

Documented Code For glossaries v4.42

Nicola L.C. Talbot

Dickimaw Books

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

2019-01-06

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 v4.42: 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	36
1.4	Xindy	46
1.5	Loops and conditionals	55
1.6	Defining new glossaries	62
1.7	Defining new entries	66
1.8	Resetting and unsetting entry flags	92
1.9	Keeping Track of How Many Times an Entry Has Been Unset	95
1.10	Loading files containing glossary entries	100
1.11	Using glossary entries in the text	101
1.12	Adding an entry to the glossary without generating text	160
1.13	Creating associated files	162
1.14	Writing information to associated files	182
1.15	Glossary Entry Cross-References	191
1.16	Displaying the glossary	193
1.17	Acronyms	221
1.18	Predefined acronym styles	225
1.19	Predefined Glossary Styles	257
1.20	Debugging Commands	258
1.21	Compatibility with version 2.07 and below	263
2	Prefix Support (glossaries-prefix Code)	264
3	Glossary Styles	271
3.1	Glossary hyper-navigation definitions (glossary-hypernav package)	271
3.2	In-line Style (glossary-inline.sty)	273
3.3	List Style (glossary-list.sty)	276
3.4	Glossary Styles using longtable (the glossary-long package)	279
3.5	Glossary Styles using longtable and booktabs (the glossary-longbooktabs) package	285
3.6	Glossary Styles using longtable (the glossary-longragged package)	290
3.7	Glossary Styles using multicol (glossary-mcols.sty)	295
3.8	Glossary Styles using supertabular environment (glossary-super package)	301
3.9	Glossary Styles using supertabular environment (glossary-superragged package)	308
3.10	Tree Styles (glossary-tree.sty)	314

4 Backwards Compatibility	324
4.1 glossaries-compatible-207	324
4.2 glossaries-compatible-307	330
5 Accessibility Support (glossaries-accsupp Code)	344
5.1 Defining Replacement Text	345
5.2 Accessing Replacement Text	348
5.3 Displaying the Glossary	364
5.4 Acronyms	365
5.5 Debugging Commands	380
6 Multi-Lingual Support	382
6.1 Polyglossia Captions	382
Glossary	384
Change History	385
Index	409

1 Main Package Code

1.1 Package Definition

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

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries}[2019/01/06 v4.42 (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. This is now a choice key.

```
32 \newif\if@gls@debug
33 \define@choicekey{glossaries.sty}{debug}[\gls@debug@val\gls@debug@nr]%
34 {true,false,showtargets}[true]{%
35   \ifcase\gls@debug@nr\relax
36     \@gls@debugtrue
37     \renewcommand*\GlossariesWarning}[1]{%
38       \PackageWarning{glossaries}{##1}%
39     }%
40     \renewcommand*\GlossariesWarningNoLine}[1]{%
41       \PackageWarningNoLine{glossaries}{##1}%
42     }%
43     \let\@glsshowtarget\@gobble
44     \PackageInfo{glossaries}{debug mode ON (nowarn option disabled)}%
45   \or
46     \@gls@debugfalse
47     \let\@glsshowtarget\@gobble
48     \PackageInfo{glossaries}{debug mode OFF}%
49   \or
50     \@gls@debugtrue
51     \renewcommand*\GlossariesWarning}[1]{%
```

```

52     \PackageWarning{glossaries}{##1}%
53   }%
54   \renewcommand*{\GlossariesWarningNoLine}[1]{%
55     \PackageWarningNoLine{glossaries}{##1}%
56   }%
57   \PackageInfo{glossaries}{debug mode ON (nowarn option disabled)}%
58   \renewcommand{\@glsshowtarget}{\glsshowtarget}%
59   \fi
60 }

```

`\glsshowtarget` If `debug=showtargets`, show the hyperlink target name in the margin.

```

61 \newcommand*{\glsshowtarget}[1]{%
62   \ifmode
63     \nfss@text{\ttfamily\small [#1]}%
64   \else
65     \ifinner
66       \texttt{\small [#1]}%
67     \else
68       \marginpar{\texttt{\small #1}}%
69     \fi
70   \fi
71 }

```

`\@glsshowtarget` `debug=showtargets` will redefine this.

```

72 \newcommand*{\@glsshowtarget}[1]{

```

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

`gls@see@noindex`

```

73 \newcommand*{\@gls@see@noindex}{%
74   \PackageError{glossaries}%
75   {'\gls@xr@key' key may only be used after \string\makeglossaries\space
76   or \string\makenoidxglossaries\space (or move
77   \string\newglossaryentry\space
78   definitions into the preamble)}%
79   {You must use \string\makeglossaries\space
80   or \string\makenoidxglossaries\space before defining
81   any entries that have a '\gls@xr@key' key. It may
82   be that the 'see' key has been written to the .glsdefs
83   file from the previous run, in which case you need to
84   move your definitions
85   to the preamble if you don't want to use
86   \string\makeglossaries\space
87   or \string\makenoidxglossaries}%
88 }

```

`seenoinde`

```

89 \define@choicekey{glossaries.sty}{seenoinde}%

```

```

90 [\gls@seenoinde@val\gls@seenoinde@nr]{error,warn,ignore}{%
91 \ifcase\gls@seenoinde@nr
92   \renewcommand*\@gls@see@noindex}{%
93     \PackageError{glossaries}%
94     {'\gls@xr@key' key may only be used after \string\makeglossaries\space
95     or \string\makenoidxglossaries}%
96     {You must use \string\makeglossaries\space
97     or \string\makenoidxglossaries\space before defining
98     any entries that have a '\gls@xr@key' key}%
99   }%
100 \or
101   \renewcommand*\@gls@see@noindex}{%
102     \GlossariesWarning{'\gls@xr@key' key ignored}%
103   }%
104 \or
105   \renewcommand*\@gls@see@noindex}{}%
106 \fi
107 }

```

`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.

```
108 \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`.

```
109 \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`.

```

110 \ifcsundef{chapter}%
111   {\newcommand*\@glossarysec}{section}}%
112   {\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`.

```

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

```

Determine whether or not to use numbered sections.

`glossarysecstar`

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

`lossaryseclabel`

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

`\glsautoprefix` Prefix to add before label if automatically generated:

```
118 \newcommand*\glsautoprefix{}
```

`numberedsection`

```
119 \define@choicekey{glossaries.sty}{numberedsection}%
120 [\gls@numberedsection@val\gls@numberedsection@nr]{%
121 false,nolabel,autolabel,nameref}[nolabel]{%
122 \ifcase\gls@numberedsection@nr\relax
123 \renewcommand*\@@glossarysecstar{*}%
124 \renewcommand*\@@glossaryseclabel{}%
125 \or
126 \renewcommand*\@@glossarysecstar{}%
127 \renewcommand*\@@glossaryseclabel{}%
128 \or
129 \renewcommand*\@@glossarysecstar{}%
130 \renewcommand*\@@glossaryseclabel{%
131 \label{\glsautoprefix\@glo@type}}%
132 \or
133 \renewcommand*\@@glossarysecstar{*}%
134 \renewcommand*\@@glossaryseclabel{%
135 \protected@edef\@currentlabelname{\glossarytoctitle}%
136 \label{\glsautoprefix\@glo@type}}%
137 \fi
138 }
```

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`

```
139 \@ifpackageloaded{classicthesis}
140 {\newcommand*\@glossary@default@style{index}}
141 {\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`.

```
142 \define@key{glossaries.sty}{style}{%
143 \def\@glossary@default@style{#1}%
144 }
```

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`


```

145 \newcommand*{\@gls@declareoption}[2]{%
146   \DeclareOptionX{#1}{#2}%
147   \DeclareOption{#1}{#2}%
148 }

```

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

```

149 \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).

```

150 \@gls@declareoption{nonumberlist}{%
151   \renewcommand*{\glossaryentrynumbers}[1]{\gls@save@numberlist{#1}}%
152 }

```

`savenumberlist` Provide means to store the number list for entries.

```

153 \define@boolkey{glossaries.sty}[gls]{savenumberlist}[true]{%
154 \glssavenumberlistfalse

```

eautionumberlist

```

155 \newcommand*\@glo@seeautionumberlist{}

```

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

```

156 \@gls@declareoption{seeautionumberlist}{%
157   \renewcommand*\@glo@seeautionumberlist}{%
158     \def\@glo@prefix{\glsnextpages}%
159   }%
160 }

```

`esclocations` When using `makeindex` or `xindy`, the locations may need to be adjusted to ensure they’re in a format that’s allowed by the indexing application. This involves a bit of hackery and isn’t needed if the locations are all guaranteed to be in the correct form (or if the user is prepared to post-process the glossary file before calling the relevant indexing application) so `esclocations=false` will switch off this mechanism allowing for a faster and more stable approach.

```

161 \define@boolkey{glossaries.sty}[gls]{esclocations}[true]{%
162 \glsclocationstrue

```

\@gls@loadlong

```

163 \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.

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

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

```
165 \IfFileExists{supertabular.sty}{%
166 \newcommand*\@gls@loadsuper{\RequirePackage{glossary-super}}}{%
167 \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.

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

`\@gls@loadlist`

```
169 \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`.

```
170 \@gls@declareoption{nolist}{%
171 \renewcommand*\@gls@loadlist}{%
172 \ifdefstring{\@glossary@default@style}{list}%
173 {\let\@glossary@default@style\relax}%
174 }%
175 }%
176 }
```

`\@gls@loadtree`

```
177 \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.

```
178 \@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).

```
179 \@gls@declareoption{nostyles}{%
180 \renewcommand*\@gls@loadlong}{}%
181 \renewcommand*\@gls@loadsuper}{}%
182 \renewcommand*\@gls@loadlist}{}%
183 \renewcommand*\@gls@loadtree}{}%
184 \let\@glossary@default@style\relax
185 }
```

`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.)

```

186 \newcommand*{\glspostdescription}{%
187   \ifglstopostdot\else.\spacefactor\sfcode'\. \fi
188 }

```

`nopostdot` Boolean option to suppress post description dot

```

189 \define@boolkey{glossaries.sty}[gls]{nopostdot}[true]{}
190 \glsnopostdotfalse

```

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

```

191 \define@boolkey{glossaries.sty}[gls]{nogroupskip}[true]{}
192 \glsnogroupskipfalse

```

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

```

193 \define@boolkey{glossaries.sty}[gls]{ucmark}[true]{}

194 \@ifclassloaded{memoir}
195 {%
196   \glsucmarktrue
197 }%
198 {%
199   \glsucmarkfalse
200 }

```

`glossaryentry` If the `entrycounter` package option has been used, define a counter to number each level 0 entry. This is now defined by an internal command for consistency.

`aryentrycounter`

```

201 \newcommand*{\@gls@define@glossaryentrycounter}{%
202   \ifglsentrycounter
203     Define the glossaryentry counter if it doesn't already exist.
204     {%
205       \ifx\@gls@counterwithin\@empty
206         \newcounter{glossaryentry}%
207       \else
208         \newcounter{glossaryentry}[\@gls@counterwithin]%
209       \fi
210       \def\theHglossaryentry{\currentglossary.\theglossaryentry}%
211     }%
212   }%
213 \fi
214 }

```

`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`.

```

215 \define@boolkey{glossaries.sty}[gls]{entrycounter}[true]{}
216 \glstentrycounterfalse

```

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

```

217 \define@key{glossaries.sty}{counterwithin}{%
218   \renewcommand*{\@gls@counterwithin}{#1}%
219   \glstentrycountertrue
220   \@gls@define@glossaryentrycounter
221 }

```

`s@counterwithin` The default value is no parent counter:

```

222 \newcommand*{\@gls@counterwithin}{}

```

`glossarysubentry` If the `subentrycounter` package option has been used, define a counter to number each level 1 entry. This is now defined by an internal command for consistency.

`subentrycounter`

```

223 \newcommand{\@gls@define@glossarysubentrycounter}{%
  Check if counter already defined.
224   \ifundef\c@glossarysubentry
225   {%
226     \ifglssubentrycounter
227     \ifglstentrycounter
228     \newcounter{glossarysubentry}[glossaryentry]%
229     \else
230     \newcounter{glossarysubentry}%
231     \fi
  }%
  As with \theHglossaryentry, this starts with \currentglossary. to help avoid duplicate
  hyper targets.
232   \def\theHglossarysubentry{\currentglossary.\currentglssubentry.\theglossarysubentry}%
233   \fi
234 }%
235 {}%
236 }

```

`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`.

```

237 \define@boolkey{glossaries.sty}[gls]{subentrycounter}[true]{}
238 \glssubentrycounterfalse

```

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

```

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

```

sort Define the sort method: sort=standard (default), sort=def (order of definition) or sort=use (order of use). If no indexing required, use sort=none.

```
240 \define@choicekey{glossaries.sty}{sort}{standard,def,use,none}{%
241   \renewcommand*{\@glo@default@sorttype}{#1}%
242   \csname @gls@setupsort@#1\endcsname
243 }
```

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.

```
244 \newcommand*{\glsprestandardsort}[3]{%
245   \glsdosanitizesort
246 }
```

glscheck@sortallowed

```
247 \newcommand*{\@glo@check@sortallowed}[1]{}
```

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

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

Store entry information when it's defined.

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

No count register required for standard sort.

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

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

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

```
252     \ifx\@glo@sort\@glsdefaultsort
```

```
253       \let\@glo@sort\@glo@name
```

```
254     \fi
```

```
255     \let\glsdosanitizesort\@gls@sanitizesort
```

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

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

```
258   }%
```

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

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

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

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

```
261 }
```

Set standard sort as the default:

```
262 \@gls@setupsort@standard
```

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

```
263 \newcommand*\glssortnumberfmt[1]{%
264   \ifnum#1<100000 0\fi
265   \ifnum#1<10000 0\fi
266   \ifnum#1<1000 0\fi
267   \ifnum#1<100 0\fi
268   \ifnum#1<10 0\fi
269   \number#1%
270 }
```

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

```
271 \newcommand*\@gls@setupsort@def{%
```

Store entry information when it's defined.

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

Defined count register associated with the glossary.

```
273 \def\@gls@defsortcount##1{%
```

```
274   \expandafter\global
```

```
275   \expandafter\newcount\csname glossary@##1@sortcount\endcsname
```

```
276 }%
```

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

```
277 \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.

```
278 \ifcsundef{glossary@##1@sortcount}%
```

```
279   {\@gls@defsortcount{##1}}%
```

```
280   {}%
```

```
281   \expandafter\global\expandafter
```

```
282   \advance\csname glossary@##1@sortcount\endcsname by 1\relax
```

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

```
284     \expandafter\glssortnumberfmt
```

```
285     {\csname glossary@##1@sortcount\endcsname}}%
```

```
286 }%
```

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

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

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

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

```
289 }
```

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

```
290 \newcommand*\@gls@setupsort@use{%
```

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

```
291 \let\do@glo@storeentry\@gobble
```

Defined count register associated with the glossary.

```
292 \def\@gls@defsortcount##1{%
293   \expandafter\global
294   \expandafter\newcount\csname glossary@##1@sortcount\endcsname
295 }%
```

Initialise the sort key to empty.

