
913  }%
914  \renewcommand*\glossaryname{\translate{Glossary}}%
915  \renewcommand*\acronymname{\translate{Acronyms}}%
916  \renewcommand*\entryname{\translate{Notation (glossaries)}}%
917  \renewcommand*\descriptionname{%
918    \translate{Description (glossaries)}}%
919  \renewcommand*\symbolname{\translate{Symbol (glossaries)}}%
920  \renewcommand*\pagelistname{%
921    \translate{Page List (glossaries)}}%
922  \renewcommand*\glssymbolsgroupname{%
923    \translate{Symbols (glossaries)}}%
924  \renewcommand*\glsnumbersgroupname{%
925    \translate{Numbers (glossaries)}}%
926  }{}%
927 \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.

```
928 \DeclareRobustCommand*\nopostdesc{}
```

`\@nopostdesc` Suppress next description terminator.

```

929 \newcommand*\@nopostdesc{%
930   \let\org@glspostdescription\glspostdescription
931   \def\glspostdescription{%
932     \let\glspostdescription\org@glspostdescription}%
933 }

```

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

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

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

```
935 \newcommand{\glspar}{\par}
```

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

```

936 \newcommand{\setStyleFile}[1]{%
937   \renewcommand*\gls@istfilebase{#1}%
   Just in case \istfilename has been modified.
938   \ifglsxindy
939     \def\istfilename{\gls@istfilebase.xdy}
940   \else
941     \def\istfilename{\gls@istfilebase.ist}
942   \fi
943 }

```

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

```
944 \@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`

```
945 \ifglsxindy
946   \def\istfilename{\gls@istfilebase.xdy}
947 \else
948   \def\istfilename{\gls@istfilebase.ist}
949 \fi
```

`gls@istfilebase`

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

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

`\@istfilename`

```
951 \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`

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

`lsSetCompositor` Sets the compositor.

```
953 \newcommand*{\glsSetCompositor}[1]{%
954   \renewcommand*{\glscompositor}{#1}}
```

Only use before `\makeglossaries`

```
955 \@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.

`Alphacompositor`

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.

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

AlphaCompositor Sets the alpha compositor.

```
957 \ifglxindy
958   \newcommand*\glsSetAlphaCompositor[1]{%
959     \renewcommand*\@glsAlphacompositor{#1}}
960 \else
961   \newcommand*\glsSetAlphaCompositor[1]{%
962     \glxnoindywarning\glsSetAlphaCompositor}
963 \fi
```

Can only be used before `\makeglossaries`

```
964 \@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.

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

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

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

Only has an effect when used before `\makeglossaries`

```
968 \@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.

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

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

```
970 \newcommand*\glsSetSuffixFF[1]{%
971   \renewcommand*\gls@suffixFF{#1}%
972 }
```

glsnumberformat The command `\glsnumberformat` indicates the default format for the page numbers in the glossary. (Note that this is not the same as `\glossaryentrynumbers`, but applies to individual numbers or groups of numbers within an entry's associated number list.) If hyperlinks are defined, it will use `\glshypernumber`, otherwise it will simply display its argument "as is".

```
973 \ifcsundef{hyperlink}%
974 {%
975   \newcommand*\glsnumberformat[1]{#1}%
976 }%
977 {%
978   \newcommand*\glsnumberformat[1]{\glshypernumber{#1}}%
979 }
```

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`

```
980 \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`

```
981 \newcommand{\delimR}{--}
```

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

`\glossarypreamble`

```
982 \newcommand*{\glossarypreamble}{%
983   \csuse{@glossarypreamble@\currentglossary}%
984 }
```

`\setglossarypreamble`

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

Code provided by Michael Pock.

```
985 \newcommand{\setglossarypreamble}[2][\glsdefaultttype]{%
986   \ifglossaryexists{#1}{%
987     \csgdef{@glossarypreamble@#1}{#2}%
988   }{%
989     \GlossariesWarning{%
990       Glossary ‘#1’ is not defined%
991     }%
992   }%
993 }
```

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{}}
```

`\glossarypostamble`

```
994 \newcommand*{\glossarypostamble}{}
```

`\glossarysection` The sectioning command that starts a glossary is given by `\glossarysection`. (This does not form part of the glossary style, and so should not be changed by a glossary style.) If `\phantomsection` is defined, it uses `\p@glossarysection`, otherwise it uses `\@glossarysection`.

```

995 \newcommand*{\glossarysection}[2][\@gls@title]{%
996   \def\@gls@title{#2}%
997   \ifcsundef{phantomsection}%
998   {%
999     \@glossarysection{#1}{#2}%
1000   }%
1001   {%
1002     \@p@glossarysection{#1}{#2}%
1003   }%

1004   \glsglossarymark{\glossarytoctitle}%
1005 }

```

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

```

1006 \ifcsundef{glossarymark}%
1007 {%
1008   \newcommand{\glsglossarymark}[1]{\glossarymark{#1}}
1009 }%
1010 {%
1011   \@ifclassloaded{memoir}
1012   {%
1013     \newcommand{\glsglossarymark}[1]{%
1014       \ifglsucmark
1015         \markboth{\memUHead{#1}}{\memUHead{#1}}%
1016       \else
1017         \markboth{#1}{#1}%
1018       \fi
1019     }
1020   }%
1021   {%
1022     \newcommand{\glsglossarymark}[1]{%
1023       \ifglsucmark
1024         \@mkboth{\mfirstucMakeUppercase{#1}}{\mfirstucMakeUppercase{#1}}%
1025       \else
1026         \@mkboth{#1}{#1}%
1027       \fi
1028     }
1029   }
1030 }

```

`\glossarymark` Provided for backward compatibility:

```

1031 \providecommand{\glossarymark}[1]{%
1032   \ifglsucmark
1033     \@mkboth{\mfirstucMakeUppercase{#1}}{\mfirstucMakeUppercase{#1}}%
1034   \else

```

```

1035 \mkboth{#1}{#1}%
1036 \fi
1037 }

```

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`

```

1038 \newcommand*\setglossarysection[1]{%
1039 \setkeys{glossaries.sty}{section=#1}}

```

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

`glossarysection`

```

1040 \newcommand*\@glossarysection[2]{%
1041 \ifdefempty\@glossarysecstar
1042 {%
1043 \csname\@glossarysec\endcsname[#1]{#2}%
1044 }%
1045 {%
1046 \csname\@glossarysec\endcsname*{#2}%
1047 \@gls@toc{#1}{\@glossarysec}%
1048 }%

```

Do automatic labelling if required

```

1049 \@glossaryseclabel
1050 }

```

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`

```

1051 \newcommand*\@pglossarysection[2]{%
1052 \glsclearpage
1053 \phantomsection
1054 \ifdefempty\@glossarysecstar
1055 {%
1056 \csname\@glossarysec\endcsname{#2}%
1057 }%
1058 {%
1059 \@gls@toc{#1}{\@glossarysec}%
1060 \csname\@glossarysec\endcsname*{#2}%
1061 }%

```

Do automatic labelling if required

```

1062 \@glossaryseclabel
1063 }

```


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

```

1064 \newcommand*{\gls@doclearpage}{%
1065   \ifthenelse{\equal{\@glossarysec}{chapter}}{%
1066     {%
1067       \ifcsundef{cleardoublepage}%
1068       {%
1069         \clearpage
1070       }%
1071     }%
1072     \ifcsdef{if@openright}%
1073     {%
1074       \if@openright
1075         \cleardoublepage
1076       \else
1077         \clearpage
1078       \fi
1079     }%
1080   }%
1081   \cleardoublepage
1082 }%
1083 }%
1084 }%
1085 {}%
1086 }

```

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

```

1087 \newcommand*{\glscclearpage}{\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`

```

1088 \newcommand*{\@gls@toc}[2]{%
1089   \ifglstoc
1090     \ifglslnumberline
1091       \addcontentsline{toc}{#2}{\protect\numberline{#1}}%
1092     \else
1093       \addcontentsline{toc}{#2}{#1}%
1094     \fi
1095   \fi
1096 }

```

1.4 Xindy

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

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

```
1097 \newcommand*{\glsnoxindywarning}[1]{%
1098   \GlossariesWarning{Not in xindy mode --- ignoring \string#1}%
1099 }
```

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

```
1100 \newcommand*{\glsnoindexwarning}[1]{%
1101   \GlossariesWarning{Not in makeindex mode --- ignoring \string#1}%
1102 }
```

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

```
1103 \ifglsxindy
1104   \edef\@xdyattributes{\string"default\string"}%
1105 \fi
```

\@xdyattributelist Comma-separated list of attributes.

```
1106 \ifglsxindy
1107   \edef\@xdyattributelist{}%
1108 \fi
```

\@xdylocref Define list of markup location references.

```
1109 \ifglsxindy
1110   \def\@xdylocref{}
1111 \fi
```

\@gls@ifinlist

```
1112 \newcommand*{\@gls@ifinlist}[4]{%
1113   \def\@do@ifinlist##1,#1,##2\end@do@ifinlist{%
1114     \def\@gls@listsuffix{##2}%
1115     \ifx\@gls@listsuffix\@empty
1116       #4%
1117     \else
1118       #3%
1119     \fi
1120   }%
1121   \@do@ifinlist,#2,#1,\end@do@ifinlist
1122 }
```

\@xdyAddXdyCounters 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.

```
1123 \ifglsxindy
1124   \newcommand*{\@xdycounters}{\@glscounter}
```

```

1125 \newcommand*\GlsAddXdyCounters[1]{%
1126   \@for\@gls@ctr:=#1\do{%
    Check if already in list before adding.
1127     \edef\@do@addcounter{%
1128       \noexpand\@gls@ifinlist{\@gls@ctr}{\@xdycounters}{}%
1129       {%
1130         \noexpand\edef\noexpand\@xdycounters{\@xdycounters,%
1131           \noexpand\@gls@ctr}%
1132       }%
1133     }%
1134     \@do@addcounter
1135   }
1136 }

    Only has an effect before \writeist:
1137 \@onlypremakeg\GlsAddXdyCounters
1138 \else
1139 \newcommand*\GlsAddXdyCounters[1]{%
1140   \glsnoxindywarning\GlsAddXdyAttribute
1141 }
1142 \fi

```

saddxdycounters Counters must all be identified before adding attributes.

```

1143 \newcommand*\@disabled@glssaddxdycounters{%
1144   \PackageError{glossaries}{\string\GlsAddXdyCounters\space
1145     can't be used after \string\GlsAddXdyAttribute}{Move all
1146     occurrences of \string\GlsAddXdyCounters\space before the first
1147     instance of \string\GlsAddXdyAttribute}%
1148 }

```

AddXdyAttribute Adds an attribute.

```

1149 \ifglxsindy

    First define internal command that adds an attribute for a given counter (2nd argument is
    the counter):
1150 \newcommand*\@glssaddxdyattribute[2]{%

    Add to xindy attribute list
1151   \edef\@xdyattributes{\@xdyattributes ^^J \string"#1\string" ^^J
1152     \string"#2#1\string"}%

    Add to xindy markup location.
1153   \expandafter\toks@\expandafter{\@xdylocref}%
1154   \edef\@xdylocref{\the\toks@ ^^J%
1155     (markup-locref
1156     :open \string"\glstildechar n%
1157       \expandafter\string\csname glsX#2X#1\endcsname
1158       \string" ^^J
1159     :close \string"\string" ^^J
1160     :attr \string"#2#1\string"))%

```

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

```
1161 \expandafter\gdef\csname glsX#2X#1\endcsname##1##2{%
1162 \setentrycounter{##1}{#2}\csname #1\endcsname{##2}%
1163 }%
1164 }
```

High-level command:

```
1165 \newcommand*\GlsAddXdyAttribute[1]{%
```

Add to comma-separated attribute list

```
1166 \ifx\@xdyattributelist\@empty
1167 \edef\@xdyattributelist{#1}%
1168 \else
1169 \edef\@xdyattributelist{\@xdyattributelist,#1}%
1170 \fi
```

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

```
1171 \@for\@this@counter:=\@xdycounters\do{%
1172 \protected@edef\gls@do@addxdyattribute{%
1173 \noexpand\@glsaddxdyattribute{#1}{\@this@counter}%
1174 }
1175 \gls@do@addxdyattribute
1176 }%
```

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

```
1177 \let\GlsAddXdyCounters\@disabled@glsaddxdycounters
1178 }
```

Only has an effect before `\writeist`:

```
1179 \@onlypremakeg\GlsAddXdyAttribute
1180 \else
1181 \newcommand*\GlsAddXdyAttribute[1]{%
1182 \glsnoxindywarning\GlsAddXdyAttribute}
1183 \fi
```

`\definedattributes` Add known attributes for all defined counters

```
1184 \ifglxindy
1185 \newcommand*\@gls@addpredefinedattributes{%
1186 \GlsAddXdyAttribute{glsnumberformat}
1187 \GlsAddXdyAttribute{textrm}
1188 \GlsAddXdyAttribute{textsf}
1189 \GlsAddXdyAttribute{texttt}
1190 \GlsAddXdyAttribute{textbf}
1191 \GlsAddXdyAttribute{textmd}
1192 \GlsAddXdyAttribute{textit}
1193 \GlsAddXdyAttribute{textup}
1194 \GlsAddXdyAttribute{textsl}
1195 \GlsAddXdyAttribute{textsc}
1196 \GlsAddXdyAttribute{emph}
1197 \GlsAddXdyAttribute{glshypernumber}
1198 \GlsAddXdyAttribute{hyperrm}
```

```

1199 \GlsAddXdyAttribute{hypersf}
1200 \GlsAddXdyAttribute{hypertt}
1201 \GlsAddXdyAttribute{hyperbf}
1202 \GlsAddXdyAttribute{hypermd}
1203 \GlsAddXdyAttribute{hyperit}
1204 \GlsAddXdyAttribute{hyperup}
1205 \GlsAddXdyAttribute{hypersl}
1206 \GlsAddXdyAttribute{hypersc}
1207 \GlsAddXdyAttribute{hyperemph}

1208 \GlsAddXdyAttribute{glsignore}
1209 }
1210 \else
1211 \let\@gls@addpredefinedattributes\relax
1212 \fi

```

`\dyuseralphabets` List of additional alphabets

```
1213 \def\@xdyuseralphabets{}
```

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

```

1214 \ifglsxindy
1215 \newcommand*\@GlsAddXdyAlphabet[2]{%
1216 \edef\@xdyuseralphabets{%
1217 \@xdyuseralphabets ^^J
1218 (define-alphabet "#1" (#2))}}
1219 \else
1220 \newcommand*\@GlsAddXdyAlphabet[2]{%
1221 \glsnoxindywarning\GlsAddXdyAlphabet}
1222 \fi

```

This code is only required for xindy:

```
1223 \ifglsxindy
```

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

```

1224 \newcommand*\@gls@xdy@locationlist{%
1225 roman-page-numbers,%
1226 Roman-page-numbers,%
1227 arabic-page-numbers,%
1228 alpha-page-numbers,%
1229 Alpha-page-numbers,%
1230 Appendix-page-numbers,%
1231 arabic-section-numbers%
1232 }

```

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

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

```

1233 \protected@edef\@gls@roman{\@roman{0}\string"
1234 \string"roman-numbers-lowercase\string" :sep \string"}}%
1235 \@onelevel@sanitize\@gls@roman
1236 \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
1237 :sep \string"}%
1238 \@onelevel@sanitize\@tmp
1239 \ifx\@tmp\@gls@roman
1240 \expandafter
1241 \edef\csname @gls@xdy@Lclass@roman-page-numbers\endcsname{%
1242 \string"roman-numbers-lowercase\string"%
1243 }%
1244 \else
1245 \expandafter
1246 \edef\csname @gls@xdy@Lclass@roman-page-numbers\endcsname{
1247 :sep \string"\@gls@roman\string"%
1248 }%
1249 \fi

```

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

```

1250 \expandafter\def\csname @gls@xdy@Lclass@Roman-page-numbers\endcsname{%
1251 \string"roman-numbers-uppercase\string"%
1252 }%

```

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

```

1253 \expandafter\def\csname @gls@xdy@Lclass@arabic-page-numbers\endcsname{%
1254 \string"arabic-numbers\string"%
1255 }%

```

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

```

1256 \expandafter\def\csname @gls@xdy@Lclass@alpha-page-numbers\endcsname{%
1257 \string"alpha\string"%
1258 }%

```

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

```

1259 \expandafter\def\csname @gls@xdy@Lclass@Alpha-page-numbers\endcsname{%
1260 \string"ALPHA\string"%
1261 }%

```

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

```

1262 \expandafter\def\csname @gls@xdy@Lclass@Appendix-page-numbers\endcsname{%
1263 \string"ALPHA\string"
1264 :sep \string"\@glsAlphacompositor\string"
1265 \string"arabic-numbers\string"%
1266 }

```

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

```

1267 \expandafter\def\csname @gls@xdy@Lclass@arabic-section-numbers\endcsname{%
1268   \string"arabic-numbers\string"
1269   :sep \string"\glscompositor\string"
1270   \string"arabic-numbers\string"%
1271 }%
```

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

```

1272 \def\@xdyuserlocationdefs{}
```

erlocationnames List of additional user location names

```

1273 \def\@xdyuserlocationnames{}
```

End of xindy-only block:

```

1274 \fi
```

xdycrossrefhook Hook used after writing cross-reference class information.

```

1275 \ifglsxindy
1276 \newcommand\@xdycrossrefhook{}
1277 \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.)

```

1278 \ifglsxindy
1279 \newcommand*\GlsAddXdyLocation[3][{}]{%
1280   \def\@gls@tmp{#1}%
1281   \ifx\@gls@tmp\@empty
1282     \edef\@xdyuserlocationdefs{%
1283       \@xdyuserlocationdefs ^^J%
1284       (define-location-class \string"#2\string"^^J\space\space
1285       \space(:sep \string"{}\glsopenbrace\string" #3
1286       :sep \string"\glsclosebrace\string"))
1287   }%
1288   \else
1289     \edef\@xdyuserlocationdefs{%
1290       \@xdyuserlocationdefs ^^J%
1291       (define-location-class \string"#2\string"^^J\space\space
1292       \space(:sep "\glsopenbrace"
1293       #1
1294       :sep "\glsclosebrace\glsopenbrace" #3
1295       :sep "\glsclosebrace"))
1296   }%
1297 \fi

1298 \edef\@xdyuserlocationnames{%
1299   \@xdyuserlocationnames^^J\space\space\space
1300   \string"#2\string"}%
1301 }
```

Only has an effect before \writeist:

```
1302 \@onlypremakeg\GlsAddXdyLocation
1303 \else
1304   \newcommand*{\GlsAddXdyLocation}[2]{%
1305     \glsnnoxindywarning\GlsAddXdyLocation}
1306 \fi
```

ationclassorder Define location class order

```
1307 \ifglxindy
1308   \def\@xdylocationclassorder{^^J\space\space\space
1309     \string"roman-page-numbers\string"^^J\space\space\space
1310     \string"arabic-page-numbers\string"^^J\space\space\space
1311     \string"arabic-section-numbers\string"^^J\space\space\space
1312     \string"alpha-page-numbers\string"^^J\space\space\space
1313     \string"Roman-page-numbers\string"^^J\space\space\space
1314     \string"Alpha-page-numbers\string"^^J\space\space\space
1315     \string"Appendix-page-numbers\string"
1316     \@xdyuserlocationnames^^J\space\space\space
1317     \string"see\string"
1318   }
1319 \fi
```

Change the location order.

ationClassOrder

```
1320 \ifglxindy
1321   \newcommand*\GlsSetXdyLocationClassOrder[1]{%
1322     \def\@xdylocationclassorder{#1}}
1323 \else
1324   \newcommand*\GlsSetXdyLocationClassOrder[1]{%
1325     \glsnnoxindywarning\GlsSetXdyLocationClassOrder}
1326 \fi
```

\@xdysortrules Define sort rules

```
1327 \ifglxindy
1328   \def\@xdysortrules{}
1329 \fi
```

\GlsAddSortRule Add a sort rule

```
1330 \ifglxindy
1331   \newcommand*\GlsAddSortRule[2]{%
1332     \expandafter\toks@\expandafter{\@xdysortrules}%
1333     \protected@edef\@xdysortrules{\the\toks@ ^^J
1334       (sort-rule \string"#1\string" \string"#2\string")}%
1335   }
1336 \else
1337   \newcommand*\GlsAddSortRule[2]{%
1338     \glsnnoxindywarning\GlsAddSortRule}
1339 \fi
```


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

```
1340 \ifglxindy
1341   \def\@xdyrequiredstyles{tex}
1342 \fi
```

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

```
1343 \ifglxindy
1344   \newcommand*\GlsAddXdyStyle[1]{%
1345     \edef\@xdyrequiredstyles{\@xdyrequiredstyles,#1}}%
1346 \else
1347   \newcommand*\GlsAddXdyStyle[1]{%
1348     \glsnnoxindywarning\GlsAddXdyStyle}
1349 \fi
```

`GlsSetXdyStyles` Reset the list of required styles

```
1350 \ifglxindy
1351   \newcommand*\GlsSetXdyStyles[1]{%
1352     \edef\@xdyrequiredstyles{#1}}
1353 \else
1354   \newcommand*\GlsSetXdyStyles[1]{%
1355     \glsnnoxindywarning\GlsSetXdyStyles}
1356 \fi
```

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

```
1357 \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.

```
1358 \def\@xdylanguage#1#2{}
```

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

```
1359 \ifglxindy
1360   \newcommand*\GlsSetXdyLanguage[2][\glsdefaulttype]{%
1361     \ifglossaryexists{#1}{%
1362       \expandafter\def\csname @xdy@#1@language\endcsname{#2}%
1363     }{%
1364       \PackageError{glossaries}{Can't set language type for
1365         glossary type '#1' --- no such glossary}{%
1366         You have specified a glossary type that doesn't exist}}
1367 \else
1368   \newcommand*\GlsSetXdyLanguage[2][]{%
1369     \glsnnoxindywarning\GlsSetXdyLanguage}
1370 \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.

```
1371 \def\@gls@codepage#1#2{}
```

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

```
1372 \ifglxindy
1373   \newcommand*\GlsSetXdyCodePage[1]{%
1374     \renewcommand*\@gls@codepage{#1}%
1375   }

  Suggested by egreg:
1376   \AtBeginDocument{%
1377     \ifx\@gls@codepage\@empty
1378       \ifpackageloaded{fontspec}{\def\@gls@codepage{utf8}}{}%
1379     \fi
1380   }
1381 \else
1382   \newcommand*\GlsSetXdyCodePage[1]{%
1383     \glsnoxywarning\GlsSetXdyCodePage%
1384 \fi
```

`\GlsXdyLetterGroups` Store letter group definitions.

```
1385 \ifglxindy
1386   \ifglx@xindy@glxnumbers
1387     \def\GlsXdyLetterGroups{(define-letter-group
1388       \string"glxnumbers\string"^^J\space\space\space
1389       :prefixes (\string"0\string" \string"1\string"
1390       \string"2\string" \string"3\string" \string"4\string"
1391       \string"5\string" \string"6\string" \string"7\string"
1392       \string"8\string" \string"9\string")^^J\space\space\space
1393       :before \string"\@glxfirstletter\string")}
1394   \else
1395     \def\GlsXdyLetterGroups{}
1396   \fi
1397 \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.

```
1398   \newcommand*\GlsAddLetterGroup[2]{%
1399     \expandafter\toks@\expandafter{\@GlsXdyLetterGroups}%
1400     \protected@edef\@GlsXdyLetterGroups{\the\toks@^^J%
1401       (define-letter-group \string"#1\string"^^J\space\space\space#2)}%
1402   }%
```

1.5 Loops and conditionals

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

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

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

```
1403 \newcommand*{\forall glossaries}[3][\@glo@types]{%
1404   \@for#2:=#1\do{\ifx#2\@empty\else#3\fi}%
1405 }
```

`\forall acronyms`

```
1406 \newcommand*{\forall acronyms}[2]{%
1407   \@for#1:=\@glsacronymlists\do{\ifx#1\@empty\else#2\fi}%
1408 }
```

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

`\forall glsentries [<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.

```
1409 \newcommand*{\forall glsentries}[3][\glsdefaulttype]{%
1410   \edef\@glo@list{\csname glist@#1\endcsname}%
1411   \@for#2:=\@glo@list\do
1412     {%
1413       \ifdefempty{#2}{#3}%
1414     }%
1415 }
```

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

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

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

```
1416 \newcommand*{\forall glsentries}[3][\@glo@types]{%
1417   \expandafter\forall glossaries\expandafter[#1]{\@this@glo@}%
1418   {%
1419     \forall glsentries[\@this@glo@]{#2}{#3}%
1420   }%
1421 }
```

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

`\if glossaryexists {<type>} {<true-text>} {<false-text>}`

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

```
1422 \newcommand{\if glossaryexists}[3]{%
1423   \ifcsundef{@glo@type@#1@out}{#3}{#2}%
1424 }
```

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`

```
1425 \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.

```
1426 \newcommand{\ifglsentryexists}[3]{%
1427   \ifcsundef{glo@\glsdetoklabel{#1}@name}{#3}{#2}%
1428 }
```

`\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>`.

```
1429 \newcommand*\ifglsused[3]{%
1430   \ifbool{glo@\glsdetoklabel{#1}@flag}{#2}{#3}%
1431 }
```

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>`.

```
1432 \newcommand{\glsdoifexists}[2]{%
1433   \ifglsentryexists{#1}{#2}{%
1434     \PackageError{glossaries}{Glossary entry '\glsdetoklabel{#1}'
1435       has not been defined}{You need to define a glossary entry before you
1436       can use it.}}%
1437 }
```

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

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

```
1438 \newcommand{\glsdoifnoexists}[2]{%
1439   \ifglentryexists{#1}{%
1440     \PackageError{glossaries}{Glossary entry ‘\glsdetoklabel{#1}’ has already
1441     been defined}{}}{#2}%
1442 }
```

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

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

```
1443 \newcommand{\glsdoifexistsorwarn}[2]{%
1444   \ifglentryexists{#1}{#2}{%
1445     \GlossariesWarning{Glossary entry ‘\glsdetoklabel{#1}’
1446     has not been defined}%
1447   }%
1448 }
```

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

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

```
1449 \newcommand{\glsdoifexistsordo}[3]{%
1450   \ifglentryexists{#1}{#2}{%
1451     \PackageError{glossaries}{Glossary entry ‘\glsdetoklabel{#1}’
1452     has not been defined}{You need to define a glossary entry before you
1453     can use it.}%
1454     #3%
1455   }%
1456 }
```

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

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

```
1457 \newcommand{\doifglossarynoexistsordo}[3]{%
1458   \ifglossaryexists{#1}%
1459   {%
1460     \PackageError{glossaries}{Glossary type ‘#1’ already exists}{}%
1461     #3%
1462   }%
1463   {#2}%
1464 }
```

```

ifglshaschildren \ifglshaschildren{<label>}{<true part>}{<false part>}
1465 \newcommand{\ifglshaschildren}[3]{%
1466   \glsdoifexists{#1}%
1467   {%
1468     \def\do@glshaschildren{#3}%
1469     \edef\@gls@thislabel{\glsdetoklabel{#1}}%
1470     \expandafter\for@gl@sentries\expandafter
1471     [\csname glo@\@gls@thislabel @type\endcsname]
1472     {\glo@label}%
1473     {%
1474       \letcs\glo@parent{glo@\glo@label @parent}%
1475       \ifdefequal\@gls@thislabel\glo@parent
1476       {%
1477         \def\do@glshaschildren{#2}%
1478         \@endfortrue
1479       }%
1480     }%
1481   }%
1482   \do@glshaschildren
1483 }%
1484 }

```

```

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

```

```

1485 \newcommand{\ifglshasparent}[3]{%
1486   \glsdoifexists{#1}%
1487   {%
1488     \ifcsemt{glo@\glsdetoklabel{#1}@parent}{#3}{#2}%
1489   }%
1490 }

```

```

\ifglshasdesc \ifglshasdesc{<label>}{<true part>}{<false part>}
1491 \newcommand*{\ifglshasdesc}[3]{%
1492   \ifcsemt{glo@\glsdetoklabel{#1}@desc}%
1493   {#3}%
1494   {#2}%
1495 }

```

```

sdescsuppressed \ifglsdscsuppressed{<label>}{<true part>}{<false part>} Does <true part> if the descrip-
tion is just \nopostdesc otherwise does <false part>.
1496 \newcommand*{\ifglsdscsuppressed}[3]{%
1497   \ifcsequal{glo@\glsdetoklabel{#1}@desc}{@no@post@desc}%
1498   {#2}%
1499   {#3}%
1500 }

```

```

\ifglshassymbol \ifglshassymbol{<label>}{<true part>}{<false part>}
1501 \newcommand*{\ifglshassymbol}[3]{%
1502   \letcs{\@glo@symbol}{glo\glsdetoklabel{#1}@symbol}%
1503   \ifdefempty\@glo@symbol
1504     {#3}%
1505     {%
1506       \ifdefequal\@glo@symbol\@gls@default@value
1507         {#3}%
1508         {#2}%
1509       }%
1510 }

```

```

\ifglshaslong \ifglshaslong{<label>}{<true part>}{<false part>}
1511 \newcommand*{\ifglshaslong}[3]{%
1512   \letcs{\@glo@long}{glo\glsdetoklabel{#1}@long}%
1513   \ifdefempty\@glo@long
1514     {#3}%
1515     {%
1516       \ifdefequal\@glo@long\@gls@default@value
1517         {#3}%
1518         {#2}%
1519       }%
1520 }

```

```

\ifglshasshort \ifglshasshort{<label>}{<true part>}{<false part>}
1521 \newcommand*{\ifglshasshort}[3]{%
1522   \letcs{\@glo@short}{glo\glsdetoklabel{#1}@short}%
1523   \ifdefempty\@glo@short
1524     {#3}%
1525     {%
1526       \ifdefequal\@glo@short\@gls@default@value
1527         {#3}%
1528         {#2}%
1529       }%
1530 }

```

```

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

```

```

1531 \newcommand*{\ifglshasfield}[4]{%
1532   \glsdoifexists{#2}%
1533   {%
1534     \letcs{\@glo@thisvalue}{glo\glsdetoklabel{#2}@#1}%

```

First check supplied field label is defined.

```

1535   \ifdef\@glo@thisvalue
1536     {%

```

Is defined, so now check if empty.

```

1537     \ifdefempty\@glo@thisvalue
1538     {%

```

Is empty, so doesn't have field set.

```

1539         #4%
1540     }%
1541     {%

```

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

```

1542     \ifdefequal\@glo@thisvalue\@gls@default@value
1543     {%

```

Value is set to the default value.

```

1544         #4%
1545     }%
1546     {%

```

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

```

1547     \let\glscurrentfieldvalue\@glo@thisvalue
1548     #3%
1549 }%
1550 }%
1551 }%
1552 {%

```

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

```

1553     \@gls@fetchfield{\@gls@thisfield}{#1}%

```

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

```

1554     \ifdef\@gls@thisfield
1555     {%

```

Is defined, so now check if empty.

```

1556     \letcs{\@glo@thisvalue}{glo@glsetoklabel{#2}@\@gls@thisfield}%
1557     \ifdefempty\@glo@thisvalue
1558     {%

```

Is empty so field hasn't been set.

```

1559         #4%
1560     }%
1561     {%

```

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

```

1562     \ifdefequal\@glo@thisvalue\@gls@default@value
1563     {%

```

Value is set to the default value.

```

1564         #4%
1565     }%
1566     {%

```


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

```
1567         \let\glscurrentfieldvalue\@glo@thisvalue
1568         #3%
1569     }%
1570 }%
1571 }%
1572 {%
    Not defined.
1573     \GlossariesWarning{Unknown entry field ‘#1’}%
1574     #4%
1575 }%
1576 }%
1577 }%
1578 }
```

`\glscurrentfieldvalue`

```
1579 \newcommand*{\glscurrentfieldvalue}{}%
```

1.6 Defining new glossaries

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

`\@glo@types`

```
1580 \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.

```
1581 \newcommand*\@gls@provide@newglossary{%
1582     \protected@write\@auxout{}\string\providecommand\string\@newglossary[4]{}%
    Only need to do this once.
```

```
1583     \let\@gls@provide@newglossary\relax
1584 }
```

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

```
1585 \newcommand*{\defglsentryfmt}[2][\glsdefaulttype]{%
1586     \csgdef{gls@#1@entryfmt}{#2}%
1587 }
```

`\gls@doentryfmt`

```
1588 \newcommand*{\gls@doentryfmt}[1]{\csuse{gls@#1@entryfmt}}
```

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

```

1589 \newcommand*{\@gls@forbidtextext}[1]{%
1590   \ifboolexpr{test {\ifdefstring{#1}{tex}}
1591             or test {\ifdefstring{#1}{TEX}}}
1592   {%
1593     \def#1{nottex}%
1594     \PackageError{glossaries}%
1595       {Forbidden '.tex' extension replaced with '.nottex'}%
1596       {I'm sorry, I can't allow you to do something so reckless.\MessageBreak
1597         Don't use '.tex' as an extension for a temporary file.}%
1598   }%
1599   {%
1600   }%
1601 }

```

`\gls@gobbleopt` Discard optional argument.

```

1602 \newcommand*{\gls@gobbleopt}{\new@ifnextchar[{\@gls@gobbleopt}{}}
1603 \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`

```

1604 \newcommand*{\newglossary}{\@ifstar\s@newglossary\ns@newglossary}

```

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

```

1605 \newcommand*{\s@newglossary}[2]{%
1606   \ns@newglossary[#1-glg]{#1}{#1-gls}{#1-glo}{#2}%
1607 }

```

`\ns@newglossary` Define the unstarred version.

```

1608 \newcommand*{\ns@newglossary}[5][glg]{%
1609   \doifglossarynoexistsordo{#2}%
1610   {%

```

Check if default has been set

```

1611   \ifundef\glsdefaultttype

```

```

1612 {%
1613     \gdef\glsdefaultttype{#2}%
1614 }{}%

```

Add this to the list of glossary types:

```

1615 \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.

```

1616 \expandafter\gdef\csname glolist@#2\endcsname{,}%

```

Store the file extensions:

```

1617 \expandafter\edef\csname @glotype@#2@log\endcsname{#1}%
1618 \expandafter\edef\csname @glotype@#2@in\endcsname{#3}%
1619 \expandafter\edef\csname @glotype@#2@out\endcsname{#4}%
1620 \expandafter\@gls@forbidtextext\csname @glotype@#2@log\endcsname
1621 \expandafter\@gls@forbidtextext\csname @glotype@#2@in\endcsname
1622 \expandafter\@gls@forbidtextext\csname @glotype@#2@out\endcsname

```

Store the title:

```

1623 \expandafter\def\csname @glotype@#2@title\endcsname{#5}%

```

```

1624 \@gls@provide@newglossary
1625 \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.

```

1626 \ifcsundef{gls@#2@entryfmt}%
1627 {%
1628     \defglsentryfmt[#2]{\glsentryfmt}%
1629 }%
1630 {}%

```

Define sort counter if required:

```

1631 \@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.)

```

1632 \@ifnextchar[{\@gls@setcounter{#2}}%
1633 {\@gls@setcounter{#2}[\glscounter]}%
1634 }%
1635 {%
1636     \gls@gobbleopt
1637 }%
1638 }

```

\altnewglossary

```

1639 \newcommand*\altnewglossary}[3]{%
1640     \newglossary[#2-glg]{#1}{#2-gls}{#2-glo}{#3}%
1641 }

```

Only define new glossaries in the preamble:

```
1642 \@onlypreamble{\newglossary}
```

Only define new glossaries before `\makeglossaries`

```
1643 \@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`

```
1644 \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`

```
1645 \def\@gls@setcounter#1[#2]{%
```

```
1646   \expandafter\def\csname @gls@#1@counter\endcsname{#2}%
```

Add counter to xindy list, if not already added:

```
1647   \ifglsxindy
```

```
1648     \GlsAddXdyCounters{#2}%
```

```
1649   \fi
```

```
1650 }
```

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

`@gls@getcounter`

```
1651 \newcommand*{\@gls@getcounter}[1]{%
```

```
1652   \csname @gls@#1@counter\endcsname
```

```
1653 }
```

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

```
1654 \glsdefmain
```

Define the “acronym” glossaries if required.

```
1655 \@gls@do@acronymsdef
```

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

```
1656 \@gls@do@symbolsdef
```

```
1657 \@gls@do@numbersdef
```

```
1658 \@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.

```
1659 \newcommand*{\newignoredglossary}[1]{%
```

```
1660   \ifdefempty\@ignored@glossaries
```

```

1661 {%
1662   \edef\@ignored@glossaries{#1}%
1663 }%
1664 {%
1665   \eappto\@ignored@glossaries{,#1}%
1666 }%
1667 \csgdef{glolist@#1}{,}%
1668 \ifcsundef{gls@#1@entryfmt}%
1669 {%
1670   \defglsentryfmt[#1]{\glsentryfmt}%
1671 }%
1672 {}%
1673 \ifdefempty\@gls@nohyperlist
1674 {%
1675   \renewcommand*{\@gls@nohyperlist}{#1}%
1676 }%
1677 {%
1678   \eappto\@gls@nohyperlist{,#1}%
1679 }%
1680 }

```

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

```
1681 \newcommand*{\@ignored@glossaries}{}

```

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

```

1682 \newcommand*{\ifignoredglossary}[3]{%
1683   \edef\@gls@igtype{#1}%
1684   \expandafter\DTLifinlist\expandafter
1685     {\@gls@igtype}{\@ignored@glossaries}{#2}{#3}%
1686 }

```

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.

```

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

```

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`.

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

descriptionplural

```
1693 \define@key{glossentry}{descriptionplural}{%
1694 \def\@glo@descplural{#1}%
1695 }
```

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>*.

```
1696 \define@key{glossentry}{sort}{%
1697 \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.

```
1698 \define@key{glossentry}{text}{%
1699 \def\@glo@text{#1}%
1700 }
```

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.

```
1701 \define@key{glossentry}{plural}{%
1702 \def\@glo@plural{#1}%
1703 }
```

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.

```
1704 \define@key{glossentry}{first}{%
1705 \def\@glo@first{#1}%
1706 }
```

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.

```
1707 \define@key{glossentry}{firstplural}{%
1708 \def\@glo@firstplural{#1}%
1709 }
```

s@default@value

```
1710 \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).

```
1711 \define@key{glossentry}{symbol}{%
1712 \def\@glo@symbol{#1}%
1713 }
```

symbolplural

```
1714 \define@key{glossentry}{symbolplural}{%
1715 \def\@glo@symbolplural{#1}%
1716 }
```

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

```
1717 \define@key{glossentry}{type}{%
1718 \def\@glo@type{#1}}
```

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

```
1719 \define@key{glossentry}{counter}{%
1720   \ifcsundef{c@#1}%
1721   {%
1722     \PackageError{glossaries}%
1723     {There is no counter called ‘#1’}%
1724     {%
1725       The counter key should have the name of a valid counter
1726       as its value%
1727     }%
1728   }%
1729   {%
1730     \def\@glo@counter{#1}%
1731   }%
1732 }
```

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

```
1733 \define@key{glossentry}{see}{%
1734   \gls@set@xr@key{see}{\@glo@see}{#1}%
1735 }
```

`\gls@set@xr@key` `\gls@set@xr@key{<key name>}{<cs>}{<value>}`

Assign a cross-reference key.

```
1736 \newcommand*{\gls@set@xr@key}[3]{%
1737   \renewcommand*{\gls@xr@key}{#1}%
1738 }
```

```

1738 \gls@checkseeallowed
1739 \def#2{#3}%
1740 \@glo@seeautonumberlist
1741 }

\gls@xr@key
1742 \newcommand*{\gls@xr@key}{see}

checkseeallowed
1743 \newcommand*{\gls@checkseeallowed}{%
1744 \gls@see@noindex
1745 }

ed@preambleonly
1746 \newcommand*{\gls@checkseeallowed@preambleonly}{%
1747 \GlossariesWarning{glossaries}%
1748 {'\gls@xr@key' key doesn't have any effect when used in the document
1749 environment. Move the definition to the preamble
1750 after \string\makeglossaries\space
1751 or \string\makenoidxglossaries}%
1752 }

parent The parent key specifies the parent entry, if required.
1753 \define@key{glossentry}{parent}{%
1754 \def\@glo@parent{#1}}

nonumberlist The nonumberlist key suppresses or activates the number list for the given entry.
1755 \define@choicekey{glossentry}{nonumberlist}[\val\nr]{true,false}[true]{%
1756 \ifcase\nr\relax
1757 \def\@glo@prefix{\glsnonextpages}%
1758 \@gls@savenonumberlist{true}%
1759 \else
1760 \def\@glo@prefix{\glsnextpages}%
1761 \@gls@savenonumberlist{false}%
1762 \fi
1763 }

savenonumberlist The nonumberlist option isn't saved by default (as it just sets the prefix) which isn't a problem
when the entries are defined in the preamble, but causes a problem when entries are defined
in the document. In this case, the value needs to be saved so that it can be written to the
.glsdefs file.
1764 \newcommand*{\@gls@savenonumberlist}[1]{%

initnonumberlist
1765 \newcommand*{\@gls@initnonumberlist}{}%

nitnonumberlist
1766 \newcommand*{\@gls@storenonumberlist}[1]{%

```


savenonumberlist Allow the nonumberlist value to be saved.

```
1767 \newcommand*{\@gls@enablesavenonumberlist}{%
1768   \renewcommand*{\@gls@initnonumberlist}{%
1769     \undef\@glo@nonumberlist
1770   }%
1771   \renewcommand*{\@gls@savenonumberlist}[1]{%
1772     \def\@glo@nonumberlist{##1}%
1773   }%
1774   \renewcommand*{\@gls@storenonumberlist}[1]{%
1775     \ifdef\@glo@nonumberlist
1776       {%
1777         \cslet{glo@glstdetoklabel{##1}@nonumberlist}{\@glo@nonumberlist}%
1778       }%
1779     }%
1780   }%
1781   \appto\@gls@keymap{,{nonumberlist}{nonumberlist}}%
1782 }
```

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

user1

```
1783 \define@key{glossentry}{user1}{%
1784   \def\@glo@useri{#1}%
1785 }
```

user2

```
1786 \define@key{glossentry}{user2}{%
1787   \def\@glo@userii{#1}%
1788 }
```

user3

```
1789 \define@key{glossentry}{user3}{%
1790   \def\@glo@useriii{#1}%
1791 }
```

user4

```
1792 \define@key{glossentry}{user4}{%
1793   \def\@glo@useriv{#1}%
1794 }
```

user5

```
1795 \define@key{glossentry}{user5}{%
1796   \def\@glo@userv{#1}%
1797 }
```

user6

```
1798 \define@key{glossentry}{user6}{%
1799   \def\@glo@uservi{#1}%
1800 }
```

`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.

```
1801 \define@key{glossentry}{short}{%
1802   \def\@glo@short{#1}%
1803 }
```

`shortplural` This key is provided for use by `\newacronym`.

```
1804 \define@key{glossentry}{shortplural}{%
1805   \def\@glo@shortpl{#1}%
1806 }
```

`long` This key is provided for use by `\newacronym`.

```
1807 \define@key{glossentry}{long}{%
1808   \def\@glo@long{#1}%
1809 }
```

`longplural` This key is provided for use by `\newacronym`.

```
1810 \define@key{glossentry}{longplural}{%
1811   \def\@glo@longpl{#1}%
1812 }
```

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

```
1813 \newcommand*{\@glsnoname}{%
1814   \PackageError{glossaries}{name key required in
1815     \string\newglossaryentry\space for entry '\@glo@label'}{You
1816     haven't specified the entry name}}
```

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

```
1817 \newcommand*{\@glsnodels}{%
1818   \PackageError{glossaries}
1819   {%
1820     description key required in \string\newglossaryentry\space
1821     for entry '\@glo@label'%
1822   }%
1823   {%
1824     You haven't specified the entry description%
1825   }%
1826 }
```

`lsdefaultplural` Now obsolete. Don't use.

```
1827 \newcommand*{\@glsdefaultplural}{}
```

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

```
1828 \newcommand*{\@gls@missingnumberlist}[1]{%
1829   ??%
1830   \ifglssavenumberlist
1831     \GlossariesWarning{Missing number list for entry '#1'.
1832       Maybe makeglossaries + rerun required}%
1833 }
```

```

1833 \else
1834   \PackageError{glossaries}%
1835   {Package option ‘savenumberlist=true’ required}%
1836   {%
1837     You must use the ‘savenumberlist’ package option
1838     to reference location lists.%
1839   }%
1840 \fi
1841 }

```

`\glsdefaultsort` Define command to set default sort.

```
1842 \newcommand*{\@glsdefaultsort}{\@glo@name}
```

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

```
1843 \newcount\gls@level
```

`@noexpand@field`

```

1844 \newcommand{\@@gls@noexpand@field}[3]{%
1845   \expandafter\global\expandafter
1846   \let\csname glo@#1@#2\endcsname#3%
1847 }

```

`noexpand@fields`

```

1848 \newcommand{\@gls@noexpand@fields}[4]{%
1849   \ifcsdef{gls@assign@#3@field}
1850   {%
1851     \ifdefequal{#4}{\@gls@default@value}%
1852     {%
1853       \edef\@gls@value{\expandonce{#1}}%
1854       \csuse{gls@assign@#3@field}{#2}{\@gls@value}%
1855     }%
1856     {%
1857       \csuse{gls@assign@#3@field}{#2}{#4}%
1858     }%
1859   }%
1860   {%
1861     \ifdefequal{#4}{\@gls@default@value}%
1862     {%
1863       \edef\@gls@value{\expandonce{#1}}%
1864       \@@gls@noexpand@field{#2}{#3}{\@gls@value}%
1865     }%
1866     {%
1867       \@@gls@noexpand@field{#2}{#3}{#4}%
1868     }%
1869   }%
1870 }

```

`ls@expand@field`

```

1871 \newcommand{\@gls@expand@field}[3]{%
1872   \expandafter
1873   \protected@xdef\csname glo@#1@#2\endcsname{#3}%
1874 }

```

s@expand@fields

```

1875 \newcommand{\@gls@expand@fields}[4]{%
1876   \ifcsdef{gls@assign@#3@field}
1877   {%
1878     \ifdefequal{#4}{\@gls@default@value}%
1879     {%
1880       \edef\@gls@value{\expandonce{#1}}%
1881       \csuse{gls@assign@#3@field}{#2}{\@gls@value}%
1882     }%
1883     {%
1884       \expandafter\@gls@startswithexpandonce#4\relax\relax\gls@endcheck
1885       {%
1886         \@gls@expand@field{#2}{#3}{#4}%
1887       }%
1888       {%
1889         \csuse{gls@assign@#3@field}{#2}{#4}%
1890       }%
1891     }%
1892   }%
1893   {%
1894     \ifdefequal{#4}{\@gls@default@value}%
1895     {%
1896       \@gls@expand@field{#2}{#3}{#1}%
1897     }%
1898     {%
1899       \@gls@expand@field{#2}{#3}{#4}%
1900     }%
1901   }%
1902 }

```

swithexpandonce

```

1903 \def\@gls@expandonce{\expandonce}
1904 \def\@gls@startswithexpandonce#1#2\gls@endcheck#3#4{%
1905   \def\@gls@tmp{#1}%
1906   \ifdefequal{\@gls@expandonce}{\@gls@tmp}{#3}{#4}%
1907 }

```

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.

```

1908 \let\gls@assign@field\@gls@expand@fields

```

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

```
1909 \newcommand*{\glsexpandfields}{%
1910   \let\gls@assign@field\@gls@expand@fields
1911 }
```

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

```
1912 \newcommand*{\glsnoexpandfields}{%
1913   \let\gls@assign@field\@gls@noexpand@fields
1914 }
```

`newglossaryentry` Define `\newglossaryentry` $\langle label \rangle$ $\langle key-val list \rangle$. There are two required fields in $\langle key-val list \rangle$: name (or parent) and description. (See above.)

```
1915 \newrobustcmd{\newglossaryentry}[2]{%
    Check to see if this glossary entry has already been defined:
1916   \glsdoifnoexists{#1}%
1917   {%
1918     \gls@defglossaryentry{#1}{#2}%
1919   }%
1920 }
```

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

```
1921 \newcommand*{\gls@defdocnewglossaryentry}{%
1922   \let\gls@checkseeallowed\gls@checkseeallowed@preambleonly
1923   \let\newglossaryentry\new@glossaryentry
1924 }
```

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

```
1925 \newrobustcmd{\provideglossaryentry}[2]{%
1926   \ifglstryexists{#1}%
1927   {%
1928     {%
1929       \gls@defglossaryentry{#1}{#2}%
1930     }%
1931   }%
1932 \@onlypreamble{\provideglossaryentry}
```

`w@glossaryentry` For use in document environment.

```
1933 \newrobustcmd{\new@glossaryentry}[2]{%
1934   \ifundef\@gls@deffile
1935   {%
1936     \global\newwrite\@gls@deffile
1937     \immediate\openout\@gls@deffile=\jobname.glsdefs
1938   }%
1939   {%
```

```

1940 \ifglstryexists{#1}{}%
1941 {%
1942   \gls@defglossaryentry{#1}{#2}%
1943 }%
1944 \@gls@writedef{#1}%
1945 }
1946 \AtBeginDocument
1947 {
1948   \@gls@enablesavenonumberlist
1949   \makeatletter
1950   \InputIfFileExists{\jobname.glsdefs}{%}{%
1951     \makeatother
1952     \gls@defdocnewglossaryentry
1953   }
1954 \AtEndDocument{\ifdef\@gls@deffile{\closeout\@gls@deffile}{%}}

```

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

```

1955 \newcommand*{\@gls@writedef}[1]{%
1956   \immediate\write\@gls@deffile
1957   {%
1958     \string\ifglstryexists{#1}{%\glspercentchar^^J%
1959     \expandafter\@gobble\string\{\glspercentchar^^J%
1960     \string\gls@defglossaryentry{\glsdetoklabel{#1}}\glspercentchar^^J%
1961     \expandafter\@gobble\string\{\glspercentchar%
1962   }%

```

Write key value information:

```

1963 \@for\@gls@map:=\@gls@keymap\do
1964 {%
1965   \letcs\gls@value{gls@detoklabel{#1}}\@expandafter\@secondoftwo\@gls@map}%
1966   \ifdef\gls@value
1967   {%
1968     \@onelevel@sanitize\gls@value
1969     \immediate\write\@gls@deffile
1970     {%
1971       \expandafter\@firstoftwo\@gls@map
1972       =\expandafter\@gobble\string\{\gls@value\expandafter\@gobble\string\},%
1973       \glspercentchar
1974     }%
1975   }%
1976 }%
1977 }%

```

Provide hook:

```

1978 \gls@writedefhook
1979 \immediate\write\@gls@deffile
1980 {%
1981   \glspercentchar^^J%
1982   \expandafter\@gobble\string\}\glspercentchar^^J%
1983   \expandafter\@gobble\string\}\glspercentchar%

```

```

1984 }%
1985 }

```

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

```

1986 \newcommand*{\@gls@keymap}{%
1987   {name}{name},%
1988   {sort}{sortvalue},% unescaped sort value
1989   {type}{type},%
1990   {first}{first},%
1991   {firstplural}{firstpl},%
1992   {text}{text},%
1993   {plural}{plural},%
1994   {description}{desc},%
1995   {descriptionplural}{descplural},%
1996   {symbol}{symbol},%
1997   {symbolplural}{symbolplural},%
1998   {user1}{useri},%
1999   {user2}{userii},%
2000   {user3}{useriii},%
2001   {user4}{useriv},%
2002   {user5}{userv},%
2003   {user6}{uservi},%
2004   {long}{long},%
2005   {longplural}{longpl},%
2006   {short}{short},%
2007   {shortplural}{shortpl},%
2008   {counter}{counter},%
2009   {parent}{parent}}%
2010 }

```

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

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

```

2011 \newcommand*{\@gls@fetchfield}[2]{%

```

Ensure user field name is fully expanded

```

2012   \edef\@gls@thisval{#2}%

```

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

```

2013   \@for\@gls@map:=\@gls@keymap\do{%
2014     \edef\@this@key{\expandafter\@firstoftwo\@gls@map}%
2015     \ifdefequal{\@this@key}{\@gls@thisval}%
2016     {%

```

Found it.

```

2017       \edef#1{\expandafter\@secondoftwo\@gls@map}%

```

Break out of loop.

```

2018      \@endfortrue
2019    }%
2020    {}%
2021  }%
2022 }

```

`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.

```

2023 \newcommand*{\glsaddstoragekey}{\@ifstar\@sglsaddstoragekey\@glsaddstoragekey}

```

Starred version switches on expansion for this key.

```

2024 \newcommand*{\@sglsaddstoragekey}[1]{%
2025   \key@ifundefined{glossentry}{#1}%
2026   {%
2027     \expandafter\newcommand\expandafter*\expandafter
2028     {\csname gls@assign@#1@field\endcsname}[2]{%
2029       \@gls@expand@field{##1}{#1}{##2}%
2030     }%
2031   }%
2032   {}%
2033   \@glsaddstoragekey{#1}%
2034 }

```

Unstarred version doesn't override default expansion.

```

2035 \newcommand*{\@glsaddstoragekey}[3]{%

```

Check the specified key doesn't already exist.

```

2036   \key@ifundefined{glossentry}{#1}%
2037   {%

```

Set up the key.

```

2038     \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
2039     \appto\@gls@keymap{, {#1}{#1}}%

```

Set the default value.

```

2040     \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%

```

Assignment code.

```

2041     \appto\@newglossaryentryposthook{%
2042       \letcs{\@glo@tmp}{@glo@#1}%
2043       \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
2044     }%

```

Define the no-link commands.

```

2045     \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
2046   }%
2047   {%
2048     \PackageError{glossaries}{Key ‘#1’ already exists}{}%

```



```
2049 }%
2050 }
```

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

Allow user to add their own custom keys.

```
2051 \newcommand*{\glsaddkey}{\@ifstar\sglsaddkey\@glsaddkey}
```

Starred version switches on expansion for this key.

```
2052 \newcommand*{\@sglsaddkey}[1]{%
2053   \key@ifundefined{glossentry}{#1}%
2054   {%
2055     \expandafter\newcommand\expandafter*\expandafter
2056     {\csname gls@assign@#1@field\endcsname}[2]{%
2057       \@gls@expand@field{##1}{#1}{##2}%
2058     }%
2059   }%
2060   {}%
2061   \@glsaddkey{#1}%
2062 }
```

Unstarred version doesn't override default expansion.

```
2063 \newcommand*{\@glsaddkey}[7]{%
```

Check the specified key doesn't already exist.

```
2064   \key@ifundefined{glossentry}{#1}%
2065   {%
```

Set up the key.

```
2066     \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
2067     \appto\@gls@keymap{, {#1}{#1}}%
```

Set the default value.

```
2068     \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
```

Assignment code.

```
2069     \appto\@newglossaryentryposthook{%
2070       \letcs{\@glo@tmp}{@glo@#1}%
2071       \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
2072     }%
```

Define the no-link commands.

```
2073     \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
2074     \newcommand*{#4}[1]{\@Gls@entry@field{##1}{#1}}%
```

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

```
2075     \ifcsdef{@gls@user@#1@}%
2076     {%
2077       \PackageError{glossaries}%
2078       {Can't define '\string#5' as helper command
```

```

2079         '\expandafter\string\csname @gls@user@#1@\endcsname' already exists}%
2080     }%
2081 }%
2082 {%

2083     \expandafter\newcommand\expandafter*\expandafter
2084     {\csname @gls@user@#1@\endcsname}[2] [] {%
2085         \new@ifnextchar[%
2086             {\csuse{@gls@user@#1@}{##1}{##2}}}%
2087             {\csuse{@gls@user@#1@}{##1}{##2} [] }}}%
2088     \csdef{@gls@user@#1@}##1##2[##3] {%
2089         \@gls@field@link{##1}{##2}{#3{##2}##3}%
2090     }%
2091     \newrobustcmd*{#5}{%
2092         \expandafter\@gls@hyp@opt\csname @gls@user@#1@\endcsname}%
2093 }%

```

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

```

2094     \ifcsdef{@Gls@user@#1@}%
2095     {%
2096         \PackageError{glossaries}%
2097         {Can't define '\string#6' as helper command
2098         '\expandafter\string\csname @Gls@user@#1@\endcsname' already exists}%
2099         }%
2100     }%
2101     {%

2102     \expandafter\newcommand\expandafter*\expandafter
2103     {\csname @Gls@user@#1@\endcsname}[2] [] {%
2104         \new@ifnextchar[%
2105             {\csuse{@Gls@user@#1@}{##1}{##2}}}%
2106             {\csuse{@Gls@user@#1@}{##1}{##2} [] }}}%
2107     \csdef{@Gls@user@#1@}##1##2[##3] {%
2108         \@gls@field@link{##1}{##2}{#4{##2}##3}%
2109     }%
2110     \newrobustcmd*{#6}{%
2111         \expandafter\@gls@hyp@opt\csname @Gls@user@#1@\endcsname}%
2112     }%

```

Finally the all caps version:

```

2113     \ifcsdef{@GLS@user@#1@}%
2114     {%
2115         \PackageError{glossaries}%
2116         {Can't define '\string#7' as helper command
2117         '\expandafter\string\csname @GLS@user@#1@\endcsname' already exists}%
2118         }%
2119     }%
2120     {%

2121     \expandafter\newcommand\expandafter*\expandafter
2122     {\csname @GLS@user@#1@\endcsname}[2] [] {%

```

```

2123         \new@ifnextchar[%
2124             {\csuse{@GLS@user@#1@}{##1}{##2}}}%
2125             {\csuse{@GLS@user@#1@}{##1}{##2} []}}}%
2126     \csdef{@GLS@user@#1@}##1##2[##3]{%
2127         \@gls@field@link{##1}{##2}{\mfirstucMakeUppercase{#3{##2}##3}}}%
2128     }%
2129     \newrobustcmd*{#7}{%
2130         \expandafter\@gls@hyp@opt\csname @GLS@user@#1\endcsname}%
2131     }%
2132 }%
2133 {%
2134     \PackageError{glossaries}{Key ‘#1’ already exists}{}%
2135 }%
2136 }

```

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

```

2137 \newcommand{\glsfieldxdef}[3]{%
2138     \glsdoifexists{#1}%
2139     {%
2140         \edef\@glo@label{\glsdetoklabel{#1}}%
2141         \ifcsdef{glo@\@glo@label @#2}%
2142             {%
2143                 \expandafter\xdef\csname glo@\@glo@label @#2\endcsname{#3}%
2144             }%
2145             {%
2146                 \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2147             }%
2148     }%
2149 }

```

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

```

2150 \newcommand{\glsfielddedef}[3]{%
2151     \glsdoifexists{#1}%
2152     {%
2153         \edef\@glo@label{\glsdetoklabel{#1}}%
2154         \ifcsdef{glo@\@glo@label @#2}%
2155             {%
2156                 \expandafter\edef\csname glo@\@glo@label @#2\endcsname{#3}%
2157             }%
2158             {%
2159                 \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2160             }%
2161     }%

```

2162 }

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

```
2163 \newcommand{\glsfieldgdef}[3]{%
2164   \glsdoifexists{#1}%
2165   {%
2166     \edef\@glo@label{\glsdetoklabel{#1}}%
2167     \ifcsdef{glo@\@glo@label @#2}%
2168     {%
2169       \expandafter\gdef\csname glo@\@glo@label @#2\endcsname{#3}%
2170     }%
2171     {%
2172       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2173     }%
2174   }%
2175 }
```

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

```
2176 \newcommand{\glsfielddef}[3]{%
2177   \glsdoifexists{#1}%
2178   {%
2179     \edef\@glo@label{\glsdetoklabel{#1}}%
2180     \ifcsdef{glo@\@glo@label @#2}%
2181     {%
2182       \expandafter\def\csname glo@\@glo@label @#2\endcsname{#3}%
2183     }%
2184     {%
2185       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2186     }%
2187   }%
2188 }
```

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

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

```
2189 \newcommand{\glsfieldfetch}[3]{%
2190   \glsdoifexists{#1}%
2191   {%
2192     \edef\@glo@label{\glsdetoklabel{#1}}%
2193     \ifcsdef{glo@\@glo@label @#2}%
2194     {%
```

```

2195      \letcs#3{glo@\@glo@label @#2}%
2196    }%
2197    {%
2198      \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2199    }%
2200  }%
2201 }

```

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

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

```

2202 \newcommand{\ifglsfieldeq}[5]{%
2203   \glsdoifexists{#1}%
2204   {%
2205     \edef\@glo@label{\glsdetoklabel{#1}}%
2206     \ifcsdef{glo@\@glo@label @#2}%
2207     {%
2208       \ifcsstring{glo@\@glo@label @#2}{#3}{#4}{#5}%
2209     }%
2210     {%
2211       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2212     }%
2213   }%
2214 }

```

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

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

```

2215 \newcommand{\ifglsfieldddefeq}[5]{%
2216   \glsdoifexists{#1}%
2217   {%
2218     \edef\@glo@label{\glsdetoklabel{#1}}%
2219     \ifcsdef{glo@\@glo@label @#2}%
2220     {%
2221       \expandafter\ifdefstrequal
2222       \csname glo@\@glo@label @#2\endcsname{#3}{#4}{#5}%
2223     }%
2224     {%
2225       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2226     }%
2227   }%
2228 }

```

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

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

```

2229 \newcommand{\ifglsfieldcseq}[5]{%
2230   \glsdoifexists{#1}%
2231   {%
2232     \edef\@glo@label{\glsdetoklabel{#1}}%
2233     \ifcsdef{glo@\@glo@label @#2}%
2234     {%
2235       \ifcsstrequal{glo@\@glo@label @#2}{#3}{#4}{#5}%
2236     }%
2237     {%
2238       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2239     }%
2240   }%
2241 }
```

`gls.writedefhook`

```

2242 \newcommand*\gls.writedefhook{}
```

`gls@assign@desc`

```

2243 \newcommand*\gls@assign@desc}[1]{%
2244   \gls@assign@field{#1}{desc}{\@glo@desc}%
2245   \gls@assign@field{\@glo@desc}{#1}{descplural}{\@glo@descplural}%
2246 }
```

`ewglossaryentry`

```

2247 \newcommand{\longnewglossaryentry}[3]{%
2248   \glsdoifnoexists{#1}%
2249   {%
2250     \bgroup
2251     \let\@org@newglossaryentryprehook\@newglossaryentryprehook
2252     \long\def\@newglossaryentryprehook{%
2253       \long\def\@glo@desc{#3\leavevmode\unskip\nopostdesc}%
2254       \@org@newglossaryentryprehook
2255     }%
2256     \renewcommand*\gls@assign@desc}[1]{%
2257       \global\cslet{glo@\glsdetoklabel{#1}@desc}{\@glo@desc}%
2258       \global\cslet{glo@\glsdetoklabel{#1}@descplural}{\@glo@desc}%
2259     }
2260     \gls@defglossaryentry{#1}{#2}%
2261   \egroup
2262 }
2263 }
```

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

```

2264 \@onlypreamble{\longnewglossaryentry}
```

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

```

2265 \newcommand{\longprovideglossaryentry}[3]{%
```

```

2266 \ifglentryexists{#1}{}%
2267 {\longnewglossaryentry{#1}{#2}{#3}}%
2268 }
2269 \onlypreamble{\longprovideglossaryentry}

```

```

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

```

Defines a new entry without checking if it already exists.

```

2270 \newcommand{\gls@defglossaryentry}[2]{%

```

Prevent any further use of \GlsSetQuote:

```

2271 \let\GlsSetQuote\gls@nosetquote

```

Store label

```

2272 \edef\@glo@label{\glsdetoklabel{#1}}%

```

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

```

2273 \let\glslabel\@glo@label

```

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

```

2274 \let\@glo@name\@gls@name

```

```

2275 \let\@glo@desc\@gls@desc

```

```

2276 \let\@glo@descplural\@gls@default@value

```

```

2277 \let\@glo@type\@gls@default@value

```

```

2278 \let\@glo@symbol\@gls@default@value

```

```

2279 \let\@glo@symbolplural\@gls@default@value

```

```

2280 \let\@glo@text\@gls@default@value

```

```

2281 \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.)

```

2282 \let\@glo@first\@gls@default@value

```

```

2283 \let\@glo@firstplural\@gls@default@value

```

Set the default sort:

```

2284 \let\@glo@sort\@gls@default@value

```

Set the default counter:

```

2285 \let\@glo@counter\@gls@default@value

```

```

2286 \def\@glo@see{}%

```

```

2287 \def\@glo@parent{}%

```

```

2288 \def\@glo@prefix{}%

```

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

```
2289 \@gls@initnonnumberlist
```

```
2290 \def\@glo@useri{}%
2291 \def\@glo@userii{}%
2292 \def\@glo@useriii{}%
2293 \def\@glo@useriv{}%
2294 \def\@glo@userv{}%
2295 \def\@glo@uservi{}%
```

```
2296 \def\@glo@short{}%
2297 \def\@glo@shortpl{}%
2298 \def\@glo@long{}%
2299 \def\@glo@longpl{}%
```

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

```
2300 \@newglossaryentryprehook
```

Extract key-val information from third parameter:

```
2301 \setkeys{glossentry}{#2}%
```

Check there is a default glossary.

```
2302 \ifundef\glsdefaulttype
2303 {%
2304   \PackageError{glossaries}%
2305     {No default glossary type (have you used 'nomain' by mistake?)}%
2306     {If you use package option 'nomain' you must define
2307      a new glossary before you can define entries}%
2308 }%
2309 {}%
```

Assign type. This must be fully expandable

```
2310 \gls@assign@field{\glsdefaulttype}{\@glo@label}{type}{\@glo@type}%
2311 \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.

```
2312 \ifcsundef{glolist@\@glo@type}%
2313 {%
2314   \PackageError{glossaries}%
2315     {Glossary type '\@glo@type' has not been defined}%
2316     {You need to define a new glossary type, before making entries
2317      in it}%
2318 }%
2319 {}%
```

Check if it's an ignored glossary

```
2320 \ifignoredglossary\@glo@type
2321 {%
```


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

```
2322     \ifx\@glo@desc\@glsnodesc
2323         \let\@glo@desc\@empty
2324     \fi
2325 }%
2326 {%
2327 }%
2328 \protected@edef\@glo@list@{\csname glo@list@\@glo@type\endcsname}%
2329 \expandafter\xdef\csname glo@list@\@glo@type\endcsname{%
2330     \@glo@list@{\@glo@label},}%
2331 }%
```

Initialise level to 0.

```
2332 \gls@level=0\relax
```

Has this entry been assigned a parent?

```
2333 \ifx\@glo@parent\@empty
```

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

```
2334     \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%
2335 \else
```

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

```
2336     \ifdefequal\@glo@label\@glo@parent%
2337     {%
2338         \PackageError{glossaries}{Entry '@glo@label' can't be its own parent}{}%
2339         \def\@glo@parent{}%
2340         \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%
2341     }%
2342     {%
```

Check the parent exists:

```
2343     \ifglstryexists{\@glo@parent}%
2344     {%
```

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

```
2345     \expandafter\xdef\csname glo@\@glo@label @parent\endcsname{%
2346         \@glo@parent}%
```

Determine level.

```
2347     \gls@level=\csname glo@\@glo@parent @level\endcsname\relax
2348     \advance\gls@level by 1\relax
```

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

```
2349     \ifx\@glo@name\@glsnoname
2350         \expandafter\let\expandafter\@glo@name
2351         \csname glo@\@glo@parent @name\endcsname
```

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

```
2352     \ifx\@glo@plural\@gls@default@value
2353         \expandafter\let\expandafter\@glo@plural
2354         \csname glo@\@glo@parent @plural\endcsname
2355     \fi
```

```

2356     \fi
2357 }%
2358 {%

```

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

```

2359     \PackageError{glossaries}%
2360     {%
2361         Invalid parent '\@glo@parent'
2362         for entry '\@glo@label' - parent doesn't exist%
2363     }%
2364     {%
2365         Parent entries must be defined before their children%
2366     }%
2367     \def\@glo@parent{%
2368         \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{%
2369     }%
2370 }%
2371 \fi

```

Set the level for this entry

```

2372 \expandafter\xdef\csname glo@\@glo@label @level\endcsname{\number\gls@level}%

```

Define commands associated with this entry:

```

2373 \gls@assign@field{\@glo@name}{\@glo@label}{sortvalue}{\@glo@sort}%
2374 \letcs\@glo@sort{glo@\@glo@label @sortvalue}%
2375 \gls@assign@field{\@glo@name}{\@glo@label}{text}{\@glo@text}%
2376 \expandafter\gls@assign@field\expandafter
2377     {\csname glo@\@glo@label @text\endcsname\glspluralsuffix}%
2378     {\@glo@label}{plural}{\@glo@plural}%
2379 \expandafter\gls@assign@field\expandafter
2380     {\csname glo@\@glo@label @text\endcsname}%
2381     {\@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.

```

2382 \ifx\@glo@first\@gls@default@value
2383     \expandafter\gls@assign@field\expandafter
2384         {\csname glo@\@glo@label @plural\endcsname}%
2385         {\@glo@label}{firstpl}{\@glo@firstplural}%
2386 \else
2387     \expandafter\gls@assign@field\expandafter
2388         {\csname glo@\@glo@label @first\endcsname\glspluralsuffix}%
2389         {\@glo@label}{firstpl}{\@glo@firstplural}%
2390 \fi

2391 \ifcsundef{@glotype@\@glo@type @counter}%
2392 {%
2393     \def\@glo@defaultcounter{\glscounter}%
2394 }%
2395 {%
2396     \letcs\@glo@defaultcounter{@glotype@\@glo@type @counter}%

```

```

2397 }%
2398 \gls@assign@field{\@glo@defaultcounter}{\@glo@label}{counter}{\@glo@counter}%
2399 \gls@assign@field{}{\@glo@label}{useri}{\@glo@useri}%
2400 \gls@assign@field{}{\@glo@label}{userii}{\@glo@userii}%
2401 \gls@assign@field{}{\@glo@label}{useriii}{\@glo@useriii}%
2402 \gls@assign@field{}{\@glo@label}{useriv}{\@glo@useriv}%
2403 \gls@assign@field{}{\@glo@label}{uservi}{\@glo@uservi}%
2404 \gls@assign@field{}{\@glo@label}{short}{\@glo@short}%
2405 \gls@assign@field{}{\@glo@label}{shortpl}{\@glo@shortpl}%
2406 \gls@assign@field{}{\@glo@label}{long}{\@glo@long}%
2407 \gls@assign@field{}{\@glo@label}{longpl}{\@glo@longpl}%
2408 \ifx\@glo@name\@glsnname
2409   \@glsnname
2410   \let\@glo@name\@gls@default@value
2411 \fi
2412 \gls@assign@field{}{\@glo@label}{name}{\@glo@name}%

```

Set default numberlist if not defined:

```

2414 \ifcsundef{glo@\@glo@label @numberlist}%
2415 {%
2416   \csxdef{glo@\@glo@label @numberlist}{%
2417     \noexpand\@gls@missingnumberlist{\@glo@label}}%
2418 }%
2419 {}%

```

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

```

2420 \@gls@storenonumberlist{\@glo@label}%

```

The smaller and smallcaps options set the description to \@glo@first. Need to check for this, otherwise it won't get expanded if the description gets sanitized.

```

2421 \def\@glo@@desc{\@glo@first}%
2422 \ifx\@glo@desc\@glo@@desc
2423   \let\@glo@desc\@glo@first
2424 \fi
2425 \ifx\@glo@desc\@glsnnode
2426   \@glsnnode
2427   \let\@glo@desc\@gls@default@value
2428 \fi
2429 \gls@assign@desc{\@glo@label}%

```

Set the sort key for this entry:

```

2430 \@gls@defsort{\@glo@type}{\@glo@label}%

2431 \def\@glo@@symbol{\@glo@text}%
2432 \ifx\@glo@symbol\@glo@@symbol
2433   \let\@glo@symbol\@glo@text
2434 \fi
2435 \gls@assign@field{\relax}{\@glo@label}{symbol}{\@glo@symbol}%
2436 \expandafter
2437 \gls@assign@field\expandafter

```

```

2438   {\csname glo@\@glo@label @symbol\endcsname}
2439   {\@glo@label}{symbolplural}{\@glo@symbolplural}%

```

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

```

2440   \expandafter\xdef\csname glo@\@glo@label @flagfalse\endcsname{%
2441     \noexpand\global
2442     \noexpand\let\expandafter\noexpand
2443     \csname ifglo@\@glo@label @flag\endcsname\noexpand\iffalse
2444   }%
2445   \expandafter\xdef\csname glo@\@glo@label @flagtrue\endcsname{%
2446     \noexpand\global
2447     \noexpand\let\expandafter\noexpand
2448     \csname ifglo@\@glo@label @flag\endcsname\noexpand\iftrue
2449   }%
2450   \csname glo@\@glo@label @flagfalse\endcsname

```

Sort out any cross-referencing if required.

```

2451   \@glo@autosee

```

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

```

2452   \ifignoredglossary\@glo@type
2453   {%
2454     \csdef{glo@\@glo@label @index}{}%
2455   }
2456   {%
2457     \do@glo@storeentry{\@glo@label}%
2458   }%

```

Define entry counters if enabled:

```

2459   \@newglossaryentry@defcounters

```

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

```

2460   \@newglossaryentryposthook
2461 }

```

`\@glo@autosee` Automatically implement `\glssee`.

```

2462 \newcommand*{\@glo@autosee}{%
2463   \ifdefined@glo@see{%
2464     {%
2465       \protected@edef\@do@glssee{%
2466         \noexpand\@gls@fixbraces\noexpand\@glo@list\@glo@see\noexpand\@nil
2467         \noexpand\expandafter\noexpand\@glssee\noexpand\@glo@list{\@glo@label}}%
2468       \do@glssee
2469     }%
2470     \@glo@autoseehook
2471 }%

```

`glo@autoseehook`

```

2472 \newcommand*{\@glo@autoseehook}{%

```

aryentryprehook Allow extra information to be added to glossary entries:

```
2473 \newcommand*{\@newglossaryentryprehook}{}
```

ryentryposthook Allow extra information to be added to glossary entries:

```
2474 \newcommand*{\@newglossaryentryposthook}{}
```

try@defcounters

```
2475 \newcommand*{\@newglossaryentry@defcounters}{}
```

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

```
2476 \newcommand*{\glsmoveentry}[2]{%
2477   \edef\@glo@thislabel{\glsdetoklabel{#1}}%
2478   \edef\glo@type{\csname glo@\@glo@thislabel @type\endcsname}%
2479   \def\glo@list{,%}
2480   \for\glseentries[\glo@type]{\glo@label}%
2481   {%
2482     \ifdefequal\@glo@thislabel\glo@label
2483     {\eappto\glo@list{\glo@label,}}%
2484   }%
2485   \cslet\glolist@\glo@type{\glo@list}%
2486   \csdef{glo@\@glo@thislabel @type}{#2}%
2487 }
```

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

```
2488 \ifglxindy
2489   \newcommand*{\@glossaryentryfield}{\string\glossentry}
2490 \else
2491   \newcommand*{\@glossaryentryfield}{\string\glossentry}
2492 \fi
```

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

```
2493 \ifglxindy
2494   \newcommand*{\@glossarysubentryfield}{%
2495     \string\subglossentry}
2496 \else
2497   \newcommand*{\@glossarysubentryfield}{%
2498     \string\subglossentry}
2499 \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.)

```
2500 \newcommand{\@glo@storeentry}[1]{%
```

Escape makeindex/xindy special characters in the label:

```
2501 \edef\@glo@esclabel{#1}%
```

```
2502 \@gls@checkmkidxchars\@glo@esclabel
```

Get the sort string and escape any special characters

```
2503 \protected@edef\@glo@sort{\csname glo@#1@sort\endcsname}%
```

```
2504 \@gls@checkmkidxchars\@glo@sort
```

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

```
2505 \@gls@checkmkidxchars\@glo@prefix
```

Get the parent, if one exists

```
2506 \edef\@glo@parent{\csname glo@#1@parent\endcsname}%
```

Write the information to the glossary file.

```
2507 \ifglxindy
```

Store using xindy syntax.

```
2508 \ifx\@glo@parent\@empty
```

Entry doesn't have a parent

```
2509 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
```

```
2510 (\string"\@glo@sort\string" %
```

```
2511 \string"\@glo@prefix\@glossaryentryfield{\@glo@esclabel}\string") %
```

```
2512 }%
```

```
2513 \else
```

Entry has a parent

```
2514 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
```

```
2515 \csname glo@\@glo@parent @index\endcsname
```

```
2516 (\string"\@glo@sort\string" %
```

```
2517 \string"\@glo@prefix\@glossarysubentryfield
```

```
2518 {\csname glo@#1@level\endcsname}{\@glo@esclabel}\string") %
```

```
2519 }%
```

```
2520 \fi
```

```
2521 \else
```

Store using makeindex syntax.

```
2522 \ifx\@glo@parent\@empty
```

Sanitize \@glo@prefix

```
2523 \@onelevel@sanitize\@glo@prefix
```

Entry doesn't have a parent

```
2524 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
```

```
2525 \@glo@sort\@gls@actualchar\@glo@prefix
```

```
2526 \@glossaryentryfield{\@glo@esclabel}%
```

```
2527 }%
```

```
2528 \else
```

Entry has a parent

```

2529 \expandafter\protected\xdef\csname glo@#1@index\endcsname{%
2530 \csname glo@\@glo@parent @index\endcsname\@gls@levelchar
2531 \@glo@sort\@gls@actualchar\@glo@prefix
2532 \@glossarysubentryfield
2533 {\csname glo@#1@level\endcsname}\@glo@esclabel}%
2534 }%
2535 \fi
2536 \fi
2537 }

```

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`

```

2538 \AtBeginDocument{%
2539 \ifpackageloaded{amsmath}%
2540 {\let\gls@ifnotmeasuring\@gls@ifnotmeasuring}%
2541 {}%
2542 }
2543 \newcommand*\@gls@ifnotmeasuring[1]{%
2544 \ifmeasuring@
2545 \else
2546 #1%
2547 \fi
2548 }
2549 \newcommand*\gls@ifnotmeasuring[1]{#1}

```

`\lspatchtabularx` Patch `\TX@trial` (as per David Carlisle's answer in <http://tex.stackexchange.com/a/94895>). This does nothing if `\TX@trial` hasn't been defined.

```

2550 \def\@gls@patchtabularx#1\hbox#2#3!!{%
2551 \def\TX@trial##1{#1\hbox{\let\glsunset\@gobble#2}#3}%
2552 }
2553 \newcommand*\glspatchtabularx{%
2554 \ifdef\TX@trial
2555 {%
2556 \expandafter\@gls@patchtabularx\TX@trial{##1}!!%
2557 \let\glspatchtabularx\relax
2558 }%
2559 {}%
2560 }

```

`\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.

```

2561 \newcommand*{\glsreset}[1]{%
2562   \gls@ifnotmeasuring
2563   {%
2564     \glsdoifexists{#1}%
2565     {%
2566       \@glsreset{#1}%
2567     }%
2568   }%
2569 }

```

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

```

2570 \newcommand*{\glslocalreset}[1]{%
2571   \gls@ifnotmeasuring
2572   {%
2573     \glsdoifexists{#1}%
2574     {%
2575       \@glslocalreset{#1}%
2576     }%
2577   }%
2578 }

```

`\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.

```

2579 \newcommand*{\glsunset}[1]{%
2580   \gls@ifnotmeasuring
2581   {%
2582     \glsdoifexists{#1}%
2583     {%
2584       \@glsunset{#1}%
2585     }%
2586   }%
2587 }

```

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

```

2588 \newcommand*{\glslocalunset}[1]{%
2589   \gls@ifnotmeasuring
2590   {%
2591     \glsdoifexists{#1}%
2592     {%
2593       \@glslocalunset{#1}%
2594     }%
2595   }%
2596 }

```

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

```

2597 \newcommand*{\@glslocalunset}{\@glslocalunset}

```


`@@glslocalunset` Local unset without checks.

```

2598 \newcommand*{\@@glslocalunset}[1]{%
2599   \expandafter\let\csname ifglo@glsdetoklabel{#1}@flag\endcsname\iftrue
2600 }
```

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

```

2601 \newcommand*{\@glsunset}{\@@glsunset}
```

`\@@glsunset` Global unset without checks.

```

2602 \newcommand*{\@@glsunset}[1]{%
2603   \expandafter\global\csname glo@glsdetoklabel{#1}@flagtrue\endcsname
2604 }
```

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

```

2605 \newcommand*{\@glslocalreset}{\@@glslocalreset}
```

`@@glslocalreset` Local reset without checks.

```

2606 \newcommand*{\@@glslocalreset}[1]{%
2607   \expandafter\let\csname ifglo@glsdetoklabel{#1}@flag\endcsname\iffalse
2608 }
```

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

```

2609 \newcommand*{\@glsreset}{\@@glsreset}
```

`\@@glsreset` Global reset without checks.

```

2610 \newcommand*{\@@glsreset}[1]{%
2611   \expandafter\global\csname glo@glsdetoklabel{#1}@flagfalse\endcsname
2612 }
```

Reset all entries for the named glossaries (supplied in a comma-separated list). Syntax:

```
\glsresetall[<glossary-list>]
```

`\glsresetall`

```

2613 \newcommand*{\glsresetall}[1][\@glo@types]{%
2614   \forallglsentries[#1]{\@glsentry}%
2615   {%
2616     \glsreset{\@glsentry}%
2617   }%
2618 }
```

As above, but with only a local effect:

```

\lslocalresetall
2619 \newcommand*{\glslocalresetall}[1][\@glo@types]{%
2620   \forallglsentries[#1]{\@glsentry}%
2621   {%
2622     \glslocalreset{\@glsentry}%
2623   }%
2624 }
```

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

`\glsunsetall`

```
2625 \newcommand*{\glsunsetall}[1][\@glo@types]{%
2626   \forallglsentries[#1]{\@glsentry}%
2627   {%
2628     \glsunset{\@glsentry}%
2629   }%
2630 }
```

As above, but with only a local effect:

`\glslocalunsetall`

```
2631 \newcommand*{\glslocalunsetall}[1][\@glo@types]{%
2632   \forallglsentries[#1]{\@glsentry}%
2633   {%
2634     \glslocalunset{\@glsentry}%
2635   }%
2636 }
```

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.

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

```
2637 \newcommand*{\@newglossaryentry@defcounters}{%
2638   \csdef{glo@\@glo@label @currcount}{0}%
2639   \csdef{glo@\@glo@label @prevcount}{0}%
2640 }
```

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

```
2641 \newcommand*{\glsenableentrycount}{%
```

Enable new entry fields.

```
2642   \let\@newglossaryentry@defcounters\@newglossaryentry@defcounters
```

Disable `\newglossaryentry` in the document environment.

```
2643   \renewcommand*{\gls@defdocnewglossaryentry}{%
2644     \renewcommand*{\newglossaryentry[2]{%
2645       \PackageError{glossaries}{\string\newglossaryentry\space
2646         may only be used in the preamble when entry counting has
```

```

2647     been activated}{If you use \string\glsenableentrycount\space
2648     you must place all entry definitions in the preamble not in
2649     the document environment}%
2650 }%
2651 }%

```

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

```

2652 \newcommand*\glsentrycurrcount}[1]{%
2653   \ifcsundef{glo@\glsdetoklabel{##1}@currcount}%
2654   {0}{\@gls@entry@field{##1}{currcount}}}%
2655 }%
2656 \newcommand*\glsentryprevcount}[1]{%
2657   \ifcsundef{glo@\glsdetoklabel{##1}@prevcount}%
2658   {0}{\@gls@entry@field{##1}{prevcount}}}%
2659 }%

```

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

```

2660 \renewcommand*\@glsunset}[1]{%
2661   \@@glsunset{##1}%
2662   \@gls@increment@currcount{##1}%
2663 }%
2664 \renewcommand*\@glslocalunset}[1]{%
2665   \@@glslocalunset{##1}%
2666   \@gls@local@increment@currcount{##1}%
2667 }%
2668 \renewcommand*\@glsreset}[1]{%
2669   \@@glsreset{##1}%
2670   \csgdef{glo@\glsdetoklabel{##1}@currcount}{0}%
2671 }%
2672 \renewcommand*\@glslocalreset}[1]{%
2673   \@@glslocalreset{##1}%
2674   \csdef{glo@\glsdetoklabel{##1}@currcount}{0}%
2675 }%

```

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.)

```

2676 \def\@cgl@s@##1##2[##3]{%
2677   \ifnum\glsentryprevcount{##2}=1\relax
2678     \cgl@sformat{##2}{##3}%
2679     \glsunset{##2}%
2680   \else
2681     \@gls@{##1}{##2}[##3]%
2682   \fi
2683 }%

```

Similarly for the analogous commands. No case change plural:

```

2684 \def\@cgl spl@##1##2[##3]{%
2685   \ifnum\glsentryprevcount{##2}=1\relax
2686     \cgl splformat{##2}{##3}%
2687     \glsunset{##2}%

```

```

2688 \else
2689 \cGlspl@{##1}{##2}[##3]%
2690 \fi
2691 }%

```

First letter uppercase singular:

```

2692 \def\cGls@##1##2[##3]{%
2693 \ifnum\glsentryprevcount{##2}=1\relax
2694 \cGlsformat{##2}{##3}%
2695 \glsunset{##2}%
2696 \else
2697 \cGls@{##1}{##2}[##3]%
2698 \fi
2699 }%

```

First letter uppercase plural:

```

2700 \def\cGlspl@##1##2[##3]{%
2701 \ifnum\glsentryprevcount{##2}=1\relax
2702 \cGlsplformat{##2}{##3}%
2703 \glsunset{##2}%
2704 \else
2705 \cGlspl@{##1}{##2}[##3]%
2706 \fi
2707 }%

```

Write information to aux file at the end of the document

```

2708 \AtEndDocument{\cGls@write@entrycounts}%

```

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

```

2709 \renewcommand*\cGls@entry@count[2]{%
2710 \csxdef{glo@cGlsdetoklabel{##1}@prevcount}{##2}%
2711 }%

```

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

```

2712 \let\glsenableentrycount\relax
2713 }
2714 \@onlypreamble\glsenableentrycount

```

ement@currcount

```

2715 \newcommand*\cGls@increment@currcount[1]{%
2716 \csxdef{glo@cGlsdetoklabel{##1}@currcount}{%
2717 \number\numexpr\glsentrycurrcount{##1}+1}%
2718 }

```

ement@currcount

```

2719 \newcommand*\cGls@local@increment@currcount[1]{%
2720 \csedef{glo@cGlsdetoklabel{##1}@currcount}{%
2721 \number\numexpr\glsentrycurrcount{##1}+1}%
2722 }

```

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.)

```
2723 \newcommand*{\@gls@write@entrycounts}{%
2724   \immediate\write\@auxout
2725     {\string\providecommand*{\string\@gls@entry@count}[2]{}}%
2726   \forallglsentries{\@glsentry}{%
2727     \ifglsused{\@glsentry}%
2728     {\immediate\write\@auxout
2729       {\string\@gls@entry@count{\@glsentry}{\glsentrycurrcount{\@glsentry}}}%
2730     }%
2731   }%
2732 }
```

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

```
2733 \newcommand*{\@gls@entry@count}[2]{}
```

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

```
2734 \newrobustcmd*{\cgl}{\@gls@hyp@opt\@cgl}
```

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

```
2735 \newcommand*{\@cgl}[2][ ]{%
2736   \new@ifnextchar[{\@cgl@{#1}{#2}}{\@cgl@{#1}{#2}[ ]}%
2737 }
```

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

```
2738 \def\@cgl@#1#2[#3]{%
2739   \GlossariesWarning{\string\cgl\space is defaulting to
2740     \string\gls\space since you haven't enabled entry counting}%
2741   \@gls@{#1}{#2}[#3]%
2742 }
```

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

```
2743 \newcommand*{\cglformat}[2]{%
2744   \ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}#2%
2745 }
```

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

```
2746 \newrobustcmd*{\cGl}{\@gls@hyp@opt\@cGl}
```

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

```
2747 \newcommand*{\@cGl}[2][ ]{%
```

```

2748 \new@ifnextchar[{\@cGls@{#1}{#2}}{\@cGls@{#1}{#2} []}%
2749 }

```

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

```

2750 \def\@cGls@#1#2[#3]{%
2751 \GlossariesWarning{\string\cGls\space is defaulting to
2752 \string\Gls\space since you haven't enabled entry counting}%
2753 \@Gls@{#1}{#2}[#3]%
2754 }

```

`\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.

```

2755 \newcommand*{\cGlsformat}[2]{%
2756 \ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}#2%
2757 }

```

`\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.)

```

2758 \newrobustcmd*{\cglsp1}{\@gls@hyp@opt\@cglsp1}

```

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

```

2759 \newcommand*{\@cglsp1}[2] []{%
2760 \new@ifnextchar[{\@cglsp1@{#1}{#2}}{\@cglsp1@{#1}{#2} []}%
2761 }

```

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

```

2762 \def\@cglsp1@#1#2[#3]{%
2763 \GlossariesWarning{\string\cglsp1\space is defaulting to
2764 \string\glsp1\space since you haven't enabled entry counting}%
2765 \@glsp1@{#1}{#2}[#3]%
2766 }

```

`\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.

```

2767 \newcommand*{\cglsp1format}[2]{%
2768 \ifglshaslong{#1}{\glsp1entrylongpl{#1}}{\glsp1entryfirstplural{#1}}#2%
2769 }

```

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

```

2770 \newrobustcmd*{\cGlspl}{\@gls@hyp@opt\@cGlspl}

```

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

```

2771 \newcommand*{\@cGlspl}[2] []{%
2772 \new@ifnextchar[{\@cGlspl@{#1}{#2}}{\@cGlspl@{#1}{#2} []}%
2773 }

```

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

```
2774 \def\@cGlspl@#1#2[#3]{%
2775 \GlossariesWarning{\string\cGlspl\space is defaulting to
2776 \string\Glspl\space since you haven't enabled entry counting}%
2777 \@Glspl@{#1}{#2}[#3]%
2778 }
```

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

```
2779 \newcommand*{\cGlsplformat}[2]{%
2780 \ifglshaslong{#1}{\Glsentrylongpl{#1}}{\Glsentryfirstplural{#1}}#2%
2781 }
```

1.10 Loading files containing glossary entries

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

`\loadglsentries[<type>]{<filename>}`

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

`\loadglsentries`

```
2782 \newcommand*{\loadglsentries}[2][\@gls@default]{%
2783 \let\@gls@default\glsdefaulttype
2784 \def\glsdefaulttype{#1}\input{#2}%
2785 \let\glsdefaulttype\@gls@default
2786 }
```

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

```
2787 \@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”.

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

`\glstextformat`

```
2788 \newcommand*{\glstextformat}[1]{#1}
```

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

```
2789 \newcommand*{\glentryfmt}{%
2790   \@gls@default@entryfmt\glsdisplayfirst\glsdisplay
2791 }
```

Format that provides backwards compatibility:

```
2792 \newcommand*{\@gls@default@entryfmt}[2]{%
2793   \ifdefempty\glscustomtext
2794     {%
2795       \glsifplural
2796       {%
```

Plural form

```
2797       \glscapscase
2798       {%
```

Don't adjust case

```
2799       \ifglsused\glslabel
2800       {%
```

Subsequent use

```
2801       #2{\glentryplural{\glslabel}}%
2802       {\glentrydescplural{\glslabel}}%
2803       {\glentrysymbolplural{\glslabel}}{\glsinsert}%
2804     }%
2805     {%
```

First use

```
2806       #1{\glentryfirstplural{\glslabel}}%
2807       {\glentrydescplural{\glslabel}}%
2808       {\glentrysymbolplural{\glslabel}}{\glsinsert}%
2809     }%
2810   }%
2811   {%
```

Make first letter upper case

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

Subsequent use. (Expansion was used in version 3.07 and below in case the name wasn't the first thing to be displayed, but now the user can sort out the upper casing in `\defglentryfmt`, which avoids the issues caused by fragile commands.)

```
2814       \ifbool{glscompatible-3.07}%
2815       {%
2816       \protected@edef\glo@etext{%
```



```

2817         #2{\glsentryplural{\glslabel}}%
2818         {\glsentrydescplural{\glslabel}}%
2819         {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2820     \xmakefirstuc\@glo@etext
2821 }%
2822 {%
2823     #2{\Glsentryplural{\glslabel}}%
2824     {\glsentrydescplural{\glslabel}}%
2825     {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2826 }%
2827 }%
2828 {%

```

First use

```

2829     \ifbool{glscompatible-3.07}%
2830     {%
2831         \protected@edef\@glo@etext{%
2832             #1{\glsentryfirstplural{\glslabel}}%
2833             {\glsentrydescplural{\glslabel}}%
2834             {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2835         \xmakefirstuc\@glo@etext
2836     }%
2837     {%
2838         #1{\Glsentryfirstplural{\glslabel}}%
2839         {\glsentrydescplural{\glslabel}}%
2840         {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2841     }%
2842 }%
2843 }%
2844 {%

```

Make all upper case

```

2845     \ifglsused\glslabel
2846     {%

```

Subsequent use

```

2847         \mfirstucMakeUppercase{#2{\glsentryplural{\glslabel}}%
2848         {\glsentrydescplural{\glslabel}}%
2849         {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2850     }%
2851     {%

```

First use

```

2852         \mfirstucMakeUppercase{#1{\glsentryfirstplural{\glslabel}}%
2853         {\glsentrydescplural{\glslabel}}%
2854         {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2855     }%
2856 }%
2857 }%
2858 {%

```

Singular form

```
2859 \glscapscase
2860 {%
```

Don't adjust case

```
2861 \ifglused\glslabel
2862 {%
```

Subsequent use

```
2863 #2{\glentrytext{\glslabel}}%
2864 {\glentrydesc{\glslabel}}%
2865 {\glentrysymbol{\glslabel}}{\glinsert}%
2866 }%
2867 {%
```

First use

```
2868 #1{\glentryfirst{\glslabel}}%
2869 {\glentrydesc{\glslabel}}%
2870 {\glentrysymbol{\glslabel}}{\glinsert}%
2871 }%
2872 }%
2873 {%
```

Make first letter upper case

```
2874 \ifglused\glslabel
2875 {%
```

Subsequent use

```
2876 \ifbool{glcompatible-3.07}%
2877 {%
2878 \protected@edef\@glo@etext{%
2879 #2{\glentrytext{\glslabel}}%
2880 {\glentrydesc{\glslabel}}%
2881 {\glentrysymbol{\glslabel}}{\glinsert}}%
2882 \xmakefirstuc\@glo@etext
2883 }%
2884 {%
2885 #2{\Glsentrytext{\glslabel}}%
2886 {\glentrydesc{\glslabel}}%
2887 {\glentrysymbol{\glslabel}}{\glinsert}%
2888 }%
2889 }%
2890 {%
```

First use

```
2891 \ifbool{glcompatible-3.07}%
2892 {%
2893 \protected@edef\@glo@etext{%
2894 #1{\glentryfirst{\glslabel}}%
2895 {\glentrydesc{\glslabel}}%
2896 {\glentrysymbol{\glslabel}}{\glinsert}}%
2897 \xmakefirstuc\@glo@etext
```

```

2898         }%
2899         {%
2900         #1{\Glsentryfirst{\glslabel}}%
2901         {\Glsentrydesc{\glslabel}}%
2902         {\Glsentrysymbol{\glslabel}}{\Glsinsert}}%
2903     }%
2904 }%
2905 }%
2906 {%

```

Make all upper case

```

2907     \ifglsused\glslabel
2908     {%

```

Subsequent use

```

2909         \mfirstucMakeUppercase{#2{\Glsentrytext{\glslabel}}%
2910         {\Glsentrydesc{\glslabel}}%
2911         {\Glsentrysymbol{\glslabel}}{\Glsinsert}}%
2912     }%
2913     {%

```

First use

```

2914         \mfirstucMakeUppercase{#1{\Glsentryfirst{\glslabel}}%
2915         {\Glsentrydesc{\glslabel}}%
2916         {\Glsentrysymbol{\glslabel}}{\Glsinsert}}%
2917     }%
2918 }%
2919 }%
2920 }%
2921 {%

```

Custom text provided in \glsdisp

```

2922     \ifglsused{\glslabel}%
2923     {%

```

Subsequent use

```

2924         #2{\glscustomtext}%
2925         {\Glsentrydesc{\glslabel}}%
2926         {\Glsentrysymbol{\glslabel}}{}%
2927     }%
2928     {%

```

First use

```

2929         #1{\glscustomtext}%
2930         {\Glsentrydesc{\glslabel}}%
2931         {\Glsentrysymbol{\glslabel}}{}%
2932     }%
2933 }%
2934 }

```

`\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.

```

2935 \newcommand*{\glsgenentryfmt}{%
2936   \ifdefempty\glscustomtext
2937   {%
2938     \glssifplural
2939     {%

```

Plural form

```

2940     \glscapscase
2941     {%

```

Don't adjust case

```

2942     \ifglssused\glslabel
2943     {%

```

Subsequent use

```

2944     \glssentryplural{\glslabel}\glssinsert
2945     }%
2946     {%

```

First use

```

2947     \glssentryfirstplural{\glslabel}\glssinsert
2948     }%
2949     }%
2950     {%

```

Make first letter upper case

```

2951     \ifglssused\glslabel
2952     {%

```

Subsequent use.

```

2953     \Glssentryplural{\glslabel}\glssinsert
2954     }%
2955     {%

```

First use

```

2956     \Glssentryfirstplural{\glslabel}\glssinsert
2957     }%
2958     }%
2959     {%

```

Make all upper case

```

2960     \ifglssused\glslabel
2961     {%

```

Subsequent use

```

2962     \mfirstucMakeUppercase
2963     {\glssentryplural{\glslabel}\glssinsert}%
2964     }%
2965     {%

```

First use

```

2966     \mfirstucMakeUppercase
2967     {\glssentryfirstplural{\glslabel}\glssinsert}%

```

2968 }%

2969 }%

2970 }%

2971 {%

Singular form

2972 \glscapscase

2973 {%

Don't adjust case

2974 \ifglused\glslabel

2975 {%

Subsequent use

2976 \glentrytext{\glslabel}\glinsert

2977 }%

2978 {%

First use

2979 \glentryfirst{\glslabel}\glinsert

2980 }%

2981 }%

2982 {%

Make first letter upper case

2983 \ifglused\glslabel

2984 {%

Subsequent use

2985 \Glentrytext{\glslabel}\glinsert

2986 }%

2987 {%

First use

2988 \Glentryfirst{\glslabel}\glinsert

2989 }%

2990 }%

2991 {%

Make all upper case

2992 \ifglused\glslabel

2993 {%

Subsequent use

2994 \mfirstucMakeUppercase{\glentrytext{\glslabel}\glinsert}%

2995 }%

2996 {%

First use

2997 \mfirstucMakeUppercase{\glentryfirst{\glslabel}\glinsert}%

2998 }%

2999 }%

3000 }%

```

3001 }%
3002 {%
    Custom text provided in \glsdisp. (The insert is most likely to be empty at this point.)
3003 \glscustomtext\glsinsert
3004 }%
3005 }

```

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

```

3006 \newcommand*{\glsngenacfmt}{%
3007 \ifdefempty\glscustomtext
3008 {%
3009 \ifglsused\glslabel
3010 {%

```

Subsequent use:

```

3011 \glsifplural
3012 {%

```

Subsequent plural form:

```

3013 \glscapscase
3014 {%

```

Subsequent plural form, don't adjust case:

```

3015 \acronymfont{\glsentryshortpl{\glslabel}}\glsinsert
3016 }%
3017 {%

```

Subsequent plural form, make first letter upper case:

```

3018 \acronymfont{\Glsentryshortpl{\glslabel}}\glsinsert
3019 }%
3020 {%

```

Subsequent plural form, all caps:

```

3021 \mfirstucMakeUppercase
3022 {\acronymfont{\glsentryshortpl{\glslabel}}\glsinsert}%
3023 }%
3024 }%
3025 {%

```

Subsequent singular form

```

3026 \glscapscase
3027 {%

```

Subsequent singular form, don't adjust case:

```

3028 \acronymfont{\glsentryshort{\glslabel}}\glsinsert
3029 }%
3030 {%

```

Subsequent singular form, make first letter upper case:

```

3031 \acronymfont{\Glsentryshort{\glslabel}}\glsinsert
3032 }%
3033 {%

```

Subsequent singular form, all caps:

```
3034      \mfirstucMakeUppercase
3035      {\acronymfont{\glentryshort{\glslabel}}\glsinsert}%
3036      }%
3037      }%
3038      }%
3039      {%
```

First use:

```
3040      \glsifplural
3041      {%
```

First use plural form:

```
3042      \glscapscase
3043      {%
```

First use plural form, don't adjust case:

```
3044      \genplacrfullformat{\glslabel}{\glsinsert}%
3045      }%
3046      {%
```

First use plural form, make first letter upper case:

```
3047      \Genplacrfullformat{\glslabel}{\glsinsert}%
3048      }%
3049      {%
```

First use plural form, all caps:

```
3050      \mfirstucMakeUppercase
3051      {\genplacrfullformat{\glslabel}{\glsinsert}}%
3052      }%
3053      }%
3054      {%
```

First use singular form

```
3055      \glscapscase
3056      {%
```

First use singular form, don't adjust case:

```
3057      \genacrfullformat{\glslabel}{\glsinsert}%
3058      }%
3059      {%
```

First use singular form, make first letter upper case:

```
3060      \Genacrfullformat{\glslabel}{\glsinsert}%
3061      }%
3062      {%
```

First use singular form, all caps:

```
3063      \mfirstucMakeUppercase
3064      {\genacrfullformat{\glslabel}{\glsinsert}}%
3065      }%
3066      }%
3067      }%
```

```

3068 }%
3069 {%
    User supplied text.
3070 \glscustomtext
3071 }%
3072 }

```

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

The full format used by \glsgenacfmt (singular).

```

3073 \newcommand*{\genacrfullformat}[2]{%
3074   \glentrylong{#1}#2\space
3075   (\protect\firstacronymfont{\glentryshort{#1}})%
3076 }

```

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

As above but makes the first letter upper case.

```

3077 \newcommand*{\Genacrfullformat}[2]{%
3078   \protected@edef\gls@text{\genacrfullformat{#1}{#2}}%
3079   \xmakefirstuc\gls@text
3080 }

```

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

The full format used by \glsgenacfmt (plural).

```

3081 \newcommand*{\genplacrfullformat}[2]{%
3082   \glentrylongpl{#1}#2\space
3083   (\protect\firstacronymfont{\glentryshortpl{#1}})%
3084 }

```

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

As above but makes the first letter upper case.

```

3085 \newcommand*{\Genplacrfullformat}[2]{%
3086   \protected@edef\gls@text{\genplacrfullformat{#1}{#2}}%
3087   \xmakefirstuc\gls@text
3088 }

```

glsdisplayfirst Deprecated. Kept for backward compatibility.

```

3089 \newcommand*{\glsdisplayfirst}[4]{#1#4}

```


`\glsdisplay` Deprecated. Kept for backward compatibility.

```
3090 \newcommand*{\glsdisplay}[4]{#1#4}
```

`\defglsdisplay` Deprecated. Kept for backward compatibility.

```
3091 \newcommand*{\defglsdisplay}[2][\glsdefaulttype]{%
3092   \GlossariesWarning{\string\defglsdisplay\space is now obsolete.^^J
3093   Use \string\defglsentryfmt\space instead}%
3094   \expandafter\def\csname gls@#1@display\endcsname##1##2##3##4{#2}%
3095   \edef\@gls@doentrydef{%
3096     \noexpand\defglsentryfmt[#1]{%
3097       \noexpand\ifcsdef{gls@#1@displayfirst}%
3098       {%
3099         \noexpand\@gls@default@entryfmt
3100         {\noexpand\csuse{gls@#1@displayfirst}}%
3101         {\noexpand\csuse{gls@#1@display}}%
3102       }%
3103       {%
3104         \noexpand\@gls@default@entryfmt
3105         {\noexpand\glsdisplayfirst}%
3106         {\noexpand\csuse{gls@#1@display}}%
3107       }%
3108     }%
3109   }%
3110   \@gls@doentrydef
3111 }
```

`glsdisplayfirst` Deprecated. Kept for backward compatibility.

```
3112 \newcommand*{\defglsdisplayfirst}[2][\glsdefaulttype]{%
3113   \GlossariesWarning{\string\defglsdisplayfirst\space is now obsolete.^^J
3114   Use \string\defglsentryfmt\space instead}%
3115   \expandafter\def\csname gls@#1@displayfirst\endcsname##1##2##3##4{#2}%
3116   \edef\@gls@doentrydef{%
3117     \noexpand\defglsentryfmt[#1]{%
3118       \noexpand\ifcsdef{gls@#1@display}%
3119       {%
3120         \noexpand\@gls@default@entryfmt
3121         {\noexpand\csuse{gls@#1@displayfirst}}%
3122         {\noexpand\csuse{gls@#1@display}}%
3123       }%
3124       {%
3125         \noexpand\@gls@default@entryfmt
3126         {\noexpand\csuse{gls@#1@displayfirst}}%
3127         {\noexpand\glsdisplay}%
3128       }%
3129     }%
3130   }%
3131   \@gls@doentrydef
3132 }
```

Links to glossary entries

The links to glossary entries all have a first optional argument that can be used to change the format and counter of the associated entry number. Except for `\glslink` and `\glsdisp`, the commands like `\gls` have a final optional argument that can be used to insert additional text in the link (this will usually be appended, but can be redefined using `\defentryfmt`). It goes against the \TeX norm to have an optional argument after the mandatory arguments, but it makes more sense to write, say, `\gls{label}[‘s]` rather than, say, `\gls[append=‘s]{label}`. Since these control sequences are defined to include the final square bracket, spaces will be ignored after them. This is likely to lead to confusion as most users would not expect, say, `\gls{<label>}` to ignore following spaces, so `\new@ifnextchar` from the package is required.

The following keys can be used in the first optional argument. The counter key checks that the value is the name of a valid counter.

```
3133 \define@key{glslink}{counter}{%
3134   \ifcsundef{c@#1}%
3135   {%
3136     \PackageError{glossaries}%
3137     {There is no counter called ‘#1’}%
3138     {%
3139       The counter key should have the name of a valid counter
3140       as its value%
3141     }%
3142   }%
3143   {%
3144     \def\@gls@counter{#1}%
3145   }%
3146 }
```

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.

```
3147 \define@key{glslink}{format}{%
3148   \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.

```
3149 \define@boolkey{glslink}{hyper}[true]{}
```

Initialise hyper key.

```
3150 \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.

```
3151 \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.

```
3152 \newcommand*{\glslinkvar}[3]{#1}
```

`\glsifhyper` Now deprecated.

```
3153 \newcommand*{\glsifhyper}[2]{%
3154   \glslinkvar{#1}{#2}{#1}%
3155   \GlossariesWarning{\string\glsifhyper\space is deprecated. Did
3156   you mean \string\glsifhyperon\space or \string\glslinkvar?}%
3157 }
```

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

```
3158 \newcommand*{\@gls@hyp@opt}[1]{%
3159   \let\glslinkvar\@firstofthree
3160   \let\@gls@hyp@opt@cs#1\relax
3161   \@ifstar{\s@gls@hyp@opt}%
3162   {\@ifnextchar+{\@firstoftwo{\p@gls@hyp@opt}}{#1}}%
3163 }
```

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

```
3164 \newcommand*{\s@gls@hyp@opt}[1] []{%
3165   \let\glslinkvar\@secondofthree
3166   \@gls@hyp@opt@cs[hyper=false,#1]}
```

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

```
3167 \newcommand*{\p@gls@hyp@opt}[1] []{%
3168   \let\glslinkvar\@thirdofthree
3169   \@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`

```
3170 \newrobustcmd*{\glslink}{%  
3171 \@gls@hyp@opt\@gls@@link  
3172 }
```

`\@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.

```
3173 \newcommand*{\@gls@@link}[3][\@gls@link]{%  
3174 \glsdoifexistsordo{#2}%  
3175 {%  
3176 \let\do@gls@link@checkfirsthyper\relax  
3177 \@gls@link[#1]{#2}{#3}%  
3178 }%
```

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

```
3179 \glstextformat{#3}%  
3180 }%  
  
3181 \glspostlinkhook  
3182 }
```

`glspostlinkhook`

```
3183 \newcommand*{\glspostlinkhook}{}
```

`checkfirsthyper`

Check for first use and switch off hyper key if hyperlink not wanted. (Should be off if first use and `hyper=false` is on or if first use and both the entry is in an acronym list and the `acrfootnote` setting is on.) This assumes the glossary type is stored in `\glstype` and the label is stored in `\glslabel`.

```
3184 \newcommand*{\@gls@link@checkfirsthyper}{%  
3185 \ifglsused{\glslabel}%  
3186 {%  
3187 }%  
3188 {%  
3189 \gls@checkisacronymlist\glstype  
3190 \ifglshyperfirst  
3191 \if@glsisacronymlist  
3192 \ifglsacrfootnote  
3193 \KV@glslink@hyperfalse  
3194 \fi  
3195 \fi  
3196 \else  
3197 \KV@glslink@hyperfalse  
3198 \fi  
3199 }%
```

Allow user to hook into this

```
3200 \glslinkcheckfirsthyperhook  
3201 }
```

`checkfirsthyperhook` Allow used to hook into the `\@gls@link@checkfirsthyper` macro
3202 `\newcommand*{\glslinkcheckfirsthyperhook}{}`

`linkpostsetkeys`
3203 `\newcommand*{\glslinkpostsetkeys}{}`

`\glsifhyperon` Check the value of the hyper key:
3204 `\newcommand{\glsifhyperon}[2]{\ifKV@glslink@hyper#1\else#2\fi}`

`disablehyperinlist` Disable hyperlink if in the “nohyper” list.
3205 `\newcommand*{\do@glssdisablehyperinlist}{%`
3206 `\expandafter\DTLifinlist\expandafter{\gls@type}{\@gls@nohyperlist}%`
3207 `{\KV@glslink@hyperfalse}{}}%`
3208 `}`

`let@glslink@opts` Hook to set default options for `\@glslink`.
3209 `\newcommand*{\@gls@setdefault@glslink@opts}{}`

`\@gls@link`
3210 `\def\@gls@link[#1]#2#3{%`
Inserting `\leavevmode` suggested by Donald Arseneau (avoids problem with tabularx).
3211 `\leavevmode`
3212 `\edef\glslabel{\glsdetoklabel{#2}}%`
Save options in `\@gls@link@opts` and label in `\@gls@link@label`
3213 `\def\@gls@link@opts{#1}%`
3214 `\let\@gls@link@label\glslabel`
3215 `\def\@glsnumberformat{glsnumberformat}%`
3216 `\edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%`
If this is in one of the “nohypertypes” glossaries, suppress the hyperlink by default
3217 `\edef\gls@type{\csname glo@\glslabel @type\endcsname}%`
Save original setting
3218 `\let\org@ifKV@glslink@hyper\ifKV@glslink@hyper`
Set defaults:
3219 `\@gls@setdefault@glslink@opts`
Switch off hyper setting if the glossary type has been identified in nohyperlist.
3220 `\do@glssdisablehyperinlist`
Macros must set this before calling `\@gls@link`. The commands that check the first use flag should set this to `\@gls@link@checkfirsthyper` otherwise it should be set to `\relax`.
3221 `\do@gls@link@checkfirsthyper`
3222 `\setkeys{glslink}{#1}%`
Add a hook for the user to customise things after the keys have been set.
3223 `\glslinkpostsetkeys`

```

Store the entry's counter in \theglentrycounter
3224 \@gls@saveentrycounter

Define sort key if necessary:
3225 \@gls@setsort{\glslabel}%

(De-tok'ing done by \@do@wrglossary)
3226 \@do@wrglossary{#2}%
3227 \ifKV@glslink@hyper
3228 \@glslink{\glolinkprefix\glslabel}{\glstextformat{#3}}%
3229 \else

3230 \glsdonohyperlink{\glolinkprefix\glslabel}{\glstextformat{#3}}%
3231 \fi

Restore original setting
3232 \let\ifKV@glslink@hyper\org@ifKV@glslink@hyper
3233 }

\glolinkprefix
3234 \newcommand*{\glolinkprefix}{glo:}

glsentrycounter Set default value of entry counter
3235 \def\glsentrycounter{\glscounter}%

saveentrycounter Need to check if using equation counter in align environment:
3236 \newcommand*{\@gls@saveentrycounter}{%
3237 \def\@gls@Hcounter{}}%

Are we using equation counter?
3238 \ifthenelse{\equal{\@gls@counter}{equation}}{%
3239 {

If we're in align environment, \xatlevel@ will be defined. (Can't test for \@currentenv as
may be inside an inner environment.)
3240 \ifcsundef{xatlevel@}%
3241 {%
3242 \edef\theglentrycounter{\expandafter\noexpand
3243 \csname the\@gls@counter\endcsname}%
3244 }%
3245 {%
3246 \ifx\xatlevel@\@empty
3247 \edef\theglentrycounter{\expandafter\noexpand
3248 \csname the\@gls@counter\endcsname}%
3249 \else
3250 \savecounters@
3251 \advance\c@equation by 1\relax
3252 \edef\theglentrycounter{\csname the\@gls@counter\endcsname}%

```

Check if hyperref version of this counter

```

3253     \ifcsundef{theH\@gls@counter}%
3254     {%
3255         \def\@gls@Hcounter{\theglsentrycounter}%
3256     }%
3257     {%
3258         \def\@gls@Hcounter{\csname theH\@gls@counter\endcsname}%
3259     }%
3260     \protected@edef\theHglentrycounter{\@gls@Hcounter}%
3261     \restorecounters@
3262 \fi
3263 }%
3264 }%
3265 {%

```

Not using equation counter so no special measures:

```

3266     \edef\theglsentrycounter{\expandafter\noexpand
3267         \csname the\@gls@counter\endcsname}%
3268 }%

```

Check if hyperref version of this counter

```

3269 \ifx\@gls@Hcounter\@empty
3270 \ifcsundef{theH\@gls@counter}%
3271 {%
3272     \def\theHglentrycounter{\theglsentrycounter}%
3273 }%
3274 {%
3275     \protected@edef\theHglentrycounter{\expandafter\noexpand
3276         \csname theH\@gls@counter\endcsname}%
3277 }%
3278 \fi
3279 }

```

`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.

```

3280 \def\@set@glo@numformat#1#2#3#4{%
3281     \expandafter\@glo@check@mkidxrangechar#3\@nil
3282     \protected@edef#1{%
3283         \@glo@prefix setentrycounter[#4]{#2}%
3284         \expandafter\string\csname\@glo@suffix\endcsname
3285     }%
3286     \@gls@checkmkidxchars#1%
3287 }

```

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.

```

3288 \def\@glo@check@mkidxrangechar#1#2\@nil{%
3289 \if#1(\relax
3290   \def\@glo@prefix{()%
3291   \if\relax#2\relax
3292     \def\@glo@suffix{glsnumberformat}%
3293   \else
3294     \def\@glo@suffix{#2}%
3295   \fi
3296 \else
3297   \if#1)\relax
3298     \def\@glo@prefix{}}}%
3299   \if\relax#2\relax
3300     \def\@glo@suffix{glsnumberformat}%
3301   \else
3302     \def\@glo@suffix{#2}%
3303   \fi
3304 \else
3305   \def\@glo@prefix{}\def\@glo@suffix{#1#2}%
3306 \fi
3307 \fi}

```

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

```

3308 \newcommand*{\@gls@escbsdq}[1]{%
3309   \def\@gls@checkedmkidx{%
3310     \let\gls@xdystring=#1\relax
3311     \@onelevel@sanitize\gls@xdystring
3312     \edef\do@gls@xdycheckbackslash{%
3313       \noexpand\@gls@xdycheckbackslash\gls@xdystring\noexpand\@nil
3314       \@backslashchar\@backslashchar\noexpand\null}%
3315     \do@gls@xdycheckbackslash
3316     \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdystring}%
3317     \def\@gls@checkedmkidx{%
3318       \expandafter\@gls@xdycheckquote\gls@xdystring\@nil""\null
3319       \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdystring}%

```

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

```

3320 \@for\@gls@tmp:=\gls@protected@pagefmts\do
3321 {%
3322   \edef\@gls@sanitized@tmp{\expandafter\@gobble\string\\expandonce\@gls@tmp}%
3323   \@onelevel@sanitize\@gls@sanitized@tmp
3324   \edef\gls@dosubst{%
3325     \noexpand\DTLsubstituteall\noexpand\gls@xdystring
3326     {\@gls@sanitized@tmp}{\expandonce\@gls@tmp}%
3327   }%
3328   \gls@dosubst
3329 }%

```

Assign to required control sequence

```

3330 \let#1=\gls@xdystring

```


3331 }

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

checkmkidxchars

```
3332 \newcommand{\@gls@checkmkidxchars}[1]{%
3333   \ifglxsindy
3334     \@gls@escbsdq{#1}%
3335   \else
3336     \def\@gls@checkedmkidx{%
3337       \expandafter\@gls@checkquote#1\@nil""\null
3338       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3339     \def\@gls@checkedmkidx{%
3340       \expandafter\@gls@checkescquote#1\@nil\\"\null
3341       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3342     \def\@gls@checkedmkidx{%
3343       \expandafter\@gls@checkescactual#1\@nil??\null
3344       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3345     \def\@gls@checkedmkidx{%
3346       \expandafter\@gls@checkactual#1\@nil??\null
3347       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3348     \def\@gls@checkedmkidx{%
3349       \expandafter\@gls@checkbar#1\@nil||\null
3350       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3351     \def\@gls@checkedmkidx{%
3352       \expandafter\@gls@checkescbar#1\@nil|||\null
3353       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3354     \def\@gls@checkedmkidx{%
3355       \expandafter\@gls@checklevel#1\@nil!!\null
3356       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3357     \fi
3358 }
```

Update the control sequence and strip trailing \@nil:

s@updatechecked

```
3359 \def\@gls@updatechecked#1\@nil#2{\def#2{#1}}
```

\@gls@tmpb Define temporary token

```
3360 \newtoks\@gls@tmpb
```

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

```
3361 \def\@gls@checkquote#1"#2"#3\null{%
3362   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3363   \toks@={#1}%
3364   \ifx\@nil#2\null
3365   \ifx\@nil#3\null
3366     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3367   \def\@gls@checkquote{\relax}%
3368   \else
```

```

3369 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3370 \@gls@quotechar\@gls@quotechar\@gls@quotechar\@gls@quotechar}%
3371 \def\@@gls@checkquote{\@gls@checkquote#3\null}%
3372 \fi
3373 \else
3374 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3375 \@gls@quotechar\@gls@quotechar}%
3376 \ifx\null#3\null
3377 \def\@@gls@checkquote{\@gls@checkquote#2""\null}%
3378 \else
3379 \def\@@gls@checkquote{\@gls@checkquote#2"#3\null}%
3380 \fi
3381 \fi
3382 \@@gls@checkquote
3383 }

```

s@checkescquote Do the same for \":

```

3384 \def\@gls@checkescquote#1\"#2\"#3\null{%
3385 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3386 \toks@={#1}%
3387 \ifx\null#2\null
3388 \ifx\null#3\null
3389 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3390 \def\@@gls@checkescquote{\relax}%
3391 \else
3392 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3393 \@gls@quotechar\string\"@gls@quotechar
3394 \@gls@quotechar\string\"@gls@quotechar}%
3395 \def\@@gls@checkescquote{\@gls@checkescquote#3\null}%
3396 \fi
3397 \else
3398 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3399 \@gls@quotechar\string\"@gls@quotechar}%
3400 \ifx\null#3\null
3401 \def\@@gls@checkescquote{\@gls@checkescquote#2\"\" \null}%
3402 \else
3403 \def\@@gls@checkescquote{\@gls@checkescquote#2\"#3\null}%
3404 \fi
3405 \fi
3406 \@@gls@checkescquote
3407 }

```

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

```

3408 \def\@gls@checkescactual#1\?#2\?#3\null{%
3409 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3410 \toks@={#1}%
3411 \ifx\null#2\null
3412 \ifx\null#3\null
3413 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%

```

```

3414 \def\@gls@checkescactual{\relax}%
3415 \else
3416 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3417 \gls@quotechar\string"\@gls@actualchar
3418 \gls@quotechar\string"\@gls@actualchar}%
3419 \def\@gls@checkescactual{\@gls@checkescactual#3\null}%
3420 \fi
3421 \else
3422 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3423 \gls@quotechar\string"\@gls@actualchar}%
3424 \ifx\null#3\null
3425 \def\@gls@checkescactual{\@gls@checkescactual#2\?\?\null}%
3426 \else
3427 \def\@gls@checkescactual{\@gls@checkescactual#2\?#3\null}%
3428 \fi
3429 \fi
3430 \@gls@checkescactual
3431 }

```

`\gls@checkescbar` Similarly for `\|`:

```

3432 \def\@gls@checkescbar#1\|#2\|#3\null{%
3433 \gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3434 \toks@={#1}%
3435 \ifx\null#2\null
3436 \ifx\null#3\null
3437 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3438 \def\@gls@checkescbar{\relax}%
3439 \else
3440 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3441 \gls@quotechar\string"\@gls@encapchar
3442 \gls@quotechar\string"\@gls@encapchar}%
3443 \def\@gls@checkescbar{\@gls@checkescbar#3\null}%
3444 \fi
3445 \else
3446 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3447 \gls@quotechar\string"\@gls@encapchar}%
3448 \ifx\null#3\null
3449 \def\@gls@checkescbar{\@gls@checkescbar#2\|\|\null}%
3450 \else
3451 \def\@gls@checkescbar{\@gls@checkescbar#2\|#3\null}%
3452 \fi
3453 \fi
3454 \@gls@checkescbar
3455 }

```

`\s@checkesclevel` Similarly for `\!`:

```

3456 \def\@gls@checkesclevel#1\!#2\!#3\null{%
3457 \gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3458 \toks@={#1}%

```

```

3459 \ifx\null#2\null
3460 \ifx\null#3\null
3461 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3462 \def\@gls@checkesclevel{\relax}%
3463 \else
3464 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3465 \@gls@quotechar\string"\@gls@levelchar
3466 \@gls@quotechar\string"\@gls@levelchar}%
3467 \def\@gls@checkesclevel{\@gls@checkesclevel#3\null}%
3468 \fi
3469 \else
3470 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3471 \@gls@quotechar\string"\@gls@levelchar}%
3472 \ifx\null#3\null
3473 \def\@gls@checkesclevel{\@gls@checkesclevel#2!!\null}%
3474 \else
3475 \def\@gls@checkesclevel{\@gls@checkesclevel#2!#3\null}%
3476 \fi
3477 \fi
3478 \@gls@checkesclevel
3479 }

```

\@gls@checkbar and for |:

```

3480 \def\@gls@checkbar#1|#2|#3\null{%
3481 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3482 \toks@={#1}%
3483 \ifx\null#2\null
3484 \ifx\null#3\null
3485 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3486 \def\@gls@checkbar{\relax}%
3487 \else
3488 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3489 \@gls@quotechar\@gls@encapchar\@gls@quotechar\@gls@encapchar}%
3490 \def\@gls@checkbar{\@gls@checkbar#3\null}%
3491 \fi
3492 \else
3493 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3494 \@gls@quotechar\@gls@encapchar}%
3495 \ifx\null#3\null
3496 \def\@gls@checkbar{\@gls@checkbar#2||\null}%
3497 \else
3498 \def\@gls@checkbar{\@gls@checkbar#2|#3\null}%
3499 \fi
3500 \fi
3501 \@gls@checkbar
3502 }

```

@gls@checklevel and for !:

```

3503 \def\@gls@checklevel#1!#2!#3\null{%

```

```

3504 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3505 \toks@={#1}%
3506 \ifx\null#2\null
3507   \ifx\null#3\null
3508     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3509     \def\@gls@checklevel{\relax}%
3510   \else
3511     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3512       \@gls@quotechar\@gls@levelchar\@gls@quotechar\@gls@levelchar}%
3513     \def\@gls@checklevel{\@gls@checklevel#3\null}%
3514   \fi
3515 \else
3516   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3517     \@gls@quotechar\@gls@levelchar}%
3518   \ifx\null#3\null
3519     \def\@gls@checklevel{\@gls@checklevel#2!!\null}%
3520   \else
3521     \def\@gls@checklevel{\@gls@checklevel#2!#3\null}%
3522   \fi
3523 \fi
3524 \@gls@checklevel
3525 }

```

gls@checkactual and for ?:

```

3526 \def\@gls@checkactual#1?#2?#3\null{%
3527   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3528   \toks@={#1}%
3529   \ifx\null#2\null
3530     \ifx\null#3\null
3531       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3532       \def\@gls@checkactual{\relax}%
3533     \else
3534       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3535         \@gls@quotechar\@gls@actualchar\@gls@quotechar\@gls@actualchar}%
3536       \def\@gls@checkactual{\@gls@checkactual#3\null}%
3537     \fi
3538   \else
3539     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3540       \@gls@quotechar\@gls@actualchar}%
3541     \ifx\null#3\null
3542       \def\@gls@checkactual{\@gls@checkactual#2??\null}%
3543     \else
3544       \def\@gls@checkactual{\@gls@checkactual#2?#3\null}%
3545     \fi
3546   \fi
3547   \@gls@checkactual
3548 }

```

s@xdycheckquote As before but for use with xindy

```

3549 \def\@gls@xdycheckquote#1"#2"#3\null{%
3550   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3551   \toks@={#1}%
3552   \ifx\null#2\null
3553     \ifx\null#3\null
3554       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3555       \def\@gls@xdycheckquote{\relax}%
3556     \else
3557       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3558         \string"\string"}%
3559       \def\@gls@xdycheckquote{\@gls@xdycheckquote#3\null}%
3560     \fi
3561   \else
3562     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3563       \string"}%
3564     \ifx\null#3\null
3565       \def\@gls@xdycheckquote{\@gls@xdycheckquote#2""\null}%
3566     \else
3567       \def\@gls@xdycheckquote{\@gls@xdycheckquote#2"#3\null}%
3568     \fi
3569   \fi
3570   \@gls@xdycheckquote
3571 }

```

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

```

3572 \edef\def\@gls@xdycheckbackslash{%
3573   \noexpand\def\noexpand\@gls@xdycheckbackslash##1\@backslashchar
3574     ##2\@backslashchar##3\noexpand\null{%
3575     \noexpand\@gls@tmpb=\noexpand\expandafter
3576       {\noexpand\@gls@checkedmkidx}%
3577     \noexpand\toks@={##1}%
3578     \noexpand\ifx\noexpand\null##2\noexpand\null
3579     \noexpand\ifx\noexpand\null##3\noexpand\null
3580       \noexpand\edef\noexpand\@gls@checkedmkidx{%
3581         \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
3582       \noexpand\def\noexpand\@gls@xdycheckbackslash{\relax}%
3583     \noexpand\else
3584       \noexpand\edef\noexpand\@gls@checkedmkidx{%
3585         \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
3586         \@backslashchar\@backslashchar\@backslashchar\@backslashchar}%
3587     \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3588       \noexpand\@gls@xdycheckbackslash##3\noexpand\null}%
3589     \noexpand\fi
3590   \noexpand\else
3591     \noexpand\edef\noexpand\@gls@checkedmkidx{%
3592       \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
3593       \@backslashchar\@backslashchar}%
3594   \noexpand\ifx\noexpand\null##3\noexpand\null
3595     \noexpand\def\noexpand\@gls@xdycheckbackslash{%

```

```

3596      \noexpand\@gls@xdycheckbackslash##2\@backslashchar
3597      \@backslashchar\noexpand\null}%
3598  \noexpand\else
3599      \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3600      \noexpand\@gls@xdycheckbackslash##2\@backslashchar
3601      ##3\noexpand\null}%
3602  \noexpand\fi
3603  \noexpand\fi
3604  \noexpand\@gls@xdycheckbackslash
3605  }%
3606  }

```

Now go ahead and define \@gls@xdycheckbackslash

```

3607 \def@gls@xdycheckbackslash

```

glsdohypertarget

```

3608 \newlength\gls@tmplen
3609 \newcommand*\glsdohypertarget}[2]{%
3610   \settoheight{\gls@tmplen}{#2}%
3611   \raisebox{\gls@tmplen}{\hypertarget{#1}{}}#2%
3612 }

```

glsdohyperlink

```

3613 \newcommand*\glsdohyperlink}[2]{%
3614   \hyperlink{#1}{#2}%
3615 }

```

glsdonohyperlink

```

3616 \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.

```

3617 \ifcsundef{hyperlink}%
3618 {%
3619   \let\@glslink\glsdonohyperlink
3620 }%
3621 {%
3622   \let\@glslink\glsdohyperlink
3623 }

```

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

```

3624 \ifcsundef{hypertarget}%
3625 {%
3626   \let\@glstarget\@secondoftwo
3627 }%
3628 {%
3629   \let\@glstarget\glsdohypertarget
3630 }

```

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

`\glsdisablehyper`

```
3631 \newcommand{\glsdisablehyper}{%
3632   \KV@glslink@hyperfalse
3633   \let\@glslink\glsdonohyperlink
3634   \let\@glstarget\@secondoftwo
3635 }
```

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

`\glsenablehyper`

```
3636 \newcommand{\glsenablehyper}{%
3637   \KV@glslink@hypertrue
3638   \let\@glslink\glsdohyperlink
3639   \let\@glstarget\glsdohypertarget
3640 }
```

Provide some convenience commands if not already defined:

```
3641 \providecommand{\@firstofthree}[3]{#1}
3642 \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`

```
3643 \newrobustcmd*{\gls}{\@gls@hyp@opt\@gls}
```

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

`\@gls`

```
3644 \newcommand*{\@gls}[2][ ]{%
3645   \new@ifnextchar[{\@gls@{#1}{#2}}{\@gls@{#1}{#2}[ ]}%
3646 }
```

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

```
3647 \def\@gls@#1#2[#3]{%
3648   \glsdoifexists{#2}%
3649   {%
3650     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
```



```

3651 \let\glsifplural\@secondoftwo
3652 \let\glschapscase\@firstofthree
3653 \let\glscustomtext\@empty
3654 \def\glsinsert{#3}%

```

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

```

3655 \def\@glo@text{\csname gls@\gls@type @entryfmt\endcsname}%

```

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

```

3656 \@gls@link[#1]{#2}{\@glo@text}%

```

Indicate that this entry has now been used

```

3657 \ifKV@gls@link@local
3658 \glslocalunset{#2}%
3659 \else
3660 \glsunset{#2}%
3661 \fi
3662 }%

```

```

3663 \gls@postlinkhook
3664 }

```

\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

```

3665 \newrobustcmd*{\Gls}{\@gls@hyp@opt\@Gls}

```

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

```

3666 \newcommand*{\@Gls}[2][{}]{%
3667 \new@ifnextchar[\@Gls@{#1}{#2}}{\@Gls@{#1}{#2}}}%
3668 }

```

\@Gls@ Read in the final optional argument:

```

3669 \def\@Gls@#1#2[#3]{%
3670 \glsdoifexists{#2}%
3671 {%
3672 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3673 \let\glsifplural\@secondoftwo
3674 \let\glschapscase\@secondofthree
3675 \let\glscustomtext\@empty
3676 \def\glsinsert{#3}%

```

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

```

3677 \def\@glo@text{\csname gls@\gls@type @entryfmt\endcsname}%

```

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

```
3678 \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3679 \ifKV@glslink@local
3680 \glslocalunset{#2}%
3681 \else
3682 \glsunset{#2}%
3683 \fi
3684 }%
```

```
3685 \glspostlinkhook
3686 }
```

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

`\GLS`

```
3687 \newrobustcmd*{\GLS}{\@gls@hyp@opt\@GLS}
```

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

```
3688 \newcommand*{\@GLS}[2][\@GLS@{#1}{#2}]{\@GLS@{#1}{#2}[\@GLS@{#1}{#2}[]]}%
3689 \new@ifnextchar[\@GLS@{#1}{#2}]{\@GLS@{#1}{#2}[\@GLS@{#1}{#2}[]]}%
3690 }
```

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

```
3691 \def\@GLS@#1#2[#3]{%
3692 \glsdoifexists{#2}%
3693 {%
3694 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
3695 \let\glsifplural\@secondoftwo
3696 \let\glscapscase\@thirdofthree
3697 \let\glscustomtext\@empty
3698 \def\glsinsert{#3}%
```

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

```
3699 \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%
```

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

```
3700 \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3701 \ifKV@glslink@local
3702 \glslocalunset{#2}%
3703 \else
3704 \glsunset{#2}%
3705 \fi
3706 }%
```

```

3707 \glspostlinkhook
3708 }

```

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

`\glspl`

```

3709 \newrobustcmd*{\glspl}{\@gls@hyp@opt\@glspl}

```

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

```

3710 \newcommand*{\@glspl}[2][{}]{%
3711   \new@ifnextchar[{\@glspl@{#1}{#2}}{\@glspl@{#1}{#2}[]}%
3712 }

```

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

```

3713 \def\@glspl@#1#2[#3]{%
3714   \glsdoifexists{#2}%
3715   {%
3716     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3717     \let\glsifplural\@firstoftwo
3718     \let\glsapscase\@firstofthree
3719     \let\glscustomtext\@empty
3720     \def\glsinsert{#3}%

```

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

```

3721   \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%

```

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

```

3722   \@gls@link[#1]{#2}{\@glo@text}%

```

Indicate that this entry has now been used

```

3723   \ifKV@glslink@local
3724     \glslocalunset{#2}%
3725   \else
3726     \glsunset{#2}%
3727   \fi
3728 }%

```

```

3729 \glspostlinkhook
3730 }

```

`\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`

```

3731 \newrobustcmd*{\Glspl}{\@gls@hyp@opt\@Glspl}

```

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

```
3732 \newcommand*{\@Glspl}[2] [] {%
3733   \new@ifnextchar[{\@Glspl@{#1}{#2}}{\@Glspl@{#1}{#2} []}%
3734 }
```

\@Glspl@ Read in the final optional argument:

```
3735 \def\@Glspl@#1#2[#3] {%
3736   \glsdoifexists{#2}%
3737   {%
3738     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
3739     \let\glsifplural\@firstoftwo
3740     \let\glsupcase\@secondofthree
3741     \let\glscustomtext\@empty
3742     \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 \glstyp.

```
3743   \def\@glo@text{\csname gls@\glstyp @entryfmt\endcsname}%
```

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

```
3744   \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3745   \ifKV@glslink@local
3746     \glslocalunset{#2}%
3747   \else
3748     \glsunset{#2}%
3749   \fi
3750 }%
```

```
3751 \glspostlinkhook
3752 }
```

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

\GLSp1

```
3753 \newrobustcmd*{\GLSp1}{\@gls@hyp@opt\@GLSp1}
```

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

```
3754 \newcommand*{\@GLSp1}[2] [] {%
3755   \new@ifnextchar[{\@GLSp1@{#1}{#2}}{\@GLSp1@{#1}{#2} []}%
3756 }
```

\@GLSp1 Read in the final optional argument:

```
3757 \def\@GLSp1@#1#2[#3] {%
3758   \glsdoifexists{#2}%
3759   {%
3760     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
```

```

3761 \let\glsifplural\@firstoftwo
3762 \let\glsifscapscase\@thirdofthree
3763 \let\glsifcustomtext\@empty
3764 \def\glsinsert{#3}%

```

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

```

3765 \def\@glo@text{\csname gls@\glstyp @entryfmt\endcsname}%

```

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

```

3766 \@gls@link[#1]{#2}{\@glo@text}%

```

Indicate that this entry has now been used

```

3767 \ifKV@glslink@local
3768 \glslocalunset{#2}%
3769 \else
3770 \glsunset{#2}%
3771 \fi
3772 }%

```

```

3773 \glspostlinkhook
3774 }

```

`\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:

```

3775 \newrobustcmd*{\glsdisp}{\@gls@hyp@opt\@glsdisp}

```

Defined the un-starred form.

`\@glsdisp`

```

3776 \newcommand*{\@glsdisp}[3][{}]{%
3777 \glsdoifexists{#2}{%

3778 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3779 \let\glsifplural\@secondoftwo
3780 \let\glsifscapscase\@firstofthree
3781 \def\glsifcustomtext{#3}%
3782 \def\glsinsert{}%

```

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

```

3783 \def\@glo@text{\csname gls@\glstyp @entryfmt\endcsname}%

```

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

```

3784 \@gls@link[#1]{#2}{\@glo@text}%

```

Indicate that this entry has now been used

```
3785 \ifKV@glslink@local
3786 \glsllocalunset{#2}%
3787 \else
3788 \glunset{#2}%
3789 \fi
3790 }%

3791 \glspostlinkhook
3792 }
```

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

```
3793 \newcommand*{\@gl@link@nocheckfirsthyper}{}
```

@gl@field@link

```
3794 \newcommand{\@gl@field@link}[3]{%
3795 \glstdoifexists{#2}%
3796 {%
3797 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
3798 \@gl@link[#1]{#2}{#3}%
3799 }%

3800 \glspostlinkhook
3801 }
```

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

\glstext

```
3802 \newrobustcmd*{\glstext}{\@gl@hyp@opt\@glstext}
```

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

```
3803 \newcommand*{\@glstext}[2][{}]{%
3804 \new@ifnextchar[{\@glstext@{#1}{#2}}{\@glstext@{#1}{#2}[]}}
```

Read in the final optional argument:

```
3805 \def\@glstext@#1#2[#3]{%
3806 \@gl@field@link{#1}{#2}{\glstrytext{#2}#3}%
3807 }
```

\GLStext behaves like \glstext except the text is converted to uppercase.

\GLStext

```
3808 \newrobustcmd*{\GLStext}{\@gl@hyp@opt\@GLStext}
```

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

```
3809 \newcommand*{\@GLStext}[2][{}]{%
3810 \new@ifnextchar[{\@GLStext@{#1}{#2}}{\@GLStext@{#1}{#2}[]}]}
```

Read in the final optional argument:

```
3811 \def\@GLStext@#1#2[#3]{%
3812   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glstentrytext{#2}#3}}%
3813 }
```

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

`\Glstext`

```
3814 \newrobustcmd*{\Glstext}{\@gls@hyp@opt\@GLstext}
```

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

```
3815 \newcommand*{\@GLstext}[2] [] {%
3816   \new@ifnextchar[{\@GLstext@{#1}{#2}}{\@GLstext@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3817 \def\@GLstext@#1#2[#3]{%
3818   \@gls@field@link{#1}{#2}{\Glstentrytext{#2}#3}%
3819 }
```

`\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`

```
3820 \newrobustcmd*{\glsfirst}{\@gls@hyp@opt\@glsfirst}
```

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

```
3821 \newcommand*{\@glsfirst}[2] [] {%
3822   \new@ifnextchar[{\@glsfirst@{#1}{#2}}{\@glsfirst@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3823 \def\@glsfirst@#1#2[#3]{%
3824   \@gls@field@link{#1}{#2}{\glstentryfirst{#2}#3}%
3825 }
```

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

`\Glsfirst`

```
3826 \newrobustcmd*{\Glsfirst}{\@gls@hyp@opt\@Glsfirst}
```

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

```
3827 \newcommand*{\@Glsfirst}[2] [] {%
3828   \new@ifnextchar[{\@Glsfirst@{#1}{#2}}{\@Glsfirst@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3829 \def\@Glsfirst@#1#2[#3]{%
3830   \@gls@field@link{#1}{#2}{\Glsentryfirst{#2}#3}%
3831 }
```

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

`\GLSfirst`

```
3832 \newrobustcmd*{\GLSfirst}{\@gls@hyp@opt\@GLSfirst}
```

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

```
3833 \newcommand*{\@GLSfirst}[2] [] {%  
3834   \new@ifnextchar[{\@GLSfirst@{#1}{#2}}{\@GLSfirst@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3835 \def\@GLSfirst@#1#2[#3] {%  
3836   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryfirst{#2}#3}}%  
3837 }
```

`\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`

```
3838 \newrobustcmd*{\glsplural}{\@gls@hyp@opt\@glsplural}
```

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

```
3839 \newcommand*{\@glsplural}[2] [] {%  
3840   \new@ifnextchar[{\@glsplural@{#1}{#2}}{\@glsplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3841 \def\@glsplural@#1#2[#3] {%  
3842   \@gls@field@link{#1}{#2}{\glsentryplural{#2}#3}}%  
3843 }
```

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

`\Glsplural`

```
3844 \newrobustcmd*{\Glsplural}{\@gls@hyp@opt\@Glsplural}
```

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

```
3845 \newcommand*{\@Glsplural}[2] [] {%  
3846   \new@ifnextchar[{\@Glsplural@{#1}{#2}}{\@Glsplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3847 \def\@Glsplural@#1#2[#3] {%  
3848   \@gls@field@link{#1}{#2}{\Glsentryplural{#2}#3}}%  
3849 }
```

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

`\GLSplural`

```
3850 \newrobustcmd*{\GLSplural}{\@gls@hyp@opt\@GLSplural}
```

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

```
3851 \newcommand*{\@GLSplural}[2] [] {%  
3852   \new@ifnextchar[{\@GLSplural@{#1}{#2}}{\@GLSplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3853 \def\@GLSplural@#1#2[#3] {%  
3854   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryplural{#2}#3}}%  
3855 }
```

`\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`

```
3856 \newrobustcmd*{\glsfirstplural}{\@gls@hyp@opt\@glsfirstplural}
```

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

```
3857 \newcommand*{\@glsfirstplural}[2] [] {%
```

```
3858   \new@ifnextchar[{\@glsfirstplural@{#1}{#2}}{\@glsfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3859 \def\@glsfirstplural@#1#2[#3] {%
```

```
3860   \@gls@field@link{#1}{#2}{\glsentryfirstplural{#2}#3}%
```

```
3861 }
```

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

`\Glsfirstplural`

```
3862 \newrobustcmd*{\Glsfirstplural}{\@gls@hyp@opt\@Glsfirstplural}
```

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

```
3863 \newcommand*{\@Glsfirstplural}[2] [] {%
```

```
3864   \new@ifnextchar[{\@Glsfirstplural@{#1}{#2}}{\@Glsfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3865 \def\@Glsfirstplural@#1#2[#3] {%
```

```
3866   \@gls@field@link{#1}{#2}{\Glsentryfirstplural{#2}#3}%
```

```
3867 }
```

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

`\GLSfirstplural`

```
3868 \newrobustcmd*{\GLSfirstplural}{\@gls@hyp@opt\@GLSfirstplural}
```

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

```
3869 \newcommand*{\@GLSfirstplural}[2] [] {%
```

```
3870   \new@ifnextchar[{\@GLSfirstplural@{#1}{#2}}{\@GLSfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3871 \def\@GLSfirstplural@#1#2[#3] {%
```

```
3872   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryfirstplural{#2}#3}}%
```

```
3873 }
```

`\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`

```
3874 \newrobustcmd*{\glsname}{\@gls@hyp@opt\@glsname}
```

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

```
3875 \newcommand*{\@glsname}[2] [] {%
```

```
3876   \new@ifnextchar[{\@glsname@{#1}{#2}}{\@glsname@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3877 \def\@glsname@#1#2[#3]{%
3878   \@gls@field@link{#1}{#2}{\glsentryname{#2}#3}%
3879 }
```

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

\Glsname

```
3880 \newrobustcmd*{\Glsname}{\@gls@hyp@opt\@Glsname}
```

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

```
3881 \newcommand*{\@Glsname}[2][{}]{%
3882   \new@ifnextchar[{\@Glsname@{#1}{#2}}{\@Glsname@{#1}{#2}[]}}
```

Read in the final optional argument:

```
3883 \def\@Glsname@#1#2[#3]{%
3884   \@gls@field@link{#1}{#2}{\Glsentryname{#2}#3}%
3885 }
```

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

\GLSname

```
3886 \newrobustcmd*{\GLSname}{\@gls@hyp@opt\@GLSname}
```

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

```
3887 \newcommand*{\@GLSname}[2][{}]{%
3888   \new@ifnextchar[{\@GLSname@{#1}{#2}}{\@GLSname@{#1}{#2}[]}]}
```

Read in the final optional argument:

```
3889 \def\@GLSname@#1#2[#3]{%
3890   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryname{#2}#3}}%
3891 }
```

\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

```
3892 \newrobustcmd*{\glsdesc}{\@gls@hyp@opt\@glsdesc}
```

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

```
3893 \newcommand*{\@glsdesc}[2][{}]{%
3894   \new@ifnextchar[{\@glsdesc@{#1}{#2}}{\@glsdesc@{#1}{#2}[]}]}
```

Read in the final optional argument:

```
3895 \def\@glsdesc@#1#2[#3]{%
3896   \@gls@field@link{#1}{#2}{\glsentrydesc{#2}#3}%
3897 }
```

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

\Glsdesc

```
3898 \newrobustcmd*{\Glsdesc}{\@gls@hyp@opt\@Glsdesc}
```

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

```
3899 \newcommand*{\@GLSdesc}[2] [] {%  
3900   \new@ifnextchar [{\@GLSdesc@{#1}{#2}}{\@GLSdesc@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3901 \def\@GLSdesc@#1#2[#3] {%  
3902   \@gls@field@link{#1}{#2}{\@Glsentrydesc{#2}#3}%  
3903 }
```

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

\GLSdesc

```
3904 \newrobustcmd*{\GLSdesc}{\@gls@hyp@opt\@GLSdesc}
```

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

```
3905 \newcommand*{\@GLSdesc}[2] [] {%  
3906   \new@ifnextchar [{\@GLSdesc@{#1}{#2}}{\@GLSdesc@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3907 \def\@GLSdesc@#1#2[#3] {%  
3908   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\@Glsentrydesc{#2}#3}}%  
3909 }
```

\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

```
3910 \newrobustcmd*{\glsdescplural}{\@gls@hyp@opt\@glsdescplural}
```

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

```
3911 \newcommand*{\@glsdescplural}[2] [] {%  
3912   \new@ifnextchar [{\@glsdescplural@{#1}{#2}}{\@glsdescplural@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3913 \def\@glsdescplural@#1#2[#3] {%  
3914   \@gls@field@link{#1}{#2}{\@Glsentrydescplural{#2}#3}%  
3915 }
```

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

\Glsdescplural

```
3916 \newrobustcmd*{\Glsdescplural}{\@gls@hyp@opt\@Glsdescplural}
```

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

```
3917 \newcommand*{\@Glsdescplural}[2] [] {%  
3918   \new@ifnextchar [{\@Glsdescplural@{#1}{#2}}{\@Glsdescplural@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3919 \def\@Glsdescplural@#1#2[#3] {%  
3920   \@gls@field@link{#1}{#2}{\@Glsentrydescplural{#2}#3}%  
3921 }
```

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

`\GLSdescplural`

```
3922 \newrobustcmd*{\GLSdescplural}{\@gls@hyp@opt\@GLSdescplural}
```

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

```
3923 \newcommand*{\@GLSdescplural}[2] [] {%
```

```
3924   \new@ifnextchar[{\@GLSdescplural@{#1}{#2}}{\@GLSdescplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3925 \def\@GLSdescplural@#1#2[#3] {%
```

```
3926   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrydescplural{#2}#3}}%
```

```
3927 }
```

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

`\glsymbol`

```
3928 \newrobustcmd*{\glsymbol}{\@gls@hyp@opt\@glsymbol}
```

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

```
3929 \newcommand*{\@glsymbol}[2] [] {%
```

```
3930   \new@ifnextchar[{\@glsymbol@{#1}{#2}}{\@glsymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3931 \def\@glsymbol@#1#2[#3] {%
```

```
3932   \@gls@field@link{#1}{#2}{\glsentrysymbol{#2}#3}}%
```

```
3933 }
```

`\Glsymbol` behaves like `\glsymbol` except that the first letter is converted to uppercase.

`\Glsymbol`

```
3934 \newrobustcmd*{\Glsymbol}{\@gls@hyp@opt\@Glsymbol}
```

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

```
3935 \newcommand*{\@Glsymbol}[2] [] {%
```

```
3936   \new@ifnextchar[{\@Glsymbol@{#1}{#2}}{\@Glsymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3937 \def\@Glsymbol@#1#2[#3] {%
```

```
3938   \@gls@field@link{#1}{#2}{\Glsentrysymbol{#2}#3}}%
```

```
3939 }
```

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

`\GLSsymbol`

```
3940 \newrobustcmd*{\GLSsymbol}{\@gls@hyp@opt\@GLSsymbol}
```

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

```
3941 \newcommand*{\@GLSsymbol}[2] [] {%
```

```
3942   \new@ifnextchar[{\@GLSsymbol@{#1}{#2}}{\@GLSsymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3943 \def\@GLSsymbol@#1#2[#3]{%
3944   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrysymbol{#2}#3}}%
3945 }
```

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

`glsymbolplural`

```
3946 \newrobustcmd*{\glsymbolplural}{\@gls@hyp@opt\@glsymbolplural}
```

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

```
3947 \newcommand*{\@glsymbolplural}[2][ ]{%
3948   \new@ifnextchar[{\@glsymbolplural@{#1}{#2}}{\@glsymbolplural@{#1}{#2}[ ]}}
```

Read in the final optional argument:

```
3949 \def\@glsymbolplural@#1#2[#3]{%
3950   \@gls@field@link{#1}{#2}{\glsentrysymbolplural{#2}#3}%
3951 }
```

`\Glsymbolplural` behaves like `\glsymbolplural` except that the first letter is converted to uppercase.

`Glsymbolplural`

```
3952 \newrobustcmd*{\Glsymbolplural}{\@gls@hyp@opt\@Glsymbolplural}
```

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

```
3953 \newcommand*{\@Glsymbolplural}[2][ ]{%
3954   \new@ifnextchar[{\@Glsymbolplural@{#1}{#2}}{\@Glsymbolplural@{#1}{#2}[ ]}}
```

Read in the final optional argument:

```
3955 \def\@Glsymbolplural@#1#2[#3]{%
3956   \@gls@field@link{#1}{#2}{\Glsentrysymbolplural{#2}#3}%
3957 }
```

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

`GLSsymbolplural`

```
3958 \newrobustcmd*{\GLSsymbolplural}{\@gls@hyp@opt\@GLSsymbolplural}
```

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

```
3959 \newcommand*{\@GLSsymbolplural}[2][ ]{%
3960   \new@ifnextchar[{\@GLSsymbolplural@{#1}{#2}}{\@GLSsymbolplural@{#1}{#2}[ ]}}
```

Read in the final optional argument:

```
3961 \def\@GLSsymbolplural@#1#2[#3]{%
3962   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrysymbolplural{#2}#3}}%
3963 }
```

`\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

```
3964 \newrobustcmd*{\glsuseri}{\@gls@hyp@opt\@glsuseri}
```

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

```
3965 \newcommand*{\@glsuseri}[2] [] {%
```

```
3966   \new@ifnextchar[{\@glsuseri@{#1}{#2}}{\@glsuseri@{#1}{#2} []}}
```

Read in the final optional argument:

```
3967 \def\@glsuseri@#1#2[#3]{%
```

```
3968   \@gls@field@link{#1}{#2}{\glsentryuseri{#2}#3}%
```

```
3969 }
```

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

\Glsuseri

```
3970 \newrobustcmd*{\Glsuseri}{\@gls@hyp@opt\@Glsuseri}
```

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

```
3971 \newcommand*{\@Glsuseri}[2] [] {%
```

```
3972   \new@ifnextchar[{\@Glsuseri@{#1}{#2}}{\@Glsuseri@{#1}{#2} []}}
```

Read in the final optional argument:

```
3973 \def\@Glsuseri@#1#2[#3]{%
```

```
3974   \@gls@field@link{#1}{#2}{\Glsentryuseri{#2}#3}%
```

```
3975 }
```

\GLSuseri behaves like \glsuseri except that the link text is converted to uppercase.

\GLSuseri

```
3976 \newrobustcmd*{\GLSuseri}{\@gls@hyp@opt\@GLSuseri}
```

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

```
3977 \newcommand*{\@GLSuseri}[2] [] {%
```

```
3978   \new@ifnextchar[{\@GLSuseri@{#1}{#2}}{\@GLSuseri@{#1}{#2} []}}
```

Read in the final optional argument:

```
3979 \def\@GLSuseri@#1#2[#3]{%
```

```
3980   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseri{#2}#3}}%
```

```
3981 }
```

\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

```
3982 \newrobustcmd*{\glsuserii}{\@gls@hyp@opt\@glsuserii}
```

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

```
3983 \newcommand*{\@glsuserii}[2] [] {%
```

```
3984   \new@ifnextchar[{\@glsuserii@{#1}{#2}}{\@glsuserii@{#1}{#2} []}}
```

Read in the final optional argument:

```
3985 \def\@glsuserii@#1#2[#3]{%
```

```
3986   \@gls@field@link{#1}{#2}{\glsentryuserii{#2}#3}%
```

```
3987 }
```

\Glsuserii behaves like \glsuserii except that the first letter is converted to uppercase.

\Glsuserii

```
3988 \newrobustcmd*{\Glsuserii}{\@gls@hyp@opt\@Glsuserii}
```

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

```
3989 \newcommand*{\@Glsuserii}[2] [] {%
```

```
3990   \new@ifnextchar[{\@Glsuserii@{#1}{#2}}{\@Glsuserii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3991 \def\@Glsuserii@#1#2[#3] {%
```

```
3992   \@gls@field@link{#1}{#2}{\Glsentryuserii{#2}#3}%
```

```
3993 }
```

\GLSuserii behaves like \glsuserii except that the link text is converted to uppercase.

\GLSuserii

```
3994 \newrobustcmd*{\GLSuserii}{\@gls@hyp@opt\@GLSuserii}
```

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

```
3995 \newcommand*{\@GLSuserii}[2] [] {%
```

```
3996   \new@ifnextchar[{\@GLSuserii@{#1}{#2}}{\@GLSuserii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3997 \def\@GLSuserii@#1#2[#3] {%
```

```
3998   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentryuserii{#2}#3}}%
```

```
3999 }
```

\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

```
4000 \newrobustcmd*{\glsuseriii}{\@gls@hyp@opt\@glsuseriii}
```

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

```
4001 \newcommand*{\@glsuseriii}[2] [] {%
```

```
4002   \new@ifnextchar[{\@glsuseriii@{#1}{#2}}{\@glsuseriii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4003 \def\@glsuseriii@#1#2[#3] {%
```

```
4004   \@gls@field@link{#1}{#2}{\glsentryuseriii{#2}#3}%
```

```
4005 }
```

\Glsuseriii behaves like \glsuseriii except that the first letter is converted to uppercase.

\Glsuseriii

```
4006 \newrobustcmd*{\Glsuseriii}{\@gls@hyp@opt\@Glsuseriii}
```

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

```
4007 \newcommand*{\@Glsuseriii}[2] [] {%
```

```
4008   \new@ifnextchar[{\@Glsuseriii@{#1}{#2}}{\@Glsuseriii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4009 \def\@Glsuseriii@#1#2[#3]{%
4010   \@gls@field@link{#1}{#2}{\Glsentryuseriii{#2}#3}%
4011 }
```

\Glsuseriii behaves like \glsuseriii except that the link text is converted to uppercase.

\Glsuseriii

```
4012 \newrobustcmd*{\Glsuseriii}{\@gls@hyp@opt\@Glsuseriii}
```

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

```
4013 \newcommand*{\@Glsuseriii}[2][\@Glsuseriii@#1]{%
4014   \new@ifnextchar[\@Glsuseriii@{#1}{#2}]{\@Glsuseriii@{#1}{#2}[]}}
```

Read in the final optional argument:

```
4015 \def\@Glsuseriii@#1#2[#3]{%
4016   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentryuseriii{#2}#3}}%
4017 }
```

\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

```
4018 \newrobustcmd*{\glsuseriv}{\@gls@hyp@opt\@glsuseriv}
```

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

```
4019 \newcommand*{\@glsuseriv}[2][\@glsuseriv@#1]{%
4020   \new@ifnextchar[\@glsuseriv@{#1}{#2}]{\@glsuseriv@{#1}{#2}[]}}
```

Read in the final optional argument:

```
4021 \def\@glsuseriv@#1#2[#3]{%
4022   \@gls@field@link{#1}{#2}{\Glsentryuseriv{#2}#3}%
4023 }
```

\Glsuseriv behaves like \glsuseriv except that the first letter is converted to uppercase.

\Glsuseriv

```
4024 \newrobustcmd*{\Glsuseriv}{\@gls@hyp@opt\@Glsuseriv}
```

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

```
4025 \newcommand*{\@Glsuseriv}[2][\@Glsuseriv@#1]{%
4026   \new@ifnextchar[\@Glsuseriv@{#1}{#2}]{\@Glsuseriv@{#1}{#2}[]}}
```

Read in the final optional argument:

```
4027 \def\@Glsuseriv@#1#2[#3]{%
4028   \@gls@field@link{#1}{#2}{\Glsentryuseriv{#2}#3}%
4029 }
```

\GLSuseriv behaves like \glsuseriv except that the link text is converted to uppercase.

\GLSuseriv

```
4030 \newrobustcmd*{\GLSuseriv}{\@gls@hyp@opt\@GLSuseriv}
```


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

```
4031 \newcommand*{\@GLSuseriv}[2] [] {%  
4032   \new@ifnextchar [{\@GLSuseriv@{#1}{#2}}]{\@GLSuseriv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4033 \def\@GLSuseriv@#1#2[#3] {%  
4034   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseriv{#2}#3}}%  
4035 }
```

\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

```
4036 \newrobustcmd*{\glsuserv}{\@gls@hyp@opt\@glsuserv}
```

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

```
4037 \newcommand*{\@glsuserv}[2] [] {%  
4038   \new@ifnextchar [{\@glsuserv@{#1}{#2}}]{\@glsuserv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4039 \def\@glsuserv@#1#2[#3] {%  
4040   \@gls@field@link{#1}{#2}{\glsentryuserv{#2}#3}}%  
4041 }
```

\Glsuserv behaves like \glsuserv except that the first letter is converted to uppercase.

\Glsuserv

```
4042 \newrobustcmd*{\Glsuserv}{\@gls@hyp@opt\@Glsuserv}
```

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

```
4043 \newcommand*{\@Glsuserv}[2] [] {%  
4044   \new@ifnextchar [{\@Glsuserv@{#1}{#2}}]{\@Glsuserv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4045 \def\@Glsuserv@#1#2[#3] {%  
4046   \@gls@field@link{#1}{#2}{\Glsentryuserv{#2}#3}}%  
4047 }
```

\GLSuserv behaves like \glsuserv except that the link text is converted to uppercase.

\GLSuserv

```
4048 \newrobustcmd*{\GLSuserv}{\@gls@hyp@opt\@GLSuserv}
```

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

```
4049 \newcommand*{\@GLSuserv}[2] [] {%  
4050   \new@ifnextchar [{\@GLSuserv@{#1}{#2}}]{\@GLSuserv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4051 \def\@GLSuserv@#1#2[#3] {%  
4052   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuserv{#2}#3}}%  
4053 }
```

\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

```
4054 \newrobustcmd*{\glsuservi}{\@gls@hyp@opt\@glsuservi}
```

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

```
4055 \newcommand*{\@glsuservi}[2][\%
```

```
4056 \new@ifnextchar[{\@glsuservi@{#1}{#2}}{\@glsuservi@{#1}{#2}[]}]
```

Read in the final optional argument:

```
4057 \def\@glsuservi@#1#2[#3]{\%
```

```
4058 \@gls@field@link{#1}{#2}{\glsentryuservi{#2}#3}%
```

```
4059 }
```

\Glsuservi behaves like \glsuservi except that the first letter is converted to uppercase.

\Glsuservi

```
4060 \newrobustcmd*{\Glsuservi}{\@gls@hyp@opt\@Glsuservi}
```

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

```
4061 \newcommand*{\@Glsuservi}[2][\%
```

```
4062 \new@ifnextchar[{\@Glsuservi@{#1}{#2}}{\@Glsuservi@{#1}{#2}[]}]
```

Read in the final optional argument:

```
4063 \def\@Glsuservi@#1#2[#3]{\%
```

```
4064 \@gls@field@link{#1}{#2}{\Glsentryuservi{#2}#3}%
```

```
4065 }
```

\GLSuservi behaves like \glsuservi except that the link text is converted to uppercase.

\GLSuservi

```
4066 \newrobustcmd*{\GLSuservi}{\@gls@hyp@opt\@GLSuservi}
```

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

```
4067 \newcommand*{\@GLSuservi}[2][\%
```

```
4068 \new@ifnextchar[{\@GLSuservi@{#1}{#2}}{\@GLSuservi@{#1}{#2}[]}]
```

Read in the final optional argument:

```
4069 \def\@GLSuservi@#1#2[#3]{\%
```

```
4070 \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuservi{#2}#3}}%
```

```
4071 }
```

Now deal with acronym related keys. First the short form:

\acrshort

```
4072 \newrobustcmd*{\acrshort}{\@gls@hyp@opt\@ns@acrshort}
```

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

```
4073 \newcommand*{\@ns@acrshort}[2][\%
```

```
4074 \new@ifnextchar[{\@acrshort{#1}{#2}}{\@acrshort{#1}{#2}[]}]
```

```
4075 }
```

Read in the final optional argument:

```
4076 \def\@acrshort#1#2[#3]{\%
```

```
4077 \glsdoifexists{#2}%
```

```
4078 {\%
```

```

4079 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper

4080 \let\gl@sifplural\@secondoftwo
4081 \let\gl@scapscase\@firstofthree
4082 \let\gl@insert\@empty
4083 \def\gl@customtext{%
4084     \acronymfont{\gl@sentryshort{#2}}#3%
4085 }%

    Call \@gl@link Note that \@gl@link sets \glstype.
4086 \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4087 }%

4088 \glspostlinkhook
4089 }

```

\Acrshort

```

4090 \newrobustcmd*{\Acrshort}{\@gl@hyp@opt\@ns@Acrshort}

    Define the un-starred form. Need to determine if there is a final optional argument
4091 \newcommand*{\ns@Acrshort}[2][{}]{%
4092     \new@ifnextchar[{\@Acrshort{#1}{#2}}{\@Acrshort{#1}{#2}[]}%
4093 }

    Read in the final optional argument:
4094 \def\@Acrshort#1#2[#3]{%
4095     \gl@sdoifexists{#2}%
4096     {%
4097         \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper

4098         \def\gl@label{#2}%
4099         \let\gl@sifplural\@secondoftwo
4100         \let\gl@scapscase\@secondofthree
4101         \let\gl@insert\@empty
4102         \def\gl@customtext{%
4103             \acronymfont{\gl@sentryshort{#2}}#3%
4104         }%

        Call \@gl@link Note that \@gl@link sets \glstype.
4105         \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4106         }%

4107     \glspostlinkhook
4108 }

```

\ACRshort

```

4109 \newrobustcmd*{\ACRshort}{\@gl@hyp@opt\@ns@ACRshort}

```

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

```
4110 \newcommand*{\ns@ACRshort}[2] [] {%
4111   \new@ifnextchar[{\@ACRshort{#1}{#2}}{\@ACRshort{#1}{#2} []}]%
4112 }
```

Read in the final optional argument:

```
4113 \def\@ACRshort#1#2[#3] {%
4114   \glsdoifexists{#2}%
4115   {%
4116     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4117     \def\glslabel{#2}%
4118     \let\glsifplural\@secondoftwo
4119     \let\glscapscase\@thirdofthree
4120     \let\glsinsert\@empty
4121     \def\glscustomtext{%
4122       \mfirstucMakeUppercase{\acronymfont{\glsentryshort{#2}}#3}%
4123     }%
```

Call \@gl@link Note that \@gl@link sets \gls type.

```
4124   \@gl@link[#1]{#2}{\csname gls@\gls type @entryfmt\endcsname}%
4125   }%
4126   \glspostlinkhook
4127 }
```

Short plural:

\acrshortpl

```
4128 \newrobustcmd*{\acrshortpl}{\@gl@hyp@opt\@ns@acrshortpl}
```

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

```
4129 \newcommand*{\ns@acrshortpl}[2] [] {%
4130   \new@ifnextchar[{\@acrshortpl{#1}{#2}}{\@acrshortpl{#1}{#2} []}]%
4131 }
```

Read in the final optional argument:

```
4132 \def\@acrshortpl#1#2[#3] {%
4133   \glsdoifexists{#2}%
4134   {%
4135     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4136     \def\glslabel{#2}%
4137     \let\glsifplural\@firstoftwo
4138     \let\glscapscase\@firstofthree
4139     \let\glsinsert\@empty
4140     \def\glscustomtext{%
4141       \acronymfont{\glsentryshortpl{#2}}#3%
4142     }%
```

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

```
4143   \@gls@link[#1]{#2}{\csname gls@\glsstyle @entryfmt\endcsname}%  
4144   }%  
  
4145   \glspostlinkhook  
4146 }
```

\Acrshortpl

```
4147 \newrobustcmd*{\Acrshortpl}{\@gls@hyp@opt\ns@Acrshortpl}
```

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

```
4148 \newcommand*{\ns@Acrshortpl}[2] [] {%  
4149   \new@ifnextchar[{\@Acrshortpl{#1}{#2}}{\@Acrshortpl{#1}{#2} []}%  
4150 }
```

Read in the final optional argument:

```
4151 \def\@Acrshortpl#1#2[#3] {%  
4152   \glsdoifexists{#2}%  
4153   {%  
  
4154     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper  
  
4155     \def\glslabel{#2}%  
4156     \let\glsifplural\@firstoftwo  
4157     \let\gls caps case\@secondofthree  
4158     \let\glsinsert\@empty  
4159     \def\gls custom text{%  
4160       \acronymfont{\Glsentryshortpl{#2}}#3%  
4161     }%
```

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

```
4162   \@gls@link[#1]{#2}{\csname gls@\glsstyle @entryfmt\endcsname}%  
4163   }%  
  
4164   \glspostlinkhook  
4165 }
```

\ACRshortpl

```
4166 \newrobustcmd*{\ACRshortpl}{\@gls@hyp@opt\ns@ACRshortpl}
```

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

```
4167 \newcommand*{\ns@ACRshortpl}[2] [] {%  
4168   \new@ifnextchar[{\@ACRshortpl{#1}{#2}}{\@ACRshortpl{#1}{#2} []}%  
4169 }
```

Read in the final optional argument:

```
4170 \def\@ACRshortpl#1#2[#3] {%  
4171   \glsdoifexists{#2}%  
4172   {%  
  
4173     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
```

```

4174 \def\glslabel{#2}%
4175 \let\glsifplural\@firstoftwo
4176 \let\glsifscapscase\@thirdofthree
4177 \let\glsinsert\@empty
4178 \def\glscustomtext{%
4179     \mfirstucMakeUppercase{\acronymfont{\glsentryshortpl{#2}}#3}%
4180 }%

```

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

```

4181 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4182 }%

4183 \glspostlinkhook
4184 }

```

\acrlong

```

4185 \newrobustcmd*{\acrlong}{\@gls@hyp@opt\ns@acrlong}

```

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

```

4186 \newcommand*{\ns@acrlong}[2] [] {%
4187     \new@ifnextchar[{\@acrlong{#1}{#2}}{\@acrlong{#1}{#2} []}%
4188 }

```

Read in the final optional argument:

```

4189 \def\@acrlong#1#2[#3] {%
4190     \glsdoifexists{#2}%
4191     {%
4192         \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

4193         \def\glslabel{#2}%
4194         \let\glsifplural\@secondoftwo
4195         \let\glsifscapscase\@firstofthree
4196         \let\glsinsert\@empty

```

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

```

4197     \def\glscustomtext{%
4198         \glsentrylong{#2}#3%
4199     }%

```

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

```

4200     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4201     }%

4202     \glspostlinkhook
4203 }

```

\Acrlong

```

4204 \newrobustcmd*{\Acrlong}{\@gls@hyp@opt\ns@Acrlong}

```

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

```
4205 \newcommand*{\ns@Acrlong}[2][\]{%
4206   \new@ifnextchar[{\@Acrlong{#1}{#2}}{\@Acrlong{#1}{#2}[]}%
4207 }
```

Read in the final optional argument:

```
4208 \def\@Acrlong#1#2[#3]{%
4209   \glsdoifexists{#2}%
4210   {%
4211     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4212     \def\glslabel{#2}%
4213     \let\glsifplural\@secondoftwo
4214     \let\glscapscase\@secondofthree
4215     \let\glsinsert\@empty
```

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

```
4216   \def\glscustomtext{%
4217     \Glsentrylong{#2}#3%
4218   }%
```

Call \@gl@link. Note that \@gl@link sets \glstype.

```
4219   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4220   }%
4221   \glspostlinkhook
4222 }
```

\ACRlong

```
4223 \newrobustcmd*{\ACRlong}{\@gl@hyp@opt\ns@ACRlong}
```

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

```
4224 \newcommand*{\ns@ACRlong}[2][\]{%
4225   \new@ifnextchar[{\@ACRlong{#1}{#2}}{\@ACRlong{#1}{#2}[]}%
4226 }
```

Read in the final optional argument:

```
4227 \def\@ACRlong#1#2[#3]{%
4228   \glsdoifexists{#2}%
4229   {%
4230     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4231     \def\glslabel{#2}%
4232     \let\glsifplural\@secondoftwo
4233     \let\glscapscase\@thirdofthree
4234     \let\glsinsert\@empty
```

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

```
4235 \def\glscustomtext{%
4236 \mfirstucMakeUppercase{\glsentrylong{#2}#3}%
4237 }%
```

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

```
4238 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4239 }%

4240 \glspostlinkhook
4241 }
```

Short plural:

\acrlongpl

```
4242 \newrobustcmd*{\acrlongpl}{\@gls@hyp@opt\ns@acrlongpl}
```

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

```
4243 \newcommand*{\ns@acrlongpl}[2][{}]{%
4244 \new@ifnextchar[{\@acrlongpl{#1}{#2}}{\@acrlongpl{#1}{#2}[]}%
4245 }
```

Read in the final optional argument:

```
4246 \def\@acrlongpl#1#2[#3]{%
4247 \glsdoifexists{#2}%
4248 {%
4249 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

4250 \def\glslabel{#2}%
4251 \let\glsifplural\@firstoftwo
4252 \let\glscapscase\@firstofthree
4253 \let\glsinsert\@empty
```

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

```
4254 \def\glscustomtext{%
4255 \glentrylongpl{#2}#3%
4256 }%
```

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

```
4257 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4258 }%

4259 \glspostlinkhook
4260 }
```

\Acrlongpl

```
4261 \newrobustcmd*{\Acrlongpl}{\@gls@hyp@opt\ns@Acrlongpl}
```


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

```
4262 \newcommand*{\ns@Acrlongpl}[2][\%
4263   \new@ifnextchar[\@Acrlongpl{#1}{#2}]{\@Acrlongpl{#1}{#2}[]}%
4264 }
```

Read in the final optional argument:

```
4265 \def\@Acrlongpl#1#2[#3]{%
4266   \glsdoifexists{#2}%
4267   {%
4268     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4269     \def\glslabel{#2}%
4270     \let\glsifplural\@firstoftwo
4271     \let\glscapscase\@secondofthree
4272     \let\glsinsert\@empty
```

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

```
4273   \def\glscustomtext{%
4274     \Glsentrylongpl{#2}#3%
4275   }%
```

Call \@gl@link. Note that \@gl@link sets \glstype.

```
4276   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4277   }%
4278   \glspostlinkhook
4279 }
```

\ACRlongpl

```
4280 \newrobustcmd*{\ACRlongpl}{\@gl@hyp@opt\ns@ACRlongpl}
```

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

```
4281 \newcommand*{\ns@ACRlongpl}[2][\%
4282   \new@ifnextchar[\@ACRlongpl{#1}{#2}]{\@ACRlongpl{#1}{#2}[]}%
4283 }
```

Read in the final optional argument:

```
4284 \def\@ACRlongpl#1#2[#3]{%
4285   \glsdoifexists{#2}%
4286   {%
4287     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4288     \def\glslabel{#2}%
4289     \let\glsifplural\@firstoftwo
4290     \let\glscapscase\@thirdofthree
4291     \let\glsinsert\@empty
```

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

```

4292 \def\glscustomtext{%
4293   \mfirstucMakeUppercase{\glsenrylongpl{#2}#3}%
4294 }%

Call \@gls@link. Note that \@gls@link sets \glstype.
4295 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4296 }%

4297 \glspostlinkhook
4298 }

```

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>}
```

```

4299 \newcommand*{\@gls@entry@field}[2]{%
4300   \csname glo@\glsdetoklabel{#1}@#2\endcsname
4301 }

```

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

```

4302 \newcommand*{\glsletentryfield}[3]{%
4303   \letcs{#1}{glo@\glsdetoklabel{#2}@#3}%
4304 }

```

`\Gls@entry@field` Generic first letter uppercase version.

```
\@Gls@entry@field{<label>}{<field>}
```

```

4305 \newcommand*{\@Gls@entry@field}[2]{%
4306   \glsdoifexistsordo{#1}%
4307   {%
4308     \letcs{\@glo@text}{glo@\glsdetoklabel{#1}@#2}%
4309     \ifdef\@glo@text
4310     {%
4311       \xmakefirstuc{\@glo@text}%
4312     }%
4313     {%
4314       ??\PackageError{glossaries}{The field ‘#2’ doesn’t exist for glossary
4315       entry ‘\glsdetoklabel{#1}’}{Check you have correctly spelt the entry

```

```

4316      label and the field name}%
4317    }%
4318  }%
4319  {%
4320    ??%
4321  }%
4322 }

```

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`

```

4323 \newcommand*{\glsentryname}[1]{\@gls@entry@field{#1}{name}}

```

`\Glsentryname`

```

4324 \newrobustcmd*{\Glsentryname}[1]{%
4325   \@Gls@entryname{#1}%
4326 }

```

`\@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:

```

4327 \newcommand*{\@Gls@entryname}[1]{%
4328   \@Gls@entry@field{#1}{name}%
4329 }

```

`\ls@acrentryname` Now the behaviour when `\setacronymstyle` is used:

```

4330 \newcommand*{\@Gls@acrentryname}[1]{%
4331   \ifglshaslong{#1}%
4332   {%
4333     \letcs\@glo@text{glo\@glsdetoklabel{#1}@name}%
4334     \expandafter\@gls@getbody\@glo@text{}\@nil
4335     \expandafter\ifx\@gls@body\glsentrylong\relax
4336     \expandafter\Glsentrylong\@gls@rest
4337   \else
4338     \expandafter\ifx\@gls@body\glsentryshort\relax
4339     \expandafter\Glsentryshort\@gls@rest
4340   \else
4341     \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.

```

4342     {%
4343       \let\glsentryshort\Glsentryshort
4344       \@glo@text
4345     }%
4346   \else

```

```

4347         \xmakefirstuc{\@glo@text}%
4348     \fi
4349 \fi
4350 \fi
4351 }%
4352 {%

```

Not an acronym

```

4353     \@Gls@entry@field{#1}{name}%
4354 }%
4355 }

```

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`

```

4356 \newcommand*{\glsentrydesc}[1]{\@gls@entry@field{#1}{desc}}

```

`\Glsentrydesc`

```

4357 \newrobustcmd*{\Glsentrydesc}[1]{%
4358     \@Gls@entry@field{#1}{desc}%
4359 }

```

Plural form:

`entrydescplural`

```

4360 \newcommand*{\glsentrydescplural}[1]{%
4361     \@gls@entry@field{#1}{descplural}%
4362 }

```

`entrydescplural`

```

4363 \newrobustcmd*{\Glsentrydescplural}[1]{%
4364     \@Gls@entry@field{#1}{descplural}%
4365 }

```

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`

```

4366 \newcommand*{\glsentrytext}[1]{\@gls@entry@field{#1}{text}}

```

`\Glsentrytext`

```

4367 \newrobustcmd*{\Glsentrytext}[1]{%
4368     \@Gls@entry@field{#1}{text}%
4369 }

```

Get the plural form:

\glentryplural

```
4370 \newcommand*{\glentryplural}[1]{%
4371   \@gls@entry@field{#1}{plural}%
4372 }
```

\Glsentryplural

```
4373 \newrobustcmd*{\Glsentryplural}[1]{%
4374   \@Gls@entry@field{#1}{plural}%
4375 }
```

Get the symbol associated with this entry. The argument is the label associated with the entry.

\glentrysymbol

```
4376 \newcommand*{\glentrysymbol}[1]{%
4377   \@gls@entry@field{#1}{symbol}%
4378 }
```

\Glsentrysymbol

```
4379 \newrobustcmd*{\Glsentrysymbol}[1]{%
4380   \@Gls@entry@field{#1}{symbol}%
4381 }
```

Plural form:

trysymbolplural

```
4382 \newcommand*{\glentrysymbolplural}[1]{%
4383   \@gls@entry@field{#1}{symbolplural}%
4384 }
```

trysymbolplural

```
4385 \newrobustcmd*{\Glsentrysymbolplural}[1]{%
4386   \@Gls@entry@field{#1}{symbolplural}%
4387 }
```

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).

\glentryfirst

```
4388 \newcommand*{\glentryfirst}[1]{%
4389   \@gls@entry@field{#1}{first}%
4390 }
```

\Glsentryfirst

```
4391 \newrobustcmd*{\Glsentryfirst}[1]{%
4392   \@Gls@entry@field{#1}{first}%
4393 }
```

Get the plural form (as specified by the firstplural key when the entry was defined).

entryfirstplural

```
4394 \newcommand*{\glsentryfirstplural}[1]{%
4395   \@gls@entry@field{#1}{firstpl}%
4396 }
```

entryfirstplural

```
4397 \newrobustcmd*{\Glsentryfirstplural}[1]{%
4398   \@Gls@entry@field{#1}{firstpl}%
4399 }
```

entrytitlecase

```
4400 \newrobustcmd*{\@glsentrytitlecase}[2]{%
4401   \glsfieldfetch{#1}{#2}{\@gls@value}%
4402   \xcapitalisewords{\@gls@value}%
4403 }
4404 \ifdef\texorpdfstring
4405 {
4406   \newcommand*{\glsentrytitlecase}[2]{%
4407     \texorpdfstring
4408       {\@glsentrytitlecase{#1}{#2}}%
4409       {\@gls@entry@field{#1}{#2}}%
4410   }
4411 }
4412 {
4413   \newcommand*{\glsentrytitlecase}[2]{\@glsentrytitlecase{#1}{#2}}
4414 }
```

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

\glsentrytype

```
4415 \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

```
4416 \newcommand*{\glsentrysort}[1]{%
4417   \@gls@entry@field{#1}{sort}%
4418 }
```

\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.

```
4419 \newcommand*{\glsentryuseri}[1]{%
4420   \@gls@entry@field{#1}{useri}%
4421 }
```

\Glsentryuseri

```
4422 \newrobustcmd*{\Glsentryuseri}[1]{%
```

```

4423 \@Gls@entry@field{#1}{useri}%
4424 }

```

`\glentryuserii` Get the second user key (as specified by the user2 when the entry was defined). The argument is the label associated with the entry.

```

4425 \newcommand*{\glentryuserii}[1]{%
4426 \@Gls@entry@field{#1}{userii}%
4427 }

```

`\Glsentryuserii`

```

4428 \newrobustcmd*{\Glsentryuserii}[1]{%
4429 \@Gls@entry@field{#1}{userii}%
4430 }

```

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

```

4431 \newcommand*{\glentryuseriii}[1]{%
4432 \@Gls@entry@field{#1}{useriii}%
4433 }

```

`Glsentryuseriii`

```

4434 \newrobustcmd*{\Glsentryuseriii}[1]{%
4435 \@Gls@entry@field{#1}{useriii}%
4436 }

```

`\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.

```

4437 \newcommand*{\glentryuseriv}[1]{%
4438 \@Gls@entry@field{#1}{useriv}%
4439 }

```

`\Glsentryuseriv`

```

4440 \newrobustcmd*{\Glsentryuseriv}[1]{%
4441 \@Gls@entry@field{#1}{useriv}%
4442 }

```

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

```

4443 \newcommand*{\glentryuserv}[1]{%
4444 \@Gls@entry@field{#1}{userv}%
4445 }

```

`\Glsentryuserv`

```

4446 \newrobustcmd*{\Glsentryuserv}[1]{%
4447 \@Gls@entry@field{#1}{userv}%
4448 }

```

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

```
4449 \newcommand*{\glentryuservi}[1]{%
4450   \@gls@entry@field{#1}{uservi}%
4451 }
```

`\Glsentryuservi`

```
4452 \newrobustcmd*{\Glsentryuservi}[1]{%
4453   \@Gls@entry@field{#1}{uservi}%
4454 }
```

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

```
4455 \newcommand*{\glentryshort}[1]{\@gls@entry@field{#1}{short}}
```

`\Glsentryshort`

```
4456 \newrobustcmd*{\Glsentryshort}[1]{%
4457   \@Gls@entry@field{#1}{short}%
4458 }
```

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

```
4459 \newcommand*{\glentryshortpl}[1]{\@gls@entry@field{#1}{shortpl}}
```

`Glsentryshortpl`

```
4460 \newrobustcmd*{\Glsentryshortpl}[1]{%
4461   \@Gls@entry@field{#1}{shortpl}%
4462 }
```

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

```
4463 \newcommand*{\glentrylong}[1]{\@gls@entry@field{#1}{long}}
```

`\Glsentrylong`

```
4464 \newrobustcmd*{\Glsentrylong}[1]{%
4465   \@Gls@entry@field{#1}{long}%
4466 }
```

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

```
4467 \newcommand*{\glentrylongpl}[1]{\@gls@entry@field{#1}{longpl}}
```

`\Glsentrylongpl`

```
4468 \newrobustcmd*{\Glsentrylongpl}[1]{%
4469   \@Gls@entry@field{#1}{longpl}%
4470 }
```


Short cut macros to access full form:

`\glsentryfull`

```
4471 \newcommand*{\glsentryfull}[1]{%
4472   \acrfullformat{\glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%
4473 }
```

`\Glsentryfull`

```
4474 \newrobustcmd*{\Glsentryfull}[1]{%
4475   \acrfullformat{\Glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%
4476 }
```

`\glsentryfullpl`

```
4477 \newcommand*{\glsentryfullpl}[1]{%
4478   \acrfullformat{\glsentrylongpl{#1}}{\acronymfont{\glsentryshortpl{#1}}}%
4479 }
```

`\Glsentryfullpl`

```
4480 \newrobustcmd*{\Glsentryfullpl}[1]{%
4481   \acrfullformat{\Glsentrylongpl{#1}}{\acronymfont{\glsentryshortpl{#1}}}%
4482 }
```

`entrynumberlist` Displays the number list as is.

```
4483 \newcommand*{\glsentrynumberlist}[1]{%
4484   \glsdoifexists{#1}%
4485   {%
4486     \@gls@entry@field{#1}{numberlist}%
4487   }%
4488 }
```

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

```
4489 \@ifpackageloaded{hyperref} {%
4490   \newcommand*{\glsdisplaynumberlist}[1]{%
4491     \GlossariesWarning
4492     {%
4493       \string\glsdisplaynumberlist\space
4494       doesn't work with hyperref.^^JUsing
4495       \string\glsentrynumberlist\space instead%
4496     }%
4497     \glsentrynumberlist{#1}%
4498   }%
4499 }%
4500 {%
4501   \newcommand*{\glsdisplaynumberlist}[1]{%
4502     \glsdoifexists{#1}%
4503     {%
4504       \bgroup
```

```

4505 \edef\@glo@label{\glsdetoklabel{#1}}%
4506 \let\@org@glssnumberformat\glssnumberformat
4507 \def\glssnumberformat##1{##1}%
4508 \protected@edef\the@numberlist{%
4509   \csname glo@\@glo@label @numberlist\endcsname}%
4510 \def\@gls@numlist@sep{%
4511 \def\@gls@numlist@nextsep{%
4512 \def\@gls@numlist@lastsep{%
4513 \def\@gls@thislist{%
4514 \def\@gls@donext@def{%
4515 \renewcommand\do[1]{%
4516   \protected@edef\@gls@thislist{%
4517     \@gls@thislist
4518     \noexpand\@gls@numlist@sep
4519     ##1%
4520   }%
4521   \let\@gls@numlist@sep\@gls@numlist@nextsep
4522   \def\@gls@numlist@nextsep{\glssnumlistsep}%
4523   \@gls@donext@def
4524   \def\@gls@donext@def{%
4525     \def\@gls@numlist@lastsep{\glssnumlistlastsep}%
4526   }%
4527 }%
4528 \expandafter \glssnumlistparser \expandafter{\the@numberlist}%
4529 \let\@gls@numlist@sep\@gls@numlist@lastsep
4530 \@gls@thislist
4531 \egroup
4532 }%
4533 }
4534 }

```

`\glssnumlistsep`

```

4535 \newcommand*{\glssnumlistsep}{, }

```

`\glssnumlistlastsep`

```

4536 \newcommand*{\glssnumlistlastsep}{ \& }

```

`\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.

```

4537 \newcommand*{\gls hyperlink}[2][\glssentrytext{\@glo@label}]{%
4538   \def\@glo@label{#2}%
4539   \@gls link{\gls link prefix\glsdetoklabel{#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`:

```

4540 \define@key{glossadd}{counter}{\def\@gls@counter{#1}}
4541 \define@key{glossadd}{format}{\def\@glsnumberformat{#1}}

```

This key is only used by `\glsaddall`:

```

4542 \define@key{glossadd}{types}{\def\@glo@type{#1}}

```

```
\glsadd[<options>]{<label>}
```

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

`\glsadd`

```

4543 \newrobustcmd*{\glsadd}[2][]{%

```

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

```

4544 \@gls@adjustmode
4545 \glsdoifexists{#2}%
4546 {%
4547 \def\@glsnumberformat{glsnumberformat}%
4548 \edef\@gls@counter{\csname glo@%glsdetoklabel{#2}@counter\endcsname}%
4549 \setkeys{glossadd}{#1}%

```

Store the entry's counter in `\theglsentrycounter`

```

4550 \@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.

```

4551 \@@do@wrglossary{#2}%
4552 }%
4553 }

```

`@gls@adjustmode`

```

4554 \newcommand*{\@gls@adjustmode}{}
4555 \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`

```

4556 \newrobustcmd*{\glsaddall}[1][]{%
4557 \edef\@glo@type{\@glo@types}%
4558 \setkeys{glossadd}{#1}%
4559 \forallglsentries[\@glo@type]{\@glo@entry}{%

```

```

4560 \glsadd[#1]{\@glo@entry}%
4561 }%
4562 }

```

```

\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.

```

4563 \newrobustcmd*{\glsaddallunused}[1][\@glo@types]{%
4564 \forallglsentries[#1]{\@glo@entry}%
4565 {%
4566 \ifglsused{\@glo@entry}{\glsadd[format=glsignore]{\@glo@entry}}%
4567 }%
4568 }

```

`\glsignore`

```

4569 \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 `\glsgroupname` 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.