```
296 \def\@gls@defsort##1##2{%
297   \expandafter\gdef\csname glo@##2@sort\endcsname{%}
298 }%
```

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

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

Get the parent, if one exists

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

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

```
301   \ifx\@glo@parent\@empty
302   \else
303     \expandafter\@gls@setsort\expandafter{\@glo@parent}%
304   \fi
```

Set index information for this entry

```
305   \edef\@glo@type{\csname glo@##1@type\endcsname}%
306   \edef\@gls@tmp{\csname glo@##1@sort\endcsname}%
307   \ifx\@gls@tmp\@empty
308     \expandafter\global\expandafter
309     \advance\csname glossary@\@glo@type @sortcount\endcsname by 1\relax
310     \expandafter\protected\xdef\csname glo@##1@sort\endcsname{%
311       \expandafter\gls@sortnumberfmt
312       {\csname glossary@\@glo@type @sortcount\endcsname}}%
313     \@glo@storeentry{##1}%
314   \fi
315 }%
```

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

```
316 \let\@glo@check@sortallowed\@gobble
317 }
```

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

```
318 \newcommand*\@gls@setupsort@none{%
```

Don't store entry index information.

```
319 \def\do@glo@storeentry##1{%
```

No count register required for standard sort.

```
320 \def\@gls@defsortcount##1{%
```

Don't modify sort value.

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

```

322 \expandafter\global\expandafter\let\csname glo###2@sort\endcsname\@glo@sort
323 }%

```

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

```

324 \def\@gls@setsort##1{}%

```

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

```

325 \renewcommand\@glo@check@sortallowed[1]{\PackageError{glossaries}
326 {Option sort=none not allowed with \string##1}%
327 {(Use sort=def instead)}}%
328 }

```

`\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.)

```

329 \newcommand*\glsdefmain{%
330 \if@gls@docloaded
331 \newglossary[glg2]{main}{gls2}{glo2}{\glossaryname}%
332 \else
333 \newglossary{main}{gls}{glo}{\glossaryname}%
334 \fi

```

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

```

335 \newcommand*\gls@tr@set@main@toctitle{%
336 \translatelet{\glossarytoctitle}{Glossary}%
337 }%
338 }

```

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`

```

339 \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`

```

340 \newcommand*\acronymtype{\glsdefaulttype}

```

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

```

341 \@gls@declareoption{nomain}{%
342 \let\glsdefaulttype\relax
343 \renewcommand*\glsdefmain}{}%
344 }

```


`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.

```
345 \define@boolkey{glossaries.sty}[gls]{acronym}[true]{%
346   \ifglsacronym
347     \renewcommand{\@gls@do@acronymsdef}{%
348       \DeclareAcronymList{acronym}%
349       \newglossary[alg]{acronym}{acr}{acn}{\acronymname}%
350       \renewcommand*\@acronymtype}{acronym}%
```

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

```
351     \newcommand*\@gls@tr@set@acronym@toctitle}{%
352       \translatelet{\glossarytoctitle}{Acronyms}%
353     }%
354   }%
355 \else
356   \let\@gls@do@acronymsdef\relax
357 \fi
358 }
```

`\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.

```
359 \AtBeginDocument{%
360   \ifglsacronym
361     \ifbool{glscompatible-3.07}{%
362       {}%
363     }{%
364       \providecommand*\printacronyms[1][]{%
365         \printglossary[type=\acronymtype,#1]}%
366     }%
367 \fi
368 }
```

`@do@acronymsdef` Set default value

```
369 \newcommand*\@gls@do@acronymsdef{}
```

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

```
370 \@gls@declareoption{acronyms}{%
371   \glsacronymtrue
372   \renewcommand{\@gls@do@acronymsdef}{%
373     \DeclareAcronymList{acronym}%
374     \newglossary[alg]{acronym}{acr}{acn}{\acronymname}%
375     \renewcommand*\@acronymtype}{acronym}%
```

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

```
376     \newcommand*\@gls@tr@set@acronym@toctitle}{%
377       \translatelet{\glossarytoctitle}{Acronyms}%
378     }%
379   }%
380 }
```

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

```
381 \newcommand*{\@glsacronymlists}{}
```

`addtoacronymlists`

```
382 \newcommand*{\@addtoacronymlists}[1]{%
383   \ifx\@glsacronymlists\@empty
384     \protected@xdef\@glsacronymlists{#1}%
385   \else
386     \protected@xdef\@glsacronymlists{\@glsacronymlists,#1}%
387   \fi
388 }
```

`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.)

```
389 \newcommand*{\DeclareAcronymList}[1]{%
390   \glsIfListOfAcronyms{#1}{-}{\@addtoacronymlists{#1}}%
391 }
```

`IfListOfAcronyms`

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

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

```
392 \newcommand{\glsIfListOfAcronyms}[1]{%
393   \edef\@do@gls@islistofacronyms{%
394     \noexpand\@gls@islistofacronyms{#1}{\@glsacronymlists}}%
395   \@do@gls@islistofacronyms
396 }
```

Internal command requires label and list to be expanded:

```
397 \newcommand{\@gls@islistofacronyms}[4]{%
398   \def\gls@islistofacronyms##1,#1,##2\end@gls@islistofacronyms{%
399     \def\@before{##1}\def\@after{##2}}%
400   \gls@islistofacronyms,#2,#1,\@nil\end@gls@islistofacronyms
401   \ifx\@after\@nnil
```

Not found

```
402   #4%
403   \else
```

Found

```
404   #3%
405   \fi
406 }
```

`glsisacronymlist` Convenient boolean.

```
407 \newif\if@glsisacronymlist
```

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

```
408 \newcommand*{\gls@ckisacronymlist}[1]{%
409   \glsIfListOfAcronyms{#1}%
410   {\@glsisacronymlisttrue}{\@glsisacronymlistfalse}%
411 }
```

`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.)

```
412 \newcommand*{\SetAcronymLists}[1]{%
413   \renewcommand*{\@glsacronymlists}{#1}%
414 }
```

`acronymlists`

```
415 \define@key{glossaries.sty}{acronymlists}{%
416   \DeclareAcronymList{#1}%
417 }
```

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`

```
418 \newcommand{\glscounter}{page}
```

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

```
419 \define@key{glossaries.sty}{counter}{%
420   \renewcommand*{\glscounter}{#1}%
421 }
```

`gls@nohyperlist`

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

`lareNoHyperList`

```
423 \newcommand*{\GlsDeclareNoHyperList}[1]{%
424   \ifdefempty\@gls@nohyperlist
425   {%
426     \renewcommand*{\@gls@nohyperlist}{#1}%
427   }%
428   {%
429     \appto\@gls@nohyperlist{,#1}%
430   }%
431 }
```

`nohypertypes`

```
432 \define@key{glossaries.sty}{nohypertypes}{%
433   \GlsDeclareNoHyperList{#1}%
434 }
```

`glossariesWarning` Prints a warning message.

```

435 \newcommand*\GlossariesWarning}[1]{%
436   \PackageWarning{glossaries}{#1}%
437 }
```

`glossariesWarningNoLine` Prints a warning message without the line number.

```

438 \newcommand*\GlossariesWarningNoLine}[1]{%
439   \PackageWarningNoLine{glossaries}{#1}%
440 }
```

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

```

441 \newcommand\glossortentrieswarning{%
442   \typeout{Using TeX to sort glossary entries---this may
443   take a while}%
444 }
```

`nowarn` Define package option to suppress warnings

```

445 \@gls@declareoption{nowarn}{%
446   \ifgls@debug
447     \GlossariesWarning{Warnings can't be suppressed in debug mode}%
448   \else
449     \renewcommand*\GlossariesWarning}[1]{}%
450     \renewcommand*\GlossariesWarningNoLine}[1]{}%
451     \renewcommand*\glossortentrieswarning}{%
452     \renewcommand*\@gls@missinglang@warn}[2]{}%
453   \fi
454 }
```

`missinglang@warn` Missing language warning.

```

455 \newcommand*\@gls@missinglang@warn}[2]{%
456   \PackageWarningNoLine{glossaries}%
457   {No language module detected for '#1'.\MessageBreak
458   Language modules need to be installed separately.\MessageBreak
459   Please check on CTAN for a bundle called\MessageBreak
460   'glossaries-#2' or similar}%
461 }
```

`nolangwarn` Suppress warning if language support not found.

```

462 \@gls@declareoption{nolangwarn}{%
463   \renewcommand*\@gls@missinglang@warn}[2]{}%
464 }
```

`nonglossdefined` Issue a warning if overriding `\printglossary`

```

465 \newcommand*\@gls@warnonglossdefined{%
466   \GlossariesWarning{Overriding \string\printglossary}%
467 }
```

theglossdefined Issue a warning if overriding theglossary

```

468 \newcommand*\@gls@warnontheglossdefined}{%
469 \GlossariesWarning{Overriding 'theglossary' environment}%
470 }
```

noredefwarn Suppress warning on redefinition of \printglossary

```

471 \@gls@declareoption{noredefwarn}{%
472 \renewcommand*\@gls@warnonglossdefined}{}%
473 \renewcommand*\@gls@warnontheglossdefined}{}%
474 }
```

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

```

475 \newcommand*\@gls@sanitizedesc}{%
476 }
```

lssetexpandfield `\glssetexpandfield{<field>}`

Sets field to always expand.

```

477 \newcommand*\glssetexpandfield}[1]{%
478 \csdef{gls@assign@#1@field}##1##2{%
479 \@@gls@expand@field{##1}{#1}{##2}%
480 }%
481 }
```

setnoexpandfield `\glssetnoexpandfield{<field>}`

Sets field to never expand.

```

482 \newcommand*\glssetnoexpandfield}[1]{%
483 \csdef{gls@assign@#1@field}##1##2{%
484 \@@gls@noexpand@field{##1}{#1}{##2}%
485 }%
486 }
```

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

```
487 \glssetexpandfield{type}
```

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

```
488 \glssetnoexpandfield{desc}
```

escplural@field

```
489 \glssetnoexpandfield{descplural}
```

ls@sanitizename

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

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

```
491 \glssetnoexpandfield{name}
```

@sanitizesymbol

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

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

```
493 \glssetnoexpandfield{symbol}
```

bolplural@field

```
494 \glssetnoexpandfield{symbolplural}
```

Sanitizing stuff:

ls@sanitizesort

```
495 \newcommand*{\@gls@sanitizesort}{%  
496   \ifglssanitizesort  
497     \@gls@sanitizesort  
498   \else  
499     \@gls@nosanitizesort  
500   \fi  
501 }
```

ls@sanitizesort

```
502 \newcommand*\@gls@sanitizesort{%  
503   \@onelevel@sanitize\@glo@sort  
504 }
```

@nosanitizesort

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

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

```
506 \newcommand*\@gls@noidx@sanitizesort{%  
507   \ifdefvoid\@glo@sort  
508   }%  
509   {%  
510     \expandafter\@gls@noidx@sanitizesort\@glo@sort\gls@end@sanitizesort  
511   }%  
512 }  
513 \def\@gls@noidx@sanitizesort#1#2\gls@end@sanitizesort{%  
514   \def\@glo@sort{#1#2}%  
515   \@onelevel@sanitize\@glo@sort  
516 }
```

@nosanitizesort

```
517 \newcommand*{\@@gls@noidx@nosanitizesort}{%
518   \ifdefvoid\@glo@sort
519   }%
520   {%
521     \expandafter\@@gls@noidx@no@sanitizesort\@glo@sort\gls@end@sanitizesort
522   }%
523 }
524 \def\@@gls@noidx@no@sanitizesort#1#2\gls@end@sanitizesort{%
525   \bgroup
526     \glsnoidxstripaccents
527     \protected@xdef\@@glo@sort{#1#2}%
528   \egroup
529   \let\@glo@sort\@@glo@sort
530 }
```

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.

```
531 \newcommand*\glsnoidxstripaccents{%
532   \let\IeC\@firstofone
533   \let\'@\@firstofone
534   \let\'@\@firstofone
535   \let\~@\@firstofone
536   \let\"@\@firstofone
537   \let\u@\@firstofone
538   \let\t@\@firstofone
539   \let\d@\@firstofone
540   \let\r@\@firstofone
541   \let=\@\@firstofone
542   \let.\@\@firstofone
543   \let\~@\@firstofone
544   \let\v@\@firstofone
545   \let\H@\@firstofone
546   \let\c@\@firstofone
547   \let\b@\@firstofone
548   \let\a@\secondoftwo
549   \def\AE{AE}%
550   \def\ae{ae}%
551   \def\OE{OE}%
552   \def\oe{oe}%
553   \def\AA{AA}%
554   \def\aa{aa}%
555   \def\L{L}%
556   \def\l{l}%
557   \def\O{O}%
558   \def\o{o}%
559   \def\SS{SS}%
```

```

560 \def\ss{ss}%
561 \def\th{th}%

562 \def\TH{TH}%
563 \def\dh{dh}%
564 \def\DH{DH}%
565 }

```

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.

```

566 \define@boolkey[glS]{sanitize}{description}[true]{%
567 \GlossariesWarning{sanitize={description} package option deprecated}%
568 \ifglS@sanitize@description
569 \glSsetnoexpandfield{desc}%
570 \glSsetnoexpandfield{descplural}%
571 \else
572 \glSsetexpandfield{desc}%
573 \glSsetexpandfield{descplural}%
574 \fi
575 }

576 \define@boolkey[glS]{sanitize}{name}[true]{%
577 \GlossariesWarning{sanitize={name} package option deprecated}%
578 \ifglS@sanitize@name
579 \glSsetnoexpandfield{name}%
580 \else
581 \glSsetexpandfield{name}%
582 \fi
583 }

584 \define@boolkey[glS]{sanitize}{symbol}[true]{%
585 \GlossariesWarning{sanitize={symbol} package option deprecated}%
586 \ifglS@sanitize@symbol
587 \glSsetnoexpandfield{symbol}%
588 \glSsetnoexpandfield{symbolplural}%
589 \else
590 \glSsetexpandfield{symbol}%
591 \glSsetexpandfield{symbolplural}%
592 \fi
593 }

```

sanitizesort

```

594 \define@boolkey{glossaries.sty}[glS]{sanitizesort}[true]{%
595 \ifglSsanitizesort
596 \glSsetnoexpandfield{sortvalue}%
597 \renewcommand*{\@glS@noidx@setsanitizesort}{%
598 \glSsanitizesorttrue
599 \glSsetnoexpandfield{sortvalue}%
600 }%
601 \else

```



```

602   \glsssetexpandfield{sortvalue}%
603   \renewcommand*{\@gls@noidx@setsanitizesort}{%
604     \glsssanitizesortfalse
605     \glsssetexpandfield{sortvalue}%
606   }%
607 \fi
608 }

```

Default setting:

```

609 \glsssanitizesorttrue
610 \glsssetnoexpandfield{sortvalue}%

```

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

```

611 \newcommand*{\@gls@noidx@setsanitizesort}{%
612   \glsssanitizesortfalse
613   \glsssetexpandfield{sortvalue}%
614 }

615 \define@choicekey[gls]{sanitize}{sort}{true,false}[true]{%
616   \setbool{glsssanitizesort}{#1}%
617   \ifglsssanitizesort
618     \glsssetnoexpandfield{sortvalue}%
619   \else
620     \glsssetexpandfield{sortvalue}%
621   \fi
622   \GlossariesWarning{sanitize={sort} package option
623     deprecated. Use sanitizesort instead}%
624 }