```

4570 \edef\glsopenbrace{\expandafter\@gobble\string\{ }

```

`\glsclosebrace` Define `\glsclosebrace` to make it easier to write an opening brace to a file.

```

4571 \edef\glsclosebrace{\expandafter\@gobble\string\} }

```

`\glsbackslash` Define `\glsbackslash` to make it easier to write a backslash to a file.

```

4572 \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.

```
4573 \edef\glsquote#1{\string"#1\string"}
```

`\glspercentchar` Define `\glspercentchar` to make it easier to write a percent character to a file.

```
4574 \edef\glspercentchar{\expandafter\@gobble\string\%}
```

`\glstildechar` Define `\glstildechar` to make it easier to write a tilde character to a file.

```
4575 \edef\glstildechar{\string~}
```

`@glsfirstletter` Define the first letter to come after the digits 0,...,9. Only required for xindy.

```
4576 \ifglsxindy
4577   \newcommand*{\@glsfirstletter}{A}
4578 \fi
```

`letterAfterDigits` Sets the first letter to come after the digits 0,...,9.

```
4579 \ifglsxindy
4580   \newcommand*{\GlsSetXdyFirstLetterAfterDigits}[1]{%
4581     \renewcommand*{\@glsfirstletter}{#1}}
4582 \else
4583   \newcommand*{\GlsSetXdyFirstLetterAfterDigits}[1]{%
4584     \glsnoxywarning\GlsSetXdyFirstLetterAfterDigits}
4585 \fi
```

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

```
4586 \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.

```
4587 \ifglsxindy
4588   \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
4589     \renewcommand*{\@glsminrange}{#1}}
4590 \else
4591   \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
4592     \glsnoxywarning\GlsSetXdyMinRangeLength}
4593 \fi
```

`\writeist`

```
4594 \ifglsxindy
  Code to use if xindy is required.
4595   \def\writeist{%
    Define write register if not already defined
4596     \ifundef{\glswrite}{\newwrite\glswrite}{}%
    Update attributes list
4597     \@gls@addpredefinedattributes
```

Open the file.

```
4598 \openout\glswrite=\istfilename
```

Write header comment at the start of the file

```
4599 \write\glswrite{;; xindy style file created by the glossaries
4600 package}%
4601 \write\glswrite{;; for document '\jobname' on
4602 \the\year-\the\month-\the\day}%
```

Specify the required styles

```
4603 \write\glswrite{^^J; required styles^^J}
4604 \@for\@xdystyle:=\@xdyrequiredstyles\do{%
4605 \ifx\@xdystyle\@empty
4606 \else
4607 \protected@write\glswrite{}{(require
4608 \string"\@xdystyle.xdy\string")}%
4609 \fi
4610 }%
```

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

```
4611 \write\glswrite{^^J%
4612 ; list of allowed attributes (number formats)^^J}%
4613 \write\glswrite{(define-attributes ((\@xdyattributes)))}%
```

Define any additional alphabets

```
4614 \write\glswrite{^^J; user defined alphabets^^J}%
4615 \write\glswrite{\@xdyuseralphabets}%
```

Define location classes.

```
4616 \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.

```
4617 \@for\@gls@classI:=\@gls@xdy@locationlist\do{%
```

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

```
4618 \protected@write\glswrite{}{(define-location-class
4619 \string"\@gls@classI\string"^^J\space\space\space
4620 (
4621 :sep "{}{"
4622 \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
4623 :sep "}"
4624 )
4625 ^^J\space\space\space
4626 :min-range-length \@glsminrange^^J%
4627 )
4628 }%
```

Nested iteration over all classes:

```
4629 {%
4630 \@for\@gls@classII:=\@gls@xdy@locationlist\do{%
4631 \protected@write\glswrite{}{(define-location-class
```

```

4632         \string"\@gls@classII-\@gls@classI\string"
4633         ^^J\space\space\space
4634     (
4635         :sep "{"
4636         \csname @gls@xdy@Lclass@\@gls@classII\endcsname\space
4637         :sep "{{"
4638         \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
4639         :sep "}"
4640     )
4641     ^^J\space\space\space
4642     :min-range-length \@glsminrange^^J%
4643 )
4644 }%
4645 }%
4646 }%
4647 }%

```

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

```

4648 \write\glswrite{^^J; user defined location classes}%
4649 \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.)

```

4650 \write\glswrite{^^J; define cross-reference class^^J}%
4651 \write\glswrite{(define-crossref-class \string"see\string"
4652     :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.)

```

4653 \write\glswrite{(markup-crossref-list
4654     :class \string"see\string"^^J\space\space\space
4655     :open \string"\string\glsseeformat\string"
4656     :close \string"{}\string")}%

```

Provide hook to write extra material here (used by `glossaries-extra` to define a `seealso` class).

```

4657 \@xdycrossrefhook

```

List the order to sort the classes.

```

4658 \write\glswrite{^^J; define the order of the location classes}%
4659 \write\glswrite{(define-location-class-order
4660     (\@xdylocationclassorder))}%

```

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

```

4661 \write\glswrite{^^J; define the glossary markup^^J}%
4662 \write\glswrite{(markup-index^^J\space\space\space
4663     :open \string"\string
4664     \glossarysection[\string\glossarytoctitle]{\string
4665     \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)

```

4666 \@for\@this@ctr:=\@xdycounters\do{%
4667   {%
4668     \@for\@this@attr:=\@xdyattributelist\do{%
4669       \protected@write\glswrite{}\string\providecommand*%
4670       \expandafter\string
4671       \csname glsX\@this@ctr X\@this@attr\endcsname[2]%
4672       {%
4673         \string\setentrycounter
4674         [\expandafter@gobble\string\#1]{\@this@ctr}%
4675         \expandafter\string
4676         \csname\@this@attr\endcsname
4677         {\expandafter@gobble\string\#2}%
4678       }%
4679     }%
4680   }%
4681 }%
4682 }%

```

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

```

4683 \write\glswrite{%
4684   \string\begin
4685   {theglossary}\string\glossaryheader\glstildechar n\string" ^^J\space
4686   \space\space:close \string"\glspersentchar\glstildechar n\string
4687   \end{theglossary}\string\glossarypostamble
4688   \glstildechar n\string" ^^J\space\space\space
4689   :tree)}}%

```

Specify what to put between letter groups

```

4690 \write\glswrite{(markup-letter-group-list
4691   :sep \string"\string\glsgroupskip\glstildechar n\string")}%

```

Specify what to put between entries

```

4692 \write\glswrite{(markup-indexentry
4693   :open \string"\string\relax \string\glsresetentrylist
4694   \glstildechar n\string")}%

```

Specify how to format entries

```

4695 \write\glswrite{(markup-locclass-list :open
4696   \string"\glsopenbrace\string\glossaryentrynumbers
4697   \glsopenbrace\string\relax\space \string"^^J\space\space\space
4698   :sep \string", \string"
4699   :close \string"\glsclosebrace\glsclosebrace\string")}%

```

Specify how to separate location numbers

```

4700 \write\glswrite{(markup-locref-list
4701   :sep \string"\string\delimN\space\string")}%

```

Specify how to indicate location ranges

```

4702 \write\glswrite{(markup-range
4703   :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.

```

4704 \@onelevel@sanitize\gls@suffixF
4705 \@onelevel@sanitize\gls@suffixFF
4706 \ifx\gls@suffixF\@empty
4707 \else
4708 \write\glswrite{(markup-range
4709 :close "\gls@suffixF" :length 1 :ignore-end)}}%
4710 \fi
4711 \ifx\gls@suffixFF\@empty
4712 \else
4713 \write\glswrite{(markup-range
4714 :close "\gls@suffixFF" :length 2 :ignore-end)}}%
4715 \fi

```

Specify how to format locations.

```

4716 \write\glswrite{^^J; define format to use for locations^^J}%
4717 \write\glswrite{\@xdylocref}%

```

Specify how to separate letter groups.

```

4718 \write\glswrite{^^J; define letter group list format^^J}%
4719 \write\glswrite{(markup-letter-group-list
4720 :sep \string"\string\glsgroupskip\glstildechar n\string")}%

```

Define letter group headings.

```

4721 \write\glswrite{^^J; letter group headings^^J}%
4722 \write\glswrite{(markup-letter-group
4723 :open-head \string"\string\glsgroupheading
4724 \glsopenbrace\string"^^J\space\space\space
4725 :close-head \string"\glsclosebrace\string")}%

```

Define additional letter groups.

```

4726 \write\glswrite{^^J; additional letter groups^^J}%
4727 \write\glswrite{\@xdylettergroups}%

```

Define additional sort rules

```

4728 \write\glswrite{^^J; additional sort rules^^J}
4729 \write\glswrite{\@xdysortrules}%

```

Hook for any additional information:

```

4730 \@gls@writeisthook

```

Close the style file

```

4731 \closeout\glswrite

```

Suppress any further calls.

```

4732 \let\writeist\relax
4733 }
4734 \else

```

Code to use if makeindex is required.

```

4735 \edef\@gls@actualchar{\string?}
4736 \edef\@gls@encapchar{\string|}
4737 \edef\@gls@levelchar{\string!}
4738 \edef\@gls@quotechar{\string"}%
4739 \let\GlsSetQuote\gls@nosetquote
4740 \def\writeist{\relax
4741 \ifundef{\glswrite}{\newwrite\glswrite}{}\relax
4742 \openout\glswrite=\istfilename
4743 \write\glswrite{\glspersentchar\space makeindex style file
4744 created by the glossaries package}
4745 \write\glswrite{\glspersentchar\space for document
4746 '\jobname' on \the\year-\the\month-\the\day}
4747 \write\glswrite{actual '\@gls@actualchar'}
4748 \write\glswrite{encap '\@gls@encapchar'}
4749 \write\glswrite{level '\@gls@levelchar'}
4750 \write\glswrite{quote '\@gls@quotechar'}
4751 \write\glswrite{keyword \string\string\glossaryentry\string}
4752 \write\glswrite{preamble \string\string\glossarysection[\string
4753 \glossarytoctitle]{\string\glossarytitle}\string
4754 \glossarypreamble\string\n\string\begin{theglossary}\string
4755 \glossaryheader\string\n\string}
4756 \write\glswrite{postamble \string\string%\string\n\string
4757 \end{theglossary}\string\glossarypostamble\string\n
4758 \string}
4759 \write\glswrite{group_skip \string\string\glsgroupskip\string\n
4760 \string}
4761 \write\glswrite{item_0 \string\string%\string\n\string}
4762 \write\glswrite{item_1 \string\string%\string\n\string}
4763 \write\glswrite{item_2 \string\string%\string\n\string}
4764 \write\glswrite{item_01 \string\string%\string\n\string}
4765 \write\glswrite{item_x1
4766 \string\string\relax \string\glsresetentrylist\string\n
4767 \string}
4768 \write\glswrite{item_12 \string\string%\string\n\string}
4769 \write\glswrite{item_x2
4770 \string\string\relax \string\glsresetentrylist\string\n
4771 \string}

4772 \write\glswrite{delim_0 \string\string\{\string
4773 \glossaryentrynumbers\string\{\string\relax \string}
4774 \write\glswrite{delim_1 \string\string\{\string
4775 \glossaryentrynumbers\string\{\string\relax \string}
4776 \write\glswrite{delim_2 \string\string\{\string
4777 \glossaryentrynumbers\string\{\string\relax \string}
4778 \write\glswrite{delim_t \string\string\}\string\}\string}
4779 \write\glswrite{delim_n \string\string\delimN \string}
4780 \write\glswrite{delim_r \string\string\delimR \string}
4781 \write\glswrite{headings_flag 1}
4782 \write\glswrite{heading_prefix

```

```

4783     \string"\string\glsgroupheading\string\{\string"}
4784 \write\glswrite{heading_suffix
4785     \string"\string\}\string\relax
4786     \string\glsgroupheading\string"}
4787 \write\glswrite{symhead_positive \string"glssymbols\string"}
4788 \write\glswrite{numhead_positive \string"glsgroupheading\string"}
4789 \write\glswrite{page_compositor \string"glsgroupheading\string"}
4790 \@gls@escbsdq\gls@suffixF
4791 \@gls@escbsdq\gls@suffixFF
4792 \ifx\gls@suffixF\@empty
4793 \else
4794     \write\glswrite{suffix_2p \string"\gls@suffixF\string"}
4795 \fi
4796 \ifx\gls@suffixFF\@empty
4797 \else
4798     \write\glswrite{suffix_3p \string"\gls@suffixFF\string"}
4799 \fi

```

Hook for any additional information:

```

4800 \@gls@writeisthook

```

Close the file and disable \writeist.

```

4801 \closeout\glswrite
4802 \let\writeist\relax
4803 }
4804 \fi

```

SetWriteIstHook Allow user to append information to the style file.

```

4805 \newcommand*\GlsSetWriteIstHook[1]{\renewcommand*\@gls@writeisthook{#1}}
4806 \@onlypremake\GlsSetWriteIstHook

```

ls@writeisthook

```

4807 \newcommand*\@gls@writeisthook{}

```

\GlsSetQuote Allow user to set the makeindex quote character. This is primarily for ngerman users who want to use makeindex's -g option.

```

4808 \ifglxindy
4809 \newcommand*\GlsSetQuote[1]{\glsnomakeindexwarning\GlsSetQuote}
4810 \newcommand*\Gls@nosetquote[1]{\glsnomakeindexwarning\GlsSetQuote}
4811 \else
4812 \newcommand*\GlsSetQuote[1]{\edef\@gls@quotechar{\string#1}%

```

If German is in use, set the extra makeindex option so makeglossaries can pick it up.

```

4813 \@ifpackageloaded{tracklang}%
4814 {%
4815     \IfTrackedLanguage{german}%
4816     {%
4817         \def\@gls@extramakeindexopts{-g}%
4818     }%
4819 }%

```

```

4820 }%
4821 {}%

Need to redefine \@gls@checkquote
4822 \edef\@gls@docheckquotedef{%
4823   \noexpand\def\noexpand\@gls@checkquote####1#1####2#1####3\noexpand\null{%
4824     \noexpand\@gls@tmpb=\noexpand\expandafter{\noexpand\@gls@checkedmkidx}%
4825     \noexpand\toks@={####1}%
4826     \noexpand\ifx\noexpand\null####2\noexpand\null
4827     \noexpand\ifx\noexpand\null####3\noexpand\null
4828     \noexpand\edef\noexpand\@gls@checkedmkidx{%
4829       \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
4830     \noexpand\def\noexpand\@gls@checkquote{\noexpand\relax}%
4831     \noexpand\else
4832     \noexpand\edef\noexpand\@gls@checkedmkidx{%
4833       \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
4834       \noexpand\@gls@quotechar\noexpand\@gls@quotechar
4835       \noexpand\@gls@quotechar\noexpand\@gls@quotechar}%
4836     \noexpand\def\noexpand\@gls@checkquote{%
4837       \noexpand\@gls@checkquote####3\noexpand\null}%
4838     \noexpand\fi
4839   \noexpand\else
4840     \noexpand\edef\noexpand\@gls@checkedmkidx{%
4841       \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
4842       \noexpand\@gls@quotechar\noexpand\@gls@quotechar}%
4843     \noexpand\ifx\noexpand\null####3\noexpand\null
4844     \noexpand\def\noexpand\@gls@checkquote{%
4845       \noexpand\@gls@checkquote####2#1#1\noexpand\null}%
4846     \noexpand\else
4847     \noexpand\def\noexpand\@gls@checkquote{%
4848       \noexpand\@gls@checkquote####2#1####3\noexpand\null}%
4849     \noexpand\fi
4850   \noexpand\fi
4851   \noexpand\@gls@checkquote
4852 }%
4853 }%
4854 \@gls@docheckquotedef
4855 \edef\@gls@docheckquotedef{%
4856   \noexpand\renewcommand{\noexpand\@gls@checkmkidxchars}[1]{%
4857     \noexpand\def\noexpand\@gls@checkedmkidx{%
4858       \noexpand\expandafter\noexpand\@gls@checkquote####1\noexpand\@nil
4859       #1#1\noexpand\null
4860       \noexpand\expandafter\noexpand\@gls@updatechecked
4861       \noexpand\@gls@checkedmkidx{####1}%
4862       \noexpand\def\noexpand\@gls@checkedmkidx{%
4863         \noexpand\expandafter\noexpand\@gls@checkescquote####1\noexpand\@nil
4864         \expandonce{\csname#1\endcsname}\expandonce{\csname#1\endcsname}%
4865         \noexpand\null
4866       \noexpand\expandafter\noexpand\@gls@updatechecked
4867       \noexpand\@gls@checkedmkidx{####1}%

```

```

4868 \noexpand\def\noexpand\@gls@checkedmkidx{%
4869 \noexpand\expandafter\noexpand\@gls@checkescactual####1\noexpand\@nil
4870 \noexpand\?\noexpand\?\noexpand\null
4871 \noexpand\expandafter\noexpand\@gls@updatechecked
4872 \noexpand\@gls@checkedmkidx{####1}%
4873 \noexpand\def\noexpand\@gls@checkedmkidx{%
4874 \noexpand\expandafter\noexpand\@gls@checkactual####1\noexpand\@nil
4875 \noexpand?\noexpand?\noexpand\null
4876 \noexpand\expandafter\noexpand\@gls@updatechecked
4877 \noexpand\@gls@checkedmkidx{####1}%
4878 \noexpand\def\noexpand\@gls@checkedmkidx{%
4879 \noexpand\expandafter\noexpand\@gls@checkbar####1\noexpand\@nil
4880 \noexpand|\noexpand|\noexpand\null
4881 \noexpand\expandafter\noexpand\@gls@updatechecked
4882 \noexpand\@gls@checkedmkidx{####1}%
4883 \noexpand\def\noexpand\@gls@checkedmkidx{%
4884 \noexpand\expandafter\noexpand\@gls@checkescbar####1\noexpand\@nil
4885 \noexpand||\noexpand||\noexpand\null
4886 \noexpand\expandafter\noexpand\@gls@updatechecked
4887 \noexpand\@gls@checkedmkidx{####1}%
4888 \noexpand\def\noexpand\@gls@checkedmkidx{%
4889 \noexpand\expandafter\noexpand\@gls@checklevel####1\noexpand\@nil
4890 \noexpand!\noexpand!\noexpand\null
4891 \noexpand\expandafter\noexpand\@gls@updatechecked
4892 \noexpand\@gls@checkedmkidx{####1}%
4893 }%
4894 }%
4895 \@gls@docheckquotedef
4896 \edef\@gls@docheckquotedef{%
4897 \noexpand\def\noexpand\@gls@checkescquote####1%
4898 \expandonce{\csname#1\endcsname}####2\expandonce{\csname#1\endcsname}%
4899 ####3\noexpand\null{%
4900 \noexpand\@gls@tmpb=\noexpand\expandafter{\noexpand\@gls@checkedmkidx}%
4901 \noexpand\toks@={####1}%
4902 \noexpand\ifx\noexpand\null####2\noexpand\null
4903 \noexpand\ifx\noexpand\null####3\noexpand\null
4904 \noexpand\edef\noexpand\@gls@checkedmkidx{%
4905 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
4906 \noexpand\def\noexpand\@gls@checkescquote{\noexpand\relax}%
4907 \noexpand\else
4908 \noexpand\edef\noexpand\@gls@checkedmkidx{%
4909 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
4910 \noexpand\@gls@quotechar\noexpand\string\expandonce{%
4911 \csname#1\endcsname}\noexpand\@gls@quotechar
4912 \noexpand\@gls@quotechar\noexpand\string\expandonce{%
4913 \csname#1\endcsname}\noexpand\@gls@quotechar}%
4914 \noexpand\def\noexpand\@gls@checkescquote{%
4915 \noexpand\@gls@checkescquote####3\noexpand\null}%
4916 \noexpand\fi

```

```

4917 \noexpand\else
4918 \noexpand\edef\noexpand\@gls@checkedmkidx{%
4919 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
4920 \noexpand\@gls@quotearch\noexpand\string
4921 \expandonce{\csname#1\endcsname}\noexpand\@gls@quotearch}%
4922 \noexpand\ifx\noexpand\null####3\noexpand\null
4923 \noexpand\def\noexpand\@gls@checkescquote{%
4924 \noexpand\@gls@checkescquote####2\expandonce{\csname#1\endcsname}%
4925 \expandonce{\csname#1\endcsname}\noexpand\null}%
4926 \noexpand\else
4927 \noexpand\def\noexpand\@gls@checkescquote{%
4928 \noexpand\@gls@checkescquote####2\expandonce{\csname#1\endcsname}%
4929 ####3\noexpand\null}%
4930 \noexpand\fi
4931 \noexpand\fi
4932 \noexpand\@gls@checkescquote
4933 }%
4934 }%
4935 \@gls@docheckquotedef
4936 }
4937 \newcommand*{\gls@nosetquote}[1]{\PackageError{glossaries}%
4938 {\string\GlsSetQuote\space not permitted here}%
4939 {Move \string\GlsSetQuote\space earlier in the preamble, as
4940 soon as possible after glossaries.sty has been loaded}}
4941 \fi

```

ramakeindexopts

```

4942 \newcommand*{\@gls@extramakeindexopts}[1]{

```

The command `\noist` will suppress the creation of the `.ist` file. Obviously you need to use this command before `\writeist` to have any effect.

`\noist`

```

4943 \newcommand{\noist}{%
  Update attributes list
4944 \@gls@addpredefinedattributes
4945 \let\writeist\relax
4946 }

```

`\@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

```
4947 \newcommand*{\@makeglossary}[1]{%
4948   \ifglossaryexists{#1}%
4949   {%
```

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

```
4950   \ifglssavewrites
4951     \expandafter\newtoks\csname glo@#1@filetok\endcsname
4952   \else
4953     \expandafter\newwrite\csname glo@#1@file\endcsname
4954     \expandafter\@glsopenfile\csname glo@#1@file\endcsname{#1}%
4955   \fi
4956   \@gls@renewglossary
4957   \writeist
4958 }%
4959 {%
4960   \PackageError{glossaries}%
4961   {Glossary type ‘#1’ not defined}%
4962   {New glossaries must be defined before using \string\makeglossary}%
4963 }%
4964 }
```

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

```
4965 \newcommand*{\@glsopenfile}[2]{%
4966   \immediate\openout#1=\jobname.\csname @glotype@#2@out\endcsname
4967   \PackageInfo{glossaries}{Writing glossary file
4968     \jobname.\csname @glotype@#2@out\endcsname}%
4969 }
```

\@closegls

```
4970 \newcommand*{\@closegls}[1]{%
4971   \closeout\csname glo@#1@file\endcsname
4972 }
```

\@gls@automake

```
4973 \ifglsxindy
4974 \newcommand*{\@gls@automake}[1]{%
4975   \ifglossaryexists{#1}
4976   {%
4977     \@closegls{#1}%
4978     \ifdefstring{\glsorder}{letter}%
4979     {\def\@gls@order{-M ord/letorder }}%
4980     {\let\@gls@order\@empty}%
4981     \ifcsundef{@xdy@#1@language}%
4982     {\let\@gls@langmod\@xdy@main@language}%
4983     {\letcs\@gls@langmod{@xdy@#1@language}}%
4984     \edef\@gls@dothiswrite{\noexpand\write18{xindy
4985       -I xindy
```

```

4986      \@gls@order
4987      -L \@gls@langmod\space
4988      -M \@gls@istfilebase\space
4989      -C \@gls@codepage\space
4990      -t \jobname.\csuse{@glotype@#1@log}
4991      -o \jobname.\csuse{@glotype@#1@in}
4992      \jobname.\csuse{@glotype@#1@out}}}%
4993  }%
4994  \@gls@dothiswrite
4995  }%
4996  {%
4997      \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
4998  }%
4999  }
5000 \else
5001 \newcommand*{\@gls@automake}[1]{%
5002     \ifglossaryexists{#1}
5003     {%
5004         \@closegls{#1}%
5005         \ifdefstring{\glsorder}{letter}%
5006             {\def\@gls@order{-l }}%
5007             {\let\@gls@order\@empty}%
5008         \edef\@gls@dothiswrite{\noexpand\write18{makeindex \@gls@order
5009             -s \istfilename\space
5010             -t \jobname.\csuse{@glotype@#1@log}
5011             -o \jobname.\csuse{@glotype@#1@in}
5012             \jobname.\csuse{@glotype@#1@out}}}%
5013     }%
5014     \@gls@dothiswrite
5015     }%
5016     {%
5017         \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
5018     }%
5019 }
5020 \fi

```

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

```
5021 \newcommand*{\@warn@nomakeglossaries}{}
```

Only use this if warning if `\printglossary` has been used without `\makeglossaries`

```
5022 \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`

```
5023 \newcommand*{\makeglossaries}{%
```

Define the write used for style file also used for all other output files if `savewrites=true`.


```
5024 \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.

```
5025 \protected@write\@auxout{}{\string\providecommand\string\@glsorder[1]{}%
5026 \protected@write\@auxout{}{\string\providecommand\string\@istfilename[1]{}%}
```

If \@@gls@extramakeindexopts has been defined, write it:

```
5027 \ifundef\@@gls@extramakeindexopts
5028 {}%
5029 {%
5030 \protected@write\@auxout{}{\string\providecommand
5031 \string\@gls@extramakeindexopts[1]{}%
5032 \protected@write\@auxout{}{\string\@gls@extramakeindexopts
5033 {\@gls@extramakeindexopts}}%
5034 }%
```

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

```
5035 \protected@write\@auxout{}{\string\@istfilename{\istfilename}}%
5036 \protected@write\@auxout{}{\string\@glsorder{\glsorder}}%
```

Iterate through each glossary type and activate it.

```
5037 \@for\@glo@type:=\@glo@types\do{%
5038 \ifthenelse{equal{\@glo@type}{}}{ }{%
5039 \makeglossary{\@glo@type}}%
5040 }%
```

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

```
5041 \renewcommand*\newglossary[4][]{%
5042 \PackageError{glossaries}{New glossaries
5043 must be created before \makeglossaries}{You need
5044 to move \string\makeglossaries\space after all your
5045 \string\newglossary\space commands}}%
```

Any subsequent instances of this command should have no effect

```
5046 \let\makeglossary\relax
5047 \let\makeglossary\relax
5048 \let\makeglossaries\relax
```

Disable all commands that have no effect after \makeglossaries

```
5049 \@disable@onlypremakeg
```

Allow see key:

```
5050 \let\gls@checkseeallowed\relax
```

Suppress warning about no \makeglossaries

```
5051 \let\warn@nomakeglossaries\relax
```

Activate warning about missing \printglossary

```
5052 \def\warn@noprntglossary{%
5053 \ifdefstring{\@glo@types}{,}%
5054 {%
5055 \GlossariesWarningNoLine{No glossaries have been defined}}%
```

```

5056 }%
5057 {%
5058     \GlossariesWarningNoLine{No \string\printglossary\space
5059     or \string\printglossaries\space
5060     found. ^^J(Remove \string\makeglossaries\space if you
5061     don't want any glossaries.) ^^JThis document will not
5062     have a glossary}%
5063 }%
5064 }%

```

Declare list parser for \glsdisplaynumberlist

```

5065 \ifglssavenumberlist
5066     \edef\@gls@dodolistparser{\noexpand\DeclareListParser
5067     {\noexpand\glsnumlistparser}{\delimN}}}%
5068 \@gls@dodolistparser
5069 \fi

```

Prevent user from also using \makenoidxglossaries

```

5070 \let\makenoidxglossaries\@no@makeglossaries

```

Prohibit sort key in printgloss family:

```

5071 \renewcommand*{\@printgloss@setsort}{%
5072     \let\@glo@assign@sortkey\@glo@no@assign@sortkey
5073 }%

```

Check the automake setting:

```

5074 \ifglsautomake
5075     \renewcommand*{\@gls@doautomake}{%
5076         \@for\@gls@type:=\@glo@types\do{%
5077             \ifdefempty{\@gls@type}{}%
5078             {\@gls@automake{\@gls@type}}}%
5079         }%
5080     }%
5081 \fi

```

Check the sort setting:

```

5082 \@glo@check@sortallowed\makeglossaries
5083 }

```

Must occur in the preamble:

```

5084 \@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`

```

5085 \let\makeglossary\makeglossaries

```

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

```
5086 \AtEndDocument{%
5087   \warn@nomakeglossaries
5088   \warn@noprintglossary
5089 }
```

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

```
5090 \newcommand*{\makenoidxglossaries}{%
```

Redefine empty glossary warning:

```
5091   \renewcommand{\@gls@noref@warn}[1]{%
5092     \GlossariesWarning{Empty glossary for
5093       \string\printnoidxglossary[type={##1}].
5094     Rerun may be required (or you may have forgotten to use
5095     commands like \string\gls)}%
5096   }%
```

Don't escape makeindex/xindy characters

```
5097   \let\@gls@checkmkidxchars\@gobble
```

Write glossary information to aux instead of glossary files

```
5098   \let\@do@wrglossary\gls@noidxglossary
```

Switch on group headings that use the character code:

```
5099   \let\@gls@getgrouptitle\@gls@noidx@getgrouptitle
```

Allow see key:

```
5100   \let\gls@checkseeallowed\relax
```

Redefine cross-referencing macro:

```
5101   \renewcommand{\@do@seeglossary}[2]{%
5102     \edef\@gls@label{\glsdetoklabel{##1}}%
5103     \protected@write\@auxout{}{%
5104       \string\@gls@reference
5105       {\csname glo@\@gls@label @type\endcsname}%
5106       {\@gls@label}%
5107       {%
5108         \string\glsseeformat##2}%
5109       }%
5110     }%
5111   }%
```

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.

```
5112   \AtBeginDocument
5113   {%
5114     \write\@auxout{\string\providecommand\string\@gls@reference[3]{}}%
5115   }%
```

Change warning about no glossaries

```
5116 \def\warn@noprintglossary{%
5117   \GlossariesWarningNoLine{No \string\printnoidxglossary\space
5118     or \string\printnoidxglossaries ^^J
5119     found. (Remove \string\makenoidxglossaries\space if you
5120     don't want any glossaries.)^^JThis document will not have a glossary}%
5121 }%
```

Suppress warning about no \makeglossaries

```
5122 \let\warn@nomakeglossaries\relax
```

Prevent user from also using \makeglossaries

```
5123 \let\makeglossaries\@no@makeglossaries
```

Allow sort key in printgloss family:

```
5124 \renewcommand*{@printgloss@setsort}{%
5125   \let@glo@assign@sortkey@@glo@assign@sortkey
```

Initialise default sort order:

```
5126   \def@glo@sorttype{\@glo@default@sorttype}%
5127 }%
```

All entries must be defined in the preamble:

```
5128 \renewcommand*\new@glossaryentry[2]{%
5129   \PackageError{glossaries}{Glossary entries must be
5130     defined in the preamble^^Jwhen you use
5131     \string\makenoidxglossaries}%
5132   {Either move your definitions to the preamble or use
5133     \string\makeglossaries}%
5134 }%
```

Redefine \glsentrynumberlist

```
5135 \renewcommand*{\glsentrynumberlist}[1]{%
5136   \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
5137   \ifdef\@gls@loclist
5138     {%
5139       \glsnoidxloclist{\@gls@loclist}%
5140     }%
5141     {%
5142       ??\glsdoifexists{##1}%
5143       {%
5144         \GlossariesWarning{Missing location list for '##1'. Either
5145           a rerun is required or you haven't referenced the entry}%
5146       }%
5147     }%
5148 }%
```

Redefine \glsdisplaynumberlist

```
5149 \renewcommand*{\glsdisplaynumberlist}[1]{%
5150   \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
5151   \ifdef\@gls@loclist
5152     {%
```

```

5153     \def\@gls@noidxloclist@sep{%
5154     \def\@gls@noidxloclist@sep{%
5155     \def\@gls@noidxloclist@sep{%
5156     \glsnumlistsep
5157     }%
5158     \def\@gls@noidxloclist@finalsep{\glsnumlistlastsep}%
5159     }%
5160     }%
5161     \def\@gls@noidxloclist@finalsep{}}%
5162     \def\@gls@noidxloclist@prev{}}%
5163     \forlistloop{\glsnoidxdisplayloclisthandler}{\@gls@loclist}%
5164     \@gls@noidxloclist@finalsep
5165     \@gls@noidxloclist@prev
5166     }%
5167     {%
5168     ??\glsdoifexists{##1}%
5169     {%
5170     \GlossariesWarning{Missing location list for ‘##1’. Either
5171     a rerun is required or you haven’t referenced the entry}%
5172     }%
5173     }%
5174     }%

```

Provide a generic way of iterating through the number list:

```

5175 \renewcommand*\glsnumberlistloop}[3]{%
5176 \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
5177 \let\@gls@org@glsnoidxdisplayloc\glsnoidxdisplayloc
5178 \let\@gls@org@glsseeformat\glsseeformat
5179 \let\glsnoidxdisplayloc##2\relax
5180 \let\glsseeformat##3\relax
5181 \ifdef\@gls@loclist
5182 {%
5183 \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
5184 }%
5185 {%
5186 ??\glsdoifexists{##1}%
5187 {%
5188 \GlossariesWarning{Missing location list for ‘##1’. Either
5189 a rerun is required or you haven’t referenced the entry}%
5190 }%
5191 }%
5192 \let\glsnoidxdisplayloc\@gls@org@glsnoidxdisplayloc
5193 \let\glsseeformat\@gls@org@glsseeformat
5194 }%

```

Modify sanitize sort function

```

5195 \let\@gls@sanitizesort\@gls@noidx@sanitizesort
5196 \let\@gls@nosanitizesort\@gls@noidx@nosanitizesort
5197 \@gls@noidx@setsanitizesort

```

Check sort option allowed.

```

5198 \glo@check@sortallowed\makenoidxglossaries
5199 }

```

Preamble-only command:

```

5200 \@onlypreamble{\makenoidxglossaries}

```

```

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

```

```

5201 \newcommand*{\glsnumberlistloop}[2]{%
5202   \PackageError{glossaries}{\string\glsnumberlistloop\space
5203     only works with \string\makenoidxglossaries}{}%
5204 }

```

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

```

5205 \newcommand*{\glsnoidxnumberlistloophandler}[1]{%
5206   #1%
5207 }

```

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

```

5208 \newcommand*{\@no@makeglossaries}{%
5209   \PackageError{glossaries}{You can't use both
5210     \string\makeglossaries\space and \string\makenoidxglossaries}%
5211   {Either use one or other (or none) of those commands but not both
5212     together.}%
5213 }

```

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

```

5214 \newcommand{\@gls@noref@warn}[1]{%
5215   \GlossariesWarning{\string\makenoidxglossaries\space
5216     is required to make \string\printnoidxglossary[type={#1}] work}%
5217 }

```

`s@noidxglossary` Write the glossary information to the aux file:

```

5218 \newcommand*{\gls@noidxglossary}{%
5219   \protected@write\@auxout{}{%
5220     \string\@gls@reference
5221       {\csname glo@\@gls@label @type\endcsname}%
5222       {\@gls@label}%
5223       {\string\glsnoidxdisplayloc
5224         {\@glo@counterprefix}%
5225         {\@gls@counter}%
5226         {\@glsnumberformat}%
5227         {\@glslocref}%
5228       }%
5229   }%
5230 }

```

1.14 Writing information to associated files

`\istfile` Deprecated.

```
5231 \def\istfile{\glswrite}
```

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

```
5232 \AtEndDocument{%
```

```
5233   \glswritefiles
```

```
5234 }
```

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

```
5235 \newcommand*{\@glswritefiles}{%
```

Iterate through all the glossaries

```
5236   \forallglossaries{\@glo@type}{%
```

Check for empty glossaries (patch provided by Patrick Häcker)

```
5237     \ifcsundef{glo@\@glo@type @filetok}%
```

```
5238     {%
```

```
5239       \def\gls@tmp{}%
```

```
5240     }%
```

```
5241     {%
```

```
5242       \edef\gls@tmp{\expandafter\the
```

```
5243         \csname glo@\@glo@type @filetok\endcsname}%
```

```
5244     }%
```

```
5245     \ifx\gls@tmp\@empty
```

```
5246       \ifx\@glo@type\glsdefaulttype
```

```
5247         \GlossariesWarningNoLine{Glossary ‘\@glo@type’ has no
```

```
5248           entries.^^JRemember to use package option ‘nomain’ if
```

```
5249 you
```

```
5250           don’t want to^^Juse the main glossary}%
```

```
5251       \else
```

```
5252         \GlossariesWarningNoLine{Glossary ‘\@glo@type’ has no
```

```
5253           entries}%
```

```
5254       \fi
```

```
5255     \else
```

```
5256       \@glsopenfile{\glswrite}{\@glo@type}%
```

```
5257       \immediate\write\glswrite{%
```

```
5258         \expandafter\the
```

```
5259         \csname glo@\@glo@type @filetok\endcsname}%
```

```
5260       \immediate\closeout\glswrite
```

```
5261     \fi
```

```
5262   }%
```

```
5263 }
```

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`

```
5264 \if@gls@docloaded
5265 \else
5266   \renewcommand*{\glossary}[1][main]{\gls@glossary{#1}}
5267 \fi
```

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

`\gls@glossary`

```
5268 \newcommand*{\gls@glossary}[1]{%
5269   \@gls@glossary{#1}%
5270 }
```

`\@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.

```
5271 \newcommand*{\@gls@glossary}[2]{%
5272   \if@gls@debug
5273     \PackageInfo{glossaries}{wrglossary(#1)(#2)}%
5274   \fi
5275   \index{#2}%
5276 }
```

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`

```
5277 \newcommand{\@gls@renewglossary}{%
5278   \gdef\@gls@glossary##1{\@bsphack\beginingroup\gls@wrglossary{##1}}%
5279   \let\@gls@renewglossary\empty
5280 }
```

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`

```
5281 \newcommand*{\gls@wrglossary}[2]{%
5282   \ifglssavewrites
5283     \protected@edef\@gls@tmp{\the\csname glo@#1@filetok\endcsname#2}%
5284     \expandafter\global\expandafter\csname glo@#1@filetok\endcsname
```



```

5285     \expandafter{\@gls@tmp^^J}%
5286 \else

5287     \ifcsdef{glo@#1@file}%
5288     {%
5289         \expandafter\protected@write\csname glo@#1@file\endcsname{%
5290             \gls@disablepagerefexpansion}{#2}%
5291     }%
5292     {%
5293         \ifignoredglossary{#1}{}%
5294         {%
5295             \GlossariesWarning{No file defined for glossary ‘#1’}%
5296         }%
5297     }%
5298 \fi
5299 \endgroup\@esphack
5300 }

```

\do@wrglossary

```

5301 \newcommand*{\do@wrglossary}[1]{%
5302     \glswriteentry{#1}{\do@wrglossary{#1}}%
5303 }

```

\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.

```

5304 \newcommand*{\glswriteentry}[2]{%
5305     \ifglsindexonlyfirst
5306         \ifglsused{#1}{#2}%
5307     \else
5308         #2%
5309     \fi
5310 }

```

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

```

5311 \newcommand{\gls@protected@pagefmts}{%
5312     \gls@numberpage,\gls@alphpage,\gls@Alphpage,\gls@romanpage,\gls@Romanpage,\gls@arabicpage%
5313 }

```

pagerefexpansion

```

5314 \newcommand*{\gls@disablepagerefexpansion}{%
5315     \@for\@gls@this:=\gls@protected@pagefmts\do
5316     {%
5317         \expandafter\let\@gls@this\relax
5318     }%
5319 }

```

\gls@alphpage

```

5320 \newcommand*{\gls@alphpage}{\@alph\c@page}

```

```

\gls@Alphpage
5321 \newcommand*{\gls@Alphpage}{\@Alph\c@page}

\gls@numberpage
5322 \newcommand*{\gls@numberpage}{\number\c@page}

\gls@arabicpage
5323 \newcommand*{\gls@arabicpage}{\@arabic\c@page}

\gls@romanpage
5324 \newcommand*{\gls@romanpage}{\romannumeral\c@page}

\gls@Romanpage
5325 \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).

5326 \newcommand*{\glsaddprotectedpagefmt}[1]{%
5327   \eappto\gls@protected@pagefmts{\expandonce{\csname gls#1page\endcsname}}%
5328   \csedef{gls#1page}{\expandonce{\csname#1\endcsname}\noexpand\c@page}%
5329   \eappto\@wrglossarynumberhook{%
5330     \noexpand\let\expandonce{\csname org@gls#1\endcsname}%
5331     \expandonce{\csname#1\endcsname}%
5332     \noexpand\def\expandonce{\csname#1\endcsname}{%
5333       \noexpand\@wrglossary@pageformat
5334       \expandonce{\csname gls#1page\endcsname}%
5335       \expandonce{\csname org@gls#1\endcsname}%
5336     }%
5337   }%
5338 }

ssarynumberhook Hook used by \@@do@wrglossary
5339 \newcommand*\@wrglossarynumberhook{}

sary@pageformat
5340 \newcommand{\@wrglossary@pageformat}[3]{%
5341   \ifx#3\c@page #1\else #2#3\fi
5342 }

owprimitivemods Conditional to determine whether or not \@@do@wrglossary should be allowed to temporar-
ily redefine \the and \number.
5343 \newif\ifglswrallowprimitivemods
5344 \glswrallowprimitivemodstrue

```

`@do@wrglossary` Write the glossary entry in the appropriate format. (Need to set `\@glsnumberformat` and `\@gls@counter` prior to use.) The argument is the entry's label.

```
5345 \newcommand*{\@do@wrglossary}[1]{%
5346   \begingroup
```

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

```
5347   \let\orgthe\the
5348   \let\orgnumber\number

5349   \let\orgarabic\@arabic
5350   \let\orgromannumeral\romannumeral
5351   \let\orgalph\@alph
5352   \let\orgAlph\@Alph
5353   \let\orgRoman\@Roman
```

Redefine:

```
5354   \ifglswrallowprimitivemods
5355     \def\the##1{%
5356       \ifx##1\c@page \gls@numberpage\else\orgthe##1\fi}%
5357     \def\number##1{%
5358       \ifx##1\c@page \gls@numberpage\else\orgnumber##1\fi}%
5359     \fi
5360     \def\@arabic##1{%
5361       \ifx##1\c@page \gls@arabicpage\else\orgarabic##1\fi}%
5362     \def\romannumeral##1{%
5363       \ifx##1\c@page \gls@romanpage\else\orgromannumeral##1\fi}%
5364     \def\@Roman##1{%
5365       \ifx##1\c@page \gls@Romanpage\else\orgRoman##1\fi}%
5366     \def\@alph##1{%
5367       \ifx##1\c@page \gls@alphpage\else\orgalph##1\fi}%
5368     \def\@Alph##1{%
5369       \ifx##1\c@page \gls@Alphpage\else\orgAlph##1\fi}%

```

Add hook to allow for other number formats:

```
5370   \@wrglossarynumberhook
```

Prevent expansion:

```
5371   \gls@disablepagerefexpansion
```

Now store location in `\@glslocref`:

```
5372   \protected@xdef\@glslocref{\theHglentrycounter}%
5373   \endgroup
```

Escape any special characters

```
5374   \@gls@checkmkidxchars\@glslocref
```

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

```
5375   \expandafter\ifx\theHglentrycounter\theHglentrycounter\relax
5376   \def\@glo@counterprefix{}%
5377   \else
5378     \protected@edef\@glsHlocref{\theHglentrycounter}%
5379     \@gls@checkmkidxchars\@glsHlocref
```

```

5380 \edef\@do@glS@getcounterprefix{\noexpand\@glS@getcounterprefix
5381 {\@glS@locref}{\@glS@Hlocref}%
5382 }%
5383 \@do@glS@getcounterprefix
5384 \fi

```

De-tok label if required

```

5385 \edef\@glS@label{\glSdetoklabel{#1}}%

```

Write the information to file:

```

5386 \@do@do@wrglossary
5387 }

```

@do@do@wrglossary

```

5388 \newcommand*{\@do@do@wrglossary}{%

```

Determine whether to use xindy or makeindex syntax

```

5389 \ifglSxindy

```

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

```

5390 \expandafter\@glo@check@mkidxrangechar\@glS@numberformat\@nil
5391 \def\@glo@range{}%
5392 \expandafter\if\@glo@prefix(\relax
5393 \def\@glo@range{:open-range}%
5394 \else
5395 \expandafter\if\@glo@prefix)\relax
5396 \def\@glo@range{:close-range}%
5397 \fi
5398 \fi

```

Write to the glossary file using xindy syntax.

```

5399 \glS@glossary{\csname glo@\@glS@label @type\endcsname}{%
5400 (indexentry :tkey (\csname glo@\@glS@label @index\endcsname)

5401 :locref \string"{\@glo@counterprefix}{\@glS@locref}\string" %
5402 :attr \string"\@glS@counter\@glo@suffix\string"
5403 \@glo@range
5404 )
5405 }%
5406 \else

```

Convert the format information into the format required for makeindex

```

5407 \@set@glo@numformat{\@glo@numfmt}{\@glS@counter}{\@glS@numberformat}%
5408 {\@glo@counterprefix}%

```

Write to the glossary file using makeindex syntax.

```

5409 \glS@glossary{\csname glo@\@glS@label @type\endcsname}{%
5410 \string\glossaryentry{\csname glo@\@glS@label @index\endcsname
5411 \@glS@encapchar\@glo@numfmt}{\@glS@locref}}%
5412 \fi
5413 }

```

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.)

```

5414 \newcommand*\@gls@getcounterprefix[2]{%
5415   \edef\@gls@thisloc{#1}\edef\@gls@thisHloc{#2}%
5416   \ifx\@gls@thisloc\@gls@thisHloc
5417     \def\@glo@counterprefix{}%
5418   \else
5419     \def\@gls@get@counterprefix##1.#1##2\end@getprefix{%
5420       \def\@glo@tmp{##2}%
5421       \ifx\@glo@tmp\@empty
5422         \def\@glo@counterprefix{}%
5423       \else
5424         \def\@glo@counterprefix{##1}%
5425       \fi
5426     }%
5427     \@gls@get@counterprefix#2.#1\end@getprefix

```

Warn if no prefix can be formed.

```

5428   \ifx\@glo@counterprefix\@empty
5429     \GlossariesWarning{Hyper target ‘#2’ can’t be formed by
5430       prefixing^^Jlocation ‘#1’. You need to modify the
5431       definition of \string\theH\@gls@counter^^Jotherwise you
5432       will get the warning: “name{\@gls@counter.#1}’ has been^^J
5433       referenced but does not exist"%
5434   \fi
5435 \fi
5436 }

```

1.15 Glossary Entry Cross-References

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

```

5437 \newcommand{\@do@seeglossary}[2]{%
5438   \def\@gls@xref{#2}%
5439   \@onelevel@sanitize\@gls@xref
5440   \@gls@checkmkidxchars\@gls@xref
5441   \ifglxindy
5442     \gls@glossary{\csname glo@#1@type\endcsname}{%
5443       (indexentry
5444         :tkey (\csname glo@#1@index\endcsname)
5445         :xref (\string"\@gls@xref\string")
5446         :attr \string"see\string"
5447       )
5448     }%

```

```

5449 \else
5450   \gls@glossary{\csname glo@#1@type\endcsname}{%
5451     \string\glossaryentry{\csname glo@#1@index\endcsname
5452       \gls@encapchar glsseeformat\@gls@xref}{Z}}}%
5453 \fi
5454 }

```

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

```

5455 \def\@gls@fixbraces#1#2#3\@nil{%
5456   \ifx#2[\relax
5457     \@gls@fixbraces#1#2#3\@end@fixbraces
5458   \else
5459     \def#1{{#2#3}}%
5460   \fi
5461 }

```

`@@gls@fixbraces`

```

5462 \def@@gls@fixbraces#1[#2]#3\@end@fixbraces{%
5463   \def#1{[#2]{#3}}%
5464 }

```

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

```

5465 \DeclareRobustCommand*\glssee[3][\seename]{%
5466   \@do@seeglossary{#2}{#1}{#3}}
5467 \newcommand*\@glssee[3][\seename]{%
5468   \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.

```

5469 \DeclareRobustCommand*\glsseeformat[3][\seename]{%
5470   \emph{#1} \glsseelist{#2}}

```

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

```

5471 \DeclareRobustCommand*\glsseelist[1]{%

```

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

```

5472   \let\@gls@dolast\relax

```

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

```

5473   \let\@gls@donext\relax

```

Iterate through the labels

```

5474   \@for\@gls@thislabel:=#1\do{%

```

Check if on last iteration of loop

```

5475     \ifx\@xfor@nextelement\@nnil
5476       \@gls@dolast
5477     \else
5478       \@gls@donext
5479     \fi

```

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

```
5480 \expandafter\glsseeitem\expandafter{\@gls@thislabel}%
```

Update separators

```
5481 \let\@gls@dolast\glsseelastsep
```

```
5482 \let\@gls@donext\glsseesep
```

```
5483 }%
```

```
5484 }
```

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

```
5485 \newcommand*{\glsseelastsep}{\space\andname\space}
```

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

```
5486 \newcommand*{\glsseesep}{, }
```

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

```
5487 \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, although this is no longer a problem now.)

```
5488 \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.

```
5489 \newcommand*{\gls@save@numberlist}[1]{%
```

```
5490 \ifglssavenumberlist
```

```
5491 \toks@{#1}%
```

```
5492 \edef\@do@writeaux@info{%
```

```
5493 \noexpand\csgdef{glo@\glscurrententrylabel @numberlist}{\the\toks@}%
```

```
5494 }%
```

```
5495 \@onelevel@sanitize\@do@writeaux@info
```

```
5496 \protected@write\@auxout{}{\@do@writeaux@info}%
```

```
5497 \fi
```

```
5498 }
```

`\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.)

```
5499 \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.

```
5500 \ifcsundef{printglossary}{}%
5501 {%
```

If `\printglossary` is already defined, issue a warning and undefine it.

```
5502 \@gls@warnonglossdefined
5503 \undef\printglossary
5504 }
```

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

```
5505 \newcommand*{\printglossary}[1][type=\glsdefaulttype]{%
5506 \@printglossary{#1}{\@print@glossary}%
5507 }
```

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`

```
5508 \newcommand*{\printglossaries}{%
5509 \forallglossaries{\@glo@type}{\printglossary[type=\@glo@type]}%
5510 }
```

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

```
5511 \newcommand*{\printnoidxglossary}[1][type=\glsdefaulttype]{%
5512 \@printglossary{#1}{\@print@noidx@glossary}%
5513 }
```

`noidxglossaries` Analogous to `\printglossaries`

```
5514 \newcommand*{\printnoidxglossaries}{%
5515 \forallglossaries{\@glo@type}{\printnoidxglossary[type=\@glo@type]}%
5516 }
```

`ntgloss@setsort` Initialise to do nothing.

```
5517 \newcommand*{\@printgloss@setsort}{}%
```

`preglossaryhook`

```
5518 \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.

```
5519 \newcommand{\@printglossary}[2]{%
```

Set up defaults.

```
5520 \def\@glo@type{\glsdefaulttype}%
```

```
5521 \def\glossarytitle{\csname @glo@type\@glo@type @title\endcsname}%
```

```
5522 \def\glossarytoctitle{\glossarytitle}%
```

```
5523 \let\org@glossarytitle\glossarytitle
```

```
5524 \def\@glossarystyle{%
```

```
5525 \ifx\@glossary@default@style\relax
```

```
5526 \GlossariesWarning{No default glossary style provided \MessageBreak
```

```
5527 for the glossary ‘\@glo@type’. \MessageBreak
```

```
5528 Using deprecated fallback. \MessageBreak
```

```
5529 To fix this set the style with \MessageBreak
```

```
5530 \string\setglossarystyle\space or use the \MessageBreak
```

```
5531 style key=value option}%
```

```
5532 \fi
```

```
5533 }%
```

```
5534 \def\gls@dotoc@title{\glssettoc@title{\@glo@type}}%
```

Store current value of `\glossaryentrynumbers`. (This may be changed via the optional argument)

```
5535 \let\@org@glossaryentrynumbers\glossaryentrynumbers
```

Localise the effects of the optional argument

```
5536 \bgroup
```

Activate or deactivate sort key:

```
5537 \@printgloss@setsort
```

Determine settings specified in the optional argument.

```
5538 \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)

```
5539 \ifx\glossarytitle\org@glossarytitle
```

```
5540 \else
```

```
5541 \expandafter\let\csname @glo@type\@glo@type @title\endcsname
```

```
5542 \glossarytitle
```

```
5543 \fi
```

Allow a high-level user command to indicate the current glossary

```
5544 \let\currentglossary\@glo@type
```

Enable individual number lists to be suppressed.

```
5545 \let\org@glossaryentrynumbers\glossaryentrynumbers
```

```
5546 \let\glsnonextpages\glsnonextpages
```

Enable individual number list to be activated:

```
5547 \let\glsnextpages\@glsnextpages
```

Enable suppression of description terminators.

```
5548 \let\nopostdesc\@nopostdesc
```

Set up the entry for the TOC

```
5549 \gls@dotocitle
```

Set the glossary style

```
5550 \@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):

```
5551 \let\gls@org@glossaryentryfield\glossentry
5552 \let\gls@org@glossarysubentryfield\subglossentry
5553 \renewcommand{\glossentry}[1]{%
5554   \xdef\glscurrententrylabel{\glsdetoklabel{##1}}%
5555   \gls@org@glossaryentryfield{##1}%
5556 }%
5557 \renewcommand{\subglossentry}[2]{%
5558   \xdef\glscurrententrylabel{\glsdetoklabel{##2}}%
5559   \gls@org@glossarysubentryfield{##1}{##2}%
5560 }%
```

```
5561 \@gls@preglossaryhook
```

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

```
5562 #2%
```

End the current scope

```
5563 \egroup
```

Reset `\glossaryentrynumbers`

```
5564 \global\let\glossaryentrynumbers\@org@glossaryentrynumbers
```

Suppress warning about no `\printglossary`

```
5565 \global\let\warn@noprintglossary\relax
5566 }
```

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

```
5567 \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.)

```
5568 \makeatletter
```

Input the glossary file, if it exists.

```
5569 \@input@{\jobname.\csname @glo@type\@glo@type @in\endcsname}%
```

If the glossary file doesn't exist, do \null. (This ensures that the page is shipped out and all write commands are done.) This might produce an empty page, but at this point the document isn't complete, so it shouldn't matter.

```
5570 \IfFileExists{\jobname.\csname @glo@type @in\endcsname}%
5571 {}%
5572 {\null}%
```

If xindy is being used, need to write the language dependent information to the .aux file for makeglossaries.

```
5573 \ifglxindy
5574 \ifcsundef{@xdy@\@glo@type @language}%
5575 {%
5576 \edef\@do@auxoutstuff{%
5577 \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.

```
5578 \noexpand\immediate\noexpand\write\@auxout{%
5579 \string\providecommand\string\@xdylanguage[2]{}}%
5580 \noexpand\immediate\noexpand\write\@auxout{%
5581 \string\@xdylanguage{\@glo@type}{\@xdy@main@language}}%
5582 }%
5583 }%
5584 }%
5585 {%
5586 \edef\@do@auxoutstuff{%
5587 \noexpand\AtEndDocument{%
5588 \noexpand\immediate\noexpand\write\@auxout{%
5589 \string\providecommand\string\@xdylanguage[2]{}}%
5590 \noexpand\immediate\noexpand\write\@auxout{%
5591 \string\@xdylanguage{\@glo@type}{\csname @xdy@\@glo@type
5592 @language\endcsname}}%
5593 }%
5594 }%
5595 }%
5596 \@do@auxoutstuff
5597 \edef\@do@auxoutstuff{%
5598 \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.

```
5599 \noexpand\immediate\noexpand\write\@auxout{%
5600 \string\providecommand\string\@gls@codepage[2]{}}%
5601 \noexpand\immediate\noexpand\write\@auxout{%
5602 \string\@gls@codepage{\@glo@type}{\@gls@codepage}}%
5603 }%
5604 }%
5605 \@do@auxoutstuff
5606 \fi
```

Activate warning if \makeglossaries hasn't been used.

```

5607 \renewcommand*{\@warn@nomakeglossaries}{%
5608   \GlossariesWarningNoLine{\string\makeglossaries\space
5609     hasn't been used,^^Jthe glossaries will not be updated}%
5610 }%
5611 }

```

The sort macros all have the syntax:

$\backslash@glo@sortmacro@<order>\{<type>\}$

where $<order>$ is the sort order as specified by the sort key and $<type>$ is the glossary type. (The referenced entry list is stored in $\backslash@gl{sref@<type>}$. The actual sorting is done by $\backslash@glo@sortentries\{<handler>\}\{<type>\}$.

$glo@sortentries$

```

5612 \newcommand*{\@glo@sortentries}[2]{%
5613   \glosortentrieswarning
5614   \def\@glo@sortinglist{}%
5615   \def\@glo@sortinghandler{#1}%
5616   \edef\@glo@type{#2}%
5617   \forlistcsloop{\@glo@do@sortentries}\@gl{sref@#2}%
5618   \csdef{\@gl{sref@#2}}{}%
5619   \@for\@this@label:=\@glo@sortinglist\do{%

```

Has this entry already been added?

```

5620   \xifinlistcs{\@this@label}\@gl{sref@#2}%
5621   {}%
5622   {%
5623     \listcsxadd{\@gl{sref@#2}}{\@this@label}%
5624     }%
5625     \ifcsdef{\@glo@sortingchildren@\@this@label}%
5626     {%
5627       \@glo@addchildren{#2}\@this@label}%
5628       }%
5629       {}%
5630     }%
5631 }

```

$@glo@addchildren$

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

```

5632 \newcommand*{\@glo@addchildren}[2]{%

```

Scope to allow nesting.

```

5633   \bgroup
5634   \letcs{\@glo@childlist}\@glo@sortingchildren@#2}%
5635   \@for\@this@childlabel:=\@glo@childlist\do
5636   {%

```

Check this label hasn't already been added.

```

5637      \xifinlistcs{\@this@childlabel}{@glsref@#1}%
5638      {%
5639      {%
5640      \listcsxadd{@glsref@#1}{\@this@childlabel}%
5641      }%

```

Does this child have children?

```

5642      \ifcsdef{@glo@sortingchildren@\@this@childlabel}%
5643      {%
5644      \@glo@addchildren{#1}{\@this@childlabel}%
5645      }%
5646      {%
5647      }%
5648      }%
5649 \egroup
5650 }

```

@do@sortentries

```

5651 \newcommand*{\@glo@do@sortentries}[1]{%
5652 \ifglshasparent{#1}%
5653 {%

```

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

```

5654 \edef\@glo@parent{\csuse{glo@glstetoklabel{#1}@parent}}%
5655 \ifcsundef{@glo@sortingchildren@\@glo@parent}%
5656 {%
5657 \csdef{@glo@sortingchildren@\@glo@parent}{}%
5658 }%
5659 }%
5660 \expandafter\@glo@sortedinsert
5661 \csname @glo@sortingchildren@\@glo@parent\endcsname{#1}%

```

Has the parent been added?

```

5662 \xifinlistcs{\@glo@parent}{@glsref@\@glo@type}%
5663 {%

```

Yes, it has so do nothing.

```

5664 }%
5665 {%

```

No, it hasn't so add it now.

```

5666 \expandafter\@glo@do@sortentries\expandafter{\@glo@parent}%
5667 }%
5668 }%
5669 {%
5670 \@glo@sortedinsert{\@glo@sortinglist}{#1}%
5671 }%
5672 }

```

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

Insert into list.

```
5673 \newcommand*{\@glo@sortedinsert}[2]{%
5674   \dtl@insertinto{#2}{#1}{\@glo@sortinghandler}%
5675 }%
```

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

```
5676 \newcommand*{\@glo@sorthandler@word}[2]{%
5677   \letcs\@gls@sort@A{glo@glstetoklabel{#1}@sort}%
5678   \letcs\@gls@sort@B{glo@glstetoklabel{#2}@sort}%
5679   \edef\glo@do@compare{%
5680     \noexpand\dtlwordindexcompare{\noexpand\dtl@sortresult}%
5681     {\expandonce\@gls@sort@B}%
5682     {\expandonce\@gls@sort@A}%
5683   }%
5684   \glo@do@compare
5685 }
```

thandler@letter

```
5686 \newcommand*{\@glo@sorthandler@letter}[2]{%
5687   \letcs\@gls@sort@A{glo@glstetoklabel{#1}@sort}%
5688   \letcs\@gls@sort@B{glo@glstetoklabel{#2}@sort}%
5689   \edef\glo@do@compare{%
5690     \noexpand\dtlletterindexcompare{\noexpand\dtl@sortresult}%
5691     {\expandonce\@gls@sort@B}%
5692     {\expandonce\@gls@sort@A}%
5693   }%
5694   \glo@do@compare
5695 }
```

orthandler@case Case-sensitive sort.

```
5696 \newcommand*{\@glo@sorthandler@case}[2]{%
5697   \letcs\@gls@sort@A{glo@glstetoklabel{#1}@sort}%
5698   \letcs\@gls@sort@B{glo@glstetoklabel{#2}@sort}%
5699   \edef\glo@do@compare{%
5700     \noexpand\dtlcompare{\noexpand\dtl@sortresult}%
5701     {\expandonce\@gls@sort@B}%
5702     {\expandonce\@gls@sort@A}%
5703   }%
5704   \glo@do@compare
5705 }
```

thandler@nocase Case-insensitive sort.

```

5706 \newcommand*{\@glo@sorthandler@nocase}[2]{%
5707   \letcs\@gls@sort@A{glo\glsdetoklabel{#1}@sort}%
5708   \letcs\@gls@sort@B{glo\glsdetoklabel{#2}@sort}%
5709   \edef\glo@do@compare{%
5710     \noexpand\dtlicompare{\noexpand\dtl@sortresult}%
5711     {\expandonce\@gls@sort@B}%
5712     {\expandonce\@gls@sort@A}%
5713   }%
5714   \glo@do@compare
5715 }

```

@sortmacro@word Sort macro for ‘word’

```

5716 \newcommand*{\@glo@sortmacro@word}[1]{%
5717   \ifdefstring{\@glo@default@sorttype}{standard}%
5718   {%
5719     \@glo@sortentries{\@glo@sorthandler@word}{#1}%
5720   }%
5721   {%
5722     \PackageError{glossaries}{Conflicting sort options:^^J
5723       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5724       \string\printnoidxglossary[sort=word]}{}%
5725   }%
5726 }

```

ortmacro@letter Sort macro for ‘letter’

```

5727 \newcommand*{\@glo@sortmacro@letter}[1]{%
5728   \ifdefstring{\@glo@default@sorttype}{standard}%
5729   {%
5730     \@glo@sortentries{\@glo@sorthandler@letter}{#1}%
5731   }%
5732   {%
5733     \PackageError{glossaries}{Conflicting sort options:^^J
5734       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5735       \string\printnoidxglossary[sort=letter]}{}%
5736   }%
5737 }

```

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

```

5738 \newcommand*{\@glo@sortmacro@standard}[1]{%
5739   \ifdefstring{\@glo@default@sorttype}{standard}%
5740   {%
5741     \ifcsdef{\@glo@sorthandler@\glsorder}%
5742     {%
5743       \@glo@sortentries{\csuse{\@glo@sorthandler@\glsorder}}{#1}%
5744     }%
5745     {%
5746       \PackageError{glossaries}{Unknown sort handler ‘\glsorder’}{}%
5747     }%
5748   }%

```

```

5749  {%
5750    \PackageError{glossaries}{Conflicting sort options:^^J
5751      \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5752      \string\printnoidxglossary[sort=standard]}{ }%
5753  }%
5754 }

```

@sortmacro@case Sort macro for ‘case’

```

5755 \newcommand*{\@glo@sortmacro@case}[1]{%
5756   \ifdefstring{\@glo@default@sorttype}{standard}%
5757   {%
5758     \@glo@sortentries{\@glo@sorthandler@case}{#1}%
5759   }%
5760   {%
5761     \PackageError{glossaries}{Conflicting sort options:^^J
5762       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5763       \string\printnoidxglossary[sort=case]}{ }%
5764   }%
5765 }

```

@sortmacro@nocase Sort macro for ‘nocase’

```

5766 \newcommand*{\@glo@sortmacro@nocase}[1]{%
5767   \ifdefstring{\@glo@default@sorttype}{standard}%
5768   {%
5769     \@glo@sortentries{\@glo@sorthandler@nocase}{#1}%
5770   }%
5771   {%
5772     \PackageError{glossaries}{Conflicting sort options:^^J
5773       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5774       \string\printnoidxglossary[sort=nocase]}{ }%
5775   }%
5776 }

```

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

```

5777 \newcommand*{\@glo@sortmacro@def}[1]{%
5778   \def\@glo@sortinglist{%
5779     \for@gl@sentries[#1]{\@gls@thislabel}%
5780     {%
5781       \xifinlistcs{\@gls@thislabel}{\@gls@ref@#1}%
5782     }%
5783     \list@add{\@glo@sortinglist}{\@gls@thislabel}%
5784   }%
5785   {%

```

Hasn't been referenced.

```

5786   }%
5787 }%
5788 \cslet{\@gls@ref@#1}{\@glo@sortinglist}%
5789 }

```


ortmacro@def@do This won't include parent entries that haven't been referenced.

```
5790 \newcommand*{\@glo@sortmacro@def@do}[1]{%
5791   \ifinlistcs{#1}{\@glsref@\@glo@type}%
5792   {%
5793     {%
5794       \listcsadd{\@glsref@\@glo@type}{#1}%
5795     }%
5796     \ifcsdef{\@glo@sortingchildren@#1}%
5797     {%
5798       \@glo@addchildren{\@glo@type}{#1}%
5799     }%
5800   }%
5801 }
```

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

```
5802 \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.

```
5803 \newcommand*{\@print@noidx@glossary}{%
5804   \ifcsdef{\@glsref@\@glo@type}%
5805   {%
```

Sort the entries:

```
5806   \ifcsdef{\@glo@sortmacro@\@glo@sorttype}%
5807   {%
5808     \csuse{\@glo@sortmacro@\@glo@sorttype}{\@glo@type}%
5809   }%
5810   {%
5811     \PackageError{glossaries}{Unknown sort handler '\@glo@sorttype'}{ }%
5812   }%
```

Do the glossary heading and preamble

```
5813   \glossarysection[\glossarytoctitle]{\glossarytitle}%
5814   \glossarypreamble
```

The glossary style might use a tabular-like environment, which may cause scoping problems when setting the current letter group. The predefined tabular-like styles don't support letter group headings, but there's nothing to stop the user from defining their own custom style that might, so any redefinition of this command within theglossary will have to be done globally.

```
5815   \def\@gls@currentlettergroup{%
5816     \begin{theglossary}%
5817     \glossaryheader
5818     \glsresetentrylist
```

Iterate through the entries.

```
5819 \forlistcsloop{\@gls@noidx@do}{\@glsref@{\@glo@type}}%
```

Finally end the glossary and do the postamble:

```
5820 \end{theglossary}%  
5821 \glossarypostamble  
5822 }%  
5823 {%  
5824 \@gls@noref@warn{\@glo@type}%  
5825 }%  
5826 }
```

\glo@grabfirst

```
5827 \def\glo@grabfirst#1#2\@nil{%  
5828 \def\@gls@firsttok{#1}%  
5829 \ifdefempty\@gls@firsttok  
5830 {%  
5831 \def\@glo@thislettergrp{0}%  
5832 }%  
5833 {%
```

Sanitize it:

```
5834 \@onelevel@sanitize\@gls@firsttok
```

Fetch the first letter:

```
5835 \expandafter\@glo@grabfirst\@gls@firsttok{}{}\@nil  
5836 }%  
5837 }
```

\@glo@grabfirst

```
5838 \def\@glo@grabfirst#1#2\@nil{%  
5839 \ifdefempty\@glo@thislettergrp  
5840 {%  
5841 \def\@glo@thislettergrp{glssymbols}%  
5842 }%  
5843 {%  
5844 \count@=\uccode'#1\relax  
5845 \ifnum\count@=0\relax  
5846 \def\@glo@thislettergrp{glssymbols}%  
5847 \else  
5848 \ifdefstring\@glo@sorttype{case}%  
5849 {%  
5850 \count@='#1\relax  
5851 }%  
5852 {%  
5853 }%  
5854 \edef\@glo@thislettergrp{\the\count@}%  
5855 \fi  
5856 }%  
5857 }
```

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

```
5858 \newcommand{\@gls@noidx@do}[1]{%
```

Get this entry's location list

```
5859 \global\letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
```

Does this entry have a parent?

```
5860 \ifglshasparent{#1}%
```

```
5861 {%
```

Has a parent.

```
5862 \gls@level=\csuse{glo@\glsdetoklabel{#1}@level}\relax
```

```
5863 \ifdefvoid{\@gls@loclist}
```

```
5864 {%
```

```
5865 \subglossentry{\gls@level}{#1}{}%
```

```
5866 }%
```

```
5867 {%
```

```
5868 \subglossentry{\gls@level}{#1}%
```

```
5869 {%
```

```
5870 \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
```

```
5871 }%
```

```
5872 }%
```

```
5873 }%
```

```
5874 {%
```

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

```
5875 \letcs{\@gls@sort}{glo@\glsdetoklabel{#1}@sort}%
```

Fetch the first letter:

```
5876 \expandafter\glo@grabfirst\@gls@sort{}{}\@nil
```

```
5877 \ifdefequal{\@glo@thislettergrp}{\@gls@currentlettergroup}%
```

```
5878 {}%
```

```
5879 {%
```

Do the group header:

```
5880 \ifdefempty{\@gls@currentlettergroup}{}%
```

```
5881 {%
```

The group skip may start a new scope, so make a global assignment.

```
5882 \global\let\@glo@thislettergrp\@glo@thislettergrp
```

```
5883 \glsgroupskip
```

```
5884 }%
```

```
5885 \glsgroupheading{\@glo@thislettergrp}%
```

```
5886 }%
```

```
5887 \global\let\@gls@currentlettergroup\@glo@thislettergrp
```

Do this entry:

```
5888 \ifdefvoid{\@gls@loclist}
```

```
5889 {%
```

```
5890 \glossentry{#1}{}%
```

```

5891 }%
5892 {%
5893   \glossentry{#1}%
5894   {%
5895     \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
5896   }%
5897 }%
5898 }%
5899 }

```

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

Display location list.

```

5900 \newcommand*{\glsnoidxloclist}[1]{%
5901   \def\@gls@noidxloclist@sep{}%
5902   \def\@gls@noidxloclist@prev{}%
5903   \forlistloop{\glsnoidxloclisthandler}{#1}%
5904 }

```

`loclisthandler` Handler for location list iterator.

```

5905 \newcommand*{\glsnoidxloclisthandler}[1]{%
5906   \ifdefstring{\@gls@noidxloclist@prev}{#1}%
5907   {%

```

Same as previous location so skip.

```

5908 }%
5909 {%
5910   \@gls@noidxloclist@sep
5911   #1%
5912   \def\@gls@noidxloclist@sep{\delimN}%
5913   \def\@gls@noidxloclist@prev{#1}%
5914 }%
5915 }

```

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

```

5916 \newcommand*{\glsnoidxdisplayloclisthandler}[1]{%
5917   \ifdefstring{\@gls@noidxloclist@prev}{#1}%
5918   {%

```

Same as previous location so skip.

```

5919 }%
5920 {%
5921   \@gls@noidxloclist@sep
5922   \@gls@noidxloclist@prev
5923   \def\@gls@noidxloclist@prev{#1}%
5924 }%
5925 }

```

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

Display a location in the location list.

```
5926 \newcommand*\glsnoidxdisplayloc[4]{%
5927   \setentrycounter[#1]{#2}%
5928   \csuse{#3}{#4}%
5929 }
```

`\@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.

```
5930 \newcommand*\@gls@reference}[3]{%
```

Add to label list

```
5931   \glsdoifexistsorwarn{#2}%
5932   {%
5933     \ifcsundef{@glsref@#1}{\csgdef{@glsref@#1}{}}{}%
5934     \ifinlistcs{#2}{@glsref@#1}%
5935     {}%
5936     {\listcsgadd{@glsref@#1}{#2}}%
5937 }
```

Add to location list

```
5937   \ifcsundef{glo@\glsdetoklabel{#2}@loclist}%
5938   {\csgdef{glo@\glsdetoklabel{#2}@loclist}{}}{}%
5939   {}%
5940   \listcsgadd{glo@\glsdetoklabel{#2}@loclist}{#3}%
5941   }%
5942 }
```

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

```
5943 \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`.

```
5944 \define@key{printgloss}{title}{%
5945   \def\glossarytitle{#1}%
5946   \let\gls@dotoc@title\relax
5947 }
```

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

```
5948 \define@key{printgloss}{toctitle}{%
5949   \def\glossarytoctitle{#1}%
5950   \let\gls@dotoc@title\relax
5951 }
```

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

```

5952 \define@key{printgloss}{style}{%
5953   \ifcsundef{@glsstyle@#1}%
5954   {%
5955     \PackageError{glossaries}%
5956     {Glossary style ‘#1’ undefined}{}%
5957   }%
5958   {%
5959     \def\@glossarystyle{\setglossentrycompatibility
5960       \csname @glsstyle@#1\endcsname}%
5961   }%
5962 }

```

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

```

5963 \define@choicekey{printgloss}{numberedsection}[\val\nr]{%
5964 false,nolabel,autolabel,nameref}[nolabel]{%
5965   \ifcase\nr\relax
5966     \renewcommand*{\@glossarysecstar}{*}%
5967     \renewcommand*{\@glossaryseclabel}{}%
5968   \or
5969     \renewcommand*{\@glossarysecstar}{}%
5970     \renewcommand*{\@glossaryseclabel}{}%
5971   \or
5972     \renewcommand*{\@glossarysecstar}{}%
5973     \renewcommand*{\@glossaryseclabel}{\label{\glsautoprefix\@glo@type}}%
5974   \or
5975     \renewcommand*{\@glossarysecstar}{*}%
5976     \renewcommand*{\@glossaryseclabel}{%
5977       \protected@edef\@currentlabelname{\glossarytoctitle}%
5978       \label{\glsautoprefix\@glo@type}}%
5979   \fi
5980 }

```

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

```

5981 \define@choicekey{printgloss}{nogroupskip}{true,false}[true]{%
5982   \csuse{glsnogroupskip#1}%
5983 }

```

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

```

5984 \define@choicekey{printgloss}{nopostdot}{true,false}[true]{%
5985   \csuse{glsnopostdot#1}%
5986 }

```

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

```

5987 \define@choicekey{printgloss}{entrycounter}{true,false}[true]{%
5988   \csuse{glsentrycounter#1}%
5989   \ifglsentrycounter
5990     \ifx\@gls@counterwithin\@empty
5991       \newcounter{glossaryentry}%
5992     \else

```

```

5993     \newcounter{glossaryentry}[\@gls@counterwithin]%
5994 \fi
5995 \def\theHglossaryentry{\currentglossary.\theglossaryentry}%
5996 \renewcommand*{\glsresetentrycounter}{%
5997     \setcounter{glossaryentry}{0}%
5998 }%
5999 \renewcommand*{\glsstepentry}[1]{%
6000     \refstepcounter{glossaryentry}%
6001     \label{glsentry-\glsdetoklabel{##1}}%
6002 }%
6003 \renewcommand*{\glsentrycounterlabel}{\theglossaryentry.\space}%
6004 \renewcommand*{\glsentryitem}[1]{%
6005     \glsstepentry{##1}\glsentrycounterlabel
6006 }%
6007 \else
6008     \renewcommand*{\glsresetentrycounter}{}%
6009     \renewcommand*{\glsstepentry}[1]{}%
6010     \renewcommand*{\glsentrycounterlabel}{}%
6011     \renewcommand*{\glsentryitem}[1]{\glsresetsubentrycounter}
6012 \fi
6013 }

```

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.

```

6014 \define@choicekey{printgloss}{subentrycounter}{true,false}[true]{%
6015     \csuse{glssubentrycounter#1}%
6016     \ifglssubentrycounter
6017         \ifundef\c@glossarysubentry
6018         {%
6019             \ifglssentrycounter
6020                 \newcounter{glossarysubentry}[glossaryentry]%
6021             \else
6022                 \newcounter{glossarysubentry}
6023             \fi
6024         }{}%
6025     \renewcommand*{\glsstepsubentry}[1]{%
6026         \edef\currentglssubentry{\glsdetoklabel{##1}}%
6027         \refstepcounter{glossarysubentry}%
6028         \label{glsentry-\currentglssubentry}%
6029     }%
6030     \renewcommand*{\glsresetsubentrycounter}{%
6031         \setcounter{glossarysubentry}{0}%
6032     }%
6033     \renewcommand*{\glssubentryitem}[1]{%
6034         \glsstepsubentry{##1}\glssubentrycounterlabel
6035     }%
6036     \renewcommand*{\glssubentrycounterlabel}{\theglossarysubentry.\space}%
6037     \def\theHglossarysubentry{\currentglssubentry.\theglossarysubentry}
6038 \else

```

```

6039 \renewcommand*{\glssubentryitem}[1]{}%
6040 \renewcommand*{\glsstepsubentry}[1]{}%
6041 \renewcommand*{\glsresetsubentrycounter}{}%
6042 \renewcommand*{\glssubentrycounterlabel}{}%
6043 \fi
6044 }

```

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

```

6045 \define@boolkey{printgloss}[gls]{nonnumberlist}[true]{%
6046 \ifglsnonnumberlist
6047 \def\glossaryentrynumbers##1{}%
6048 \else
6049 \def\glossaryentrynumbers##1{##1}%
6050 \fi}

```

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

```

6051 \define@key{printgloss}{sort}{\@glo@assign@sortkey{#1}}

```

@assign@sortkey Issue error if used with \printglossary

```

6052 \newcommand*{\@glo@no@assign@sortkey}[1]{%
6053 \PackageError{glossaries}{'sort' key not permitted with
6054 \string\printglossary}%
6055 {The 'sort' key may only be used with \string\printnoidxglossary}%
6056 }

```

@assign@sortkey For use with \printnoidxglossary

```

6057 \newcommand*{\@glo@assign@sortkey}[1]{%
6058 \def\@glo@sorttype{#1}%
6059 }

```

@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 place in the entry's description and 3 column tabular style glossary is used.) \org@glossaryentrynumbers needs to be set at the start of each glossary, in the event that \glossaryentrynumber is re-defined.

```

6060 \newcommand*{\@glsnonextpages}{%
6061 \gdef\glossaryentrynumbers##1{%
6062 \glsresetentrylist
6063 }%
6064 }

```

\@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 place in the entry's description and 3 column tabular style glossary is used.) \org@glossaryentrynumbers needs to be set at the start of each glossary, in the event that \glossaryentrynumber is re-defined.

```

6065 \newcommand*{\@glsnextpages}{%
6066 \gdef\glossaryentrynumbers##1{%
6067 ##1\glsresetentrylist}}

```


sresetentrylist Resets `\glossaryentrynumbers`

```

6068 \newcommand*{\glsresetentrylist}{%
6069   \global\let\glossaryentrynumbers\org@glossaryentrynumbers}

```

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

```

6070 \newcommand*{\glsnonextpages}{}

```

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

```

6071 \newcommand*{\glsnextpages}{}

```

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

```

6072 \ifglssentrycounter
6073   \ifx\@gls@counterwithin\@empty
6074     \newcounter{glossaryentry}
6075   \else
6076     \newcounter{glossaryentry}[\@gls@counterwithin]
6077   \fi
6078   \def\theHglossaryentry{\currentglossary.\theglossaryentry}
6079 \fi

```

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

```

6080 \ifglsssubentrycounter
6081   \ifglssentrycounter
6082     \newcounter{glossarysubentry}[glossaryentry]
6083   \else
6084     \newcounter{glossarysubentry}
6085   \fi
6086   \def\theHglossarysubentry{\currentglsssubentry.\theglossarysubentry}
6087 \fi

```

subentrycounter Resets the `glossarysubentry` counter.

```

6088 \ifglsssubentrycounter
6089   \newcommand*{\glsresetsubentrycounter}{%
6090     \setcounter{glossarysubentry}{0}%
6091   }
6092 \else
6093   \newcommand*{\glsresetsubentrycounter}{}
6094 \fi

```

subentrycounter Resets the `glossareentry` counter.

```

6095 \ifglssentrycounter
6096   \newcommand*{\glsresetentrycounter}{%
6097     \setcounter{glossaryentry}{0}%
6098   }
6099 \else
6100   \newcommand*{\glsresetentrycounter}{}
6101 \fi

```

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

```

6102 \ifglsentrycounter
6103   \newcommand*{\glsstepentry}[1]{%
6104     \refstepcounter{glossaryentry}%
6105     \label{glsentry-\glsdetoklabel{#1}}%
6106   }
6107 \else
6108   \newcommand*{\glsstepentry}[1]{%
6109 \fi

```

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

```

6110 \ifglssubentrycounter
6111   \newcommand*{\glsstepsubentry}[1]{%
6112     \edef\currentglssubentry{\glsdetoklabel{#1}}%
6113     \refstepcounter{glossarysubentry}%
6114     \label{glsentry-\currentglssubentry}%
6115   }
6116 \else
6117   \newcommand*{\glsstepsubentry}[1]{%
6118 \fi

```

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

```

6119 \ifglsentrycounter
6120   \newcommand*{\glsrefentry}[1]{\ref{glsentry-\glsdetoklabel{#1}}}
6121 \else
6122   \ifglssubentrycounter
6123     \newcommand*{\glsrefentry}[1]{\ref{glsentry-\glsdetoklabel{#1}}}
6124   \else
6125     \newcommand*{\glsrefentry}[1]{\gls{#1}}
6126   \fi
6127 \fi

```

`trycounterlabel` Defines how to display the glossaryentry counter.

```

6128 \ifglsentrycounter
6129   \newcommand*{\glsentrycounterlabel}{\theglossaryentry.\space}
6130 \else
6131   \newcommand*{\glsentrycounterlabel}{}
6132 \fi

```

`trycounterlabel` Defines how to display the glossarysubentry counter.

```

6133 \ifglssubentrycounter
6134   \newcommand*{\glssubentrycounterlabel}{\theglossarysubentry}\space}
6135 \else
6136   \newcommand*{\glssubentrycounterlabel}{}
6137 \fi

```

`\glsentryitem` Step and display glossaryentry counter, if appropriate.

```
6138 \ifglssentrycounter
6139   \newcommand*{\glsentryitem}[1]{%
6140     \glstepentry{#1}\glsentrycounterlabel
6141   }
6142 \else
6143   \newcommand*{\glsentryitem}[1]{\glsresetsubentrycounter}
6144 \fi
```

`glssubentryitem` Step and display glossarysubentry counter, if appropriate.

```
6145 \ifglssubentrycounter
6146   \newcommand*{\glssubentryitem}[1]{%
6147     \glstepsubentry{#1}\glssubentrycounterlabel
6148   }
6149 \else
6150   \newcommand*{\glssubentryitem}[1]{}
6151 \fi
```

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

```
6152 \ifcsundef{theglossary}%
6153 {%
6154   \newenvironment{theglossary}{}{}%
6155 }%
6156 {%
6157   \@gls@warnontheGLOSSdefined
6158   \renewenvironment{theglossary}{}{}%
6159 }
```

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`

```
6160 \newcommand*{\glossaryheader}{}%
```

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

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

```
6161 \newcommand*{\glstarget}[2]{\@glstarget{\glolinkprefix#1}{#2}}
```

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

atibleglossentry

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

```
6162 \providecommand*{\compatibleglossentry}[2]{%
6163   \toks@{#2}%
6164   \protected@edef\@do@glossentry{\noexpand\glossaryentryfield{#1}%
6165     {\noexpand\glsnamefont
6166       {\expandafter\expandonce\csname glo@#1@name\endcsname}}}%
6167     {\expandafter\expandonce\csname glo@#1@desc\endcsname}%
6168     {\expandafter\expandonce\csname glo@#1@symbol\endcsname}%
6169     {\the\toks@}}%
6170   }%
6171   \@do@glossentry
6172 }
```

\glossentryname

```
6173 \newcommand*{\glossentryname}[1]{%
6174   \glsdoifexistsorwarn{#1}%
6175   {%
6176     \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
6177     \expandafter\glsnamefont\expandafter{\glo@name}%
6178   }%
6179 }
```

\Glossentryname

```
6180 \newcommand*{\Glossentryname}[1]{%
6181   \glsdoifexistsorwarn{#1}%
6182   {%
6183     \glsnamefont{\Glsentryname{#1}}%
6184   }%
6185 }
```

\glossentrydesc

```
6186 \newcommand*{\glossentrydesc}[1]{%
6187   \glsdoifexistsorwarn{#1}%
6188   {%
6189     \glsentrydesc{#1}%
6190   }%
6191 }
```

\Glossentrydesc

```
6192 \newcommand*{\Glossentrydesc}[1]{%
6193   \glsdoifexistsorwarn{#1}%
6194   {%
6195     \Glsentrydesc{#1}%
6196   }%
6197 }
```

lossentrysymbol

```

6198 \newcommand*{\glossentrysymbol}[1]{%
6199   \glsdoifexistsorwarn{#1}%
6200   {%
6201     \glsentrysymbol{#1}%
6202   }%
6203 }

```

lossentrysymbol

```

6204 \newcommand*{\Glossentrysymbol}[1]{%
6205   \glsdoifexistsorwarn{#1}%
6206   {%
6207     \Glsentrysymbol{#1}%
6208   }%
6209 }

```

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

```

6210 \providecommand*{\compatiblesubglossentry}[3]{%
6211   \toks@{#3}%
6212   \protected@edef\@do@subglossentry{\noexpand\glossarysubentryfield{\number#1}%
6213     {#2}%
6214     {\noexpand\glsnamefont
6215       {\expandafter\expandonce\csname glo@#2@name\endcsname}}}%
6216     {\expandafter\expandonce\csname glo@#2@desc\endcsname}%
6217     {\expandafter\expandonce\csname glo@#2@symbol\endcsname}%
6218     {\the\toks@}}%
6219   }%
6220   \@do@subglossentry
6221 }

```

rycompatibility

```

6222 \newcommand*{\setglossentrycompatibility}{%
6223   \let\glossentry\compatibleglossentry
6224   \let\subglossentry\compatiblesubglossentry
6225 }
6226 \setglossentrycompatibility

```

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

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

```

6227 \newcommand{\glossaryentryfield}[5]{%
6228   \GlossariesWarning
6229   {Deprecated use of \string\glossaryentryfield.^^J
6230     I recommend you change to \string\glossentry.^^J

```

```

6231   If you've just upgraded, try removing your gls auxiliary
6232   files^^J and recompile}%
6233   \noindent\textbf{\glstarget{#1}{#2}} #4 #3. #5\par}

```

arysubentryfield

```

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

```

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

```

6234 \newcommand*{\glossarysubentryfield}[6]{%
6235   \GlossariesWarning
6236   {Deprecated use of \string\glossarysubentryfield.^^J
6237   I recommend you change to \string\subglossentry.^^J
6238   If you've just upgraded, try removing your gls auxiliary
6239   files^^J and recompile}%
6240   \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

```

6241 \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

```

6242 \newcommand*{\glsgroupheading}[1]{}

```

It is possible to “trick” `makeindex` into treating entries as though they belong to the same group, even if the terms don’t start with the same letter, by modifying the sort key. For example, all entries belonging to one group could be defined so that the sort key starts with an a, while entries belonging to another group could be defined so that the sort key starts with a b, and so on. If you want each group to have a heading, you would then need to modify the translation control sequences `\glsgetgrouptitle` and `\glsgetgrouplabel` so that the label is translated into the required title (and vice-versa).

`\glsgetgrouptitle{<label>}`

This command produces the title for the glossary group whose label is given by *<label>*. By default, the group labelled `glssymbols` produces `\glssymbolsgroupname`, the group labelled `glsnumbers` produces `\glsnumbersgroupname` and all the other groups simply produce their label. As mentioned above, the group labels are: `glssymbols`, `glsnumbers`, `A`, ..., `Z`. If you want to redefine the group titles, you will need to redefine this command. Languages other than English may produce labels that are non-expandable, so we need to check for that otherwise it will create a “missing `\endcsname` inserted” error.

`lsgetgrouptitle`

```
6243 \newcommand*{\glsgetgrouptitle}[1]{%
6244   \@gls@getgrouptitle{#1}{\@gls@grptitle}%
6245   \@gls@grptitle
6246 }
```

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

```
6247 \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.
6248   \dtl@ifsingle{#1}%
6249   {%
6250     \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
6251   }%
6252   {%
6253     \ifboolexpr{test{\ifstrequal{#1}{glssymbols}}
6254               or test{\ifstrequal{#1}{glsnumbers}}}%
6255     {%
6256       \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
6257     }%
6258     {%
6259       \def#2{#1}%
6260     }%
6261   }%
6262 }
```

`x@getgrouptitle` Version for the no-indexing app option:

```
6263 \newcommand*{\@gls@noidx@getgrouptitle}[2]{%
6264   \DTLifint{#1}%
6265   {\edef#2{\char#1\relax}}%
6266   {%
6267     \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
6268   }%
6269 }
```

`\glsgetgrouplabel{<title>}`

This command does the reverse to the previous command. The argument is the group title, and it produces the group label. Note that if you redefine `\glsgetgrouptitle`, you will also need to redefine `\glsgetgrouplabel`.

`\glsgetgrouplabel`

```
6270 \newcommand*{\glsgetgrouplabel}[1]{%
6271 \ifthenelse{\equal{#1}{\glssymbolsgroupname}}{\glssymbols}{%
6272 \ifthenelse{\equal{#1}{\glsnumbersgroupname}}{\glsnumbers}{#1}}
```

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

`\setentrycounter`

```
6273 \newcommand*{\setentrycounter}[2][ ]{%
6274   \def\@glo@counterprefix{#1}%
6275   \ifx\@glo@counterprefix\empty
6276     \def\@glo@counterprefix{.}%
6277   \else
6278     \def\@glo@counterprefix{.#1.}%
6279   \fi
6280   \def\glsentrycounter{#2}%
6281 }
```

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

`\setglossarystyle`

```
6282 \newcommand*{\setglossarystyle}[1]{%
6283   \ifcsundef{@glsstyle@#1}%
6284   {%
6285     \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
6286   }%
6287   {%
6288     \csname @glsstyle@#1\endcsname
6289   }%
```

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

```
6290   \ifx\@glossary@default@style\relax
6291     \protected@edef\@glossary@default@style{#1}%
6292   \fi
6293 }
```

`\glossarystyle`

```
6294 \newcommand*{\glossarystyle}[1]{%
6295   \ifcsundef{@glsstyle@#1}%
6296   {%
6297     \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
6298   }%
6299   {%
6300     \GlossariesWarning
```



```

6301   {Deprecated command \string\glossarystyle.^~J
6302   I recommend you switch to \string\setglossarystyle\space unless
6303   you want to maintain backward compatibility}%
6304   \setglossentrycompatibility
6305   \csname @glsstyle@#1\endcsname
6306   \ifcsdef{@glscompstyle@#1}%
6307   {\setglossentrycompatibility\csuse{@glscompstyle@#1}}%
6308   {}%
6309   }%

```

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

```

6310   \ifx\@glossary@default@style\relax
6311   \protected@edef\@glossary@default@style{#1}%
6312   \fi
6313 }

```

`\newglossarystyle` New glossary styles can be defined using:

```
\newglossarystyle{<name>}{<definition>}
```

The *<definition>* argument should redefine `\theglossary`, `\glossaryheader`, `\glsgroupheading`, `\glossaryentryfield` and `\glsgroupskip` (see [section 1.19](#) for the definitions of predefined styles). Glossary styles should not redefine `\glossarypreamble` and `\glossarypostamble`, as the user should be able to switch between styles without affecting the pre- and postambles.

```

6314 \newcommand{\newglossarystyle}[2]{%
6315   \ifcsundef{@glsstyle@#1}%
6316   {%
6317     \expandafter\def\csname @glsstyle@#1\endcsname{#2}%
6318   }%
6319   {%
6320     \PackageError{glossaries}{Glossary style ‘#1’ is already defined}{}%
6321   }%
6322 }

```

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

```

6323 \newcommand{\renewglossarystyle}[2]{%
6324   \ifcsundef{@glsstyle@#1}%
6325   {%
6326     \PackageError{glossaries}{Glossary style ‘#1’ isn’t already defined}{}%
6327   }%
6328   {%
6329     \csdef{@glsstyle@#1}{#2}%
6330   }%
6331 }

```

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`

```
6332 \newcommand*{\glsnamefont}[1]{#1}
```

Each glossary entry has an associated number list (usually page numbers) that indicate where in the document the entry has been used. The format for these number lists can be changed using the format key in commands like `\glslink`. The default format is given by `\glshypernumber`. This takes a single argument which may be a single number, a number range or a number list. The number ranges are delimited with `\delimR`, the number lists are delimited with `\delimN`.

If the document doesn't have hyperlinks, the numbers can be displayed just as they are, but if the document supports hyperlinks, the numbers should link to the relevant location. This means extracting the individual numbers from the list or ranges. The package does this with the `\hyperpage` command, but this is encoded for comma and dash delimiters and only for the page counter, but this code needs to be more general. So I have adapted the code used in the package.

`\glshypernumber`

```
6333 \ifcsundef{hyperlink}%
6334 {%
6335   \def\glshypernumber#1{#1}%
6336 }%
6337 {%
6338   \def\glshypernumber#1{\@glshypernumber#1\nohyperpage{}}\@nil}
6339 }
```

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

```
6340 \def\@glshypernumber#1\nohyperpage#2#3\@nil{%
6341   \ifx\#1\%
6342   \else
6343     \@delimR#1\delimR\delimR\%
6344   \fi
6345   \ifx\#2\%
6346   \else
6347     #2%
6348   \fi
6349   \ifx\#3\%
6350   \else
6351     \@glshypernumber#3\@nil
6352   \fi
6353 }
```

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

\@delimR

```
6354 \def\@delimR#1\delimR #2\delimR #3\\{%
6355 \ifx\\#2\\%
6356   \@delimN{#1}%
6357 \else
6358   \@gls@numberlink{#1}\delimR\@gls@numberlink{#2}%
6359 \fi}
```

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

\@delimN

```
6360 \def\@delimN#1{\@delimN#1\delimN \delimN\\}
6361 \def\@delimN#1\delimN #2\delimN#3\\{%
6362 \ifx\\#3\\%
6363   \@gls@numberlink{#1}%
6364 \else
6365   \@gls@numberlink{#1}\delimN\@gls@numberlink{#2}%
6366 \fi
6367 }
```

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

```
6368 \def\@gls@numberlink#1{%
6369 \begingroup
6370 \toks@={}%
6371 \@gls@removespaces#1 \@nil
6372 \endgroup}

6373 \def\@gls@removespaces#1 #2\@nil{%
6374 \toks@=\expandafter{\the\toks@#1}%
6375 \ifx\\#2\\%
6376   \edef\x{\the\toks@}%
6377   \ifx\x\empty
6378     \else

6379     \hyperlink{\glsentrycounter\@glo@counterprefix\the\toks@}%
6380               {\the\toks@}%
6381   \fi
6382 \else
6383   \@gls@ReturnAfterFi{%
6384     \@gls@removespaces#2\@nil
6385   }%
6386 \fi
6387 }
6388 \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`
6389 `\newcommand*{\hyperrm}[1]{\textrm{\glshypernumber{#1}}}`

`\hypersf`
6390 `\newcommand*{\hypersf}[1]{\textsf{\glshypernumber{#1}}}`

`\hypertt`
6391 `\newcommand*{\hypertt}[1]{\texttt{\glshypernumber{#1}}}`

`\hyperbf`
6392 `\newcommand*{\hyperbf}[1]{\textbf{\glshypernumber{#1}}}`

`\hypermd`
6393 `\newcommand*{\hypermd}[1]{\textmd{\glshypernumber{#1}}}`

`\hyperit`
6394 `\newcommand*{\hyperit}[1]{\textit{\glshypernumber{#1}}}`

`\hypersl`
6395 `\newcommand*{\hypersl}[1]{\textsl{\glshypernumber{#1}}}`

`\hyperup`
6396 `\newcommand*{\hyperup}[1]{\textup{\glshypernumber{#1}}}`

`\hypersc`
6397 `\newcommand*{\hypersc}[1]{\textsc{\glshypernumber{#1}}}`

`\hyperemph`
6398 `\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.

```

6399 \newcommand{\oldacronym}[4][\gls@label]{%
6400   \def\gls@label{#2}%
6401   \newacronym[#4]{#1}{#2}{#3}%
6402   \ifcsundef{xspace}%
6403   {%
6404     \expandafter\edef\csname#1\endcsname{%
6405       \noexpand\@ifstar{\noexpand\Gls{#1}}{\noexpand\gls{#1}}}%
6406   }%
6407 }%
6408 {%
6409   \expandafter\edef\csname#1\endcsname{%
6410     \noexpand\@ifstar{\noexpand\Gls{#1}\noexpand\xspace}{%
6411       \noexpand\gls{#1}\noexpand\xspace}%
6412   }%
6413 }%
6414 }

```

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

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

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

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

`\newacronym`

```

6415 \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.

```

6416 \newcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}

```

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

`\glstextup`

```

6417 \newrobustcmd*{\glstextup}[1]{\ifdef\textulc{\textulc{#1}}{\textup{#1}}}

```

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

```

\glsshortkey
6418 \newcommand*{\glsshortkey}{short}

sshortpluralkey
6419 \newcommand*{\glsshortpluralkey}{shortplural}

\glslongkey
6420 \newcommand*{\glslongkey}{long}

lslongpluralkey
6421 \newcommand*{\glslongpluralkey}{longplural}

\acrfull  Full form of the acronym.
6422 \newrobustcmd*{\acrfull}{\@gls@hyp@opt\@ns@acrfull}

6423 \newcommand*\ns@acrfull[2][{}]{%
6424   \new@ifnextchar[{\@acrfull{#1}{#2}}]{%
6425     {\@acrfull{#1}{#2}[]}%
6426 }

\@acrfull  Low-level macro:
6427 \def\@acrfull#1#2[#3]{%
  Make it easier for acronym styles to change this:
6428   \acrfullfmt{#1}{#2}{#3}%
6429 }

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

\acrfullfmt  No case change full format.
6430 \newcommand*{\acrfullfmt}[3]{%
6431   \acrlinkfullformat{\@acrlong}{\@acrshort}{#1}{#2}{#3}%
6432 }

acrlinkfullformat  Format for full links like \acrfull. Syntax: \acrlinkfullformat{<long cs>}{<short cs>}
  {<options>}{<label>}{<insert>}
6433 \newcommand{\acrlinkfullformat}[5]{%
6434   \acrfullformat{#1}{#3}{#4}[#5]{#2}{#3}{#4}[]}%
6435 }

\acrfullformat  Default full form is <long> (<short>).
6436 \newcommand{\acrfullformat}[2]{#1\glsspace(#2)}

\glsspace  Robust space to ensure it's written to the .glsdefs file.
6437 \newrobustcmd{\glsspace}{\space}

```

Default format for full acronym

`\Acrfull`

```
6438 \newrobustcmd*{\Acrfull}{\@gls@hyp@opt\ns@Acrfull}

6439 \newcommand*\ns@Acrfull[2] [] {%
6440   \new@ifnextchar[{\@Acrfull{#1}{#2}}%
6441     {\@Acrfull{#1}{#2} []}%
6442 }
```

Low-level macro:

```
6443 \def\@Acrfull#1#2[#3] {%
```

Make it easier for acronym styles to change this:

```
6444   \Acrfullfmt{#1}{#2}{#3}%
6445 }
```

`\Acrfullfmt` First letter upper case full format.

```
6446 \newcommand*{\Acrfullfmt}[3] {%
6447   \acrlinkfullformat{\@Acrlong}{\@acrshort}{#1}{#2}{#3}%
6448 }
```

`\ACRfull`

```
6449 \newrobustcmd*{\ACRfull}{\@gls@hyp@opt\ns@ACRfull}

6450 \newcommand*\ns@ACRfull[2] [] {%
6451   \new@ifnextchar[{\@ACRfull{#1}{#2}}%
6452     {\@ACRfull{#1}{#2} []}%
6453 }
```

Low-level macro:

```
6454 \def\@ACRfull#1#2[#3] {%
```

Make it easier for acronym styles to change this:

```
6455   \ACRfullfmt{#1}{#2}{#3}%
6456 }
```

`\ACRfullfmt` All upper case full format.

```
6457 \newcommand*{\ACRfullfmt}[3] {%
6458   \acrlinkfullformat{\@ACRlong}{\@ACRshort}{#1}{#2}{#3}%
6459 }
```

Plural:

`\acrfullpl`

```
6460 \newrobustcmd*{\acrfullpl}{\@gls@hyp@opt\ns@acrfullpl}

6461 \newcommand*\ns@acrfullpl[2] [] {%
6462   \new@ifnextchar[{\@acrfullpl{#1}{#2}}%
6463     {\@acrfullpl{#1}{#2} []}%
6464 }
```

Low-level macro:

```
6465 \def\@acrfullpl#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6466 \acrfullplfmt{#1}{#2}{#3}%  
6467 }
```

\acrfullplfmt No case change plural full format.

```
6468 \newcommand*\@acrfullplfmt}[3]{%  
6469 \acrlinkfullformat{\@acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%  
6470 }
```

\Acrfullpl

```
6471 \newrobustcmd*\@Acrfullpl{\@gls@hyp@opt\ns@Acrfullpl}  
  
6472 \newcommand*\ns@Acrfullpl[2][ ]{%  
6473 \new@ifnextchar[{\@Acrfullpl{#1}{#2}}%  
6474 {\@Acrfullpl{#1}{#2}[ ]}%  
6475 }
```

Low-level macro:

```
6476 \def\@Acrfullpl#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6477 \Acrfullplfmt{#1}{#2}{#3}%  
6478 }
```

\Acrfullplfmt First letter upper case plural full format.

```
6479 \newcommand*\@Acrfullplfmt}[3]{%  
6480 \acrlinkfullformat{\@acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%  
6481 }
```

\ACRfullpl

```
6482 \newrobustcmd*\@ACRfullpl{\@gls@hyp@opt\ns@ACRfullpl}  
  
6483 \newcommand*\ns@ACRfullpl[2][ ]{%  
6484 \new@ifnextchar[{\@ACRfullpl{#1}{#2}}%  
6485 {\@ACRfullpl{#1}{#2}[ ]}%  
6486 }
```

Low-level macro:

```
6487 \def\@ACRfullpl#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6488 \ACRfullplfmt{#1}{#2}{#3}%  
6489 }
```

\ACRfullplfmt All upper case plural full format.

```
6490 \newcommand*\@ACRfullplfmt}[3]{%  
6491 \acrlinkfullformat{\@ACRlongpl}{\@ACRshortpl}{#1}{#2}{#3}%  
6492 }
```


1.18 Predefined acronym styles

`\acronymfont` This is only used with the additional acronym styles:
6493 `\newcommand{\acronymfont}[1]{#1}`

`\firstacronymfont` This is only used with the additional acronym styles:
6494 `\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.
6495 `\newcommand*{\acrnameformat}[2]{\acronymfont{#1}}`

Define some tokens used by `\newacronym`:

`\glskeylisttok`
6496 `\newtoks\glskeylisttok`

`\glslabeltok`
6497 `\newtoks\glslabeltok`

`\glsshorttok`
6498 `\newtoks\glsshorttok`

`\glslongtok`
6499 `\newtoks\glslongtok`

`\newacronymhook` Provide a hook for `\newacronym`:
6500 `\newcommand*{\newacronymhook}{}`

`\genericNewAcronym` New improved version of setting the acronym style.
6501 `\newcommand*{\SetGenericNewAcronym}{%`
Change the behaviour of `\Glsentryname` to workaround expansion issues that cause a problem for `\makefirstuc`
6502 `\let\@Gls@entryname\@Gls@acrentryname`
Change the way acronyms are defined:
6503 `\renewcommand{\newacronym}[4][\{%`
6504 `\ifdefempty{\@glsacronymlists}%`
6505 `{%`
6506 `\def\@glo@type{\acronymtype}%`
6507 `\setkeys{glossentry}{##1}%`
6508 `\DeclareAcronymList{\@glo@type}%`
6509 `}%`
6510 `{}}%`
6511 `\glskeylisttok{##1}%`
6512 `\glslabeltok{##2}%`
6513 `\glsshorttok{##3}%`
6514 `\glslongtok{##4}%`

```

6515 \newacronymhook
6516 \protected@edef\@do@newglossaryentry{%
6517 \noexpand\newglossaryentry{\the\glslabeltok}%
6518 {%
6519 type=\acronymtype,%
6520 name={\expandonce{\acronymentry{##2}}},%
6521 sort={\acronymsort{\the\glsshorttok}{\the\glslongtok}},%
6522 text={\the\glsshorttok},%
6523 short={\the\glsshorttok},%
6524 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6525 long={\the\glslongtok},%
6526 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6527 \GenericAcronymFields,%
6528 \the\glskeylisttok
6529 }%
6530 }%
6531 \@do@newglossaryentry
6532 }%

```

Make sure that \acrfull etc reflects the new style:

```

6533 \renewcommand*{\acrfullfmt}[3]{%
6534 \glslink[##1]{##2}{\genacrfullformat{##2}{##3}}}%
6535 \renewcommand*{\Acrfullfmt}[3]{%
6536 \glslink[##1]{##2}{\Genacrfullformat{##2}{##3}}}%
6537 \renewcommand*{\ACRfullfmt}[3]{%
6538 \glslink[##1]{##2}{%
6539 \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}}%
6540 \renewcommand*{\acrfullplfmt}[3]{%
6541 \glslink[##1]{##2}{\genplacrfullformat{##2}{##3}}}%
6542 \renewcommand*{\Acrfullplfmt}[3]{%
6543 \glslink[##1]{##2}{\Genplacrfullformat{##2}{##3}}}%
6544 \renewcommand*{\ACRfullplfmt}[3]{%
6545 \glslink[##1]{##2}{%
6546 \mfirstucMakeUppercase{\genplacrfullformat{##2}{##3}}}}%

```

Make sure that \glsentryfull etc reflects the new style:

```

6547 \renewcommand*{\glsentryfull}[1]{\genacrfullformat{##1}{}}%
6548 \renewcommand*{\Glsentryfull}[1]{\Genacrfullformat{##1}{}}%
6549 \renewcommand*{\glsentryfullpl}[1]{\genplacrfullformat{##1}{}}%
6550 \renewcommand*{\Glsentryfullpl}[1]{\Genplacrfullformat{##1}{}}%
6551 }

```

\icAcronymFields Fields used by \SetGenericNewAcronym that can be changed by the acronym style.

```

6552 \newcommand*{\GenericAcronymFields}{description={\the\glslongtok}}

```

\acronymentry `\acronymentry{<label>}`

Display style for the name field in the list of acronyms.

```

6553 \newcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{#1}}}

```

`\acronymsort` `\acronymsort{<short>}{<long>}`

Default sort format for acronyms.

```
6554 \newcommand*{\acronymsort}[2]{#1}
```

`\setacronymstyle` `\setacronymstyle{<style name>}`

```
6555 \newcommand*{\setacronymstyle}[1]{%
6556   \ifcsundef{@glsacr@dispstyle@#1}
6557   {%
6558     \PackageError{glossaries}{Undefined acronym style ‘#1’}{}%
6559   }%
6560   {%
6561     \ifdefempty{\@glsacronymlists}%
6562     {%
6563       \DeclareAcronymList{\acronymtype}%
6564     }%
6565     {}%
6566     \SetGenericNewAcronym
6567     \GlsUseAcrStyleDefs{#1}%
6568     \@for\@gls@type:=\@glsacronymlists\do{%
6569       \defglentryfmt[\@gls@type]{\GlsUseAcrEntryDispStyle{#1}}%
6570     }%
6571   }%
6572 }
```

`\newacronymstyle` `\newacronymstyle{<style name>}{<entry format definition>}{<display definitions>}`

Defines a new acronym style called *<style name>*.

```
6573 \newcommand*{\newacronymstyle}[3]{%
6574   \ifcsdef{@glsacr@dispstyle@#1}%
6575   {%
6576     \PackageError{glossaries}{Acronym style ‘#1’ already exists}{}%
6577   }%
6578   {%
6579     \csdef{@glsacr@dispstyle@#1}{#2}%
6580     \csdef{@glsacr@styledefs@#1}{#3}%
6581   }%
6582 }
```

`\renewacronymstyle` Redefines the given acronym style.

```
6583 \newcommand*{\renewacronymstyle}[3]{%
6584   \ifcsdef{@glsacr@dispstyle@#1}%
6585   {%
```

```

6586 \csdef{@glsacr@dispstyle@#1}{#2}%
6587 \csdef{@glsacr@styledefs@#1}{#3}%
6588 }%
6589 {%
6590 \PackageError{glossaries}{Acronym style ‘#1’ doesn’t exist}{}%
6591 }%
6592 }

```

rEntryDispStyle

```

6593 \newcommand*{\GlsUseAcrEntryDispStyle}[1]{\csuse{@glsacr@dispstyle@#1}}

```

UseAcrStyleDefs

```

6594 \newcommand*{\GlsUseAcrStyleDefs}[1]{\csuse{@glsacr@styledefs@#1}}

```

Predefined acronym styles:

long-short *(long)* (*short*) acronym style.

```

6595 \newacronymstyle{long-short}%
6596 {%

```

Check for long form in case this is a mixed glossary.

```

6597 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6598 }%
6599 {%
6600 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6601 \renewcommand*{\genacrfullformat}[2]{%
6602 \glstrylong{##1}##2\space
6603 (\protect\firstacronymfont{\glstryshort{##1}})%
6604 }%
6605 \renewcommand*{\Genacrfullformat}[2]{%
6606 \Glsentrylong{##1}##2\space
6607 (\protect\firstacronymfont{\glstryshort{##1}})%
6608 }%
6609 \renewcommand*{\genplacrfullformat}[2]{%
6610 \glstrylongpl{##1}##2\space
6611 (\protect\firstacronymfont{\glstryshortpl{##1}})%
6612 }%
6613 \renewcommand*{\Genplacrfullformat}[2]{%
6614 \Glsentrylongpl{##1}##2\space
6615 (\protect\firstacronymfont{\glstryshortpl{##1}})%
6616 }%
6617 \renewcommand*{\acronymentry}[1]{\acronymfont{\glstryshort{##1}}}%
6618 \renewcommand*{\acronymsort}[2]{##1}%
6619 \renewcommand*{\acronymfont}[1]{##1}%
6620 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
6621 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6622 }

```

long-sp-short Similar to the previous style but allows the space between the long and short form to be customized.

```

6623 \newacronymstyle{long-sp-short}%
6624 {%
    Check for long form in case this is a mixed glossary.
6625 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6626 }%
6627 {%
6628 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6629 \renewcommand*{\genacrfullformat}[2]{%
6630 \glentrylong{##1}##2\glsacspace{##1}%
6631 (\protect\firstacronymfont{\glentryshort{##1}})%
6632 }%
6633 \renewcommand*{\Genacrfullformat}[2]{%
6634 \Glsentrylong{##1}##2\glsacspace{##1}%
6635 (\protect\firstacronymfont{\glentryshort{##1}})%
6636 }%
6637 \renewcommand*{\genplacrfullformat}[2]{%
6638 \glentrylongpl{##1}##2\glsacspace{##1}%
6639 (\protect\firstacronymfont{\glentryshortpl{##1}})%
6640 }%
6641 \renewcommand*{\Genplacrfullformat}[2]{%
6642 \Glsentrylongpl{##1}##2\glsacspace{##1}%
6643 (\protect\firstacronymfont{\glentryshortpl{##1}})%
6644 }%
6645 \renewcommand*{\acronymentry}[1]{\acronymfont{\glentryshort{##1}}}%
6646 \renewcommand*{\acronymsort}[2]{##1}%
6647 \renewcommand*{\acronymfont}[1]{##1}%
6648 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
6649 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6650 }

```

`\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.

```

6651 \newcommand*{\glsacspace}[1]{%
6652 \settowidth{\dimen@}{(\firstacronymfont{\glentryshort{##1}})}%
6653 \ifdim\dimen@<3em~\else\space\fi
6654 }

```

`short-long` (*short*) (*long*) acronym style.

```

6655 \newacronymstyle{short-long}%
6656 {%
    Check for long form in case this is a mixed glossary.
6657 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6658 }%
6659 {%
6660 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6661 \renewcommand*{\genacrfullformat}[2]{%
6662 \protect\firstacronymfont{\glentryshort{##1}}##2\space
6663 (\glentrylong{##1})%

```

```

6664 }%
6665 \renewcommand*{\Genacrfullformat}[2]{%
6666   \protect\firstacronymfont{\Glsentryshort{##1}}##2\space
6667   (\glsentrylong{##1})%
6668 }%
6669 \renewcommand*{\genplacrfullformat}[2]{%
6670   \protect\firstacronymfont{\glsentryshortpl{##1}}##2\space
6671   (\glsentrylongpl{##1})%
6672 }%
6673 \renewcommand*{\Genplacrfullformat}[2]{%
6674   \protect\firstacronymfont{\Glsentryshortpl{##1}}##2\space
6675   (\glsentrylongpl{##1})%
6676 }%

6677 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6678 \renewcommand*{\acronymsort}[2]{##1}%
6679 \renewcommand*{\acronymfont}[1]{##1}%
6680 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
6681 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6682 }

```

long-sc-short *<long>* (\textsc{<short>}) acronym style.

```

6683 \newacronymstyle{long-sc-short}%
6684 {%
6685   \GlsUseAcrEntryDisplayStyle{long-short}%
6686 }%
6687 {%
6688   \GlsUseAcrStyleDefs{long-short}%
6689   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6690   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6691 }

```

long-sm-short *<long>* (\textsmaller{<short>}) acronym style.

```

6692 \newacronymstyle{long-sm-short}%
6693 {%
6694   \GlsUseAcrEntryDisplayStyle{long-short}%
6695 }%
6696 {%
6697   \GlsUseAcrStyleDefs{long-short}%
6698   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6699   \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6700 }

```

sc-short-long *<short>* (\textsc{<long>}) acronym style.

```

6701 \newacronymstyle{sc-short-long}%
6702 {%
6703   \GlsUseAcrEntryDisplayStyle{short-long}%
6704 }%
6705 {%

```

```

6706 \GlsUseAcrStyleDefs{short-long}%
6707 \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6708 \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6709 }

```

sm-short-long *<short>* (*\textsmaller{<long>}*) acronym style.

```

6710 \newacronymstyle{sm-short-long}%
6711 {%
6712 \GlsUseAcrEntryDisplayStyle{short-long}%
6713 }%
6714 {%
6715 \GlsUseAcrStyleDefs{short-long}%
6716 \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6717 \renewcommand*{\acrpluralsuffix}{\glacrpluralsuffix}%
6718 }

```

long-short-desc *<long>* (*{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

6719 \newacronymstyle{long-short-desc}%
6720 {%
6721 \GlsUseAcrEntryDisplayStyle{long-short}%
6722 }%
6723 {%
6724 \GlsUseAcrStyleDefs{long-short}%
6725 \renewcommand*{\GenericAcronymFields}{}%
6726 \renewcommand*{\acronymsort}[2]{##2}%
6727 \renewcommand*{\acronymentry}[1]{%
6728 \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6729 }

```

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`.

```

6730 \newacronymstyle{long-sp-short-desc}%
6731 {%
6732 \GlsUseAcrEntryDisplayStyle{long-sp-short}%
6733 }%
6734 {%
6735 \GlsUseAcrStyleDefs{long-sp-short}%
6736 \renewcommand*{\GenericAcronymFields}{}%
6737 \renewcommand*{\acronymsort}[2]{##2}%
6738 \renewcommand*{\acronymentry}[1]{%
6739 \glentrylong{##1}\glsacspace{##1}(\acronymfont{\glentryshort{##1}})}%
6740 }

```

g-sc-short-desc *<long>* (*\textsc{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

6741 \newacronymstyle{long-sc-short-desc}%
6742 {%

```

```

6743 \GlsUseAcrEntryDispStyle{long-sc-short}%
6744 }%
6745 {%
6746 \GlsUseAcrStyleDefs{long-sc-short}%
6747 \renewcommand*{\GenericAcronymFields}{}%
6748 \renewcommand*{\acronymsort}[2]{##2}%
6749 \renewcommand*{\acronymentry}[1]{%
6750     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6751 }

```

g-sm-short-desc *<long>* (\textsmaller{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

6752 \newacronymstyle{long-sm-short-desc}%
6753 {%
6754 \GlsUseAcrEntryDispStyle{long-sm-short}%
6755 }%
6756 {%
6757 \GlsUseAcrStyleDefs{long-sm-short}%
6758 \renewcommand*{\GenericAcronymFields}{}%
6759 \renewcommand*{\acronymsort}[2]{##2}%
6760 \renewcommand*{\acronymentry}[1]{%
6761     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6762 }

```

short-long-desc *<short>* ({<long>}) acronym style that has an accompanying description (which the user needs to supply).

```

6763 \newacronymstyle{short-long-desc}%
6764 {%
6765 \GlsUseAcrEntryDispStyle{short-long}%
6766 }%
6767 {%
6768 \GlsUseAcrStyleDefs{short-long}%
6769 \renewcommand*{\GenericAcronymFields}{}%
6770 \renewcommand*{\acronymsort}[2]{##2}%
6771 \renewcommand*{\acronymentry}[1]{%
6772     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6773 }

```

short-long-desc *<long>* (\textsc{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

6774 \newacronymstyle{sc-short-long-desc}%
6775 {%
6776 \GlsUseAcrEntryDispStyle{sc-short-long}%
6777 }%
6778 {%
6779 \GlsUseAcrStyleDefs{sc-short-long}%
6780 \renewcommand*{\GenericAcronymFields}{}%
6781 \renewcommand*{\acronymsort}[2]{##2}%
6782 \renewcommand*{\acronymentry}[1]{%

```



```

6783 \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6784 }

```

short-long-desc <long> (\textsmaller{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

6785 \newacronymstyle{sm-short-long-desc}%
6786 {%
6787 \GlsUseAcrEntryDispStyle{sm-short-long}%
6788 }%
6789 {%
6790 \GlsUseAcrStyleDefs{sm-short-long}%
6791 \renewcommand*{\GenericAcronymFields}{}%
6792 \renewcommand*{\acronymsort}[2]{##2}%
6793 \renewcommand*{\acronymentry}[1]{%
6794 \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6795 }

```

dua <long> only acronym style.

```

6796 \newacronymstyle{dua}%
6797 {%

```

Check for long form in case this is a mixed glossary.

```

6798 \ifdefempty\glscustomtext
6799 {%
6800 \ifglshaslong{\glslabel}%
6801 {%
6802 \glsifplural
6803 {%

```

Plural form:

```

6804 \glscapscase
6805 {%

```

Plural form, don't adjust case:

```

6806 \glsentrylongpl{\glslabel}\glsinsert
6807 }%
6808 {%

```

Plural form, make first letter upper case:

```

6809 \Glsentrylongpl{\glslabel}\glsinsert
6810 }%
6811 {%

```

Plural form, all caps:

```

6812 \mfirstucMakeUppercase
6813 {\glsentrylongpl{\glslabel}\glsinsert}%
6814 }%
6815 }%
6816 {%

```

Singular form

```
6817      \glscapscase
6818      {%
```

Singular form, don't adjust case:

```
6819      \glentrylong{\glslabel}\glsinsert
6820      }%
6821      {%
```

Subsequent singular form, make first letter upper case:

```
6822      \Glsentrylong{\glslabel}\glsinsert
6823      }%
6824      {%
```

Subsequent singular form, all caps:

```
6825      \mfirstucMakeUppercase
6826      {\glentrylong{\glslabel}\glsinsert}%
6827      }%
6828      }%
6829      }%
6830      {%
```

Not an acronym:

```
6831      \glsgenentryfmt
6832      }%
6833      }%
6834      {\glscustomtext\glsinsert}%
6835      }%
6836      {%
6837      \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%

6838      \renewcommand*{\acrfullfmt}[3]{%
6839      \glslink[##1]{##2}{\glentrylong{##2}##3\space
6840      (\acronymfont{\glentryshort{##2}})}}%
6841      \renewcommand*{\Acrfullfmt}[3]{%
6842      \glslink[##1]{##2}{\Glsentrylong{##2}##3\space
6843      (\acronymfont{\glentryshort{##2}})}}%
6844      \renewcommand*{\ACRfullfmt}[3]{%
6845      \glslink[##1]{##2}{%
6846      \mfirstucMakeUppercase{\glentrylong{##2}##3\space
6847      (\acronymfont{\glentryshort{##2}})}}}%

6848      \renewcommand*{\acrfullplfmt}[3]{%
6849      \glslink[##1]{##2}{\glentrylongpl{##2}##3\space
6850      (\acronymfont{\glentryshortpl{##2}})}}%

6851      \renewcommand*{\Acrfullplfmt}[3]{%
6852      \glslink[##1]{##2}{\Glsentrylongpl{##2}##3\space
6853      (\acronymfont{\glentryshortpl{##2}})}}%
6854      \renewcommand*{\ACRfullplfmt}[3]{%
6855      \glslink[##1]{##2}{%
```

```

6856      \mfirstucMakeUppercase{\glentrylongpl{##2}##3\space
6857      (\acronymfont{\glentryshortpl{##2}})}}}%
6858 \renewcommand*{\glentryfull}[1]{%
6859   \glentrylong{##1}\space(\acronymfont{\glentryshort{##1}})%
6860 }%
6861 \renewcommand*{\Glsentryfull}[1]{%
6862   \Glsentrylong{##1}\space(\acronymfont{\glentryshort{##1}})%
6863 }%
6864 \renewcommand*{\glentryfullpl}[1]{%
6865   \glentrylongpl{##1}\space(\acronymfont{\glentryshortpl{##1}})%
6866 }%
6867 \renewcommand*{\Glsentryfullpl}[1]{%
6868   \Glsentrylongpl{##1}\space(\acronymfont{\glentryshortpl{##1}})%
6869 }%
6870 \renewcommand*{\acronymentry}[1]{\acronymfont{\glentryshort{##1}}}%
6871 \renewcommand*{\acronymsort}[2]{##1}%
6872 \renewcommand*{\acronymfont}[1]{##1}%
6873 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6874 }

```

dua-desc *<long>* only acronym style with user-supplied description.

```

6875 \newacronymstyle{dua-desc}%
6876 {%
6877   \GlsUseAcrEntryDispStyle{dua}%
6878 }%
6879 {%
6880   \GlsUseAcrStyleDefs{dua}%
6881   \renewcommand*{\GenericAcronymFields}{}%
6882   \renewcommand*{\acronymentry}[1]{\acronymfont{\glentrylong{##1}}}%
6883   \renewcommand*{\acronymsort}[2]{##2}%
6884 }%

```

footnote *<short>*\footnote{*<long>*} acronym style.

```

6885 \newacronymstyle{footnote}%
6886 {%
6887   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6888 }%
6889 {%
6890   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6891   \glshyperfirstfalse
6892   \renewcommand*{\genacrfullformat}[2]{%
6893     \protect\firstacronymfont{\glentryshort{##1}}##2%
6894     \protect\footnote{\glentrylong{##1}}%
6895   }%
6896   \renewcommand*{\Genacrfullformat}[2]{%

```

```

6897 \firstacronymfont{\Glsentryshort{##1}}##2%
6898 \protect\footnote{\glsentrylong{##1}}%
6899 }%
6900 \renewcommand*{\genplacrfullformat}[2]{%
6901 \protect\firstacronymfont{\glsentryshortpl{##1}}##2%
6902 \protect\footnote{\glsentrylongpl{##1}}%
6903 }%
6904 \renewcommand*{\Genplacrfullformat}[2]{%
6905 \protect\firstacronymfont{\Glsentryshortpl{##1}}##2%
6906 \protect\footnote{\glsentrylongpl{##1}}%
6907 }%
6908 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6909 \renewcommand*{\acronymsort}[2]{##1}%
6910 \renewcommand*{\acronymfont}[1]{##1}%
6911 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%

```

Don't use footnotes for \acrfull:

```

6912 \renewcommand*{\acrfullfmt}[3]{%
6913 \glslink[##1]{##2}{\acronymfont{\glsentryshort{##2}}##3\space
6914 (\glsentrylong{##2})}%
6915 \renewcommand*{\Acrfullfmt}[3]{%
6916 \glslink[##1]{##2}{\acronymfont{\Glsentryshort{##2}}##3\space
6917 (\glsentrylong{##2})}%
6918 \renewcommand*{\ACRfullfmt}[3]{%
6919 \glslink[##1]{##2}{%
6920 \mfirstucMakeUppercase{\acronymfont{\glsentryshort{##2}}##3\space
6921 (\glsentrylong{##2})}}}%
6922 \renewcommand*{\acrfullplfmt}[3]{%
6923 \glslink[##1]{##2}{\acronymfont{\glsentryshortpl{##2}}##3\space
6924 (\glsentrylongpl{##2})}%
6925 \renewcommand*{\Acrfullplfmt}[3]{%
6926 \glslink[##1]{##2}{\acronymfont{\Glsentryshortpl{##2}}##3\space
6927 (\glsentrylongpl{##2})}%
6928 \renewcommand*{\ACRfullplfmt}[3]{%
6929 \glslink[##1]{##2}{%
6930 \mfirstucMakeUppercase{\acronymfont{\glsentryshortpl{##2}}##3\space
6931 (\glsentrylongpl{##2})}}}%

```

Similarly for \glsentryfull etc:

```

6932 \renewcommand*{\glsentryfull}[1]{%
6933 \acronymfont{\glsentryshort{##1}}\space(\glsentrylong{##1})}%
6934 \renewcommand*{\Glsentryfull}[1]{%
6935 \acronymfont{\Glsentryshort{##1}}\space(\glsentrylong{##1})}%
6936 \renewcommand*{\glsentryfullpl}[1]{%
6937 \acronymfont{\glsentryshortpl{##1}}\space(\glsentrylongpl{##1})}%
6938 \renewcommand*{\Glsentryfullpl}[1]{%
6939 \acronymfont{\Glsentryshortpl{##1}}\space(\glsentrylongpl{##1})}%
6940 }

```

footnote-sc \textsc{<short>}\footnote{<long>} acronym style.

```

6941 \newacronymstyle{footnote-sc}%
6942 {%
6943   \GlsUseAcrEntryDisplayStyle{footnote}%
6944 }%
6945 {%
6946   \GlsUseAcrStyleDefs{footnote}%
6947   \renewcommand{\acronymentry}[1]{\acronymfont{\glentryshort{##1}}}
6948   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6949   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6950 }%

```

footnote-sm \textsmaller{<short>}\footnote{<long>} acronym style.

```

6951 \newacronymstyle{footnote-sm}%
6952 {%
6953   \GlsUseAcrEntryDisplayStyle{footnote}%
6954 }%
6955 {%
6956   \GlsUseAcrStyleDefs{footnote}%
6957   \renewcommand{\acronymentry}[1]{\acronymfont{\glentryshort{##1}}}
6958   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6959   \renewcommand*{\acrpluralsuffix}{\glacrpluralsuffix}%
6960 }%

```

footnote-desc <short>\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

6961 \newacronymstyle{footnote-desc}%
6962 {%
6963   \GlsUseAcrEntryDisplayStyle{footnote}%
6964 }%
6965 {%
6966   \GlsUseAcrStyleDefs{footnote}%
6967   \renewcommand*{\GenericAcronymFields}{}%
6968   \renewcommand*{\acronymsort}[2]{##2}%
6969   \renewcommand*{\acronymentry}[1]{%
6970     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6971 }

```

footnote-sc-desc \textsc{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

6972 \newacronymstyle{footnote-sc-desc}%
6973 {%
6974   \GlsUseAcrEntryDisplayStyle{footnote-sc}%
6975 }%
6976 {%
6977   \GlsUseAcrStyleDefs{footnote-sc}%
6978   \renewcommand*{\GenericAcronymFields}{}%
6979   \renewcommand*{\acronymsort}[2]{##2}%
6980   \renewcommand*{\acronymentry}[1]{%
6981     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%

```

6982 }

ootnote-sm-desc \textsmaller{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

6983 \newacronymstyle{footnote-sm-desc}%

6984 {%

6985 \GlsUseAcrEntryDispStyle{footnote-sm}%

6986 }%

6987 {%

6988 \GlsUseAcrStyleDefs{footnote-sm}%

6989 \renewcommand*{\GenericAcronymFields}{}%

6990 \renewcommand*{\acronymsort}[2]{##2}%

6991 \renewcommand*{\acronymentry}[1]{%

6992 \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%

6993 }

AcronymSynonyms

6994 \newcommand*{\DefineAcronymSynonyms}{%

Short form

\acs

6995 \let\acs\acrshort

First letter uppercase short form

\Acs

6996 \let\Acs\Acrshort

Plural short form

\acsp

6997 \let\acsp\acrshortpl

First letter uppercase plural short form

\Acsp

6998 \let\Acsp\Acrshortpl

Long form

\acl

6999 \let\acl\aclong

Plural long form

\aclp

7000 \let\aclp\aclongpl

First letter upper case long form

```

\Acl
7001 \let\Acl\Acrlong
    First letter upper case plural long form

\Aclp
7002 \let\Aclp\Acrlongpl
    Full form

\acf
7003 \let\acf\acrfull
    Plural full form

\acfp
7004 \let\acfp\acrfullpl
    First letter upper case full form

\Acf
7005 \let\Acf\Acrfull
    First letter upper case plural full form

\Acfp
7006 \let\Acfp\Acrfullpl
    Standard form

\ac
7007 \let\ac\gls
    First upper case standard form

\Ac
7008 \let\Ac\Gls
    Standard plural form

\acp
7009 \let\acp\glspl
    Standard first letter upper case plural form

\Acp
7010 \let\Acp\Glspl
7011 }
    Define synonyms if required
7012 \ifglsacrshortcuts
7013 \DefineAcronymSynonyms
7014 \fi

```

These commands for setting the style are now deprecated but are kept for backward compatibility.

`\newcommand{\SetDefaultAcronymDisplayStyle}` Sets the default acronym display style for given glossary.

```
7015 \newcommand*{\SetDefaultAcronymDisplayStyle}[1]{%
7016   \def\glsentryfmt[#1]{\glsentryfmt}%
7017 }
```

`\newcommand{\DefaultNewAcronymDef}` Sets up the acronym definition for the default style. The information is provided by the tokens `\glslabeltok`, `\glsshorttok`, `\glslongtok` and `\glskeylisttok`.

```
7018 \newcommand*{\DefaultNewAcronymDef}{%
7019   \edef\@do@newglossaryentry{%
7020     \noexpand\newglossaryentry{\the\glslabeltok}%
7021     {%
7022       type=\acronymtype,%
7023       name={\the\glsshorttok},%
7024       sort={\the\glsshorttok},%
7025       text={\the\glsshorttok},%
7026       first={\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
7027       plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7028       firstplural={\acrfullformat{\noexpand\expandonce\noexpand\@glo@longpl}%
7029                     {\noexpand\expandonce\noexpand\@glo@shortpl}},%
7030       short={\the\glsshorttok},%
7031       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7032       long={\the\glslongtok},%
7033       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7034       description={\the\glslongtok},%
7035       descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
```

Remaining options specified by the user:

```
7036       \the\glskeylisttok
7037     }%
7038   }%
7039   \let\@org@gls@assign@firstpl\gls@assign@firstpl
7040   \let\@org@gls@assign@plural\gls@assign@plural
7041   \let\@org@gls@assign@descplural\gls@assign@descplural
7042   \def\gls@assign@firstpl##1##2{%
7043     \@gls@expand@field{##1}{firstpl}{##2}%
7044   }%
7045   \def\gls@assign@plural##1##2{%
7046     \@gls@expand@field{##1}{plural}{##2}%
7047   }%
7048   \def\gls@assign@descplural##1##2{%
7049     \@gls@expand@field{##1}{descplural}{##2}%
7050   }%
7051   \@do@newglossaryentry
7052   \let\gls@assign@firstpl\@org@gls@assign@firstpl
7053   \let\gls@assign@plural\@org@gls@assign@plural
7054   \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7055 }
```


ultAcronymStyle Set up the default acronym style:

```
7056 \newcommand*{\SetDefaultAcronymStyle}{%
```

Set the display style:

```
7057 \@for\@gls@type:=\@glsacronymlists\do{%
```

```
7058 \SetDefaultAcronymDisplayStyle{\@gls@type}%
```

```
7059 }%
```

Set up the definition of \newacronym:

```
7060 \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).

```
7061 \ifx\@glsacronymlists\@empty
```

```
7062 \def\@glo@type{\acronymtype}%
```

```
7063 \setkeys{glossentry}{##1}%
```

```
7064 \DeclareAcronymList{\@glo@type}%
```

```
7065 \SetDefaultAcronymDisplayStyle{\@glo@type}%
```

```
7066 \fi
```

```
7067 \glskeylisttok{##1}%
```

```
7068 \glslabeltok{##2}%
```

```
7069 \glsshorttok{##3}%
```

```
7070 \glslongtok{##4}%
```

```
7071 \newacronymhook
```

```
7072 \DefaultNewAcronymDef
```

```
7073 }%
```

```
7074 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
```

```
7075 }
```

\acrfootnote Used by the footnote acronym styles.

```
7076 \newcommand*{\acrfootnote}[3]{\acrlinkfootnote{#1}{#2}{#3}}
```

acrlinkfootnote

```
7077 \newcommand*{\acrlinkfootnote}[3]{%
```

```
7078 \footnote{\glslink[#1]{#2}{#3}}%
```

```
7079 }
```

acrnolinkfootnote

```
7080 \newcommand*{\acrnolinkfootnote}[3]{%
```

```
7081 \footnote{#3}%
```

```
7082 }
```

acronymDisplayStyle Sets the acronym display style for given glossary for the description and footnote combination.

```
7083 \newcommand*{\SetDescriptionFootnoteAcronymDisplayStyle}[1]{%
```

```
7084 \defglsentryfmt[#1]{%
```

```
7085 \ifdefempty\glscustomtext
```

```
7086 {%
```

```
7087 \ifglsused{\glslabel}%
```

```

7088     {%
7089         \acronymfont{\glsgenentryfmt}%
7090     }%
7091     {%
7092         \firstacronymfont{\glsgenentryfmt}%
7093         \ifglshassymbol{\glslabel}%
7094     {%
7095         \expandafter\protect\expandafter\acrfootnote\expandafter
7096         {\@gls@link@opts}{\@gls@link@label}%
7097     {%
7098         \glsifplural
7099         {\glsentrysymbolplural{\glslabel}}%
7100         {\glsentrysymbol{\glslabel}}%
7101     }%
7102     }%
7103     }%
7104     }%
7105     {\glscustomtext\glsinsert}%
7106 }%
7107 }

```

teNewAcronymDef

```

7108 \newcommand*{\DescriptionFootnoteNewAcronymDef}{%
7109     \edef\@do@newglossaryentry{%
7110         \noexpand\newglossaryentry{\the\glslabeltok}%
7111         {%
7112             type=\acronymtype,%
7113             name={\noexpand\acronymfont{\the\glsshorttok}},%
7114             sort={\the\glsshorttok},%
7115             first={\the\glsshorttok},%
7116             firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7117             text={\the\glsshorttok},%
7118             plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7119             short={\the\glsshorttok},%
7120             shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7121             long={\the\glslongtok},%
7122             longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7123             symbol={\the\glslongtok},%
7124             symbolplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7125             \the\glskeylisttok
7126         }%
7127     }%
7128     \let\@org@gls@assign@firstpl\gls@assign@firstpl
7129     \let\@org@gls@assign@plural\gls@assign@plural
7130     \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7131     \def\gls@assign@firstpl##1##2{%
7132         \@gls@expand@field{##1}{firstpl}{##2}%
7133     }%
7134     \def\gls@assign@plural##1##2{%

```

```

7135 \@@gls@expand@field{##1}{plural}{##2}%
7136 }%
7137 \def\gls@assign@symbolplural##1##2{%
7138 \@@gls@expand@field{##1}{symbolplural}{##2}%
7139 }%
7140 \do@newglossaryentry
7141 \let\gls@assign@plural\@org@gls@assign@plural
7142 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7143 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7144 }

```

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.

```

7145 \newcommand*\SetDescriptionFootnoteAcronymStyle{%
7146 \renewcommand{\newacronym}[4][]{%
7147 \ifx\@glsacronymlists\@empty
7148 \def\@glo@type{\acronymtype}%
7149 \setkeys{glossentry}{##1}%
7150 \DeclareAcronymList{\@glo@type}%
7151 \SetDescriptionFootnoteAcronymDisplayStyle{\@glo@type}%
7152 \fi
7153 \glskeylisttok{##1}%
7154 \glslabeltok{##2}%
7155 \glsshorttok{##3}%
7156 \glslongtok{##4}%
7157 \newacronymhook
7158 \DescriptionFootnoteNewAcronymDef
7159 }%

```

If footnote package option is specified, set the first use to append the long form (stored in symbol) as a footnote.

```

7160 \@for\@gls@type:=\@glsacronymlists\do{%
7161 \SetDescriptionFootnoteAcronymDisplayStyle{\@gls@type}%
7162 }%

```

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.

```

7163 \ifglsacrsmallcaps
7164 \renewcommand*\acronymfont[1]{\textsc{##1}}%
7165 \renewcommand*\acrpluralsuffix{\glsupacrpluralsuffix}%
7166 \else
7167 \ifglsacrsmaller
7168 \renewcommand*\acronymfont[1]{\textsmaller{##1}}%
7169 \fi
7170 \fi

```

Check for package option clash

```

7171 \ifglsacrdua
7172   \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’
7173     can’t both be set}{}%
7174 \fi
7175 }%

```

nymDisplayStyle Sets the acronym display style for given glossary with description and dua combination.

```

7176 \newcommand*{\SetDescriptionDUAAcronymDisplayStyle}[1]{%
7177   \defglsentryfmt[#1]{\glsentryfmt}%
7178 }

```

UANewAcronymDef

```

7179 \newcommand*{\DescriptionDUANewAcronymDef}{%
7180   \edef\@do@newglossaryentry{%
7181     \noexpand\newglossaryentry{\the\glslabeltok}%
7182     {%
7183       type=\acronymtype,%
7184       name={\the\glslongtok},%
7185       sort={\the\glslongtok},%
7186       text={\the\glslongtok},%
7187       first={\the\glslongtok},%
7188       plural={\noexpand\expandonce\noexpand\@glo@longpl},%
7189       firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7190       short={\the\glsshorttok},%
7191       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7192       long={\the\glslongtok},%
7193       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7194       symbol={\the\glsshorttok},%
7195       symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7196       \the\glskeylisttok
7197     }%
7198   }%
7199   \let\@org@gls@assign@firstpl\gls@assign@firstpl
7200   \let\@org@gls@assign@plural\gls@assign@plural
7201   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7202   \def\gls@assign@firstpl##1##2{%
7203     \@gls@expand@field{##1}{firstpl}{##2}%
7204   }%
7205   \def\gls@assign@plural##1##2{%
7206     \@gls@expand@field{##1}{plural}{##2}%
7207   }%
7208   \def\gls@assign@symbolplural##1##2{%
7209     \@gls@expand@field{##1}{symbolplural}{##2}%
7210   }%
7211   \@do@newglossaryentry
7212   \let\gls@assign@firstpl\@org@gls@assign@firstpl
7213   \let\gls@assign@plural\@org@gls@assign@plural
7214   \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7215 }

```

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.

```

7216 \newcommand*{\SetDescriptionDUAACronymStyle}{%
7217   \ifglsmallcaps
7218     \PackageError{glossaries}{Option clash: 'smallcaps' and 'dua'
7219       can't both be set}{}%
7220   \else
7221     \ifglsmaller
7222       \PackageError{glossaries}{Option clash: 'smaller' and 'dua'
7223         can't both be set}{}%
7224     \fi
7225   \fi
7226   \renewcommand{\newacronym}[4][{}]{%
7227     \ifx\@glsacronymlists\@empty
7228       \def\@glo@type{\acronymtype}%
7229       \setkeys{glossentry}{##1}%
7230       \DeclareAcronymList{\@glo@type}%
7231       \SetDescriptionDUAACronymDisplayStyle{\@glo@type}%
7232     \fi
7233     \glskeylisttok{##1}%
7234     \glslabeltok{##2}%
7235     \glsshorttok{##3}%
7236     \glslongtok{##4}%
7237     \newacronymhook
7238     \DescriptionDUANewAcronymDef
7239   }%

```

Set display.

```

7240   \@for\@gls@type:=\@glsacronymlists\do{%
7241     \SetDescriptionDUAACronymDisplayStyle{\@gls@type}%
7242   }%
7243 }%

```

nymDisplayStyle Sets the acronym display style for given glossary using the description setting (but not footnote or dua).

```

7244 \newcommand*{\SetDescriptionAcronymDisplayStyle}[1]{%
7245   \defglentryfmt[#1]{%

7246     \ifdefempty\glscustomtext
7247     {%
7248       \ifglused{\glslabel}%
7249     }%

```

Move the inserted text outside of \acronymfont

```

7250       \let\gls@org@insert\glsinsert
7251       \let\glsinsert\@empty
7252       \acronymfont{\glsgenentryfmt}\gls@org@insert
7253     }%

```

```

7254     {%
7255         \glsgenentryfmt
7256         \ifglshassymbol{\glslabel}%
7257         {%
7258             \glsifplural
7259             {%
7260                 \def\@glo@symbol{\glsentrysymbolplural{\glslabel}}%
7261             }%
7262             {%
7263                 \def\@glo@symbol{\glsentrysymbol{\glslabel}}%
7264             }%
7265             \space(\protect\firstacronymfont
7266             {\glscapscase
7267              {\@glo@symbol}
7268              {\@glo@symbol}
7269              {\mfirstucMakeUppercase{\@glo@symbol}}})%
7270         }%
7271     }%
7272 }%
7273 }%
7274 {\glscustomtext\glsinsert}%
7275 }%
7276 }

```

onNewAcronymDef

```

7277 \newcommand*{\DescriptionNewAcronymDef}{%
7278     \edef\@do@newglossaryentry{%
7279         \noexpand\newglossaryentry{\the\glslabeltok}%
7280         {%
7281             type=\acronymtype,%
7282             name={\noexpand
7283                 \acrnameformat{\the\glsshorttok}{\the\glslongtok}},%
7284             sort={\the\glsshorttok},%
7285             first={\the\glslongtok},%
7286             firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7287             text={\the\glsshorttok},%
7288             plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7289             short={\the\glsshorttok},%
7290             shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7291             long={\the\glslongtok},%
7292             longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7293             symbol={\noexpand\@glo@text},%
7294             symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7295             \the\glskeylisttok}%
7296     }%
7297     \let\@org@gls@assign@firstpl\gls@assign@firstpl
7298     \let\@org@gls@assign@plural\gls@assign@plural
7299     \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7300     \def\gls@assign@firstpl##1##2{%

```

```

7301 \@@gls@expand@field{##1}{firstpl}{##2}%
7302 }%
7303 \def\gls@assign@plural##1##2{%
7304 \@@gls@expand@field{##1}{plural}{##2}%
7305 }%
7306 \def\gls@assign@symbolplural##1##2{%
7307 \@@gls@expand@field{##1}{symbolplural}{##2}%
7308 }%
7309 \do@newglossaryentry
7310 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7311 \let\gls@assign@plural\@org@gls@assign@plural
7312 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7313 }

```

ionAcronymStyle Option description is used, but not dua or footnote. Store long form in first key and short form in text and symbol key. The name is stored using \acrnameformat to allow the user to override the way the name is displayed in the list of acronyms.

```

7314 \newcommand*{\SetDescriptionAcronymStyle}{%
7315 \renewcommand{\newacronym}[4][\]{%
7316 \ifx\@glsacronymlists\@empty
7317 \def\@glo@type{\acronymtype}%
7318 \setkeys{glossentry}{##1}%
7319 \DeclareAcronymList{\@glo@type}%
7320 \SetDescriptionAcronymDisplayStyle{\@glo@type}%
7321 \fi
7322 \glskeylisttok{##1}%
7323 \glslabeltok{##2}%
7324 \glsshorttok{##3}%
7325 \glslongtok{##4}%
7326 \newacronymhook
7327 \DescriptionNewAcronymDef
7328 }%

```

Set display.

```

7329 \@for\@gls@type:=\@glsacronymlists\do{%
7330 \SetDescriptionAcronymDisplayStyle{\@gls@type}%
7331 }%

```

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.

```

7332 \ifglsacrsmallcaps
7333 \renewcommand{\acronymfont}[1]{\textsc{##1}}
7334 \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7335 \else
7336 \ifglsacrsmaller
7337 \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
7338 \fi
7339 \fi
7340 }%

```

nymDisplayStyle Sets the acronym display style for given glossary with footnote setting (but not description or dua).

```

7341 \newcommand*{\SetFootnoteAcronymDisplayStyle}[1]{%
7342   \defglentryfmt[#1]{%

7343     \ifdefempty\glscustomtext
7344     {%

      Move the inserted text outside of \acronymfont

7345       \let\gls@org@insert\glsinsert
7346       \let\glsinsert\@empty
7347       \ifglused{\glslabel}%
7348       {%
7349         \acronymfont{\glsgenentryfmt}\gls@org@insert
7350       }%
7351       {%
7352         \firstacronymfont{\glsgenentryfmt}\gls@org@insert
7353         \ifglshaslong{\glslabel}%
7354         {%
7355           \expandafter\protect\expandafter\acrfootnote\expandafter
7356           {\@gls@link@opts}{\@gls@link@label}%
7357           {%
7358             \glsifplural
7359             {\glsentrylongpl{\glslabel}}%
7360             {\glsentrylong{\glslabel}}%
7361           }%
7362         }%

7363       }%
7364     }%
7365   }%
7366   {\glscustomtext\glsinsert}%
7367 }%
7368 }
```

teNewAcronymDef

```

7369 \newcommand*{\FootnoteNewAcronymDef}{%
7370   \edef\@do@newglossaryentry{%
7371     \noexpand\newglossaryentry{\the\glslabeltok}%
7372     {%
7373       type=\acronymtype,%
7374       name={\noexpand\acronymfont{\the\glsshorttok}},%
7375       sort={\the\glsshorttok},%
7376       text={\the\glsshorttok},%
7377       plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7378       first={\the\glsshorttok},%
7379       firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7380       short={\the\glsshorttok},%
7381       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7382       long={\the\glslongtok},%
```



```

7383     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7384     description={\the\glslongtok},%
7385     descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7386     \the\glskeylisttok
7387   }%
7388 }%
7389 \let\@org@gls@assign@plural\gls@assign@plural
7390 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7391 \let\@org@gls@assign@descplural\gls@assign@descplural
7392 \def\gls@assign@firstpl##1##2{%
7393   \@gls@expand@field{##1}{firstpl}{##2}%
7394 }%
7395 \def\gls@assign@plural##1##2{%
7396   \@gls@expand@field{##1}{plural}{##2}%
7397 }%
7398 \def\gls@assign@descplural##1##2{%
7399   \@gls@expand@field{##1}{descplural}{##2}%
7400 }%
7401 \do@newglossaryentry
7402 \let\gls@assign@plural\@org@gls@assign@plural
7403 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7404 \let\gls@assign@descplural\@org@gls@assign@descplural
7405 }

```

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.

```

7406 \newcommand*{\SetFootnoteAcronymStyle}{%
7407   \renewcommand{\newacronym}[4][]{%
7408     \ifx\@glsacronymlists\@empty
7409       \def\@glo@type{\acronymtype}%
7410       \setkeys{glossentry}{##1}%
7411       \DeclareAcronymList{\@glo@type}%
7412       \SetFootnoteAcronymDisplayStyle{\@glo@type}%
7413     \fi
7414     \glskeylisttok{##1}%
7415     \glslabeltok{##2}%
7416     \glsshorttok{##3}%
7417     \glslongtok{##4}%
7418     \newacronymhook
7419     \FootnoteNewAcronymDef
7420   }%

```

Set display

```

7421   \@for\@gls@type:=\@glsacronymlists\do{%
7422     \SetFootnoteAcronymDisplayStyle{\@gls@type}%
7423   }%

```

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.

```

7424   \ifglsacrsmallcaps

```

```

7425     \renewcommand*{\acronymfont}[1]{\textsc{##1}}%
7426     \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7427 \else
7428     \ifglssacrsmaller
7429         \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
7430     \fi
7431 \fi

    Check for option clash
7432 \ifglssacrdua
7433     \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’
7434         can’t both be set}{}%
7435 \fi
7436 }%

```

`\doparenifnotempty` 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.

```

7437 \DeclareRobustCommand*{\glsdoparenifnotempty}[2]{%
7438     \protected@edef\gls@tmp{#1}%
7439     \ifdefempty\gls@tmp
7440     {}%
7441     {%
7442         \ifx\gls@tmp\gls@default@value
7443         \else
7444             \space (#2{#1})%
7445         \fi
7446     }%
7447 }

```

`\setacronymdisplaystyle` Sets the acronym display style for given glossary where neither footnote nor description is required, but smallcaps or smaller specified.

```

7448 \newcommand*{\SetSmallAcronymDisplayStyle}[1]{%
7449     \defglsenentryfmt[#1]{%

7450     \ifdefempty\glscustomtext
7451     {%

```

Move the inserted text outside of `\acronymfont`

```

7452         \let\gls@org@insert\glsinsert
7453         \let\glsinsert\@empty
7454         \ifglssused{\glslabel}%
7455         {%
7456             \acronymfont{\glsgenentryfmt}\gls@org@insert
7457         }%
7458         {%
7459             \glsgenentryfmt
7460             \ifglshassymbol{\glslabel}%
7461             {%
7462                 \glsifplural
7463                 {%

```

```

7464         \def\@glo@symbol{\glentrysymbolplural{\glslabel}}%
7465     }%
7466     {%
7467         \def\@glo@symbol{\glentrysymbol{\glslabel}}%
7468     }%
7469     \space
7470     (\glscapscase
7471     {\firstacronymfont{\@glo@symbol}}%
7472     {\firstacronymfont{\@glo@symbol}}%
7473     {\firstacronymfont{\mfirstucMakeUppercase{\@glo@symbol}}})%
7474 }%
7475 {}%
7476 }%
7477 }%
7478 {\glscustomtext\glsinsert}%
7479 }%
7480 }

```

11NewAcronymDef

```

7481 \newcommand*{\SmallNewAcronymDef}{%
7482     \edef\@do@newglossaryentry{%
7483         \noexpand\newglossaryentry{\the\glslabeltok}%
7484         {%
7485             type=\acronymtype,%
7486             name={\noexpand\acronymfont{\the\glsshorttok}},%
7487             sort={\the\glsshorttok},%
7488             text={\the\glsshorttok},%
7489             plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7490             first={\the\glslongtok},%
7491             firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7492             short={\the\glsshorttok},%
7493             shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7494             long={\the\glslongtok},%
7495             longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7496             description={\noexpand\@glo@first},%
7497             descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7498             symbol={\the\glsshorttok},%
7499             symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7500             \the\glskeylisttok
7501         }%
7502     }%
7503     \let\@org@gls@assign@firstpl\gls@assign@firstpl
7504     \let\@org@gls@assign@plural\gls@assign@plural
7505     \let\@org@gls@assign@descplural\gls@assign@descplural

```

```

7506 \let\org@glsgls@assign@symbolplural\glsgls@assign@symbolplural
7507 \def\glsgls@assign@firstpl##1##2{%
7508   \@glsgls@expand@field{##1}{firstpl}{##2}%
7509 }%
7510 \def\glsgls@assign@plural##1##2{%
7511   \@glsgls@expand@field{##1}{plural}{##2}%
7512 }%
7513 \def\glsgls@assign@descplural##1##2{%
7514   \@glsgls@expand@field{##1}{descplural}{##2}%
7515 }%
7516 \def\glsgls@assign@symbolplural##1##2{%
7517   \@glsgls@expand@field{##1}{symbolplural}{##2}%
7518 }%
7519 \do@newglossaryentry
7520 \let\glsgls@assign@firstpl\org@glsgls@assign@firstpl
7521 \let\glsgls@assign@plural\org@glsgls@assign@plural
7522 \let\glsgls@assign@descplural\org@glsgls@assign@descplural
7523 \let\glsgls@assign@symbolplural\org@glsgls@assign@symbolplural
7524 }

```

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.

```

7525 \newcommand*{\SetSmallAcronymStyle}{%
7526   \renewcommand{\newacronym}[4][]{%
7527     \ifx\org@glsgls@acronymlists\@empty
7528       \def\org@type{\acronymtype}%
7529       \setkeys{glossentry}{##1}%
7530       \DeclareAcronymList{\org@type}%
7531       \SetSmallAcronymDisplayStyle{\org@type}%
7532     \fi
7533     \glskeylisttok{##1}%
7534     \glslabeltok{##2}%
7535     \glsshorttok{##3}%
7536     \glslongtok{##4}%
7537     \newacronymhook
7538     \SmallNewAcronymDef
7539   }%

```

Change the display since first only contains long form.

```

7540 \@for\org@type:=\org@glsgls@acronymlists\do{%
7541   \SetSmallAcronymDisplayStyle{\org@type}%
7542 }%

```

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.

```

7543 \ifglsglsacrsmallcaps
7544   \renewcommand*{\acronymfont}[1]{\textsc{##1}}
7545   \renewcommand*{\acrpluralsuffix}{\glsglsupacrpluralsuffix}%
7546 \else
7547   \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}

```

```

7548 \fi
      check for option clash
7549 \ifglsacrdua
7550   \ifglsacrsmallcaps
7551     \PackageError{glossaries}{Option clash: ‘smallcaps’ and ‘dua’
7552       can’t both be set}{}%
7553   \else
7554     \PackageError{glossaries}{Option clash: ‘smaller’ and ‘dua’
7555       can’t both be set}{}%
7556   \fi
7557 \fi
7558 }%

```

DUADisplayStyle Sets the acronym display style for given glossary with dua setting.

```

7559 \newcommand*{\SetDUADisplayStyle}[1]{%
7560   \defglsentryfmt[#1]{\glsentryfmt}%
7561 }

```

UANewAcronymDef

```

7562 \newcommand*{\DUANewAcronymDef}{%
7563   \edef\@do@newglossaryentry{%
7564     \noexpand\newglossaryentry{\the\glslabeltok}%
7565     {%
7566       type=\acronymtype,%
7567       name={\the\glsshorttok},%
7568       text={\the\glslongtok},%
7569       first={\the\glslongtok},%
7570       plural={\noexpand\expandonce\noexpand\@glo@longpl},%
7571       firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7572       short={\the\glsshorttok},%
7573       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7574       long={\the\glslongtok},%
7575       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7576       description={\the\glslongtok},%
7577       descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7578       symbol={\the\glsshorttok},%
7579       symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7580       \the\glskeylisttok
7581     }%
7582   }%
7583   \let\@org@gls@assign@firstpl\gls@assign@firstpl
7584   \let\@org@gls@assign@plural\gls@assign@plural
7585   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7586   \let\@org@gls@assign@descplural\gls@assign@descplural
7587   \def\gls@assign@firstpl##1##2{%
7588     \@@gls@expand@field{##1}{firstpl}{##2}%
7589   }%
7590   \def\gls@assign@plural##1##2{%
7591     \@@gls@expand@field{##1}{plural}{##2}%

```

```

7592 }%
7593 \def\gls@assign@symbolplural##1##2{%
7594   \@gls@expand@field{##1}{symbolplural}{##2}%
7595 }%
7596 \def\gls@assign@descplural##1##2{%
7597   \@gls@expand@field{##1}{descplural}{##2}%
7598 }%
7599 \do@newglossaryentry
7600 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7601 \let\gls@assign@plural\@org@gls@assign@plural
7602 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7603 \let\gls@assign@descplural\@org@gls@assign@descplural
7604 }

```

\SetDUASStyle Always expand acronyms.

```

7605 \newcommand*{\SetDUASStyle}{%
7606   \renewcommand{\newacronym}[4][]{%
7607     \ifx\@glsacronymlists\@empty
7608       \def\@glo@type{\acronymtype}%
7609       \setkeys{glossentry}{##1}%
7610       \DeclareAcronymList{\@glo@type}%
7611       \SetDUADisplayStyle{\@glo@type}%
7612     \fi
7613     \glskeylisttok{##1}%
7614     \glslabeltok{##2}%
7615     \glsshorttok{##3}%
7616     \glslongtok{##4}%
7617     \newacronymhook
7618     \DUANewAcronymDef
7619   }%
7620   \Set the display
7621   \@for\@gls@type:=\@glsacronymlists\do{%
7622     \SetDUADisplayStyle{\@gls@type}%
7623 }%

```

SetAcronymStyle

```

7624 \newcommand*{\SetAcronymStyle}{%
7625   \SetDefaultAcronymStyle
7626   \ifglsacrdescription
7627     \ifglsacrfootnote
7628       \SetDescriptionFootnoteAcronymStyle
7629     \else
7630       \ifglsacrdua
7631         \SetDescriptionDUAAcronymStyle
7632       \else
7633         \SetDescriptionAcronymStyle
7634       \fi
7635     \fi

```

```

7636 \else
7637   \ifglsacrfootnote
7638     \SetFootnoteAcronymStyle
7639   \else
7640     \ifthenelse{\boolean{glsacrsmallcaps}}\OR
7641       \boolean{glsacrsmaller}}}%
7642   {%
7643     \SetSmallAcronymStyle
7644   }%
7645   {%
7646     \ifglsacrdua
7647       \SetDUASStyle
7648     \fi
7649   }%
7650 \fi
7651 \fi
7652 }

```

Set the acronym style according to the package options

```
7653 \SetAcronymStyle
```

Allow user to define their own custom acronyms. (For compatibility with versions before v3.0, the short form is stored in the user1 key, the plural short form is stored in the user2 key, the long form is stored in the user3 key and the plural long form is stored in the user4 key.) Defaults to displaying only the acronym with the long form as the description.

`\setacronymdisplaystyle` Sets the acronym display style.

```

7654 \newcommand*{\SetCustomDisplayStyle}[1]{%
7655   \defglsentryfmt[#1]{\glsentryfmt}%
7656 }

```

`\setacronymfields`

```

7657 \newcommand*{\CustomAcronymFields}{%
7658   name={\the\glsshorttok},%
7659   description={\the\glslongtok},%
7660   first={\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
7661   firstplural={\acrfullformat
7662     {\noexpand\glsentrylongpl{\the\glslabeltok}}}%
7663     {\noexpand\glsentryshortpl{\the\glslabeltok}}},%

7664   text={\the\glsshorttok},%
7665   plural={\the\glsshorttok\noexpand\acrpluralsuffix}%
7666 }

```

`\setnewacronymdef`

```

7667 \newcommand*{\CustomNewAcronymDef}{%
7668   \protected@edef\@do@newglossaryentry{%
7669     \noexpand\newglossaryentry{\the\glslabeltok}%
7670     {%

```

```

7671     type=\acronymtype,%
7672     short={\the\glsshorttok},%
7673     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7674     long={\the\glslongtok},%
7675     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7676     user1={\the\glsshorttok},%
7677     user2={\the\glsshorttok\noexpand\acrpluralsuffix},%
7678     user3={\the\glslongtok},%
7679     user4={\the\glslongtok\noexpand\acrpluralsuffix},%
7680     \CustomAcronymFields,%
7681     \the\glskeylisttok
7682   }%
7683 }%
7684 \@do@newglossaryentry
7685 }

```

\SetCustomStyle

```

7686 \newcommand*{\SetCustomStyle}{%
7687   \renewcommand{\newacronym}[4][]{%
7688     \ifx\@glsacronymlists\@empty
7689       \def\@gls@type{\acronymtype}%
7690       \setkeys{glossentry}{##1}%
7691       \DeclareAcronymList{\@gls@type}%
7692       \SetCustomDisplayStyle{\@gls@type}%
7693     \fi
7694     \glskeylisttok{##1}%
7695     \glslabeltok{##2}%
7696     \glsshorttok{##3}%
7697     \glslongtok{##4}%
7698     \newacronymhook
7699     \CustomNewAcronymDef
7700   }%
7701   Set the display
7702   \@for\@gls@type:=\@glsacronymlists\do{%
7703     \SetCustomDisplayStyle{\@gls@type}%
7704   }%
7705 }

```

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.

```
7705 \RequirePackage{glossary-hypernav}
```

The styles that use list-like environments. These are not loaded if the nolist option is used:

```
7706 \@gls@loadlist
```


The styles that use the longtable environment. These are not loaded if the nolong package option is used.

```
7707 \@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.

```
7708 \@gls@loadsuper
```

The tree-like styles. These are not loaded if the notree package option is used.

```
7709 \@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.

```
7710 \ifx\@glossary@default@style\relax
```

```
7711 \else
```

```
7712   \setglossarystyle{\@glossary@default@style}
```

```
7713 \fi
```

1.20 Debugging Commands

```
\showgloparent \showgloparent{\label{}}
```

```
7714 \newcommand*{\showgloparent}[1]{%
```

```
7715   \expandafter\show\csname glo@\glsdetoklabel{#1}@parent\endcsname
```

```
7716 }
```

```
\showglolevel \showglolevel{\label{}}
```

```
7717 \newcommand*{\showglolevel}[1]{%
```

```
7718   \expandafter\show\csname glo@\glsdetoklabel{#1}@level\endcsname
```

```
7719 }
```

```
\showglotext \showglotext{\label{}}
```

```
7720 \newcommand*{\showglotext}[1]{%
```

```
7721   \expandafter\show\csname glo@\glsdetoklabel{#1}@text\endcsname
```

```
7722 }
```

```
\showgloplural \showgloplural{\label{}}
```

```

7723 \newcommand*{\showgloplural}[1]{%
7724   \expandafter\show\csname glo@\glsdetoklabel{#1}@plural\endcsname
7725 }

```

\showglofirst \showglofirst{<label>}

```

7726 \newcommand*{\showglofirst}[1]{%
7727   \expandafter\show\csname glo@\glsdetoklabel{#1}@first\endcsname
7728 }

```

\showglofirstpl \showglofirstpl{<label>}

```

7729 \newcommand*{\showglofirstpl}[1]{%
7730   \expandafter\show\csname glo@\glsdetoklabel{#1}@firstpl\endcsname
7731 }

```

\showgloftype \showgloftype{<label>}

```

7732 \newcommand*{\showgloftype}[1]{%
7733   \expandafter\show\csname glo@\glsdetoklabel{#1}@type\endcsname
7734 }

```

\showglocounter \showglocounter{<label>}

```

7735 \newcommand*{\showglocounter}[1]{%
7736   \expandafter\show\csname glo@\glsdetoklabel{#1}@counter\endcsname
7737 }

```

\showglouser \showglouser{<label>}

```

7738 \newcommand*{\showglouser}[1]{%
7739   \expandafter\show\csname glo@\glsdetoklabel{#1}@user\endcsname
7740 }

```

\showglouserii \showglouserii{<label>}

```

7741 \newcommand*{\showglouserii}[1]{%
7742   \expandafter\show\csname glo@glstdetoklabel{#1}@userii\endcsname
7743 }

```

\showglouseriii \showglouseriii{<label>}

```

7744 \newcommand*{\showglouseriii}[1]{%
7745   \expandafter\show\csname glo@glstdetoklabel{#1}@useriii\endcsname
7746 }

```

\showglouseriv \showglouseriv{<label>}

```

7747 \newcommand*{\showglouseriv}[1]{%
7748   \expandafter\show\csname glo@glstdetoklabel{#1}@useriv\endcsname
7749 }

```

\showglouserv \showglouserv{<label>}

```

7750 \newcommand*{\showglouserv}[1]{%
7751   \expandafter\show\csname glo@glstdetoklabel{#1}@userv\endcsname
7752 }

```

\showglouservi \showglouservi{<label>}

```

7753 \newcommand*{\showglouservi}[1]{%
7754   \expandafter\show\csname glo@glstdetoklabel{#1}@uservi\endcsname
7755 }

```

\showgloname \showgloname{<label>}

```

7756 \newcommand*{\showgloname}[1]{%
7757   \expandafter\show\csname glo@glstdetoklabel{#1}@name\endcsname
7758 }

```

\showglodesc \showglodesc{<label>}

```

7759 \newcommand*{\showglodesc}[1]{%
7760   \expandafter\show\csname glo@\glsdetoklabel{#1}@desc\endcsname
7761 }

```

showglodescplural `\showglodescplural{<label>}`

```

7762 \newcommand*{\showglodescplural}[1]{%
7763   \expandafter\show\csname glo@\glsdetoklabel{#1}@descplural\endcsname
7764 }

```

\showglosort `\showglosort{<label>}`

```

7765 \newcommand*{\showglosort}[1]{%
7766   \expandafter\show\csname glo@\glsdetoklabel{#1}@sort\endcsname
7767 }

```

\showglosymbol `\showglosymbol{<label>}`

```

7768 \newcommand*{\showglosymbol}[1]{%
7769   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbol\endcsname
7770 }

```

showglosymbolplural `\showglosymbolplural{<label>}`

```

7771 \newcommand*{\showglosymbolplural}[1]{%
7772   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolplural\endcsname
7773 }

```

\showgloshort `\showgloshort{<label>}`

```

7774 \newcommand*{\showgloshort}[1]{%
7775   \expandafter\show\csname glo@\glsdetoklabel{#1}@short\endcsname
7776 }

```

\showglolong `\showglolong{<label>}`

```

7777 \newcommand*{\showglolong}[1]{%
7778   \expandafter\show\csname glo@\glsdetoklabel{#1}@long\endcsname
7779 }

```

\showgloindex \showgloindex{<label>}

```

7780 \newcommand*{\showgloindex}[1]{%
7781   \expandafter\show\csname glo@\glsdetoklabel{#1}@index\endcsname
7782 }

```

\showgloflag \showgloflag{<label>}

```

7783 \newcommand*{\showgloflag}[1]{%
7784   \expandafter\show\csname ifglo@\glsdetoklabel{#1}@flag\endcsname
7785 }

```

\showgloloclist \showgloloclist{<label>}

```

7786 \newcommand*{\showgloloclist}[1]{%
7787   \expandafter\show\csname glo@\glsdetoklabel{#1}@loclist\endcsname
7788 }

```

\showglofield \showglofield{<label>}{<field>}

```

7789 \newcommand*{\showglofield}[2]{%
7790   \csshow{glo@\glsdetoklabel{#1}@#2}%
7791 }

```

showacronymlists \showacronymlists

Show list of glossaries that have been flagged as a list of acronyms.

```

7792 \newcommand*{\showacronymlists}{%
7793   \show\@glsacronymlists
7794 }

```

\showglossaries \showglossaries

Show list of defined glossaries.

```

7795 \newcommand*{\showglossaries}{%

```

```

7796 \show\@glo@types
7797 }

```

`\showglossaryin` `\showglossaryin{<glossary-label>}`

Show the ‘in’ extension for the given glossary.

```

7798 \newcommand*{\showglossaryin}[1]{%
7799 \expandafter\show\csname @glotype@#1@in\endcsname
7800 }

```

`\showglossaryout` `\showglossaryout{<glossary-label>}`

Show the ‘out’ extension for the given glossary.

```

7801 \newcommand*{\showglossaryout}[1]{%
7802 \expandafter\show\csname @glotype@#1@out\endcsname
7803 }

```

`showglossarytitle` `\showglossarytitle{<glossary-label>}`

Show the title for the given glossary.

```

7804 \newcommand*{\showglossarytitle}[1]{%
7805 \expandafter\show\csname @glotype@#1@title\endcsname
7806 }

```

`wglossarycounter` `\showglossarycounter{<glossary-label>}`

Show the counter for the given glossary.

```

7807 \newcommand*{\showglossarycounter}[1]{%
7808 \expandafter\show\csname @glotype@#1@counter\endcsname
7809 }

```

`wglossaryentries` `\showglossaryentries{<glossary-label>}`

Show the list of entry labels for the given glossary.

```

7810 \newcommand*{\showglossaryentries}[1]{%
7811 \expandafter\show\csname glolist@#1\endcsname
7812 }

```

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.

```
7813 \csname ifglscpatible-2.07\endcsname
7814   \RequirePackage{glossaries-compatible-207}
7815 \fi
```

2 Prefix Support (glossaries-prefix Code)

This package provides a means of adding prefixes to your glossary entries. For example, you may want to use “`\gls{<label>}`” on first use but use “`\an\gls{<label>}`” on subsequent use.

```
7816 \NeedsTeXFormat{LaTeX2e}
```

```
7817 \ProvidesPackage{glossaries-prefix}[2017/08/10 v4.31 (NLCT)]
```

Pass all options to glossaries:

```
7818 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
7819 \ProcessOptions
```

Load glossaries:

```
7820 \RequirePackage{glossaries}
```

Add the new keys:

```
7821 \define@key{glossentry}{prefixfirst}{\def\@glo@entryprefixfirst{#1}}%
```

```
7822 \define@key{glossentry}{prefixfirstplural}{\def\@glo@entryprefixfirstplural{#1}}%
```

```
7823 \define@key{glossentry}{prefix}{\def\@glo@entryprefix{#1}}%
```

```
7824 \define@key{glossentry}{prefixplural}{\def\@glo@entryprefixplural{#1}}%
```

Add them to `\gls@keymap`:

```
7825 \appto\@gls@keymap{,%
```

```
7826   {prefixfirst}{prefixfirst},%
```

```
7827   {prefixfirstplural}{prefixfirstplural},%
```

```
7828   {prefix}{prefix},%
```

```
7829   {prefixplural}{prefixplural}}%
```

```
7830 }
```

Set the default values:

```
7831 \appto\@newglossaryentryprehook{%
```

```
7832   \def\@glo@entryprefix{}}%
```

```
7833   \def\@glo@entryprefixplural{}}%
```

```
7834   \let\@glo@entryprefixfirst\@gls@default@value
```

```
7835   \let\@glo@entryprefixfirstplural\@gls@default@value
```

```
7836 }
```

Set the assignment code:

```
7837 \appto\@newglossaryentryposthook{%
```

```
7838   \gls@assign@field{}{\@glo@label}{prefix}{\@glo@entryprefix}}%
```

```
7839   \gls@assign@field{}{\@glo@label}{prefixplural}{\@glo@entryprefixplural}}%
```

If `prefixfirst` has not been supplied, make it the same as `prefix`.

```
7840 \expandafter\gls@assign@field\expandafter
```

```
7841   {\csname glo@\@glo@label @prefix\endcsname}{\@glo@label}{prefixfirst}}%
```

```
7842   {\@glo@entryprefixfirst}}%
```


If prefixfirstplural has not been supplied, make it the same as prefixplural.

```

7843 \expandafter\gls@assign@field\expandafter
7844   {\csname glo@\@glo@label @prefixplural\endcsname}{\@glo@label}%
7845   {prefixfirstplural}{\@glo@entryprefixfirstplural}%
7846 }

```

Define commands to access these fields:

entryprefixfirst

```

7847 \newcommand*{\glsentryprefixfirst}[1]{\csuse{glo@#1@prefixfirst}}

```

entryfirstplural

```

7848 \newcommand*{\glsentryprefixfirstplural}[1]{\csuse{glo@#1@prefixfirstplural}}

```

\glsentryprefix

```

7849 \newcommand*{\glsentryprefix}[1]{\csuse{glo@#1@prefix}}

```

entryprefixplural

```

7850 \newcommand*{\glsentryprefixplural}[1]{\csuse{glo@#1@prefixplural}}

```

Now for the initial upper case variants:

entryprefixfirst

```

7851 \newrobustcmd*{\Glsentryprefixfirst}[1]{%
7852   \protected@edef\@glo@text{\csname glo@#1@prefixfirst\endcsname}%
7853   \xmakefirstuc\@glo@text
7854 }

```

entryfirstplural

```

7855 \newrobustcmd*{\Glsentryprefixfirstplural}[1]{%
7856   \protected@edef\@glo@text{\csname glo@#1@prefixfirstplural\endcsname}%
7857   \xmakefirstuc\@glo@text
7858 }

```

\Glsentryprefix

```

7859 \newrobustcmd*{\Glsentryprefix}[1]{%
7860   \protected@edef\@glo@text{\csname glo@#1@prefix\endcsname}%
7861   \xmakefirstuc\@glo@text
7862 }

```

entryprefixplural

```

7863 \newrobustcmd*{\Glsentryprefixplural}[1]{%
7864   \protected@edef\@glo@text{\csname glo@#1@prefixplural\endcsname}%
7865   \xmakefirstuc\@glo@text
7866 }

```

Define commands to determine if the prefix keys have been set:

\ifglshasprefix

```
7867 \newcommand*{\ifglshasprefix}[3]{%
7868   \ifcempty{glo@#1@prefix}%
7869   {#3}%
7870   {#2}%
7871 }
```

hasprefixplural

```
7872 \newcommand*{\ifglshasprefixplural}[3]{%
7873   \ifcempty{glo@#1@prefixplural}%
7874   {#3}%
7875   {#2}%
7876 }
```

shasprefixfirst

```
7877 \newcommand*{\ifglshasprefixfirst}[3]{%
7878   \ifcempty{glo@#1@prefixfirst}%
7879   {#3}%
7880   {#2}%
7881 }
```

efixfirstplural

```
7882 \newcommand*{\ifglshasprefixfirstplural}[3]{%
7883   \ifcempty{glo@#1@prefixfirstplural}%
7884   {#3}%
7885   {#2}%
7886 }
```

Define commands that insert the prefix before commands like \gls:

\pgls

```
7887 \newrobustcmd{\pgls}{\@gls@hyp@opt\@pgls}
```

\@pgls Unstarred version.

```
7888 \newcommand*{\@pgls}[2][ ]{%
7889   \new@ifnextchar[%
7890   {\@pgls@{#1}{#2}}%
7891   {\@pgls@{#1}{#2}[ ]}%
7892 }
```

\@pgls@ Read in the final optional argument:

```
7893 \def\@pgls@#1#2[#3]{%
7894   \glsdoifexists{#2}%
7895   {%
7896     \ifglsused{#2}%
7897     {%
7898       \glsentryprefix{#2}%
7899     }%

```

```

7900   {%
7901     \glsentryprefixfirst{#2}%
7902   }%
7903   \@gls@{#1}{#2}[#3]%
7904 }%
7905 }

```

Similarly for the plural version:

```

\pglsp1
7906 \newrobustcmd{\pglsp1}{\@gls@hyp@opt\@pglsp1}

```

\@pglsp1 Unstarred version.

```

7907 \newcommand*{\@pglsp1}[2][ ]{%
7908   \new@ifnextchar[%
7909     {\@pglsp1@{#1}{#2}}%
7910     {\@pglsp1@{#1}{#2}[ ]}%
7911 }

```

\@pglsp1@ Read in the final optional argument:

```

7912 \def\@pglsp1@#1#2[#3]{%
7913   \glsdoifexists{#2}%
7914   {%
7915     \ifglsused{#2}%
7916     {%
7917       \glsentryprefixplural{#2}%
7918     }%
7919     {%
7920       \glsentryprefixfirstplural{#2}%
7921     }%
7922     \@glspl@{#1}{#2}[#3]%
7923   }%
7924 }

```

Now for the first letter upper case versions:

```

\Pgls
7925 \newrobustcmd{\Pgls}{\@gls@hyp@opt\@Pgls}

```

\@Pgls Unstarred version.

```

7926 \newcommand*{\@Pgls}[2][ ]{%
7927   \new@ifnextchar[%
7928     {\@Pgls@{#1}{#2}}%
7929     {\@Pgls@{#1}{#2}[ ]}%
7930 }

```

\@Pgls@ Read in the final optional argument:

```

7931 \def\@Pgls@#1#2[#3]{%

```

```

7932 \glsdoifexists{#2}%
7933 {%
7934   \ifglsused{#2}%
7935   {%
7936     \ifglshasprefix{#2}%
7937     {%
7938       \Glsentryprefix{#2}%
7939       \@gls@{#1}{#2}[#3]%
7940     }%
7941     {\@Gls@{#1}{#2}[#3]}%
7942   }%
7943   {%
7944     \ifglshasprefixfirst{#2}%
7945     {%
7946       \Glsentryprefixfirst{#2}%
7947       \@gls@{#1}{#2}[#3]%
7948     }%
7949     {\@Gls@{#1}{#2}[#3]}%
7950   }%
7951 }%
7952 }

```

Similarly for the plural version:

```

\Pglspl
7953 \newrobustcmd{\Pglspl}{\@gls@hyp@opt\@Pglspl}

```

\@Pglspl Unstarred version.

```

7954 \newcommand*{\@Pglspl}[2] [] {%
7955   \new@ifnextchar[%
7956   {\@Pglspl@{#1}{#2}}%
7957   {\@Pglspl@{#1}{#2} []}%
7958 }

```

\@Pglspl@ Read in the final optional argument:

```

7959 \def\@Pglspl@#1#2[#3] {%
7960   \glsdoifexists{#2}%
7961   {%
7962     \ifglsused{#2}%
7963     {%
7964       \ifglshasprefixplural{#2}%
7965       {%
7966         \Glsentryprefixplural{#2}%
7967         \@glspl@{#1}{#2}[#3]%
7968       }%
7969       {\@Glspl@{#1}{#2}[#3]}%
7970     }%
7971     {%
7972       \ifglshasprefixfirstplural{#2}%

```

```

7973      {%
7974      \Glsentryprefixfirstplural{#2}%
7975      \@glsp1@{#1}{#2}[#3]%
7976      }%
7977      {\@Glspl@{#1}{#2}[#3]}%
7978      }%
7979      }%
7980 }

```

Finally the all upper case versions:

\PGLS

```

7981 \newrobustcmd{\PGLS}{\@gls@hyp@opt\PGLS}

```

\@PGLS Unstarred version.

```

7982 \newcommand*{\@PGLS}[2][{}]{%
7983   \new@ifnextchar[%
7984   {\@PGLS@{#1}{#2}}%
7985   {\@PGLS@{#1}{#2}[]}%
7986 }

```

\@PGLS@ Read in the final optional argument:

```

7987 \def\@PGLS@#1#2[#3]{%
7988   \glsdoifexists{#2}%
7989   {%
7990     \ifglsused{#2}%
7991     {%
7992       \mfirstucMakeUppercase{\glsentryprefix{#2}}%
7993     }%
7994     {%
7995       \mfirstucMakeUppercase{\glsentryprefixfirst{#2}}%
7996     }%
7997     \@GLS@{#1}{#2}[#3]%
7998   }%
7999 }

```

Plural version:

\PGLSp1

```

8000 \newrobustcmd{\PGLSp1}{\@gls@hyp@opt\PGLSp1}

```

\@PGLSp1 Unstarred version.