```

`sanitize`

```

625 \define@key{glossaries.sty}{sanitize}[description=true,symbol=true,name=true]{%
626   \ifthenelse{\equal{#1}{none}}{%
627     {%
628       \GlossariesWarning{sanitize package option deprecated}%
629       \glsssetexpandfield{name}%
630       \glsssetexpandfield{symbol}%
631       \glsssetexpandfield{symbolplural}%
632       \glsssetexpandfield{desc}%
633       \glsssetexpandfield{descplural}%
634     }%
635   }%
636   \setkeys[gls]{sanitize}{#1}%
637 }%
638 }

```

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

```

639 \newif\ifglstranslate

```

`notranslatorhook` `\@gls@notranslatorhook` has been removed.

s@usetranslator

```
640 \newcommand*\@gls@usetranslator{%
polyglossia tricks \@ifpackageloaded into thinking that babel has been loaded, so check for
polyglossia as well.
641 \@ifpackageloaded{polyglossia}%
642 {%
643   \let\glsifusetranslator\@secondoftwo
644 }%
645 {%
646   \@ifpackageloaded{babel}%
647   {%
648     \IfFileExists{translator.sty}%
649     {%
650       \RequirePackage{translator}%
651       \let\glsifusetranslator\@firstoftwo
652     }%
653   }%
654 }%
655 {}%
656 }%
657 }
```

dtranslatordict Checks if given translator dictionary has been loaded.

```
658 \newcommand{\glsifusedtranslatordict}[3]{%
659   \glsifusetranslator
660   {\ifcsdef{ver@glossaries-dictionary-#1.dict}{#2}{#3}}%
661   {#3}%
662 }
```

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

```
663 \@gls@declareoption{notranslate}{%
664   \glstranslatefalse
665   \let\@gls@usetranslator\relax
666   \let\glsifusetranslator\@secondoftwo
667 }
```

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

```
668 \define@choicekey{glossaries.sty}{translate}%
669   [\gls@translate@val\gls@translate@nr]%
670   {true,false,babel}[true]%
671   {%
672     \ifcase\gls@translate@nr\relax
673     \glstranslatetrue
674     \renewcommand*\@gls@usetranslator{%
675       \@ifpackageloaded{polyglossia}%
676       {%
677         \let\glsifusetranslator\@secondoftwo
678       }%
679     }%
680   }
```

```

679     {%
680     \@ifpackageloaded{babel}%
681     {%
682     \IfFileExists{translator.sty}%
683     {%
684     \RequirePackage{translator}%
685     \let\glsifusetranslator\@firstoftwo
686     }%
687     {}%
688     }%
689     {}%
690     }%
691     }%
692 \or
693 \glstranslatefalse
694 \let\@gls@usetranslator\relax
695 \let\glsifusetranslator\@secondoftwo
696 \or
697 \glstranslatetrue
698 \let\@gls@usetranslator\relax
699 \let\glsifusetranslator\@secondoftwo
700 \fi
701 }

```

Set the default value:

```

702 \glstranslatefalse
703 \let\glsifusetranslator\@secondoftwo
704 \@ifpackageloaded{translator}%
705 {%
706 \glstranslatetrue
707 \let\glsifusetranslator\@firstoftwo
708 }%
709 {%
710 \@for\gls@thissty:=tracklang,babel,ngerman,polyglossia\do
711 {
712 \@ifpackageloaded{\gls@thissty}%
713 {%
714 \glstranslatetrue
715 \@endfortrue
716 }%
717 {}%
718 }
719 }

```

indexonlyfirst Set whether to only index on first use.

```

720 \define@boolkey{glossaries.sty}[gls]{indexonlyfirst}[true]{}
721 \glsindexonlyfirstfalse

```

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

```
722 \define@boolkey{glossaries.sty}[gls]{hyperfirst}[true]{}
723 \glshyperfirsttrue
```

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

```
724 \newcommand*{\@gls@setacrstyle}{}
```

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

```
725 \define@boolkey{glossaries.sty}[glsacr]{footnote}[true]{}%
726 \ifbool{glsacrdescription}%
727 {}%
728 {}%
729 \renewcommand*{\@gls@sanitizedesc}{}%
730 }%
731 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
732 }
```

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

```
733 \define@boolkey{glossaries.sty}[glsacr]{description}[true]{}%
734 \renewcommand*{\@gls@sanitizesymbol}{}%
735 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
736 }
```

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

```
737 \define@boolkey{glossaries.sty}[glsacr]{smallcaps}[true]{}%
738 \renewcommand*{\@gls@sanitizesymbol}{}%
739 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
740 }
```

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

```
741 \define@boolkey{glossaries.sty}[glsacr]{smaller}[true]{}%
742 \renewcommand*{\@gls@sanitizesymbol}{}%
743 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
744 }
```

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

```
745 \define@boolkey{glossaries.sty}[glsacr]{dua}[true]{}%
746 \renewcommand*{\@gls@sanitizesymbol}{}%
747 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
748 }
```

`shortcuts` Define acronym shortcuts.

```
749 \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.

```
750 \newcommand*{\glsorder}{word}
```

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

```
751 \newcommand*{\@glsorder}[1] {}
```

order

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

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

```
754 \newif\ifglsxindy
```

The default is `makeindex`:

```
755 \glsxindyfalse
```

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

```
756 \@gls@declareoption{makeindex}{\glsxindyfalse}
```

The `xindy` package option may have a value which in turn can be a `key=value` list. First define the keys for this sub-list. The boolean `glsnumbers` determines whether to automatically add the `glsnumbers` letter group.

```
757 \define@boolkey[gls]{xindy}{glsnumbers}[true] {}  
758 \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.)

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

Define key to set the language

```
760 \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.

```
761 \ifcsundef{inputencodingname}{%  
762 \def\gls@codepage{}}{%  
763 \def\gls@codepage{\inputencodingname}  
764 }
```

Define a key to set the code page.

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

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

```
766 \define@key{glossaries.sty}{xindy}[] {%  
767 \glsxindytrue  
768 \setkeys[gls]{xindy}{#1}%  
769 }
```

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

```
770 \@gls@declareoption{xindygloss}{%
771   \glsxindytrue
772 }
```

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

```
773 \@gls@declareoption{xindynoglsnumbers}{%
774   \glsxindytrue
775   \gls@xindy@glsnumbersfalse
776 }
```

`\ifglsautomake`

```
777 \newif\ifglsautomake
```

`gls@automake@nr`

```
778 \newcommand{\gls@automake@nr}{1}
```

`automake` If this setting is on, automatically run `makeindex/xindy` at the end of the document. Must be used with `\makeglossaries`. Default is false. As from v4.42, this is now a choice rather than boolean key.

```
779 \define@choicekey{glossaries.sty}{automake}{%
780   [\gls@automake@val\gls@automake@nr]{true,false,immediate}[true]{%
781     \ifnum\gls@automake@nr=1\relax
782     \glsautomakefalse
783   \else
784     \glsautomaketrue
785   \fi
786   \ifglsautomake
787     \renewcommand*{\@gls@doautomake}{%
788       \PackageError{glossaries}{You must use
789       \string\makeglossaries\space with automake=true}
790       {%
791         Either remove the automake=true setting or
792         add \string\makeglossaries\space to your document preamble.%
793       }%
794     }%
795   \else
796     \renewcommand*{\@gls@doautomake}{}%
797   \fi
798 }
799 \glsautomakefalse
```

`@gls@doautomake`

```
800 \newcommand*{\@gls@doautomake}{}
801 \AtEndDocument{\@gls@doautomake}
```

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

```
802 \define@boolkey{glossaries.sty}[gls]{savewrites}[true]{%
803   \ifglssavewrites
804     \renewcommand*{\glswritefiles}{\@glswritefiles}%
805   \else
806     \let\glswritefiles\@empty
807   \fi
808 }
```

Set default:

```
809 \glssavewritesfalse
810 \let\glswritefiles\@empty
```

`compatible-3.07`

```
811 \define@boolkey{glossaries.sty}[gls]{compatible-3.07}[true]{%
812 \boolfalse{glscpatible-3.07}
```

`compatible-2.07`

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

Also set 3.07 compatibility if this option is set.

```
814 \ifbool{glscpatible-2.07}{%
815   {%
816     \booltrue{glscpatible-3.07}%
817   }%
818   {%
819 }
820 \boolfalse{glscpatible-2.07}
```

`al@makeglossary` Store the original definition.

```
821 \let\gls@original@makeglossary\makeglossary
```

`iginal@glossary` Store the original definition.

```
822 \let\gls@original@glossary\glossary
```

`\makeglossary` The `\makeglossary` command is redefined to be identical to `\makeglossaries`. (This is done partly to reinforce the message that you must either use `\@makeglossary` for all the glossaries or for none of them, but is also a legacy from the old `glossary` package.)

```
823 \def\makeglossary{%
824   \GlossariesWarning{Use of \string\makeglossary\space with
825   glossaries.sty is \MessageBreak deprecated. Use \string\makeglossaries\space
826   instead. If you \MessageBreak need the original definition of
827   \string\makeglossary\space use \MessageBreak the package options
828   kernelglossredefs=false (to \MessageBreak restore the former definition of
829   \string\makeglossary) and \MessageBreak nomain (if the file extensions cause a
830   conflict)}%
831   \makeglossaries
832 }
```

override@glossary

```
833 \newcommand*{\@gls@override@glossary}[1][main]{%
834 \GlossariesWarning{Use of \string\glossary\space with
835 glossaries.sty is deprecated. \MessageBreak Indexing should be performed
836 with the user level \MessageBreak commands, such as \string\gls\space or
837 \string\glsadd. If you need the \MessageBreak original definition of
838 \string\glossary\space use the package \MessageBreak options
839 kernelglossredefs=false (to restore the \MessageBreak former definition of
840 \string\glossary) and nomain (if the \MessageBreak file extensions cause a
841 conflict)}}%
842 \gls@glossary{#1}%
843 }
```

In v4.10, the redefinition of `\glossary` was removed since it was never intended as a user level command (and wasn't documented in the user manual), 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.) As from v4.41, the use of `\glossary` now triggers a warning. The package option `kernelglossredefs=nowarn` may be used to remove the warning, but it's better not to use `\glossary`.

`\glossary`

```
844 \if@gls@docloaded
845 \else
846 \def\glossary{\@gls@override@glossary}
847 \fi
```

kernelglossredefs

The `glossaries` package redefines the kernel commands `\makeglossary` and `\glossary` as a legacy action from the former `glossary` package. In hindsight that wasn't a good idea as it's possible that the `glossaries` package may need to be used with another class or package that needs these commands. Neither of these commands are documented in the main user manual and their use is not encouraged. The preferred commands are `\makeglossaries` (to open all associated glossary files) and `\gls`, `\gls\text` etc or `\glsadd` for indexing.

```
848 \define@choicekey{glossaries.sty}{kernelglossredefs}%
849 [\gls@debug@val\gls@debug@nr]{true,false,nowarn}[true]%
850 {%
851 \ifcase\gls@debug@nr\relax
852 \def\glossary{\@gls@override@glossary}%
853 \def\makeglossary{%
854 \GlossariesWarning{Use of \string\makeglossary\space with
855 glossaries.sty is deprecated. Use \string\makeglossaries\space
856 instead. If you need the original definition of
857 \string\makeglossary\space use the package options
858 kernelglossredefs=false (to prevent redefinition of
859 \string\makeglossary) and nomain (if the file extensions cause a
860 conflict)}}%
861 \makeglossaries
862 }%
```



```

863 \or
864 \let\glossary\gls@original@glossary
865 \let\makeglossary\gls@original@makeglossary
866 \or
867 \def\makeglossary{\makeglossaries}%
868 \renewcommand*\@gls@override@glossary}[1][main]{%
869 \gls@glossary{##1}%
870 }%
871 \fi
872 }

```

symbols Create a “symbols” glossary type

```

873 \@gls@declareoption{symbols}{%
874 \let\@gls@do@symbolsdef\@gls@symbolsdef
875 }

```

Default is not to define the symbols glossary:

```

876 \newcommand*\@gls@do@symbolsdef{}

```

@gls@symbolsdef

```

877 \newcommand*\@gls@symbolsdef){%
878 \newglossary[slg]{symbols}{sls}{slo}{\glsymbolsgroupname}%
879 \newcommand*\@printsymbols}[1][\]{\printglossary[type=symbols,##1]}%

```

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

```

880 \newcommand*\@gls@tr@set@symbols@toctitle){%
881 \translatelet{\glossarytoctitle}{Symbols (glossaries)}%
882 }%
883 }%

```

numbers Create a “symbols” glossary type

```

884 \@gls@declareoption{numbers}{%
885 \let\@gls@do@numbersdef\@gls@numbersdef
886 }

```

Default is not to define the numbers glossary:

```

887 \newcommand*\@gls@do@numbersdef{}

```

@gls@numbersdef

```

888 \newcommand*\@gls@numbersdef){%
889 \newglossary[nlg]{numbers}{nls}{nlo}{\glsnumbersgroupname}%
890 \newcommand*\@printnumbers}[1][\]{\printglossary[type=numbers,##1]}%

```

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

```

891 \newcommand*\@gls@tr@set@numbers@toctitle){%
892 \translatelet{\glossarytoctitle}{Numbers (glossaries)}%
893 }%
894 }%

```

`index` Create an “index” glossary type

```
895 \@gls@declareoption{index}{%
896   \ifx\@gls@do@indexdef\@empty
897     \let\@gls@do@indexdef\@gls@indexdef
898   \fi
899 }
```

`noglossaryindex` Counteract index if it happens to be globally used in the document class.