```

8001 \newcommand*{\@PGLSp1}[2][{}]{%
8002   \new@ifnextchar[%
8003   {\@PGLSp1@{#1}{#2}}%
8004   {\@PGLSp1@{#1}{#2}[]}%
8005 }

```

\@PGLSp1@ Read in the final optional argument:

```
8006 \def\@PGLSp1@#1#2[#3]{%
8007   \glsdoifexists{#2}%
8008   {%
8009     \ifglsused{#2}%
8010     {%
8011       \mfirstucMakeUppercase{\glsentryprefixplural{#2}}%
8012     }%
8013     {%
8014       \mfirstucMakeUppercase{\glsentryprefixfirstplural{#2}}%
8015     }%
8016     \@GLSp1@{#1}{#2}[#3]%
8017   }%
8018 }
```

3 Glossary Styles

3.1 Glossary hyper-navigation definitions (glossary-hypernav package)

Package Definition:

```
8019 \ProvidesPackage{glossary-hypernav}[2017/08/10 v4.31 (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`

```
8020 \newcommand*{\glsnavhyperlink}[3][\@glo@type]{%
8021   \edef\gls@grplabel{#2}\protected@edef\gls@grptitle{#3}%
8022   \@glslink{\glsnavhyperlinkname{#1}{#2}}{#3}}
```

`glsnavhyperlinkname` Expands to the hypertarget name. The first argument is the glossary type. The second argument is the group label.

```
8023 \newcommand*{\glsnavhyperlinkname}[2]{\glsn:#1@#2}
```

```
\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`

```
8024 \newcommand*{\glsnavhypertarget}[3][\@glo@type]{%
  Add this group to the aux file for re-run check.
8025   \protected@write\auxout{}{\string\gls@hypergroup{#1}{#2}}%
  Add the target.
8026   \@glstarget{\glsnavhyperlinkname{#1}{#2}}{#3}%
```

Check list of known groups to determine if a re-run is required.

```
8027 \expandafter\let
8028 \expandafter\@gls@list\csname @gls@hypergroup@list@#1\endcsname
```

Iterate through list and terminate loop if this group is found.

```
8029 \@for\@gls@elem:=\@gls@list\do{%
8030 \ifthenelse{\equal{\@gls@elem}{#2}}{\@endfortrue}{}}%
```

Check if list terminated prematurely.

```
8031 \if@endfor
8032 \else
```

This group was not included in the list, so issue a warning.

```
8033 \GlossariesWarningNoLine{Navigation panel
8034 for glossary type ‘#1’^^Jmissing group ‘#2’}%
8035 \gdef\gls@hypergroup@rerun{%
8036 \GlossariesWarningNoLine{Navigation panel
8037 has changed. Rerun LaTeX}}%
8038 \fi
8039 }
```

`hypergroup@rerun` Give a warning at the end if re-run required

```
8040 \let\gls@hypergroup@rerun\relax
8041 \AtEndDocument{\gls@hypergroup@rerun}
```

`@gls@hypergroup` This adds to (or creates) the command `\@gls@hypergroup@list@<glossary type>` which lists all groups for a given glossary, so that the navigation bar only contains those groups that are present. However it requires at least 2 runs to ensure the information is up-to-date.

```
8042 \newcommand*{\@gls@hypergroup}[2]{%
8043 \@ifundefined{\@gls@hypergroup@list@#1}{%
8044 \expandafter\xdef\csname @gls@hypergroup@list@#1\endcsname{#2}%
8045 }{%
8046 \expandafter\let\expandafter\@gls@tmp
8047 \csname @gls@hypergroup@list@#1\endcsname
8048 \expandafter\xdef\csname @gls@hypergroup@list@#1\endcsname{%
8049 \@gls@tmp,#2}%
8050 }%
8051 }
```

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`

```
8052 \newcommand*{\glsnavigation}{%
8053 \def\@gls@between{}%
8054 \ifcsundef{\@gls@hypergroup@list@\@glo@type}%

```



```

8055 {%
8056   \def\@gls@list{%
8057 }%
8058 {%
8059   \expandafter\let\expandafter\@gls@list
8060     \csname @gls@hypergroup@list@\@glo@type\endcsname
8061 }%
8062 \@for\@gls@tmp:=\@gls@list\do{%
8063   \@gls@between

8064   \@gls@getgrouptitle{\@gls@tmp}{\@gls@grptitle}%
8065   \glsnavhyperlink{\@gls@tmp}{\@gls@grptitle}%
8066   \let\@gls@between\glshypernavsep
8067 }%
8068 }

```

`\glshypernavsep` Separator for the hyper navigation bar.

```

8069 \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`

```

8070 \newcommand*{\glssymbolnav}{%
8071   \glsnavhyperlink{glsymbols}{\glsgetgrouptitle{glsymbols}}%
8072   \glshypernavsep
8073   \glsnavhyperlink{glsnumbers}{\glsgetgrouptitle{glsnumbers}}%
8074   \glshypernavsep
8075 }

```

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.

```

8076 \ProvidesPackage{glossary-inline}[2017/08/10 v4.31 (NLCT)]

```

`inline` Define the inline style.

```

8077 \newglossarystyle{inline}{%
  Start of glossary sets up first empty separator between entries. (This is then changed by
  \glossentry)
8078   \renewenvironment{theglossary}%
8079     {%
8080       \def\gls@inlinesep{%
8081         \def\gls@inlinesubsep{%
8082           \def\gls@inlinepostchild{%
8083             }%
8084             {\glspostinline}%

```

No header:

```
8085 \renewcommand*{\glossaryheader}{}%
```

No group headings (if heading is required, add `\glsinlinedopostchild` to start definition in case heading follows a child entry):

```
8086 \renewcommand*{\glsgroupheading}[1]{}%
```

Just display separator followed by name and description:

```
8087 \renewcommand{\glossentry}[2]{%
8088   \glsinlinedopostchild
8089   \gls@inlinesep
8090   \glsentryitem{##1}%
8091   \glsinlinenameformat{##1}{%
8092     \glossentryname{##1}%
8093   }%
8094   \ifglsdescsuppressed{##1}%
8095   {%
8096     \glsinlineemptydescformat
8097     {%
8098       \glossentrysymbol{##1}%
8099     }%
8100     {%
8101       ##2%
8102     }%
8103   }%
8104   {%
8105     \ifglshasdesc{##1}%
8106     {\glsinlinedescformat{\glossentrydesc{##1}}{\glossentrysymbol{##1}}{##2}}%
8107     {\glsinlineemptydescformat{\glossentrysymbol{##1}}{##2}}%
8108   }%
8109   \ifglshaschildren{##1}%
8110   {%
8111     \glsresetsubentrycounter
8112     \glsinlineparentchildseparator
8113     \def\gls@inlinesubsep{}%
8114     \def\gls@inlinepostchild{\glsinlinepostchild}%
8115   }%
8116   {}%
8117   \def\gls@inlinesep{\glsinlineseparator}%
8118 }%
```

Sub-entries display description:

```
8119 \renewcommand{\subglossentry}[3]{%
8120   \gls@inlinesubsep%
8121   \glsinlinesubnameformat{##2}{%
8122     \glossentryname{##2}%
8123   }%
8124   \glsinlinesubdescformat{\glossentrydesc{##2}}{\glossentrysymbol{##2}}{##3}%
8125   \def\gls@inlinesubsep{\glsinlinesubseparator}%
8126 }%
```

Nothing special between groups:

```
8127 \renewcommand*{\glsgroupskip}{}%
8128 }
```

linedopostchild

```
8129 \newcommand*{\glsinlinedopostchild}{%
8130     \gls@inlinepostchild
8131     \def\gls@inlinepostchild{}%
8132 }
```

inlineseparator Separator to use between entries.

```
8133 \newcommand*{\glsinlineseparator}{;\space}
```

inlinesubseparator Separator to use between sub-entries.

```
8134 \newcommand*{\glsinlinesubseparator}{,\space}
```

parentchildseparator Separator to use between parent and children.

```
8135 \newcommand*{\glsinlineparentchildseparator}{:\space}
```

inlinepostchild Hook to use between child and next entry

```
8136 \newcommand*{\glsinlinepostchild}{}%
```

\glspostinline Terminator for inline glossary.

```
8137 \newcommand*{\glspostinline}{\glspostdescription\space}
```

inlinenameformat Formats the name of the entry (first argument label, second argument name):

```
8138 \newcommand*{\glsinlinenameformat}[2]{\glstarget{#1}{#2}}
```

inlinedescformat Formats the entry's description, symbol and location list:

```
8139 \newcommand*{\glsinlinedescformat}[3]{\space#1}
```

emptydescformat Formats the entry's symbol and location list when the description is empty:

```
8140 \newcommand*{\glsinlineemptydescformat}[2]{}
```

inlinesubnameformat Formats the name of the subentry (first argument label, second argument name):

```
8141 \newcommand*{\glsinlinesubnameformat}[2]{\glstarget{#1}{}}
```

inlinesubdescformat Formats the subentry's description, symbol and location list:

```
8142 \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.

```
8143 \ProvidesPackage{glossary-list}[2017/08/10 v4.31 (NLCT)]
```

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```

8144 \providecommand{\indexspace}{%
8145   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
8146 }

```

`tgrouphdheaderfmt` Provide a way of adjusting the format of the group headings.

```

8147 \newcommand*{\glslistgrouphdheaderfmt}[1]{#1}

```

`tnavigationitem` Provide a way of adjusting the format of the navigation header. This puts the navigation line inside the optional argument of `item` to prevent unwanted space occurring at the start, but this can cause a problem if the navigation line is too long. With this command, it makes it easier for the user to customise the style without having to remember to modify `\glossaryheader` after the style has been set.

```

8148 \newcommand*{\glslistnavigationitem}[1]{\item[#1]}

```

`list` The list glossary style uses the description environment. The group separator `\glsgroupskip` is redefined as `\indexspace` which produces a gap between groups. The glossary heading and the group headings do nothing. Sub-entries immediately follow the main entry without the sub-entry name. This style does not use the entry's symbol. This is used as the default style for the glossaries package.

```

8149 \newglossarystyle{list}{%

```

Use description environment:

```

8150   \renewenvironment{theglossary}%
8151     {\begin{description}}{\end{description}}%

```

No header at the start of the environment:

```

8152   \renewcommand*{\glossaryheader}{}%

```

No group headings:

```

8153   \renewcommand*{\glsgroupheading}[1]{}%

```

Main (level 0) entries start a new item in the list:

```

8154   \renewcommand*{\glossentry}[2]{%
8155     \item[\glsentryitem{##1}%
8156       \glstarget{##1}{\glossentryname{##1}}]
8157     \glossentrydesc{##1}\glspostdescription\space ##2}%

```

Sub-entries continue on the same line:

```

8158   \renewcommand*{\subglossentry}[3]{%
8159     \glssubentryitem{##2}%
8160     \glstarget{##2}{\strut}\space
8161     \glossentrydesc{##2}\glspostdescription\space ##3.}%

```

Add vertical space between groups:

```

8162   \renewcommand*{\glsgroupskip}{\ifglslnogroupskip\else\indexspace\fi}%
8163 }

```

listgroup The listgroup style is like the list style, but the glossary groups have headings.

```
8164 \newglossarystyle{listgroup}{%  
    Base it on the list style:  
8165 \setglossarystyle{list}%  
    Each group has a heading:  
8166 \renewcommand*{\glsgroupheading}[1]{%  
8167 \item[\glslistgroupheaderfmt{\glsgrouptitle{##1}}]}
```

listhypergroup The listhypergroup style is like the listgroup style, but has a set of links to the groups at the start of the glossary.

```
8168 \newglossarystyle{listhypergroup}{%  
    Base it on the list style:  
8169 \setglossarystyle{list}%  
    Add navigation links at the start of the environment.  
8170 \renewcommand*{\glossaryheader}{%  
8171 \glslistnavigationitem{\glsnavigation}}%  
    Each group has a heading with a hypertarget:  
8172 \renewcommand*{\glsgroupheading}[1]{%  
8173 \item[\glslistgroupheaderfmt  
8174 {\glsnavigationhypertarget{##1}{\glsgrouptitle{##1}}}]}
```

altlist The altlist glossary style is like the list style, but places the description on a new line. Sub-entries follow in separate paragraphs without the sub-entry name. This style does not use the entry's symbol.

```
8175 \newglossarystyle{altlist}{%  
    Base it on the list style:  
8176 \setglossarystyle{list}%  
    Main (level 0) entries start a new item in the list with a line break after the entry name:  
8177 \renewcommand*{\glossentry}[2]{%  
8178 \item[\glssentryitem{##1}%  
8179 \glstarget{##1}{\glossentryname{##1}}}%  
    Version 3.04 changed \newline to the following paragraph break stuff (thanks to Daniel Geb-  
    hardt for supplying the fix) to prevent a page break occurring at this point.  
8180 \mbox{}\par\nobreak\@afterheading  
8181 \glossentrydesc{##1}\glspostdescription\space ##2}%  
    Sub-entries start a new paragraph:  
8182 \renewcommand{\subglossentry}[3]{%  
8183 \par  
8184 \glssubentryitem{##2}%  
8185 \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space ##3}%  
8186 }
```

altlistgroup The altlistgroup glossary style is like the altlist style, but the glossary groups have headings.

```
8187 \newglossarystyle{altlistgroup}{%  
    Base it on the altlist style:  
8188 \setglossarystyle{altlist}%  
    Each group has a heading:  
8189 \renewcommand*{\glsgroupheading}[1]{%  
8190 \item[\glslistgroupheaderfmt{\glsgrouptitle{##1}}]}
```

altlisthypergroup The altlisthypergroup glossary style is like the altlistgroup style, but has a set of links to the groups at the start of the glossary.

```
8191 \newglossarystyle{altlisthypergroup}{%  
    Base it on the altlist style:  
8192 \setglossarystyle{altlist}%  
    Add navigation links at the start of the environment.  
8193 \renewcommand*{\glossaryheader}{%  
8194 \glslistnavigationitem{\glsnavigation}}%  
    Each group has a heading with a hypertarget:  
8195 \renewcommand*{\glsgroupheading}[1]{%  
8196 \item[\glslistgroupheaderfmt  
8197 {\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}]}
```

listdotted The listdotted glossary style was supplied by Axel Menzel. I've modified it slightly so that the distance from the start of the name to the end of the dotted line is specified by `\glslistdottedwidth`. Note that this style ignores the page numbers as well as the symbol. Sub-entries are displayed in the same way as top-level entries.

```
8198 \newglossarystyle{listdotted}{%  
    Base it on the list style:  
8199 \setglossarystyle{list}%  
    Each main (level 0) entry starts a new item:  
8200 \renewcommand*{\glossentry}[2]{%  
8201 \item[]\makebox[\glslistdottedwidth][l]{%  
8202 \glssentryitem{##1}%  
8203 \glstarget{##1}{\glossentryname{##1}}%  
8204 \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##1}}%  
    Sub entries have the same format as main entries:  
8205 \renewcommand*{\subglossentry}[3]{%  
8206 \item[]\makebox[\glslistdottedwidth][l]{%  
8207 \glssubentryitem{##2}%  
8208 \glstarget{##2}{\glossentryname{##2}}%  
8209 \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##2}}%  
8210 }
```

listdottedwidth

```
8211 \newlength\glslistdottedwidth
8212 \setlength{\glslistdottedwidth}{.5\hsize}
```

sublistdotted This style is similar to the `glostylelistdotted` style, except that the main entries just have the name displayed.

```
8213 \newglossarystyle{sublistdotted}{%
    Base it on the listdotted style:
8214 \setglossarystyle{listdotted}%
    Main (level 0) entries just display the name:
8215 \renewcommand*{\glossentry}[2]{%
8216 \item[\glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}}]}%
8217 }
```

3.4 Glossary Styles using `longtable` (the `glossary-long` package)

The glossary styles defined in the package used the `longtable` environment in the glossary.

```
8218 \ProvidesPackage{glossary-long}[2017/08/10 v4.31 (NLCT)]
```

Requires the package:

```
8219 \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`.)

```
8220 \@ifundefined{glsdescwidth}{%
8221 \newlength\glsdescwidth
8222 \setlength{\glsdescwidth}{0.6\hsize}
8223 }{}
```

`\glspagelistwidth` This is a length that governs the width of the page list column.

```
8224 \@ifundefined{glspagelistwidth}{%
8225 \newlength\glspagelistwidth
8226 \setlength{\glspagelistwidth}{0.1\hsize}
8227 }{}
```

`long` The long glossary style command which uses the `longtable` environment:

```
8228 \newglossarystyle{long}{%
    Use longtable with two columns:
8229 \renewenvironment{theglossary}%
8230 {\begin{longtable}\lp{\glsdescwidth}}%
8231 {\end{longtable}}%
    Do nothing at the start of the environment:
8232 \renewcommand*{\glossaryheader}{}%
```

No heading between groups:

```
8233 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries displayed in a row:

```
8234 \renewcommand{\glossentry}[2]{%
8235   \glentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8236   \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline
8237 }%
```

Sub entries displayed on the following row without the name:

```
8238 \renewcommand{\subglossentry}[3]{%
8239   &
8240   \glssubentryitem{##2}%
8241   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
8242   ##3\tabularnewline
8243 }%
```

Blank row between groups: The check for nogroupskip must occur outside `\glsgroupskip`
(<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```
8244 \ifglsgroupskip
8245 \renewcommand*{\glsgroupskip}{}%
8246 \else
8247 \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
8248 \fi
8249 }
```

longborder The `longborder` style is like the above, but with horizontal and vertical lines:

```
8250 \newglossarystyle{longborder}{%
```

Base it on the `glostylelong` style:

```
8251 \setglossarystyle{long}%
```

Use `longtable` with two columns with vertical lines between each column:

```
8252 \renewenvironment{theglossary}{%
8253   \begin{longtable}{|l|p{\glstdescwidth}|}{\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
8254 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8255 }
```

longheader The `longheader` style is like the long style but with a header:

```
8256 \newglossarystyle{longheader}{%
```

Base it on the `glostylelong` style:

```
8257 \setglossarystyle{long}%
```

Set the table's header:

```
8258 \renewcommand*{\glossaryheader}{%
8259   \bfseries \entryname & \bfseries \descriptionname\tabularnewline\endhead}%
8260 }
```


ongheaderborder The longheaderborder style is like the long style but with a header and border:

```
8261 \newglossarystyle{longheaderborder}{%
```

Base it on the glostylelongborder style:

```
8262 \setglossarystyle{longborder}{%
```

Set the table's header and add horizontal line to table's foot:

```
8263 \renewcommand*{\glossaryheader}{%
8264 \hline\bfseries \entryname & \bfseries
8265 \descriptionname\tabularnewline\hline
8266 \endhead
8267 \hline\endfoot}%
8268 }
```

long3col The long3col style is like long but with 3 columns

```
8269 \newglossarystyle{long3col}{%
```

Use a longtable with 3 columns:

```
8270 \renewenvironment{theglossary}%
8271 {\begin{longtable}{lp{\glstdescwidth}p{\glspagelistwidth}}}%
8272 {\end{longtable}}%
```

No table header:

```
8273 \renewcommand*{\glossaryheader}{}%
```

No headings between groups:

```
8274 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
8275 \renewcommand{\glossentry}[2]{%
8276 \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8277 \glossentrydesc{##1} & ##2\tabularnewline
8278 }%
```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```
8279 \renewcommand{\subglossentry}[3]{%
8280 &
8281 \glssubentryitem{##2}%
8282 \glstarget{##2}{\strut}\glossentrydesc{##2} &
8283 ##3\tabularnewline
8284 }%
```

Blank row between groups: The check for nogroupskip must occur outside \glsgroupskip (<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```
8285 \ifglsnogroupskip
8286 \renewcommand*{\glsgroupskip}{}%
8287 \else
8288 \renewcommand*{\glsgroupskip}{ & & \tabularnewline}%
8289 \fi
8290 }
```

long3colborder The long3colborder style is like the long3col style but with a border:

```
8291 \newglossarystyle{long3colborder}{%
      Base it on the glostylelong3col style:
8292   \setglossarystyle{long3col}%
      Use a longtable with 3 columns with vertical lines around them:
8293   \renewenvironment{theglossary}%
8294     {\begin{longtable}{|l|p{\glstdescwidth}|p{\glspagelistwidth}|}%
8295     {\end{longtable}}}%
      Place horizontal lines at the head and foot of the table:
8296   \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8297 }
```

long3colheader The long3colheader style is like long3col but with a header row:

```
8298 \newglossarystyle{long3colheader}{%
      Base it on the glostylelong3col style:
8299   \setglossarystyle{long3col}%
      Set the table's header:
8300   \renewcommand*{\glossaryheader}{%
8301     \bfseries\entryname&\bfseries\descriptionname&
8302     \bfseries\pagelistname\tabularnewline\endhead}%
8303 }
```

colheaderborder The long3colheaderborder style is like the above but with a border

```
8304 \newglossarystyle{long3colheaderborder}{%
      Base it on the glostylelong3colborder style:
8305   \setglossarystyle{long3colborder}%
      Set the table's header and add horizontal line at table's foot:
8306   \renewcommand*{\glossaryheader}{%
8307     \hline
8308     \bfseries\entryname&\bfseries\descriptionname&
8309     \bfseries\pagelistname\tabularnewline\hline\endhead
8310     \hline\endfoot}%
8311 }
```

long4col The long4col style has four columns where the third column contains the value of the associated symbol key.

```
8312 \newglossarystyle{long4col}{%
      Use a longtable with 4 columns:
8313   \renewenvironment{theglossary}%
8314     {\begin{longtable}{l1l1l}%
8315     {\end{longtable}}}%
      No table header:
8316   \renewcommand*{\glossaryheader}{}%

```

No group headings:

```
8317 \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):

```
8318 \renewcommand{\glossentry}[2]{%
8319   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8320   \glossentrydesc{##1} &
8321   \glossentrysymbol{##1} &
8322   ##2\tabularnewline
8323 }%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
8324 \renewcommand{\subglossentry}[3]{%
8325   &
8326   \glssubentryitem{##2}%
8327   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8328   \glossentrysymbol{##2} & ##3\tabularnewline
8329 }%
```

Blank row between groups: The check for nogroupskip must occur outside \glsgroupskip (<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```
8330 \ifglsnogroupskip
8331 \renewcommand*{\glsgroupskip}{}%
8332 \else
8333 \renewcommand*{\glsgroupskip}{ & & & \tabularnewline}%
8334 \fi
8335 }
```

long4colheader The long4colheader style is like long4col but with a header row.

```
8336 \newglossarystyle{long4colheader}{%
```

Base it on the glostylelong4col style:

```
8337 \setglossarystyle{long4col}{%
```

Table has a header:

```
8338 \renewcommand*{\glossaryheader}{%
8339   \bfseries\entryname&\bfseries\descriptionname&
8340   \bfseries \symbolname&
8341   \bfseries\pagelistname\tabularnewline\endhead}%
8342 }
```

long4colborder The long4colborder style is like long4col but with a border.

```
8343 \newglossarystyle{long4colborder}{%
```

Base it on the glostylelong4col style:

```
8344 \setglossarystyle{long4col}{%
```

Use a longtable with 4 columns surrounded by vertical lines:

```
8345 \renewenvironment{theglossary}{%
```

```

8346    {\begin{longtable}{|l|l|l|l|}}%
8347    {\end{longtable}}%

```

Add horizontal lines to the head and foot of the table:

```

8348    \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8349 }

```

colheaderborder The long4colheaderborder style is like the above but with a border.

```

8350 \newglossarystyle{long4colheaderborder}{%

```

Base it on the glostylelong4col style:

```

8351    \setglossarystyle{long4col}%

```

Use a longtable with 4 columns surrounded by vertical lines:

```

8352    \renewenvironment{theglossary}%
8353    {\begin{longtable}{|l|l|l|l|}}%
8354    {\end{longtable}}%

```

Add table header and horizontal line at the table's foot:

```

8355    \renewcommand*{\glossaryheader}{%
8356        \hline\bfseries\entryname&\bfseries\descriptionname&
8357        \bfseries \symbolname&
8358        \bfseries\pagelistname\tabularnewline\hline\endhead
8359        \hline\endfoot}%
8360 }

```

altlong4col The altlong4col style is like the long4col style but can have multiline descriptions and page lists.

```

8361 \newglossarystyle{altlong4col}{%

```

Base it on the glostylelong4col style:

```

8362    \setglossarystyle{long4col}%

```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```

8363    \renewenvironment{theglossary}%
8364    {\begin{longtable}{lp{\glstdescwidth}lp{\glspagelistwidth}}}%
8365    {\end{longtable}}%
8366 }

```

altlong4colheader The altlong4colheader style is like altlong4col but with a header row.

```

8367 \newglossarystyle{altlong4colheader}{%

```

Base it on the glostylelong4colheader style:

```

8368    \setglossarystyle{long4colheader}%

```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```

8369    \renewenvironment{theglossary}%
8370    {\begin{longtable}{lp{\glstdescwidth}lp{\glspagelistwidth}}}%
8371    {\end{longtable}}%
8372 }

```

`altlong4colborder` The `altlong4colborder` style is like `altlong4col` but with a border.

```
8373 \newglossarystyle{altlong4colborder}{%
```

Base it on the `glostylelong4colborder` style:

```
8374 \setglossarystyle{long4colborder}{%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8375 \renewenvironment{theglossary}{%
```

```
8376 {\begin{longtable}{|l|p{\glstdescwidth}|l|p{\glspagelistwidth}|}}%
```

```
8377 {\end{longtable}}%
```

```
8378 }
```

`colheaderborder` The `altlong4colheaderborder` style is like the above but with a header as well as a border.

```
8379 \newglossarystyle{altlong4colheaderborder}{%
```

Base it on the `glostylelong4colheaderborder` style:

```
8380 \setglossarystyle{long4colheaderborder}{%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8381 \renewenvironment{theglossary}{%
```

```
8382 {\begin{longtable}{|l|p{\glstdescwidth}|l|p{\glspagelistwidth}|}}%
```

```
8383 {\end{longtable}}%
```

```
8384 }
```

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>

```
8385 \ProvidesPackage{glossary-longbooktabs}[2017/08/10 v4.31 (NLCT)]
```

Requires `booktabs` package:

```
8386 \RequirePackage{booktabs}
```

and the base packages for long styles:

```
8387 \RequirePackage{glossary-long}
```

```
8388 \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.

```
8389 \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.

```
8390 \glspatchLToutput
```

As with the longheader style, use the long style as a base.

```
8391 \setglossarystyle{long}%
```

Add a header with rules.

```
8392 \renewcommand*{\glossaryheader}{%
8393   \toprule \bfseries \entryname & \bfseries
8394   \descriptionname\tabularnewline\midrule\endhead
8395   \bottomrule\endfoot}%
```

Check for the nogroupskip package option. If there should be a gap between groups, insert the penalty and the vertical space. The check for nogroupskip should occur outside \glsgroupskip to be on the safe side.

```
8396 \ifglsgroupskip
8397   \renewcommand*{\glsgroupskip}{}%
8398 \else
8399   \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
8400 \fi
8401 }
```

ng3col-booktabs The long3col-booktabs style is similar to the long3colheader style but uses the booktabs rules and patches longtable to check for group skip occurring at a page break.

```
8402 \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.

```
8403 \glspatchLToutput
```

Use the long3col style as a base.

```
8404 \setglossarystyle{long3col}%
```

Add a header with rules.

```
8405 \renewcommand*{\glossaryheader}{%
8406   \toprule \bfseries \entryname &
8407   \bfseries \descriptionname &
8408   \bfseries \pagelistname
8409   \tabularnewline\midrule\endhead
8410   \bottomrule\endfoot}%
```

Check for the nogroupskip package option. If there should be a gap between groups, insert the penalty and the vertical space. The check for nogroupskip should occur outside \glsgroupskip to be on the safe side.

```
8411 \ifglsgroupskip
8412   \renewcommand*{\glsgroupskip}{}%
8413 \else
8414   \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
8415 \fi
8416 }
```

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.

```
8417 \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.

```
8418 \glspatchLToutput
```

Use the long4col style as a base.

```
8419 \setglossarystyle{long4col}%
```

Add a header with rules.

```
8420 \renewcommand*{\glossaryheader}{%
8421   \toprule \bfseries \entryname &
8422   \bfseries \descriptionname &
8423   \bfseries \symbolname &
8424   \bfseries \pagelistname
8425   \tabularnewline\midrule\endhead
8426   \bottomrule\endfoot}%
```

Check for the nogroupskip package option. If there should be a gap between groups, insert the penalty and the vertical space. The check for nogroupskip should occur outside \glsgroupskip to be on the safe side.

```
8427 \ifglsgnogroupskip
8428   \renewcommand*{\glsgroupskip}{}%
8429 \else
8430   \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
8431 \fi
8432 }
```

long4col-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.

```
8433 \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.

```
8434 \glspatchLToutput
```

Use the long4col-booktabs style as a base.

```
8435 \setglossarystyle{long4col-booktabs}%
```

Change the column specifications:

```
8436 \renewenvironment{theglossary}%
8437   {\begin{longtable}{lp{\glstdescwidth}lp{\glspagelistwidth}}}%
8438   {\end{longtable}}%
8439 }
```

Ragged styles.

longragged-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.

```
8440 \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.

```
8441 \glspatchLToutput
```

Use the long-booktabs style as a base.

```
8442 \setglossarystyle{long-booktabs}%
```

Adjust the column specification.

```
8443 \renewenvironment{theglossary}%  
8444     {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}}}%  
8445     {\end{longtable}}%  
8446 }
```

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.

```
8447 \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.

```
8448 \glspatchLToutput
```

Use the long3col-booktabs style as a base.

```
8449 \setglossarystyle{long3col-booktabs}%
```

Adjust the column specification.

```
8450 \renewenvironment{theglossary}%  
8451     {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}%  
8452     >{\raggedright}p{\glspagelistwidth}}}%  
8453     {\end{longtable}}%  
8454 }
```

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.

```
8455 \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.

```
8456 \glspatchLToutput
```

Use the altlong4col-booktabs style as a base.

```
8457 \setglossarystyle{altlong4col-booktabs}%
```

Adjust the column specification.

```
8458 \renewenvironment{theglossary}%  
8459     {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}l%  
8460     >{\raggedright}p{\glspagelistwidth}}}%  
8461     {\end{longtable}}%  
8462 }
```

sLTpenaltycheck

```
8463 \newcommand*{\glslTpenaltycheck}{%  
8464 \ifnum\outputpenalty=-50\vskip-\normalbaselineskip\relax\fi  
8465 }
```


penaltygroupskip

```
8466 \newcommand{\glspenaltygroupskip}{%
8467   \noalign{\penalty-50\vskip\normalbaselineskip}}
```

restoreLToutput Provide a way of restoring \LT@output for the user.

```
8468 \let\@gls@org@LT@output\LT@output
8469 \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

```
8470 \newcommand*{\glspatchLToutput}{%
8471   \renewcommand*{\LT@output}{%
8472     \ifnum\outputpenalty <-\@Mi
8473       \ifnum\outputpenalty > -\LT@end@pen
8474         \LT@err{floats and marginpars not allowed in a longtable}\@ehc
8475       \else
8476         \setbox\z@\vbox{\unvbox\@cclv}%
8477         \ifdim \ht\LT@lastfoot>\ht\LT@foot
8478           \dimen@\pagegoal
8479           \advance\dimen@-\ht\LT@lastfoot
8480           \ifdim\dimen@<\ht\z@
8481             \setbox\@cclv\vbox{\unvbox\z@\copy\LT@foot\vss}%
8482             \@makecol
8483             \@outputpage
8484             \setbox\z@\vbox{\box\LT@head\glsLTpenaltycheck}%
8485           \fi
8486         \fi
8487         \global\@colroom\@colht
8488         \global\vsizel\@colht
8489         {\unvbox\z@\box\ifvoid\LT@lastfoot\LT@foot\else\LT@lastfoot\fi}%
8490       \fi
8491     \else
8492       \setbox\@cclv\vbox{\unvbox\@cclv\copy\LT@foot\vss}%
8493       \@makecol
8494       \@outputpage
8495       \global\vsizel\@colroom
8496       \copy\LT@head
8497       \glsLTpenaltycheck
8498       \nobreak
8499     \fi
8500   }%
8501 }
```

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.

8502 \ProvidesPackage{glossary-longragged}[2017/08/10 v4.31 (NLCT)]

Requires the package:

8503 \RequirePackage{array}

Requires the package:

8504 \RequirePackage{longtable}

`\glsdescwidth` This is a length that governs the width of the description column. This may have already been defined.

8505 \@ifundefined{glsdescwidth}{%

8506 \newlength{glsdescwidth}

8507 \setlength{glsdescwidth}{0.6\hsize}

8508 }{}

`glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

8509 \@ifundefined{glspagelistwidth}{%

8510 \newlength{glspagelistwidth}

8511 \setlength{glspagelistwidth}{0.1\hsize}

8512 }{}

`longragged` The longragged glossary style is like the long but uses ragged right formatting for the description column.

8513 \newglossarystyle{longragged}{%

Use longtable with two columns:

8514 \renewenvironment{theglossary}{%

8515 {\begin{longtable}{l>{\raggedright}p{glsdescwidth}}}%

8516 {\end{longtable}}}%

Do nothing at the start of the environment:

8517 \renewcommand*{\glossaryheader}{}%

No heading between groups:

8518 \renewcommand*{\glsgroupheading}[1]{}%

Main (level 0) entries displayed in a row:

8519 \renewcommand{\glossentry}[2]{%

8520 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &

8521 \glossentrydesc{##1}\glspostdescription\space ##2%

8522 \tabularnewline

8523 }%

Sub entries displayed on the following row without the name:

```
8524 \renewcommand{\subglossentry}[3]{%
8525     &
8526     \glssubentryitem{##2}%
8527     \glstarget{##2}{\strut}\glossentrydesc{##2}%
8528     \glspostdescription\space ##3%
8529     \tabularnewline
8530 }%
```

Blank row between groups: The check for nogroupskip must occur outside `\glsgroupskip`
(<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```
8531 \ifglsnogroupskip
8532 \renewcommand*{\glsgroupskip}{}%
8533 \else
8534 \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
8535 \fi
8536 }
```

`ongraggedborder` The `longraggedborder` style is like the above, but with horizontal and vertical lines:

```
8537 \newglossarystyle{longraggedborder}{%
```

Base it on the `glostylelongragged` style:

```
8538 \setglossarystyle{longragged}%
```

Use `longtable` with two columns with vertical lines between each column:

```
8539 \renewenvironment{theglossary}{%
8540 \begin{longtable}{|l|>{\raggedright}p{\glstdescwidth}|}%
8541 {\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
8542 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8543 }
```

`ongraggedheader` The `longraggedheader` style is like the `longragged` style but with a header:

```
8544 \newglossarystyle{longraggedheader}{%
```

Base it on the `glostylelongragged` style:

```
8545 \setglossarystyle{longragged}%
```

Set the table's header:

```
8546 \renewcommand*{\glossaryheader}{%
8547 \bfseries \entryname & \bfseries \descriptionname
8548 \tabularnewline\endhead}%
8549 }
```

`ongraggedheaderborder` The `longraggedheaderborder` style is like the `longragged` style but with a header and border:

```
8550 \newglossarystyle{longraggedheaderborder}{%
```

Base it on the `glostylelongraggedborder` style:

```
8551 \setglossarystyle{longraggedborder}%
```

Set the table's header and add horizontal line to table's foot:

```
8552 \renewcommand*{\glossaryheader}{%
8553 \hline\bfseries \entryname & \bfseries \descriptionname
8554 \tabularnewline\hline
8555 \endhead
8556 \hline\endfoot}%
8557 }
```

`longragged3col` The `longragged3col` style is like `longragged` but with 3 columns

```
8558 \newglossarystyle{longragged3col}{%
```

Use a longtable with 3 columns:

```
8559 \renewenvironment{theglossary}%
8560 {\begin{longtable}{l>{\raggedright}p{\glstdescwidth}%
8561 >{\raggedright}p{\glspagelistwidth}}}%
8562 {\end{longtable}}%
```

No table header:

```
8563 \renewcommand*{\glossaryheader}{}%
```

No headings between groups:

```
8564 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
8565 \renewcommand{\glossentry}[2]{%
8566 \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8567 \glossentrydesc{##1} & ##2\tabularnewline
8568 }%
```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```
8569 \renewcommand{\subglossentry}[3]{%
8570 &
8571 \glssubentryitem{##2}%
8572 \glstarget{##2}{\strut}\glossentrydesc{##2} &
8573 ##3\tabularnewline
8574 }%
```

Blank row between groups: The check for `nogroupskip` must occur outside `\glsgroupskip` (<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```
8575 \ifglsnogroupskip
8576 \renewcommand*{\glsgroupskip}{}%
8577 \else
8578 \renewcommand*{\glsgroupskip}{ & & \tabularnewline}%
8579 \fi
8580 }
```

`ragged3colborder` The `longragged3colborder` style is like the `longragged3col` style but with a border:

```
8581 \newglossarystyle{longragged3colborder}{%
```

Base it on the `glostylelongragged3col` style:

```
8582 \setglossarystyle{longragged3col}%
```

Use a `longtable` with 3 columns with vertical lines around them:

```
8583 \renewenvironment{theglossary}%  
8584 {\begin{longtable}{|l|>{\raggedright}p{\glsgdescwidth}|%  
8585 >{\raggedright}p{\glspagelistwidth}|}%  
8586 {\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
8587 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%  
8588 }
```

`ragged3colheader` The `longragged3colheader` style is like `longragged3col` but with a header row:

```
8589 \newglossarystyle{longragged3colheader}{%
```

Base it on the `glostylelongragged3col` style:

```
8590 \setglossarystyle{longragged3col}%
```

Set the table's header:

```
8591 \renewcommand*{\glossaryheader}{%  
8592 \bfseries\entryname&\bfseries\descriptionname&  
8593 \bfseries\pagelistname\tabularnewline\endhead}%  
8594 }
```

`colheaderborder` The `longragged3colheaderborder` style is like the above but with a border

```
8595 \newglossarystyle{longragged3colheaderborder}{%
```

Base it on the `glostylelongragged3colborder` style:

```
8596 \setglossarystyle{longragged3colborder}%
```

Set the table's header and add horizontal line at table's foot:

```
8597 \renewcommand*{\glossaryheader}{%  
8598 \hline  
8599 \bfseries\entryname&\bfseries\descriptionname&  
8600 \bfseries\pagelistname\tabularnewline\hline\endhead  
8601 \hline\endfoot}%  
8602 }
```

`longragged4col` The `altlongragged4col` style is like the `altlong4col` style defined in the package, except that ragged right formatting is used for the description and page list columns.

```
8603 \newglossarystyle{altlongragged4col}{%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8604 \renewenvironment{theglossary}%  
8605 {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}l%  
8606 >{\raggedright}p{\glspagelistwidth}}}%  
8607 {\end{longtable}}%
```

No table header:

```
8608 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8609 \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):

```
8610 \renewcommand{\glossentry}[2]{%
8611   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8612   \glossentrydesc{##1} & \glossentrysymbol{##1} &
8613   ##2\tabularnewline
8614 }%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
8615 \renewcommand{\subglossentry}[3]{%
8616   &
8617   \glssubentryitem{##2}%
8618   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8619   \glossentrysymbol{##2} & ##3\tabularnewline
8620 }%
```

Blank row between groups: The check for nogroupskip must occur outside \glsgroupskip (<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```
8621 \ifglsnogroupskip
8622 \renewcommand*\glsgroupskip{}{%
8623   \else
8624   \renewcommand*\glsgroupskip{ & & & \tabularnewline}%
8625 \fi
8626 }
```

ragged4colheader The altlongragged4colheader style is like altlongragged4col but with a header row.

```
8627 \newglossarystyle{altlongragged4colheader}{%
```

Base it on the glostylealtlongragged4col style:

```
8628 \setglossarystyle{altlongragged4col}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8629 \renewenvironment{theglossary}%
8630   {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}l%
8631     >{\raggedright}p{\glspagelistwidth}}}%
8632   {\end{longtable}}%
```

Table has a header:

```
8633 \renewcommand*\glossaryheader{%
8634   \bfseries\entryname&\bfseries\descriptionname&
8635   \bfseries \symbolname&
8636   \bfseries\pagelistname\tabularnewline\endhead}%
8637 }
```

ragged4colborder The altlongragged4colborder style is like altlongragged4col but with a border.

```
8638 \newglossarystyle{altlongragged4colborder}{%
```

Base it on the `glostylealtlongragged4col` style:

```
8639 \setglossarystyle{altlongragged4col}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8640 \renewenvironment{theglossary}%  
8641 {\begin{longtable}{|l|>{\raggedright}p{\glsgdescwidth}|l|}%  
8642 >{\raggedright}p{\glspagelistwidth}|}}%  
8643 {\end{longtable}}%
```

Add horizontal lines to the head and foot of the table:

```
8644 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%  
8645 }
```

`colheaderborder` The `altlongragged4colheaderborder` style is like the above but with a header as well as a border.

```
8646 \newglossarystyle{altlongragged4colheaderborder}{%
```

Base it on the `glostylealtlongragged4col` style:

```
8647 \setglossarystyle{altlongragged4col}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8648 \renewenvironment{theglossary}%  
8649 {\begin{longtable}{|l|>{\raggedright}p{\glsgdescwidth}|l|}%  
8650 >{\raggedright}p{\glspagelistwidth}|}}%  
8651 {\end{longtable}}%
```

Add table header and horizontal line at the table's foot:

```
8652 \renewcommand*{\glossaryheader}{%  
8653 \hline\bfseries\entryname&\bfseries\descriptionname&  
8654 \bfseries \symbolname&  
8655 \bfseries\pagelistname\tabularnewline\hline\endhead  
8656 \hline\endfoot}%  
8657 }
```

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.

```
8658 \ProvidesPackage{glossary-mcols}[2017/08/10 v4.31 (NLCT)]
```

Required packages:

```
8659 \RequirePackage{multicol}  
8660 \RequirePackage{glossary-tree}
```

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```
8661 \providecommand{\indexspace}{%  
8662 \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax  
8663 }
```

`\glsmcols` Define macro in which to store the number of columns. (Defaults to 2.)

```
8664 \newcommand*{\glsmcols}{2}
```

`mcolindex` Multi-column index style. Same as the index, but puts the glossary in multiple columns. (Ideally the glossary title should go in the optional argument of multicols, but the title isn't part of the glossary style.)

```
8665 \newglossarystyle{mcolindex}{%
8666   \setglossarystyle{index}%
8667   \renewenvironment{theglossary}%
8668     {%
8669       \begin{multicols}{\glsmcols}
8670       \setlength{\parindent}{0pt}%
8671       \setlength{\parskip}{0pt plus 0.3pt}%
8672       \let\item\glstreeitem
8673       \let\subitem\glstreesubitem
8674       \let\subsubitem\glstreesubsubitem
8675     }%
8676     {\end{multicols}}%
8677 }
```

`mcolindexgroup` As `mcolindex` but has headings:

```
8678 \newglossarystyle{mcolindexgroup}{%
8679   \setglossarystyle{mcolindex}%
8680   \renewcommand*{\glsgroupheading}[1]{%
8681     \item\glstreegroupheaderfmt{\glsgrouptitle{##1}}\indexspace}%
8682 }
```

`indexhypergroup` The `mcolindexhypergroup` style is like the `mcolindexgroup` style but has hyper navigation.

```
8683 \newglossarystyle{mcolindexhypergroup}{%
```

Base it on the `glostylemcolindex` style:

```
8684   \setglossarystyle{mcolindex}%
```

Put navigation links to the groups at the start of the glossary:

```
8685   \renewcommand*{\glossaryheader}{%
8686     \item\glstreenavigationfmt{\glsnavigation}\indexspace}%
```

Add a heading for each group (with a target). The group's title is in bold followed by a vertical gap.

```
8687   \renewcommand*{\glsgroupheading}[1]{%
8688     \item\glstreegroupheaderfmt
8689       {\glsnavigationtarget{##1}{\glsgrouptitle{##1}}}%
8690     \indexspace}%
8691 }
```

`colindexspannav` Similar to `mcolindexhypergroup`, but puts the navigation line in the optional argument of multicols.


```

8692 \newglossarystyle{mcolindexspannav}{%
8693   \setglossarystyle{index}%
8694   \renewenvironment{theglossary}%
8695     {%

8696     \begin{multicols}{\glsmcols}\[noindent\glstreenavigationfmt{\glsnavigation}]
8697     \setlength{\parindent}{0pt}%
8698     \setlength{\parskip}{0pt plus 0.3pt}%

8699     \let\item\glstreeitem}%
8700   {\end{multicols}}%

```

Add a heading for each group (with a target). The group's title is in bold followed by a vertical gap.

```

8701 \renewcommand*{\glsgroupheading}[1]{%
8702   \item\glstreegroupheaderfmt
8703     {\glsnahypertarget{##1}{\glsgrouptitle{##1}}}%
8704   \indexspace}%
8705 }

```

mcoltree Multi-column index style. Same as the tree, but puts the glossary in multiple columns.

```

8706 \newglossarystyle{mcoltree}{%
8707   \setglossarystyle{tree}%
8708   \renewenvironment{theglossary}%
8709     {%

8710     \begin{multicols}{\glsmcols}
8711     \setlength{\parindent}{0pt}%
8712     \setlength{\parskip}{0pt plus 0.3pt}%
8713   }%
8714   {\end{multicols}}%
8715 }

```

mcoltreegroup Like the mcoltree style but the glossary groups have headings.

```

8716 \newglossarystyle{mcoltreegroup}{%
  Base it on the glostylemcoltree style:
8717   \setglossarystyle{mcoltree}%
  Each group has a heading (in bold) followed by a vertical gap):
8718   \renewcommand{\glsgroupheading}[1]{\par
8719     \noindent\glstreegroupheaderfmt{\glsgrouptitle{##1}}\par\indexspace}%
8720 }

```

1treehypergroup The mcoltreehypergroup style is like the treegroup style, but has a set of links to the groups at the start of the glossary.

```

8721 \newglossarystyle{mcoltreehypergroup}{%
  Base it on the glostylemcoltree style:
8722   \setglossarystyle{mcoltree}%

```

Put navigation links to the groups at the start of the theglossary environment:

```
8723 \renewcommand*{\glossaryheader}{%
8724 \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
8725 \renewcommand*{\glsgroupheading}[1]{%
8726 \par\noindent
8727 \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8728 \indexspace}%
8729 }
```

mcoltreespannav Similar to the mcoltreehypergroup style but the navigation line is put in the optional argument of the multicols environment.

```
8730 \newglossarystyle{mcoltreespannav}{%
8731 \setglossarystyle{tree}%
8732 \renewenvironment{theglossary}%
8733 {%
8734 \begin{multicols}{\glsmcols}[\noindent\glstreenavigationfmt{\glsnavigation}]
8735 \setlength{\parindent}{0pt}%
8736 \setlength{\parskip}{0pt plus 0.3pt}%
8737 }%
8738 {\end{multicols}}}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
8739 \renewcommand*{\glsgroupheading}[1]{%
8740 \par\noindent
8741 \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8742 \indexspace}%
8743 }
```

mcoltreenoname Multi-column index style. Same as the treenoname, but puts the glossary in multiple columns.

```
8744 \newglossarystyle{mcoltreenoname}{%
8745 \setglossarystyle{treenoname}%
8746 \renewenvironment{theglossary}%
8747 {%
8748 \begin{multicols}{\glsmcols}
8749 \setlength{\parindent}{0pt}%
8750 \setlength{\parskip}{0pt plus 0.3pt}%
8751 }%
8752 {\end{multicols}}}%
8753 }
```

treenonamegroup Like the mcoltreenoname style but the glossary groups have headings.

```
8754 \newglossarystyle{mcoltreenonamegroup}{%
Base it on the glostylemcoltreenoname style:
8755 \setglossarystyle{mcoltreenoname}%
```

Give each group a heading:

```
8756 \renewcommand{\glsgroupheading}[1]{\par
8757 \noindent\glstreegroupheaderfmt{\glsgrouptitle{##1}}\par\indexspace}%
8758 }
```

`onamehypergroup` The `mcoltreenonamehypergroup` style is like the `mcoltreenonamegroup` style, but has a set of links to the groups at the start of the glossary.

```
8759 \newglossarystyle{mcoltreenonamehypergroup}{%
```

Base it on the `glostylemcoltreenoname` style:

```
8760 \setglossarystyle{mcoltreenoname}%
```

Put navigation links to the groups at the start of the `theglossary` environment:

```
8761 \renewcommand*{\glossaryheader}{%
8762 \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
8763 \renewcommand*{\glsgroupheading}[1]{%
8764 \par\noindent
8765 \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}\par
8766 \indexspace}%
8767 }
```

`eenonamespannav` Similar to the `mcoltreenonamehypergroup` style but the navigation line is put in the optional argument of the `multicols` environment.

```
8768 \newglossarystyle{mcoltreenonamespannav}{%
8769 \setglossarystyle{treenoname}%
8770 \renewenvironment{theglossary}%
8771 {%
8772 \begin{multicols}{\glsmcols}\noindent\glstreenavigationfmt{\glsnavigation}]
8773 \setlength{\parindent}{0pt}%
8774 \setlength{\parskip}{0pt plus 0.3pt}%
8775 }%
8776 {\end{multicols}}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
8777 \renewcommand*{\glsgroupheading}[1]{%
8778 \par\noindent
8779 \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}\par
8780 \indexspace}%
8781 }
```

`mcolalttree` Multi-column index style. Same as the `alttree`, but puts the glossary in multiple columns.

```
8782 \newglossarystyle{mcolalttree}{%
8783 \setglossarystyle{alttree}%
8784 \renewenvironment{theglossary}%
8785 {%
8786 \begin{multicols}{\glsmcols}
8787 \def\@gls@prevlevel{-1}%
```

```

8788     \mbox{}\par
8789   }%
8790   {\par\end{multicols}}}%
8791 }

```

colalmtreegroup Like the mcolalmtree style but the glossary groups have headings.

```

8792 \newglossarystyle{mcolalmtreegroup}{%
      Base it on the glostylemcolalmtree style:
8793   \setglossarystyle{mcolalmtree}%
      Give each group a heading.
8794   \renewcommand{\glsgroupheading}[1]{\par
8795     \def\@gls@prevlevel{-1}%
8796     \hangindent0pt\relax
8797     \parindent0pt\relax
8798     \glstreegroupheaderfmt{\glsgrouptitle{##1}}\par\indexspace}%
8799 }

```

treehypergroup The mcolalmtreehypergroup style is like the mcolalmtreegroup style, but has a set of links to the groups at the start of the glossary.

```

8800 \newglossarystyle{mcolalmtreehypergroup}{%
      Base it on the glostylemcolalmtree style:
8801   \setglossarystyle{mcolalmtree}%
      Put the navigation links in the header
8802   \renewcommand*{\glossaryheader}{%
8803     \par
8804     \def\@gls@prevlevel{-1}%
8805     \hangindent0pt\relax
8806     \parindent0pt\relax
8807     \glstreenavigationfmt{\glsnavigation}\par\indexspace}%
      Put a hypertarget at the start of each group
8808   \renewcommand*{\glsgroupheading}[1]{%
8809     \par
8810     \def\@gls@prevlevel{-1}%
8811     \hangindent0pt\relax
8812     \parindent0pt\relax
8813     \glstreegroupheaderfmt{\glshypertarget{##1}{\glsgrouptitle{##1}}}\par
8814     \indexspace}%
8815 }

```

almtreepannav Similar to the mcolalmtreehypergroup style but the navigation line is put in the optional argument of the multicols environment.

```

8816 \newglossarystyle{mcolalmtreepannav}{%
8817   \setglossarystyle{almtree}%
8818   \renewenvironment{theglossary}%
8819   {%
8820     \begin{multicols}{\glsmcols}\noindent\glstreenavigationfmt{\glsnavigation}]

```

```

8821     \def\@gls@prevlevel{-1}%
8822     \mbox{}\par
8823 }%
8824 {\par\end{multicols}}}%

Put a hypertarget at the start of each group
8825 \renewcommand*\@gls@groupheading[1]{%
8826     \par
8827     \def\@gls@prevlevel{-1}%
8828     \hangindent0pt\relax
8829     \parindent0pt\relax
8830     \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8831     \indexspace}
8832 }

```

3.8 Glossary Styles using supertabular environment (glossary-super package)

The glossary styles defined in the package use the supertabular environment.

```
8833 \ProvidesPackage{glossary-super}[2017/08/10 v4.31 (NLCT)]
```

Requires the package:

```
8834 \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.

```

8835 \@ifundefined{glsdescwidth}{%
8836     \newlength\glsdescwidth
8837     \setlength{\glsdescwidth}{0.6\hsize}
8838 }{}

```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined if has been loaded.

```

8839 \@ifundefined{glspagelistwidth}{%
8840     \newlength\glspagelistwidth
8841     \setlength{\glspagelistwidth}{0.1\hsize}
8842 }{}

```

`super` The super glossary style uses the supertabular environment (it uses lengths defined in the package.)

```
8843 \newglossarystyle{super}{%
```

Put the glossary in a supertabular environment with two columns and no head or tail:

```

8844     \renewenvironment{theglossary}%
8845     {\tablehead{}\tabletail{}}%
8846     \begin{supertabular}{lp{\glsdescwidth}}}%
8847     {\end{supertabular}}}%

```

Do nothing at the start of the table:

```
8848 \renewcommand*\glossaryheader{}\%
```

No group headings:

```
8849 \renewcommand*\glsgroupheading[1]{}%
```

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```
8850 \renewcommand\glossentry[2]{%
8851   \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8852   \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline
8853 }%
```

Sub entries put in a row (no name, description and page list in second column):

```
8854 \renewcommand\subglossentry[3]{%
8855   &
8856   \glssubentryitem{##2}%
8857   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
8858   ##3\tabularnewline
8859 }%
```

Blank row between groups: The check for nogroupskip must occur outside \glsgroupskip (<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```
8860 \ifglsgroupskip
8861 \renewcommand*\glsgroupskip{}%
8862 \else
8863 \renewcommand*\glsgroupskip{& \tabularnewline}%
8864 \fi
8865 }
```

superborder The superborder style is like the above, but with horizontal and vertical lines:

```
8866 \newglossarystyle{superborder}{%
```

Base it on the glosstylesuper style:

```
8867 \setglossarystyle{super}%
```

Put the glossary in a supertabular environment with two columns and a horizontal line in the head and tail:

```
8868 \renewenvironment{theglossary}%
8869 {\tablehead{\hline}\tabletail{\hline}%
8870 \begin{supertabular}{|l|p{\glstdescwidth}|}%
8871 {\end{supertabular}}%
8872 }
```

superheader The superheader style is like the super style, but with a header:

```
8873 \newglossarystyle{superheader}{%
```

Base it on the glosstylesuper style:

```
8874 \setglossarystyle{super}%
```

Put the glossary in a supertabular environment with two columns, a header and no tail:

```

8875 \renewenvironment{theglossary}%
8876   {\tablehead{\bfseries \entryname &
8877     \bfseries \descriptionname \tabularnewline}%
8878    \tabletail{}}%
8879    \begin{supertabular}{lp{\glsgdescwidth}}%
8880   {\end{supertabular}}%
8881 }
```

superheaderborder The superheaderborder style is like the super style but with a header and border:

```
8882 \newglossarystyle{superheaderborder}{%
```

Base it on the glostylessuper style:

```
8883 \setglossarystyle{super}%
```

Put the glossary in a supertabular environment with two columns, a header and horizontal lines above and below the table:

```

8884 \renewenvironment{theglossary}%
8885   {\tablehead{\hline\bfseries \entryname &
8886     \bfseries \descriptionname \tabularnewline\hline}%
8887    \tabletail{\hline}
8888    \begin{supertabular}{llp{\glsgdescwidth}}%
8889   {\end{supertabular}}%
8890 }
```

super3col The super3col style is like the super style, but with 3 columns:

```
8891 \newglossarystyle{super3col}{%
```

Put the glossary in a supertabular environment with three columns and no head or tail:

```

8892 \renewenvironment{theglossary}%
8893   {\tablehead{}\tabletail{}}%
8894    \begin{supertabular}{lp{\glsgdescwidth}p{\glspagelistwidth}}%
8895   {\end{supertabular}}%
```

Do nothing at the start of the table:

```
8896 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8897 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```

8898 \renewcommand{\glossentry}[2]{%
8899   \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8900   \glossentrydesc{##1} & ##2 \tabularnewline
8901 }
```

Sub entries on a row (no name, description in second column, page list in last column):

```

8902 \renewcommand{\subglossentry}[3]{%
8903   &
8904   \glssubentryitem{##2}%
```

```

8905     \glstarget{##2}{\strut}\glossentrydesc{##2} &
8906     ##3\tabularnewline
8907 }%

```

Blank row between groups: The check for nogroupskip must occur outside \glsgroupskip (<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```

8908 \ifglsgroupskip
8909 \renewcommand*{\glsgroupskip}{}%
8910 \else
8911 \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
8912 \fi
8913 }

```

super3colborder The super3colborder style is like the super3col style, but with a border:

```
8914 \newglossarystyle{super3colborder}{%
```

Base it on the glostylesuper3col style:

```
8915 \setglossarystyle{super3col}%
```

Put the glossary in a supertabular environment with three columns and a horizontal line in the head and tail:

```

8916 \renewenvironment{theglossary}%
8917 {\tablehead{\hline}\tabletail{\hline}%
8918 \begin{supertabular}{|l|p{\glsdescwidth}|p{\glspagelistwidth}|}%
8919 {\end{supertabular}}%
8920 }

```

super3colheader The super3colheader style is like the super3col style but with a header row:

```
8921 \newglossarystyle{super3colheader}{%
```

Base it on the glostylesuper3col style:

```
8922 \setglossarystyle{super3col}%
```

Put the glossary in a supertabular environment with three columns, a header and no tail:

```

8923 \renewenvironment{theglossary}%
8924 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8925 \bfseries\pagelistname\tabularnewline}\tabletail{}}%
8926 \begin{supertabular}{lp{\glsdescwidth}p{\glspagelistwidth}}}%
8927 {\end{supertabular}}%
8928 }

```

colheaderborder The super3colheaderborder style is like the super3col style but with a header and border:

```
8929 \newglossarystyle{super3colheaderborder}{%
```

Base it on the glostylesuper3colborder style:

```
8930 \setglossarystyle{super3colborder}%
```

Put the glossary in a supertabular environment with three columns, a header with horizontal lines and a horizontal line in the tail:

```

8931 \renewenvironment{theglossary}%
8932 {\tablehead{\hline

```



```

8933      \bfseries\entryname&\bfseries\descriptionname&
8934      \bfseries\pagelistname\tabularnewline\hline}%
8935      \tabletail{\hline}%
8936      \begin{supertabular}{|l|p{\glsdescwidth}|p{\glspagelistwidth}|}%
8937      {\end{supertabular}}}%
8938 }

```

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.

```

8939 \newglossarystyle{super4col}{%

```

Put the glossary in a supertabular environment with four columns and no head or tail:

```

8940 \renewenvironment{theglossary}%
8941 {\tablehead{}\tabletail}%
8942 \begin{supertabular}{|l|l|l|l|}%
8943 \end{supertabular}}%

```

Do nothing at the start of the table:

```

8944 \renewcommand*{\glossaryheader}{}%

```

No group headings:

```

8945 \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:

```

8946 \renewcommand{\glossentry}[2]{%
8947 \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8948 \glossentrydesc{##1} &
8949 \glossentrysymbol{##1} & ##2\tabularnewline
8950 }%

```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```

8951 \renewcommand{\subglossentry}[3]{%
8952 &
8953 \glssubentryitem{##2}%
8954 \glstarget{##2}{\strut}\glossentrydesc{##2} &
8955 \glossentrysymbol{##2} & ##3\tabularnewline
8956 }%

```

Blank row between groups: The check for nogroupskip must occur outside \glsgroupskip (<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```

8957 \ifglsnogroupskip
8958 \renewcommand*{\glsgroupskip}{}%
8959 \else
8960 \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
8961 \fi
8962 }

```

super4colheader The super4colheader style is like the super4col but with a header row.

```

8963 \newglossarystyle{super4colheader}{%

```

Base it on the `glostylesuper4col` style:

```
8964 \setglossarystyle{super4col}%
```

Put the glossary in a `supertabular` environment with four columns, a header and no tail:

```
8965 \renewenvironment{theglossary}%
8966 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8967 \bfseries\symbolname &
8968 \bfseries\pagelistname\tabularnewline}%
8969 \tabletail{}}%
8970 \begin{supertabular}{|l|l|l|l|}%
8971 {\end{supertabular}}%
8972 }
```

`super4colborder` The `super4colborder` style is like the `super4col` but with a border.

```
8973 \newglossarystyle{super4colborder}{%
```

Base it on the `glostylesuper4col` style:

```
8974 \setglossarystyle{super4col}%
```

Put the glossary in a `supertabular` environment with four columns and a horizontal line in the head and tail:

```
8975 \renewenvironment{theglossary}%
8976 {\tablehead{\hline}\tabletail{\hline}%
8977 \begin{supertabular}{|l|l|l|l|}%
8978 {\end{supertabular}}%
8979 }
```

`colheaderborder` The `super4colheaderborder` style is like the `super4col` but with a header and border.

```
8980 \newglossarystyle{super4colheaderborder}{%
```

Base it on the `glostylesuper4col` style:

```
8981 \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:

```
8982 \renewenvironment{theglossary}%
8983 {\tablehead{\hline\bfseries\entryname&\bfseries\descriptionname&
8984 \bfseries\symbolname &
8985 \bfseries\pagelistname\tabularnewline\hline}%
8986 \tabletail{\hline}%
8987 \begin{supertabular}{|l|l|l|l|}%
8988 {\end{supertabular}}%
8989 }
```

`altsuper4col` The `altsuper4col` glossary style is like `super4col` but has provision for multiline descriptions.

```
8990 \newglossarystyle{altsuper4col}{%
```

Base it on the `glostylesuper4col` style:

```
8991 \setglossarystyle{super4col}%
```

Put the glossary in a supertabular environment with four columns and no head or tail:

```
8992 \renewenvironment{theglossary}%
8993 {\tablehead{}\tabletail{}}%
8994 \begin{supertabular}{lp{\glsdescwidth}lp{\glspagelistwidth}}%
8995 {\end{supertabular}}%
8996 }
```

super4colheader The `altsuper4colheader` style is like the `altsuper4col` but with a header row.

```
8997 \newglossarystyle{altsuper4colheader}{%
```

Base it on the `glostylesuper4colheader` style:

```
8998 \setglossarystyle{super4colheader}%
```

Put the glossary in a supertabular environment with four columns, a header and no tail:

```
8999 \renewenvironment{theglossary}%
9000 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
9001 \bfseries\symbolname &
9002 \bfseries\pagelistname\tabularnewline}\tabletail{}}%
9003 \begin{supertabular}{lp{\glsdescwidth}lp{\glspagelistwidth}}%
9004 {\end{supertabular}}%
9005 }
```

super4colborder The `altsuper4colborder` style is like the `altsuper4col` but with a border.

```
9006 \newglossarystyle{altsuper4colborder}{%
```

Base it on the `glostylesuper4colborder` style:

```
9007 \setglossarystyle{super4colborder}%
```

Put the glossary in a supertabular environment with four columns and a horizontal line in the head and tail:

```
9008 \renewenvironment{theglossary}%
9009 {\tablehead{\hline}\tabletail{\hline}%
9010 \begin{supertabular}%
9011 {lllp{\glsdescwidth}lllp{\glspagelistwidth}}}%
9012 {\end{supertabular}}%
9013 }
```

colheaderborder The `altsuper4colheaderborder` style is like the `altsuper4col` but with a header and border.

```
9014 \newglossarystyle{altsuper4colheaderborder}{%
```

Base it on the `glostylesuper4colheaderborder` style:

```
9015 \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:

```
9016 \renewenvironment{theglossary}%
9017 {\tablehead{\hline
9018 \bfseries\entryname &
9019 \bfseries\descriptionname &
9020 \bfseries\symbolname &
9021 \bfseries\pagelistname\tabularnewline\hline}%
```

```

9022     \tabletail{\hline}%
9023     \begin{supertabular}%
9024         {l|p{\glsdescwidth}|l|p{\glspagelistwidth}|}%
9025     {\end{supertabular}}%
9026 }

```

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.

```

9027 \ProvidesPackage{glossary-superragged}[2017/08/10 v4.31 (NLCT)]

```

Requires the package:

```

9028 \RequirePackage{array}

```

Requires the package:

```

9029 \RequirePackage{supertabular}

```

`\glsdescwidth` This is a length that governs the width of the description column. This may already have been defined.

```

9030 \@ifundefined{glsdescwidth}{%
9031     \newlength{glsdescwidth}
9032     \setlength{glsdescwidth}{0.6\hsize}
9033 }{}

```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```

9034 \@ifundefined{glspagelistwidth}{%
9035     \newlength{glspagelistwidth}
9036     \setlength{glspagelistwidth}{0.1\hsize}
9037 }{}

```

`superragged` The superragged glossary style uses the supertabular environment.

```

9038 \newglossarystyle{superragged}{%

```

Put the glossary in a supertabular environment with two columns and no head or tail:

```

9039     \renewenvironment{theglossary}%
9040     {\tablehead{}\tabletail{}}%
9041     \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}}%
9042     {\end{supertabular}}%

```

Do nothing at the start of the table:

```

9043     \renewcommand*{\glossaryheader}{}%

```

No group headings:

```

9044     \renewcommand*{\glsgroupheading}[1]{}%

```

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```

9045 \renewcommand{\glossentry}[2]{%
9046   \glentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
9047   \glossentrydesc{##1}\glspostdescription\space ##2%
9048   \tabularnewline
9049 }%

```

Sub entries put in a row (no name, description and page list in second column):

```

9050 \renewcommand{\subglossentry}[3]{%
9051   &
9052   \glssubentryitem{##2}%
9053   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
9054   ##3%
9055   \tabularnewline
9056 }%

```

Blank row between groups: The check for nogroupskip must occur outside \glsgroupskip (<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```

9057 \ifglsgroupskip
9058   \renewcommand*{\glsgroupskip}{}%
9059 \else
9060   \renewcommand*{\glsgroupskip}{& \tabularnewline}%
9061 \fi
9062 }

```

superraggedborder The superraggedborder style is like the above, but with horizontal and vertical lines:

```

9063 \newglossarystyle{superraggedborder}{%

```

Base it on the glostylesuperragged style:

```

9064 \setglossarystyle{superragged}%

```

Put the glossary in a supertabular environment with two columns and a horizontal line in the head and tail:

```

9065 \renewenvironment{theglossary}%
9066   {\tablehead{\hline}\tabletail{\hline}%
9067   \begin{supertabular}{|l|>{\raggedright}p{\glsglwidth}}}%
9068   {\end{supertabular}}%
9069 }

```

superraggedheader The superraggedheader style is like the super style, but with a header:

```

9070 \newglossarystyle{superraggedheader}{%

```

Base it on the glostylesuperragged style:

```

9071 \setglossarystyle{superragged}%

```

Put the glossary in a supertabular environment with two columns, a header and no tail:

```

9072 \renewenvironment{theglossary}%
9073   {\tablehead{\bfseries \entryname & \bfseries \descriptionname
9074     \tabularnewline}%
9075   \tabletail{}}%

```

```

9076 \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}}}%
9077 {\end{supertabular}}}%
9078 }

```

gedheaderborder The superraggedheaderborder style is like the superragged style but with a header and border:

```

9079 \newglossarystyle{superraggedheaderborder}{%

```

Base it on the glostylesuper style:

```

9080 \setglossarystyle{superragged}%

```

Put the glossary in a supertabular environment with two columns, a header and horizontal lines above and below the table:

```

9081 \renewenvironment{theglossary}%
9082 {\tablehead{\hline\bfseries \entryname &
9083 \bfseries \descriptionname\tabularnewline\hline}%
9084 \tabletail{\hline}
9085 \begin{supertabular}{l|>{\raggedright}p{\glsgdescwidth}|}%
9086 {\end{supertabular}}}%
9087 }

```

superragged3col The superragged3col style is like the superragged style, but with 3 columns:

```

9088 \newglossarystyle{superragged3col}{%

```

Put the glossary in a supertabular environment with three columns and no head or tail:

```

9089 \renewenvironment{theglossary}%
9090 {\tablehead{}\tabletail{}}%
9091 \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}%
9092 >{\raggedright}p{\glspagelistwidth}}}%
9093 {\end{supertabular}}%

```

Do nothing at the start of the table:

```

9094 \renewcommand*{\glossaryheader}{}%

```

No group headings:

```

9095 \renewcommand*{\glsgroupheading}[1]{}%

```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```

9096 \renewcommand{\glossentry}[2]{%
9097 \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
9098 \glossentrydesc{##1} &
9099 ##2\tabularnewline
9100 }%

```

Sub entries on a row (no name, description in second column, page list in last column):

```

9101 \renewcommand{\subglossentry}[3]{%
9102 &
9103 \glssubentryitem{##2}%
9104 \glstarget{##2}{\strut}\glossentrydesc{##2} &
9105 ##3\tabularnewline
9106 }%

```

Blank row between groups: The check for nogroupskip must occur outside \glsgroupskip (<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```
9107 \ifglsgroupskip
9108 \renewcommand*{\glsgroupskip}{}%
9109 \else
9110 \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
9111 \fi
9112 }
```

superragged3colborder The superragged3colborder style is like the superragged3col style, but with a border:

```
9113 \newglossarystyle{superragged3colborder}{%
```

Base it on the glostypesuperragged3col style:

```
9114 \setglossarystyle{superragged3col}%
```

Put the glossary in a supertabular environment with three columns and a horizontal line in the head and tail:

```
9115 \renewenvironment{theglossary}%
9116 {\tablehead{\hline}\tabletail{\hline}%
9117 \begin{supertabular}{|l|>{\raggedright}p{\glsdescwidth}|%
9118 >{\raggedright}p{\glspagelistwidth}|}%
9119 {\end{supertabular}}%
9120 }
```

superragged3colheader The superragged3colheader style is like the superragged3col style but with a header row:

```
9121 \newglossarystyle{superragged3colheader}{%
```

Base it on the glostypesuperragged3col style:

```
9122 \setglossarystyle{superragged3col}%
```

Put the glossary in a supertabular environment with three columns, a header and no tail:

```
9123 \renewenvironment{theglossary}%
9124 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
9125 \bfseries\pagelistname\tabularnewline}\tabletail{}%
9126 \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}%
9127 >{\raggedright}p{\glspagelistwidth}}%
9128 {\end{supertabular}}%
9129 }
```

superragged3colheaderborder The superragged3colheaderborder style is like the superragged3col style but with a header and border:

```
9130 \newglossarystyle{superragged3colheaderborder}{%
```

Base it on the glostypesuperragged3colborder style:

```
9131 \setglossarystyle{superragged3colborder}%
```

Put the glossary in a supertabular environment with three columns, a header with horizontal lines and a horizontal line in the tail:

```
9132 \renewenvironment{theglossary}%
9133 {\tablehead{\hline
```

```

9134      \bfseries\entryname&\bfseries\descriptionname&
9135      \bfseries\pagelistname\tabularnewline\hline}%
9136      \tabletail{\hline}%
9137      \begin{supertabular}{|l|>{\raggedright}p{\glsdescwidth}|%
9138      >{\raggedright}p{\glspagelistwidth}|}%
9139      {\end{supertabular}}%
9140 }

```

superragged4col The altsuperragged4col glossary style is like altsuper4col style in the package but uses ragged right formatting in the description and page list columns.

```

9141 \newglossarystyle{altsuperragged4col}{%

```

Put the glossary in a supertabular environment with four columns and no head or tail:

```

9142 \renewenvironment{theglossary}%
9143 {\tablehead{\tabletail}{}%
9144 \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}l%
9145 >{\raggedright}p{\glspagelistwidth}}}%
9146 {\end{supertabular}}%

```

Do nothing at the start of the table:

```

9147 \renewcommand*{\glossaryheader}{}%

```

No group headings:

```

9148 \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:

```

9149 \renewcommand{\glossentry}[2]{%
9150 \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
9151 \glossentrydesc{##1} &
9152 \glossentrysymbol{##1} & ##2\tabularnewline
9153 }%

```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```

9154 \renewcommand{\subglossentry}[3]{%
9155 &
9156 \glssubentryitem{##2}%
9157 \glstarget{##2}{\strut}\glossentrydesc{##2} &
9158 \glossentrysymbol{##2} & ##3\tabularnewline
9159 }%

```

Blank row between groups: The check for nogroupskip must occur outside \glsgroupskip (<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```

9160 \ifglsgroupskip
9161 \renewcommand*{\glsgroupskip}{}%
9162 \else
9163 \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
9164 \fi
9165 }

```


agged4colheader The altsuperragged4colheader style is like the altsuperragged4col style but with a header row.

```
9166 \newglossarystyle{altsuperragged4colheader}{%
```

Base it on the glostylealtsuperragged4col style:

```
9167 \setglossarystyle{altsuperragged4col}{%
```

Put the glossary in a supertabular environment with four columns, a header and no tail:

```
9168 \renewenvironment{theglossary}%
9169 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
9170 \bfseries\symbolname &
9171 \bfseries\pagelistname\tabularnewline}\tabletail{}}%
9172 \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}l%
9173 >{\raggedright}p{\glspagelistwidth}}}%
9174 {\end{supertabular}}}%
9175 }
```

agged4colborder The altsuperragged4colborder style is like the altsuperragged4col style but with a border.

```
9176 \newglossarystyle{altsuperragged4colborder}{%
```

Base it on the glostylealtsuperragged4col style:

```
9177 \setglossarystyle{altsuper4col}{%
```

Put the glossary in a supertabular environment with four columns and a horizontal line in the head and tail:

```
9178 \renewenvironment{theglossary}%
9179 {\tablehead{\hline}\tabletail{\hline}%
9180 \begin{supertabular}%
9181 {ll>{\raggedright}p{\glsgdescwidth}ll}%
9182 >{\raggedright}p{\glspagelistwidth}l}}%
9183 {\end{supertabular}}}%
9184 }
```

colheaderborder The altsuperragged4colheaderborder style is like the altsuperragged4col style but with a header and border.

```
9185 \newglossarystyle{altsuperragged4colheaderborder}{%
```

Base it on the glostylealtsuperragged4col style:

```
9186 \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:

```
9187 \renewenvironment{theglossary}%
9188 {\tablehead{\hline
9189 \bfseries\entryname &
9190 \bfseries\descriptionname &
9191 \bfseries\symbolname &
9192 \bfseries\pagelistname\tabularnewline\hline}%
9193 \tabletail{\hline}%
9194 \begin{supertabular}%
9195 {ll>{\raggedright}p{\glsgdescwidth}ll}%
9196 >{\raggedright}p{\glspagelistwidth}l}}%
```

```

9197     {\end{supertabular}}}%
9198 }

```

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.

```

9199 \ProvidesPackage{glossary-tree}[2017/08/10 v4.31 (NLCT)]

```

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```

9200 \providecommand{\indexspace}{%
9201   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
9202 }

```

`\glstreenamefmt` Format used to display the name in the tree styles. (This may be counteracted by `\glslnamefont`.) This command was previously also used to format the group headings.

```

9203 \newcommand*{\glstreenamefmt}[1]{\textbf{#1}}

```

`\glstreegroupheaderfmt` Format used to display the group header in the tree styles. Before v4.22, `\glstreenamefmt` was used for the group header, so the default definition uses that to help maintain backward-compatibility, since in previous versions redefining `\glstreenamefmt` would've also affected the group headings.

```

9204 \newcommand*{\glstreegroupheaderfmt}[1]{\glstreenamefmt{#1}}

```

`\glstreenavigationfmt` Format used to display the navigation header in the tree styles.

```

9205 \newcommand*{\glstreenavigationfmt}[1]{\glstreenamefmt{#1}}

```

Allow the user to adjust the index style without disturbing the index.

`\glstreeitem` Top level item used in index style.

```

9206 \ifdef\@idxitem
9207 {\newcommand{\glstreeitem}{\@idxitem}}
9208 {\newcommand{\glstreeitem}{\par\hangindent40\p@}}

```

`\glstreesubitem` Level 1 item used in index style.

```

9209 \ifdef\subitem
9210 {\let\glstreesubitem\subitem}
9211 {\newcommand\glstreesubitem{\glstreeitem\hspace*{20\p@}}}

```

`\glstreesubsubitem` Level 1 item used in index style.

```

9212 \ifdef\subsubitem
9213 {\let\glstreesubsubitem\subsubitem}
9214 {\newcommand\glstreesubsubitem{\glstreeitem\hspace*{30\p@}}}

```

`\glstreepredesc` Allow the user to adjust the space before the description (except for the `almtree` style).

```

9215 \newcommand{\glstreepredesc}{\space}

```

treechildpredesc Allow the user to adjust the space before the description for sub-entries (except for the treeoname and alttree style).

```
9216 \newcommand{\glstreechildpredesc}{\space}
```

index The index glossary style is similar in style to the way indices are usually typeset using `\item`, `\subitem` and `\subsubitem`. The entry name is set in bold. If an entry has a symbol, it is placed in brackets after the name. Then the description is displayed, followed by the number list. This style allows up to three levels.

```
9217 \newglossarystyle{index}{%
```

Set the paragraph indentation and skip and define `\item` to be the same as that used by theindex:

```
9218 \renewenvironment{theglossary}%
9219 {\setlength{\parindent}{0pt}%
9220 \setlength{\parskip}{0pt plus 0.3pt}%
9221 \let\item\glstreeitem
9222 \let\subitem\glstreesubitem
9223 \let\subsubitem\glstreesubsubitem
9224 }%
```

```
9225 {\par}%
```

Do nothing at the start of the environment:

```
9226 \renewcommand*{\glossaryheader}{}%
```

No group headers:

```
9227 \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.

```
9228 \renewcommand*{\glossentry}[2]{%
9229 \item\glssentryitem{##1}\glstreenamfmt{\glstarget{##1}{\glossentryname{##1}}}%
9230 \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
9231 \glstreepredesc \glossentrydesc{##1}\glspostdescription\space ##2%
9232 }%
```

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.

```
9233 \renewcommand{\subglossentry}[3]{%
9234 \ifcase##1\relax
9235 % level 0
9236 \item
9237 \or
9238 % level 1
9239 \subitem
9240 \glssubentryitem{##2}%
9241 \else
9242 % all other levels
```

```

9243     \subsubitem
9244     \fi
9245     \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
9246     \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
9247     \glstreechildpredesc\glossentrydesc{##2}\glspostdescription\space ##3%
9248 }%

```

Vertical gap between groups is the same as that used by indices:

```

9249 \renewcommand*{\glsgroupskip}{\ifglsgnogroupskip\else\indexspace\fi}

```

indexgroup The indexgroup style is like the index style but has headings.

```

9250 \newglossarystyle{indexgroup}{%

```

Base it on the glostyleindex style:

```

9251 \setglossarystyle{index}%

```

Add a heading for each group. This puts the group's title in bold followed by a vertical gap.

```

9252 \renewcommand*{\glsgroupheading}[1]{%
9253 \item\glstreegroupheaderfmt{\glsgrouptitle{##1}}%
9254 \indexspace
9255 }%
9256 }

```

indexhypergroup The indexhypergroup style is like the indexgroup style but has hyper navigation.

```

9257 \newglossarystyle{indexhypergroup}{%

```

Base it on the glostyleindex style:

```

9258 \setglossarystyle{index}%

```

Put navigation links to the groups at the start of the glossary:

```

9259 \renewcommand*{\glossaryheader}{%
9260 \item\glstreenavigationfmt{\glsgroupnavigation}\indexspace}%

```

Add a heading for each group (with a target). The group's title is in bold followed by a vertical gap.

```

9261 \renewcommand*{\glsgroupheading}[1]{%
9262 \item\glstreegroupheaderfmt
9263 {\glsgroupnavigationtarget{##1}{\glsgrouptitle{##1}}}%
9264 \indexspace}%
9265 }

```

tree The tree glossary style is similar in style to the index style, but can have arbitrary levels.

```

9266 \newglossarystyle{tree}{%

```

Set the paragraph indentation and skip:

```

9267 \renewenvironment{theglossary}%
9268 {\setlength{\parindent}{0pt}%
9269 \setlength{\parskip}{0pt plus 0.3pt}}%
9270 {}%

```

Do nothing at the start of the theglossary environment:

```

9271 \renewcommand*{\glossaryheader}{}%

```

No group headings:

```
9272 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries: name in bold, followed by symbol in brackets (if it exists), the description and the page list:

```
9273 \renewcommand{\glossentry}[2]{%
9274   \hangindent0pt\relax
9275   \parindent0pt\relax
9276   \glstentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
9277   \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
9278   \glstreepredesc\glossentrydesc{##1}\glspostdescription\space##2\par
9279 }%
```

Sub entries: level $\langle n \rangle$ is indented by $\langle n \rangle$ times `\glstreeindent`. The name is in bold, followed by the symbol in brackets (if it exists), the description and the page list.

```
9280 \renewcommand{\subglossentry}[3]{%
9281   \hangindent##1\glstreeindent\relax
9282   \parindent##1\glstreeindent\relax
9283   \ifnum##1=1\relax
9284     \glssubentryitem{##2}%
9285     \fi
9286     \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
9287     \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
9288     \glstreechildpredesc\glossentrydesc{##2}\glspostdescription\space ##3\par
9289 }%
```

Vertical gap between groups is the same as that used by indices:

```
9290 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}}
```

treegroup Like the tree style but the glossary groups have headings.

```
9291 \newglossarystyle{treegroup}{%
```

Base it on the `glostyletree` style:

```
9292 \setglossarystyle{tree}%
```

Each group has a heading (in bold) followed by a vertical gap):

```
9293 \renewcommand{\glsgroupheading}[1]{\par
9294   \noindent\glstreegroupheaderfmt{\glsgrouptitle{##1}}\par
9295   \indexspace}%
9296 }
```

treehypergroup The `treehypergroup` style is like the `treegroup` style, but has a set of links to the groups at the start of the glossary.

```
9297 \newglossarystyle{treehypergroup}{%
```

Base it on the `glostyletree` style:

```
9298 \setglossarystyle{tree}%
```

Put navigation links to the groups at the start of the `theglossary` environment:

```
9299 \renewcommand*{\glossaryheader}{%
9300   \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```

9301 \renewcommand*{\glsgroupheading}[1]{%
9302   \par\noindent
9303   \glstreegroupheaderfmt
9304   {\glshnavhypertarget{##1}{\glsgrouptitle{##1}}}\par
9305   \indexspace}%
9306 }
```

`\glstreeindent` Length governing left indent for each level of the tree style.

```

9307 \newlength\glstreeindent
9308 \setlength{\glstreeindent}{10pt}
```

`treenoname` The `treenoname` glossary style is like the tree style, but doesn't print the name or symbol for sub-levels.

```

9309 \newglossarystyle{treenoname}{%
```

Set the paragraph indentation and skip:

```

9310 \renewenvironment{theglossary}%
9311   {\setlength{\parindent}{0pt}%
9312   \setlength{\parskip}{0pt plus 0.3pt}}%
9313   {}%
```

No header:

```

9314 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```

9315 \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.

```

9316 \renewcommand{\glossentry}[2]{%
9317   \hangindent0pt\relax
9318   \parindent0pt\relax
9319   \glstryitem{##1}\glstreenamfmt{\glstarget{##1}{\glossentryname{##1}}}%
9320   \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
9321   \glstreepredesc\glossentrydesc{##1}\glspostdescription\space##2\par
9322   }%
```

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.

```

9323 \renewcommand{\subglossentry}[3]{%
9324   \hangindent##1\glstreeindent\relax
9325   \parindent##1\glstreeindent\relax
9326   \ifnum##1=1\relax
9327     \glssubentryitem{##2}%
9328   \fi
9329   \glstarget{##2}{\strut}%
9330   \glossentrydesc{##2}\glspostdescription\space##3\par
9331   }%
```

Vertical gap between groups is the same as that used by indices:

```
9332 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
9333 }
```

treenonamegroup Like the `treenoname` style but the glossary groups have headings.

```
9334 \newglossarystyle{treenonamegroup}{%
```

Base it on the `glostyletreenoname` style:

```
9335 \setglossarystyle{treenoname}%
```

Give each group a heading:

```
9336 \renewcommand{\glsgroupheading}[1]{\par
9337 \noindent\glstreegroupheaderfmt
9338 {\glsgrouptitle{##1}}\par\indexspace}%
9339 }
```

treenonamehypergroup The `treenonamehypergroup` style is like the `treenonamegroup` style, but has a set of links to the groups at the start of the glossary.

```
9340 \newglossarystyle{treenonamehypergroup}{%
```

Base it on the `glostyletreenoname` style:

```
9341 \setglossarystyle{treenoname}%
```

Put navigation links to the groups at the start of the `theglossary` environment:

```
9342 \renewcommand*{\glossaryheader}{%
9343 \par\noindent\glstreenavigationfmt{\glstravel}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
9344 \renewcommand*{\glsgroupheading}[1]{%
9345 \par\noindent
9346 \glstreegroupheaderfmt
9347 {\glstravelhypertarget{##1}{\glsgrouptitle{##1}}}\par
9348 \indexspace}%
9349 }
```

esttoplevelname Find the widest name over all parentless entries in the given glossary or glossaries.

```
9350 \newrobustcmd*{\glsfindwidesttoplevelname}[1][\@glo@types]{%
9351 \dimen@=0pt\relax
9352 \gls@tmplen=0pt\relax
9353 \forallglossaries[#1]{\@gls@type}%
9354 {%
9355 \forallsentries[\@gls@type]{\@glo@label}%
9356 {%
9357 \ifglshasparent{\@glo@label}%
9358 }%
9359 {%
9360 \settowidth{\dimen@}%
9361 {\glstreenamfmt{\glstryname{\@glo@label}}}%
9362 \ifdim\dimen@>\gls@tmplen
9363 \gls@tmplen=\dimen@
```

```

9364         \letcs{\@glswidestname}{glo@\glsetoklabel{\@glo@label}@name}%
9365         \fi
9366     }%
9367 }%
9368 }%
9369 }

```

`\glsetwidest` `\glsetwidest[<level>]{<text>}` sets the widest text for the given level. It is used by the alt-tree glossary styles to determine the indentation of each level.

```

9370 \newcommand*{\glsetwidest}[2][0]{%
9371     \expandafter\def\csname @glswidestname\romannumeral#1\endcsname{%
9372         #2}%
9373 }

```

`\@glswidestname` Initialise `\@glswidestname`.

```

9374 \newcommand*{\@glswidestname}{}

```

`\glstreenamebox` Used by the alttree style to create the box for the name and associated information.

```

9375 \newcommand*{\glstreenamebox}[2]{%
9376     \makebox[#1][l]{#2}%
9377 }

```

alttree The alttree glossary style is similar in style to the tree style, but the indentation is obtained from the width of `\@glswidestname` which is set using `\glsetwidest`.

```

9378 \newglossarystyle{alttree}{%

```

Redefine theglossary environment.

```

9379     \renewenvironment{theglossary}%
9380     {\def\@gls@prevlevel{-1}%
9381         \mbox{}\par}%
9382     {\par}%

```

Set the header and group headers to nothing.

```

9383     \renewcommand*{\glossaryheader}{}%
9384     \renewcommand*{\glsgroupheading}[1]{}%

```

Redefine the way that the level 0 entries are displayed.

```

9385     \renewcommand{\glossentry}[2]{%
9386         \ifnum\@gls@prevlevel=0\relax
9387         \else

```

Find out how big the indentation should be by measuring the widest entry.

```

9388         \settowidth{\glstreeindent}{\glstreenamefmt{\@glswidestname\space}}%
9389         \fi

```

Set the hangindent and paragraph indent.

```

9390         \hangindent\glstreeindent
9391         \parindent\glstreeindent

```

Put the name to the left of the paragraph block.

```

9392         \makebox[0pt][r]{\glstreenamebox{\glstreeindent}}%
9393         \glstryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%

```


If the symbol is missing, ignore it, otherwise put it in brackets.

```
9394 \ifglshassymbol{##1}{(\glossentrysymbol{##1})\space}{}%
```

Do the description followed by the description terminator and location list.

```
9395 \glossentrydesc{##1}\glspostdescription \space ##2\par
```

Set the previous level to 0.

```
9396 \def\@gls@prevlevel{0}%  
9397 }%
```

Redefine the way sub-entries are displayed.

```
9398 \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.

```
9399 \ifnum##1=1\relax  
9400 \glssubentryitem{##2}%  
9401 \fi
```

If the level hasn't changed, keep the same settings, otherwise adjust `\glstreeindent` accordingly.

```
9402 \ifnum\@gls@prevlevel=##1\relax  
9403 \else
```

Compute the widest entry for this level, or for level 0 if not defined for this level. Store in `\gls@tmplen`

```
9404 \@ifundefined{@glswidestname\romannumeral##1}{%  
9405 \settowidth{\gls@tmplen}{\glstreenamefmt{\@glswidestname\space}}{%  
9406 \settowidth{\gls@tmplen}{\glstreenamefmt{%  
9407 \csname @glswidestname\romannumeral##1\endcsname\space}}}%
```

Determine if going up or down a level

```
9408 \ifnum\@gls@prevlevel<##1\relax
```

Depth has increased, so add the width of the widest entry to `\glstreeindent`.

```
9409 \setlength\glstreeindent\gls@tmplen  
9410 \addtolength\glstreeindent\parindent  
9411 \parindent\glstreeindent  
9412 \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`.

```
9413 \@ifundefined{@glswidestname\romannumeral\@gls@prevlevel}{%  
9414 \settowidth{\glstreeindent}{\glstreenamefmt{%  
9415 \@glswidestname\space}}}%  
9416 \settowidth{\glstreeindent}{\glstreenamefmt{%  
9417 \csname @glswidestname\romannumeral\@gls@prevlevel  
9418 \endcsname\space}}}%
```

Subtract this length from the previous level's paragraph indent and set to `\glstreeindent`.

```
9419 \addtolength\parindent{-\glstreeindent}%
```

```

9420      \setlength\glstreeindent\parindent
9421      \fi
9422      \fi
    Set the hanging indentation.
9423      \hangindent\glstreeindent
    Put the name to the left of the paragraph block
9424      \makebox[0pt][r]{\glstreenamebox{\gls@tmplen}{%
9425        \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}}}%
    If the symbol is missing, ignore it, otherwise put it in brackets.
9426      \ifglshassymbol{##2}{(\glossentrysymbol{##2})\space}{}%
    Do the description followed by the description terminator and location list.
9427      \glossentrydesc{##2}\glspostdescription\space ##3\par
    Set the previous level macro to the current level.
9428      \def\@gls@prevlevel{##1}%
9429  }%
    Vertical gap between groups is the same as that used by indices:
9430  \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
9431 }

```

alttreegroup Like the **alttree** style but the glossary groups have headings.

```

9432 \newglossarystyle{alttreegroup}{%
    Base it on the glostylealttree style:
9433  \setglossarystyle{alttree}%
    Give each group a heading.
9434  \renewcommand{\glsgroupheading}[1]{\par
9435    \def\@gls@prevlevel{-1}%
9436    \hangindent0pt\relax
9437    \parindent0pt\relax
9438    \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}%
9439    \par\indexspace}%
9440 }

```

alttreehypergroup The **alttreehypergroup** style is like the **alttreegroup** style, but has a set of links to the groups at the start of the glossary.

```

9441 \newglossarystyle{alttreehypergroup}{%
    Base it on the glostylealttree style:
9442  \setglossarystyle{alttree}%
    Put the navigation links in the header
9443  \renewcommand*{\glossaryheader}{%
9444    \par
9445    \def\@gls@prevlevel{-1}%
9446    \hangindent0pt\relax
9447    \parindent0pt\relax
9448    \glstreenavigationfmt{\glsnavigation}\par\indexspace}%

```

Put a hypertarget at the start of each group

```
9449 \renewcommand*{\glsgroupheading}[1]{%
9450   \par
9451   \def\@gls@prevlevel{-1}%
9452   \hangindent0pt\relax
9453   \parindent0pt\relax
9454   \glstreegroupheaderfmt
9455   {\@glsnavhypertarget{##1}{\@glsgetgrouptitle{##1}}}\par
9456   \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.

```
9457 \NeedsTeXFormat{LaTeX2e}
9458 \ProvidesPackage{glossaries-compatible-207}[2017/08/10 v4.31 (NLCT)]
```

AddXdyAttribute Adds an attribute in old format.

```
9459 \ifglsxindy
9460   \renewcommand*\GlsAddXdyAttribute[1]{%
9461     \edef\@xdyattributes{\@xdyattributes ^^J \string"#1\string"}%
9462     \expandafter\toks@\expandafter{\@xdylocref}%
9463     \edef\@xdylocref{\the\toks@ ^^J}%
9464     (markup-locref
9465     :open \string"\string~n\string\setentrycounter
9466       {\noexpand\glscounter}%
9467       \expandafter\string\csname#1\endcsname
9468       \expandafter\@gobble\string\{\string" ^^J
9469       :close \string"\expandafter\@gobble\string\}\string" ^^J
9470       :attr \string"#1\string"))}}
```

Only has an effect before \writeist:

```
9471 \fi
```

sAddXdyCounters

```
9472 \renewcommand*\GlsAddXdyCounters[1]{%
9473   \GlossariesWarning{\string\GlsAddXdyCounters\space not available
9474     in compatibility mode.}%
9475 }
```

Add predefined attributes

```
9476 \GlsAddXdyAttribute{glsnumberformat}
9477 \GlsAddXdyAttribute{textrm}
9478 \GlsAddXdyAttribute{textsf}
9479 \GlsAddXdyAttribute{texttt}
9480 \GlsAddXdyAttribute{textbf}
9481 \GlsAddXdyAttribute{textmd}
9482 \GlsAddXdyAttribute{textit}
9483 \GlsAddXdyAttribute{textup}
9484 \GlsAddXdyAttribute{textsl}
```

```

9485 \GlsAddXdyAttribute{textsc}
9486 \GlsAddXdyAttribute{emph}
9487 \GlsAddXdyAttribute{glshypernumber}
9488 \GlsAddXdyAttribute{hyperrm}
9489 \GlsAddXdyAttribute{hypersf}
9490 \GlsAddXdyAttribute{hypertt}
9491 \GlsAddXdyAttribute{hyperbf}
9492 \GlsAddXdyAttribute{hypermd}
9493 \GlsAddXdyAttribute{hyperit}
9494 \GlsAddXdyAttribute{hyperup}
9495 \GlsAddXdyAttribute{hypersl}
9496 \GlsAddXdyAttribute{hypersc}
9497 \GlsAddXdyAttribute{hyperemph}

```

sAddXdyLocation Restore v2.07 definition:

```

9498 \ifglxindy
9499 \renewcommand*{\GlsAddXdyLocation}[2]{%
9500 \edef\xdyuserlocationdefs{%
9501 \xdyuserlocationdefs ^^J%
9502 (define-location-class \string"#1\string"^^J\space\space
9503 \space(#2))
9504 }%
9505 \edef\xdyuserlocationnames{%
9506 \xdyuserlocationnames^^J\space\space\space
9507 \string"#1\string"}%
9508 }
9509 \fi

```

\@do@wrglossary

```

9510 \renewcommand{\@do@wrglossary}[1]{%
  Determine whether to use xindy or makeindex syntax
9511 \ifglxindy
  Need to determine if the formatting information starts with a ( or ) indicating a range.
9512 \expandafter\@glo@check@mkidxrangechar\@glsnumberformat\@nil
9513 \def\@glo@range{}%
9514 \expandafter\if\@glo@prefix(\relax
9515 \def\@glo@range{:open-range}%
9516 \else
9517 \expandafter\if\@glo@prefix)\relax
9518 \def\@glo@range{:close-range}%
9519 \fi
9520 \fi

  Get the location and escape any special characters
9521 \protected@edef\@glslocref{\theglsentrycounter}%
9522 \@gls@checkmkidxchars\@glslocref

  Write to the glossary file using xindy syntax.
9523 \glossary[\csname glo@#1@type\endcsname]{%

```

```

9524 (indexentry :tkey (\csname glo@#1@index\endcsname)
9525   :locoref \string"\@glslocoref\string" %
9526   :attr \string"\@glo@suffix\string" \@glo@range
9527 )
9528 }%
9529 \else

```

Convert the format information into the format required for makeindex

```

9530 \@set@glo@numformat\@glo@numfmt\@gls@counter\@glsnumberformat

```

Write to the glossary file using makeindex syntax.

```

9531 \glossary[\csname glo@#1@type\endcsname]{%
9532 \string\glossaryentry{\csname glo@#1@index\endcsname
9533   \@gls@encapchar\@glo@numfmt}{\theglsentrycounter}}%
9534 \fi
9535 }

```

t@glo@numformat Only had 3 arguments in v2.07

```

9536 \def\@set@glo@numformat#1#2#3{%
9537   \expandafter\@glo@check@mkidxrangechar#3\@nil
9538   \protected@edef#1{%
9539     \@glo@prefix setentrycounter[] {#2}%
9540     \expandafter\string\csname\@glo@suffix\endcsname
9541   }%
9542   \@gls@checkmkidxchars#1%
9543 }

```

\writeist Redefine \writeist back to the way it was in v2.07, but change \istfile to \glswrite.

```

9544 \ifglxsindy
9545   \def\writeist{%
9546     \openout\glswrite=\istfilename
9547     \write\glswrite{;; xindy style file created by the glossaries
9548       package in compatible-2.07 mode}%
9549     \write\glswrite{;; for document '\jobname' on
9550       \the\year-\the\month-\the\day}%
9551     \write\glswrite{^^J; required styles^^J}
9552     \@for\@xdystyle:=\@xdyrequiredstyles\do{%
9553       \ifx\@xdystyle\@empty
9554       \else
9555         \protected@write\glswrite{{(require
9556           \string"\@xdystyle.xdy\string")}}%
9557       \fi
9558     }%
9559     \write\glswrite{^^J%
9560       ; list of allowed attributes (number formats)^^J}%
9561     \write\glswrite{(define-attributes ((\@xdyattributes)))}%
9562     \write\glswrite{^^J; user defined alphabets^^J}%
9563     \write\glswrite{\@xdyuseralphabets}%
9564     \write\glswrite{^^J; location class definitions^^J}%
9565     \protected@edef\@gls@roman{\@roman{0}\string"

```

```

9566     \string"roman-numbers-lowercase\string" :sep \string"}}%
9567 \@onelevel@sanitize\@gls@roman
9568 \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
9569     :sep \string"}}%
9570 \@onelevel@sanitize\@tmp
9571 \ifx\@tmp\@gls@roman
9572     \write\glswrite{(define-location-class
9573         \string"roman-page-numbers\string"^^J\space\space\space
9574         (\string"roman-numbers-lowercase\string")
9575         :min-range-length \@glsminrange)}}%
9576 \else
9577     \write\glswrite{(define-location-class
9578         \string"roman-page-numbers\string"^^J\space\space\space
9579         (:sep "\@gls@roman")
9580         :min-range-length \@glsminrange)}}%
9581 \fi
9582 \write\glswrite{(define-location-class
9583     \string"Roman-page-numbers\string"^^J\space\space\space
9584     (\string"roman-numbers-uppercase\string")
9585     :min-range-length \@glsminrange)}}%
9586 \write\glswrite{(define-location-class
9587     \string"arabic-page-numbers\string"^^J\space\space\space
9588     (\string"arabic-numbers\string")
9589     :min-range-length \@glsminrange)}}%
9590 \write\glswrite{(define-location-class
9591     \string"alpha-page-numbers\string"^^J\space\space\space
9592     (\string"alpha\string")
9593     :min-range-length \@glsminrange)}}%
9594 \write\glswrite{(define-location-class
9595     \string"Alpha-page-numbers\string"^^J\space\space\space
9596     (\string"ALPHA\string")
9597     :min-range-length \@glsminrange)}}%
9598 \write\glswrite{(define-location-class
9599     \string"Appendix-page-numbers\string"^^J\space\space\space
9600     (\string"ALPHA\string"
9601     :sep \string"\@glsAlphacompositor\string"
9602     \string"arabic-numbers\string")
9603     :min-range-length \@glsminrange)}}%
9604 \write\glswrite{(define-location-class
9605     \string"arabic-section-numbers\string"^^J\space\space\space
9606     (\string"arabic-numbers\string"
9607     :sep \string"\glscompositor\string"
9608     \string"arabic-numbers\string")
9609     :min-range-length \@glsminrange)}}%
9610 \write\glswrite{^^J; user defined location classes}%
9611 \write\glswrite{\@xdyuserlocationdefs}%
9612 \write\glswrite{^^J; define cross-reference class^^J}%
9613 \write\glswrite{(define-crossref-class \string"see\string"
9614     :unverified )}%

```

```

9615 \write\glswrite{(markup-crossref-list
9616 :class \string"see\string"^^J\space\space\space
9617 :open \string"\string\glseeformat\string"
9618 :close \string"{}\string")}%
9619 \write\glswrite{^^J; define the order of the location classes}%
9620 \write\glswrite{(define-location-class-order
9621 (\@xdylocationclassorder))}%
9622 \write\glswrite{^^J; define the glossary markup^^J}%
9623 \write\glswrite{(markup-index^^J\space\space\space
9624 :open \string"\string
9625 \glossarysection[\string\glossarytoctitle]{\string
9626 \glossarytitle}\string\glossarypreamble\string~n\string\begin
9627 {theglossary}\string\glossaryheader\string~n\string" ^^J\space
9628 \space\space:close \string"\expandafter\@gobble
9629 \string%\string~n\string
9630 \end{theglossary}\string\glossarypostamble
9631 \string~n\string" ^^J\space\space\space
9632 :tree)}}%
9633 \write\glswrite{(markup-letter-group-list
9634 :sep \string"\string\glsgroupskip\string~n\string")}%
9635 \write\glswrite{(markup-indexentry
9636 :open \string"\string\relax \string\glresetentrylist
9637 \string~n\string")}%
9638 \write\glswrite{(markup-locclass-list :open
9639 \string"\glsoopenbrace\string\glossaryentrynumbers
9640 \glsoopenbrace\string\relax\space \string"^^J\space\space\space
9641 :sep \string", \string"
9642 :close \string"\glsclosebrace\glsclosebrace\string")}%
9643 \write\glswrite{(markup-locref-list
9644 :sep \string"\string\delimN\space\string")}%
9645 \write\glswrite{(markup-range
9646 :sep \string"\string\delimR\space\string")}%
9647 \@onelevel@sanitize\gls@suffixF
9648 \@onelevel@sanitize\gls@suffixFF
9649 \ifx\gls@suffixF\@empty
9650 \else
9651 \write\glswrite{(markup-range
9652 :close "\gls@suffixF" :length 1 :ignore-end)}%
9653 \fi
9654 \ifx\gls@suffixFF\@empty
9655 \else
9656 \write\glswrite{(markup-range
9657 :close "\gls@suffixFF" :length 2 :ignore-end)}%
9658 \fi
9659 \write\glswrite{^^J; define format to use for locations^^J}%
9660 \write\glswrite{\@xdylocref}%
9661 \write\glswrite{^^J; define letter group list format^^J}%
9662 \write\glswrite{(markup-letter-group-list
9663 :sep \string"\string\glsgroupskip\string~n\string")}%

```



```

9664 \write\glswrite{^^J; letter group headings^^J}%
9665 \write\glswrite{(markup-letter-group
9666   :open-head \string"\string\glsgroupheading
9667   \glsoopenbrace\string"^^J\space\space\space
9668   :close-head \string"\glsclosebrace\string")}%
9669 \write\glswrite{^^J; additional letter groups^^J}%
9670 \write\glswrite{\@xdylettergroups}%
9671 \write\glswrite{^^J; additional sort rules^^J}
9672 \write\glswrite{\@xdysortrules}%
9673 \noist}
9674 \else
9675 \edef\@gls@actualchar{\string?}
9676 \edef\@gls@encapchar{\string|}
9677 \edef\@gls@levelchar{\string!}
9678 \edef\@gls@quotechar{\string"}
9679 \def\writeist{\relax
9680   \openout\glswrite=\istfilename
9681   \write\glswrite{\expandafter\@gobble\string\% makeindex style file
9682     created by the glossaries package}
9683   \write\glswrite{\expandafter\@gobble\string\% for document
9684     '\jobname' on \the\year-\the\month-\the\day}
9685   \write\glswrite{actual '\@gls@actualchar'}
9686   \write\glswrite{encap '\@gls@encapchar'}
9687   \write\glswrite{level '\@gls@levelchar'}
9688   \write\glswrite{quote '\@gls@quotechar'}
9689   \write\glswrite{keyword \string"\string\glossaryentry\string"}
9690   \write\glswrite{preamble \string"\string\glossarysection[\string
9691     \glossarytoctitle]{\string\glossarytitle}\string
9692     \glossarypreamble\string\n\string\begin{theglossary}\string
9693     \glossaryheader\string\n\string"}
9694   \write\glswrite{postamble \string"\string%\string\n\string
9695     \end{theglossary}\string\glossarypostamble\string\n
9696     \string"}
9697   \write\glswrite{group_skip \string"\string\glsgroupskip\string\n
9698     \string"}
9699   \write\glswrite{item_0 \string"\string%\string\n\string"}
9700   \write\glswrite{item_1 \string"\string%\string\n\string"}
9701   \write\glswrite{item_2 \string"\string%\string\n\string"}
9702   \write\glswrite{item_01 \string"\string%\string\n\string"}
9703   \write\glswrite{item_x1
9704     \string"\string\relax \string\glsresetentrylist\string\n
9705     \string"}
9706   \write\glswrite{item_12 \string"\string%\string\n\string"}
9707   \write\glswrite{item_x2
9708     \string"\string\relax \string\glsresetentrylist\string\n
9709     \string"}
9710   \write\glswrite{delim_0 \string"\string\{\string
9711     \glossaryentrynumbers\string\{\string\relax \string"}
9712   \write\glswrite{delim_1 \string"\string\{\string

```

```

9713      \glossaryentrynumbers\string\{\string\relax \string}
9714      \write\glswrite{delim_2 \string"\string\{\string
9715      \glossaryentrynumbers\string\{\string\relax \string}
9716      \write\glswrite{delim_t \string"\string\}\string\}\string}
9717      \write\glswrite{delim_n \string"\string\delimN \string}
9718      \write\glswrite{delim_r \string"\string\delimR \string}
9719      \write\glswrite{headings_flag 1}
9720      \write\glswrite{heading_prefix
9721      \string"\string\glsgroupheading\string\{\string}
9722      \write\glswrite{heading_suffix
9723      \string"\string\}\string\relax
9724      \string\glsgroupresetentrylist \string}
9725      \write\glswrite{symhead_positive \string"glssymbols\string}
9726      \write\glswrite{numhead_positive \string"glslnumbers\string}
9727      \write\glswrite{page_compositor \string"glscpositor\string}
9728      \@gls@escbsdq\gls@suffixF
9729      \@gls@escbsdq\gls@suffixFF
9730      \ifx\gls@suffixF\@empty
9731      \else
9732      \write\glswrite{suffix_2p \string"\gls@suffixF\string}
9733      \fi
9734      \ifx\gls@suffixFF\@empty
9735      \else
9736      \write\glswrite{suffix_3p \string"\gls@suffixFF\string}
9737      \fi
9738      \noist
9739    }
9740  \fi

```

\noist

```

9741 \renewcommand*{\noist}{\let\writeist\relax}

```

4.2 glossaries-compatible-307

```

9742 \NeedsTeXFormat{LaTeX2e}
9743 \ProvidesPackage{glossaries-compatible-307}[2017/08/10 v4.31 (NLCT)]

```

Compatibility macros for predefined glossary styles:

`\atglossarystyle` Defines a compatibility glossary style.

```

9744 \newcommand{\compatglossarystyle}[2]{%
9745   \ifcsundef{@glscompstyle@#1}%
9746   {%
9747     \csdef{@glscompstyle@#1}{#2}%
9748   }%
9749   {%
9750     \PackageError{glossaries}{Glossary compatibility style ‘#1’ is already defined}{}%
9751   }%
9752 }

```

Backward compatible inline style.

```

9753 \compatglossarystyle{inline}{%
9754   \renewcommand{\glossaryentryfield}[5]{%
9755     \glsinlinedopostchild
9756     \gls@inlinesep
9757     \def\glo@desc{##3}%
9758     \def\@no@post@desc{\nopostdesc}%
9759     \glstentryitem{##1}\glsinlinenameformat{##1}{##2}%
9760     \ifx\glo@desc\@no@post@desc
9761       \glsinlineemptydescformat{##4}{##5}%
9762     \else
9763       \ifstrepty{##3}%
9764         {\glsinlineemptydescformat{##4}{##5}}%
9765         {\glsinlinedescformat{##3}{##4}{##5}}%
9766     \fi
9767     \ifglshaschildren{##1}%
9768     {%
9769       \glsresetsubentrycounter
9770       \glsinlineparentchildseparator
9771       \def\gls@inlinesubsep{}%
9772       \def\gls@inlinepostchild{\glsinlinepostchild}%
9773     }%
9774     {}%
9775     \def\gls@inlinesep{\glsinlineseparator}%
9776   }%

```

Sub-entries display description:

```

9777 \renewcommand{\glossarysubentryfield}[6]{%
9778   \gls@inlinesubsep%
9779   \glsinlinesubnameformat{##2}{##3}%
9780   \glssubentryitem{##2}\glsinlinesubdescformat{##4}{##5}{##6}%
9781   \def\gls@inlinesubsep{\glsinlinesubseparator}%
9782 }%
9783 }

```

Backward compatible list style.

```

9784 \compatglossarystyle{list}{%
9785   \renewcommand*{\glossaryentryfield}[5]{%
9786     \item[\glstentryitem{##1}\glstarget{##1}{##2}]
9787       ##3\glspostdescription\space ##5}%

```

Sub-entries continue on the same line:

```

9788 \renewcommand*{\glossarysubentryfield}[6]{%
9789   \glssubentryitem{##2}%
9790   \glstarget{##2}{\strut}##4\glspostdescription\space ##6.}%
9791 }

```

Backward compatible listgroup style.

```

9792 \compatglossarystyle{listgroup}{%
9793   \csuse{@glscompstyle@list}%
9794 }%

```

Backward compatible listhypergroup style.

```
9795 \compatglossarystyle{listhypergroup}{%
9796   \csuse{@glscompstyle@list}%
9797 }%
```

Backward compatible altlist style.

```
9798 \compatglossarystyle{altlist}{%
9799   \renewcommand*{\glossaryentryfield}[5]{%
9800     \item[\glstentryitem{##1}\glstarget{##1}{##2}]%
9801       \mbox{}\par\nobreak\@afterheading
9802       ##3\glspostdescription\space ##5}%
9803   \renewcommand{\glossarysubentryfield}[6]{%
9804     \par
9805     \glssubentryitem{##2}%
9806     \glstarget{##2}{\strut}##4\glspostdescription\space ##6}%
9807 }%
```

Backward compatible altlistgroup style.

```
9808 \compatglossarystyle{altlistgroup}{%
9809   \csuse{@glscompstyle@altlist}%
9810 }%
```

Backward compatible altlisthypergroup style.

```
9811 \compatglossarystyle{altlisthypergroup}{%
9812   \csuse{@glscompstyle@altlist}%
9813 }%
```

Backward compatible listdotted style.

```
9814 \compatglossarystyle{listdotted}{%
9815   \renewcommand*{\glossaryentryfield}[5]{%
9816     \item[\makebox[\glslistdottedwidth][l]{%
9817       \glstentryitem{##1}\glstarget{##1}{##2}%
9818       \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##3}%
9819   \renewcommand*{\glossarysubentryfield}[6]{%
9820     \item[\makebox[\glslistdottedwidth][l]{%
9821       \glssubentryitem{##2}%
9822       \glstarget{##2}{##3}%
9823       \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##4}%
9824 }%
```

Backward compatible sublistdotted style.

```
9825 \compatglossarystyle{sublistdotted}{%
9826   \csuse{@glscompstyle@listdotted}%
9827   \renewcommand*{\glossaryentryfield}[5]{%
9828     \item[\glstentryitem{##1}\glstarget{##1}{##2}]}%
9829 }%
```

Backward compatible long style.

```
9830 \compatglossarystyle{long}{%
9831   \renewcommand*{\glossaryentryfield}[5]{%
9832     \glstentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
9833   \renewcommand*{\glossarysubentryfield}[6]{%
9834     \glssubentryitem{##2}\glstarget{##2}{##3} & ##4\glspostdescription\space ##5\\}%
9835 }%
```

```

9834      &
9835      \glssubentryitem{##2}%
9836      \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
9837 }%

```

Backward compatible longborder style.

```

9838 \compatglossarystyle{longborder}{%
9839 \csuse{@glscmpstyle@long}%
9840 }%

```

Backward compatible longheader style.

```

9841 \compatglossarystyle{longheader}{%
9842 \csuse{@glscmpstyle@long}%
9843 }%

```

Backward compatible longheaderborder style.

```

9844 \compatglossarystyle{longheaderborder}{%
9845 \csuse{@glscmpstyle@long}%
9846 }%

```

Backward compatible long3col style.

```

9847 \compatglossarystyle{long3col}{%
9848 \renewcommand*{\glossaryentryfield}[5]{%
9849 \glstarget{##1}{\strut}##4 & ##3 & ##5\\}%
9850 \renewcommand*{\glossarysubentryfield}[6]{%
9851 &
9852 \glssubentryitem{##2}%
9853 \glstarget{##2}{\strut}##4 & ##6\\}%
9854 }%

```

Backward compatible long3colborder style.

```

9855 \compatglossarystyle{long3colborder}{%
9856 \csuse{@glscmpstyle@long3col}%
9857 }%

```

Backward compatible long3colheader style.

```

9858 \compatglossarystyle{long3colheader}{%
9859 \csuse{@glscmpstyle@long3col}%
9860 }%

```

Backward compatible long3colheaderborder style.

```

9861 \compatglossarystyle{long3colheaderborder}{%
9862 \csuse{@glscmpstyle@long3col}%
9863 }%

```

Backward compatible long4col style.

```

9864 \compatglossarystyle{long4col}{%
9865 \renewcommand*{\glossaryentryfield}[5]{%
9866 \glstarget{##1}{\strut}##4 & ##3 & ##4 & ##5\\}%
9867 \renewcommand*{\glossarysubentryfield}[6]{%
9868 &
9869 \glssubentryitem{##2}%

```

9870 \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%
 9871 }%

Backward compatible long4colheader style.

9872 \compatglossarystyle{long4colheader}{%
 9873 \csuse{@glscompstyle@long4col}}%
 9874 }%

Backward compatible long4colborder style.

9875 \compatglossarystyle{long4colborder}{%
 9876 \csuse{@glscompstyle@long4col}}%
 9877 }%

Backward compatible long4colheaderborder style.

9878 \compatglossarystyle{long4colheaderborder}{%
 9879 \csuse{@glscompstyle@long4col}}%
 9880 }%

Backward compatible altlong4col style.

9881 \compatglossarystyle{altlong4col}{%
 9882 \csuse{@glscompstyle@long4col}}%
 9883 }%

Backward compatible altlong4colheader style.

9884 \compatglossarystyle{altlong4colheader}{%
 9885 \csuse{@glscompstyle@long4col}}%
 9886 }%

Backward compatible altlong4colborder style.

9887 \compatglossarystyle{altlong4colborder}{%
 9888 \csuse{@glscompstyle@long4col}}%
 9889 }%

Backward compatible altlong4colheaderborder style.

9890 \compatglossarystyle{altlong4colheaderborder}{%
 9891 \csuse{@glscompstyle@long4col}}%
 9892 }%

Backward compatible long style.

9893 \compatglossarystyle{longragged}{%
 9894 \renewcommand*{\glossaryentryfield}[5]{%
 9895 \glssentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
 9896 \tabularnewline}%
 9897 \renewcommand*{\glossarysubentryfield}[6]{%
 9898 &
 9899 \glssubentryitem{##2}%
 9900 \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
 9901 \tabularnewline}%
 9902 }%

Backward compatible longraggedborder style.

9903 \compatglossarystyle{longraggedborder}{%
 9904 \csuse{@glscompstyle@longragged}}%
 9905 }%

Backward compatible longraggedheader style.

```
9906 \compatglossarystyle{longraggedheader}{%  
9907 \csuse{@glscompstyle@longragged}%  
9908 }%
```

Backward compatible longraggedheaderborder style.

```
9909 \compatglossarystyle{longraggedheaderborder}{%  
9910 \csuse{@glscompstyle@longragged}%  
9911 }%
```

Backward compatible longragged3col style.

```
9912 \compatglossarystyle{longragged3col}{%  
9913 \renewcommand*{\glossaryentryfield}[5]{%  
9914 \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%  
9915 \renewcommand*{\glossarysubentryfield}[6]{%  
9916 &  
9917 \glssubentryitem{##2}%  
9918 \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%  
9919 }%
```

Backward compatible longragged3colborder style.

```
9920 \compatglossarystyle{longragged3colborder}{%  
9921 \csuse{@glscompstyle@longragged3col}%  
9922 }%
```

Backward compatible longragged3colheader style.

```
9923 \compatglossarystyle{longragged3colheader}{%  
9924 \csuse{@glscompstyle@longragged3col}%  
9925 }%
```

Backward compatible longragged3colheaderborder style.

```
9926 \compatglossarystyle{longragged3colheaderborder}{%  
9927 \csuse{@glscompstyle@longragged3col}%  
9928 }%
```

Backward compatible altlongragged4col style.

```
9929 \compatglossarystyle{altlongragged4col}{%  
9930 \renewcommand*{\glossaryentryfield}[5]{%  
9931 \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%  
9932 \renewcommand*{\glossarysubentryfield}[6]{%  
9933 &  
9934 \glssubentryitem{##2}%  
9935 \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%  
9936 }%
```

Backward compatible altlongragged4colheader style.

```
9937 \compatglossarystyle{altlongragged4colheader}{%  
9938 \csuse{@glscompstyle@altlong4col}%  
9939 }%
```

Backward compatible altlongragged4colborder style.

```
9940 \compatglossarystyle{altlongragged4colborder}{%
```

```

9941 \csuse{@glscompstyle@altlong4col}%
9942 }%

```

Backward compatible altlongragged4colheaderborder style.

```

9943 \compatglossarystyle{altlongragged4colheaderborder}{%
9944 \csuse{@glscompstyle@altlong4col}%
9945 }%

```

Backward compatible index style.

```

9946 \compatglossarystyle{index}{%
9947 \renewcommand*{\glossaryentryfield}[5]{%
9948 \item\glsentryitem{##1}\textbf{\glstarget{##1}{##2}}%
9949 \ifx\relax##4\relax
9950 \else
9951 \space{##4}%
9952 \fi
9953 \space ##3\glspostdescription \space ##5}%
9954 \renewcommand*{\glossarysubentryfield}[6]{%
9955 \ifcase##1\relax
9956 % level 0
9957 \item
9958 \or
9959 % level 1
9960 \subitem
9961 \glssubentryitem{##2}%
9962 \else
9963 % all other levels
9964 \subsubitem
9965 \fi
9966 \textbf{\glstarget{##2}{##3}}%
9967 \ifx\relax##5\relax
9968 \else
9969 \space{##5}%
9970 \fi
9971 \space##4\glspostdescription\space ##6}%
9972 }%

```

Backward compatible indexgroup style.

```

9973 \compatglossarystyle{indexgroup}{%
9974 \csuse{@glscompstyle@index}%
9975 }%

```

Backward compatible indexhypergroup style.

```

9976 \compatglossarystyle{indexhypergroup}{%
9977 \csuse{@glscompstyle@index}%
9978 }%

```

Backward compatible tree style.

```

9979 \compatglossarystyle{tree}{%
9980 \renewcommand{\glossaryentryfield}[5]{%
9981 \hangindent0pt\relax

```



```

9982 \parindent0pt\relax
9983 \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
9984 \ifx\relax##4\relax
9985 \else
9986 \space{##4}%
9987 \fi
9988 \space ##3\glspostdescription \space ##5\par}%
9989 \renewcommand{\glossarysubentryfield}[6]{%
9990 \hangindent##1\glstreeindent\relax
9991 \parindent##1\glstreeindent\relax
9992 \ifnum##1=1\relax
9993 \glssubentryitem{##2}%
9994 \fi
9995 \textbf{\glstarget{##2}{##3}}%
9996 \ifx\relax##5\relax
9997 \else
9998 \space{##5}%
9999 \fi
10000 \space##4\glspostdescription\space ##6\par}%
10001}%

```

Backward compatible treegroup style.

```

10002 \compatglossarystyle{treegroup}{%
10003 \csuse{@glscmpstyle@tree}%
10004}%

```

Backward compatible treehypergroup style.

```

10005 \compatglossarystyle{treehypergroup}{%
10006 \csuse{@glscmpstyle@tree}%
10007}%

```

Backward compatible treenoname style.

```

10008 \compatglossarystyle{treenoname}{%
10009 \renewcommand{\glossaryentryfield}[5]{%
10010 \hangindent0pt\relax
10011 \parindent0pt\relax
10012 \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
10013 \ifx\relax##4\relax
10014 \else
10015 \space{##4}%
10016 \fi
10017 \space ##3\glspostdescription \space ##5\par}%
10018 \renewcommand{\glossarysubentryfield}[6]{%
10019 \hangindent##1\glstreeindent\relax
10020 \parindent##1\glstreeindent\relax
10021 \ifnum##1=1\relax
10022 \glssubentryitem{##2}%
10023 \fi
10024 \glstarget{##2}{\strut}%
10025 ##4\glspostdescription\space ##6\par}%
10026}%

```

Backward compatible treenonamegroup style.

```
10027 \compatglossarystyle{treenonamegroup}{%
10028   \csuse{@glscompstyle@treenoname}%
10029 }%
```

Backward compatible treenonamehypergroup style.

```
10030 \compatglossarystyle{treenonamehypergroup}{%
10031   \csuse{@glscompstyle@treenoname}%
10032 }%
```

Backward compatible alttree style.

```
10033 \compatglossarystyle{alttree}{%
10034   \renewcommand{\glossaryentryfield}[5]{%
10035     \ifnum\@gls@prevlevel=0\relax
10036     \else
10037       \settowidth{\glstreeindent}{\textbf{\@glswidestname\space}}%
10038       \hangindent\glstreeindent
10039       \parindent\glstreeindent
10040     \fi
10041     \makebox[0pt][r]{\makebox[\glstreeindent][l]{%
10042       \glssentryitem{##1}\textbf{\glstarget{##1}{##2}}}%
10043     \ifx\relax##4\relax
10044     \else
10045       (##4)\space
10046     \fi
10047     ##3\glspostdescription \space ##5\par
10048     \def\@gls@prevlevel{0}%
10049   }%
10050   \renewcommand{\glossarysubentryfield}[6]{%
10051     \ifnum##1=1\relax
10052       \glssubentryitem{##2}%
10053     \fi
10054     \ifnum\@gls@prevlevel=##1\relax
10055     \else
10056       \@ifundefined{@glswidestname\romannumeral##1}{%
10057         \settowidth{\gls@tmplen}{\textbf{\@glswidestname\space}}{%
10058         \settowidth{\gls@tmplen}{\textbf{%
10059           \csname @glswidestname\romannumeral##1\endcsname\space}}}%
10060       \ifnum\@gls@prevlevel<##1\relax
10061         \setlength\glstreeindent\gls@tmplen
10062         \addtolength\glstreeindent\parindent
10063         \parindent\glstreeindent
10064       \else
10065         \@ifundefined{@glswidestname\romannumeral\@gls@prevlevel}{%
10066           \settowidth{\glstreeindent}{\textbf{%
10067             \@glswidestname\space}}{%
10068           \settowidth{\glstreeindent}{\textbf{%
10069             \csname @glswidestname\romannumeral\@gls@prevlevel
10070             \endcsname\space}}}%
10071           \addtolength\parindent{-\glstreeindent}%

```

```

10072         \setlength\glstreeindent\parindent
10073     \fi
10074 \fi
10075 \hangindent\glstreeindent
10076 \makebox[0pt][r]{\makebox[\glstemplen][l]{%
10077     \textbf{\glstarget{##2}{##3}}}%
10078 \ifx##5\relax\relax
10079 \else
10080     (##5)\space
10081 \fi
10082 ##4\glspostdescription\space ##6\par
10083 \def\@gls@prevlevel{##1}%
10084 }%
10085 }%

```

Backward compatible alttreegroup style.

```

10086 \compatglossarystyle{alttreegroup}{%
10087 \csuse{@glscompstyle@alttree}%
10088 }%

```

Backward compatible alttreehypergroup style.

```

10089 \compatglossarystyle{alttreehypergroup}{%
10090 \csuse{@glscompstyle@alttree}%
10091 }%

```

Backward compatible mcolindex style.

```

10092 \compatglossarystyle{mcolindex}{%
10093 \csuse{@glscompstyle@index}%
10094 }%

```

Backward compatible mcolindexgroup style.

```

10095 \compatglossarystyle{mcolindexgroup}{%
10096 \csuse{@glscompstyle@index}%
10097 }%

```

Backward compatible mcolindexhypergroup style.

```

10098 \compatglossarystyle{mcolindexhypergroup}{%
10099 \csuse{@glscompstyle@index}%
10100 }%

```

Backward compatible mcoltree style.

```

10101 \compatglossarystyle{mcoltree}{%
10102 \csuse{@glscompstyle@tree}%
10103 }%

```

Backward compatible mcoltreegroup style.

```

10104 \compatglossarystyle{mcolindextreegroup}{%
10105 \csuse{@glscompstyle@tree}%
10106 }%

```

Backward compatible mcoltreehypergroup style.

```

10107 \compatglossarystyle{mcolindextreehypergroup}{%

```

```

10108 \csuse{@glscompstyle@tree}%
10109 }%

    Backward compatible mcoltreenoname style.
10110 \compatglossarystyle{mcoltreenoname}{%
10111 \csuse{@glscompstyle@tree}%
10112 }%

    Backward compatible mcoltreenonamegroup style.
10113 \compatglossarystyle{mcoltreenonamegroup}{%
10114 \csuse{@glscompstyle@tree}%
10115 }%

    Backward compatible mcoltreenonamehypergroup style.
10116 \compatglossarystyle{mcoltreenonamehypergroup}{%
10117 \csuse{@glscompstyle@tree}%
10118 }%

    Backward compatible mcolalmtree style.
10119 \compatglossarystyle{mcolalmtree}{%
10120 \csuse{@glscompstyle@almtree}%
10121 }%

    Backward compatible mcolalmtreegroup style.
10122 \compatglossarystyle{mcolalmtreegroup}{%
10123 \csuse{@glscompstyle@almtree}%
10124 }%

    Backward compatible mcolalmtreehypergroup style.
10125 \compatglossarystyle{mcolalmtreehypergroup}{%
10126 \csuse{@glscompstyle@almtree}%
10127 }%

    Backward compatible superragged style.
10128 \compatglossarystyle{superragged}{%
10129 \renewcommand*{\glossaryentryfield}[5]{%
10130 \glsentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
10131 \tabularnewline}%
10132 \renewcommand*{\glossarysubentryfield}[6]{%
10133 &
10134 \glssubentryitem{##2}%
10135 \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
10136 \tabularnewline}%
10137 }%

    Backward compatible superraggedborder style.
10138 \compatglossarystyle{superraggedborder}{%
10139 \csuse{@glscompstyle@superragged}%
10140 }%

    Backward compatible superraggedheader style.
10141 \compatglossarystyle{superraggedheader}{%
10142 \csuse{@glscompstyle@superragged}%
10143 }%

```

Backward compatible superraggedheaderborder style.

```
10144 \compatglossarystyle{superraggedheaderborder}{%
10145   \csuse{@glscompstyle@superragged}%
10146 }%
```

Backward compatible superragged3col style.

```
10147 \compatglossarystyle{superragged3col}{%
10148   \renewcommand*{\glossaryentryfield}[5]{%
10149     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%
10150   \renewcommand*{\glossarysubentryfield}[6]{%
10151     &
10152     \glssubentryitem{##2}%
10153     \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%
10154 }%
```

Backward compatible superragged3colborder style.

```
10155 \compatglossarystyle{superragged3colborder}{%
10156   \csuse{@glscompstyle@superragged3col}%
10157 }%
```

Backward compatible superragged3colheader style.

```
10158 \compatglossarystyle{superragged3colheader}{%
10159   \csuse{@glscompstyle@superragged3col}%
10160 }%
```

Backward compatible superragged3colheaderborder style.

```
10161 \compatglossarystyle{superragged3colheaderborder}{%
10162   \csuse{@glscompstyle@superragged3col}%
10163 }%
```

Backward compatible altsuperragged4col style.

```
10164 \compatglossarystyle{altsuperragged4col}{%
10165   \renewcommand*{\glossaryentryfield}[5]{%
10166     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%
10167   \renewcommand*{\glossarysubentryfield}[6]{%
10168     &
10169     \glssubentryitem{##2}%
10170     \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%
10171 }%
```

Backward compatible altsuperragged4colheader style.

```
10172 \compatglossarystyle{altsuperragged4colheader}{%
10173   \csuse{@glscompstyle@altsuperragged4col}%
10174 }%
```

Backward compatible altsuperragged4colborder style.

```
10175 \compatglossarystyle{altsuperragged4colborder}{%
10176   \csuse{@glscompstyle@altsuperragged4col}%
10177 }%
```

Backward compatible altsuperragged4colheaderborder style.

```
10178 \compatglossarystyle{altsuperragged4colheaderborder}{%
```

```

10179 \csuse{@glscompstyle@altsuperragged4col}%
10180 }%

```

Backward compatible super style.

```

10181 \compatglossarystyle{super}{%
10182   \renewcommand*{\glossaryentryfield}[5]{%
10183     \glentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
10184   \renewcommand*{\glossarysubentryfield}[6]{%
10185     &
10186     \glssubentryitem{##2}%
10187     \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
10188 }%

```

Backward compatible superborder style.

```

10189 \compatglossarystyle{superborder}{%
10190   \csuse{@glscompstyle@super}%
10191 }%

```

Backward compatible superheader style.

```

10192 \compatglossarystyle{superheader}{%
10193   \csuse{@glscompstyle@super}%
10194 }%

```

Backward compatible superheaderborder style.

```

10195 \compatglossarystyle{superheaderborder}{%
10196   \csuse{@glscompstyle@super}%
10197 }%

```

Backward compatible super3col style.

```

10198 \compatglossarystyle{super3col}{%
10199   \renewcommand*{\glossaryentryfield}[5]{%
10200     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\\}%
10201   \renewcommand*{\glossarysubentryfield}[6]{%
10202     &
10203     \glssubentryitem{##2}%
10204     \glstarget{##2}{\strut}##4 & ##6\\}%
10205 }%

```

Backward compatible super3colborder style.

```

10206 \compatglossarystyle{super3colborder}{%
10207   \csuse{@glscompstyle@super3col}%
10208 }%

```

Backward compatible super3colheader style.

```

10209 \compatglossarystyle{super3colheader}{%
10210   \csuse{@glscompstyle@super3col}%
10211 }%

```

Backward compatible super3colheaderborder style.

```

10212 \compatglossarystyle{super3colheaderborder}{%
10213   \csuse{@glscompstyle@super3col}%
10214 }%

```

Backward compatible super4col style.

```
10215 \compatglossarystyle{super4col}{%
10216   \renewcommand*{\glossaryentryfield}[5]{%
10217     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%
10218   \renewcommand*{\glossarysubentryfield}[6]{%
10219     &
10220     \glssubentryitem{##2}%
10221     \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%
10222 }%
```

Backward compatible super4colheader style.

```
10223 \compatglossarystyle{super4colheader}{%
10224   \csuse{@glscompstyle@super4col}%
10225 }%
```

Backward compatible super4colborder style.

```
10226 \compatglossarystyle{super4colborder}{%
10227   \csuse{@glscompstyle@super4col}%
10228 }%
```

Backward compatible super4colheaderborder style.

```
10229 \compatglossarystyle{super4colheaderborder}{%
10230   \csuse{@glscompstyle@super4col}%
10231 }%
```

Backward compatible altsuper4col style.

```
10232 \compatglossarystyle{altsuper4col}{%
10233   \csuse{@glscompstyle@super4col}%
10234 }%
```

Backward compatible altsuper4colheader style.

```
10235 \compatglossarystyle{altsuper4colheader}{%
10236   \csuse{@glscompstyle@super4col}%
10237 }%
```

Backward compatible altsuper4colborder style.

```
10238 \compatglossarystyle{altsuper4colborder}{%
10239   \csuse{@glscompstyle@super4col}%
10240 }%
```

Backward compatible altsuper4colheaderborder style.

```
10241 \compatglossarystyle{altsuper4colheaderborder}{%
10242   \csuse{@glscompstyle@super4col}%
10243 }%
```

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.

```
10244 \NeedsTeXFormat{LaTeX2e}
```

Package version number now in line with main glossaries package number.

```
10245 \ProvidesPackage{glossaries-accsupp}[2017/08/10 v4.31 (NLCT)]
```

```
10246 Experimental glossaries accessibility]
```

Pass all options to glossaries:

```
10247 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
10248 \ProcessOptions
```

This package should be loaded before glossaries-extra, so complain if that has already been loaded.

```
10249 \@ifpackageloaded{glossaries-extra}
```

```
10250 {%
```

If the accsupp option was used, \glsxtr@doaccsupp will have been set, otherwise it will be empty.

```
10251 \ifx\glsxtr@doaccsupp\empty
```

```
10252 \GlossariesWarning{The ‘glossaries-accsupp’
```

```
10253 package has been loaded\MessageBreak
```

```
10254 after the ‘glossaries-extra’ package. This\MessageBreak
```

```
10255 can cause a failure to integrate both packages. \MessageBreak
```

```
10256 Either use the ‘accsupp’ option when you load\MessageBreak
```

```
10257 ‘glossaries-extra’ or load ‘glossaries-accsupp’\MessageBreak
```

```
10258 before loading ‘glossaries-extra’}%
```

```
10259 \fi
```

```
10260 }
```

```
10261 {}
```

tibleglossentry Override style compatibility macros:

```
10262 \def\compatibleglossentry#1#2{%
```

```
10263 \toks@{#2}%
```

```
10264 \protected@edef\do@glossentry{%
```

```
10265 \noexpand\accsuppglossaryentryfield{#1}%
```

```
10266 {\noexpand\glsnamefont
```

```
10267 {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@name\endcsname}}%
```



```

10268 {\expandafter\expandonce\csname glo@glstetoklabel{#1}@desc\endcsname}%
10269 {\expandafter\expandonce\csname glo@glstetoklabel{#1}@symbol\endcsname}%
10270 {\the\toks@}%
10271 }%
10272 \@do@glossentry
10273 }

```

lesubglossentry

```

10274 \def\compatiblesubglossentry#1#2#3{%
10275   \toks@{#3}%
10276   \protected@edef\@do@subglossentry{%
10277     \noexpand\accsuppglossarysubentryfield{\number#1}%
10278     {#2}%
10279     {\noexpand\glsnamefont
10280      {\expandafter\expandonce\csname glo@glstetoklabel{#2}@name\endcsname}}}%
10281     {\expandafter\expandonce\csname glo@glstetoklabel{#2}@desc\endcsname}%
10282     {\expandafter\expandonce\csname glo@glstetoklabel{#2}@symbol\endcsname}%
10283     {\the\toks@}%
10284   }%
10285   \@do@subglossentry
10286 }

```

Required packages:

```

10287 \RequirePackage{glossaries}
10288 \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:

```

10289 \define@key{glossentry}{access}{%
10290   \def\@glo@access{#1}%
10291 }

```

textaccess The replacement text corresponding to the text key:

```

10292 \define@key{glossentry}{textaccess}{%
10293   \def\@glo@textaccess{#1}%
10294 }

```

firstaccess The replacement text corresponding to the first key:

```

10295 \define@key{glossentry}{firstaccess}{%
10296   \def\@glo@firstaccess{#1}%
10297 }

```

pluralaccess The replacement text corresponding to the plural key:

```

10298 \define@key{glossentry}{pluralaccess}{%
10299   \def\@glo@pluralaccess{#1}%
10300 }
```

firstpluralaccess The replacement text corresponding to the firstplural key:

```

10301 \define@key{glossentry}{firstpluralaccess}{%
10302   \def\@glo@firstpluralaccess{#1}%
10303 }
```

symbolaccess The replacement text corresponding to the symbol key:

```

10304 \define@key{glossentry}{symbolaccess}{%
10305   \def\@glo@symbolaccess{#1}%
10306 }
```

symbolpluralaccess The replacement text corresponding to the symbolplural key:

```

10307 \define@key{glossentry}{symbolpluralaccess}{%
10308   \def\@glo@symbolpluralaccess{#1}%
10309 }
```

descriptionaccess The replacement text corresponding to the description key:

```

10310 \define@key{glossentry}{descriptionaccess}{%
10311   \def\@glo@descaccess{#1}%
10312 }
```

descriptionpluralaccess The replacement text corresponding to the descriptionplural key:

```

10313 \define@key{glossentry}{descriptionpluralaccess}{%
10314   \def\@glo@descpluralaccess{#1}%
10315 }
```

shortaccess The replacement text corresponding to the short key:

```

10316 \define@key{glossentry}{shortaccess}{%
10317   \def\@glo@shortaccess{#1}%
10318 }
```

shortpluralaccess The replacement text corresponding to the shortplural key:

```

10319 \define@key{glossentry}{shortpluralaccess}{%
10320   \def\@glo@shortpluralaccess{#1}%
10321 }
```

longaccess The replacement text corresponding to the long key:

```

10322 \define@key{glossentry}{longaccess}{%
10323   \def\@glo@longaccess{#1}%
10324 }
```

longpluralaccess The replacement text corresponding to the longplural key:

```

10325 \define@key{glossentry}{longpluralaccess}{%
10326   \def\@glo@longpluralaccess{#1}%
10327 }
```

There are no equivalent keys for the user1...user6 keys. The replacement text would have to be explicitly put in the value, e.g., user1={\glsacccsupp{inches}{in}}.

Append these new keys to \@gls@keymap:

```
10328 \appto\@gls@keymap{,%
10329   {access}{access},%
10330   {textaccess}{textaccess},%
10331   {firstaccess}{firstaccess},%
10332   {pluralaccess}{pluralaccess},%
10333   {firstpluralaccess}{firstpluralaccess},%
10334   {symbolaccess}{symbolaccess},%
10335   {symbolpluralaccess}{symbolpluralaccess},%
10336   {descaccess}{descaccess},%
10337   {descpluralaccess}{descpluralaccess},%
10338   {shortaccess}{shortaccess},%
10339   {shortpluralaccess}{shortpluralaccess},%
10340   {longaccess}{longaccess},%
10341   {longpluralaccess}{longpluralaccess}}%
10342 }
```

\@gls@noaccess Indicates that no replacement text has been provided.

```
10343 \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):

```
10344 \let\@gls@oldnewglossaryentryprehook\@newglossaryentryprehook
10345 \renewcommand*{\@newglossaryentryprehook}{%
10346   \@gls@oldnewglossaryentryprehook
10347   \def\@glo@access{\@glo@symbol}}%
```

Initialise the other keys:

```
10348 \def\@glo@textaccess{\@glo@access}%
10349 \def\@glo@firstaccess{\@glo@access}%
10350 \def\@glo@pluralaccess{\@glo@textaccess}%
10351 \def\@glo@firstpluralaccess{\@glo@pluralaccess}%
10352 \def\@glo@symbolaccess{\relax}%
10353 \def\@glo@symbolpluralaccess{\@glo@symbolaccess}%
10354 \def\@glo@descaccess{\relax}%
10355 \def\@glo@descpluralaccess{\@glo@descaccess}%
10356 \def\@glo@shortaccess{\relax}%
10357 \def\@glo@shortpluralaccess{\@glo@shortaccess}%
10358 \def\@glo@longaccess{\relax}%
10359 \def\@glo@longpluralaccess{\@glo@longaccess}%
10360 }
```

Add to the end hook:

```
10361 \let\@gls@oldnewglossaryentryposthook\@newglossaryentryposthook
10362 \renewcommand*{\@newglossaryentryposthook}{%
10363   \@gls@oldnewglossaryentryposthook}
```

Store the access information:

```
10364 \expandafter
10365 \protected@xdef\csname glo@\@glo@label @access\endcsname{%
10366 \@glo@access}%
10367 \expandafter
10368 \protected@xdef\csname glo@\@glo@label @textaccess\endcsname{%
10369 \@glo@textaccess}%
10370 \expandafter
10371 \protected@xdef\csname glo@\@glo@label @firstaccess\endcsname{%
10372 \@glo@firstaccess}%
10373 \expandafter
10374 \protected@xdef\csname glo@\@glo@label @pluralaccess\endcsname{%
10375 \@glo@pluralaccess}%
10376 \expandafter
10377 \protected@xdef\csname glo@\@glo@label @firstpluralaccess\endcsname{%
10378 \@glo@firstpluralaccess}%
10379 \expandafter
10380 \protected@xdef\csname glo@\@glo@label @symbolaccess\endcsname{%
10381 \@glo@symbolaccess}%
10382 \expandafter
10383 \protected@xdef\csname glo@\@glo@label @symbolpluralaccess\endcsname{%
10384 \@glo@symbolpluralaccess}%
10385 \expandafter
10386 \protected@xdef\csname glo@\@glo@label @descaccess\endcsname{%
10387 \@glo@descaccess}%
10388 \expandafter
10389 \protected@xdef\csname glo@\@glo@label @descpluralaccess\endcsname{%
10390 \@glo@descpluralaccess}%
10391 \expandafter
10392 \protected@xdef\csname glo@\@glo@label @shortaccess\endcsname{%
10393 \@glo@shortaccess}%
10394 \expandafter
10395 \protected@xdef\csname glo@\@glo@label @shortpluralaccess\endcsname{%
10396 \@glo@shortpluralaccess}%
10397 \expandafter
10398 \protected@xdef\csname glo@\@glo@label @longaccess\endcsname{%
10399 \@glo@longaccess}%
10400 \expandafter
10401 \protected@xdef\csname glo@\@glo@label @longpluralaccess\endcsname{%
10402 \@glo@longpluralaccess}%
10403 }
```

5.2 Accessing Replacement Text

`\glentryaccess` Get the value of the access key for the entry with the given label:

```
10404 \newcommand*{\glentryaccess}[1]{%
10405 \@gls@entry@field{#1}{access}%
10406 }
```

entrytextaccess Get the value of the textaccess key for the entry with the given label:

```

10407 \newcommand*{\glsentrytextaccess}[1]{%
10408   \@gls@entry@field{#1}{textaccess}%
10409 }

```

entryfirstaccess Get the value of the firstaccess key for the entry with the given label:

```

10410 \newcommand*{\glsentryfirstaccess}[1]{%
10411   \@gls@entry@field{#1}{firstaccess}%
10412 }

```

entrypluralaccess Get the value of the pluralaccess key for the entry with the given label:

```

10413 \newcommand*{\glsentrypluralaccess}[1]{%
10414   \@gls@entry@field{#1}{pluralaccess}%
10415 }

```

entryfirstpluralaccess Get the value of the firstpluralaccess key for the entry with the given label:

```

10416 \newcommand*{\glsentryfirstpluralaccess}[1]{%
10417   \csname glo@#1@firstpluralaccess\endcsname
10418 }

```

entrysymbolaccess Get the value of the symbolaccess key for the entry with the given label:

```

10419 \newcommand*{\glsentrysymbolaccess}[1]{%
10420   \@gls@entry@field{#1}{symbolaccess}%
10421 }

```

entrysymbolpluralaccess Get the value of the symbolpluralaccess key for the entry with the given label:

```

10422 \newcommand*{\glsentrysymbolpluralaccess}[1]{%
10423   \@gls@entry@field{#1}{symbolpluralaccess}%
10424 }

```

entrydescaccess Get the value of the descriptionaccess key for the entry with the given label:

```

10425 \newcommand*{\glsentrydescaccess}[1]{%
10426   \@gls@entry@field{#1}{descaccess}%
10427 }

```

entrydescpluralaccess Get the value of the descriptionpluralaccess key for the entry with the given label:

```

10428 \newcommand*{\glsentrydescpluralaccess}[1]{%
10429   \@gls@entry@field{#1}{descaccess}%
10430 }

```

entryshortaccess Get the value of the shortaccess key for the entry with the given label:

```

10431 \newcommand*{\glsentryshortaccess}[1]{%
10432   \@gls@entry@field{#1}{shortaccess}%
10433 }

```

entryshortpluralaccess Get the value of the shortpluralaccess key for the entry with the given label:

```

10434 \newcommand*{\glsentryshortpluralaccess}[1]{%
10435   \@gls@entry@field{#1}{shortpluralaccess}%
10436 }

```

entrylongaccess Get the value of the longaccess key for the entry with the given label:

```
10437 \newcommand*{\glentrylongaccess}[1]{%
10438   \@gls@entry@field{#1}{longaccess}%
10439 }
```

ongpluralaccess Get the value of the longpluralaccess key for the entry with the given label:

```
10440 \newcommand*{\glentrylongpluralaccess}[1]{%
10441   \@gls@entry@field{#1}{longpluralaccess}%
10442 }
```

\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.)

```
10443 \newcommand*{\glsaccsupp}[2]{%
10444   \BeginAccSupp{ActualText=#1}#2\EndAccSupp{}%
10445 }
```

\xglsaccsupp Fully expands replacement text before calling \glsaccsupp

```
10446 \newcommand*{\xglsaccsupp}[2]{%
10447   \protected@edef\@gls@replacementtext{#1}%
10448   \expandafter\glsaccsupp\expandafter{\@gls@replacementtext}{#2}%
10449 }
```

@access@display

```
10450 \newcommand*{\@gls@access@display}[2]{%
10451   \protected@edef\@glo@access{#2}%
10452   \ifx\@glo@access\@gls@noaccess
10453     #1%
10454   \else
10455     \xglsaccsupp{\@glo@access}{#1}%
10456   \fi
10457 }
```

meaccessdisplay Displays the first argument with the accessibility text for the entry with the label given by the second argument (if set).

```
10458 \DeclareRobustCommand*{\glsnameaccessdisplay}[2]{%
10459   \@gls@access@display{#1}{\glentryaccess{#2}}%
10460 }
```

xtaccessdisplay As above but for the textaccess replacement text.

```
10461 \DeclareRobustCommand*{\glstextaccessdisplay}[2]{%
10462   \@gls@access@display{#1}{\glentrytextaccess{#2}}%
10463 }
```

alaccessdisplay As above but for the pluralaccess replacement text.

```
10464 \DeclareRobustCommand*{\glspluralaccessdisplay}[2]{%
10465   \@gls@access@display{#1}{\glentrypluralaccess{#2}}%
10466 }
```

staccessdisplay As above but for the firstaccess replacement text.

```

10467 \DeclareRobustCommand*\glfirstaccessdisplay}[2]{%
10468   \@gls@access@display{#1}{\glentryfirstaccess{#2}}%
10469 }

```

alaccessdisplay As above but for the firstpluralaccess replacement text.

```

10470 \DeclareRobustCommand*\glfirstpluralaccessdisplay}[2]{%
10471   \@gls@access@display{#1}{\glentryfirstpluralaccess{#2}}%
10472 }

```

olaccessdisplay As above but for the symbolaccess replacement text.

```

10473 \DeclareRobustCommand*\glssymbolaccessdisplay}[2]{%
10474   \@gls@access@display{#1}{\glentrysymbolaccess{#2}}%
10475 }

```

alaccessdisplay As above but for the symbolpluralaccess replacement text.

```

10476 \DeclareRobustCommand*\glssymbolpluralaccessdisplay}[2]{%
10477   \@gls@access@display{#1}{\glentrysymbolpluralaccess{#2}}%
10478 }

```

onaccessdisplay As above but for the descriptionaccess replacement text.

```

10479 \DeclareRobustCommand*\glsdescriptionaccessdisplay}[2]{%
10480   \@gls@access@display{#1}{\glentrydescaccess{#2}}%
10481 }

```

alaccessdisplay As above but for the descriptionpluralaccess replacement text.

```

10482 \DeclareRobustCommand*\glsdescriptionpluralaccessdisplay}[2]{%
10483   \@gls@access@display{#1}{\glentrydescpluralaccess{#2}}%
10484 }

```

rtaccessdisplay As above but for the shortaccess replacement text.

```

10485 \DeclareRobustCommand*\glsshortaccessdisplay}[2]{%
10486   \@gls@access@display{#1}{\glentryshortaccess{#2}}%
10487 }

```

alaccessdisplay As above but for the shortpluralaccess replacement text.

```

10488 \DeclareRobustCommand*\glsshortpluralaccessdisplay}[2]{%
10489   \@gls@access@display{#1}{\glentryshortpluralaccess{#2}}%
10490 }

```

ngaccessdisplay As above but for the longaccess replacement text.

```

10491 \DeclareRobustCommand*\glslongaccessdisplay}[2]{%
10492   \@gls@access@display{#1}{\glentrylongaccess{#2}}%
10493 }

```

alaccessdisplay As above but for the longpluralaccess replacement text.

```

10494 \DeclareRobustCommand*\glslongpluralaccessdisplay}[2]{%
10495   \@gls@access@display{#1}{\glentrylongpluralaccess{#2}}%
10496 }

```

`\glsaccessdisplay` Gets the replacement text corresponding to the named key given by the first argument and calls the appropriate command defined above.

```

10497 \DeclareRobustCommand*\glsaccessdisplay}[3]{%
10498   \@ifundefined{gls#1accessdisplay}%
10499   {%
10500     \PackageError{glossaries-accsupp}{No accessibility support
10501       for key ‘#1’}{}%
10502   }%
10503   {%
10504     \csname gls#1accessdisplay\endcsname{#2}{#3}%
10505   }%
10506 }

```

`\default@entryfmt` Redefine the default entry format to use accessibility information

```

10507 \renewcommand*\@@gls@default@entryfmt}[2]{%
10508   \ifdefempty\glscustomtext
10509   {%
10510     \glsifplural
10511     {%

```

Plural form

```

10512       \glscapscase
10513       {%

```

Don't adjust case

```

10514       \ifglsused\glslabel
10515       {%

```

Subsequent use

```

10516         #2{\glspluralaccessdisplay
10517           {\glsentryplural{\glslabel}}{\glslabel}}%
10518         {\glsdescriptionpluralaccessdisplay
10519           {\glsentrydescplural{\glslabel}}{\glslabel}}%
10520         {\glsymbolpluralaccessdisplay
10521           {\glsentrysymbolplural{\glslabel}}{\glslabel}}
10522         {\glsinsert}%
10523       }%
10524     {%

```

First use

```

10525         #1{\glsfirstpluralaccessdisplay
10526           {\glsentryfirstplural{\glslabel}}{\glslabel}}%
10527         {\glsdescriptionpluralaccessdisplay
10528           {\glsentrydescplural{\glslabel}}{\glslabel}}%
10529         {\glsymbolpluralaccessdisplay
10530           {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10531         {\glsinsert}%
10532       }%
10533     }%
10534   }%

```


Make first letter upper case

```
10535      \ifglused\glslabel
10536      {%
```

Subsequent use.

```
10537      #2{\glsppluralaccessdisplay
10538          {\Glentryplural{\glslabel}}{\glslabel}}%
10539          {\glsdescriptionpluralaccessdisplay
10540              {\glentrydescplural{\glslabel}}{\glslabel}}%
10541              {\glssymbolpluralaccessdisplay
10542                  {\glentrysymbolplural{\glslabel}}{\glslabel}}%
10543              {\glinsert}}%
10544      }%
10545      {%
```

First use

```
10546      #1{\glfirstpluralaccessdisplay
10547          {\Glentryfirstplural{\glslabel}}{\glslabel}}%
10548          {\glsdescriptionpluralaccessdisplay
10549              {\glentrydescplural{\glslabel}}{\glslabel}}%
10550              {\glssymbolpluralaccessdisplay
10551                  {\glentrysymbolplural{\glslabel}}{\glslabel}}%
10552              {\glinsert}}%
10553      }%
10554      }%
10555      {%
```

Make all upper case

```
10556      \ifglused\glslabel
10557      {%
```

Subsequent use

```
10558      \MakeUppercase{%
10559      #2{\glsppluralaccessdisplay
10560          {\glentryplural{\glslabel}}{\glslabel}}%
10561          {\glsdescriptionpluralaccessdisplay
10562              {\glentrydescplural{\glslabel}}{\glslabel}}%
10563              {\glssymbolpluralaccessdisplay
10564                  {\glentrysymbolplural{\glslabel}}{\glslabel}}%
10565              {\glinsert}}}%
10566      }%
10567      {%
```

First use

```
10568      \MakeUppercase{%
10569      #1{\glfirstpluralaccessdisplay
10570          {\glentryfirstplural{\glslabel}}{\glslabel}}%
10571          {\glsdescriptionpluralaccessdisplay
10572              {\glentrydescplural{\glslabel}}{\glslabel}}%
10573              {\glssymbolpluralaccessdisplay
10574                  {\glentrysymbolplural{\glslabel}}{\glslabel}}%
```

```

10575         {\glsinsert}}}%
10576     }%
10577 }%
10578 }%
10579 {%

```

Singular form

```

10580     \glscapscase
10581     {%

```

Don't adjust case

```

10582     \ifglsused\glslabel
10583     {%

```

Subsequent use

```

10584     #2{\glstextaccessdisplay
10585         {\glsentrytext{\glslabel}}{\glslabel}}%
10586     {\glsdescriptionaccessdisplay
10587         {\glsentrydesc{\glslabel}}{\glslabel}}%
10588     {\glssymbolaccessdisplay
10589         {\glsentrysymbol{\glslabel}}{\glslabel}}%
10590     {\glsinsert}}%
10591 }%
10592 {%

```

First use

```

10593     #1{\glsfirstaccessdisplay
10594         {\glsentryfirst{\glslabel}}{\glslabel}}%
10595     {\glsdescriptionaccessdisplay
10596         {\glsentrydesc{\glslabel}}{\glslabel}}%
10597     {\glssymbolaccessdisplay
10598         {\glsentrysymbol{\glslabel}}{\glslabel}}%
10599     {\glsinsert}}%
10600 }%
10601 }%
10602 {%

```

Make first letter upper case

```

10603     \ifglsused\glslabel
10604     {%

```

Subsequent use

```

10605     #2{\glstextaccessdisplay
10606         {\Glsentrytext{\glslabel}}{\glslabel}}%
10607     {\glsdescriptionaccessdisplay
10608         {\glsentrydesc{\glslabel}}{\glslabel}}%
10609     {\glssymbolaccessdisplay
10610         {\glsentrysymbol{\glslabel}}{\glslabel}}%
10611     {\glsinsert}}%
10612 }%
10613 {%

```

First use

```

10614      #1{\glsfirstaccessdisplay
10615          {\Glsentryfirst{\glslabel}}{\glslabel}}%
10616          {\glsdescriptionaccessdisplay
10617              {\glsentrydesc{\glslabel}}{\glslabel}}%
10618          {\glssymbolaccessdisplay
10619              {\glsentrysymbol{\glslabel}}{\glslabel}}%
10620          {\glsinsert}}%
10621      }%
10622  }%
10623  {%

```

Make all upper case

```

10624      \ifglsused\glslabel
10625      {%

```

Subsequent use

```

10626      \MakeUppercase{%
10627          #2{\glstextaccessdisplay
10628              {\glsentrytext{\glslabel}}{\glslabel}}%
10629              {\glsdescriptionaccessdisplay
10630                  {\glsentrydesc{\glslabel}}{\glslabel}}%
10631                  {\glssymbolaccessdisplay
10632                      {\glsentrysymbol{\glslabel}}{\glslabel}}%
10633                      {\glsinsert}}}%
10634      }%
10635      {%

```

First use

```

10636      \MakeUppercase{%
10637          #1{\glsfirstaccessdisplay
10638              {\glsentryfirst{\glslabel}}{\glslabel}}%
10639              {\glsdescriptionaccessdisplay
10640                  {\glsentrydesc{\glslabel}}{\glslabel}}%
10641                  {\glssymbolaccessdisplay
10642                      {\glsentrysymbol{\glslabel}}{\glslabel}}%
10643                      {\glsinsert}}}%
10644      }%
10645  }%
10646  }%
10647  }%
10648  {%

```

Custom text provided in \glsdisp

```

10649      \ifglsused{\glslabel}%
10650      {%

```

Subsequent use

```

10651      #2{\glscustomtext}%
10652      {\glsdescriptionaccessdisplay
10653          {\glsentrydesc{\glslabel}}{\glslabel}}%

```

```

10654      {\glssymbolaccessdisplay
10655        {\glentrysymbol{\glslabel}}{\glslabel}}%
10656      {\glsinsert}%
10657    }%
10658    {%

```

First use

```

10659      #1{\glscustomtext}%
10660      {\glsdescriptionaccessdisplay
10661        {\glentrydesc{\glslabel}}{\glslabel}}%
10662      {\glssymbolaccessdisplay
10663        {\glentrysymbol{\glslabel}}{\glslabel}}%
10664      {\glsinsert}%
10665    }%
10666    }%
10667 }

```

`\glsgenentryfmt` Redefine to use accessibility information.

```

10668 \renewcommand*{\glsgenentryfmt}{%
10669   \ifdefempty\glscustomtext
10670     {%
10671       \glsifplural
10672       {%

```

Plural form

```

10673       \glscapscase
10674       {%

```

Don't adjust case

```

10675       \ifglused\glslabel
10676       {%

```

Subsequent use

```

10677       \glspluralaccessdisplay
10678       {\glentryplural{\glslabel}}{\glslabel}%
10679       \glsinsert
10680     }%
10681     {%

```

First use

```

10682       \glsfirstpluralaccessdisplay
10683       {\glentryfirstplural{\glslabel}}{\glslabel}%
10684       \glsinsert
10685     }%
10686     }%
10687     {%

```

Make first letter upper case

```

10688       \ifglused\glslabel
10689       {%

```

Subsequent use.

```
10690      \glspluralaccessdisplay
10691      {\Glsentryplural{\glslabel}}{\glslabel}%
10692      \glsinsert
10693      }%
10694      {%
```

First use

```
10695      \glsfirstpluralaccessdisplay
10696      {\Glsentryfirstplural{\glslabel}}{\glslabel}%
10697      \glsinsert
10698      }%
10699      }%
10700      {%
```

Make all upper case

```
10701      \ifglused\glslabel
10702      {%
```

Subsequent use

```
10703      \glspluralaccessdisplay
10704      {\mfirstucMakeUppercase{\Glsentryplural{\glslabel}}}%
10705      {\glslabel}%
10706      \mfirstucMakeUppercase{\glsinsert}%
10707      }%
10708      {%
```

First use

```
10709      \glsfirstpluralaccessdisplay
10710      {\mfirstucMakeUppercase{\Glsentryfirstplural{\glslabel}}}%
10711      {\glslabel}%
10712      \mfirstucMakeUppercase{\glsinsert}%
10713      }%
10714      }%
10715      }%
10716      {%
```

Singular form

```
10717      \glscapscase
10718      {%
```

Don't adjust case

```
10719      \ifglused\glslabel
10720      {%
```

Subsequent use

```
10721      \glstextaccessdisplay{\Glsentrytext{\glslabel}}{\glslabel}%
10722      \glsinsert
10723      }%
10724      {%
```

First use

```
10725      \glsfirstaccessdisplay{\glsentryfirst{\glslabel}}{\glslabel}%
10726      \glsinsert
10727      }%
10728      }%
10729      {%
```

Make first letter upper case

```
10730      \ifglsused\glslabel
10731      {%
```

Subsequent use

```
10732      \glstextaccessdisplay{\Glsentrytext{\glslabel}}{\glslabel}%
10733      \glsinsert
10734      }%
10735      {%
```

First use

```
10736      \glsfirstaccessdisplay{\Glsentryfirst{\glslabel}}{\glslabel}%
10737      \glsinsert
10738      }%
10739      }%
10740      {%
```

Make all upper case

```
10741      \ifglsused\glslabel
10742      {%
```

Subsequent use

```
10743      \glstextaccessdisplay
10744      {\mfirstucMakeUppercase{\glsentrytext{\glslabel}}}{\glslabel}%
10745      \mfirstucMakeUppercase{\glsinsert}%
10746      }%
10747      {%
```

First use

```
10748      \glsfirstaccessdisplay
10749      {\mfirstucMakeUppercase{\glsentryfirst{\glslabel}}}{\glslabel}%
10750      \mfirstucMakeUppercase{\glsinsert}%
10751      }%
10752      }%
10753      }%
10754      }%
10755      {%
```

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.

```
10756      \glscustomtext\glsinsert
10757      }%
10758 }
```

`\glsgenacfmt` Redefine to include accessibility information.

```
10759 \renewcommand*{\glsgenacfmt}{%
10760   \ifdefempty\glscustomtext
10761   {%
10762     \ifglused\glslabel
10763     {%
```

Subsequent use:

```
10764     \glsifplural
10765     {%
```

Subsequent plural form:

```
10766     \glscapscase
10767     {%
```

Subsequent plural form, don't adjust case:

```
10768     \acronymfont
10769     {\glsshortpluralaccessdisplay
10770      {\glentryshortpl{\glslabel}}{\glslabel}}%
10771     \glsinsert
10772   }%
10773   {%
```

Subsequent plural form, make first letter upper case:

```
10774     \acronymfont
10775     {\glsshortpluralaccessdisplay
10776      {\Glsentryshortpl{\glslabel}}{\glslabel}}%
10777     \glsinsert
10778   }%
10779   {%
```

Subsequent plural form, all caps:

```
10780     \mfirstucMakeUppercase
10781     {\acronymfont
10782      {\glsshortpluralaccessdisplay
10783       {\glentryshortpl{\glslabel}}{\glslabel}}%
10784      \glsinsert}%
10785   }%
10786   }%
10787   {%
```

Subsequent singular form

```
10788     \glscapscase
10789     {%
```

Subsequent singular form, don't adjust case:

```
10790     \acronymfont
10791     {\glsshortaccessdisplay{\glentryshort{\glslabel}}{\glslabel}}%
10792     \glsinsert
10793   }%
10794   {%
```

Subsequent singular form, make first letter upper case:

```
10795      \acronymfont
10796      {\glsshortaccessdisplay{\Glsentryshort{\glslabel}}{\glslabel}}%
10797      \glsinsert
10798      }%
10799      {%
```

Subsequent singular form, all caps:

```
10800      \mfirstucMakeUppercase
10801      {\acronymfont{%
10802      \glsshortaccessdisplay{\Glsentryshort{\glslabel}}{\glslabel}}%
10803      \glsinsert}%
10804      }%
10805      }%
10806      }%
10807      {%
```

First use:

```
10808      \glsifplural
10809      {%
```

First use plural form:

```
10810      \glscapscase
10811      {%
```

First use plural form, don't adjust case:

```
10812      \genplacrfullformat{\glslabel}{\glsinsert}%
10813      }%
10814      {%
```

First use plural form, make first letter upper case:

```
10815      \Genplacrfullformat{\glslabel}{\glsinsert}%
10816      }%
10817      {%
```

First use plural form, all caps:

```
10818      \mfirstucMakeUppercase
10819      {\genplacrfullformat{\glslabel}{\glsinsert}}%
10820      }%
10821      }%
10822      {%
```

First use singular form

```
10823      \glscapscase
10824      {%
```

First use singular form, don't adjust case:

```
10825      \genacrfullformat{\glslabel}{\glsinsert}%
10826      }%
10827      {%
```


First use singular form, make first letter upper case:

```
10828      \Genacrfullformat{\glslabel}{\glsinsert}%
10829      }%
10830      {%
```

First use singular form, all caps:

```
10831      \mfirstucMakeUppercase
10832      {\genacrfullformat{\glslabel}{\glsinsert}}%
10833      }%
10834      }%
10835      }%
10836      }%
10837      {%
```

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.

```
10838      \glscustomtext
10839      }%
10840 }
```

enacrfullformat Redefine to include accessibility information.

```
10841 \renewcommand*{\genacrfullformat}[2]{%
10842   \glslongaccessdisplay{\glsentrylong{#1}}{#1}#2\space
10843   (\glsshortaccessdisplay{\protect\firstacronymfont{\glsentryshort{#1}}}{#1}}%
10844 }
```

enacrfullformat Redefine to include accessibility information.

```
10845 \renewcommand*{\Genacrfullformat}[2]{%
10846   \glslongaccessdisplay{\Glsentrylong{#1}}{#1}#2\space
10847   (\glsshortaccessdisplay{\protect\firstacronymfont{\Glsentryshort{#1}}}{#1}}%
10848 }
```

placrfullformat Redefine to include accessibility information.

```
10849 \renewcommand*{\genplacrfullformat}[2]{%
10850   \glslongpluralaccessdisplay{\glsentrylongpl{#1}}{#1}#2\space
10851   (\glsshortpluralaccessdisplay
10852     {\protect\firstacronymfont{\glsentryshortpl{#1}}}{#1}}%
10853 }
```

placrfullformat Redefine to include accessibility information.

```
10854 \renewcommand*{\Genplacrfullformat}[2]{%
10855   \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}#2\space
10856   (\glsshortpluralaccessdisplay
10857     {\protect\firstacronymfont{\glsentryshortpl{#1}}}{#1}}%
10858 }
```

\@acrshort

```
10859 \def\@acrshort#1#2[#3]{%
10860   \glsdoifexists{#2}%
```

```

10861 {%
10862   \let\do@gl@s@link@checkfirsthyper\relax

10863   \let\gl@sifplural\@secondoftwo
10864   \let\gl@scapscase\@firstofthree
10865   \let\gl@sinsert\@empty
10866   \def\glscustomtext{%
10867     \acronymfont{\glsshortaccessdisplay{\gl@sentryshort{#2}}{#2}}#3%
10868   }%

   Call \@gl@s@link
10869   \@gl@s@link[#1]{#2}{\csname gl@s\glstype @entryfmt\endcsname}%
10870   }%

10871   \glspostlinkhook
10872 }

```

\@Acrshort

```

10873 \def\@Acrshort#1#2[#3]{%
10874   \gl@sdoifexists{#2}%
10875   {%
10876     \let\do@gl@s@link@checkfirsthyper\relax

10877     \let\gl@sifplural\@secondoftwo
10878     \let\gl@scapscase\@secondofthree
10879     \let\gl@sinsert\@empty
10880     \def\glscustomtext{%
10881       \acronymfont{\glsshortaccessdisplay{\gl@sentryshort{#2}}{#2}}#3%
10882     }%

     Call \@gl@s@link
10883     \@gl@s@link[#1]{#2}{\csname gl@s\glstype @entryfmt\endcsname}%
10884     }%

10885     \glspostlinkhook
10886 }

```

\@ACRshort

```

10887 \def\@ACRshort#1#2[#3]{%
10888   \gl@sdoifexists{#2}%
10889   {%
10890     \let\do@gl@s@link@checkfirsthyper\relax

10891     \let\gl@sifplural\@secondoftwo
10892     \let\gl@scapscase\@thirdofthree
10893     \let\gl@sinsert\@empty
10894     \def\glscustomtext{%
10895       \acronymfont{\glsshortaccessdisplay
10896         {\MakeUppercase{\gl@sentryshort{#2}}}{#2}}#3%
10897     }%

```

```

    Call \@gls@link
10898   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10899   }%

10900   \glspostlinkhook
10901 }

```

\@acrlong

```

10902 \def\@acrlong#1#2[#3]{%
10903   \glsdoifexists{#2}%
10904   {%
10905     \let\do@gls@link@checkfirsthyper\relax

10906     \let\glsifplural\@secondoftwo
10907     \let\glscapscase\@firstofthree
10908     \let\glsinsert\@empty
10909     \def\glscustomtext{%
10910       \acronymfont{\glslongaccessdisplay{\glsentrylong{#2}}{#2}}#3%
10911     }%

```

```

    Call \@gls@link
10912   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10913   }%

10914   \glspostlinkhook
10915 }

```

\@Acrlong

```

10916 \def\@Acrlong#1#2[#3]{%
10917   \glsdoifexists{#2}%
10918   {%
10919     \let\do@gls@link@checkfirsthyper\relax

10920     \let\glsifplural\@secondoftwo
10921     \let\glscapscase\@firstofthree
10922     \let\glsinsert\@empty
10923     \def\glscustomtext{%
10924       \acronymfont{\glslongaccessdisplay{\Glsentrylong{#2}}{#2}}#3%
10925     }%

```

```

    Call \@gls@link
10926   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10927   }%

10928   \glspostlinkhook
10929 }

```

\@ACRlong

```

10930 \def\@ACRlong#1#2[#3]{%
10931   \glsdoifexists{#2}%
10932   {%
10933     \let\do@gls@link@checkfirsthyper\relax

```

```

10934 \let\glsifplural\@secondoftwo
10935 \let\glsifscapscase\@firstofthree
10936 \let\glsinsert\@empty
10937 \def\glscustomtext{%
10938 \acronymfont{\glslongaccessdisplay{%
10939 \MakeUppercase{\glsentrylong{#2}}}{#2}#3}%
10940 }%

Call \@gls@link
10941 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10942 }%

10943 \glspostlinkhook
10944 }

```

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.

```

10945 \renewcommand*{\glossentryname}[1]{%
10946 \glsdoifexists{#1}%
10947 {%
10948 \glsnamefont{\glsnameaccessdisplay{\glsentryname{#1}}{#1}}%
10949 }%
10950 }

10951 \renewcommand*{\glossentryname}[1]{%
10952 \glsdoifexists{#1}%
10953 {%
10954 \glsnamefont{\glsnameaccessdisplay{\Glsentryname{#1}}{#1}}%
10955 }%
10956 }

10957 \renewcommand*{\glossentrydesc}[1]{%
10958 \glsdoifexists{#1}%
10959 {%
10960 \glsdescriptionaccessdisplay{\glsentrydesc{#1}}{#1}%
10961 }%
10962 }

10963 \renewcommand*{\Glossentrydesc}[1]{%
10964 \glsdoifexists{#1}%
10965 {%
10966 \glsdescriptionaccessdisplay{\Glsentrydesc{#1}}{#1}%
10967 }%
10968 }

```

```

10969 \renewcommand*{\glossentrysymbol}[1]{%
10970   \glsdoifexists{#1}%
10971   {%
10972     \glssymbolaccessdisplay{\glsentrysymbol{#1}}{#1}%
10973   }%
10974 }

10975 \renewcommand*{\Glossentrysymbol}[1]{%
10976   \glsdoifexists{#1}%
10977   {%
10978     \glssymbolaccessdisplay{\Glsentrysymbol{#1}}{#1}%
10979   }%
10980 }

```

ssaryentryfield

```

10981 \newcommand*{\accsuppglossaryentryfield}[5]{%
10982   \glossaryentryfield{#1}%
10983   {\glsnameaccessdisplay{#2}{#1}}%
10984   {\glsdescriptionaccessdisplay{#3}{#1}}%
10985   {\glssymbolaccessdisplay{#4}{#1}}{#5}%
10986 }

```

rysubentryfield

```

10987 \newcommand*{\accsuppglossarysubentryfield}[6]{%
10988   \glossarysubentryfield{#1}{#2}%
10989   {\glsnameaccessdisplay{#3}{#2}}%
10990   {\glsdescriptionaccessdisplay{#4}{#2}}%
10991   {\glssymbolaccessdisplay{#5}{#2}}{#6}%
10992 }

```

5.4 Acronyms

Redefine acronym styles provided by glossaries:

long-short *<long>* (*<short>*) acronym style.

```

10993 \renewacronymstyle{long-short}%
10994 {%

```

Check for long form in case this is a mixed glossary.

```

10995   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
10996 }%
10997 {%
10998   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
10999   \renewcommand*{\genacrfullformat}[2]{%
11000     \glslongaccessdisplay{\glsentrylong{##1}}{##1}##2\space
11001     (\glsshortaccessdisplay
11002       {\protect\firstacronymfont{\glsentryshort{##1}}}{##1})%
11003   }%
11004   \renewcommand*{\Genacrfullformat}[2]{%

```

```

11005 \glslongaccessdisplay{\Glsentrylong{##1}}{##1}##2\space
11006 (\glsshortaccessdisplay
11007 {\protect\firstacronymfont{\glsentryshort{##1}}{##1}})%
11008 }%
11009 \renewcommand*{\genplacrfullformat}[2]{%
11010 \glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}##2\space
11011 (\glsshortpluralaccessdisplay
11012 {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}})%
11013 }%
11014 \renewcommand*{\Genplacrfullformat}[2]{%
11015 \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}##2\space
11016 (\glsshortpluralaccessdisplay
11017 {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}})%
11018 }%
11019 \renewcommand*{\acronymentry}[1]{%
11020 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}{##1}}
11021 \renewcommand*{\acronymsort}[2]{##1}%
11022 \renewcommand*{\acronymfont}[1]{##1}%
11023 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
11024 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11025 }

```

short-long (*short*) (*long*) acronym style.

```

11026 \renewacronymstyle{short-long}%
11027 {%

```

Check for long form in case this is a mixed glossary.

```

11028 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
11029 }%
11030 {%
11031 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
11032 \renewcommand*{\genacrfullformat}[2]{%
11033 \glsshortaccessdisplay
11034 {\protect\firstacronymfont{\glsentryshort{##1}}{##1}##2\space
11035 (\glslongaccessdisplay{\glsentrylong{##1}}{##1}})%
11036 }%
11037 \renewcommand*{\Genacrfullformat}[2]{%
11038 \glsshortaccessdisplay
11039 {\protect\firstacronymfont{\Glsentryshort{##1}}{##1}##2\space
11040 (\glslongaccessdisplay{\glsentrylong{##1}}{##1}})%
11041 }%
11042 \renewcommand*{\genplacrfullformat}[2]{%
11043 \glsshortpluralaccessdisplay
11044 {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}##2\space
11045 (\glslongpluralaccessdisplay
11046 {\glsentrylongpl{##1}}{##1}})%
11047 }%
11048 \renewcommand*{\Genplacrfullformat}[2]{%
11049 \glsshortpluralaccessdisplay
11050 {\protect\firstacronymfont{\Glsentryshortpl{##1}}{##1}##2\space

```

```

11051 (\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1})%
11052 }%
11053 \renewcommand*{\acronymentry}[1]{%
11054   \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}%
11055 \renewcommand*{\acronymsort}[2]{##1}%
11056 \renewcommand*{\acronymfont}[1]{##1}%
11057 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
11058 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11059 }

```

long-short-desc *long* (*short*) acronym style that has an accompanying description (which the user needs to supply).

```

11060 \renewacronymstyle{long-short-desc}%
11061 {%
11062   \GlsUseAcrEntryDispStyle{long-short}%
11063 }%
11064 {%
11065   \GlsUseAcrStyleDefs{long-short}%
11066   \renewcommand*{\GenericAcronymFields}{}%
11067   \renewcommand*{\acronymsort}[2]{##2}%
11068   \renewcommand*{\acronymentry}[1]{%
11069     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11070     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11071 }

```

g-sc-short-desc *long* (\textsc{short}) acronym style that has an accompanying description (which the user needs to supply).

```

11072 \renewacronymstyle{long-sc-short-desc}%
11073 {%
11074   \GlsUseAcrEntryDispStyle{long-sc-short}%
11075 }%
11076 {%
11077   \GlsUseAcrStyleDefs{long-sc-short}%
11078   \renewcommand*{\GenericAcronymFields}{}%
11079   \renewcommand*{\acronymsort}[2]{##2}%
11080   \renewcommand*{\acronymentry}[1]{%
11081     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11082     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11083 }

```

g-sm-short-desc *long* (\textsmaller{short}) acronym style that has an accompanying description (which the user needs to supply).

```

11084 \renewacronymstyle{long-sm-short-desc}%
11085 {%
11086   \GlsUseAcrEntryDispStyle{long-sm-short}%
11087 }%
11088 {%
11089   \GlsUseAcrStyleDefs{long-sm-short}%
11090   \renewcommand*{\GenericAcronymFields}{}%

```

```

11091 \renewcommand*\acronymsort}[2]{##2}%
11092 \renewcommand*\acronymentry}[1]{%
11093   \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11094   (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11095 }

```

short-long-desc *<short>* (*<long>*) acronym style that has an accompanying description (which the user needs to supply).

```

11096 \renewacronymstyle{short-long-desc}%
11097 {%
11098   \GlsUseAcrEntryDisplayStyle{short-long}%
11099 }%
11100 {%
11101   \GlsUseAcrStyleDefs{short-long}%
11102   \renewcommand*\GenericAcronymFields{}%
11103   \renewcommand*\acronymsort}[2]{##2}%
11104   \renewcommand*\acronymentry}[1]{%
11105     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11106     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11107 }

```

short-long-desc *<long>* (\textsc{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

11108 \renewacronymstyle{sc-short-long-desc}%
11109 {%
11110   \GlsUseAcrEntryDisplayStyle{sc-short-long}%
11111 }%
11112 {%
11113   \GlsUseAcrStyleDefs{sc-short-long}%
11114   \renewcommand*\GenericAcronymFields{}%
11115   \renewcommand*\acronymsort}[2]{##2}%
11116   \renewcommand*\acronymentry}[1]{%
11117     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11118     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11119 }

```

short-long-desc *<long>* (\textsmaller{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

11120 \renewacronymstyle{sm-short-long-desc}%
11121 {%
11122   \GlsUseAcrEntryDisplayStyle{sm-short-long}%
11123 }%
11124 {%
11125   \GlsUseAcrStyleDefs{sm-short-long}%
11126   \renewcommand*\GenericAcronymFields{}%
11127   \renewcommand*\acronymsort}[2]{##2}%
11128   \renewcommand*\acronymentry}[1]{%
11129     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11130     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%

```


11131 }

dua *<long>* only acronym style.

11132 \renewacronymstyle{dua}%
11133 {%

Check for long form in case this is a mixed glossary.

11134 \ifdefempty\glscustomtext
11135 {%
11136 \ifglshaslong{\glslabel}%
11137 {%
11138 \glsifplural
11139 {%

Plural form:

11140 \glscapscase
11141 {%

Plural form, don't adjust case:

11142 \gslongpluralaccessdisplay{\glentrylongpl{\glslabel}}{\glslabel}%
11143 \glsinsert
11144 }%
11145 {%

Plural form, make first letter upper case:

11146 \gslongpluralaccessdisplay{\Glentrylongpl{\glslabel}}{\glslabel}%
11147 \glsinsert
11148 }%
11149 {%

Plural form, all caps:

11150 \gslongpluralaccessdisplay
11151 {\mfirstucMakeUppercase{\glentrylongpl{\glslabel}}}{\glslabel}%
11152 \mfirstucMakeUppercase{\glsinsert}%
11153 }%
11154 }%
11155 {%

Singular form

11156 \glscapscase
11157 {%

Singular form, don't adjust case:

11158 \gslongaccessdisplay{\glentrylong{\glslabel}}{\glslabel}\glsinsert
11159 }%
11160 {%

Subsequent singular form, make first letter upper case:

11161 \gslongaccessdisplay{\Glentrylong{\glslabel}}{\glslabel}\glsinsert
11162 }%
11163 {%

Subsequent singular form, all caps:

```

11164      \glslongaccessdisplay
11165      {\mfirstucMakeUppercase
11166       {\glsentrylong{\glslabel}\glsinsert}}{\glslabel}%
11167      \mfirstucMakeUppercase{\glsinsert}%
11168      }%
11169      }%
11170      }%
11171      {%

```

Not an acronym:

```

11172      \glsgenentryfmt
11173      }%
11174      }%
11175      {\glscustomtext\glsinsert}%
11176      }%
11177      {%
11178      \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
11179      \renewcommand*{\acrfullfmt}[3]{%
11180        \glslink[##1]{##2}{%
11181          \glslongaccessdisplay{\glsentrylong{##2}}{##2}##3\space
11182          (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}})%
11183      \renewcommand*{\Acrfullfmt}[3]{%
11184        \glslink[##1]{##2}{%
11185          \glslongaccessdisplay{\Glsentrylong{##2}}{##2}##3\space
11186          (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}})%
11187      \renewcommand*{\ACRfullfmt}[3]{%
11188        \glslink[##1]{##2}{%
11189          \glslongaccessdisplay
11190          {\mfirstucMakeUppercase{\glsentrylong{##2}}{##2}##3\space
11191          (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}})%
11192      \renewcommand*{\acrfullplfmt}[3]{%
11193        \glslink[##1]{##2}{%
11194          \glslongpluralaccessdisplay
11195          {\glsentrylongpl{##2}}{##2}##3\space
11196          (\glsshortpluralaccessdisplay
11197          {\acronymfont{\glsentryshortpl{##2}}}{##2}})%
11198      \renewcommand*{\ACRfullplfmt}[3]{%
11199        \glslink[##1]{##2}{%
11200          \glslongpluralaccessdisplay
11201          {\Glsentrylongpl{##2}}{##2}##3\space
11202          (\glsshortpluralaccessdisplay
11203          {\acronymfont{\glsentryshortpl{##2}}}{##2}})%
11204      \renewcommand*{\ACRfullplplfmt}[3]{%
11205        \glslink[##1]{##2}{%
11206          \glslongpluralaccessdisplay
11207          {\mfirstucMakeUppercase{\glsentrylongpl{##2}}{##2}##3\space
11208          (\glsshortpluralaccessdisplay
11209          {\acronymfont{\glsentryshortpl{##2}}}{##2}})%
11210      \renewcommand*{\glsentryfull}[1]{%

```

```

11211 \glslongaccessdisplay{\glsentrylong{##1}}\space
11212 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})%
11213 }%
11214 \renewcommand*{\Glsentryfull}[1]{%
11215 \glslongaccessdisplay{\Glsentrylong{##1}}{##1}\space
11216 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})%
11217 }%
11218 \renewcommand*{\glsentryfullpl}[1]{%
11219 \glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}\space
11220 (\glsshortpluralaccessdisplay{\acronymfont{\glsentryshortpl{##1}}}{##1})%
11221 }%
11222 \renewcommand*{\Glsentryfullpl}[1]{%
11223 \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}\space
11224 (\glsshortpluralaccessdisplay{\acronymfont{\glsentryshortpl{##1}}}{##1})%
11225 }%
11226 \renewcommand*{\acronymentry}[1]{%
11227 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}%
11228 \renewcommand*{\acronymsort}[2]{##1}%
11229 \renewcommand*{\acronymfont}[1]{##1}%
11230 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11231 }

```

dua-desc *<long>* only acronym style with user-supplied description.

```

11232 \renewacronymstyle{dua-desc}%
11233 {%
11234 \GlsUseAcrEntryDispStyle{dua}%
11235 }%
11236 {%
11237 \GlsUseAcrStyleDefs{dua}%
11238 \renewcommand*{\GenericAcronymFields}{}%
11239 \renewcommand*{\acronymentry}[1]{%
11240 \glslongaccessdisplay{\acronymfont{\glsentrylong{##1}}}{##1}}%
11241 \renewcommand*{\acronymsort}[2]{##2}%
11242 }%

```

footnote *<short>*\footnote{*<long>*} acronym style.