```
900 \@gls@declareoption{noglossaryindex}{%
901   \let\@gls@do@indexdef\relax
902 }
```

Default is not to define index glossary:

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

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

```
904 \newcommand*{\@gls@indexdef}{%
905   \newglossary[ilg]{index}{ind}{idx}{\indexname}%
906   \newcommand*{\printindex}[1] [] {\printglossary[type=index,##1]}%
907   \newcommand*{\newterm}[2] [] {%
908     \newglossaryentry{##2}%
909     {type={index},name={##2},description={\nopostdesc},##1}}
910   \let\@gls@do@indexdef\relax
911 }%
```

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

```
912 \@for\CurrentOption :=\@declaredoptions\do{%
913   \ifx\CurrentOption\@empty
914     \else
915       \@expandtwoargs
916       \in@ {,\CurrentOption ,}{,\@classoptionslist,\@curroptions,}%
917       \ifin@
918         \@use@ption
919         \expandafter \let\csname ds@\CurrentOption\endcsname\@empty
920       \fi
921     \fi
922 }
```

Now process options passed to the package:

```
923 \ProcessOptionsX
```

Load backward compatibility stuff:

```
924 \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:

```
925 \disable@keys{glossaries.sty}{compatible-2.07,%
926   xindy,xindygloss,xindynoglsnumbers,makeindex,%
```

```

927 acronym,translate,notranslate,nolong,nosuper,notree,nostyles,%
928 nomain,noglossaryindex}

Now define \setupglossaries:
929 \newcommand*\setupglossaries}[1]{%
930 \renewcommand*\@gls@setacrstyle}{}%
931 \ifglsacrshortcuts
932 \def\@gls@setupshortcuts{\glsacrshortcutstrue}%
933 \else
934 \def\@gls@setupshortcuts{%
935 \ifglsacrshortcuts
936 \DefineAcronymSynonyms
937 \fi
938 }%
939 \fi
940 \glsacrshortcutsfalse
941 \let\@gls@do@numbersdef\relax
942 \let\@gls@do@symbolssdef\relax
943 \let\@gls@do@indexdef\relax
944 \let\@gls@do@acronymsdef\relax
945 \ifglseentrycounter
946 \let\@gls@doentrycounterdef\relax
947 \else
948 \let\@gls@doentrycounterdef\@gls@define@glossaryentrycounter
949 \fi
950 \ifglssubentrycounter
951 \let\@gls@dosubentrycounterdef\relax
952 \else
953 \let\@gls@dosubentrycounterdef\@gls@define@glossarysubentrycounter
954 \fi
955 \setkeys{glossaries.sty}{#1}%
956 \@gls@setacrstyle
957 \@gls@setupshortcuts
958 \@gls@do@acronymsdef
959 \@gls@do@numbersdef
960 \@gls@do@symbolssdef
961 \@gls@do@indexdef
962 \@gls@doentrycounterdef
963 \@gls@dosubentrycounterdef
964 }

```

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 $\{\langle section-level \rangle . \langle n \rangle . 0\}$ non-existent warning (e.g. `\gls[counter=chapter]{label}`).

```

965 \ifthenelse{\equal{\glscounter}{section}}{%

```

```

966 {%
967   \ifcsundef{chapter}{}%
968   {%
969     \let\@gls@old@chapter\@chapter
970     \def\@chapter[#1]#2{\@gls@old@chapter[#1]{#2}%
971     \ifcsundef{hyperdef}{\hyperdef{section}{\thesection}{}}}%
972   }%
973 }%
974 {}

```

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

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

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

```

976 \newcommand*{\@onlypremakeg}[1]{%
977   \ifx\@gls@onlypremakeg\@empty
978     \def\@gls@onlypremakeg{#1}%
979   \else
980     \expandafter\toks@\expandafter{\@gls@onlypremakeg}%
981     \edef\@gls@onlypremakeg{\the\toks@,\noexpand#1}%
982   \fi
983 }

```

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

```

984 \newcommand*{\@disable@onlypremakeg}{%
985 \@for\@thiscs:=\@gls@onlypremakeg\do{%
986   \expandafter\@disable@premakecs\@thiscs%
987 }}

```

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

```

988 \newcommand*{\@disable@premakecs}[1]{%
989   \def#1{\PackageError{glossaries}{\string#1\space may only be
990   used before \string\makeglossaries}{You can't use
991   \string#1\space after \string\makeglossaries}}%
992 }

```

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`

```
993 \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`

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

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

995 `\newcommand*\glssettoctitle}[1]{%`

996 `\def\glossarytoctitle{\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`

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

`descriptionname`

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

`\symbolname`

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

`\pagelistname`

1000 `\providecommand*\pagelistname}{Page List}`

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

`symbolsgroupname`

1001 `\providecommand*\glsymbolsgroupname}{Symbols}`

`numbersgroupname`

1002 `\providecommand*\glsnumbersgroupname}{Numbers}`

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

1003 `\newcommand*\glspluralsuffix}{s}`

`\acrpluralsuffix` Default plural suffix for acronyms

1004 `\newcommand*\glsacrpluralsuffix}{\glspluralsuffix}`

`\acrpluralsuffix`

1005 `\newcommand*\glsupacrpluralsuffix}{\glstextup{\glsacrpluralsuffix}}`

`\seename`

1006 `\providecommand*\seename}{see}`

`\andname`

1007 `\providecommand*\andname}{\&}`

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

eGlossariesLang

```
1008 \newcommand*\RequireGlossariesLang}[1]{%
1009   \@ifundefined{ver@glossaries-#1.ldf}{\input{glossaries-#1.ldf}}{}%
1010 }
```

sGlossariesLang

```
1011 \newcommand*\ProvidesGlossariesLang}[1]{%
1012   \ProvidesFile{glossaries-#1.ldf}%
1013 }
```

ssarytocaptions Does nothing if translator hasn't been loaded.

```
1014 \newcommand*\addglossarytocaptions}[1]{}
```

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

```
1015 \ifglstranslate
```

Load tracklang

```
1016 \RequirePackage{tracklang}
```

Load translator if required.

```
1017 \@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.)

```
1018 \@ifpackageloaded{translator}
```

```
1019 {%
```

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.

```
1020   \ifboolexpr
```

```
1021   {
```

```
1022     test {\ifdefstring{\trans@languages}{English}}
```

```
1023     and not
```

```
1024     test {\ifdefstring{bbl@loaded}{english}}
```

```
1025   }
```

```
1026   {%
```

```
1027     \let\glsifusetranslator\@secondoftwo
```

```
1028   }%
```

```
1029   {%
```

```
1030     \usedictionary{glossaries-dictionary}%
```

```
1031     \renewcommand*\addglossarytocaptions}[1]{%
```

```
1032       \ifcsundef{captions#1}{}%
```

```
1033       {%
```

```

1034         \expandafter\let\expandafter\@gls@tmp\csname captions#1\endcsname
1035         \expandafter\toks@\expandafter{\@gls@tmp
1036         \renewcommand*{\glossaryname}{\translate{Glossary}}}%
1037     }%
1038     \expandafter\edef\csname captions#1\endcsname{\the\toks@}%
1039 }%
1040 }%
1041 }%
1042 }%
1043 {}%

```

Check for tracked languages

```

1044 \AnyTrackedLanguages
1045 {%
1046     \ForEachTrackedDialect{\this@dialect}{%
1047         \IfTrackedLanguageFileExists{\this@dialect}%
1048         {glossaries-}% prefix
1049         {.ldf}%
1050         {%
1051             \RequireGlossariesLang{\CurrentTrackedTag}%
1052         }%
1053         {%
1054             \@gls@missinglang@warn\this@dialect\CurrentTrackedLanguage
1055         }%
1056     }%
1057 }%
1058 {}%

```

if using translator use translator interface.

```

1059 \glsifusetranslator
1060 {%
1061     \renewcommand*{\glssettoctitle}[1]{%
1062         \ifcsdef{gls@tr@set@#1@toctitle}%
1063         {%
1064             \csuse{gls@tr@set@#1@toctitle}%
1065         }%
1066         {%
1067             \def\glossarytoctitle{\csname @glotype@#1@title\endcsname}%
1068         }%
1069     }%
1070     \renewcommand*{\glossaryname}{\translate{Glossary}}%
1071     \renewcommand*{\acronymname}{\translate{Acronyms}}%
1072     \renewcommand*{\entryname}{\translate{Notation (glossaries)}}%
1073     \renewcommand*{\descriptionname}{%
1074         \translate{Description (glossaries)}}%
1075     \renewcommand*{\symbolname}{\translate{Symbol (glossaries)}}%
1076     \renewcommand*{\pagelistname}{%
1077         \translate{Page List (glossaries)}}%
1078     \renewcommand*{\glssymbolsgroupname}{%
1079         \translate{Symbols (glossaries)}}%

```

```

1080   \renewcommand*{\glsnumbersgroupname}{%
1081     \translate{Numbers (glossaries)}}%
1082   }{}%
1083 \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.

```
1084 \DeclareRobustCommand*\nopostdesc{}
```

`\@nopostdesc` Suppress next description terminator.

```

1085 \newcommand*\@nopostdesc{%
1086   \let\org@glspostdescription\glspostdescription
1087   \def\glspostdescription{%
1088     \let\glspostdescription\org@glspostdescription}%
1089 }

```

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

```
1090 \newcommand*\@no@post@desc{\nopostdesc}
```

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

```
1091 \newcommand{\glspar}{\par}
```

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

```

1092 \newcommand{\setStyleFile}[1]{%
1093   \renewcommand*\gls@istfilebase{#1}%
   Just in case \istfilename has been modified.
1094   \ifglxindy
1095     \def\istfilename{\gls@istfilebase.xdy}
1096   \else
1097     \def\istfilename{\gls@istfilebase.ist}
1098   \fi
1099 }

```

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

```
1100 \@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`

```

1101 \ifglxindy
1102   \def\istfilename{\gls@istfilebase.xdy}
1103 \else
1104   \def\istfilename{\gls@istfilebase.ist}
1105 \fi

```


`gls@istfilebase`

```
1106 \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 \LaTeX , `\@istfilename` ignores its argument.

`\@istfilename`

```
1107 \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`

```
1108 \newcommand*{\glscompositor}{.}
```

`glsSetCompositor` Sets the compositor.

```
1109 \newcommand*{\glsSetCompositor}[1]{%
1110 \renewcommand*{\glscompositor}{#1}}
```

Only use before `\makeglossaries`

```
1111 \@onlypremakeg\glsSetCompositor
```

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

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

```
1112 \newcommand*{\@glsAlphaCompositor}{\glscompositor}
```

`AlphaCompositor` Sets the alpha compositor.

```
1113 \ifglsxindy
1114 \newcommand*\glsSetAlphaCompositor[1]{%
1115 \renewcommand*\@glsAlphaCompositor{#1}}
1116 \else
1117 \newcommand*\glsSetAlphaCompositor[1]{%
1118 \glsnoxindywarning\glsSetAlphaCompositor}
1119 \fi
```

Can only be used before `\makeglossaries`

```
1120 \@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.

```
1121 \newcommand*{\gls@suffixF}{}
```

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

```

1122 \newcommand*\glsSetSuffixF}[1]{%
1123   \renewcommand*\gls@suffixF{#1}}

```

Only has an effect when used before `\makeglossaries`

```

1124 \@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.

```

1125 \newcommand*\gls@suffixFF{}

```

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

```

1126 \newcommand*\glsSetSuffixFF}[1]{%
1127   \renewcommand*\gls@suffixFF{#1}%
1128 }

```

`\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”.

```

1129 \ifcsundef{hyperlink}%
1130 {%
1131   \newcommand*\glsnumberformat}[1]{#1}%
1132 }%
1133 {%
1134   \newcommand*\glsnumberformat}[1]{\glshypernumber{#1}}%
1135 }

```

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
1136 \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
1137 \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`

```
1138 \newcommand*\glossarypreamble{%
1139   \csuse{@glossarypreamble@\currentglossary}%
1140 }
```

`glossarypreamble`

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

Code provided by Michael Pock.

```
1141 \newcommand{\setglossarypreamble}[2][\glsdefaulttype]{%
1142   \ifglossaryexists{#1}{%
1143     \csgdef{@glossarypreamble@#1}{#2}%
1144   }{%
1145     \GlossariesWarning{%
1146       Glossary ‘#1’ is not defined%
1147     }%
1148   }%
1149 }
```

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`

```
1150 \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`.

```
1151 \newcommand*\glossarysection[2][\@gls@title]{%
1152   \def\@gls@title{#2}%
1153   \ifcsundef{phantomsection}%
1154   {%
1155     \@glossarysection{#1}{#2}%
1156   }%
1157   {%
1158     \p@glossarysection{#1}{#2}%
1159   }%
```

```

1160 \glsglossarymark{\glossarytoctitle}%
1161 }

```

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

```

1162 \ifcsundef{glossarymark}%
1163 {%
1164   \newcommand{\glsglossarymark}[1]{\glossarymark{#1}}
1165 }%
1166 {%
1167   \@ifclassloaded{memoir}
1168   {%
1169     \newcommand{\glsglossarymark}[1]{%
1170       \ifglsucmark
1171         \markboth{\memUHead{#1}}{\memUHead{#1}}%
1172       \else
1173         \markboth{#1}{#1}%
1174       \fi
1175     }
1176   }%
1177 {%
1178   \newcommand{\glsglossarymark}[1]{%
1179     \ifglsucmark
1180       \@mkboth{\mfirstucMakeUppercase{#1}}{\mfirstucMakeUppercase{#1}}%
1181     \else
1182       \@mkboth{#1}{#1}%
1183     \fi
1184   }
1185 }
1186 }

```

`\glossarymark` Provided for backward compatibility:

```

1187 \providecommand{\glossarymark}[1]{%
1188   \ifglsucmark
1189     \@mkboth{\mfirstucMakeUppercase{#1}}{\mfirstucMakeUppercase{#1}}%
1190   \else
1191     \@mkboth{#1}{#1}%
1192   \fi
1193 }

```

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`

```

1194 \newcommand*{\setglossarysection}[1]{%
1195 \setkeys{glossaries.sty}{section=#1}}

```

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

glossarysection

```
1196 \newcommand*{\@glossarysection}[2]{%
1197   \ifdefempty\@glossarysecstar
1198   {%
1199     \csname\@glossarysec\endcsname[#1]{#2}%
1200   }%
1201   {%
1202     \csname\@glossarysec\endcsname*{#2}%
1203     \@gls@toc{#1}{\@glossarysec}%
1204   }%
  Do automatic labelling if required
1205   \@glossaryseclabel
1206 }
```

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

```
1207 \newcommand*{\@p@glossarysection}[2]{%
1208   \glsclearpage
1209   \phantomsection
1210   \ifdefempty\@glossarysecstar
1211   {%
1212     \csname\@glossarysec\endcsname{#2}%
1213   }%
1214   {%
1215     \@gls@toc{#1}{\@glossarysec}%
1216     \csname\@glossarysec\endcsname*{#2}%
1217   }%
  Do automatic labelling if required
1218   \@glossaryseclabel
1219 }
```

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

```
1220 \newcommand*{\gls@docclearpage}{%
1221   \ifthenelse{\equal{\@glossarysec}{chapter}}{%
1222     {%
1223       \ifcsundef{cleardoublepage}%
1224       {%
1225         \clearpage
1226       }%
1227     }%
1228     \ifcsdef{if@openright}%
1229     {%
1230       \if@openright
```

```

1231         \cleardoublepage
1232     \else
1233         \clearpage
1234     \fi
1235 }%
1236 {%
1237     \cleardoublepage
1238 }%
1239 }%
1240 }%
1241 {}%
1242 }

```

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

```

1243 \newcommand*\glsclearpage{\gls@doclearpage}

```

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

`\@gls@toc`

```

1244 \newcommand*\@gls@toc[2]{%
1245     \ifglstoc
1246         \ifglsnumberline
1247             \addcontentsline{toc}{#2}{\protect\numberline{#1}}%
1248         \else
1249             \addcontentsline{toc}{#2}{#1}%
1250         \fi
1251     \fi
1252 }

```

1.4 Xindy

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

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

```

1253 \newcommand*\glsnoxywarning[1]{%
1254     \GlossariesWarning{Not in xindy mode --- ignoring \string#1}%
1255 }

```

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

```

1256 \newcommand*\glsnomakeindexwarning[1]{%
1257     \GlossariesWarning{Not in makeindex mode --- ignoring \string#1}%
1258 }

```

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

```
1259 \ifglxindy
1260 \edef\@xdyattributes{\string"default\string"}%
1261 \fi
```

`\dyattributelist` Comma-separated list of attributes.

```
1262 \ifglxindy
1263 \edef\@xdyattributelist{}%
1264 \fi
```

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

```
1265 \ifglxindy
1266 \def\@xdylocref{}
1267 \fi
```

`\@gls@ifinlist`

```
1268 \newcommand*\@gls@ifinlist}[4]{%
1269 \def\@do@ifinlist##1,#1,##2\end@ifinlist{%
1270 \def\@gls@listsuffix{##2}%
1271 \ifx\@gls@listsuffix\@empty
1272 #4%
1273 \else
1274 #3%
1275 \fi
1276 }%
1277 \@do@ifinlist,#2,#1,\end@ifinlist
1278 }
```

`\AddXdyCounters` 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.