```

11243 \renewacronymstyle{footnote}%
11244 {%
11245 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
11246 }%
11247 {%
11248 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%

```

Need to ensure hyperlinks are switched off on first use:

```

11249 \glshyperfirstfalse
11250 \renewcommand*{\genacrfullformat}[2]{%
11251 \glsshortaccessdisplay
11252 {\protect\firstacronymfont{\glsentryshort{##1}}}{##1}##2%

```

```

11253 \protect\footnote{\glslongaccessdisplay{\glsentrylong{##1}}{##1}}%
11254 }%
11255 \renewcommand*{\Genacrfullformat}[2]{%
11256 \glsshortaccessdisplay
11257   {\firstacronymfont{\Glsentryshort{##1}}{##1}##2%
11258 \protect\footnote{\glslongaccessdisplay{\glsentrylong{##1}}{##1}}%
11259 }%
11260 \renewcommand*{\genplacrfullformat}[2]{%
11261 \glsshortpluralaccessdisplay
11262   {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}##2%
11263 \protect\footnote{\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}}%
11264 }%
11265 \renewcommand*{\Genplacrfullformat}[2]{%
11266 \glsshortpluralaccessdisplay
11267   {\protect\firstacronymfont{\Glsentryshortpl{##1}}{##1}##2%
11268 \protect\footnote{\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}}%
11269 }%
11270 \renewcommand*{\acronymentry}[1]{%
11271 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}{##1}}%
11272 \renewcommand*{\acronymsort}[2]{##1}%
11273 \renewcommand*{\acronymfont}[1]{##1}%
11274 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%

```

Don't use footnotes for \acrfull:

```

11275 \renewcommand*{\acrfullfmt}[3]{%
11276 \glslink{##1}{##2}{%
11277 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}{##2}##3\space
11278 (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}}%
11279 \renewcommand*{\Acrfullfmt}[3]{%
11280 \glslink{##1}{##2}{%
11281 \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##2}}{##2}##3\space
11282 (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}}%
11283 \renewcommand*{\ACRfullfmt}[3]{%
11284 \glslink{##1}{##2}{%
11285 \glsshortaccessdisplay
11286   {\mfirstucMakeUppercase
11287   {\acronymfont{\glsentryshort{##2}}{##2}##3\space
11288   (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}}%
11289 \renewcommand*{\acrfullplfmt}[3]{%
11290 \glslink{##1}{##2}{%
11291 \glsshortpluralaccessdisplay
11292   {\acronymfont{\glsentryshortpl{##2}}{##2}##3\space
11293   (\glslongpluralaccessdisplay{\glsentrylongpl{##2}}{##2})}}}%
11294 \renewcommand*{\Acrfullplfmt}[3]{%
11295 \glslink{##1}{##2}{%
11296 \glsshortpluralaccessdisplay
11297   {\acronymfont{\Glsentryshortpl{##2}}{##2}##3\space
11298   (\glslongpluralaccessdisplay{\glsentrylongpl{##2}}{##2})}}}%
11299 \renewcommand*{\ACRfullplfmt}[3]{%
11300 \glslink{##1}{##2}{%

```

```

11301      \glsshortpluralaccessdisplay
11302      {\mfirstucMakeUppercase
11303      {\acronymfont{\glentryshortpl{##2}}}{##2}##3\space
11304      (\glslongpluralaccessdisplay{\glentrylongpl{##2}}{##2}}}%

```

Similarly for \glentryfull etc:

```

11305 \renewcommand*{\glentryfull}[1]{%
11306     \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}}{##1}\space
11307     (\glslongaccessdisplay{\glentrylong{##1}}{##1}}}%
11308 \renewcommand*{\Glsentryfull}[1]{%
11309     \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##1}}}{##1}\space
11310     (\glslongaccessdisplay{\glentrylong{##1}}{##1}}}%
11311 \renewcommand*{\glentryfullpl}[1]{%
11312     \glsshortpluralaccessdisplay
11313     {\acronymfont{\glentryshortpl{##1}}}{##1}\space
11314     (\glslongpluralaccessdisplay{\glentrylongpl{##1}}{##1}}}%
11315 \renewcommand*{\Glsentryfullpl}[1]{%
11316     \glsshortpluralaccessdisplay
11317     {\acronymfont{\Glsentryshortpl{##1}}}{##1}\space
11318     (\glslongpluralaccessdisplay{\glentrylongpl{##1}}{##1}}}%
11319 }

```

footnote-sc \textsc{<short>}\footnote{<long>} acronym style.

```

11320 \renewacronymstyle{footnote-sc}%
11321 {%
11322     \GlsUseAcrEntryDispStyle{footnote}%
11323 }%
11324 {%
11325     \GlsUseAcrStyleDefs{footnote}%
11326     \renewcommand{\acronymentry}[1]{%
11327         \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}}{##1}}
11328     \renewcommand{\acronymfont}[1]{\textsc{##1}}%
11329     \renewcommand*{\acrpluralsuffix}{\glstextup{\glspluralsuffix}}%
11330 }%

```

footnote-sm \textsmaller{<short>}\footnote{<long>} acronym style.

```

11331 \renewacronymstyle{footnote-sm}%
11332 {%
11333     \GlsUseAcrEntryDispStyle{footnote}%
11334 }%
11335 {%
11336     \GlsUseAcrStyleDefs{footnote}%
11337     \renewcommand{\acronymentry}[1]{%
11338         \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}}{##1}}
11339     \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
11340     \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11341 }%

```

footnote-desc <short>\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

11342 \renewacronymstyle{footnote-desc}%
11343 {%
11344   \GlsUseAcrEntryDisplayStyle{footnote}%
11345 }%
11346 {%
11347   \GlsUseAcrStyleDefs{footnote}%
11348   \renewcommand*{\GenericAcronymFields}{}%
11349   \renewcommand*{\acronymsort}[2]{##2}%
11350   \renewcommand*{\acronymentry}[1]{%
11351     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11352     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11353 }

```

ootnote-sc-desc \textsc{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

11354 \renewacronymstyle{footnote-sc-desc}%
11355 {%
11356   \GlsUseAcrEntryDisplayStyle{footnote-sc}%
11357 }%
11358 {%
11359   \GlsUseAcrStyleDefs{footnote-sc}%
11360   \renewcommand*{\GenericAcronymFields}{}%
11361   \renewcommand*{\acronymsort}[2]{##2}%
11362   \renewcommand*{\acronymentry}[1]{%
11363     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11364     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11365 }

```

ootnote-sm-desc \textsmaller{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

11366 \renewacronymstyle{footnote-sm-desc}%
11367 {%
11368   \GlsUseAcrEntryDisplayStyle{footnote-sm}%
11369 }%
11370 {%
11371   \GlsUseAcrStyleDefs{footnote-sm}%
11372   \renewcommand*{\GenericAcronymFields}{}%
11373   \renewcommand*{\acronymsort}[2]{##2}%
11374   \renewcommand*{\acronymentry}[1]{%
11375     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11376     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11377 }

```

Use \newacronymhook to modify the key list to set the access text to the long version by default.

```

11378 \renewcommand*{\newacronymhook}{%
11379   \edef\@gls@keylist{shortaccess=\the\glslongtok,%
11380     \the\glskeylisttok}%
11381   \expandafter\glskeylisttok\expandafter{\@gls@keylist}%

```

11382 }

1tNewAcronymDef Modify default style to use access text:

```
11383 \renewcommand*{\DefaultNewAcronymDef}{%
11384   \edef\@do@newglossaryentry{%
11385     \noexpand\newglossaryentry{\the\glslabeltok}%
11386     {%
11387       type=\acronymtype,%
11388       name={\the\glsshorttok},%
11389       description={\the\glslongtok},%
11390       descriptionaccess=\relax,
11391       text={\the\glsshorttok},%
11392       access={\noexpand\@glo@textaccess},%
11393       sort={\the\glsshorttok},%
11394       short={\the\glsshorttok},%
11395       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11396       shortaccess={\the\glslongtok},%
11397       long={\the\glslongtok},%
11398       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11399       descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11400       first={\noexpand\glslongaccessdisplay
11401         {\the\glslongtok}{\the\glslabeltok}\space
11402         (\noexpand\glsshortaccessdisplay
11403           {\the\glsshorttok}{\the\glslabeltok})},%
11404       plural={\the\glsshorttok\acrpluralsuffix},%
11405       firstplural={\noexpand\glslongpluralaccessdisplay
11406         {\noexpand\@glo@longpl}{\the\glslabeltok}\space
11407         (\noexpand\glsshortpluralaccessdisplay
11408           {\noexpand\@glo@shortpl}{\the\glslabeltok})},%
11409       firstaccess=\relax,
11410       firstpluralaccess=\relax,
11411       textaccess={\noexpand\@glo@shortaccess},%
11412       \the\glskeylisttok
11413     }%
11414   }%
11415   \let\@org@gls@assign@firstpl\gls@assign@firstpl
11416   \let\@org@gls@assign@plural\gls@assign@plural
11417   \let\@org@gls@assign@descplural\gls@assign@descplural
11418   \def\gls@assign@firstpl##1##2{%
11419     \@gls@expand@field{##1}{firstpl}{##2}%
11420   }%
11421   \def\gls@assign@plural##1##2{%
11422     \@gls@expand@field{##1}{plural}{##2}%
11423   }%
11424   \def\gls@assign@descplural##1##2{%
11425     \@gls@expand@field{##1}{descplural}{##2}%
11426   }%
11427   \@do@newglossaryentry
11428   \let\gls@assign@firstpl\@org@gls@assign@firstpl
```

```

11429 \let\gls@assign@plural\@org@gls@assign@plural
11430 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11431 }

```

teNewAcronymDef

```

11432 \renewcommand*{\DescriptionFootnoteNewAcronymDef}{%
11433   \edef\@do@newglossaryentry{%
11434     \noexpand\newglossaryentry{\the\glslabeltok}%
11435     {%
11436       type=\acronymtype,%
11437       name={\noexpand\acronymfont{\the\glsshorttok}},%
11438       sort={\the\glsshorttok},%
11439       text={\the\glsshorttok},%
11440       short={\the\glsshorttok},%
11441       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11442       shortaccess={\the\glslongtok},%
11443       long={\the\glslongtok},%
11444       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11445       access={\noexpand\@glo@textaccess},%
11446       plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11447       symbol={\the\glslongtok},%
11448       symbolplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11449       firstpluralaccess=\relax,
11450       textaccess={\noexpand\@glo@shortaccess},%
11451       \the\glskeylisttok
11452     }%
11453   }%
11454   \let\@org@gls@assign@firstpl\gls@assign@firstpl
11455   \let\@org@gls@assign@plural\gls@assign@plural
11456   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
11457   \def\gls@assign@firstpl##1##2{%
11458     \@@gls@expand@field{##1}{firstpl}{##2}%
11459   }%
11460   \def\gls@assign@plural##1##2{%
11461     \@@gls@expand@field{##1}{plural}{##2}%
11462   }%
11463   \def\gls@assign@symbolplural##1##2{%
11464     \@@gls@expand@field{##1}{symbolplural}{##2}%
11465   }%
11466   \@do@newglossaryentry
11467   \let\gls@assign@plural\@org@gls@assign@plural
11468   \let\gls@assign@firstpl\@org@gls@assign@firstpl
11469   \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11470 }

```

onNewAcronymDef

```

11471 \renewcommand*{\DescriptionNewAcronymDef}{%
11472   \edef\@do@newglossaryentry{%
11473     \noexpand\newglossaryentry{\the\glslabeltok}%

```



```

11474 {%
11475     type=\acronymtype,%
11476     name={\noexpand
11477         \acrnameformat{\the\glssshorttok}{\the\glslongtok}},%
11478     access={\noexpand\@glo@textaccess},%
11479     sort={\the\glssshorttok},%
11480     short={\the\glssshorttok},%
11481     shortplural={\the\glssshorttok\noexpand\acrpluralsuffix},%
11482     shortaccess={\the\glslongtok},%
11483     long={\the\glslongtok},%
11484     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11485     first={\the\glslongtok},%
11486     firstaccess=\relax,
11487     firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11488     text={\the\glssshorttok},%
11489     textaccess={\the\glslongtok},%
11490     plural={\the\glssshorttok\noexpand\acrpluralsuffix},%
11491     symbol={\noexpand\@glo@text},%
11492     symbolaccess={\noexpand\@glo@textaccess},%
11493     symbolplural={\noexpand\@glo@plural},%
11494     firstpluralaccess=\relax,
11495     textaccess={\noexpand\@glo@shortaccess},%
11496     \the\glskeylisttok}%
11497 }%
11498 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11499 \let\@org@gls@assign@plural\gls@assign@plural
11500 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
11501 \def\gls@assign@firstpl##1##2{%
11502     \@gls@expand@field{##1}{firstpl}{##2}%
11503 }%
11504 \def\gls@assign@plural##1##2{%
11505     \@gls@expand@field{##1}{plural}{##2}%
11506 }%
11507 \def\gls@assign@symbolplural##1##2{%
11508     \@gls@expand@field{##1}{symbolplural}{##2}%
11509 }%
11510 \@do@newglossaryentry
11511 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11512 \let\gls@assign@plural\@org@gls@assign@plural
11513 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11514 }

```

teNewAcronymDef

```

11515 \renewcommand*{\FootnoteNewAcronymDef}{%
11516     \edef\@do@newglossaryentry{%
11517         \noexpand\newglossaryentry{\the\glslabeltok}%
11518         {%
11519             type=\acronymtype,%
11520             name={\noexpand\acronymfont{\the\glssshorttok}},%

```

```

11521 sort={\the\glsshorttok},%
11522 text={\the\glsshorttok},%
11523 textaccess={\the\glslongtok},%
11524 access={\noexpand\@glo@textaccess},%
11525 plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11526 short={\the\glsshorttok},%
11527 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11528 long={\the\glslongtok},%
11529 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11530 description={\the\glslongtok},%
11531 descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11532 \the\glskeylisttok
11533 }%
11534 }%
11535 \let\@org@gls@assign@plural\gls@assign@plural
11536 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11537 \let\@org@gls@assign@descplural\gls@assign@descplural
11538 \def\gls@assign@firstpl##1##2{%
11539   \@@gls@expand@field{##1}{firstpl}{##2}%
11540 }%
11541 \def\gls@assign@plural##1##2{%
11542   \@@gls@expand@field{##1}{plural}{##2}%
11543 }%
11544 \def\gls@assign@descplural##1##2{%
11545   \@@gls@expand@field{##1}{descplural}{##2}%
11546 }%
11547 \do@newglossaryentry
11548 \let\gls@assign@plural\@org@gls@assign@plural
11549 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11550 \let\gls@assign@descplural\@org@gls@assign@descplural
11551 }

```

11NewAcronymDef

```

11552 \renewcommand*{\SmallNewAcronymDef}{%
11553   \edef\@do@newglossaryentry{%
11554     \noexpand\newglossaryentry{\the\glslabeltok}%
11555     {%
11556       type=\acronymtype,%
11557       name={\noexpand\acronymfont{\the\glsshorttok}},%
11558       access={\noexpand\@glo@symbolaccess},%
11559       sort={\the\glsshorttok},%
11560       short={\the\glsshorttok},%
11561       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11562       shortaccess={\the\glslongtok},%
11563       long={\the\glslongtok},%
11564       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11565       text={\noexpand\@glo@short},%
11566       textaccess={\noexpand\@glo@shortaccess},%
11567       plural={\noexpand\@glo@shortpl},%

```

```

11568     first={\the\glslongtok},%
11569     firstaccess=\relax,
11570     firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11571     description={\noexpand\@glo@first},%
11572     descriptionplural={\noexpand\@glo@firstplural},%
11573     symbol={\the\glsshorttok},%
11574     symbolaccess={\the\glslongtok},%
11575     symbolplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11576     \the\glskeylisttok
11577 }%
11578 }%
11579 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11580 \let\@org@gls@assign@plural\gls@assign@plural
11581 \let\@org@gls@assign@descplural\gls@assign@descplural
11582 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
11583 \def\gls@assign@firstpl##1##2{%
11584   \@@gls@expand@field{##1}{firstpl}{##2}%
11585 }%
11586 \def\gls@assign@plural##1##2{%
11587   \@@gls@expand@field{##1}{plural}{##2}%
11588 }%
11589 \def\gls@assign@descplural##1##2{%
11590   \@@gls@expand@field{##1}{descplural}{##2}%
11591 }%
11592 \def\gls@assign@symbolplural##1##2{%
11593   \@@gls@expand@field{##1}{symbolplural}{##2}%
11594 }%
11595 \do@newglossaryentry
11596 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11597 \let\gls@assign@plural\@org@gls@assign@plural
11598 \let\gls@assign@descplural\@org@gls@assign@descplural
11599 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11600 }

```

The following are kept for compatibility with versions before 3.0:

sshortaccesskey

```

11601 \newcommand*{\glsshortaccesskey}{\glsshortkey access}%

```

pluralaccesskey

```

11602 \newcommand*{\glsshortpluralaccesskey}{\glsshortpluralkey access}%

```

lslongaccesskey

```

11603 \newcommand*{\glslongaccesskey}{\glslongkey access}%

```

pluralaccesskey

```

11604 \newcommand*{\glslongpluralaccesskey}{\glslongpluralkey access}%

```

5.5 Debugging Commands

owglonameaccess

```
11605 \newcommand*{\showglonameaccess}[1]{%
11606   \expandafter\show\csname glo@\glsdetoklabel{#1}@textaccess\endcsname
11607 }
```

owglotextaccess

```
11608 \newcommand*{\showglotextaccess}[1]{%
11609   \expandafter\show\csname glo@\glsdetoklabel{#1}@textaccess\endcsname
11610 }
```

glopluralaccess

```
11611 \newcommand*{\showglopluralaccess}[1]{%
11612   \expandafter\show\csname glo@\glsdetoklabel{#1}@pluralaccess\endcsname
11613 }
```

wglofirstaccess

```
11614 \newcommand*{\showglofirstaccess}[1]{%
11615   \expandafter\show\csname glo@\glsdetoklabel{#1}@firstaccess\endcsname
11616 }
```

rstpluralaccess

```
11617 \newcommand*{\showglofirstpluralaccess}[1]{%
11618   \expandafter\show\csname glo@\glsdetoklabel{#1}@firstpluralaccess\endcsname
11619 }
```

glosymbolaccess

```
11620 \newcommand*{\showglosymbolaccess}[1]{%
11621   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolaccess\endcsname
11622 }
```

bolpluralaccess

```
11623 \newcommand*{\showglosymbolpluralaccess}[1]{%
11624   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolpluralaccess\endcsname
11625 }
```

owglodescaccess

```
11626 \newcommand*{\showglodescaccess}[1]{%
11627   \expandafter\show\csname glo@\glsdetoklabel{#1}@descaccess\endcsname
11628 }
```

escpluralaccess

```
11629 \newcommand*{\showglodescpluralaccess}[1]{%
11630   \expandafter\show\csname glo@\glsdetoklabel{#1}@descpluralaccess\endcsname
11631 }
```

wgloshortaccess

```
11632 \newcommand*{\showgloshortaccess}[1]{%  
11633   \expandafter\show\csname glo@\glsdetoklabel{#1}@shortaccess\endcsname  
11634 }
```

ortpluralaccess

```
11635 \newcommand*{\showgloshortpluralaccess}[1]{%  
11636   \expandafter\show\csname glo@\glsdetoklabel{#1}@shortpluralaccess\endcsname  
11637 }
```

owglolongaccess

```
11638 \newcommand*{\showglolongaccess}[1]{%  
11639   \expandafter\show\csname glo@\glsdetoklabel{#1}@longaccess\endcsname  
11640 }
```

ongpluralaccess

```
11641 \newcommand*{\showglolongpluralaccess}[1]{%  
11642   \expandafter\show\csname glo@\glsdetoklabel{#1}@longpluralaccess\endcsname  
11643 }
```

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.

```
11644 \NeedsTeXFormat{LaTeX2e}
11645 \ProvidesPackage{glossaries-babel}[2017/08/10 v4.31 (NLCT)]
```

Load tracklang to obtain language settings.

```
11646 \RequirePackage{tracklang}
11647 \let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
11648 \AnyTrackedLanguages
11649 {%
11650   \ForEachTrackedDialect{\this@dialect}{%
11651     \IfTrackedLanguageFileExists{\this@dialect}%
11652       {glossaries-}% prefix
11653       {.ldf}%
11654       {%
11655         \RequireGlossariesLang{\CurrentTrackedTag}%
11656       }%
11657       {%
11658         \PackageWarningNoLine{glossaries}%
11659           {No language module detected for ‘\this@dialect’.\MessageBreak
11660             Language modules need to be installed separately.\MessageBreak
11661             Please check on CTAN for a bundle called\MessageBreak
11662             ‘glossaries-\CurrentTrackedLanguage’ or similar}%
11663       }%
11664     }%
11665   }%
11666 }
```

6.1 Polyglossia Captions

Language support has now been split off into independent language modules.

```
11667 \NeedsTeXFormat{LaTeX2e}
11668 \ProvidesPackage{glossaries-polyglossia}[2017/08/10 v4.31 (NLCT)]
```

Load tracklang to obtain language settings.

```
11669 \RequirePackage{tracklang}
11670 \let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
11671 \AnyTrackedLanguages
```

```

11672 {%
11673     \ForEachTrackedDialect{\this@dialect}{%
11674         \IfTrackedLanguageFileExists{\this@dialect}%
11675         {glossaries-}% prefix
11676         {.ldf}%
11677         {%
11678             \RequireGlossariesLang{\CurrentTrackedTag}%
11679         }%
11680         {%
11681             \PackageWarningNoLine{glossaries}%
11682             {No language module detected for ‘\this@dialect’.\MessageBreak
11683             Language modules need to be installed separately.\MessageBreak
11684             Please check on CTAN for a bundle called\MessageBreak
11685             ‘glossaries-\CurrentTrackedLanguage’ or similar}%
11686         }%
11687     }%
11688 }%
11689 {}%

```

Glossary

`makeindex` An indexing application. [11](#), [26](#), [27](#), [175](#)

`xindy` An flexible indexing application with multilingual support written in Perl. [11](#), [26](#), [27](#), [175](#)

Change History

1.01 (2007-05-17)	numberline: numberline option added .. 6
General: Added range facility in format key 111	1.12 (2008-03-08)
\writeist: Added spaces after \delimN and \delimR in ist file 157	\@GLSpl: now uses
1.04 (2007-08-03)	\glentrydescplural and
General: Added \glstextformat 95	\glentrysymbolplural instead of
1.05 (2007-08-10)	\glentrydesc and
\glossarysection: added \@mkboth to \glossarysection 39	\glentrysymbol 125
\gls@defglossaryentry: Changed the default value of the sort key to just the value of the name key 79	\@Glspl@: now uses
1.07 (2007-09-13)	\glentrydescplural and
\@gls@link: fixed bug caused by \theglentrycounter setting the page number too soon 109	\glentrysymbolplural instead of
\glsadd: fixed bug caused by \theglentrycounter setting the page number too soon 155	\glentrydesc and
1.08 (2007-10-13)	\glentrysymbol 123
General: Added babel support 33	General: added check for \hypertarget separate to \hyperlink (memoir defines \hyperlink but not \hypertarget) 119
listgroup: changed listgroup style to use \glsgetgrouptitle 269	descriptionplural: new 62
altlistgroup: changed altlistgroup style to use \glsgetgrouptitle 270	\gls@defglossaryentry: Changed default first plural to be first key with s appended (was text key with s appended) 79
1.1 (2008-02-22)	descriptionplural support added 79
\@glossarysection: numbered sections and auto label added 40	symbolplural support added 79
\@gls@tmpb: changed \toksdef to \newtoks 113	\Glsentrydescplural: New 148
\@gls@toc: numberline added 41	\glentrydescplural: New 148
\@p@glossarysection: numbered sections and auto label added 40	\Glsentrysymbolplural: New 149
General: amsgen now loaded (\new@ifnextchar needed) 4	\glentrysymbolplural: New 149
translate: translate option added 24	\SetDescriptionFootnoteAcronymStyle: Added \protect before \footnote and \glslink 235
\setglossarysection: new 40	\SetFootnoteAcronymStyle: Added \protect before \footnote and \glslink 241
numberedsection: numberedsection package option added 7	symbolplural: new 63

1.13 (2008-05-10)	
General: fixed bug that ignored 3rd parameter	126–133
\ACRfullpl: new	216
\Acrfullpl: new	216
\acrfullpl: new	215
\acrpluralsuffix: New	213
\gls@defglossaryentry: Changed default first value	79
Changed default firstplural value	79
Removed restriction on only using \newglossaryentry in the preamble	84
\newacronym: Removed restriction on only using \newacronym in the preamble	213
1.14 (2008-06-17)	
\@gls@hypergroup: new	264
General: added nonumberlist key to \printglossary	200
added numberedsection key to \printglossary	198
\firstacronymfont: new	217
\glsautoprefix: new	7
\glsnavhyperlink: changed \edef to \protected@edef	263
\glsnavhypertarget: added write to aux file	263
\glsnavigation: changed to only use labels for groups that are present ..	264
1.15 (2008-08-15)	
\@gls@link: added \glslabel	109
\gls@defglossaryentry: check for \@glo@first in description	83
check for \@glo@text in symbol	83
\gls@hypergroup: new	264
\glsnavhypertarget: added check if rerun required	263
\glssettoctitle: new	32
\printglossary: changed the way the TOC title is set	184
1.16 (2008-08-27)	
\@GLS@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	122
\@GLSpl: Test glossary type is \acronymtype in addition to checking if footnote option has been used	125
\@Gls@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	122
\@Glspl@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	124
\@gls@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	121
\@glsdisp: Test glossary type is \acronymtype in addition to checking if footnote option has been used	125
\@glspl@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	123
\@glstarget: raised the hypertarget so the target text doesn't scroll off the top of the page	119
\gls@defglossaryentry: Changed def to let	79
1.17 (2008-12-26)	
\@do@wrglossary: new	179
\@do@seeglossary: new	181
\@glo@storeentry: new	85
\@gls@glossary: changed definition to use \index instead of \@index	176
\@glsdefaultplural: new	66
\@glsdefaultsort: new	67
\@glshypernumber: new	210
\@glsnoname: new	66
\@glsnonextpages: new	200
General: added xindy support	26
parent: new	64
see: new	63
\gls@defglossaryentry: added nonumberlist key	79
added parent key	79
added see key	79
Stored main part of entry format when entry is defined	84
\gls@suffixF: new	37
\gls@suffixFF: new	37
\gls@wrglossary: modified to allow for xindy support	176

\glshyperlink: new	154	\SetDescriptionFootnoteAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	235
\glshypernumber: modified to allow material to be attached to location	210	\SetFootnoteAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	241
\glshnavhyperlink: replaced \hyperlink to \@glslink	263	\SetSmallAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	244
\glshnavhypertarget: replaced \hypertarget to \@glstarget ...	263	2.01 (2009 May 30) \@glsl@link: moved \@do@wrglossary before term is displayed to prevent unwanted whatsit	110
\glsssee: new	182	\forall glossaries: replaced \ifthenelse with \ifx	51
\glssseeformat: new	182	\forall glossentries: replaced \ifthenelse with \ifx	51
\glssSetSuffixF: new	37	\glssdefmain: new	14
\glssSetSuffixFF: new	37	\glssdescwidth: changed \linewidth to \hsize	271, 293
\ifglsxindy: new	26	\glsslistdottedwidth: changed \linewidth to \hsize	271
\istfilename: added xindy support ...	36	\glsspagelistwidth: changed \linewidth to \hsize	271, 293
\newglossarystyle: made \newglossarystyle long	209	nomain: added nomain package option	14
\nopostdesc: new	35	\writeist: removed item_02 - no such makeindex key	162
nonumberlist: new	64	2.02 (2007-07-13) \@printglossary: suppressed warning globally rather than locally	186
\printglossary: added check to determine if \printglossary is already defined	184	2.02 (2009-07-13) \glossarysection: changed \@mkboth to \glossarymark	39
added print language to aux file	184	\glsglossarymark: New	39
order: order package option added	26	2.03 (2009-09-23) \@GLS@: Added check for hyperfirst	122
\writeist: added xindy support	157	\@GLSpl: Added check for hyperfirst	125
1.18 (2009-01-14) \@glsl@loadlist: new	9	\@GLs@: Added check for hyperfirst	122
\@glsl@loadlong: new	8	\@GLspl@: Added check for hyperfirst	124
\@glsl@loadsuper: new	9	\@glsl@: Added check for hyperfirst	121
\@glsl@loadtree: new	9	\@glsl@link: new	108
\glsl@defglossaryentry: Changed default value of sort to \@glsldefaultsort	79	\@glsl@link: added \leavevmode	109
moved sort sanitization to \newglossaryentry	83	Moved entry existence check to avoid duplicate code	109
\glstarget: new	203	\@glsl@disp: Added check for hyperfirst	125
\oldacronym: new	213	\@glspl@: Added check for hyperfirst	123
nolist: new	9	\glsglossarymark: Added check to see if it's already defined	39
nolong: new	8	hyperfirst: new	25
sort: moved sanitization to \newglossaryentry	62		
nostyles: new	9		
nosuper: new	9		
notree: new	9		
1.19 (2009-03-02) \glsclearpage: new	41		
\glssdisp: new	125		
\SetDescriptionAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	239		

2.04 (2009-11-10)	
\@GLS@: Changed test to check if glossary type has been identified as a list of acronyms	122
\@GLSpl: Changed test to check if glossary type has been identified as a list of acronyms	125
\@Gls@: Changed test to check if glossary type has been identified as a list of acronyms	122
\@Glspl@: Changed test to check if glossary type has been identified as a list of acronyms	124
\@glossaryentryfield: new	85
\@glossarysubentryfield: new	85
\@gls@: Changed test to check if glossary type has been identified as a list of acronyms	121
\@glsacronymlists: new	15
\@glsdisp: Changed test to check if glossary type has been identified as a list of acronyms	125
\@glspl@: Changed test to check if glossary type has been identified as a list of acronyms	123
\@newglossaryentryposthook: new ..	85
\@newglossaryentryprehook: new ...	85
acronymlists: new	17
\DeclareAcronymList: new	16
\DefineAcronymSynonyms: new	230
\gls@defglossaryentry: added user1-6 keys	80
\glsadd: fixed bug that ignored counter	155
\Glsentryuseri: new	150
\glsentryuseri: new	150
\Glsentryuserii: new	151
\glsentryuserii: new	151
\Glsentryuseriii: new	151
\glsentryuseriii: new	151
\Glsentryuseriv: new	151
\glsentryuseriv: new	151
\Glsentryuserv: new	151
\glsentryuserv: new	151
\Glsentryuservi: new	152
\glsentryuservi: new	152
\ns@newglossary: added check to determine if \gls@<type>@display and \gls@<type>@displayfirst have been defined.	59
\SetAcronymLists: new	17
\SetDefaultAcronymDisplayStyle: new	232
\SetDefaultAcronymStyle: new	233
\SetDescriptionAcronymDisplayStyle: new	237
\SetDescriptionDUAAcronymDisplayStyle: new	236
\SetDescriptionFootnoteAcronymDisplayStyle: new	233
\SetDUADisplayStyle: new	245
\SetFootnoteAcronymDisplayStyle: new	240
\SetSmallAcronymDisplayStyle: new	242
2.05 (2010-02-06)	
\@glsdisp: Added closing brace. Patch provided by Sergiu Dotenco	125
Removed spurious brace. Patch provided by Sergiu Dotenco	126
\writeist: Added \string before opening and closing braces. Patch provided by Segiu Dotenco	162
2.06 (2010-06-14)	
\altnewglossary: new	59
\CustomAcronymFields: new	247
\CustomNewAcronymDef: new	247
\SetCustomDisplayStyle: new	247
\SetCustomStyle: new	248
2.07 (2010-07-10)	
General: glsadd format key stored in \@glsnumberformat (was mistakenly stored in \@glo@format)	155
3.0 (2010-07-12)	
\@makeglossary: Added check for savewrites	167
\gls@wrglossary: modified to take into account savewrites	176
3.0 (2010/03/31)	
\@set@glo@numformat: added 4th argument	111
3.0 (2011-04-02)	
\@do@wrglossary: added check for hyper location prefix	179
modified to use new format	179
\@@glossarysec: replaced \@ifundefined with \ifcsundef ...	6
\@do@seeglossary: Sanitize and escape cross-referencing information	181
\@gls@counterwithin: new	10

\@gls@ifinlist: new	42	\glsadd: added	
\@gls@link: added		\@gls@saveentrycounter	155
\@gls@saveentrycounter	110	\GlsAddXdyCounters: new	42
added \@gls@setsort	110	\glseentrycounterlabel: new	202
\@gls@saveentrycounter: new	110	\glseentryitem: new	203
\@gls@setupsort@def: new	12	\glseentrylong: new	152
\@gls@setupsort@standard: new	11	\glseentrylong: new	152
\@gls@setupsort@use: new	12	\glseentrylongpl: new	152
\@gls@xdy@locationlist: new	45	\glseentrylongpl: new	152
\@gls@link: replaced \@ifundefined		\glseentryshort: new	152
with \ifcsundef	119	\glseentryshort: new	152
\@gls@nextpages: new	200	\glseentryshortpl: new	152
\@print@glossary: replaced		\glseentryshortpl: new	152
\@ifundefined with \ifcsundef ..	187	\glsggetgrouptitle: replaced	
\@printglossary: added		\@ifundefined with \ifcsundef ..	207
\currentglossary	185	\glsglossarymark: replaced	
added \glsgnextpages	186	\@ifundefined with \ifcsundef ..	39
make toctitle default to title	185	\glshyperlink: changed default from	
\@xdy@attributelist: new	42	\glseentryname to \glseentrytext	154
General: added prefix to hyperlink	211	\glshypernumber: replaced	
etoolbox now loaded	4	\@ifundefined with \ifcsundef ..	210
replaced \@ifundefined with		\glsgnumberformat: replaced	
\ifcsundef	31, 33, 106, 198	\@ifundefined with \ifcsundef ..	37
\acrfootnote: new	233	\glsgrefentry: new	202
\ACRfull: added starred version	215	\glsgresetsubentrycounter: new ...	201
\Acrfull: added starred version	215	\glsgseeitem: hyperlink uses	
\acrfull: added starred version	214	\glsgseeitemformat instead of	
\ACRfullpl: added starred version ...	216	\glseentryname	183
\Acrfullpl: added starred version ...	216	\glsgseeitemformat: new	183
\acrfullpl: added starred version ...	215	\glsgsortnumberfmt: new	11
\acrlinkfootnote: new	233	\glsgstepentry: new	202
\acrlinkfootnote: new	233	\glsgstepsubentry: new	202
savewrites: new	28	\glsgsubentrycounterlabel: new ...	202
see: added \@glo@seeautonumberlist	63	\glsgsubentryitem: new	203
seeautonumberlist: new	8	theglossary: replaced \@ifundefined	
\glossarysection: replaced		with \ifcsundef	203
\@ifundefined with \ifcsundef ..	39	short: new	66
\glossarystyle: replaced		shortplural: new	66
\@ifundefined with \ifcsundef ..	208	\ifglossaryexists: replaced	
\gls@codepage: replaced		\@ifundefined with \ifcsundef ..	51
\@ifundefined with \ifcsundef ..	27	\ifglseentryexists: replaced	
\gls@defglossaryentry: added		\@ifundefined with \ifcsundef ..	52
\@gls@defsort	83	\istfile: deprecated	175
added short and long keys	80	glossaryentry: new	201
replaced \@ifundefined with		glossarysubentry: new	201
\ifcsundef	80	\newglossaryentry: replaced	
\gls@doclearpage: replaced		\DeclareRobustCommand with	
\@ifundefined with \ifcsundef ..	41	\newrobustcmd	69

<code>\newglossarystyle</code> : replaced	
<code>\@ifundefined</code> with <code>\ifcsundef</code> .	209
<code>\ns@newglossary</code> : added	
<code>\@gls@defsortcount</code>	59
replaced <code>\@ifundefined</code> with	
<code>\ifcsundef</code>	59
<code>entrycounter</code> : new	10
<code>entrycounterwithin</code> : new	10
<code>\oldacronym</code> : replaced <code>\@ifundefined</code>	
with <code>\ifcsundef</code>	213
compatible-2.07: compatible-2.07	
option added	28
<code>long</code> : new	66
<code>longplural</code> : new	66
<code>nonumberlist</code> : now boolean	64
<code>sort</code> : new	10
<code>counter</code> : replaced <code>\@ifundefined</code> with	
<code>\ifcsundef</code>	63
<code>\printglossary</code> : replaced	
<code>\@ifundefined</code> with <code>\ifcsundef</code> .	184
<code>\SetDescriptionFootnoteAcronymDisplayStyle</code>	
expanded options link options . . .	233
<code>\setentrycounter</code> : added optional	
argument	208
<code>\showacronymlists</code> : new	253
<code>\showglocounter</code> : new	250
<code>\showgloDESC</code> : new	251
<code>\showgloDESCplural</code> : new	252
<code>\showglofirst</code> : new	250
<code>\showglofirstpl</code> : new	250
<code>\showgloflag</code> : new	253
<code>\showgloindex</code> : new	253
<code>\showglolevel</code> : new	249
<code>\showglongame</code> : new	251
<code>\showgloparent</code> : new	249
<code>\showgloplural</code> : new	249
<code>\showglosort</code> : new	252
<code>\showglossaries</code> : new	253
<code>\showglossarycounter</code> : new	254
<code>\showglossaryentries</code> : new	254
<code>\showglossaryin</code> : new	254
<code>\showglossaryout</code> : new	254
<code>\showglossarytitle</code> : new	254
<code>\showglosymbol</code> : new	252
<code>\showglosymbolplural</code> : new	252
<code>\showglotext</code> : new	249
<code>\showglotype</code> : new	250
<code>\showglouserii</code> : new	250
<code>\showglouseriii</code> : new	251
<code>\showglouseriv</code> : new	251
<code>\showglouserv</code> : new	251
<code>\showglouservi</code> : new	251
<code>subentrycounter</code> : new	10
<code>\writeist</code> : added xindy-only macro	
definitions to glossary open tag . . .	159
modified to support new format . . .	157
3.01 (2011-04-12)	
<code>\@glswritefiles</code> : added check for	
empty glossaries	175
General: made robust	122
<code>\ACRfull</code> : made robust	215
<code>\Acrfull</code> : made robust	215
<code>\acrfull</code> : made robust	214
<code>\acrfullformat</code> : removed	
<code>\acronymfont</code> as it should already be	
set in the second argument.	214
<code>\ACRfullpl</code> : made robust	216
<code>\Acrfullpl</code> : made robust	216
<code>\acrfullpl</code> : made robust	215
<code>\ACRlong</code> : made robust	143
<code>\Acrlong</code> : made robust	142
<code>\acrlong</code> : made robust	142
<code>\ACRlongpl</code> : made robust	145
<code>\Acrlongpl</code> : made robust	144
<code>\acrlongpl</code> : made robust	144
<code>\ACRshort</code> : made robust	139
<code>\Acrshort</code> : made robust	139
<code>\acrshort</code> : made robust	138
<code>\ACRshortpl</code> : made robust	141
<code>\Acrshortpl</code> : made robust	141
<code>\acrshortpl</code> : made robust	140
<code>\Gls</code> : made robust	121
<code>\glsadd</code> : made robust	155
<code>\glsaddall</code> : made robust	155
<code>\GLSdesc</code> : made robust	131
<code>\Glsdesc</code> : made robust	130
<code>\glsdesc</code> : made robust	130
<code>\GLSdescplural</code> : made robust	132
<code>\Glsdescplural</code> : made robust	131
<code>\glsdescplural</code> : made robust	131
<code>\glsfirst</code> : made robust	127
<code>\GLSfirstplural</code> : made robust	129
<code>\Glsfirstplural</code> : made robust	129
<code>\glsfirstplural</code> : made robust	129
<code>\glslink</code> : made robust	108
<code>\GLSname</code> : made robust	130
<code>\Glsname</code> : made robust	130

\glsname: made robust	129	\@printglossary: add a way to fetch current entry label	186
\GLSpl: made robust	124	savenumberlist: new	8
\Glspl: made robust	123	ucmark: new	10
\glspl: made robust	123	\gls@defglossaryentry: added numberlist element	83
\GLSplural: made robust	128	\gls@save@numberlist: new	183
\GLSsymbol: made robust	132	\gls@wrglossary: added check for glossary file defined	177
\Glsymbol: made robust	132	\glsdisplaynumberlist: new	153
\glssymbol: made robust	132	\glsentrycounter: set default value ..	110
\GLSsymbolplural: made robust	133	\Glsentryfull: fixed bug (replaced \glsentryshortpl with \glsentryshort)	153
\Glsymbolplural: made robust	133	\glsentryfullpl: fixed bug (replaced \glsentryshort with \glsentryshortpl)	153
\Glstext: made robust	127	\glsentrynumberlist: new	153
\glstext: made robust	126	\glsmoveentry: new	85
\GLSuseri: made robust	134	\glsresetsubentrycounter: new ...	201
\Glsuseri: made robust	134	\ifglshaschildren: new	54
\glsuseri: made robust	134	\ifglshasparent: new	54
\GLSuserii: made robust	135	\makeglossaries: added list parser ..	170
\Glsuserii: made robust	135	indexonlyfirst: new	25
\glsuserii: made robust	134	\renewglossarystyle: new	209
\GLSuseriii: made robust	136	\showglossaryentries: fixed misspelt command	254
\Glsuseriii: made robust	135	\SmallNewAcronymDef: fixed broken short and long plural	243
\glsuseriii: made robust	135		
\GLSuseriv: made robust	136		
\Glsuseriv: made robust	136		
\glsuseriv: made robust	136		
\GLSuseriv: made robust	137		
\Glsuseriv: made robust	137		
\glsuseriv: made robust	137		
\GLSuservi: made robust	138		
\Glsuservi: made robust	138		
\glsuservi: made robust	138		
		3.03 (2012/09/21)	
		\@gls@sanitizesort: new	20
		\@gls@setupsort@standard: used \@gls@sanitizesort	11
		\@printglossary: allow title to override default toctitle	185
		General: allow title to set toctitle	197
		\glsinlinedescformat: new	267
		\glsinlineemptydescformat: new ..	267
		\glsinlinenameformat: new	267
		\glsinlinepostchild: new	267
		\glsinlinesubdescformat: new	267
		\glsinlinesubnameformat: new	267
		\glspostinline: replaced “.” with \glspostdescription	267
		list: added check for glsnogroupskip ..	268
		altlongragged4col: added check for glsnogroupskip	286
		altsuperragged4col: added check for glsnogroupskip	304
3.02 (2012-05-19)			
\glsnumlistlastsep: new	154		
\glsnumlistsep: new	154		
3.02 (2012-05-21)			
\@do@wrglossary: changed \@glslocref to \theglsentrycounter	180		
\@do@wrglossary: changed \@do@wr@glossary to test for indexonlyfirst option; put old \@do@wr@glossary code into \@do@wrglossary	177		
\@gls@missingnumberlist: new	66		
\@glswritefiles: added check for existence of token in case \makeglossaries has been omitted	175		

alttree: added check for		\gls@disablepagerefexpansion: new	177
glsnogroupskip	314	\gls@numberpage: new	178
index: added check for glsnogroupskip	308	\gls@protected@pagefmts: new	177
nogroupskip: new	10	\gls@romanpage: new	178
long: added check for glsnogroupskip	272	\glsdefmain: added check for doc	
long3col: added check for		package	14
glsnogroupskip	273	\glsorg@endtheglossary: new	5
long4col: added check for		\glsorg@theglossary: new	5
glsnogroupskip	275	\PrintChanges: new	5
longragged: added check for		3.05 (2013-04-21)	
glsnogroupskip	283	@@do@wrglossary: add Roman case.	
longragged3col: added check for		Fixed bugs in the else statements ..	179
glsnogroupskip	284	@gls@link: added check for	
nopostdot: new	10	“nohypertypes”	109
tree: added check for glsnogroupskip	309	mcolalttree: replaced ‘2’ with	
treenoname: added check for		\glsmcols	291
glsnogroupskip	311	mcolindex: replaced ‘2’ with \glsmcols	288
super: added check for glsnogroupskip	294	mcolindexspannav: replaced ‘2’ with	
super3col: added check for		\glsmcols	289
glsnogroupskip	296	mcoltree: replaced ‘2’ with \glsmcols	289
super4col: added check for		mcoltreenoname: replaced ‘2’ with	
glsnogroupskip	297	\glsmcols	290
superragged: added check for		mcoltreesspannav: replaced ‘2’ with	
glsnogroupskip	301	\glsmcols	290
superragged3col: added check for		\gls@protected@pagefmts: added	
glsnogroupskip	303	Roman to list	177
3.04 (2012-11-11)		\gls@Romanpage: new	178
altlist: replaced \newline with		\glsgetgrouplabel: fixed bug (typo in	
paragraph break	269	\equal)	208
3.04 (2012-11-18)		\nopostdesc: made robust	35
@@do@wrglossary: changed		3.05 (2013/04/21)	
\theglsentrycounter back to		@gls@nohyperlist: new	17
@glslocref	180	\GlsDeclareNoHyperList: new	17
@@do@wrglossary: modified to		nohypertypes: new	17
compensate for possible incorrect		3.06 (2013/06/17)	
page number	179	@xdy@main@language: Changed back to	
@gls@escbsdq: unsanitize		using \language	27
\gls@numberpage, \gls@alphpage,		\findrootlanguage: Obsoleted	49
\gls@Alphpage and		3.07 (2013-07-05)	
\gls@romanpage	112	@gls@link: fixed bug that failed to find	
@print@glossary: Moved aux write to		entry in list	109
end of document to prevent		\glossarypreamble: modified to work	
unwanted whatsit occurring here. ..	187	with \setglossarypreamble	38
General: Added check for doc package	4	\gls@doclearpage: added check for	
added datatool-base as a required		openright	41
package	4	\glspostdescription: Added	
added local key	106	spacefactor code	9
\gls@Alphpage: new	178	\GlsSetXdyCodePage: Added check for	
\gls@alphpage: new	177	fontspec	50

\SetDescriptionAcronymDisplayStyle: now using \glsdoparenifnotempty	237	\ifglshasdesc: new	54
\setglossarypreamble: new	38	\ifglshassymbol: new	55
3.08a (2013-08-30)		altlongragged4col: updated to use \glossentry and \subglossentry	286
list: updated list style to use \glossentry and \subglossentry	268	alttree: updated to use \glossentry and \subglossentry	312
listdotted: updated listdotted style to use \glossentry and \subglossentry	270	index: added paragraph break at end of environment	307
altlist: updated altlist style to use \glossentry and \subglossentry	269	updated to use \glossentry and \subglossentry	307
inline: updated inline style to use \glossentry and \subglossentry	266	long: updated to use \glossentry and \subglossentry	272
3.08a (2013-09-28)		longragged: updated to use \glossentry and \subglossentry	282
\@glo@storeentry: no longer need to check for special characters in any of the fields other than sort	86	longragged3col: updated to use \glossentry and \subglossentry	284
updated for \glossentry	86	tree: updated to use \glossentry and \subglossentry	309
\@glossaryentryfield: switched to \glossentry	85	\setglossarystyle: new	208
\@glossarysubentryfield: switched to \subglossentry	85	\setglossentrycompatibility: new	205
General: added nogroupskip key to \printglossary	198	superragged: updated to use \glossentry and \subglossentry	301
removed definition of \@glossaryentryfield	356	3.09a (2013-10-09)	
removed definition of \@glossarysubentryfield	356	\@gls@assign@symbolplural@field: new	19
\compatibleglossentry: new	204	\@gls@default@value: new	62
\compatiblesubglossentry: new	205	\Glsentrydesc: made robust	148
\glossaryentryfield: deprecated	205	\Glsentrydescplural: made robust	148
\Glossentrydesc: new	204	\Glsentryfirst: made robust	149
\glossentrydesc: new	204	\Glsentryfirstplural: made robust	150
\Glossentryname: new	204	\Glsentryfull: made robust	153
\glossentryname: new	204	\Glsentryfullpl: made robust	153
\Glossentrysymbol: new	205	\Glsentrylong: made robust	152
\glossentrysymbol: new	204	\Glsentrylongpl: made robust	152
\gls@assign@desc@field: new	19	\Glsentryname: made robust	147
\gls@assign@descplural@field: new	19	\Glsentryplural: made robust	149
\gls@assign@field: new	68	\Glsentryshort: made robust	152
\gls@ifnotmeasuring: new	87	\Glsentryshortpl: made robust	152
\glsaddallunused: new	156	\Glsentrysymbol: made robust	149
\glsexpandfields: new	69	\Glsentrysymbolplural: made robust	149
\glsnoexpandfields: new	69	\Glsentrytext: made robust	148
\glssee: made robust	182	\Glsentryuseri: made robust	150
\glsseeformat: made robust	182	\Glsentryuserii: made robust	151
\glsseeitem: made robust	183	\Glsentryuseriii: made robust	151
\glsseelist: made robust	182	\Glsentryuseriv: made robust	151
\ifglsdessuppressed: new	54	\Glsentryuserv: made robust	152
		\glstextup: new	213

\ifglshassymbol: changed test to check for \@gls@default@symbol	55	\@Gls@: add \glsifplural, \glscapscase, \glscustomtext and \glsinsert	121
3.10a (2013-09-28)		change to using \glentryfmt style commands	121
\gls@assign@type@field: new	19	removed \makefirstuc (now dealt with in \glentryfmt)	122
3.10a (2013-10-13)		\@Glspl@: add \glsifplural, \glscapscase, \glscustomtext and \glsinsert	124
\@gls@keymap: new	71	change to using \glentryfmt style commands	124
\@gls@provide@newglossary: new ...	57	removed \makefirstuc (now dealt with in \glentryfmt)	124
\@gls@writedef: new	70	\@acrlong: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	355
\@glsdefaultplural: Obsolete	66	\@acrshort: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	354
\@glsnodelsc: new	66	\@gls@: add \glslabel, \glsifplural, \glscapscase, \glscustomtext and \glsinsert	121
\@print@glossary: Added providecommand code to aux file ..	187	change to using \glentryfmt style commands	121
\gls@defglossaryentry: Changed to using \@gls@default@value	79	\@gls@noexpand@fields: Fixed bug expand replaced with noexpand	67
new	79	\@glsdisp: add \glslabel, \glsifplural, \glscapscase, \glscustomtext and \glsinsert	125
\gls.writedefhook: new	78	change to using \glentryfmt style commands	125
\makeglossaries: Added providecommand code to aux file ..	169	\@Glspl@: add \glslabel, \glsifplural, \glscapscase, \glscustomtext and \glsinsert	123
\new@glossaryentry: new	69	change to using \glentryfmt style commands	123
\ns@newglossary: added \@gls@provide@newglossary	59	General: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	139–145
3.11a (2013-10-15)		changed to just use \Glsentrydescplural	131
\@ACRlong: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	356	changed to just use \glentrydescplural	131, 132
\@ACRshort: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	354	changed to just use \Glsentrydesc .	131
\@Acrlong: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	355	changed to just use \glentrydesc	130, 131
\@Acrshort: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	354		
\@GLS@: add \glslabel, \glsifplural, \glscapscase, \glscustomtext and \glsinsert	122		
change to using \glentryfmt style commands	122		
removed \MakeUppercase (now moved to \glentryfmt)	122		
\@Glspl: add \glslabel, \glsifplural, \glscapscase, \glscustomtext and \glsinsert	125		
change to using \glentryfmt style commands	125		
removed \MakeUppercase as now dealt with in \glentryfmt	125		

3.13a (2013-11-05)		long: switched to \tabularnewline ..	272
\@gls@assign@symbol@field: changed		long3col: switched to	
to use \glssetnoexpandfield	19	\tabularnewline	273
\@gls@assign@symbolplural@field:		long3colheader: switched to	
changed to use		\tabularnewline	274
\glssetnoexpandfield	19	long3colheaderborder: switched to	
\@gls@link: removed \relax	110	\tabularnewline	274
\@gls@notranslatorhook: new	23	long4col: switched to	
\@gls@setupsort@standard: moved		\tabularnewline	275
\@gls@santizesort to		long4colheader: switched to	
\glsprestandardsort	11	\tabularnewline	275
ucmark: added check for memoir	10	longheader: switched to	
see: added \gls@checkseeallowed ...	63	\tabularnewline	272
\glossarysection: changed		longheaderborder: switched to	
\glossarymark to		\tabularnewline	273
\gls glossarymark	39	\SetFootnoteAcronymDisplayStyle:	
\glossarystyle: fixed bug caused by		fixed missing argument bug	240
using \ifdef instead of \ifcsdef .	209	super: switched to \tabularnewline .	294
\gls@assign@desc@field: changed to		super3col: switched to	
use \glssetnoexpandfield	19	\tabularnewline	295
\gls@assign@descplural@field:		super3colheader: switched to	
changed to use		\tabularnewline	296
\glssetnoexpandfield	19	super4col: switched to	
\gls@assign@name@field: changed to		\tabularnewline	297
use \glssetnoexpandfield	19	super4colheader: switched to	
\gls@assign@type@field: changed to		\tabularnewline	298
use \glssetexpandfield	19	super4colheaderborder: switched to	
\gls@checkseeallowed: new	64	\tabularnewline	298
\glsaddallunused: set default to		superheader: switched to	
\@glo@types	156	\tabularnewline	295
\Glsentryfull: changed to use		superheaderborder: switched to	
\acrfullformat	153	\tabularnewline	295
\Glsentryfull: changed to use			
\acrfullformat	153	3.14a (2013-11-12)	
\Glsentryfullpl: changed to use		\@glswritefiles: renamed	
\acrfullformat	153	\glswritefiles to	
\Glsentryfullpl: changed to use		\@glswritefiles and used	
\acrfullformat	153	“savewrites” option to set	
\gls glossarymark: renamed		\glswritefiles	175
\gls glossarymark to		General: new	256
\gls glossarymark to avoid conflict		acronyms: new	15
with memoir	39	\gls@def glossaryentry: added check	
\glsprestandardsort: new	11	for existence of default glossary	80
\glssetexpandfield: new	19	set the default for firstplural to be the	
\glssetnoexpandfield: new	19	value of plural	82
altsuper4colheader: switched to		xindy gloss: new	27
\tabularnewline	299	\longprovideglossaryentry: new ...	78
altsuper4colheaderborder: switched		compatible-2.07: added check for 2.07	
to \tabularnewline	299	before setting 3.07 compatibility	28
		notranslate: new	24

\provideglossaryentry:new	69	index:new	29
4.0 (2013-11-14)		\newacronymstyle:new	219
\gls@defglossaryentry: added check		long-sc-short:new	222
for first key	82	long-sc-short-desc:new	223
super: fixed typo in \subglossentry		long-short:new	220
(\glossentrydesc)	294	long-short-desc:new	223
4.01 (2013-11-16)		long-sm-short:new	222
General: fixed non-value options so that		long-sm-short-desc:new	224
they can be passed to document class .	8	long-sp-short-desc:new	223
\CustomAcronymFields: inserted		footnote:new	227
missing comma	247	footnote-desc:new	229
4.02 (2013-12-05)		footnote-sc:new	228
\@acrfull: now using \acrfullfmt ..	214	footnote-sc-desc:new	229
\@gls@indexdef:new	29	footnote-sm:new	229
\@gls@numbersdef:new	29	footnote-sm-desc:new	230
\@gls@symbolsdef:new	29	\setacronymstyle:new	219
General: Removed \acronymfont .	142–146	\SetDescriptionAcronymDisplayStyle:	
\ACRfullfmt:new	215	Moved check for empty custom text to	
\Acrfullfmt:new	215	prevent unwanted parenthetical	
\acrfullfmt:new	214	material	237
\ACRfullplfmt:new	216	\SetDescriptionFootnoteAcronymDisplayStyle:	
\Acrfullplfmt:new	216	Moved check for empty custom text to	
\acrfullplfmt:new	216	prevent unwanted parenthetical	
\acronymentry:new	218	material	233
sanitize: fixed bug that caused an error		\SetFootnoteAcronymDisplayStyle:	
here	23	Moved check for empty custom text to	
sc-short-long:new	222	prevent unwanted parenthetical	
sc-short-long-desc:new	224	material	240
\Genacrfullformat:new	104	\SetGenericNewAcronym:new	217
\genacrfullformat:new	104	\SetSmallAcronymDisplayStyle:	
\GenericAcronymFields:new	218	Moved check for empty custom text to	
\Genplacrfullformat:new	104	prevent unwanted parenthetical	
\genplacrfullformat:new	104	material	242
\Glsentryfull: bug fix: added missing		dua:new	225
\acronymfont	153	dua-desc:new	227
\glsentryfull: bug fix: added missing		numberedsection: added nameref	
\acronymfont	153	option	7
\Glsentryfullpl: bug fix: added		4.02 (2013-13-05)	
missing \acronymfont	153	\makeglossaries: made preamble only	170
\glsentryfullpl: bug fix: added		4.03 (2014-01-17)	
missing \acronymfont	153	General: changed default to \@empty	
\glsgenacfmt:new	102	instead of \relax	28
\GlsUseAcrEntryDisplayStyle:new ...	220	4.03 (2014-01-20)	
\GlsUseAcrStyleDefs:new	220	\@do@wrglossary: added	
short-long:new	221	\glsdetoklabel	180
short-long-desc:new	224	\@ACRlong: removed \glslabel	
xindynoglsnumbers:new	27	(defined in \@gls@link)	356
sm-short-long:new	223	\@ACRshort: removed \glslabel	
sm-short-long-desc:new	225	(defined in \@gls@link)	354

\@Acrlong: removed \glslabel (defined in \@gls@link)	355	\genplacrfullformat: redefined to use accessibility information	353
\@Acrshort: removed \glslabel (defined in \@gls@link)	354	\glossentryname: added \glsdetoklabel	204
\@GLS@: removed \glslabel (defined in \@gls@link)	122	\gls@defglossaryentry: added \glsdetoklabel	79
\@GLSpl: removed \glslabel (defined in \@gls@link)	125	replaced #1 with \@glo@label	80
\@Gls@: removed \glslabel (defined in \@gls@link)	121	replaced \ifthenelse with \ifdefequal	81
\@Gls@entry@field: new	146	\glsadd: added \glsdetoklabel	155
\@Glspl@: removed \glslabel (defined in \@gls@link)	124	\glsaddkey: switched to using \@gls@field@link	73
\@acrlong: removed \glslabel (defined in \@gls@link)	355	\glsdetoklabel: new	52
\@acrshort: removed \glslabel (defined in \@gls@link)	354	\glsdisplaynumberlist: added \glsdetoklabel	154
\@gls@: removed \glslabel (defined in \@gls@link)	121	\glsdoifexistsorwarn: new	53
\@gls@access@display: new	342	\glsentryaccess: switched to using \@gls@entry@field	340
\@gls@entry@field: new	146	\glsentrydescaccess: switched to using \@gls@entry@field	341
\@gls@fetchfield: new	71	\glsentrydescpluralaccess: switched to using \@gls@entry@field	341
\@gls@field@link: new	126	\glsentryfirstaccess: switched to using \@gls@entry@field	341
\@gls@link: added \glsdetoklabel .	109	\glsentryfirstplural: added \glsdetoklabel	150
moved \@gls@link@opts and \@gls@link@label to \@gls@link	109	\glsentrylongaccess: switched to using \@gls@entry@field	342
\@gls@writedef: added \glsdetoklabel	70	\glsentrylongpluralaccess: switched to using \@gls@entry@field	342
\@glsdisp: removed \glslabel (defined in \@gls@link)	125	\glsentrypluralaccess: switched to using \@gls@entry@field	341
\@glspl@: removed \glslabel (defined in \@gls@link)	123	\glsentryshortaccess: switched to using \@gls@entry@field	341
\@printglossary: added \glsdetoklabel	186	\glsentryshortpluralaccess: switched to using \@gls@entry@field	341
General: removed \glslabel (defined in \@gls@link)	139	\glsentrysymbolaccess: switched to using \@gls@entry@field	341
sc-short-long-desc: redefined to use accessibility information	360	\glsentrysymbolpluralaccess: switched to using \@gls@entry@field	341
\compatibleglossentry: added \glsdetoklabel	336	\glsentrytextaccess: switched to using \@gls@entry@field	341
\compatiblesubglossentry: added \glsdetoklabel	337	\glsgenacfmt: redefined to use accessibility information	351
\Genacrfullformat: redefined to use accessibility information	353	\glsgenentryfmt: redefined to use accessibility information	348
\genacrfullformat: redefined to use accessibility information	353		
\Genplacrfullformat: redefined to use accessibility information	353		

<code>\glshyperlink</code> : added		
<code>\glsdetoklabel</code>	154	
<code>\glslocalreset</code> : added		
<code>\glsdetoklabel</code>	88	
<code>\glslocalunset</code> : added		
<code>\glsdetoklabel</code>	88	
<code>\glsmoveentry</code> : added		
<code>\glsdetoklabel</code>	85	
replaced <code>\ifthenelse</code> with		
<code>\ifdefequal</code>	85	
<code>\glsrefentry</code> : added <code>\glsdetoklabel</code>	202	
<code>\glsreset</code> : added <code>\glsdetoklabel</code> ...	87	
<code>\glsseelist</code> : added <code>\expandafter</code>		
commands	183	
<code>\glsstepentry</code> : added		
<code>\glsdetoklabel</code>	202	
<code>\glsstepsubentry</code> : added		
<code>\glsdetoklabel</code>	202	
<code>\glsunset</code> : added <code>\glsdetoklabel</code> ...	88	
short-long: commented spurious EOL	222	
redefined to use accessibility		
information	358	
short-long-desc: redefined to use		
accessibility information	360	
<code>\ifglshdescsuppressed</code> : added		
<code>\glsdetoklabel</code>	54	
fixed typo	54	
<code>\ifglshentryexists</code> : added		
<code>\glsdetoklabel</code>	52	
<code>\ifglshaschildren</code> : added		
<code>\glsdetoklabel</code>	54	
<code>\ifglshasdesc</code> : added		
<code>\glsdetoklabel</code>	54	
<code>\ifglshasfield</code> : new	55	
<code>\ifglshaslong</code> : added		
<code>\glsdetoklabel</code>	55	
<code>\ifglshasparent</code> : added		
<code>\glsdetoklabel</code>	54	
<code>\ifglshasshort</code> : added		
<code>\glsdetoklabel</code>	55	
<code>\ifglshassymbol</code> : added		
<code>\glsdetoklabel</code>	55	
replaced <code>\ifcempty</code> with		
<code>\ifdefempty</code> and replaced <code>\ifx</code> with		
<code>\ifdefequal</code>	55	
<code>\ifglshused</code> : added <code>\glsdetoklabel</code> ..	52	
sm-short-long-desc: redefined to use		
accessibility information	360	
long-sc-short-desc: redefined to use		
accessibility information	359	
long-short: redefined to use		
accessibility information	357	
long-short-desc: redefined to use		
accessibility information	359	
long-sm-short-desc: redefined to use		
accessibility information	359	
footnote: redefined to use accessibility		
information	363	
footnote-desc: redefined to use		
accessibility information	365	
footnote-sc: redefined to use		
accessibility information	365	
footnote-sc-desc: redefined to use		
accessibility information	366	
footnote-sm: redefined to use		
accessibility information	365	
footnote-sm-desc: redefined to use		
accessibility information	366	
<code>\renewacronymstyle</code> : new	219	
<code>\showglocounter</code> : added		
<code>\glsdetoklabel</code>	250	
<code>\showglodesc</code> : added <code>\glsdetoklabel</code>	251	
<code>\showglodescaccess</code> : added		
<code>\glsdetoklabel</code>	372	
<code>\showglodescplural</code> : added		
<code>\glsdetoklabel</code>	252	
<code>\showglodescpluralaccess</code> : added		
<code>\glsdetoklabel</code>	372	
<code>\showglofirst</code> : added		
<code>\glsdetoklabel</code>	250	
<code>\showglofirstaccess</code> : added		
<code>\glsdetoklabel</code>	372	
<code>\showglofirstpl</code> : added		
<code>\glsdetoklabel</code>	250	
<code>\showglofirstpluralaccess</code> : added		
<code>\glsdetoklabel</code>	372	
<code>\showgloflag</code> : added <code>\glsdetoklabel</code>	253	
<code>\showgloindex</code> : added		
<code>\glsdetoklabel</code>	253	
<code>\showglolevel</code> : added		
<code>\glsdetoklabel</code>	249	
<code>\showglolong</code> : added <code>\glsdetoklabel</code>	252	
<code>\showglolongaccess</code> : added		
<code>\glsdetoklabel</code>	373	
<code>\showglolongpluralaccess</code> : added		
<code>\glsdetoklabel</code>	373	
<code>\showglongame</code> : added <code>\glsdetoklabel</code>	251	

\showglongnameaccess: added	4.04 (2014-03-04)	
\glsdetoklabel	372	\@gls@getcounterprefix: added warning if no prefix can be formed . 181
\showgloparent: added		4.04 (2014-03-06)
\glsdetoklabel	249	\@@gls@noidx@nosanitizesort: new . 20
\showgloplural: added		\@@gls@noidx@sanitizesort: new ... 20
\glsdetoklabel	249	\@@gls@nosanitizesort: new 20
\showglopluralaccess: added		\@@gls@sanitizesort: new 20
\glsdetoklabel	372	\@glo@addchildren: new 188
\showgloshort: added		\@glo@do@sortentries: new 189
\glsdetoklabel	252	\@glo@grabfirst: new 194
\showgloshortaccess: added		\@glo@sortedinsert: new 190
\glsdetoklabel	373	\@glo@sortentries: new 188
\showgloshortpluralaccess: added		\@glo@sorthandler@case: new 190
\glsdetoklabel	373	\@glo@sorthandler@letter: new ... 190
\showglosort: added \glsdetoklabel	252	\@glo@sorthandler@nocase: new ... 190
\showglosymbol: added		\@glo@sorthandler@word: new 190
\glsdetoklabel	252	\@glo@sortmacro@case: new 192
\showglosymbolaccess: added		\@glo@sortmacro@def: new 192
\glsdetoklabel	372	\@glo@sortmacro@def@do: new 193
\showglosymbolplural: added		\@glo@sortmacro@letter: new 191
\glsdetoklabel	252	\@glo@sortmacro@nocase: new 192
\showglosymbolpluralaccess: added		\@glo@sortmacro@standard: new ... 191
\glsdetoklabel	372	\@glo@sortmacro@use: new 193
\showglotext: added \glsdetoklabel	249	\@glo@sortmacro@word: new 191
\showglotextaccess: added		\@gls@noidx@do: new 195
\glsdetoklabel	372	\@gls@noidx@getgrouptitle: new .. 207
\showglotype: added \glsdetoklabel	250	\@gls@noref@warn: new 174
\showglouser: added		\@gls@reference: new 197
\glsdetoklabel	250	\@gls@warnonglossdefined: new 18
\showglouserii: added		\@gls@warnontheGLOSSdefined: new . 18
\glsdetoklabel	250	\@no@makeglossaries: new 174
\showglouseriii: added		\@print@glossary: new 186
\glsdetoklabel	251	\@print@noidx@glossary: new 193
\showglouseriv: added		\@printgloss@setsort: new 184
\glsdetoklabel	251	\@printglossary: new 185
\showglouserv: added		General: added sort key to printgloss
\glsdetoklabel	251	group 200
\showglouservi: added		\compatibleglossentry: changed
\glsdetoklabel	251	\newcommand to \def as is may or may not be defined 336
dua: fixed bug in \acrfullfmt	226	\compatiblesubglossentry: changed
fixed bug in \Acrrfullplfmt	226	\newcommand to \def as is may or may not be defined 337
fixed bug in \acrfullplfmt	226	\defglsdisplayfirst: fixed unwanted space 105
redefined to use accessibility information	361	\glo@grabfirst: new 194
dua-desc: commented spurious EOL ..	227	\gls@defglossaryentry: replaced \ifx with \ifdefvoid 84
redefined to use accessibility information	363	

\glsnoidxdisplayloc: new	197	\@ACRshort: added	
\glsnoidxdisplayloclisthandler:		\do@gl@link@checkfirsthyper	354
new	196	\@Acrlong: added	
\glsnoidxloclist: new	196	\do@gl@link@checkfirsthyper	355
\glsnoidxloclisthandler: new	196	\@Acrshort: added	
\glsnoidxstripaccents: new	20	\do@gl@link@checkfirsthyper	354
alttree: moved hangindent and		\@GLS@: moved \glsifhyper	122
parindent assignments outside level		moved check for first use to	
test	312	\@gl@link	122
\makeglossaries: Moved definition of		\@GLSpl: moved \glsifhyper	125
\glswrite to \makeglossaries ..	168	moved check for first use to	
\makenoidxglossaries: new	171	\@gl@link	125
\printglossary: changed to use new		\@Gls@: moved \glsifhyper	121
\@printglossary	184	moved check for first use to	
\printnoidxglossaries: new	184	\@gl@link	121
\printnoidxglossary: new	184	\@Glspl@: moved \glsifhyper	124
\showgloclist: new	253	moved check for first use to	
\warn@noprintglossary: Activate		\@gl@link	124
warning in \makeglossaries	183	\@acrlong: added	
\writeist: checked for definition of		\do@gl@link@checkfirsthyper	355
\glswrite	157, 162	\@acrshort: added	
4.06 (2014-03-12)		\do@gl@link@checkfirsthyper	353
\@GLS@: added \glsifhyper	122	\@closegl@: new	167
\@GLSpl: added \glsifhyper	125	\@gl@: moved \glsifhyper	121
\@Gls@: added \glsifhyper	121	moved check for first use to	
\@Glspl@: added \glsifhyper	124	\@gl@link	121
\@gl@: added \glsifhyper	121	\@gl@automake: new	167
\@gl@numbersdef: added hook to set		\@gl@doautomake: new	28
toc title	29	\@gl@field@link: added assignment	
\@gl@symbolsdef: added hook to set		of	
toc title	29	\do@gl@link@checkfirsthyper	126
\@gl@disp: added \glsifhyper	125	\@gl@forbidtexext: new	58
\@gl@spl@: added \glsifhyper	123	\@gl@hyp@opt: new	107
General: added \glsifhyper	139–146	\@gl@link: removed redundancy	109
acronym: added hook to set toc title	15	renamed \gl@type to \glstype ...	109
acronyms: added hook to set toc title ...	15	\@gl@link@checkfirsthyper: new .	108
\gl@defmain: added hook to set toc title	14	\@gl@disp: moved \glsifhyper	125
4.07 (2014-04-04)		moved check for first use to	
\@glossarysection: added optional		\@gl@link	125
argument when using unstarred		\@gl@spl@: moved \glsifhyper	123
version	40	moved check for first use to	
\@gl@noidx@do: added \global in case		\@gl@link	123
it's used in a tabular-like style	195	\@ignored@glossaries: new	61
\Acrfullplfmt: fixed no case change		General: added entrycounter option to	
bug	216	printgloss family	198
\gl@letentryfield: new	146	added nopostdot option to	
4.08 (2014-07-30)		printgloss family	198
\@ACRlong: added		added subentrycounter option to	
\do@gl@link@checkfirsthyper	355	printgloss family	199

explicitly initialise hyper key	106	removed \@sGLSuseriii	135
moved \glsifhyper	139–146	removed \@sGlsuseriii	135
removed \@sACRlongpl	145	removed \@sglsuseriii	134
removed \@sAcrlongpl	145	removed \@sGLSuseriv	137
removed \@sacrlongpl	144	removed \@sGlsuseriv	136
removed \@sACRlong	143	removed \@sglsuseriv	136
removed \@sAcrlong	143	removed \@sGLSuseri	134
removed \@sacrlong	142	removed \@sGlsuseri	134
removed \@sACRshortpl	141	removed \@sglsuseri	134
removed \@sAcrshortpl	141	removed \@sGLSservi	138
removed \@sacrshortpl	140	removed \@sGlsservi	138
removed \@sACRshort	140	removed \@sglsuseri	138
removed \@sAcrshort	139	removed \@sGLSserv	137
removed \@sacrshort	138	removed \@sGlsuser	137
removed \@sgls@link	108	removed \@sglsuser	137
removed \@sGLSdescplural	132	removed \@sGLS	122
removed \@sGlsdescplural	131	removed \@sGls	121
removed \@sglsdescplural	131	removed \@sgls	120
removed \@sGLSdesc	131	removed \@thirdofthree (defined in	
removed \@sGlsdesc	131	kernel)	120
removed \@sglsdesc	130	removed sPGLS	261
removed \@sglsdisp	125	removed sPgl	259
removed \@sGLSfirstplural	129	removed spgls	258
removed \@sGlsfirstplural	129	removed sPGLSpl	261
removed \@sglsfirstplural	129	removed sPglspl	260
removed \@sGLSfirst	128	removed spglspl	259
removed \@sGlsfirst	127	\ACRfull: removed \s@ACRfull	215
removed \@sglsfirst	127	switched to using \@gls@hyp@opt ..	215
removed \@sGLSname	130	\Acrfull: removed \s@Acrfull	215
removed \@sGlsname	130	switched to using \@gls@hyp@opt ..	215
removed \@sglsname	129	\acrfull: removed \s@acrfull	214
removed \@sGLSplural	128	switched to using \@gls@hyp@opt ..	214
removed \@sGlsplural	128	\ACRfullpl: removed \s@ACRfullpl ..	216
removed \@sglsplural	128	switched to using \@gls@hyp@opt ..	216
removed \@sGLSpl	124	\Acrfullpl: removed \s@Acrfullpl ..	216
removed \@sGlspl	124	switched to using \@gls@hyp@opt ..	216
removed \@sglspl	123	\acrfullpl: removed \s@acrfullpl ..	215
removed \@sGLSsymbolplural	133	switched to using \@gls@hyp@opt ..	215
removed \@sGlsymbolplural	133	\ACRlong: switched to using	
removed \@sglsymbolplural	133	\@gls@hyp@opt	143
removed \@sGLSsymbol	132	\Acrlong: switched to using	
removed \@sGlsymbol	132	\@gls@hyp@opt	142
removed \@sglsymbol	132	\acrlong: switched to using	
removed \@sGLStext	126	\@gls@hyp@opt	142
removed \@sGlstext	127	\ACRlongpl: switched to using	
removed \@sglstext	126	\@gls@hyp@opt	145
removed \@sGLSuseriii	136	\Acrlongpl: switched to using	
removed \@sGlsuseriii	135	\@gls@hyp@opt	144
removed \@sglsuseriii	135		

\acrlongpl: switched to using \@gls@hyp@opt	144	\GLSfirst: switched to using \@gls@hyp@opt	127
\ACRshort: switched to using \@gls@hyp@opt	139	\Glsfirst: switched to using \@gls@hyp@opt	127
\Acrshort: switched to using \@gls@hyp@opt	139	\glsfirst: switched to using \@gls@hyp@opt	127
\acrshort: switched to using \@gls@hyp@opt	138	\GLSfirstplural: switched to using \@gls@hyp@opt	129
\ACRshortpl: switched to using \@gls@hyp@opt	141	\Glsfirstplural: switched to using \@gls@hyp@opt	129
\Acrshortpl: switched to using \@gls@hyp@opt	141	\glsfirstplural: switched to using \@gls@hyp@opt	129
\acrshortpl: switched to using \@gls@hyp@opt	140	\glsifhyper: deprecated	107
\forallacronyms: new	51	\glslink: switched to using \@gls@hyp@opt	108
\GLS: switched to using \@gls@hyp@opt	122	\glslinkcheckfirsthyperhook: new	109
\Gls: switched to using \@gls@hyp@opt	121	\glslinkvar: new	107
\gls: switched to using \@gls@hyp@opt	120	\GLSname: switched to using \@gls@hyp@opt	130
\gls@defglossaryentry: added check for ignored glossary	80	\Glsname: switched to using \@gls@hyp@opt	130
\gls@istfilebase: new	36	\glsname: switched to using \@gls@hyp@opt	129
\glsaddkey: removed \@sGLS@user@<key>	74	\GLSpl: switched to using \@gls@hyp@opt	124
removed \@sGls@user@<key>	74	\Glspl: switched to using \@gls@hyp@opt	123
removed \@sgls@user@<key>	74	\glspl: switched to using \@gls@hyp@opt	123
switched to using \@gls@hyp@opt ...	74	\GLSplural: switched to using \@gls@hyp@opt	128
\GLSdesc: switched to using \@gls@hyp@opt	131	\Glsplural: switched to using \@gls@hyp@opt	128
\Glsdesc: switched to using \@gls@hyp@opt	130	\glsplural: switched to using \@gls@hyp@opt	128
\glsdesc: switched to using \@gls@hyp@opt	130	\glsspace: new	214
\GLSdescplural: switched to using \@gls@hyp@opt	132	\GLSsymbol: switched to using \@gls@hyp@opt	132
\Glsdescplural: switched to using \@gls@hyp@opt	131	\Glsymbol: switched to using \@gls@hyp@opt	132
\glsdescplural: switched to using \@gls@hyp@opt	131	\glsymbol: switched to using \@gls@hyp@opt	132
\glsdisablehyper: added \KV@glslink@hyperfalse to definition	120	\GLSsymbolplural: switched to using \@gls@hyp@opt	133
\glsdisp: switched to using \@gls@hyp@opt	125	\Glsymbolplural: switched to using \@gls@hyp@opt	133
\glsdohyperlink: new	119	\glssymbolplural: switched to using \@gls@hyp@opt	133
\glsdohypertarget: new	119		
\glsenablehyper: added \KV@glslink@hypertrue to definition	120		

\GLStext: switched to using \@gls@hyp@opt	126	\ns@newglossary: added \@gls@hyp@opt: new	59
\Glstext: switched to using \@gls@hyp@opt	127	\p@gls@hyp@opt: new	107
\glstext: switched to using \@gls@hyp@opt	126	\PGLS: changed to use \@gls@hyp@opt	261
\glstreenamefmt: new	306	\PglS: changed to use \@gls@hyp@opt	259
\GLSuseri: switched to using \@gls@hyp@opt	134	\pglS: changed to use \@gls@hyp@opt	258
\Glsuseri: switched to using \@gls@hyp@opt	134	\PGLSpl: changed to use \@gls@hyp@opt	261
\glSuseri: switched to using \@gls@hyp@opt	134	\PglSpl: changed to use \@gls@hyp@opt	260
\GLSuserii: switched to using \@gls@hyp@opt	135	\pglSpl: changed to use \@gls@hyp@opt	259
\Glsuserii: switched to using \@gls@hyp@opt	135	\s@gls@hyp@opt: new	107
\glSuserii: switched to using \@gls@hyp@opt	135	\s@newglossary: new	58
\glSuserii: switched to using \@gls@hyp@opt	134	automake: new	27
\GLSuseriii: switched to using \@gls@hyp@opt	136	4.09 (2014-08-12)	
\glSuseriii: switched to using \@gls@hyp@opt	135	\glsaddkey: fixed bug in user commands	74
\GLSuseriv: switched to using \@gls@hyp@opt	136	4.10 (2014-08-27)	
\glSuseriv: switched to using \@gls@hyp@opt	136	\@Gls@acrentryname: new	147
\GLSuserv: switched to using \@gls@hyp@opt	137	\@Gls@entryname: new	147
\glSuserv: switched to using \@gls@hyp@opt	137	\@gls@glossary: Renamed \@glossary to \@gls@glossary	176
\ifignoredglossary: new	61	\glSpercentchar: new	157
altlongragged4col: fixed bug that displayed description instead of symbol	286	\glStildechar: new	157
\newglossary: added starred version ..	58	alttree: moved space after symbol	313, 314
\newignoredglossary: new	60	4.11 (2014-09-01)	
		\@do@wrglossary: added hook	179
		sanitize: none option	23
		\gls@wrglossary: renamed from \@wrglossary to \gls@wrglossary	176
		\glsaddprotectedpagefmt: new	178
		\glsbackslash: new	156
		4.12 (2014-11-22)	
		\@gls@addpredefinedattributes: Added glsignore attribute	45
		\@gls@adjustmode: new	155
		\@gls@notranslatorhook: removed ...	23
		\@gls@toc: added \protect to \numberline	41
		\@gls@usetranslator: new	23
		\glsacrpluralsuffix: new	32
		\glsadd: added check for vertical mode	155
		\glsaddallunused: replaced @gobble with glsignore	156
		\glsifusedtranslatordict: new	24
		\glsignore: new	156
		\glsupacrpluralsuffix: new	33
		\ProvidesGlossariesLang: new	33

\RequireGlossariesLang: new	33	4.16 (2015-07-08)	
4.13 (2015-02-03)		\@ACRlong: added \glspostlinkhook	356
\indexspace: new	268, 287, 306	\@ACRshort: added \glspostlinkhook	355
4.14 (2015-02-28)		\@Acrlong: added \glspostlinkhook	355
\@glslocalreset: new	89	\@Acrshort: added \glspostlinkhook	354
\@glslocalunset: new	89	\@GLS@: added \glspostlinkhook	123
\@glsreset: new	89	\@GLSpl: added \glspostlinkhook	125
\@glsunset: new	89	\@Gls@: added \glspostlinkhook	122
\@newglossaryentry@defcounters:		\@GLspl@: added \glspostlinkhook	124
new	90	\@acrlong: added \glspostlinkhook	355
\cGls: new	93	\@acrshort: added \glspostlinkhook	354
\cGls@: new	94	\@gls@: added \glspostlinkhook	121
\cGlspl@: new	95	\@gls@link: added	
\cgls: new	93	\glspostlinkhook	108
\cgls@: new	93	\@gls@field@link: added	
\cglspl: new	94	\glspostlinkhook	126
\cglspl@: new	94	\@gls@link: moved definition of	
\@gls@entry@count: new	93	\glsifhyperon outside of this	
\@gls@increment@currcount: new	92	macro	110
\@gls@local@increment@currcount:		\@glsdisp: added \glspostlinkhook	126
new	92	\@glspl@: added \glspostlinkhook	123
\@gls@write@entrycounts: new	93	General: added \glspostlinkhook	139–146
\@glslocalreset: new	89	\glsacspace: new	221
\@glslocalunset: new	88	\glsadd: changed \@do@wrglossary to	
\@glsreset: new	89	\@do@wrglossary	155
\@glsunset: new	89	\glsfielddef: new	76
\@newglossaryentry@defcounters:		\glsfieldedef: new	75
new	85	\glsfieldfetch: new	76
\cGls: new	93	\glsfieldgdef: new	76
\cgls: new	93	\glsfieldxdef: new	75
\cGlsformat: new	94	\glsifhyperon: moved definition of	
\cglsformat: new	93	\glsifhyperon	109
\cGlspl: new	94	\glslinkpostsetkeys: new	109
\cglspl: new	94	\glspostlinkhook: new	108
\cGlsplformat: new	95	\glswriteentry: new	177
\cglsplformat: new	94	\ifglsfieldcseq: new	78
\@gls@defdocnewglossaryentry: new	69	\ifglsfielddefeq: new	77
\@glsenableentrycount: new	90	\ifglsfieldefeq: new	77
\glslocalreset: switched to		long-sp-short: new	220
\@glslocalreset	88	\showglofield: new	253
\glslocalunset: switched to		4.18 (2015-09-09)	
\@glslocalunset	88	General: split mfirstuc into separate	
\glsreset: switched to \@glsreset	87	bundle	4
\glsunset: switched to \@glsunset	88	4.19 (2015-10-31)	
4.15 (2015-03-16)		\glstreenamibox: new	312
General: bug fix replaced \@glo@type		4.19 (2015-11-22)	
with \glstype	145	\@gls@link@nocheckfirsthyper: new	126
4.16 (2015-06-18)		\@gls@preglossaryhook: new	184
\glsaddstoragekey: new	72		

\@printglossary: added		\glslistgroupheaderfmt: new	268
\@gls@preglossaryhook	186	\glslistnavigationitem: new	268
\do@glsglisablehyperinlist: new	109	\glstreegroupheaderfmt: new	306
\doifglossarynoexistsordo: new	53	\glstreenavigationfmt: new	306
\gls@gobbleopt: new	58	\ifglswrallowprimitivemods: new	178
\glsdoifexistsordo: new	53	list: fixed missing space before	
4.20 (2015-11-30)		description	268
\@gls@link: added		long: fixed typo in \glossentrydesc	272
\@gls@setdefault@glslink@opts	109	super4col: fixed bug in \glossentry	297
added \glsdonohyperlink when		4.23 (2016-04-30)	
hyperlink is suppressed	110	\glscurrentfieldvalue: new	57
\@gls@setdefault@glslink@opts:		\ifglshasfield: added	
new	109	\glscurrentfieldvalue	56, 57
\gls@checkseeallowed@preambleonly:		altlongragged4col: check for	
new	64	nogroupskip changed	286
\glsdonohyperlink: new	119	altsuperragged4col: check for	
4.21 (2016-01-24)		nogroupskip changed	304
\@printglossary: warn if no style has		long: check for nogroupskip changed	272
been set	185	long-booktabs: check for nogroupskip	
General: changed checkfirsthyper		changed	278
assignment	139–145	long3col: check for nogroupskip	
\glossarystyle: set default style if not		changed	273
already set	209	long3col-booktabs: check for	
\glsLTpenaltycheck: new	280	nogroupskip changed	278
\glspatchLToutput: new	281	long4col: check for nogroupskip	
\glspenaltygroupskip: new	281	changed	275
altlong4col-booktabs: new	279	long4col-booktabs: check for	
altlongragged4col-booktabs: new	280	nogroupskip changed	279
long-booktabs: new	277	longragged: check for nogroupskip	
long3col-booktabs: new	278	changed	283
long4col-booktabs: new	278	longragged3col: check for nogroupskip	
longragged-booktabs: new	279	changed	284
longragged3col-booktabs: new	280	super: check for nogroupskip changed	294
\setglossarystyle: set default style if		super3col: check for nogroupskip	
not already set	208	changed	296
4.22 (2016-04-19)		super4col: check for nogroupskip	
\@do@wrglossary: added check for		changed	297
\@arabic	179	superragged: check for nogroupskip	
added test to allow temporary primitive		changed	301
modifications and added arabic case	179	superragged3col: check for	
mcolalttreespannav: new	292	nogroupskip changed	303
mcolindexspannav: new	288	4.24 (2016-05-27)	
mcoltreenonamespannav: new	291	\@gls@extramakeindexopts: new	166
mcoltreespannav: new	290	\@gls@glossary: added check for debug	
\gls@arabicpage: new	178	mode	176
\gls@protected@pagefmts: added		\@gls@see@noindex: new	5
arabic to list	177	debug: new	5
\glsentrytitlecase: new	150	seenoindex: new	6
\glsfindwidesttoplevelname: new	311	\glsnomakeindexwarning: new	42

\GlsSetQuote: new	163	General: added check for	
\GlsSetWriteIstHook: new	163	\@glstr@doaccsupp	336
4.25 (2016-06-09)		\glshyperlinkname: new	263
\@gls@enablesavenonumberlist: new	65	4.30 (2017-06-11)	
\@gls@initnonumberlist: new	64	\@glo@autosee: new	84
\@gls@savenonumberlist: new	64	\@glo@autoseehook: new	84
4.26 (2016-10-12)		\@glo@check@sortallowed: new	11
\@glossary@default@style: added		\@gls@noidx@do: letter group	
check for classicthesis	7	assignment made global	195
mcindex: replaced \@idxitem with		\@gls@setupsort@def: added check for	
\glstreeitem	288	register	12
mcindexspannav: replaced \@idxitem		\@gls@setupsort@none: new	13
with \glstreeitem	289	\@xdycrossrefhook: new	47
\glstreechildpredesc: new	307	\@xdylocationclassorder: bug fix:	
\glstreeitem: new	306	changed \edef to \def	48
\glstreepredesc: new	306	\glosortentrieswarning: new	18
\glstreesubitem: new	306	\gls@set@xr@key: new	63
\glstreesubsubitem: new	306	\gls@xr@key: new	64
4.28 (2017-01-07)		\GlsAddXdyLocation: bug fix: changed	
\glspatchtabularx: new	87	#1 to #2	47
4.29 (2017-01-19)		\glsnoidxstripaccents: added \a ...	21
\@gls@noidx@do: current letter group		added \TH, \dh and \DH	21
assignment made global	195	4.31 (2017-08-10)	
\@print@noidx@glossary: moved		nolist: added check for “list” style	9
definition of		4.31 (2017-09-10)	
\@gls@currentlettergroup outside		style: changed \renewcommand to \def .	7
of theglossary environment	193		

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	
\!	115, 116
\"	21, 113–116, 118
\#	160
\%	157, 162, 320, 321
\&	33, 154
\'	21
\.	9, 21
\=	21
\?	113–115, 165
\@@delimN	211
\@@do@wrglossary	171, 180
\@@do@wrglossary	155, 177
\@@glo@assign@sortkey	172
\@@glo@list	51
\@@glo@sort	20
\@@glo@type	184
\@@glossarysec	6, 40, 41
\@@glossaryseclabel	7, 40, 198
\@@glossarysecstar	7, 40, 198
\@@gls@checkactual	117
\@@gls@checkbar	116
\@@gls@checkescactual	115
\@@gls@checkescbar	115
\@@gls@checkesclevel	116
\@@gls@checkescquote	114, 165, 166
\@@gls@checklevel	117
\@@gls@checkquote	113, 114, 164
\@@gls@default@entryfmt	96, 105
\@@gls@expand@field	19, 68, 72, 73, 232, 234–236, 239, 241, 244–246, 367–371
\@@gls@extramakeindexopts	163, 169
\@@gls@fixbraces	182
\@@gls@noexpand@field	19, 67
\@@gls@noidx@no@sanitizesort	20
\@@gls@noidx@nosanitizesort	173
\@@gls@nosanitizesort	20, 173
\@@gls@sanitizesort	20, 173
\@@gls@xdycheckbackslash	118, 119
\@@gls@xdycheckquote	118
\@@gls@localreset	89, 91
\@@gls@localunset	88, 91
\@@gls@reset	89, 91
\@@gls@unset	89, 91
\@@newglossaryentry@defcounters	90
\@@this@glo@	51
\@ACRfull	215
\@ACRfullpl	216
\@ACRlong	143, 215
\@ACRlongpl	145, 216
\@ACRshort	140, 215
\@ACRshortpl	141, 216
\@Acrfull	215
\@Acrfullpl	216
\@Acrlong	143, 215
\@Acrlongpl	145, 216
\@Acrshort	139
\@Acrshortpl	141
\@Alph	178, 179
\@GLS	122
\@GLS@	122, 261
\@GLSdesc	131
\@GLSdesc@	131
\@GLSdescplural	132
\@GLSdescplural@	132
\@GLSfirst	127, 128
\@GLSfirst@	128
\@GLSfirstplural	129
\@GLSfirstplural@	129
\@GLSname	130
\@GLSname@	130
\@GLSpl	124
\@GLSpl@	124, 262
\@GLSplural	128