```
1279 \ifglxindy
1280 \newcommand*\@xdycounters{\glscounter}
1281 \newcommand*\GlsAddXdyCounters[1]{%
1282 \@for\@gls@ctr:=#1\do{%
    Check if already in list before adding.
1283 \edef\@do@addcounter{%
1284 \noexpand\@gls@ifinlist{\@gls@ctr}{\@xdycounters}{}%
1285 {%
1286 \noexpand\edef\noexpand\@xdycounters{\@xdycounters,%
1287 \noexpand\@gls@ctr}%
1288 }%
1289 }%
1290 \@do@addcounter
1291 }
1292 }
```

Only has an effect before `\writeist`:

```
1293 \@onlypremakeg\GlsAddXdyCounters
1294 \else
1295 \newcommand*\GlsAddXdyCounters[1]{%
1296 \glsnoxindywarning\GlsAddXdyAttribute
1297 }
1298 \fi
```

`saddxdycounters` Counters must all be identified before adding attributes.

```
1299 \newcommand*\@disabled@glsaddxdycounters{%
1300 \PackageError{glossaries}{\string\GlsAddXdyCounters\space
1301 can't be used after \string\GlsAddXdyAttribute}{Move all
1302 occurrences of \string\GlsAddXdyCounters\space before the first
1303 instance of \string\GlsAddXdyAttribute}%
1304 }
```

`AddXdyAttribute` Adds an attribute.

```
1305 \ifglxindy
```

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

```
1306 \newcommand*\@glsaddxdyattribute[2]{%
```

Add to xindy attribute list

```
1307 \edef\@xdyattributes{\@xdyattributes ^^J \string"#1\string" ^^J
1308 \string"#2#1\string"}%
```

Add to xindy markup location.

```
1309 \expandafter\toks@\expandafter{\@xdylocref}%
1310 \edef\@xdylocref{\the\toks@ ^^J%
1311 (markup-locref
1312 :open \string"\glstildechar n%
1313 \expandafter\string\csname glsX#2X#1\endcsname
1314 \string" ^^J
1315 :close \string"\string" ^^J
1316 :attr \string"#2#1\string")}%
```

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

```
1317 \expandafter\gdef\csname glsX#2X#1\endcsname##1##2{%
1318 \setentrycounter[##1]{#2}\csname #1\endcsname{##2}%
1319 }%
1320 }
```

High-level command:

```
1321 \newcommand*\GlsAddXdyAttribute[1]{%
```

Add to comma-separated attribute list

```
1322 \ifx\@xdyattributelist\@empty
1323 \edef\@xdyattributelist{#1}%
1324 \else
1325 \edef\@xdyattributelist{\@xdyattributelist,#1}%
1326 \fi
```


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

```
1327 \for\@this@counter:=\@xdycounters\do{%
1328   \protected@edef\gls@do@addxdyattribute{%
1329     \noexpand\@glsaddxdyattribute{#1}{\@this@counter}%
1330   }
1331   \gls@do@addxdyattribute
1332 }%
```

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

```
1333 \let\GlsAddXdyCounters\@disabled@glsaddxdycounters
1334 }
```

Only has an effect before `\writeist`:

```
1335 \@onlypremakeg\GlsAddXdyAttribute
1336 \else
1337 \newcommand*\GlsAddXdyAttribute[1]{%
1338   \glsnoxindywarning\GlsAddXdyAttribute}
1339 \fi
```

`definedattributes` Add known attributes for all defined counters

```
1340 \ifglsexindy
1341 \newcommand*\@gls@addpredefinedattributes{%
1342   \GlsAddXdyAttribute{glsnumberformat}
1343   \GlsAddXdyAttribute{textrm}
1344   \GlsAddXdyAttribute{textsf}
1345   \GlsAddXdyAttribute{texttt}
1346   \GlsAddXdyAttribute{textbf}
1347   \GlsAddXdyAttribute{textmd}
1348   \GlsAddXdyAttribute{textit}
1349   \GlsAddXdyAttribute{textup}
1350   \GlsAddXdyAttribute{textsl}
1351   \GlsAddXdyAttribute{textsc}
1352   \GlsAddXdyAttribute{emph}
1353   \GlsAddXdyAttribute{glshypernumber}
1354   \GlsAddXdyAttribute{hyperrm}
1355   \GlsAddXdyAttribute{hypersf}
1356   \GlsAddXdyAttribute{hypertt}
1357   \GlsAddXdyAttribute{hyperbf}
1358   \GlsAddXdyAttribute{hypermd}
1359   \GlsAddXdyAttribute{hyperit}
1360   \GlsAddXdyAttribute{hyperup}
1361   \GlsAddXdyAttribute{hypersl}
1362   \GlsAddXdyAttribute{hypersc}
1363   \GlsAddXdyAttribute{hyperemph}

1364   \GlsAddXdyAttribute{glsignore}
1365 }
1366 \else
1367 \let\@gls@addpredefinedattributes\relax
1368 \fi
```

dyuseralphabets List of additional alphabets

```
1369 \def\@xdyuseralphabets{}
```

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

```
1370 \ifglxsindy
1371   \newcommand*\GlsAddXdyAlphabet}[2]{%
1372     \edef\@xdyuseralphabets{%
1373       \@xdyuseralphabets ^^J
1374       (define-alphabet "#1" (#2))}}
1375 \else
1376   \newcommand*\GlsAddXdyAlphabet}[2]{%
1377     \glsnnoxindywarning\GlsAddXdyAlphabet}
1378 \fi
```

This code is only required for xindy:

```
1379 \ifglxsindy
```

dy@locationlist List of predefined location names.

```
1380 \newcommand*\@gls@xdy@locationlist}{%
1381   roman-page-numbers,%
1382   Roman-page-numbers,%
1383   arabic-page-numbers,%
1384   alpha-page-numbers,%
1385   Alpha-page-numbers,%
1386   Appendix-page-numbers,%
1387   arabic-section-numbers%
1388 }
```

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.

```
1389 \protected@edef\@gls@roman{\@roman{0}\string"
1390   \string"roman-numbers-lowercase\string" :sep \string"}%
1391 \@onelevel@sanitize\@gls@roman
1392 \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
1393   :sep \string"}%
1394 \@onelevel@sanitize\@tmp
1395 \ifx\@tmp\@gls@roman
1396   \expandafter
1397     \edef\csname @gls@xdy@Lclass@roman-page-numbers\endcsname{%
1398       \string"roman-numbers-lowercase\string"%
1399     }%
1400 \else
1401   \expandafter
```

```

1402     \edef\csname @gls@xdy@Lclass@roman-page-numbers\endcsname{
1403         :sep \string"\@gls@roman\string"%
1404     }%
1405 \fi

```

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

```

1406 \expandafter\def\csname @gls@xdy@Lclass@Roman-page-numbers\endcsname{%
1407     \string"roman-numbers-uppercase\string"%
1408 }%

```

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

```

1409 \expandafter\def\csname @gls@xdy@Lclass@arabic-page-numbers\endcsname{%
1410     \string"arabic-numbers\string"%
1411 }%

```

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

```

1412 \expandafter\def\csname @gls@xdy@Lclass@alpha-page-numbers\endcsname{%
1413     \string"alpha\string"%
1414 }%

```

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

```

1415 \expandafter\def\csname @gls@xdy@Lclass@Alpha-page-numbers\endcsname{%
1416     \string"ALPHA\string"%
1417 }%

```

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

```

1418 \expandafter\def\csname @gls@xdy@Lclass@Appendix-page-numbers\endcsname{%
1419     \string"ALPHA\string"
1420     :sep \string"\@glsAlpha compositor\string"
1421     \string"arabic-numbers\string"%
1422 }

```

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

```

1423 \expandafter\def\csname @gls@xdy@Lclass@arabic-section-numbers\endcsname{%
1424     \string"arabic-numbers\string"
1425     :sep \string"\gls compositor\string"
1426     \string"arabic-numbers\string"%
1427 }%

```

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

```

1428 \def\@xdyuserlocationdefs{}

```

userlocationnames List of additional user location names

```

1429 \def\@xdyuserlocationnames{}

```

End of xindy-only block:

```

1430 \fi

```

`\xdycrossrefhook` Hook used after writing cross-reference class information.

```
1431 \ifglxindy
1432 \newcommand\@xdycrossrefhook{}
1433 \fi
```

`\GlsAddXdyLocation` [*prefix-loc*]{*name*}{*definition*} Define a new location called *name*. The definition must use xindy syntax. (Note that this doesn't check to see if the location is already defined. That is left to xindy to complain about.)

```
1434 \ifglxindy
1435 \newcommand*\GlsAddXdyLocation[3][\%
1436 \def\@gls@tmp{#1}%
1437 \ifx\@gls@tmp\@empty
1438 \edef\@xdyuserlocationdefs{%
1439 \@xdyuserlocationdefs ^^J%
1440 (define-location-class \string"#2\string"^^J\space\space
1441 \space(:sep \string"{}\glsopenbrace\string" #3
1442 :sep \string"\glsclosebrace\string"))
1443 }%
1444 \else
1445 \edef\@xdyuserlocationdefs{%
1446 \@xdyuserlocationdefs ^^J%
1447 (define-location-class \string"#2\string"^^J\space\space
1448 \space(:sep "\glsopenbrace"
1449 #1
1450 :sep "\glsclosebrace\glsopenbrace" #3
1451 :sep "\glsclosebrace"))
1452 }%
1453 \fi

1454 \edef\@xdyuserlocationnames{%
1455 \@xdyuserlocationnames^^J\space\space\space
1456 \string"#2\string"}%
1457 }
```

Only has an effect before `\writeist`:

```
1458 \@onlypremakeg\GlsAddXdyLocation
1459 \else
1460 \newcommand*\GlsAddXdyLocation[2]{%
1461 \glsnoxindywarning\GlsAddXdyLocation}
1462 \fi
```

`\locationclassorder` Define location class order

```
1463 \ifglxindy
1464 \def\@xdylocationclassorder{^^J\space\space\space
1465 \string"roman-page-numbers\string"^^J\space\space\space
1466 \string"arabic-page-numbers\string"^^J\space\space\space
1467 \string"arabic-section-numbers\string"^^J\space\space\space
1468 \string"alpha-page-numbers\string"^^J\space\space\space
1469 \string"Roman-page-numbers\string"^^J\space\space\space
```

```

1470   \string"Alpha-page-numbers\string"^^J\space\space\space
1471   \string"Appendix-page-numbers\string"
1472   \@xdyuserlocationnames^^J\space\space\space
1473   \string"see\string"
1474   }
1475 \fi

```

Change the location order.

ationClassOrder

```

1476 \ifglxindy
1477   \newcommand*\GlsSetXdyLocationClassOrder[1]{%
1478     \def\@xdylocationclassorder{#1}}
1479 \else
1480   \newcommand*\GlsSetXdyLocationClassOrder[1]{%
1481     \glsnoxywarning\GlsSetXdyLocationClassOrder}
1482 \fi

```

\@xdysortrules Define sort rules

```

1483 \ifglxindy
1484   \def\@xdysortrules{}
1485 \fi

```

\GlsAddSortRule Add a sort rule

```

1486 \ifglxindy
1487   \newcommand*\GlsAddSortRule[2]{%
1488     \expandafter\toks@\expandafter{\@xdysortrules}%
1489     \protected@edef\@xdysortrules{\the\toks@ ^^J
1490       (sort-rule \string"#1\string" \string"#2\string")}%
1491   }
1492 \else
1493   \newcommand*\GlsAddSortRule[2]{%
1494     \glsnoxywarning\GlsAddSortRule}
1495 \fi

```

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

```

1496 \ifglxindy
1497   \def\@xdyrequiredstyles{tex}
1498 \fi

```

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

```

1499 \ifglxindy
1500   \newcommand*\GlsAddXdyStyle[1]{%
1501     \edef\@xdyrequiredstyles{\@xdyrequiredstyles,#1}}%
1502 \else
1503   \newcommand*\GlsAddXdyStyle[1]{%
1504     \glsnoxywarning\GlsAddXdyStyle}
1505 \fi

```

`GlsSetXdyStyles` Reset the list of required styles

```
1506 \ifglxindy
1507   \newcommand*\GlsSetXdyStyles[1]{%
1508     \edef\xdyrequiredstyles{#1}}
1509 \else
1510   \newcommand*\GlsSetXdyStyles[1]{%
1511     \glsnoxindywarning\GlsSetXdyStyles}
1512 \fi
```

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

```
1513 \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.

```
1514 \def\xdylanguage#1#2{}
```

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

```
1515 \ifglxindy
1516   \newcommand*\GlsSetXdyLanguage[2][\glsdefaulttype]{%
1517     \ifglossaryexists{#1}{%
1518       \expandafter\def\csname @xdy@#1@language\endcsname{#2}%
1519     }{%
1520       \PackageError{glossaries}{Can't set language type for
1521         glossary type '#1' --- no such glossary}{%
1522         You have specified a glossary type that doesn't exist}}
1523 \else
1524   \newcommand*\GlsSetXdyLanguage[2][]{%
1525     \glsnoxindywarning\GlsSetXdyLanguage}
1526 \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.

```
1527 \def\gls@codepage#1#2{}
```

`GlsSetXdyCodePage` Define command to set the code page.

```
1528 \ifglxindy
1529   \newcommand*\GlsSetXdyCodePage[1]{%
1530     \renewcommand*\gls@codepage{#1}%
1531   }
```

Suggested by egreg:

```
1532 \AtBeginDocument{%
```

```

1533 \ifx\gls@codepage\@empty
1534 \ifpackageloaded{fontspec}{\def\gls@codepage{utf8}}{}%
1535 \fi
1536 }
1537 \else
1538 \newcommand*\GlsSetXdyCodePage}[1]{%
1539 \glsnoxywarning\GlsSetXdyCodePage}
1540 \fi

```

`\xdylettergroups` Store letter group definitions.

```

1541 \ifglxindy
1542 \ifglsxindy@glsnumbers
1543 \def\@xdylettergroups{(define-letter-group
1544 \string"glslnumbers\string"^^J\space\space\space
1545 :prefixes (\string"0\string" \string"1\string"
1546 \string"2\string" \string"3\string" \string"4\string"
1547 \string"5\string" \string"6\string" \string"7\string"
1548 \string"8\string" \string"9\string")^^J\space\space\space
1549 \@xdynumbergrouporder)}
1550 \else
1551 \def\@xdylettergroups{}
1552 \fi
1553 \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.

```

1554 \newcommand*\GlsAddLetterGroup[2]{%
1555 \expandafter\toks@\expandafter{\@xdylettergroups}%
1556 \protected@edef\@xdylettergroups{\the\toks@^^J%
1557 (define-letter-group \string"#1\string"^^J\space\space\space#2)}%
1558 }%

```

1.5 Loops and conditionals

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

```

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

```

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

```

1559 \newcommand*\forallglossaries}[3][\@glo@types]{%
1560 \@for#2:=#1\do{\ifx#2\@empty\else#3\fi}%
1561 }

```

`\forallacronyms`

```

1562 \newcommand*\forallacronyms}[2]{%

```

```
1563 \@for#1:=\@glsacronymlists\do{\ifx#1\@empty\else#2\fi}%
1564 }
```

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

```
\forglentries[⟨type⟩]{⟨cmd⟩}{⟨code⟩}
```

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

```
1565 \newcommand*\forglentries}[3][\glsdefaulttype]{%
1566 \edef\@glo@list{\csname glolist@#1\endcsname}%
1567 \@for#2:=\@glo@list\do
1568 {%
1569 \ifdefempty{#2}{#3}%
1570 }%
1571 }
```

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

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

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

```
1572 \newcommand*\forallglsentries}[3][\@glo@types]{%
1573 \expandafter\forallglsentries\expandafter[#1]{\@this@glo@}%
1574 {%
1575 \forglentries[\@this@glo@]{#2}{#3}%
1576 }%
1577 }
```

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

```
\ifglossaryexists{⟨type⟩}{⟨true-text⟩}{⟨false-text⟩}
```

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

```
1578 \newcommand{\ifglossaryexists}[3]{%
1579 \ifcsundef{@glo@type@#1@out}{#3}{#2}%
1580 }
```

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`

```
1581 \newcommand*\glsdetoklabel[1]{#1}
```

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

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

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

```
1582 \newcommand{\ifglentryexists}[3]{%
1583   \ifcsundef{glo@\glsdetoklabel{#1}@name}{#3}{#2}%
1584 }
```

`\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>*.

```
1585 \newcommand*\ifglsused[3]{%
1586   \ifbool{glo@\glsdetoklabel{#1}@flag}{#2}{#3}%
1587 }
```

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>*.

```
1588 \newcommand{\glsdoifexists}[2]{%
1589   \ifglentryexists{#1}{#2}{%
1590     \PackageError{glossaries}{Glossary entry '\glsdetoklabel{#1}'
1591       has not been defined}{You need to define a glossary entry before you
1592       can use it.}}%
1593 }
```

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

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

```
1594 \newcommand{\glsdoifnoexists}[2]{%
1595   \ifglentryexists{#1}{%
1596     \PackageError{glossaries}{Glossary entry '\glsdetoklabel{#1}' has already
1597       been defined}{}}{#2}%
1598 }
```

doifexistsorwarn

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

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

```

1599 \newcommand{\glsdoifexistsorwarn}[2]{%
1600   \ifglentryexists{#1}{#2}{%
1601     \GlossariesWarning{Glossary entry ‘\glsdetoklabel{#1}’
1602       has not been defined}%
1603   }%
1604 }

```

lsdoifexistsordo

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

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

```

1605 \newcommand{\glsdoifexistsordo}[3]{%
1606   \ifglentryexists{#1}{#2}{%
1607     \PackageError{glossaries}{Glossary entry ‘\glsdetoklabel{#1}’
1608       has not been defined}{You need to define a glossary entry before you
1609       can use it.}%
1610   #3%
1611 }%
1612 }

```

sarynoexistsordo

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

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

```

1613 \newcommand{\doifglossarynoexistsordo}[3]{%
1614   \ifglossaryexists{#1}%
1615   {%
1616     \PackageError{glossaries}{Glossary type ‘#1’ already exists}{}%
1617     #3%
1618   }%
1619   {#2}%
1620 }

```

fglshaschildren

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

```

1621 \newcommand{\ifglshaschildren}[3]{%
1622   \glsdoifexists{#1}%
1623   {%
1624     \def\do@glshaschildren{#3}%
1625     \edef\@gls@thislabel{\glsdetoklabel{#1}}%
1626     \expandafter\forglentries\expandafter
1627     [\csname glo@\@gls@thislabel @type\endcsname]
1628     {\glo@label}%

```

```

1629   {%
1630     \letcs\glo@parent{glo@\glo@label @parent}%
1631     \ifdefequal\@gls@thislabel\glo@parent
1632     {%
1633       \def\do@glshaschildren{#2}%
1634       \@endfortrue
1635     }%
1636     {}%
1637   }%
1638   \do@glshaschildren
1639 }%
1640 }

```

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

```

1641 \newcommand{\ifglshasparent}[3]{%
1642   \glsdoifexists{#1}%
1643   {%
1644     \ifcsequal{glo@\glsdetoklabel{#1}@parent}{#3}{#2}%
1645   }%
1646 }

```

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

```

1647 \newcommand*{\ifglshasdesc}[3]{%
1648   \ifcsequal{glo@\glsdetoklabel{#1}@desc}%
1649   {#3}%
1650   {#2}%
1651 }

```

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

```

1652 \newcommand*{\ifglsdescsuppressed}[3]{%
1653   \ifcsequal{glo@\glsdetoklabel{#1}@desc}{@no@post@desc}%
1654   {#2}%
1655   {#3}%
1656 }

```

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

```

1657 \newcommand*{\ifglshassymbol}[3]{%
1658   \letcs{\@glo@symbol}{glo@\glsdetoklabel{#1}@symbol}%
1659   \ifdefempty\@glo@symbol
1660   {#3}%
1661   {%
1662     \ifdefequal\@glo@symbol\@gls@default@value
1663     {#3}%
1664     {#2}%
1665   }%

```

1666 }

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

```
1667 \newcommand*{\ifglshaslong}[3]{%
1668   \letcs{\@glo@long}{glo@glsdetoklabel{#1}@long}%
1669   \ifdefempty\@glo@long
1670     {#3}%
1671     {%
1672       \ifdefequal\@glo@long\@gls@default@value
1673         {#3}%
1674         {#2}%
1675     }%
1676 }
```

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

```
1677 \newcommand*{\ifglshasshort}[3]{%
1678   \letcs{\@glo@short}{glo@glsdetoklabel{#1}@short}%
1679   \ifdefempty\@glo@short
1680     {#3}%
1681     {%
1682       \ifdefequal\@glo@short\@gls@default@value
1683         {#3}%
1684         {#2}%
1685     }%
1686 }
```

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

```
1687 \newcommand*{\ifglshasfield}[4]{%
1688   \glsdoifexists{#2}%
1689   {%
1690     \letcs{\@glo@thisvalue}{glo@glsdetoklabel{#2}@#1}%

```

First check supplied field label is defined.

```
1691   \ifdef\@glo@thisvalue
1692     {%
```

Is defined, so now check if empty.

```
1693     \ifdefempty\@glo@thisvalue
1694     {%
```

Is empty, so doesn't have field set.

```
1695       #4%
1696     }%
1697   {%
```

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

```
1698     \ifdefequal\@glo@thisvalue\@gls@default@value
1699     {%
```

Value is set to the default value.

```
1700         #4%
1701         }%
1702         {%
```

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

```
1703         \let\glscurrentfieldvalue\@glo@thisvalue
1704         #3%
1705         }%
1706     }%
1707 }%
1708 {%
```

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

```
1709     \@gls@fetchfield{\@gls@thisfield}{#1}%
```

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

```
1710     \ifdef\@gls@thisfield
1711     {%
```

Is defined, so now check if empty.

```
1712         \letcs{\@glo@thisvalue}{glo@glsetoklabel{#2}@@gls@thisfield}%
1713         \ifdefempty\@glo@thisvalue
1714         {%
```

Is empty so field hasn't been set.

```
1715         #4%
1716         }%
1717     {%
```

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

```
1718         \ifdefequal\@glo@thisvalue\@gls@default@value
1719         {%
```

Value is set to the default value.

```
1720         #4%
1721         }%
1722     {%
```

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

```
1723         \let\glscurrentfieldvalue\@glo@thisvalue
1724         #3%
1725         }%
1726     }%
1727 }%
1728 {%
```

Not defined.

```
1729     \GlossariesWarning{Unknown entry field '#1'}%
1730     #4%
```

```

1731     }%
1732   }%
1733 }%
1734 }

```

urrentfieldvalue

```
1735 \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`

```
1736 \newcommand*\@glo@types{,}
```

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

```

1737 \newcommand*\gls@provide@newglossary{%
1738   \protected@write\@auxout{}\string\providecommand\string\@newglossary[4]{}%

```

Only need to do this once.

```

1739   \let\gls@provide@newglossary\relax
1740 }

```

\defglsentryfmt Allow different glossaries to have different display styles.

```

1741 \newcommand*\defglsentryfmt}[2][\glsdefaulttype]{%
1742   \csgdef{gls@#1@entryfmt}{#2}%
1743 }

```

\gls@doentryfmt

```
1744 \newcommand*\gls@doentryfmt}[1]{\csuse{gls@#1@entryfmt}}
```

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

```

1745 \newcommand*\@gls@forbidtexext}[1]{%
1746   \ifboolexpr{test {\ifdefstring{#1}{tex}}
1747     or test {\ifdefstring{#1}{TEX}}}
1748   {%
1749     \def#1{nottex}%
1750     \PackageError{glossaries}%
1751       {Forbidden '.tex' extension replaced with '.nottex'}%
1752     {I'm sorry, I can't allow you to do something so reckless.\MessageBreak
1753       Don't use '.tex' as an extension for a temporary file.}%
1754   }%

```

```

1755 {%
1756 }%
1757 }

```

`\gls@gobbleopt` Discard optional argument.

```

1758 \newcommand*{\gls@gobbleopt}{\new@ifnextchar[{\@gls@gobbleopt}{}]}
1759 \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`

```

1760 \newcommand*{\newglossary}{\@ifstar\s@newglossary\@ns@newglossary}

```

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

```

1761 \newcommand*{\s@newglossary}[2]{%
1762 \ns@newglossary[#1-glg]{#1}{#1-gls}{#1-glo}{#2}%
1763 }

```

`\ns@newglossary` Define the unstarred version.

```

1764 \newcommand*{\ns@newglossary}[5][glg]{%
1765 \doifglossarynoexistsordo{#2}%
1766 {%

```

Check if default has been set

```

1767 \ifundef\glsdefaultttype
1768 {%
1769 \gdef\glsdefaultttype{#2}%
1770 }{}%

```

Add this to the list of glossary types:

```

1771 \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.

```

1772 \expandafter\gdef\csname glolist@#2\endcsname{,}%

```

Store the file extensions:

```
1773 \expandafter\edef\csname @glotype@#2@log\endcsname{#1}%
1774 \expandafter\edef\csname @glotype@#2@in\endcsname{#3}%
1775 \expandafter\edef\csname @glotype@#2@out\endcsname{#4}%
1776 \expandafter\@gls@forbidtextext\csname @glotype@#2@log\endcsname
1777 \expandafter\@gls@forbidtextext\csname @glotype@#2@in\endcsname
1778 \expandafter\@gls@forbidtextext\csname @glotype@#2@out\endcsname
```

Store the title:

```
1779 \expandafter\def\csname @glotype@#2@title\endcsname{#5}%

1780 \@gls@provide@newglossary
1781 \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.

```
1782 \ifcsundef{gls@#2@entryfmt}%
1783 {%
1784   \defglsentryfmt[#2]{\glsentryfmt}%
1785 }%
1786 {}%
```

Define sort counter if required:

```
1787 \@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.)

```
1788 \@ifnextchar[{\@gls@setcounter{#2}}%
1789   {\@gls@setcounter{#2}[\glscounter]}%
1790 }%
1791 {%
1792   \gls@gobbleopt
1793 }%
1794 }
```

`\altnewglossary`

```
1795 \newcommand*{\altnewglossary}[3]{%
1796   \newglossary[#2-glg]{#1}{#2-gls}{#2-glo}{#3}%
1797 }
```

Only define new glossaries in the preamble:

```
1798 \@onlypreamble{\newglossary}
```

Only define new glossaries before `\makeglossaries`

```
1799 \@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`

```
1800 \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`

```
1801 \def\@gls@setcounter#1[#2]{%
1802 \expandafter\def\csname @gls@#1@counter\endcsname{#2}%
```

Add counter to xindy list, if not already added:

```
1803 \ifglsxindy
1804 \GlsAddXdyCounters{#2}%
1805 \fi
1806 }
```

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

`@gls@getcounter`

```
1807 \newcommand*{\@gls@getcounter}[1]{%
1808 \csname @gls@#1@counter\endcsname
1809 }
```

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

```
1810 \glsdefmain
```

Define the “acronym” glossaries if required.

```
1811 \@gls@do@acronymsdef
```

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

```
1812 \@gls@do@symbolsdef
1813 \@gls@do@numbersdef
1814 \@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.

```
1815 \newcommand*{\newignoredglossary}[1]{%
1816 \ifdefempty\@ignored@glossaries
1817 {%
1818 \edef\@ignored@glossaries{#1}%
1819 }%
1820 {%
1821 \eappto\@ignored@glossaries{,#1}%
1822 }%
1823 \csgdef{glolist@#1}{,}%
1824 \ifcsundef{gls@#1@entryfmt}%
1825 {%
```

```

1826   \defglsentryfmt [#1]{\glsentryfmt}%
1827 }%
1828 {}%
1829 \ifdefempty\@gls@nohyperlist
1830 {%
1831   \renewcommand*\@gls@nohyperlist}{#1}%
1832 }%
1833 {}%
1834   \eappto\@gls@nohyperlist{,#1}%
1835 }%
1836 }

```

`ignored@glossaries` List of ignored glossaries.

```
1837 \newcommand*\@ignored@glossaries{}
```

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

```

1838 \newcommand*\ifignoredglossary}[3]{%
1839   \edef\@gls@igtype{#1}%
1840   \expandafter\DTLifinlist\expandafter
1841   {\@gls@igtype}\@ignored@glossaries}{#2}{#3}%
1842 }

```

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.

```

1843 \define@key{glossentry}{name}{%
1844 \def\@glo@name{#1}%
1845 }

```

`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`.

```

1846 \define@key{glossentry}{description}{%
1847 \def\@glo@desc{#1}%
1848 }

```

descriptionplural

```
1849 \define@key{glossentry}{descriptionplural}{%
1850 \def\@glo@descplural{#1}%
1851 }
```

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 $\langle name \rangle \langle description \rangle$.

```
1852 \define@key{glossentry}{sort}{%
1853 \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.

```
1854 \define@key{glossentry}{text}{%
1855 \def\@glo@text{#1}%
1856 }
```

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.

```
1857 \define@key{glossentry}{plural}{%
1858 \def\@glo@plural{#1}%
1859 }
```

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.

```
1860 \define@key{glossentry}{first}{%
1861 \def\@glo@first{#1}%
1862 }
```

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.