\@GLSplural@	128	\@Glsuseriii@	135, 136
\@GLSsymbol	132	\@Glsuseriv	136
\@GLSsymbol@	132, 133	\@Glsuseriv@	136
\@GLSsymbolplural	133	\@Glsuserv	137
\@GLSsymbolplural@	133	\@Glsuserv@	137
\@GLStext	126	\@Glsuservi	138
\@GLStext@	126, 127	\@Glsuservi@	138
\@GLSuseri	134	\@Mi	281
\@GLSuseri@	134	\@PGLS	261
\@GLSuserii	135	\@PGLS@	261
\@GLSuserii@	135	\@PGLSpl	261
\@GLSuseriii	136	\@PGLSpl@	261
\@GLSuseriii@	136	\@Pgls	259
\@GLSuseriv	136, 137	\@Pgls@	259
\@GLSuseriv@	137	\@Pglspl	260
\@GLSuserv	137	\@Pglspl@	260
\@GLSuserv@	137	\@Roman	178, 179
\@GLSuservi	138	\@acrfull	214
\@GLSuservi@	138	\@acrfullpl	215, 216
\@Gls	121	\@acrlong	142, 214
\@Gls@	92, 94, 121, 260	\@acrlongpl	144, 216
\@Gls@acrentryname	217	\@acrshort	138, 214, 215
\@Gls@entry@field	73, 147–152	\@acrshortpl	140, 216
\@Gls@entryname	147, 217	\@addtoacronymlists	15, 16
\@Glsdesc	130, 131	\@after	16
\@Glsdesc@	131	\@afterheading	269, 324
\@Glsdescplural	131	\@alph	177, 179
\@Glsdescplural@	131	\@arabic	178, 179
\@Glsfirst	127	\@auxout	57, 59, 93, 169, 171, 174, 183, 187, 263
\@Glsfirst@	127	\@backslashchar	112, 118, 119
\@Glsfirstplural	129	\@before	16
\@Glsfirstplural@	129	\@bsphack	176
\@Glsname	130	\@cGls	93
\@Glsname@	130	\@cGls@	92, 94
\@Glspl	123, 124	\@cGlspl	94
\@Glspl@	92, 95, 124, 260, 261	\@cGlspl@	92, 94
\@Glsplural	128	\@cclv	281
\@Glsplural@	128	\@cgls	93
\@Glsymbol	132	\@cgls@	91, 93
\@Glsymbol@	132	\@cglspl	94
\@Glsymbolplural	133	\@cglspl@	91, 94
\@Glsymbolplural@	133	\@chapter	31
\@Glstext	127	\@classoptionslist	30
\@Glstext@	127	\@closegls	167, 168
\@Glsuseri	134	\@colht	281
\@Glsuseri@	134	\@colroom	281
\@Glsuserii	135	\@currentlabelname	7, 198
\@Glsuserii@	135	\@curroptions	30
\@Glsuseriii	135	\@declaredoptions	30

\@delimN	211	\@glo@counterprefix	174, 179–181, 208, 211
\@delimR	210	\@glo@default@sorttype ..	10, 172, 191, 192
\@disable@onlypremakeg	169	\@glo@defaultcounter	82, 83
\@disable@premakecs	31	\@glo@desc	62, 78, 79, 81, 83
\@disabled@gl\$addxdycounters	44	\@glo@descaccess	338–340
\@do@addcounter	43	\@glo@descplural	62, 78, 79
\@do@auxoutstuff	187	\@glo@descpluralaccess	338–340
\@do@glossentry	204, 336, 337	\@glo@do@sortentries	188
\@do@gl\$@getcounterprefix	180	\@glo@entry	155, 156
\@do@gl\$@islistofacronyms	16	\@glo@entryprefix	256
\@do@gl\$see	84	\@glo@entryprefixfirst	256
\@do@ifinlist	42	\@glo@entryprefixfirstplural ..	256, 257
\@do@newglossaryentry	218,	\@glo@entryprefixplural	256
	232, 234–236, 238–241, 243–248, 367–371	\@glo@esclabel	86, 87
\@do@seeglossary	171, 182	\@glo@etext	96–98
\@do@subglossentry	205, 337	\@glo@first	62, 79, 82, 83, 243, 371
\@do@wrglossary	110	\@glo@firstaccess	337, 339, 340
\@do@writeaux@info	183	\@glo@firstplural	62, 79, 82, 371
\@ehc	281	\@glo@firstpluralaccess	338–340
\@empty	13, 15, 28, 30, 31, 42, 44, 47,	\@glo@grabfirst	194
	50, 51, 81, 86, 110, 111, 121–125, 139–	\@glo@label	66, 72,
	145, 158, 161, 163, 167, 168, 175, 176,		73, 75–84, 90, 154, 256, 257, 311, 312, 340
	181, 198, 201, 208, 233, 235, 237, 239–	\@glo@list	84
	242, 244, 246, 248, 318, 320, 322, 354–356	\@glo@long	55, 66, 80, 83
\@end@fixbraces	182	\@glo@longaccess	338–340
\@end@fortrue	25, 54, 72, 264	\@glo@longpl	66,
\@esphack	177		80, 83, 232, 234, 236, 238, 241, 243, 245, 367
\@expandtwoargs	30	\@glo@longpluralaccess	338–340
\@first@one	21	\@glo@name	11, 61, 67, 79, 81–83
\@first@three	107, 120,	\@glo@no@assign@sortkey	170
	121, 123, 125, 139, 140, 142, 144, 354–356	\@glo@nonumberlist	65
\@first@two	23–	\@glo@numfmt	180, 318
	25, 70, 71, 107, 123–125, 140–142, 144, 145	\@glo@parent ...	13, 64, 79, 81, 82, 86, 87, 189
\@for	25,	\@glo@plural	62, 79, 81, 82, 369
	30, 31, 43, 44, 51, 70, 71, 112, 158, 160,	\@glo@pluralaccess	338–340
	169, 170, 177, 182, 188, 219, 233, 235,	\@glo@prefix	
	237, 239, 241, 244, 246, 248, 264, 265, 318		.. 8, 64, 79, 86, 87, 111, 112, 180, 317, 318
\@glo@@desc	83	\@glo@range	180, 317, 318
\@glo@@symbol	83	\@glo@see	63, 79, 84
\@glo@access	337, 339, 340, 342	\@glo@seeautonumberlist	8, 64
\@glo@addchildren	188, 193	\@glo@short	55, 66, 80, 83, 370
\@glo@assign@sortkey	170, 172, 200	\@glo@shortaccess	338–340, 367–370
\@glo@autosee	84	\@glo@shortpl	66, 80, 83,
\@glo@autoseehook	84		232, 234, 236, 238, 240, 243, 245, 367, 370
\@glo@check@m\$idxrangechar		\@glo@shortpluralaccess	338–340
	111, 112, 180, 317, 318	\@glo@sort	11, 13, 20, 62, 79, 82, 86, 87
\@glo@check@sortallowed ..	11–13, 170, 174	\@glo@sortedinsert	189
\@glo@childlist	188	\@glo@sortentries	191, 192
\@glo@counter	63, 79, 83	\@glo@sorthandler@case	192

\@glo@sorthandler@letter	191	\@gls@between	264, 265
\@glo@sorthandler@nocase	192	\@gls@body	147
\@glo@sorthandler@word	191	\@gls@checkactual	113, 165
\@glo@sortinghandler	188, 190	\@gls@checkbar	113, 165
\@glo@sortinglist	188, 189, 192	\@gls@checkedmkidx	112–118, 164–166
\@glo@sorttype	172, 193, 194, 200	\@gls@checkescactual	113, 165
\@glo@storeentry	11–13	\@gls@checkescbar	113, 165
\@glo@suffix	111, 112, 180, 318	\@gls@checkescquote	113, 164–166
\@glo@symbol	55, 63, 79, 83, 238, 243, 339	\@gls@checklevel	113, 165
\@glo@symbolaccess	338–340, 370	\@gls@checkmkidxchars	
\@glo@symbolplural	63, 79, 84	86, 111, 164, 171, 179, 181, 317, 318
\@glo@symbolpluralaccess	338–340	\@gls@checkquote	113, 164
\@glo@text	62,	\@gls@classI	158, 159
79, 82, 83, 121–125, 146–148, 238, 257, 369		\@gls@classII	158, 159
\@glo@textaccess ...	337, 339, 340, 367–370	\@gls@codepage	187
\@glo@thislabel	85	\@gls@counter	
\@glo@thislettergrp	194, 195	106, 109–111, 155, 174, 180, 181, 318
\@glo@thisvalue	55–57	\@gls@counterwithin	10, 198, 199, 201
\@glo@tmp	72, 73, 181	\@gls@ctr	43
\@glo@type .	7, 13, 63, 79–84, 155, 169, 175,	\@gls@currentlettergroup	193, 195
185–189, 193, 194, 197, 198, 217, 233,		\@gls@declareoption	
235, 237, 239, 241, 244, 246, 248, 263–265		8, 9, 14, 15, 18, 24, 26–29
\@glo@types	51,	\@gls@default	95
59, 89, 90, 155, 156, 169, 170, 254, 311		\@gls@default@value	
\@glo@useri	65, 80, 83	55, 56, 67, 68, 79, 81–83, 242, 256
\@glo@userii	65, 80, 83	\@gls@deffile	69, 70
\@glo@useriii	65, 80, 83	\@gls@defsort	11–13, 83
\@glo@useriv	65, 80, 83	\@gls@defsortcount	11–13, 59
\@glo@userv	65, 80, 83	\@gls@do@acronymsdef	14, 15, 30, 60
\@glo@uservi	65, 80, 83	\@gls@do@indexdef	29, 30, 60
\@glodesc	83	\@gls@do@numbersdef	29, 30, 60
\@glolist@	81	\@gls@do@symbolsdef	28, 29, 60
\@gloname	83	\@gls@do@symbolssdef	30
\@glossary@default@style		\@gls@doautomake	27, 28, 170
.....	8, 9, 185, 208, 209, 249	\@gls@docheckquotedef	164–166
\@glossaryentryfield	86	\@gls@docloadedfalse	4
\@glossarysection	39	\@gls@docloadedtrue	4
\@glossarystyle	185, 186, 198	\@gls@dodeflistparser	170
\@glossarysubentryfield	86, 87	\@gls@doentrydef	105
\@gls	120	\@gls@dolast	182, 183
\@gls@	91, 93, 120, 259, 260	\@gls@donext	182, 183
\@gls@link	108	\@gls@donext@def	154
\@gls@Hcounter	110, 111	\@gls@dothiswrite	167, 168
\@gls@ReturnAfterFi	211	\@gls@elem	264
\@gls@access@display	342, 343	\@gls@enablesavenonumberlist	70
\@gls@actualchar ..	86, 87, 115, 117, 162, 321	\@gls@encapchar	
\@gls@addpredefinedattributes .	157, 166	115, 116, 162, 180, 182, 318, 321
\@gls@adjustmode	155	\@gls@entry@count	92, 93
\@gls@automake	170		

<code>\@gls@entry@field</code>	<code>\@gls@noaccess</code>	342
..... 72, 73, 91, 147–153, 340–342	<code>\@gls@noexpand@fields</code>	69
<code>\@gls@escbsdq</code>	<code>\@gls@nohyperlist</code>	17, 61, 109
113, 163, 322	<code>\@gls@noidx@do</code>	194
<code>\@gls@expand@fields</code>	<code>\@gls@noidx@getgrouptitle</code>	171
68, 69	<code>\@gls@noidx@sanitizesort</code>	20, 173
<code>\@gls@expandonce</code>	<code>\@gls@noidx@setsanitizesort</code>	22, 173
68	<code>\@gls@noidx@loclist@finalsep</code>	173
<code>\@gls@extramakeindexopts</code>	<code>\@gls@noidx@loclist@prev</code>	173, 196
169	<code>\@gls@noidx@loclist@sep</code>	173, 196
<code>\@gls@fetchfield</code>	<code>\@gls@noref@warn</code>	171, 194
56	<code>\@gls@numberlink</code>	211
<code>\@gls@field@link</code>	<code>\@gls@numbersdef</code>	29
74, 75, 126–138	<code>\@gls@numlist@lastsep</code>	154
<code>\@gls@firsttok</code>	<code>\@gls@numlist@nextsep</code>	154
194	<code>\@gls@numlist@sep</code>	154
<code>\@gls@fixbraces</code>	<code>\@gls@old@chapter</code>	31
84	<code>\@gls@oldnewglossaryentryposthook</code> .	339
<code>\@gls@forbidtexext</code>	<code>\@gls@oldnewglossaryentryprehook</code> ..	339
59	<code>\@gls@onlypremakeg</code>	31
<code>\@gls@get@counterprefix</code>	<code>\@gls@order</code>	167, 168
181	<code>\@gls@org@LT@output</code>	281
<code>\@gls@getbody</code>	<code>\@gls@org@glsnoidxdisplayloc</code>	173
147	<code>\@gls@org@glssseeformat</code>	173
<code>\@gls@getcounterprefix</code>	<code>\@gls@patchtabularx</code>	87
180	<code>\@gls@preglossaryhook</code>	186
<code>\@gls@getgrouptitle</code>	<code>\@gls@prevlevel</code> . 291–293, 312–315, 330, 331	
171, 207, 265	<code>\@gls@provide@newglossary</code>	59
<code>\@gls@glossary</code>	<code>\@gls@quotechar</code> 114–117, 162–166, 321	
176	<code>\@gls@reference</code>	171, 174
<code>\@gls@gobbleopt</code>	<code>\@gls@removespaces</code>	211
58	<code>\@gls@renewglossary</code>	167
<code>\@gls@grptitle</code>	<code>\@gls@replacementtext</code>	342
207, 263, 265	<code>\@gls@rest</code>	147
<code>\@gls@hyp@opt</code>	<code>\@gls@roman</code>	46, 318, 319
74,	<code>\@gls@sanitized@tmp</code>	112
75, 93, 94, 108, 120–145, 214–216, 258–261	<code>\@gls@sanitizedesc</code>	25
<code>\@gls@hyp@opt@cs</code>	<code>\@gls@sanitizesort</code>	11
107	<code>\@gls@sanitizesymbol</code>	26
<code>\@gls@hypergroup</code>	<code>\@gls@saveentrycounter</code>	110, 155
263	<code>\@gls@savenonumberlist</code>	64, 65
<code>\@gls@ifinlist</code>	<code>\@gls@see@noindex</code>	6, 64
43	<code>\@gls@setacrstyle</code>	25, 26, 30
<code>\@gls@ifnotmeasuring</code>	<code>\@gls@setcounter</code>	59
87	<code>\@gls@setdefault@glslink@opts</code>	109
<code>\@gls@igtype</code>	<code>\@gls@setsort</code>	11–13, 110
61	<code>\@gls@setupshortcuts</code>	30
<code>\@gls@increment@currcount</code>	<code>\@gls@sort</code>	195
91	<code>\@gls@sort@A</code>	190, 191
<code>\@gls@indexdef</code>		
29		
<code>\@gls@initnonumberlist</code>		
65, 80		
<code>\@gls@islistofacronyms</code>		
16		
<code>\@gls@keylist</code>		
366		
<code>\@gls@keymap</code>		
65, 70–73, 256, 339		
<code>\@gls@label</code>		
171, 174, 180		
<code>\@gls@langmod</code>		
167, 168		
<code>\@gls@levelchar</code> 87, 116, 117, 162, 321		
<code>\@gls@link</code> .. 108, 121–126, 139–146, 354–356		
<code>\@gls@link@checkfirsthyper</code> 120–125		
<code>\@gls@link@label</code>		
109, 234, 240		
<code>\@gls@link@nocheckfirsthyper</code> 126, 139–145		
<code>\@gls@link@opts</code>		
109, 234, 240		
<code>\@gls@list</code>		
264, 265		
<code>\@gls@listsuffix</code>		
42		
<code>\@gls@loadlist</code>		
9, 248		
<code>\@gls@loadlong</code>		
8, 9, 249		
<code>\@gls@loadsuper</code>		
9, 249		
<code>\@gls@loadtree</code>		
9, 249		
<code>\@gls@local@increment@currcount</code> 91		
<code>\@gls@loclist</code>		
172, 173, 195, 196		
<code>\@gls@map</code>		
70, 71		
<code>\@gls@missingnumberlist</code>		
83		

\@gls@sort@B	190, 191	\@glshypernumber	210
\@gls@startswithexpandonce	68	\@glsisacronymlistfalse	16
\@gls@storenonumberlist	64, 65, 83	\@glsisacronymlisttrue	16
\@gls@symbolsdef	28	\@glslink	110, 120, 154, 263
\@gls@this	177	\@glslocalreset	88, 91
\@gls@thisHloc	181	\@glslocalunset	88, 91
\@gls@thisfield	56	\@glslocref	174, 179, 180, 317, 318
\@gls@thislabel	54, 182, 183, 192	\@glsminrange	157–159, 319
\@gls@thislist	154	\@glsname	129
\@gls@thisloc	181	\@glsname@	129, 130
\@gls@thisval	71	\@glsnextpages	186
\@gls@title	39	\@glsnodesc	79, 81, 83
\@gls@tmp 13, 34, 47, 68, 112, 176, 177, 264, 265		\@glsnoname	79, 81, 83
\@gls@tmpb	113–118, 164–166	\@glsnonextpages	185
\@gls@toc	40	\@glsnumberformat	
\@gls@type	170, 219, 233, 235, 237, 239, 241, 244, 246, 248, 311	106, 109, 155, 174, 180, 317, 318
\@gls@updatechecked	112, 113, 164, 165	\@glsopenfile	167, 175
\@gls@usetranslator	24, 33	\@glsorder	169
\@gls@value	67, 68, 150	\@glspl	123
\@gls@warnonglossdefined	18, 184	\@glspl@	92, 94, 123, 259–261
\@gls@warnontheGLOSSdefined	18, 203	\@glsplural	128
\@gls@write@entrycounts	92	\@glsplural@	128
\@gls@writedef	70	\@glsreset	88, 91
\@gls@writeisthook	161, 163	\@glssee	84, 182
\@gls@xdy@locationlist	158	\@glssymbol	132
\@gls@xdycheckbackslash	112	\@glssymbol@	132
\@gls@xdycheckquote	112	\@glssymbolplural	133
\@gls@xref	181, 182	\@glssymbolplural@	133
\@gls@Alphacompositor	37, 46, 319	\@glstarget	120, 203, 263
\@glsHlocref	179, 180	\@glstext	126
\@glsacronymlists .. 15–17, 51, 217, 219, 233, 235, 237, 239, 241, 244, 246, 248, 253		\@glstext@	126
\@glsaddkey	73	\@glsunset	88, 91
\@glsaddstoragekey	72	\@glsuseri	134
\@glsaddxdyattribute	43, 44	\@glsuseri@	134
\@glsdefaultsort	11	\@glsuserii	134
\@glsdesc	130	\@glsuseriii	135
\@glsdesc@	130	\@glsuseriii@	135
\@glsdescplural	131	\@glsuseriv	136
\@glsdescplural@	131	\@glsuseriv@	136
\@glsdisp	125	\@glsuserv	137
\@glsentry	89, 90, 93	\@glsuserv@	137
\@glstrytitlecase	150	\@glsuservi	138
\@glsfirst	127	\@glsuservi@	138
\@glsfirst@	127	\@glswidestname	312, 313, 330
\@glsfirstletter	50, 157	\@glswritefiles	28
\@glsfirstplural	129	\@glxtr@doaccsupp	336
\@glsfirstplural@	129	\@gobble	11–13, 70, 87, 112, 156, 157, 160, 171, 316, 320, 321

\@idxitem	306	\@print@glossary	184
\@ifclassloaded	4, 10, 39	\@print@noidx@glossary	184
\@ifnextchar	59, 107	\@printgloss@setsort	170, 172, 185
\@ifpackageloaded		\@printglossary	184
..... 4, 7, 23–25, 33, 50, 87, 153, 163, 336		\@roman	46, 318
\@ifstar	58, 72, 73, 107, 213	\@secondofthree	
\@ifundefined		107, 120, 121, 124, 139, 141, 143, 145, 354	
. 33, 264, 271, 282, 293, 300, 313, 330, 344		\@secondoftwo . 21, 23–25, 34, 70, 71, 119–	
\@ignored@glossaries	60, 61	122, 125, 139, 140, 142, 143, 354–356, 374	
\@input@	186	\@set@glo@numformat	180, 318
\@istfilename	169	\@sglsaddkey	73
\@makecol	281	\@sglsaddstoragekey	72
\@makeglossary	169	\@thirdofthree	
\@minus	268, 287, 306 107, 122, 125, 140, 142, 143, 145, 354	
\@mkboth	39, 40	\@this@attr	160
\@newglossary	57, 59	\@this@childlabel	188, 189
\@newglossaryentry@defcounters ..	84, 90	\@this@counter	44
\@newglossaryentryposthook		\@this@ctr	160
..... 72, 73, 84, 256, 339		\@this@key	71
\@newglossaryentryprehook		\@this@label	188
..... 72, 73, 78, 80, 256, 339		\@thiscs	31
\@nil	16, 84, 111–113, 147, 164,	\@tmp	46, 319
165, 180, 182, 194, 195, 210, 211, 317, 318		\@use@option	30
\@nnil	16, 182	\@warn@nomakeglossaries	168, 188
\@no@makeglossaries	170, 172	\@wrglossary@pageformat	178
\@no@post@desc	323	\@wrglossarynumberhook	178, 179
\@nopostdesc	186	\@xdy@main@language	27, 167, 187
\@onelevel@sanitize	20,	\@xdy@attributelist	44, 160
46, 70, 86, 112, 161, 181, 183, 194, 319, 320		\@xdy@attributes	43, 158, 316, 318
\@onlypreamble 60, 69, 78, 79, 92, 95, 170, 174		\@xdy@counters	42–44, 160
\@onlypremakeg 35–37, 43, 44, 48, 60, 163		\@xdy@crossrefhook	159
\@org@glossaryentrynumbers 185, 186		\@xdy@language	187
\@org@gls@assign@descplural		\@xdy@lettergroups	50, 161, 321
..... 232, 241, 243–246, 367, 370, 371		\@xdy@locationclassorder 48, 159, 320	
\@org@gls@assign@firstpl	232,	\@xdy@locref	43, 161, 316, 320
234–236, 238, 239, 241, 243–246, 367–371		\@xdy@requiredstyles	49, 158, 318
\@org@gls@assign@plural	232,	\@xdy@sortrules	48, 161, 321
234–236, 238, 239, 241, 243–246, 367–371		\@xdystyle	158, 318
\@org@gls@assign@symbolplural ..	232,	\@xdy@useralphabets	45, 158, 318
234–236, 238, 239, 244–246, 368, 369, 371		\@xdy@userlocationdefs ... 47, 159, 317, 319	
\@org@gls@numberformat	154	\@xdy@userlocationnames	47, 48, 317
\@org@newglossaryentryprehook	78	\@xfor@nextelement	182
\@outputpage	281	\\	85, 112, 156, 162,
\@p@glossarysection	39	163, 210, 211, 321, 322, 324–326, 334, 335	
\@pgls	258	\{	70, 156, 162, 163, 316, 321, 322
\@pgls@	258	\}	70, 156, 162, 163, 316, 322
\@pglspl	259	\^	21
\@pglspl@	259	\‘	21
\@plus	268, 287, 306	\ 	113, 115, 165

\~	21	\advance	12, 13, 81, 110, 281
A		\AE	21
\a	21	\ae	21
\AA	21	amsgen package	4, 106
\aa	21	amsmath package	87
accsupp package	336	\andname	183
\accsuppglossaryentryfield	336	\AnyTrackedLanguages	34, 374
\accsuppglossarysubentryfield	337	\appto	17, 65, 72, 73, 256, 339
\acrfootnote	234, 240	array package	277, 282, 300
\Acrfull	231	article class	181
\acrfull	231	\AtBeginDocument	15, 50, 70, 87, 155, 171
\ACRfullfmt	215, 218, 226, 228, 362, 364	\AtEndDocument .	28, 70, 92, 171, 175, 187, 264
\Acrfullfmt	215, 218, 226, 228, 362, 364	B	
\acrfullfmt	214, 218, 226, 228, 362, 364	\b	21
\acrfullformat	153, 214, 232, 247	babel package	23, 32, 33, 49
\Acrfullpl	231	\begin	160, 193, 268, 271–274, 276, 277, 279, 280, 282–305, 320
\acrfullpl	231	\BeginAccSupp	342
\ACRfullplfmt ...	216, 218, 226, 228, 362, 364	\begingroup	5, 176, 179, 211
\Acrfullplfmt ...	216, 218, 226, 228, 362, 364	\bfseries	272– 276, 278, 279, 283–287, 295–299, 301–305
\acrfullplfmt ...	216, 218, 226, 228, 362, 364	\bgroup	20, 78, 153, 185, 188
\acrlinkfootnote	233	booktabs package	277–280
\acrlinkfullformat	214–216	\boolean	247
\Acrlong	231	\boolfalse	28
\acrlong	230	\booltrue	28
\Acrlongpl	231	\bottomrule	278, 279
\acrlongpl	230	\box	281
\acrnameformat	238, 369	C	
\acronymentry	218, 220–225, 227–230, 358–360, 363–366	\c	21
\acronymfont	102, 103, 139–142, 147, 153, 217, 218, 220–230, 234, 235, 237, 239, 240, 242–244, 351, 352, 354–356, 358–360, 362–366, 368–370	\c@equation	110
\acronymname	14, 15, 35	\c@glossarysubentry	199
\acronymsort	218, 220–225, 227–230, 358–360, 363, 364, 366	\c@page	177–179
\acronymtype	14, 15, 217–219, 232–241, 243–246, 248, 367–370	\cGls	94
\acrpluralsuffix	218, 220–223, 227–229, 232–236, 238– 245, 247, 248, 358, 359, 363–365, 367–371	\cgl	93
\Acrshort	230	\cGlsformat	92
\acrshort	230	\cglformat	91
\Acrshortpl	230	\cGlspl	95
\acrshortpl	230	\cglpl	94
\addcontentsline	41	\cGlsplformat	92
\addglossarytocaptions	34	\cglplformat	91
\addtolength	313, 330	\char	207
		classicthesis package	7
		\cleardoublepage	41
		\clearpage	41
		\closeout	70, 161, 163, 167, 175
		\compatglossarystyle	323–335
		\compatibleglossentry	205

<code>\compatiblesubglossentry</code>	205	<code>\def@glx@xdycheckbackslash</code>	118, 119
<code>\copy</code>	281	<code>\DefaultNewAcronymDef</code>	233
<code>\count@</code>	194	<code>\defglentryfmt</code>	59, 61, 105, 219, 232, 233, 236, 237, 240, 242, 245, 247
<code>\csdef</code>	19, 72–75, 84, 85, 90, 91, 188, 189, 209, 219, 220, 322	<code>\define@boolkey</code> 5, 6, 8, 10, 14, 21, 22, 25–28, 106, 200
<code>\csedef</code>	92, 178	<code>\define@choicekey</code> 6, 7, 10, 22, 24, 26, 64, 198, 199
<code>\csgdef</code>	38, 57, 61, 91, 92, 183, 197	<code>\define@key</code>	7, 10, 17, 23, 27, 61–66, 72, 73, 106, 155, 197, 198, 200, 256, 337, 338
<code>\cslet</code>	65, 78, 85, 192	<code>\DefineAcronymSynonyms</code>	30, 231
<code>\csname</code> ...	10–13, 30, 32, 34, 35, 40, 43, 44, 46, 47, 49, 51, 54, 59, 60, 67, 68, 72–77, 81, 82, 84–87, 89, 105, 109–111, 121– 125, 139–146, 154, 155, 158–160, 164– 167, 171, 174–178, 180–182, 185–187, 189, 198, 204, 205, 208, 209, 213, 249– 257, 264, 265, 312, 313, 316–318, 330, 336, 337, 340, 341, 344, 354–356, 372, 373	<code>\delimN</code>	160, 170, 196, 211, 320
<code>\csshow</code>	253	<code>\delimR</code>	160, 210, 211, 320
<code>\csuse</code>	34, 38, 57, 67, 68, 74, 75, 105, 168, 189, 191, 193, 195, 197–199, 209, 220, 257, 323–335	<code>\DescriptionDUANewAcronymDef</code>	237
<code>\csxdef</code>	83, 92	<code>\DescriptionFootnoteNewAcronymDef</code> .	235
<code>\currentglossary</code>	38, 185, 199, 201	<code>\descriptionname</code>	35, 272– 276, 278, 279, 283–287, 295–299, 301–305
<code>\currentglssubentry</code>	199, 201, 202	<code>\DescriptionNewAcronymDef</code>	239
<code>\CurrentOption</code>	30, 256, 336	<code>\DH</code>	21
<code>\CurrentTrackedLanguage</code>	34, 374, 375	<code>\dh</code>	21
<code>\CurrentTrackedTag</code>	34, 374, 375	<code>\dimen@</code>	221, 281, 311
<code>\CustomAcronymFields</code>	248	<code>\disable@keys</code>	30
<code>\CustomNewAcronymDef</code>	248	<code>\do</code>	25, 30, 31, 43, 44, 51, 70, 71, 112, 154, 158, 160, 169, 170, 177, 182, 188, 219, 233, 235, 237, 239, 241, 244, 246, 248, 264, 265, 318
D		<code>\do@glo@storeentry</code>	11–13, 84
<code>\d</code>	21	<code>\do@glx@link@checkfirsthyper</code> 108, 109, 120–126, 139–145, 354, 355
<code>datatool package</code>	190	<code>\do@glx@xdycheckbackslash</code>	112
<code>\day</code>	158, 162, 318, 321	<code>\do@glx@disablehyperinlist</code>	109
<code>\DeclareAcronymList</code>	14, 15, 17, 217, 219, 233, 235, 237, 239, 241, 244, 246, 248	<code>\do@glx@haschildren</code>	54
<code>\DeclareListParser</code>	170	<code>doc package</code>	4, 5, 14
<code>\DeclareOption</code>	8, 256, 336	<code>\doifglossarynoexistsordo</code>	58
<code>\DeclareOptionX</code>	8	<code>\dtl@ifsingle</code>	207
<code>\DeclareRobustCommand</code> 35, 182, 183, 242, 342–344	<code>\dtl@insertinto</code>	190
<code>\def</code>	8, 11–13, 16, 20, 21, 26, 27, 30–32, 35, 36, 39, 42, 45–50, 54, 58–66, 68, 76, 78–83, 85, 87, 91–95, 105, 106, 109–146, 154, 155, 157, 162–169, 172, 173, 175, 178–182, 185, 188, 192–194, 196–201, 207–211, 213–217, 232–239, 241, 243–246, 248, 256, 258–262, 264– 267, 291–293, 312–315, 317, 318, 321, 323, 330, 331, 336–339, 353–356, 367–371	<code>\dtl@sortresult</code>	190, 191
		<code>\dtlcompare</code>	190
		<code>\dtlicompare</code>	191
		<code>\DTLifinlist</code>	61, 109
		<code>\DTLifint</code>	207
		<code>\dtlletterindexcompare</code>	190
		<code>\DTLsubstituteall</code>	112
		<code>\dtlwordindexcompare</code>	190
		<code>\DUANewAcronymDef</code>	246
E		<code>\eappto</code>	61, 85, 178

<code>\edef</code> ..	13, 16, 31, 34, 42–47, 49, 51, 54, 59, 61, 67, 68, 71, 75–80, 85, 86, 105, 109–118, 154–157, 162–168, 170, 171, 175, 180, 181, 183, 187–191, 194, 199, 202, 207, 211, 213, 232, 234, 236, 238, 240, 243, 245, 263, 316, 317, 319, 321, 366–370
<code>\egroup</code>	20, 78, 154, 186, 189
<code>\else</code>	5, 9, 13–16, 18, 20–23, 28, 30, 31, 35–37, 39, 41–51, 64, 67, 81, 82, 85–87, 91, 92, 108–110, 112–119, 121–126, 147, 157, 158, 161, 163–168, 175–182, 185, 194, 198–203, 208, 210, 211, 221, 235, 237, 239, 242, 244–247, 249, 264, 268, 272, 273, 275, 278, 279, 281, 283, 284, 286, 294, 296, 297, 301, 303, 304, 307–309, 311–314, 317–323, 328–331, 342
<code>\emph</code>	182, 212
<code>\empty</code>	211, 336
<code>\end</code>	160, 194, 268, 271–274, 276, 277, 279, 280, 282–306, 320
<code>\end@doifinlist</code>	42
<code>\end@getprefix</code>	181
<code>\end@glis@islistofacronyms</code>	16
<code>\EndAccSupp</code>	342
<code>\endcsname</code>	10–13, 30, 32, 34, 35, 40, 43, 44, 46, 47, 49, 51, 54, 59, 60, 67, 68, 72–77, 81, 82, 84–87, 89, 105, 109–111, 121–125, 139–146, 154, 155, 158–160, 164–167, 171, 174–178, 180–182, 185–187, 189, 198, 204, 205, 208, 209, 213, 249–257, 264, 265, 312, 313, 316–318, 330, 336, 337, 340, 341, 344, 354–356, 372, 373
<code>\endfoot</code> ..	272–274, 276, 278, 279, 283–285, 287
<code>\endgroup</code>	5, 177, 179, 211
<code>\endhead</code>	272–276, 278, 279, 283–287
<code>\endtheglossary</code>	5
<code>\entryname</code>	35, 272–276, 278, 279, 283–287, 295–299, 301–305
<code>\equal</code>	23, 31, 41, 110, 169, 208, 264
<code>equation (counter)</code>	110, 111
<code>etoolbox package</code>	4
<code>\expandafter</code>	11–13, 20, 30, 31, 34, 43, 44, 46–51, 54, 59–61, 67, 68, 70–77, 81–84, 86, 87, 89, 105, 109–118, 147, 154, 156, 157, 160, 164, 165, 167, 175–177, 179, 180, 183, 185, 189, 194, 195, 204, 205, 209, 211, 213, 234, 240, 249–254, 256, 257, 264, 265, 312, 316–318, 320, 321, 336, 337, 340, 342, 366, 372, 373
<code>\expandonce</code>	67, 68, 112, 164–166, 178, 190, 191, 204, 205, 218, 232, 234, 236, 238, 240, 241, 243, 245, 336, 337
F	
<code>\fi</code>	5–7, 9, 11–16, 18, 20–23, 25, 28, 30, 31, 35–37, 39–51, 60, 64, 67, 81–83, 85–87, 91, 92, 108–119, 121–126, 148, 155, 157, 158, 161, 163–168, 170, 175–183, 185, 187, 194, 198–203, 208–211, 221, 231, 233, 235–237, 239, 241, 242, 244–249, 255, 264, 268, 272, 273, 275, 278–281, 283, 284, 286, 294, 296, 297, 301, 303, 304, 308–314, 316–320, 322, 323, 328–331, 336, 342
file types	
<code>.aux</code>	187
<code>.glo</code>	85
<code>.ist</code>	156, 166
<code>.toc</code>	41
<code>.xdy</code>	36
<code>glo</code>	254
<code>\firstacronymfont</code> ..	104, 220–222, 227, 228, 234, 238, 240, 243, 353, 357–359, 363, 364
<code>\footnote</code>	227, 228, 233, 364
<code>\FootnoteNewAcronymDef</code>	241
<code>\forall glossaries</code>	51, 175, 184, 311
<code>\forall glossentries</code>	89, 90, 93, 155, 156
<code>\ForEachTrackedDialect</code>	34, 374, 375
<code>\forall glossentries</code>	51, 54, 85, 192, 311
<code>\forall listcsloop</code>	188, 194
<code>\forall listloop</code>	173, 196
G	
<code>garamondx package</code>	213
<code>\gdef</code>	12, 44, 59, 76, 81, 82, 176, 200, 264
<code>\Genacrfullformat</code>	103, 218, 220–222, 227, 353, 357, 358, 364
<code>\genacrfullformat</code>	103, 104, 218, 220, 221, 227, 352, 353, 357, 358, 363
<code>\GenericAcronymFields</code> ..	218, 220, 221, 223–227, 229, 230, 357–360, 362, 363, 366
<code>\Genplacrfullformat</code>	103, 218, 220–222, 228, 352, 358, 364
<code>\genplacrfullformat</code>	103, 104, 218, 220–222, 228, 352, 358, 364
<code>\glo@desc</code>	323
<code>\glo@do@compare</code>	190, 191
<code>\glo@grabfirst</code>	195
<code>\glo@label</code>	54, 85

<code>\glo@list</code>	85	<code>index</code>	7, 288, 306–308, 328
<code>\glo@name</code>	204	<code>indexgroup</code>	308, 328
<code>\glo@parent</code>	54	<code>indexhypergroup</code>	308, 328
<code>\glo@type</code>	85	<code>inline</code>	323
<code>\glo@value</code>	70	<code>list</code>	7, 9, 268–270, 323
<code>\global</code>	12, 13, 67, 69, 78, 84, 89, 176, 186, 195, 201, 281	<code>listdotted</code>	270, 271, 324
<code>\glo@linkprefix</code>	110, 154, 203	<code>listgroup</code>	269, 323
<code>\glosortentrieswarning</code>	18, 188	<code>listhypergroup</code>	269, 324
<code>glossareentry (counter)</code>	201	<code>long</code>	271–273, 278, 282, 324, 326
<code>glossaries package</code>	<code>long-booktabs</code>	277, 280
<code>glossaries-accsupp package</code>	29, 49, 50, 157, 248, 256, 268, 316, 336	<code>long3col</code>	273, 274, 278, 325
<code>glossaries-extra package</code>	85, 336	<code>long3col-booktabs</code>	278, 280
<code>\GlossariesWarning</code>	159, 336	<code>long3colborder</code>	274, 325
.....	5, 6, 18, 21–23, 38, 42, 53, 57, 64, 66, 93–95, 105, 107, 153, 168, 171– 174, 177, 181, 185, 205, 206, 208, 316, 336	<code>long3colheader</code>	274, 278, 325
<code>\GlossariesWarningNoLine</code>	<code>long3colheaderborder</code>	274, 325
.....	5, 18, 169, 170, 172, 175, 188, 264	<code>long4col</code>	274–276, 279, 325
<code>\glossary</code>	317, 318	<code>long4col-booktabs</code>	278, 279
<code>glossary package</code>	1, 212	<code>long4colborder</code>	275, 326
<code>glossary styles:</code>		<code>long4colheader</code>	275, 278, 326
<code>altlist</code>	269, 270, 324	<code>long4colheaderborder</code>	276, 326
<code>altlistgroup</code>	270, 324	<code>longborder</code>	272, 325
<code>altlisthypergroup</code>	270, 324	<code>longheader</code>	272, 277, 278, 325
<code>altlong4col</code>	276, 277, 285, 326	<code>longheaderborder</code>	273, 325
<code>altlong4col-booktabs</code>	279, 280	<code>longragged</code>	279, 282–284
<code>altlong4colborder</code>	277, 326	<code>longragged-booktabs</code>	279
<code>altlong4colheader</code>	276, 279, 326	<code>longragged3col</code>	280, 284, 285, 327
<code>altlong4colheaderborder</code>	277, 326	<code>longragged3col-booktabs</code>	280
<code>altlongragged4col</code> ...	280, 285, 286, 327	<code>longragged3colborder</code>	284, 327
<code>altlongragged4col-booktabs</code>	280	<code>longragged3colheader</code>	285, 327
<code>altlongragged4colborder</code>	286, 327	<code>longragged3colheaderborder</code> .	285, 327
<code>altlongragged4colheader</code>	286, 327	<code>longraggedborder</code>	283, 326
<code>altlongragged4colheaderborder</code>	287, 328	<code>longraggedheader</code>	283, 327
<code>altsuper4col</code>	298, 299, 304, 335	<code>longraggedheaderborder</code>	283, 327
<code>altsuper4colborder</code>	299, 335	<code>mcolalttree</code>	292, 332
<code>altsuper4colheader</code>	299, 335	<code>mcolalttreegroup</code>	292, 332
<code>altsuper4colheaderborder</code> ...	299, 335	<code>mcolalttreehypergroup</code>	292, 332
<code>altsuperragged4col</code>	304, 305, 333	<code>mcolindex</code>	288, 331
<code>altsuperragged4colborder</code> ...	305, 333	<code>mcolindexgroup</code>	288, 331
<code>altsuperragged4colheader</code> ...	305, 333	<code>mcolindexhypergroup</code>	288, 331
<code>altsuperragged4colheaderborder</code>	<code>mcoltree</code>	289, 331
.....	305, 333	<code>mcoltreegroup</code>	331
<code>alttree</code>	291, 306, 307, 312, 314, 330	<code>mcoltreehypergroup</code>	289, 290, 331
<code>alttreegroup</code>	314, 331	<code>mcoltreenoname</code>	290, 332
<code>alttreehypergroup</code>	314, 331	<code>mcoltreenonamegroup</code>	291, 332
		<code>mcoltreenonamehypergroup</code> ...	291, 332
		<code>sublistdotted</code>	324
		<code>super</code>	293–295, 301, 334
		<code>super3col</code>	295, 296, 334

super3colborder	296, 334	\glossarytitle	159, 185, 193, 197, 320
super3colheader	296, 334	\glossarytoctitle	7, 14, 15,
super3colheaderborder	296, 334		29, 32, 35, 39, 159, 185, 193, 197, 198, 320
super4col	297, 298, 335	\glossentry	85, 186, 195, 196, 205,
super4colborder	298, 335		266, 268–273, 275, 282, 284, 286, 294,
super4colheader	297, 335		295, 297, 301, 302, 304, 307, 309, 310, 312
super4colheaderborder	298, 335	\Glossentrydesc	356
superborder	294, 334	\glossentrydesc	266, 268–
superheader	294, 334		270, 272, 273, 275, 282–284, 286, 294–
superheaderborder	295, 334		297, 301, 302, 304, 307–310, 313, 314, 356
superragged	300, 302, 332	\glossentryname	266,
superragged3col	302, 303, 333		268–273, 275, 282, 284, 286, 294, 295,
superragged3colborder	303, 333		297, 301, 302, 304, 307–310, 312, 314, 356
superragged3colheader	303, 333	\Glossentrysymbol	357
superragged3colheaderborder	303, 333	\glossentrysymbol	266,
superraggedborder	301, 332		275, 286, 297, 304, 307–310, 313, 314, 357
superraggedheader	301, 332	\Gls	94, 213, 231
superraggedheaderborder	302, 333	\gls	93, 171, 202, 213, 231
tree	289, 308–310, 312, 328	\gls@Alphpage	177, 179
treegroup	289, 309, 329	\gls@alphpage	177, 179
treehypergroup	309, 329	\gls@arabicpage	177, 179
treenoname	290, 307, 310, 311, 329	\gls@assign@desc	78, 83
treenonamegroup	311, 330	\gls@assign@descplural	
treenonamehypergroup	311, 330		232, 241, 243–246, 367, 370, 371
glossary-hypernav package	156	\gls@assign@field	
glossary-list package	7, 9, 267		69, 72, 73, 78, 80, 82, 83, 256, 257
glossary-long package	8, 271, 285, 293	\gls@assign@firstpl	232,
glossary-longragged package	282		234–236, 238, 239, 241, 243–246, 367–371
glossary-mcols package	287	\gls@assign@plural	232,
glossary-super package ...	9, 271, 293, 300, 304		234–236, 238, 239, 241, 243–246, 367–371
glossary-superragged package	300	\gls@assign@symbolplural	232,
glossary-tree package	9, 306		234–236, 238, 239, 244–246, 368, 369, 371
\glossaryentry	180, 182, 318	\gls@checkisacronymlist	108
glossaryentry (counter)	10, 202, 203	\gls@checkseeallowed	64, 69, 169, 171
\glossaryentryfield		\gls@checkseeallowed@preambleonly ..	69
	204, 323–330, 332–335, 357	\gls@codepage	50, 168, 187
\glossaryentrynumbers		\gls@defdocnewglossaryentry	70, 90
	8, 160, 185, 186, 195, 196, 200, 201, 320	\gls@defglossaryentry	69, 70, 78
\glossaryheader	160, 193, 266, 268–	\gls@disablepagerefexpansion ..	177, 179
	276, 278, 279, 282–288, 290–292, 294,	\gls@do@addxdyattribute	44
	295, 297, 300, 302, 304, 307–312, 314, 320	\gls@doclearpage	41
\glossarymark	39	\gls@dosubst	112
\glossaryname	14, 34, 35	\gls@dotocitle	185, 186, 197
\glossarypostamble	160, 194, 320	\gls@end@sanitizesort	20
\glossarypreamble	159, 193, 320	\gls@endcheck	68
\glossarysection	159, 193, 320	\gls@glossary	176, 180–182
glossarysubentry (counter)	10, 201–203	\gls@gobbleopt	59
\glossarysubentryfield		\gls@grplabel	263
	205, 323–330, 332–335, 357	\gls@hypergroup prerun	264

<code>\gls@ifnotmeasuring</code>	88	<code>\glsautomakefalse</code>	28
<code>\gls@inlinepostchild</code>	265–267, 323	<code>\glsautoprefix</code>	7, 198
<code>\gls@inlinessep</code>	265, 266, 323	<code>\glscapscase</code>	96, 98, 100–103, 121–125, 139–145, 225, 226, 238, 243, 344, 346, 348, 349, 351, 352, 354–356, 361
<code>\gls@inlinesubsep</code>	265, 266, 323	<code>\glsclearpage</code>	40
<code>\gls@islistofacronyms</code>	16	<code>\glsclosebrace</code>	47, 160, 161, 320, 321
<code>\gls@istfilebase</code>	35, 36, 168	<code>\glscompositor</code>	36, 47, 163, 319, 322
<code>\gls@label</code>	213	<code>\glscounter</code>	17, 31, 42, 59, 82, 110, 316
<code>\gls@level</code>	81, 82, 195	<code>\glscurrententrylabel</code>	183, 186
<code>\gls@noidxglossary</code>	171	<code>\glscurrentfieldvalue</code>	56, 57
<code>\gls@nosetquote</code>	79, 162, 163, 166	<code>\glscustomtext</code>	96, 99, 100, 102, 104, 121–125, 139–146, 225, 226, 233, 234, 237, 238, 240, 242, 243, 344, 347, 348, 350, 351, 353–356, 361, 362
<code>\gls@numberpage</code>	177, 179	<code>\GlsDeclareNoHyperList</code>	17
<code>\gls@org@glossaryentryfield</code>	186	<code>\glsdefaulttype</code>	14, 38, 49, 51, 57–59, 80, 95, 105, 175, 184, 185
<code>\gls@org@glossarysubentryfield</code>	186	<code>\glsdefmain</code>	14, 60
<code>\gls@org@insert</code>	237, 240, 242	<code>\glsdescriptionaccessdisplay</code>	346–348, 356, 357
<code>\gls@protected@pagefmts</code>	112, 177, 178	<code>\glsdescriptionpluralaccessdisplay</code>	344, 345
<code>\gls@Romanpage</code>	177, 179	<code>\glsdescwidth</code>	271–274, 276, 277, 279, 280, 282–287, 293–297, 299–305
<code>\gls@romanpage</code>	177, 179	<code>\glsdetoklabel</code> .	52–56, 65, 70, 75–79, 85, 89, 91, 92, 109, 146, 147, 154, 155, 171– 173, 180, 186, 189–191, 195, 197, 199, 202, 204, 249–253, 312, 336, 337, 372, 373
<code>\gls@save@numberlist</code>	8	<code>\glsdisplay</code>	96, 105
<code>\gls@set@xr@key</code>	63	<code>\glsdisplayfirst</code>	96, 105
<code>\gls@suffixF</code>	37, 161, 163, 320, 322	<code>\glsdisplaynumberlist</code>	172
<code>\gls@suffixFF</code>	37, 161, 163, 320, 322	<code>\glsdohyperlink</code>	119, 120
<code>\gls@text</code>	104	<code>\glsdohypertarget</code>	119, 120
<code>\gls@thissty</code>	25	<code>\glsdoifexists</code>	54, 55, 75–78, 88, 120–126, 138– 145, 153, 155, 172, 173, 258–262, 353–357
<code>\gls@tmp</code>	175, 242	<code>\glsdoifexistsordo</code>	108, 146
<code>\gls@tmplen</code>	119, 311, 313, 314, 330, 331	<code>\glsdoifexistsorwarn</code>	197, 204, 205
<code>\gls@tr@set@acronym@toctitle</code>	15	<code>\glsdoifnoexists</code>	69, 78
<code>\gls@tr@set@main@toctitle</code>	14	<code>\glsdonohyperlink</code>	110, 119, 120
<code>\gls@tr@set@numbers@toctitle</code>	29	<code>\glsdosanitizesort</code>	11
<code>\gls@tr@set@symbols@toctitle</code>	29	<code>\glsentryaccess</code>	342
<code>\gls@wrglossary</code>	176	<code>\glsentrycounter</code>	208, 211
<code>\gls@xdystring</code>	112	<code>\glsentrycounterfalse</code>	10
<code>\gls@xindy@glsnumbersfalse</code>	27	<code>\glsentrycounterlabel</code>	199, 203
<code>\gls@xindy@glsnumberstrue</code>	27	<code>\glsentrycountertrue</code>	10
<code>\gls@xr@key</code>	6, 63, 64	<code>\glsentrycurrcount</code>	91–93
<code>\glsaccsupp</code>	342	<code>\Glsentrydesc</code>	131, 204, 356
<code>\glsacronymtrue</code>	15		
<code>\glsacrpluralsuffix</code>	33, 213, 222, 223, 227–229, 233		
<code>\glsacrshortcutsfalse</code>	30		
<code>\glsacrshortcutstrue</code>	30		
<code>\glsacspace</code>	221, 223		
<code>\glsadd</code>	156		
<code>\glsadd options</code>			
counter	155		
format	155, 210		
<code>\glsaddall options</code>			
types	155		
<code>\GlsAddXdyAttribute</code>	43–45, 316, 317		
<code>\GlsAddXdyCounters</code>	43, 44, 60		

<code>\glstentrydesc</code>	<code>\glstentryprevcount</code>
..... 98, 99, 130, 131, 204, 346–348, 356	91, 92
<code>\glstentrydescaccess</code>	<code>\Glstentryshort</code>
343	102,
<code>\Glstentrydescplural</code>	139, 147, 222, 228, 352–354, 358, 364, 365
131	<code>\glstentryshort</code> 102–104, 139, 140, 147, 153,
<code>\glstentrydescplural</code> 96, 97, 131, 132, 344, 345	218, 220–230, 351–354, 357–360, 362–366
<code>\glstentrydescpluralaccess</code>	<code>\glstentryshortaccess</code>
343	343
<code>\Glstentryfirst</code> 94, 99, 101, 127, 347, 350	<code>\Glstentryshortpl</code>
<code>\glstentryfirst</code> 102, 141, 222, 228, 351, 358, 364, 365
... 93, 98, 99, 101, 127, 128, 346, 347, 350	<code>\glstentryshortpl</code>
<code>\glstentryfirstaccess</code> 102, 104, 140, 142, 153, 220–
343	222, 226–228, 247, 351, 353, 358, 362–365
<code>\Glstentryfirstplural</code>	<code>\glstentryshortpluralaccess</code>
..... 95, 97, 100, 129, 345, 349	343
<code>\glstentryfirstplural</code>	<code>\Glstentrysymbol</code>
... 94, 96, 97, 100, 129, 344, 345, 348, 349	132, 205, 357
<code>\glstentryfirstpluralaccess</code>	<code>\glstentrysymbol</code>
343	98, 99,
<code>\glstentryfmt</code>	132, 133, 205, 234, 238, 243, 346–348, 357
59, 61	<code>\glstentrysymbolaccess</code>
<code>\Glstentryfull</code>	343
218, 227, 228, 363, 365	<code>\Glstentrysymbolplural</code>
<code>\glstentryfull</code>	133
218, 227, 228, 362, 365	<code>\glstentrysymbolplural</code>
<code>\Glstentryfullpl</code> 218, 227, 228, 363, 365 96, 97, 133, 234, 238, 243, 344, 345
<code>\glstentryfullpl</code> 218, 227, 228, 363, 365	<code>\glstentrysymbolpluralaccess</code>
<code>\glstentryitem</code> ... 199, 266, 268–273, 275,	343
282, 284, 286, 294, 295, 297, 301, 302,	<code>\Glstentrytext</code>
304, 307, 309, 310, 312, 323–330, 332–335	98, 101, 127, 346, 350
<code>\Glstentrylong</code>	<code>\glstentrytext</code>
94, 143, 147, 153,	98, 99,
220, 221, 226, 227, 353, 355, 358, 361–363	101, 126, 127, 154, 183, 346, 347, 349, 350
<code>\glstentrylong</code>	<code>\glstentrytextaccess</code>
93, 104, 142,	342
144, 147, 153, 220–230, 240, 353, 355–366	<code>\glstentrytype</code>
<code>\glstentrylongaccess</code>	80
343	<code>\Glstentryuseri</code>
<code>\Glstentrylongpl</code>	134
95, 145,	<code>\glstentryuseri</code>
153, 220, 221, 225–227, 353, 358, 361–363	134
<code>\glstentrylongpl</code>	<code>\Glstentryuserii</code>
..... 94, 104, 144, 146, 153, 220–222,	135
225–228, 240, 247, 353, 358, 359, 361–365	<code>\glstentryuserii</code>
<code>\glstentrylongpluralaccess</code>	134, 135
343	<code>\Glstentryuseriii</code>
<code>\Glstentryname</code>	136
130, 204, 356	<code>\glstentryuseriii</code>
<code>\glstentryname</code>	135, 136
130, 311, 356	<code>\Glstentryuseriv</code>
<code>\glstentrynumberlist</code>	136
153, 172	<code>\glstentryuseriv</code>
<code>\Glstentryplural</code> 97, 100, 128, 345, 349	136, 137
<code>\glstentryplural</code>	<code>\Glstentryuserv</code>
..... 96, 97, 100, 128, 344, 345, 348, 349	137
<code>\glstentrypluralaccess</code>	<code>\glstentryuserv</code>
342	137
<code>\Glstentryprefix</code>	<code>\Glstentryuservi</code>
260	138
<code>\glstentryprefix</code>	<code>\glstentryuservi</code>
258, 261	138
<code>\Glstentryprefixfirst</code>	<code>\glstfieldfetch</code>
260	150
<code>\glstentryprefixfirst</code>	<code>\glstfirstaccessdisplay</code> 346, 347, 350
259, 261	<code>\glstfirstpluralaccessdisplay</code>
<code>\Glstentryprefixfirstplural</code> 344, 345, 348, 349
261	<code>\glstfirstpluralacessdisplay</code>
<code>\glstentryprefixfirstplural</code> 259, 262	349
<code>\Glstentryprefixplural</code>	<code>\glstgenacfmt</code> 220, 221, 227, 357, 358, 363
260	<code>\glstgenentryfmt</code>
<code>\glstentryprefixplural</code> 220, 221, 226, 227, 232, 234, 236–
259, 262	238, 240, 242, 245, 247, 357, 358, 362, 363
<code>\glstgetgrouptitle</code>	<code>\glstglossarymark</code>
265, 269, 270, 288–293, 308–311, 314, 315	39

<code>\glsgroupheading</code>	161, 195, 266, 268–270, 272, 273, 275, 282, 284, 286, 288–295, 297, 300, 302, 304, 307–312, 314, 315, 321	<code>\glslistgroupheaderfmt</code>	269, 270
<code>\glsgroupskip</code>	160, 161, 195, 267, 268, 272, 273, 275, 278, 279, 283, 284, 286, 294, 296, 297, 301, 303, 304, 308, 309, 311, 314, 320	<code>\glslistnavigationitem</code>	269, 270
<code>\glshyperfirstfalse</code>	227, 363	<code>\glslocalreset</code>	89
<code>\glshyperfirsttrue</code>	25	<code>\glslocalunset</code>	90, 121–126
<code>\glshyperlink</code>	183	<code>\glslongaccessdisplay</code>	353, 355–367
<code>\glshypernavsep</code>	265	<code>\glslongkey</code>	371
<code>\glshypernumber</code>	37, 212	<code>\glslongpluralaccessdisplay</code>	353, 358, 359, 361–365, 367
<code>\glsifhyperon</code>	107	<code>\glslongpluralkey</code>	371
<code>\glsIfListOfAcronyms</code>	16	<code>\glslongtok</code>	217, 218, 220, 221, 226, 227, 232–241, 243–248, 357, 358, 362, 363, 366–371
<code>\glsifplural</code>	96, 100, 102, 103, 121–125, 139–145, 225, 234, 238, 240, 242, 344, 348, 351, 352, 354–356, 361	<code>\glsLTpenaltycheck</code>	281
<code>\glsifusetranslator</code>	23–25, 34, 374	<code>\glsmccols</code>	288–292
<code>\glsindexonlyfirstfalse</code>	25	<code>\glsnameaccessdisplay</code>	356, 357
<code>\glsinlinedescformat</code>	266, 323	<code>\glsnamefont</code>	204, 205, 336, 337, 356
<code>\glsinlinedopostchild</code>	266, 323	<code>\glsnavhyperlink</code>	265
<code>\glsinlineemptydescformat</code>	266, 323	<code>\glsnavhyperlinkname</code>	263
<code>\glsinlinenameformat</code>	266, 323	<code>\glsnavhypertarget</code>	269, 270, 288–293, 308, 310, 311, 315
<code>\glsinlineparentchildseparator</code>	266, 323	<code>\glsnavigation</code>	269, 270, 288–292, 308, 309, 311, 314
<code>\glsinlinepostchild</code>	266, 323	<code>\glsnextpages</code>	8, 64, 186
<code>\glsinlineseparator</code>	266, 323	<code>\glsnogroupskipfalse</code>	10
<code>\glsinlinesubdescformat</code>	266, 323	<code>\glsnoidxdisplayloc</code>	173, 174
<code>\glsinlinesubnameformat</code>	266, 323	<code>\glsnoidxdisplaylocclishandler</code>	173
<code>\glsinlinesubseparator</code>	266, 323	<code>\glsnoidxloclist</code>	172, 195, 196
<code>\glsinsert</code>	96–103, 121–125, 139–145, 225, 226, 234, 237, 238, 240, 242, 243, 344–356, 361, 362	<code>\glsnoidxloclisthandler</code>	196
<code>\glskeylisttok</code>	217, 218, 232–239, 241, 243–246, 248, 366–371	<code>\glsnoidxnumberlistloophandler</code>	173
<code>\glslabel</code>	79, 96–103, 108–110, 139–145, 220, 221, 225–227, 233, 234, 237, 238, 240, 242, 243, 344–353, 357, 358, 361–363	<code>\glsnoidxstripaccents</code>	20
<code>\glslabeltok</code>	217, 218, 232–241, 243–248, 367–370	<code>\glsnomakeindexwarning</code>	163
<code>\glslink</code>	218, 226, 228, 233, 362, 364	<code>\glsnonextpages</code>	64, 185
<code>\glslink options</code>		<code>\glsnopostdotfalse</code>	10
counter	106, 120, 255	<code>\glsnoxindywarning</code>	37, 43–45, 48–50, 157
format	106, 120, 210	<code>\glsnumberformat</code>	154
hyper	106, 108, 109, 120	<code>\glsnumberlistloop</code>	173
local	106	<code>\glsnumbersgroupname</code>	29, 35, 208
<code>\glslinkcheckfirsthyperhook</code>	108	<code>\glsnumlistlastsep</code>	154, 173
<code>\glslinkpostsetkeys</code>	109	<code>\glsnumlistparser</code>	154, 170
<code>\glslinkvar</code>	107	<code>\glsnumlistsep</code>	154, 173
<code>\glslistdottedwidth</code>	270, 324	<code>\glsopenbrace</code>	47, 160, 161, 320, 321
		<code>\glsorder</code>	26, 167–169, 191
		<code>\glsorg@endtheglossary</code>	5
		<code>\glsorg@PrintChanges</code>	5
		<code>\glsorg@theglossary</code>	5
		<code>\glspagelistwidth</code>	273, 274, 276, 277, 279, 280, 284–287, 295–297, 299, 300, 302–305
		<code>\glspatchLToutput</code>	277–280
		<code>\glspenaltygroupskip</code>	278, 279

<code>\glsperscentchar</code>	70, 160, 162	
<code>\Glspl</code>	95, 231	283, 284, 286, 294, 295, 297, 301, 302, 304, 307, 309, 310, 313, 323–330, 332–335
<code>\glspl</code>	94, 231	<code>\glssymbolaccessdisplay</code>
<code>\glspluralaccessdisplay</code>	344, 345, 348, 349	346–348, 357
<code>\glspluralsuffix</code>		<code>\glssymbolpluralaccessdisplay</code> .
.....	32, 82, 220–222, 358, 359, 363–365	344, 345
<code>\glspostdescription</code>		<code>\glssymbolsgroupname</code>
..	35, 267–269, 272, 282, 283, 294, 301, 307–310, 313, 314, 323–326, 328–332, 334	29, 35, 208
<code>\glspostinline</code>	265	<code>\glstarget</code>
<code>\glspostlinkhook</code>		206, 267–273, 275, 282–284, 286, 294–297, 301, 302, 304, 307–310, 312, 314, 323–335
.....	108, 121–126, 139–146, 354–356	<code>\glstextaccessdisplay</code> ..
<code>\glsprestandardsort</code>	11	346, 347, 349, 350
<code>\glreset</code>	89	<code>\glstextformat</code>
<code>\glresetentrycounter</code>	199, 201	108, 110
<code>\glresetentrylist</code>	160, 193, 200, 320	<code>\glstextup</code>
<code>\glresetsubentrycounter</code>		33, 365
.....	199, 200, 203, 266, 323	<code>\glstildechar</code>
<code>\glssanitizesortfalse</code>	22	43, 160, 161
<code>\glssanitizesorttrue</code>	22	<code>\glstranslatefalse</code>
<code>\glssavenumberlistfalse</code>	8	24, 25
<code>\glssavewritesfalse</code>	28	<code>\glstranslatetrue</code>
<code>\glseeformat</code>	159, 171, 173, 320	24, 25
<code>\glseeitem</code>	183	<code>\glstreechildpredesc</code>
<code>\glseeitemformat</code>	183	308, 309
<code>\glseeelastsep</code>	183	<code>\glstreegroupheaderfmt</code>
<code>\glseeelist</code>	182
<code>\glseeesep</code>	183	288–293, 308–311, 314, 315
<code>\glssetexpandfield</code>	19, 21–23	<code>\glstreeindent</code> ..
<code>\glssetnoexpandfield</code>	19, 21–23	309, 310, 312–314, 329–331
<code>\GlsSetQuote</code>	79, 162	<code>\glstreeitem</code>
<code>\glsettocitle</code>	34, 185	288, 289, 306, 307
<code>\glsshortaccessdisplay</code>		<code>\glstreenamebox</code>
.....	351–354, 357–360, 362–367	312, 314
<code>\glsshortkey</code>	371	<code>\glstreenamefmt</code>
<code>\glsshortpluralaccessdisplay</code>		306–314
.....	351, 353, 358, 362–365, 367	<code>\glstreenavigationfmt</code>
<code>\glsshortpluralkey</code>	371
<code>\glsshorttok</code>		288–292, 308, 309, 311, 314
.....	217, 218, 232–241, 243–248, 367–371	<code>\glstreepredesc</code>
<code>\glssortnumberfmt</code>	12, 13	307, 309, 310
<code>\glsspace</code>	214	<code>\glstreesubitem</code>
<code>\glsstepeentry</code>	199, 203	288, 307
<code>\glsstepeentry</code>	199, 200, 203	<code>\glstreesubsubitem</code>
<code>\glssubentrycounterfalse</code>	10	288, 307
<code>\glssubentrycounterlabel</code> ..	199, 200, 203	<code>\glstype</code> .
<code>\glssubentryitem</code>		108, 109, 121–125, 139–146, 354–356
. 199, 200, 266, 268–270, 272, 273, 275,		<code>\glsucmarkfalse</code>
		10
		<code>\glsucmarktrue</code>
		10
		<code>\glunset</code>
		87, 90–92, 121–126
		<code>\glsupacrpluralsuffix</code>
	
		222, 223, 229, 235, 239, 242, 244
		<code>\GlsUseAcrEntryDispStyle</code> ...
		219, 222–
		225, 227, 229, 230, 359, 360, 363, 365, 366
		<code>\GlsUseAcrStyleDefs</code>
		219, 222–
		225, 227, 229, 230, 359, 360, 363, 365, 366
		<code>\glswrallowprimitivemodstrue</code>
		178
		<code>\glswrite</code>
		157–163, 169, 175, 318–322
		<code>\glswritedefhook</code>
		70
		<code>\glswriteentry</code>
		177
		<code>\glswritefiles</code>
		28, 175
		<code>\glsxindyfalse</code>
		26
		<code>\glsxindytrue</code>
		27
H		
<code>\H</code>	21	
<code>\hangindent</code>	292,	
	293, 306, 309, 310, 312, 314, 315, 328–331	
<code>\hbox</code>	87, 270, 324	

<code>\hfill</code>	270, 324	<code>\ifgls@xindy@glsnumbers</code>	50
<code>\hline</code> ...	272–274, 276, 283–285, 287, 294–305	<code>\ifglsacrdescription</code>	246
<code>\hsize</code>	271, 282, 293, 300	<code>\ifglsacrdua</code>	236, 242, 245–247
<code>\hspace</code>	306	<code>\ifglsacrfootnote</code>	108, 246, 247
<code>\hss</code>	270, 324	<code>\ifglsacronym</code>	14, 15
<code>\ht</code>	281	<code>\ifglsacrshortcuts</code>	30, 231
<code>\hyperdef</code>	31	<code>\ifglsacrsmalldcaps</code>	
<code>\hyperlink</code>	106, 119, 211	235, 237, 239, 241, 244, 245
<code>hyperref</code> package	181, 184, 210, 255	<code>\ifglsacrsmaller</code>	235, 237, 239, 242
<code>\hypertarget</code>	119	<code>\ifglsautomake</code>	27, 170
I			
<code>\IeC</code>	21	<code>\ifglsdescsuppressed</code>	266
<code>\if</code>	112, 180, 317	<code>\ifglsentrycounter</code>	198, 199, 201–203
<code>\if@endfor</code>	264	<code>\ifglsentryexists</code>	52, 53, 69, 70, 79, 81
<code>\if@gls@debug</code>	5, 18, 176	<code>\ifglshaschildren</code>	266, 323
<code>\if@gls@docloaded</code>	4, 14, 176	<code>\ifglshasdesc</code>	266
<code>\if@glssisacronymlist</code>	108	<code>\ifglshaslong</code>	93–95, 147,
<code>\if@openright</code>	41	220, 221, 225, 227, 240, 357, 358, 361, 363	
<code>\ifbool</code>	15, 25, 28, 52, 96–98	<code>\ifglshasparent</code>	189, 195, 311
<code>\ifboolexpr</code>	33, 58, 207	<code>\ifglshasprefix</code>	260
<code>\ifcase</code>	6, 7, 24, 64, 198, 307, 328	<code>\ifglshasprefixfirst</code>	260
<code>\ifcsdef</code>	24, 34, 41, 67, 68,	<code>\ifglshasprefixfirstplural</code>	260
73–78, 105, 177, 188, 189, 191, 193, 209, 219		<code>\ifglshasprefixplural</code>	260
<code>\ifcseempty</code>	54, 258	<code>\ifglshassymbol</code>	
<code>\ifcsequal</code>	54	234, 238, 242, 307–310, 313, 314
<code>\ifcsstrequal</code>	78	<code>\ifglshyperfirst</code>	108
<code>\ifcsstring</code>	77	<code>\ifglsindeonlyfirst</code>	177
<code>\ifcsundef</code>	6, 12, 27, 31, 34, 37, 39,	<code>\ifglsnogroupskip</code>	268, 272,
41, 51, 52, 59, 61, 63, 80, 82, 83, 91, 106,		273, 275, 278, 279, 283, 284, 286, 294,	
110, 111, 119, 167, 175, 184, 187, 189,		296, 297, 301, 303, 304, 308, 309, 311, 314	
197, 198, 203, 207–210, 213, 219, 264, 322		<code>\ifglslonnumberlist</code>	200
<code>\ifdef</code>	55, 56,	<code>\ifglslonopostdot</code>	9
65, 70, 87, 106, 146, 150, 172, 173, 213, 306		<code>\ifglslnumberline</code>	41
<code>\ifdefempty</code>	17, 40, 51, 55, 56, 60,	<code>\ifglssanitizesort</code>	20, 22
61, 96, 100, 102, 170, 194, 195, 217, 219,		<code>\ifglssavenumberlist</code>	66, 170, 183
225, 233, 237, 240, 242, 344, 348, 351, 361		<code>\ifglssavewrites</code>	28, 167, 176
<code>\ifdefequal</code>	54–56, 67, 68, 71, 81, 85, 195	<code>\ifglssubentrycounter</code>	199, 201–203
<code>\ifdefstrequal</code>	77	<code>\ifglstoc</code>	41
<code>\ifdefstring</code>		<code>\ifglstranslate</code>	33
. 9, 33, 34, 58, 167–169, 191, 192, 194, 196		<code>\ifglsucmark</code>	39
<code>\ifdefvoid</code>	20, 84, 195	<code>\ifglused</code>	93, 96–102, 108, 156,
<code>\ifdim</code>	221, 281, 311	177, 233, 237, 240, 242, 258–262, 344–351	
<code>\iffalse</code>	84, 89	<code>\ifglswrallowprimitivemods</code>	179
<code>\IfFileExists</code>	9, 23, 24, 187	<code>\ifglsxindy</code> 35–37, 42–45, 47–50, 60, 85, 86,	
<code>\ifglossaryexists</code>	38, 49, 53, 167, 168	113, 157, 163, 167, 180, 181, 187, 316–318	
<code>\ifgls@sanitize@description</code>	21	<code>\ifignoredglossary</code>	80, 84, 177
<code>\ifgls@sanitize@name</code>	22	<code>\ifin@</code>	30
<code>\ifgls@sanitize@symbol</code>	22	<code>\ifinlistcs</code>	193, 197
		<code>\ifKV@glslink@hyper</code>	109, 110
		<code>\ifKV@glslink@local</code>	121–126

<code>\ifmeasuring@</code>	87	186, 195, 197, 201, 205, 217, 230–232,
<code>\ifnum</code>	11, 12, 91, 92,	234–246, 256, 264, 265, 281, 288, 289,
	194, 280, 281, 309, 310, 312, 313, 329, 330	306, 307, 322, 339, 354–356, 367–371, 374
<code>\ifstrempy</code>	323	<code>\letcs</code> 54–56, 70, 72, 73, 77, 82, 146, 147, 167,
<code>\ifstrequal</code>	207	172, 173, 188, 190, 191, 195, 204, 207, 312
<code>\ifthenelse</code>	23, 31, 41, 110, 169, 208, 247, 264	link text
<code>\IfTrackedLanguage</code>	163	95
<code>\IfTrackedLanguageFileExists</code>	34, 374, 375	<code>\listcsadd</code>
<code>\iftrue</code>	84, 89	193
<code>\ifundef</code>	58, 69, 80, 157, 162, 169, 199	<code>\listcsgadd</code>
<code>\ifvmode</code>	155	197
<code>\ifvoid</code>	281	<code>\listcsxadd</code>
<code>\ifx</code>	11, 13, 15, 16, 30, 31, 42, 44, 46,	188, 189
	47, 50, 51, 81–83, 86, 110, 111, 113–118,	<code>\listead</code>
	147, 158, 161, 163–166, 175, 178, 179,	192
	181, 182, 185, 198, 201, 208–211, 233,	<code>\loadglsentries</code>
	235, 237, 239, 241, 242, 244, 246, 248,	95
	249, 318–320, 322, 323, 328–331, 336, 342	<code>\long</code>
<code>\immediate</code>	69, 70, 93, 167, 175, 187	78, 211
<code>\in@</code>	30	<code>\longnewglossaryentry</code>
<code>\index</code>	176	79
<code>\indexname</code>	29	longtable package
<code>\indexspace</code>	268, 288–293, 308–311, 314, 315	271, 277, 282
<code>\input</code>	33, 95	<code>\LT@end@open</code>
<code>\inputencodingname</code>	27	281
<code>\InputIfFileExists</code>	70	<code>\LT@err</code>
<code>\istfilename</code>	35, 158, 162, 168, 169, 318, 321	281
<code>\item</code>	268–271, 288, 289, 307, 308, 323, 324, 328	<code>\LT@foot</code>
		281
		<code>\LT@head</code>
		281
		<code>\LT@lastfoot</code>
		281
		<code>\LT@output</code>
		281
		M
		<code>\makeatletter</code>
		70, 186
		<code>\makeatother</code>
		70
		<code>\makebox</code>
		270, 312, 314, 324, 330, 331
		makeglossaries
		26, 36, 49, 50, 58, 163, 169, 187
		<code>\makeglossaries</code>
		6, 28, 31, 64, 170, 172, 174, 188
		<code>\makeglossary</code>
		167, 169
		makeindex
		376
		makeindex
		11, 26, 27, 32,
		36–38, 42, 58, 60, 62, 86, 111, 114, 156,
		159, 162, 163, 166, 175, 180, 206, 317, 318
		<code>delim_n</code>
		37
		<code>delim_r</code>
		38
		<code>page_compositor</code>
		36
		<code>special characters</code>
		113, 156
		<code>\makenoidxglossaries</code>
		6, 64, 170, 174
		<code>\MakeTextUppercase</code>
		4
		<code>\MakeUppercase</code>
		345, 347, 354, 356
		<code>\markboth</code>
		39
		<code>\mbox</code>
		155, 269, 292, 293, 312, 324
		memoir class
		176
		<code>\memUHead</code>
		39
		<code>\MessageBreak</code>
		34, 58, 185, 336, 374, 375
		mfistuc package
		1
		<code>\mfistucMakeUppercase</code>
	
		4, 39, 75, 97, 99–103, 127–138,
		140, 142, 144, 146, 218, 225–228, 238,
		243, 261, 262, 349–353, 361, 362, 364, 365

\midrule 278, 279
 \month 158, 162, 318, 321
 multicol package 287

N

\n 162, 321
 \NeedsTeXFormat ... 4, 256, 316, 322, 336, 374
 \new@glossaryentry 69, 172
 \new@ifnextchar 58, 74, 75, 93,
 94, 120–124, 126–145, 214–216, 258–261
 \newacronym 213,
 217, 233, 235, 237, 239, 241, 244, 246, 248
 \newacronymhook 218,
 233, 235, 237, 239, 241, 244, 246, 248, 366
 \newacronymstyle ... 220–225, 227, 229, 230
 \newcommand 5–26, 28–33, 35–45,
 47–55, 57–79, 84–96, 100, 102, 104, 105,
 107–110, 112, 113, 119–157, 163, 166–
 168, 171, 174–186, 188–193, 195–197,
 200–210, 212–221, 230, 232–254, 257–
 261, 263–265, 267, 268, 280, 281, 288,
 306, 307, 312, 322, 340–342, 357, 371–373
 \newcount 12, 67
 \newcounter 198, 199, 201
 \newenvironment 203
 \newglossary 14, 15, 29, 59, 60, 169
 \newglossaryentry 29, 66, 69, 90, 218, 232,
 234, 236, 238, 240, 243, 245, 247, 367–370
 \newglossaryentry options
 access 339, 340
 counter 63
 description 25,
 61, 62, 66, 69, 79, 130, 148, 213, 241, 338
 descriptionaccess 341, 343
 descriptionplural 131, 338
 descriptionpluralaccess 341, 343
 first 62, 82, 120, 127, 149, 239, 244, 337
 firstaccess 341, 343
 firstplural 62, 128, 149, 338
 firstpluralaccess 341, 343
 format 158
 long 102, 152, 338
 longaccess 342, 343
 longplural 152, 338
 longpluralaccess 342, 343
 name ... 61, 62, 66, 69, 79, 129, 147, 183, 337
 nonumberlist 64, 65
 parent 64, 69
 plural 62, 82, 128, 338

pluralaccess 341, 342
 prefix 256
 prefixfirst 256
 prefixfirstplural 257
 prefixplural 257
 see 5, 8, 63, 69, 169, 171
 short 102, 152, 338
 shortaccess 341, 343
 shortplural 152, 338
 shortpluralaccess 341, 343
 sort 62, 150, 206
 symbol 61, 63,
 132, 235, 237, 239, 244, 274, 297, 337–339
 symbolaccess 341, 343
 symbolplural 133, 338
 symbolpluralaccess 341, 343
 text 62, 120, 126, 148, 235, 239, 337
 textaccess 341, 342
 type 14, 63, 95, 150
 user1 133, 150, 339
 user2 134, 151
 user3 135, 151
 user4 136, 151
 user5 137, 151
 user6 137, 152, 339
 \newglossarystyle
 ... 265, 268–280, 282–305, 307–312, 314
 \newif 4, 16, 23, 26, 178
 \newlength 119, 271, 282, 293, 300, 310
 \newrobustcmd
 ... 69, 74, 75, 93, 94, 108, 120–145,
 147–153, 155, 156, 213–216, 257–261, 311
 \newterm 29
 \newtoks 113, 167, 217
 \newwrite 69, 157, 162, 167, 169
 ngerman package 163
 \noalign 281
 \nobreak 269, 281, 324
 \noexpand 16, 31, 43, 44, 83, 84, 105,
 110–112, 118, 119, 154, 164–168, 170,
 178, 180, 183, 187, 190, 191, 204, 205,
 213, 218, 232, 234, 236, 238, 240, 241,
 243, 245, 247, 248, 316, 336, 337, 367–371
 \nohyperpage 210
 \noindent 206, 289–292, 309–311
 \noist 321, 322
 \nopostdesc 29, 35, 78, 186, 323
 \normalbaselineskip 280, 281
 \nr 6, 7, 24, 64, 198

<code>\ns@ACRfull</code>	215
<code>\ns@Acrfull</code>	215
<code>\ns@acrfull</code>	214
<code>\ns@ACRfullpl</code>	216
<code>\ns@Acrfullpl</code>	216
<code>\ns@acrfullpl</code>	215
<code>\ns@ACRlong</code>	143
<code>\ns@Acrlong</code>	142, 143
<code>\ns@acrlong</code>	142
<code>\ns@ACRlongpl</code>	145
<code>\ns@Acrlongpl</code>	144, 145
<code>\ns@acrlongpl</code>	144
<code>\ns@ACRshort</code>	139, 140
<code>\ns@Acrshort</code>	139
<code>\ns@acrshort</code>	138
<code>\ns@ACRshortpl</code>	141
<code>\ns@Acrshortpl</code>	141
<code>\ns@acrshortpl</code>	140
<code>\ns@newglossary</code>	58
<code>\null</code>	112–119, 164–166, 187
<code>\number</code>	12, 82, 92, 178, 179, 205, 337
<code>\numberline</code>	41
<code>\numexpr</code>	92

O

<code>\O</code>	21
<code>\o</code>	21
<code>\OE</code>	21
<code>\oe</code>	21
<code>\openout</code>	69, 158, 162, 167, 318, 321
<code>\OR</code>	247
<code>\or</code>	6, 7, 24, 198, 307, 328
<code>\org@glossaryentrynumbers</code>	185, 201
<code>\org@glossarytitle</code>	185
<code>\org@glspostdescription</code>	35
<code>\org@ifKV@glslink@hyper</code>	109, 110
<code>\orgAlph</code>	179
<code>\orgalph</code>	179
<code>\orgarabic</code>	179
<code>\orgnumber</code>	179
<code>\orgRoman</code>	179
<code>\orgromannumeral</code>	179
<code>\orgthe</code>	179
<code>\outputpenalty</code>	280, 281

P

<code>\p@</code>	268, 287, 306
<code>\p@glshyp@opt</code>	107
package options:	
acronym	14, 15, 32, 184, 213

true	15
counter	17
description	239, 240
dua	237, 239, 240
entrycounter	199, 201
true	10
footnote	121–125, 235, 237, 239, 241
hyperfirst	
false	121–125
indexonlyfirst	383
makeindex	160, 255
nogroupskip	272, 273, 275, 278, 279, 283, 284, 286, 294, 296, 297, 301, 303, 304
nolist	248
nolong	249, 271
nomain	14
nonumberlist	8
nosuper	249
notree	249
nowarn	5
numberline	6
sanitize	21, 61, 147, 148
sanitizesort	18
savewrites	28, 380
false	167
true	168, 175
section	6, 40
sort	
def	10, 11
standard	10
use	10, 11
style	7, 248, 249
subentrycounter	199, 201
toc	6
true	6
translate	24
false	24
translator	23
xindy	27, 160, 255
<code>\PackageError</code>	5, 6, 13, 27, 31, 43, 49, 52, 53, 58, 63, 66, 67, 72–78, 80–82, 90, 106, 146, 166, 167, 169, 172, 174, 191–193, 198, 200, 208, 209, 219, 220, 236, 237, 242, 245, 322, 344
<code>\PackageInfo</code>	5, 167, 176
<code>\PackageWarning</code>	5, 17
<code>\PackageWarningNoLine</code>	5, 18, 34, 374, 375
<code>\pagegoal</code>	281

<code>\pagelistname</code>	35, 274–276, 278, 279, 285–287, 296–299, 303–305
<code>\par</code>	35, 206, 268, 269, 287, 289–293, 306, 307, 309–315, 324, 329–331
<code>\parindent</code>	288–293, 307–310, 312–315, 329–331
<code>\parskip</code>	288–291, 307, 308, 310
<code>\PassOptionsToPackage</code>	256, 336
<code>\penalty</code>	281
<code>\phantomsection</code>	40
<code>polyglossia package</code>	23, 33
<code>\printglossaries</code>	170
<code>\printglossary</code>	15, 18, 29, 170, 184, 200
<code>\printglossary options</code>	
<code>entrycounter</code>	198
<code>nogroupskip</code>	198
<code>nonumberlist</code>	200
<code>noposted</code>	198
<code>numberedsection</code>	198
<code>style</code>	198
<code>subentrycounter</code>	199
<code>title</code>	197
<code>toctitle</code>	197
<code>type</code>	14, 183, 197
<code>\printindex</code>	29
<code>\printnoidxglossaries</code>	172
<code>\printnoidxglossary</code>	171, 172, 174, 184, 191, 192, 200
<code>\printnoidxglossary options</code>	
<code>sort</code>	200
<code>\printnumbers</code>	29
<code>\printsymbols</code>	29
<code>\ProcessOptions</code>	256, 336
<code>\ProcessOptionsX</code>	30
<code>\protect</code>	41, 104, 220–222, 227, 228, 234, 238, 240, 353, 357, 358, 363, 364
<code>\protected@edef</code>	7, 44, 46, 48, 50, 81, 84, 86, 96–98, 104, 111, 154, 176, 179, 198, 204, 205, 208, 209, 218, 242, 247, 257, 263, 317, 318, 336, 337, 342
<code>\protected@write</code>	57, 59, 158, 160, 169, 171, 174, 177, 183, 263, 318
<code>\protected@xdef</code>	11–13, 16, 20, 68, 86, 87, 179, 340
<code>\providecommand</code>	15, 32, 33, 39, 57, 93, 120, 160, 169, 171, 187, 204, 205, 268, 287, 306
<code>\ProvidesFile</code>	33
<code>\ProvidesPackage</code>	4, 256, 263, 265, 267, 271, 277, 282, 287, 293, 300, 306, 316, 322, 336, 374
R	
<code>\r</code>	21
<code>\raggedright</code>	280, 282–287, 300–305
<code>\raisebox</code>	119
<code>\ref</code>	202
<code>\refstepcounter</code>	199, 202
<code>\relax</code>	7, 9, 12–15, 24, 30, 45, 57, 62, 64, 68, 81, 83, 87, 91, 92, 107, 108, 110, 112–118, 147, 160–166, 169, 171–173, 177, 179, 180, 182, 185, 186, 194, 195, 197, 198, 207–209, 249, 264, 268, 280, 287, 292, 293, 306, 307, 309–315, 317, 320–322, 328–331, 339, 354, 355, 367–369, 371
<code>\renewacronymstyle</code> ..	357–361, 363, 365, 366
<code>\renewcommand</code>	4–10, 13–15, 17, 18, 22, 24–28, 30, 34–37, 50, 61, 63, 65, 78, 90–92, 154, 155, 157, 163, 164, 169–173, 176, 186, 188, 198–200, 217, 218, 220–230, 233, 235, 237, 239, 241, 242, 244, 246, 248, 266–276, 278, 279, 281–297, 300–304, 307–317, 322–330, 332–335, 339, 344, 348, 351, 353, 356–360, 362–370
<code>\renewenvironment</code> ..	203, 265, 268, 271–277, 279, 280, 282–305, 307, 308, 310, 312
<code>\RequireGlossariesLang</code>	34, 374, 375
<code>\RequirePackage</code>	4, 8, 9, 23, 24, 30, 33, 248, 255, 256, 271, 277, 282, 287, 293, 300, 337, 374
<code>\restorecounters@</code>	111
<code>\romannumeral</code>	178, 179, 312, 313, 330
S	
<code>\s@glshyp@opt</code>	107
<code>\s@newglossary</code>	58
<code>\savecounters@</code>	110
<code>\seename</code>	182
<code>\SetAcronymStyle</code>	25, 26
<code>\setbool</code>	22
<code>\setbox</code>	281
<code>\setcounter</code>	199, 201
<code>\SetCustomDisplayStyle</code>	248
<code>\SetDefaultAcronymDisplayStyle</code>	233
<code>\SetDefaultAcronymStyle</code>	246
<code>\SetDescriptionAcronymDisplayStyle</code> ..	239
<code>\SetDescriptionAcronymStyle</code>	246
<code>\SetDescriptionDUAAcronymDisplayStyle</code>	237
<code>\SetDescriptionDUAAcronymStyle</code>	246
<code>\SetDescriptionFootnoteAcronymDisplayStyle</code>	235

<code>\SetDescriptionFootnoteAcronymStyle</code>	246	<code>\symbolname</code> 35, 275, 276, 279, 286, 287, 298, 299, 305
<code>\SetDUADisplayStyle</code> 246		
<code>\SetDUASyle</code> 247		
<code>\setentrycounter</code> 44, 160, 197, 316		
<code>\SetFootnoteAcronymDisplayStyle</code>	... 241		
<code>\SetFootnoteAcronymStyle</code> 247		
<code>\SetGenericNewAcronym</code> 219		
<code>\setglossarystyle</code> 185, 209, 249, 269–280, 283, 285–292, 294–296, 298, 299, 301–303, 305, 308, 309, 311, 314		
<code>\setglossentrycompatibility</code>	... 198, 209		
<code>\setkeys</code>	.. 23, 27, 30, 40, 80, 109, 155, 185, 217, 233, 235, 237, 239, 241, 244, 246, 248		
<code>\setlength</code> 271, 282, 288–291, 293, 300, 307, 308, 310, 313, 314, 330, 331		
<code>\SetSmallAcronymDisplayStyle</code> 244		
<code>\SetSmallAcronymStyle</code> 247		
<code>\settoheight</code> 119		
<code>\settowidth</code> 221, 311–313, 330		
<code>\sfcode</code> 9		
<code>\show</code> 249–254, 372, 373		
<code>\SmallNewAcronymDef</code> 244		
<code>\space</code> 6, 28, 31, 43, 47, 48, 50, 64, 66, 90, 91, 93–95, 104, 105, 107, 153, 158–162, 166, 168– 170, 172, 174, 183, 185, 188, 199, 202, 209, 214, 220–230, 238, 242, 243, 265, 267–269, 272, 282, 283, 294, 301, 306– 310, 312–314, 316, 317, 319–321, 323– 326, 328–332, 334, 353, 357–360, 362–367		
<code>\spacefactor</code> 9		
<code>\SS</code> 21		
<code>\ss</code> 21		
<code>\string</code> 6, 13, 18, 28, 31, 42, 43, 46–48, 50, 57, 59, 64, 66, 70, 73, 74, 85, 86, 90, 91, 93–95, 105, 107, 111, 112, 114–116, 118, 153, 156–163, 165–167, 169–172, 174, 180–182, 185, 187, 188, 191, 192, 200, 205, 206, 209, 263, 316–322		
<code>\strut</code> 206, 268–270, 272, 273, 275, 283, 284, 286, 294, 296, 297, 301, 302, 304, 310, 323–327, 329, 332–335		
<code>\subglossentry</code> 85, 186, 195, 205, 206, 266, 268– 270, 272, 273, 275, 283, 284, 286, 294, 295, 297, 301, 302, 304, 307, 309, 310, 313		
<code>\subitem</code> 288, 306, 307, 328		
<code>\subsubitem</code> 288, 306–308, 328		
<code>supertabular package</code> 9, 249, 293, 300		
<code>\symbolname</code> 35, 275, 276, 279, 286, 287, 298, 299, 305		
		T	
<code>\t</code> 21		
<code>\tablehead</code> 293–305		
<code>\tabletail</code> 293–305		
<code>\tabularnewline</code>	. 272–276, 278, 279, 282– 287, 294–299, 301–305, 326, 327, 332, 333		
<code>\texorpdfstring</code> 150		
<code>\textbar</code> 265		
<code>\textbf</code> 206, 212, 306, 328–331		
<code>textcase package</code> 4		
<code>\textit</code> 212		
<code>\textmd</code> 212		
<code>\textrm</code> 212		
<code>\textsc</code> 212, 222, 223, 229, 235, 239, 242, 244, 365		
<code>\textsf</code> 212		
<code>\textsl</code> 212		
<code>\textsmaller</code> 222, 223, 229, 235, 239, 242, 244, 365		
<code>\texttt</code> 212		
<code>\textulc</code> 213		
<code>\textup</code> 212, 213		
<code>\TH</code> 21		
<code>\th</code> 21		
<code>\the</code> 31, 34, 43, 48, 50, 59, 113–118, 158, 162, 164–166, 175, 176, 179, 183, 194, 204, 205, 211, 218, 220, 221, 226, 227, 232, 234, 236, 238, 240, 241, 243, 245, 247, 248, 316, 318, 321, 337, 357, 358, 362, 363, 366–371		
<code>\the@numberlist</code> 154		
<code>\theglossary</code> 5		
<code>\theglossaryentry</code> 199, 201, 202		
<code>\theglossarysubentry</code> 199, 201, 202		
<code>\theglentrycounter</code>	110, 111, 179, 317, 318		
<code>\theH</code> 181		
<code>\theHglossaryentry</code> 199, 201		
<code>\theHglossarysubentry</code> 199, 201		
<code>\theHglentrycounter</code> 111, 179		
<code>\thesection</code> 31		
<code>\this@dialect</code> 34, 374, 375		
<code>\toks@</code> 31, 34, 43, 48, 50, 59, 113–118, 164–166, 183, 204, 205, 211, 316, 336, 337		
<code>\toprule</code> 278, 279		
<code>tracklang package</code> 33, 374		
<code>\trans@languages</code> 33		

<code>\translate</code>	34, 35	<code>\warn@noprintglossary</code> ..	169, 171, 172, 186
<code>\translatelet</code>	14, 15, 29	<code>\write</code>	70, 93,
<code>translator package</code>	14, 15, 24, 29, 33, 34, 184		158–163, 167, 168, 171, 175, 187, 318–322
<code>\TX@trial</code>	87	<code>\writeist</code>	166, 167, 322
<code>\typeout</code>	18		
U		X	
<code>\u</code>	21	<code>\x</code>	211
<code>\uccode</code>	194	<code>\xatlevel@</code>	110
<code>\undef</code>	65, 184	<code>\xcapitalisewords</code>	150
<code>\unskip</code>	78, 270, 324	<code>\xdef</code>	75, 81, 82, 84, 186, 264
<code>\unvbox</code>	281	<code>\xglaccsupp</code>	342
<code>\usedictionary</code>	34	<code>\xifinlistcs</code>	188, 189, 192
<code>\usepackage</code>	191, 192	<code>xindy</code>	376
V		<code>xindy</code>	11, 26, 27, 36, 42, 45, 47, 49, 50, 86, 117, 118, 157, 159, 175, 180, 187, 206, 255, 317
<code>\v</code>	21	<code>\xmakefirstuc</code>	97, 98, 104, 146, 148, 257
<code>\val</code>	6, 7, 24, 64, 198	<code>\xspace</code>	213
<code>\vbox</code>	281	<code>xspace package</code>	4, 212, 213
<code>\vsize</code>	281		
<code>\vskip</code>	268, 280, 281, 287, 306	Y	
<code>\vss</code>	281	<code>\year</code>	158, 162, 318, 321
W		Z	
<code>\warn@nomakeglossaries</code>	169, 171, 172	<code>\z@</code>	281