```
1863 \define@key{glossentry}{firstplural}{%
1864 \def\@glo@firstplural{#1}%
1865 }
```

s@default@value

```
1866 \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).

```

1867 \define@key{glossentry}{symbol}{%
1868 \def\@glo@symbol{#1}%
1869 }

```

symbolplural

```

1870 \define@key{glossentry}{symbolplural}{%
1871 \def\@glo@symbolplural{#1}%
1872 }

```

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

```

1873 \define@key{glossentry}{type}{%
1874 \def\@glo@type{#1}}

```

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

```

1875 \define@key{glossentry}{counter}{%
1876 \ifcsundef{c@#1}%
1877 {%
1878 \PackageError{glossaries}%
1879 {There is no counter called ‘#1’}%
1880 {%
1881 The counter key should have the name of a valid counter
1882 as its value%
1883 }%
1884 }%
1885 {%
1886 \def\@glo@counter{#1}%
1887 }%
1888 }

```

see The see key specifies a list of cross-references

```

1889 \define@key{glossentry}{see}{%
1890 \gls@set@xr@key{see}{\@glo@see}{#1}%
1891 }

```

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

Assign a cross-reference key.

```

1892 \newcommand*{\gls@set@xr@key}[3]{%
1893 \renewcommand*{\gls@xr@key}{#1}%
1894 \gls@checkseeallowed
1895 \def#2{#3}%
1896 \@glo@seeautonumberlist
1897 }

```

`\gls@xr@key`

```

1898 \newcommand*{\gls@xr@key}{see}

```

checkseeallowed

```
1899 \newcommand*{\gls@checkseeallowed}{%
1900 \@gls@see@noindex
1901 }
```

ed@preambleonly

```
1902 \newcommand*{\gls@checkseeallowed@preambleonly}{%
1903 \GlossariesWarning{glossaries}%
1904 {'\gls@xr@key' key doesn't have any effect when used in the document
1905 environment. Move the definition to the preamble
1906 after \string\makeglossaries\space
1907 or \string\makenoidxglossaries}%
1908 }
```

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

```
1909 \define@key{glossentry}{parent}{%
1910 \def\@glo@parent{#1}}
```

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

```
1911 \define@choicekey{glossentry}{nonumberlist}%
1912 [\gls@nonumberlist@val\gls@nonumberlist@nr]{true,false}[true]%
1913 {%
1914 \ifcase\gls@nonumberlist@nr\relax
1915 \def\@glo@prefix{\glsnonextpages}%
1916 \@gls@savenonumberlist{true}%
1917 \else
1918 \def\@glo@prefix{\glsnextpages}%
1919 \@gls@savenonumberlist{false}%
1920 \fi
1921 }
```

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.

```
1922 \newcommand*{\@gls@savenonumberlist}[1]{}
```

initnonumberlist

```
1923 \newcommand*{\@gls@initnonumberlist}{}%
```

storenonumberlist

```
1924 \newcommand*{\@gls@storenonumberlist}[1]{}
```

enablesavenonumberlist Allow the nonumberlist value to be saved.

```
1925 \newcommand*{\@gls@enablesavenonumberlist}{%
1926 \renewcommand*{\@gls@initnonumberlist}{%
1927 \undef\@glo@nonumberlist
```

```

1928 }%
1929 \renewcommand*{\@gls@savenonumberlist}[1]{%
1930   \def\@glo@nonumberlist{##1}%
1931 }%
1932 \renewcommand*{\@gls@storenonumberlist}[1]{%
1933   \ifdef\@glo@nonumberlist
1934     {%
1935       \cslet{glo@glstetoklabel{##1}@nonumberlist}{\@glo@nonumberlist}%
1936     }%
1937   }%
1938 }%
1939 \appto\@gls@keymap{,{nonumberlist}{nonumberlist}}%
1940 }

```

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

user1

```

1941 \define@key{glossentry}{user1}{%
1942   \def\@glo@useri{#1}%
1943 }

```

user2

```

1944 \define@key{glossentry}{user2}{%
1945   \def\@glo@userii{#1}%
1946 }

```

user3

```

1947 \define@key{glossentry}{user3}{%
1948   \def\@glo@useriii{#1}%
1949 }

```

user4

```

1950 \define@key{glossentry}{user4}{%
1951   \def\@glo@useriv{#1}%
1952 }

```

user5

```

1953 \define@key{glossentry}{user5}{%
1954   \def\@glo@userv{#1}%
1955 }

```

user6

```

1956 \define@key{glossentry}{user6}{%
1957   \def\@glo@uservi{#1}%
1958 }

```

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.

```

1959 \define@key{glossentry}{short}{%
1960   \def\@glo@short{#1}%
1961 }

```

shortplural This key is provided for use by \newacronym.

```

1962 \define@key{glossentry}{shortplural}{%
1963   \def\@glo@shortpl{#1}%
1964 }

```

long This key is provided for use by \newacronym.

```

1965 \define@key{glossentry}{long}{%
1966   \def\@glo@long{#1}%
1967 }

```

longplural This key is provided for use by \newacronym.

```

1968 \define@key{glossentry}{longplural}{%
1969   \def\@glo@longpl{#1}%
1970 }

```

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

```

1971 \newcommand*\@glsnoname{%
1972   \PackageError{glossaries}{name key required in
1973     \string\newglossaryentry\space for entry '\@glo@label'}{You
1974     haven't specified the entry name}}

```

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

```

1975 \newcommand*\@glsnodesc{%
1976   \PackageError{glossaries}
1977   {%
1978     description key required in \string\newglossaryentry\space
1979     for entry '\@glo@label'%
1980   }%
1981   {%
1982     You haven't specified the entry description%
1983   }%
1984 }%

```

lsdefaultplural Now obsolete. Don't use.

```

1985 \newcommand*\@glsdefaultplural{}

```

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

```

1986 \newcommand*\@gls@missingnumberlist[1]{%
1987   ??%
1988   \ifglssavenumberlist
1989     \GlossariesWarning{Missing number list for entry '#1'.
1990       Maybe makeglossaries + rerun required}%
1991   \else
1992     \PackageError{glossaries}%

```

```

1993   {Package option ‘savenumberlist=true’ required}%
1994   {%
1995       You must use the ‘savenumberlist’ package option
1996       to reference location lists.%
1997   }%
1998   \fi
1999 }

```

`@glsdefaultsort` Define command to set default sort.

```
2000 \newcommand*{\@glsdefaultsort}{\@glo@name}
```

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

```
2001 \newcount\gls@level
```

`@noexpand@field`

```

2002 \newcommand{\@@gls@noexpand@field}[3]{%
2003   \expandafter\global\expandafter
2004     \let\csname glo@#1@#2\endcsname#3%
2005 }

```

`noexpand@fields`

```

2006 \newcommand{\@gls@noexpand@fields}[4]{%
2007   \ifcsdef{gls@assign@#3@field}
2008   {%
2009     \ifdefequal{#4}{\@gls@default@value}%
2010     {%
2011       \edef\@gls@value{\expandonce{#1}}%
2012       \csuse{gls@assign@#3@field}{#2}{\@gls@value}%
2013     }%
2014     {%
2015       \csuse{gls@assign@#3@field}{#2}{#4}%
2016     }%
2017   }%
2018   {%
2019     \ifdefequal{#4}{\@gls@default@value}%
2020     {%
2021       \edef\@gls@value{\expandonce{#1}}%
2022       \@@gls@noexpand@field{#2}{#3}{\@gls@value}%
2023     }%
2024     {%
2025       \@@gls@noexpand@field{#2}{#3}{#4}%
2026     }%
2027   }%
2028 }

```

`ls@expand@field`

```

2029 \newcommand{\@@gls@expand@field}[3]{%
2030   \expandafter

```



```

2031 \protected@xdef\csname glo@#1@#2\endcsname{#3}%
2032 }

```

s@expand@fields

```

2033 \newcommand{\@gls@expand@fields}[4]{%
2034 \ifcsdef{gls@assign@#3@field}
2035 {%
2036 \ifdefequal{#4}{\@gls@default@value}%
2037 {%
2038 \edef\@gls@value{\expandonce{#1}}%
2039 \csuse{gls@assign@#3@field}{#2}{\@gls@value}%
2040 }%
2041 {%
2042 \expandafter\@gls@startswithexpandonce#4\relax\relax\gls@endcheck
2043 {%
2044 \@gls@expand@field{#2}{#3}{#4}%
2045 }%
2046 {%
2047 \csuse{gls@assign@#3@field}{#2}{#4}%
2048 }%
2049 }%
2050 }%
2051 {%
2052 \ifdefequal{#4}{\@gls@default@value}%
2053 {%
2054 \@gls@expand@field{#2}{#3}{#1}%
2055 }%
2056 {%
2057 \@gls@expand@field{#2}{#3}{#4}%
2058 }%
2059 }%
2060 }

```

swithexpandonce

```

2061 \def\@gls@expandonce{\expandonce}
2062 \def\@gls@startswithexpandonce#1#2\gls@endcheck#3#4{%
2063 \def\@gls@tmp{#1}%
2064 \ifdefequal{\@gls@expandonce}{\@gls@tmp}{#3}{#4}%
2065 }

```

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.

```

2066 \let\gls@assign@field\@gls@expand@fields

```

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

```
2067 \newcommand*\glsexpandfields}{%
2068   \let\gls@assign@field\@gls@expand@fields
2069 }
```

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

```
2070 \newcommand*\glsnoexpandfields}{%
2071   \let\gls@assign@field\@gls@noexpand@fields
2072 }
```

`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.)

```
2073 \newrobustcmd{\newglossaryentry}[2]{%
```

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

```
2074   \glsdoifnoexists{#1}%
2075   {%
2076     \gls@defglossaryentry{#1}{#2}%
2077   }%
2078 }
```

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

```
2079 \newcommand*\gls@defdocnewglossaryentry}{%
2080   \let\gls@checkseeallowed\gls@checkseeallowed@preambleonly
2081   \let\newglossaryentry\new@glossaryentry
2082 }
```

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

```
2083 \newrobustcmd{\provideglossaryentry}[2]{%
2084   \ifglsentryexists{#1}%
2085   }{%
2086   {%
2087     \gls@defglossaryentry{#1}{#2}%
2088   }%
2089 }
2090 \@onlypreamble{\provideglossaryentry}
```

`new@glossaryentry` For use in document environment. This opens the `.glsdefs` file, if not already open, so that the entry definition can be saved for the next \LaTeX run. This means that any glossaries at the start of the document can access the entry information.

```
2091 \newrobustcmd{\new@glossaryentry}[2]{%
2092   \ifundef\@gls@deffile
2093   {%
2094     \global\newwrite\@gls@deffile
2095     \immediate\openout\@gls@deffile=\jobname.glsdefs
```

```

2096 }%
2097 {}%
2098 \ifglsentryexists{#1}{}%
2099 {%
2100   \gls@defglossaryentry{#1}{#2}%
2101 }%
2102 \@gls@writedef{#1}%
2103 }

```

At the start of the document input the .glsdefs file if it exists. This is now done by \gls@begindocdefs, which is redefined by glossaries-extra, so that this step can be skipped to avoid loading an obsolete .glsdefs file if the user switches to glossaries-extra with docdef=restricted.

```
2104 \AtBeginDocument{\gls@begindocdefs}
```

The end of the document needs to check if the .glsdefs file has been opened, in which case it needs to be closed.

```
2105 \AtEndDocument{\ifdef\@gls@deffile{\closeout\@gls@deffile}{}}
```

`\gls@begindocdefs` Input the .glsdefs file if it exists and enable document definitions if permitted.

```

2106 \newcommand*{\gls@begindocdefs}{%
2107   \@gls@enablesavenonumberlist
2108   \edef\@gls@restreat{\noexpand\catcode'\noexpand\@=\number\catcode'\@}\relax}%
2109   \makeatletter
2110   \InputIfFileExists{\jobname.glsdefs}{-}{-}%
2111   \@gls@restreat
2112   \undef\@gls@restreat
2113   \gls@defdocnewglossaryentry
2114 }

```

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

```

2115 \newcommand*{\@gls@writedef}[1]{%
2116   \immediate\write\@gls@deffile
2117   {%
2118     \string\ifglsentryexists{#1}{}\glspercentchar^^J%
2119     \expandafter\@gobble\string{\glspercentchar^^J%
2120       \string\gls@defglossaryentry{\glsdetoklabel{#1}}\glspercentchar^^J%
2121       \expandafter\@gobble\string{\glspercentchar%
2122     }%

```

Write key value information:

```

2123   \@for\@gls@map:=\@gls@keymap\do
2124   {%
2125     \letcs\glo@value{glo@\glsdetoklabel{#1}}\expandafter\@secondoftwo\@gls@map}%
2126     \ifdef\glo@value
2127     {%
2128       \@onelevel@sanitize\glo@value
2129       \immediate\write\@gls@deffile
2130       {%
2131         \expandafter\@firstoftwo\@gls@map

```

```

2132         =\expandafter@gobble|string\{\gls@value\expandafter@gobble|string\},%
2133         \glspercentchar
2134     }%
2135 }%
2136 {}%
2137 }%

```

Provide hook:

```

2138 \gls.writedefhook
2139 \immediate\write\@gls@deffile
2140 {%
2141     \glspercentchar^^J%
2142     \expandafter@gobble|string\}\glspercentchar^^J%
2143     \expandafter@gobble|string\}\glspercentchar%
2144 }%
2145 }

```

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

```

2146 \newcommand*\@gls@keymap{%
2147   {name}{name},%
2148   {sort}{sortvalue},% unescaped sort value
2149   {type}{type},%
2150   {first}{first},%
2151   {firstplural}{firstpl},%
2152   {text}{text},%
2153   {plural}{plural},%
2154   {description}{desc},%
2155   {descriptionplural}{descplural},%
2156   {symbol}{symbol},%
2157   {symbolplural}{symbolplural},%
2158   {user1}{useri},%
2159   {user2}{userii},%
2160   {user3}{useriii},%
2161   {user4}{useriv},%
2162   {user5}{userv},%
2163   {user6}{uservi},%
2164   {long}{long},%
2165   {longplural}{longpl},%
2166   {short}{short},%
2167   {shortplural}{shortpl},%
2168   {counter}{counter},%
2169   {parent}{parent}%
2170 }

```

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

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

```

2171 \newcommand*{\@gls@fetchfield}[2]{%
    Ensure user field name is fully expanded
2172 \edef\@gls@thisval{#2}%
    Iterate through known mappings until we find the one for this field.
2173 \@for\@gls@map:=\@gls@keymap\do{%
2174 \edef\@this@key{\expandafter\@firstoftwo\@gls@map}%
2175 \ifdefequal{\@this@key}{\@gls@thisval}%
2176 {%
    Found it.
2177 \edef#1{\expandafter\@secondoftwo\@gls@map}%
    Break out of loop.
2178 \@endfortrue
2179 }%
2180 {}%
2181 }%
2182 }

```

`\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.

```

2183 \newcommand*{\glsaddstoragekey}{\@ifstar\@sglsaddstoragekey\@glsaddstoragekey}
    Starred version switches on expansion for this key.
2184 \newcommand*{\@sglsaddstoragekey}[1]{%
2185 \key@ifundefined{glossentry}{#1}%
2186 {%
2187 \expandafter\newcommand\expandafter*\expandafter
2188 {\csname gls@assign@#1@field\endcsname}[2]{%
2189 \@gls@expand@field{##1}{#1}{##2}%
2190 }%
2191 }%
2192 {}%
2193 \@glsaddstoragekey{#1}%
2194 }

```

Unstarred version doesn't override default expansion.

```

2195 \newcommand*{\@glsaddstoragekey}[3]{%
    Check the specified key doesn't already exist.
2196 \key@ifundefined{glossentry}{#1}%
2197 {%
    Set up the key.
2198 \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
2199 \appto\@gls@keymap{, {#1}{#1}}%

```

Set the default value.

```
2200 \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
```

Assignment code.

```
2201 \appto\@newglossaryentryposthook{%  
2202 \letcs{\@glo@tmp}{@glo@#1}%  
2203 \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%  
2204 }%
```

Define the no-link commands.

```
2205 \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%  
2206 }%  
2207 {%  
2208 \PackageError{glossaries}{Key ‘#1’ already exists}{}%  
2209 }%  
2210 }
```

```
\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.

```
2211 \newcommand*{\glsaddkey}{\@ifstar\@sglsaddkey\@glsaddkey}
```

Starred version switches on expansion for this key.

```
2212 \newcommand*{\@sglsaddkey}[1]{%  
2213 \key@ifundefined{glossentry}{#1}%  
2214 {%  
2215 \expandafter\newcommand\expandafter*\expandafter  
2216 {\csname gls@assign@#1@field\endcsname}[2]{%  
2217 \@@gls@expand@field{##1}{#1}{##2}%  
2218 }%  
2219 }%  
2220 }%  
2221 \@glsaddkey{#1}%  
2222 }
```

Unstarred version doesn't override default expansion.

```
2223 \newcommand*{\@glsaddkey}[7]{%
```

Check the specified key doesn't already exist.

```
2224 \key@ifundefined{glossentry}{#1}%  
2225 {%
```

Set up the key.

```
2226 \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%  
2227 \appto\@gls@keymap{,}{#1}{#1}}%
```

Set the default value.

```
2228 \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
```

Assignment code.

```
2229 \appto\@newglossaryentryposthook{%
2230   \letcs{\@glo@tmp}{@glo@#1}%
2231   \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
2232   }%
```

Define the no-link commands.

```
2233 \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
2234 \newcommand*{#4}[1]{\@Gls@entry@field{##1}{#1}}%
```

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

```
2235 \ifcsdef{@gls@user@#1@}%
2236   {%
2237     \PackageError{glossaries}%
2238     {Can't define '\string#5' as helper command
2239     '\expandafter\string\csname @gls@user@#1@endcsname' already exists}%
2240     {}%
2241   }%
2242   {%
2243     \expandafter\newcommand\expandafter*\expandafter
2244     {\csname @gls@user@#1@endcsname}[2][ ]{%
2245       \new@ifnextchar[%
2246         {\csuse{@gls@user@#1@}{##1}{##2}}%
2247         {\csuse{@gls@user@#1@}{##1}{##2}[ ]}}%
2248     \csdef{@gls@user@#1@}##1##2[##3]{%
2249       \@gls@field@link{##1}{##2}{#3{##2}##3}%
2250     }%
2251     \newrobustcmd*{#5}{%
2252       \expandafter\@gls@hyp@opt\csname @gls@user@#1@endcsname}%
2253     }%
```

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

```
2254 \ifcsdef{@Gls@user@#1@}%
2255   {%
2256     \PackageError{glossaries}%
2257     {Can't define '\string#6' as helper command
2258     '\expandafter\string\csname @Gls@user@#1@endcsname' already exists}%
2259     {}%
2260   }%
2261   {%
2262     \expandafter\newcommand\expandafter*\expandafter
2263     {\csname @Gls@user@#1@endcsname}[2][ ]{%
2264       \new@ifnextchar[%
2265         {\csuse{@Gls@user@#1@}{##1}{##2}}%
2266         {\csuse{@Gls@user@#1@}{##1}{##2}[ ]}}%
2267     \csdef{@Gls@user@#1@}##1##2[##3]{%
2268       \@gls@field@link{##1}{##2}{#4{##2}##3}%
2269     }%
2270     \newrobustcmd*{#6}{%
```

```

2271     \expandafter\@gls@hyp@opt\csname @Gls@user@#1\endcsname}%
2272 }%

Finally the all caps version:
2273     \ifcsdef{@GLS@user@#1@}%
2274     {%
2275         \PackageError{glossaries}%
2276         {Can't define '\string#7' as helper command
2277         '\expandafter\string\csname @GLS@user@#1\endcsname' already exists}%
2278     }%
2279 }%
2280 {%

2281     \expandafter\newcommand\expandafter*\expandafter
2282     {\csname @GLS@user@#1\endcsname}[2][ ]{%
2283         \new@ifnextchar[%
2284             {\csuse{@GLS@user@#1@}{##1}{##2}}%
2285             {\csuse{@GLS@user@#1@}{##1}{##2}[ ]}}%
2286     \csdef{@GLS@user@#1@}##1##2[##3]{%
2287         \@gls@field@link{##1}{##2}{\mfirstucMakeUppercase{#3{##2}##3}}%
2288     }%
2289     \newrobustcmd*{#7}{%
2290         \expandafter\@gls@hyp@opt\csname @GLS@user@#1\endcsname}%
2291     }%
2292 }%
2293 {%
2294     \PackageError{glossaries}{Key '#1' already exists}{}%
2295 }%
2296 }

```

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

```

2297 \newcommand{\glsfieldxdef}[3]{%
2298     \glsdoifexists{#1}%
2299     {%
2300         \edef\@glo@label{\glsdetoklabel{#1}}%
2301         \ifcsdef{glo@\@glo@label @#2}%
2302         {%
2303             \protected@csxdef{glo@\@glo@label @#2}{#3}%
2304         }%
2305         {%
2306             \PackageError{glossaries}{Key '#2' doesn't exist}{}%
2307         }%
2308     }%
2309 }

```

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



```

2310 \newcommand{\glsfielddef}[3]{%
2311 \glsdoifexists{#1}%
2312 {%
2313 \edef\@glo@label{\glsdetoklabel{#1}}%
2314 \ifcsdef{glo@\@glo@label @#2}%
2315 {%
2316 \protected@csedef{glo@\@glo@label @#2}{#3}%
2317 }%
2318 {%
2319 \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2320 }%
2321 }%
2322 }

```

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

```

2323 \newcommand{\glsfieldgdef}[3]{%
2324 \glsdoifexists{#1}%
2325 {%
2326 \edef\@glo@label{\glsdetoklabel{#1}}%
2327 \ifcsdef{glo@\@glo@label @#2}%
2328 {%
2329 \expandafter\gdef\csname glo@\@glo@label @#2\endcsname{#3}%
2330 }%
2331 {%
2332 \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2333 }%
2334 }%
2335 }

```

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

```

2336 \newcommand{\glsfieldddef}[3]{%
2337 \glsdoifexists{#1}%
2338 {%
2339 \edef\@glo@label{\glsdetoklabel{#1}}%
2340 \ifcsdef{glo@\@glo@label @#2}%
2341 {%
2342 \expandafter\def\csname glo@\@glo@label @#2\endcsname{#3}%
2343 }%
2344 {%
2345 \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2346 }%
2347 }%

```

2348 }

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

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

```
2349 \newcommand{\glsfieldfetch}[3]{%
2350   \glsdoifexists{#1}%
2351   {%
2352     \edef\@glo@label{\glsdetoklabel{#1}}%
2353     \ifcsdef{glo@\@glo@label @#2}%
2354     {%
2355       \letcs#3{glo@\@glo@label @#2}%
2356     }%
2357   }%
2358   \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2359 }%
2360 }%
2361 }
```

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

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

```
2362 \newcommand{\ifglsfieldeq}[5]{%
2363   \glsdoifexists{#1}%
2364   {%
2365     \edef\@glo@label{\glsdetoklabel{#1}}%
2366     \ifcsdef{glo@\@glo@label @#2}%
2367     {%
2368       \ifcsstring{glo@\@glo@label @#2}{#3}{#4}{#5}%
2369     }%
2370   }%
2371   \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2372 }%
2373 }%
2374 }
```

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

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

```
2375 \newcommand{\ifglsfielddefeq}[5]{%
2376   \glsdoifexists{#1}%
2377   {%
2378     \edef\@glo@label{\glsdetoklabel{#1}}%
2379     \ifcsdef{glo@\@glo@label @#2}%
2380     {%
```

```

2381 \expandafter\ifdefstrequal
2382 \csname glo@\@glo@label @#2\endcsname{#3}{#4}{#5}%
2383 }%
2384 {%
2385 \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2386 }%
2387 }%
2388 }

```

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

As above but uses \ifcsstrequal instead of \ifdefstrequal

```

2389 \newcommand{\ifglsfieldcseq}[5]{%
2390 \glsdoifexists{#1}%
2391 {%
2392 \edef\@glo@label{\glsdetoklabel{#1}}%
2393 \ifcsdef{glo@\@glo@label @#2}%
2394 {%
2395 \ifcsstrequal{glo@\@glo@label @#2}{#3}{#4}{#5}%
2396 }%
2397 {%
2398 \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2399 }%
2400 }%
2401 }

```

glswritedefhook

```
2402 \newcommand*{\glswritedefhook}{}

```

gls@assign@desc

```

2403 \newcommand*{\gls@assign@desc}[1]{%
2404 \gls@assign@field{#1}{desc}{\@glo@desc}%
2405 \gls@assign@field{\@glo@desc}{#1}{descplural}{\@glo@descplural}%
2406 }

```

ewglossaryentry

```

2407 \newcommand{\longnewglossaryentry}[3]{%
2408 \glsdoifnoexists{#1}%
2409 {%
2410 \bgroup
2411 \let\@org@newglossaryentryprehook\@newglossaryentryprehook
2412 \long\def\@newglossaryentryprehook{%
2413 \long\def\@glo@desc{#3\leavevmode\unskip\nopostdesc}%
2414 \@org@newglossaryentryprehook
2415 }%
2416 \renewcommand*{\gls@assign@desc}[1]{%
2417 \global\cslet{glo@\glsdetoklabel{#1}@desc}{\@glo@desc}%

```

```

2418     \global\cslet{glo@\glsdetoklabel{#1}@descplural}{\@glo@desc}%
2419     }
2420     \gls@defglossaryentry{#1}{#2}%
2421 \egroup
2422 }
2423 }

```

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

```
2424 \@onlypreamble{\longnewglossaryentry}
```

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

```

2425 \newcommand{\longprovideglossaryentry}[3]{%
2426   \ifglseentryexists{#1}{}%
2427   {\longnewglossaryentry{#1}{#2}{#3}}%
2428 }
2429 \@onlypreamble{\longprovideglossaryentry}

```

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

Defines a new entry without checking if it already exists.

```
2430 \newcommand{\gls@defglossaryentry}[2]{%
```

Prevent any further use of `\GlsSetQuote`:

```
2431 \let\GlsSetQuote\gls@nosetquote
```

Store label

```
2432 \edef\@glo@label{\glsdetoklabel{#1}}%
```

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

```
2433 \let\glslabel\@glo@label
```

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

```
2434 \let\@glo@name\@glsnname
```

```
2435 \let\@glo@desc\@glsnodesc
```

```
2436 \let\@glo@descplural\@gls@default@value
```

```
2437 \let\@glo@type\@gls@default@value
```

```
2438 \let\@glo@symbol\@gls@default@value
```

```
2439 \let\@glo@symbolplural\@gls@default@value
```

```
2440 \let\@glo@text\@gls@default@value
```

```
2441 \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.)

```
2442 \let\@glo@first\@gls@default@value
```

```
2443 \let\@glo@firstplural\@gls@default@value
```

Set the default sort:

```
2444 \let\@glo@sort\@gls@default@value
```

Set the default counter:

```
2445 \let\@glo@counter\@gls@default@value
```

```
2446 \def\@glo@see{ }%
```

```
2447 \def\@glo@parent{ }%
```

```
2448 \def\@glo@prefix{ }%
```

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

```
2449 \@gls@initnonnumberlist
```

```
2450 \def\@glo@useri{ }%
```

```
2451 \def\@glo@userii{ }%
```

```
2452 \def\@glo@useriii{ }%
```

```
2453 \def\@glo@useriv{ }%
```

```
2454 \def\@glo@userv{ }%
```

```
2455 \def\@glo@uservi{ }%
```

```
2456 \def\@glo@short{ }%
```

```
2457 \def\@glo@shortpl{ }%
```

```
2458 \def\@glo@long{ }%
```

```
2459 \def\@glo@longpl{ }%
```

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

```
2460 \@newglossaryentryprehook
```

Extract key-val information from third parameter:

```
2461 \setkeys{glossentry}{#2}%
```

Check there is a default glossary.

```
2462 \ifundef\glsdefaultttype
```

```
2463 {%
```

```
2464   \PackageError{glossaries}%
```

```
2465     {No default glossary type (have you used 'nomain' by mistake?)}%
```

```
2466     {If you use package option 'nomain' you must define
```

```
2467       a new glossary before you can define entries}%
```

```
2468   }%
```

```
2469 { }%
```

Assign type. This must be fully expandable

```
2470 \gls@assign@field{\glsdefaultttype}{\@glo@label}{type}{\@glo@type}%
```

```
2471 \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.

```
2472 \ifcsundef{glo@list@\@glo@type}%
```

```
2473 {%
```

```
2474   \PackageError{glossaries}%
```

```

2475     {Glossary type ‘\@glo@type’ has not been defined}%
2476     {You need to define a new glossary type, before making entries
2477     in it}%
2478 }%
2479 {%

    Check if it's an ignored glossary
2480     \ifignoredglossary\@glo@type
2481     {%

        The description may be omitted for an entry in an ignored glossary.
2482         \ifx\@glo@desc\@glsnodesc
2483         \let\@glo@desc\@empty
2484         \fi
2485     }%
2486     {%
2487     }%
2488     \protected@edef\@glo@list@\csname glo@list@\@glo@type\endcsname}%
2489     \expandafter\xdef\csname glo@list@\@glo@type\endcsname{%
2490         \@glo@list@\@glo@label},}%
2491     }%

    Initialise level to 0.
2492     \gls@level=0\relax

    Has this entry been assigned a parent?
2493     \ifx\@glo@parent\@empty

        Doesn't have a parent. Set \glo@<label>@parent to empty.
2494         \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%
2495     \else

        Has a parent. Check to ensure this entry isn't its own parent.
2496         \ifdefequal\@glo@label\@glo@parent%
2497         {%
2498             \PackageError{glossaries}{Entry ‘\@glo@label’ can't be its own parent}{}%
2499             \def\@glo@parent{}%
2500             \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%
2501         }%
2502         {%

            Check the parent exists:
2503             \ifglsentryexists{\@glo@parent}%
2504             {%

                Parent exists. Set \glo@<label>@parent.
2505                 \expandafter\xdef\csname glo@\@glo@label @parent\endcsname{%
2506                     \@glo@parent}%

                Determine level.
2507                 \gls@level=\csname glo@\@glo@parent @level\endcsname\relax
2508                 \advance\gls@level by 1\relax

```

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

```
2509     \ifx\@glo@name\@glsnoname
2510     \expandafter\let\expandafter\@glo@name
2511     \csname glo@\@glo@parent @name\endcsname
```

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

```
2512     \ifx\@glo@plural\@gls@default@value
2513     \expandafter\let\expandafter\@glo@plural
2514     \csname glo@\@glo@parent @plural\endcsname
2515     \fi
2516     \fi
2517     }%
2518     {%
```

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

```
2519     \PackageError{glossaries}%
2520     {%
2521     Invalid parent '@@glo@parent'
2522     for entry '@@glo@label' - parent doesn't exist%
2523     }%
2524     {%
2525     Parent entries must be defined before their children%
2526     }%
2527     \def\@glo@parent{%
2528     \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{%
2529     }%
2530     }%
2531     \fi
```

Set the level for this entry

```
2532 \expandafter\xdef\csname glo@\@glo@label @level\endcsname{\number\gls@level}%
```

Define commands associated with this entry:

```
2533 \gls@assign@field{\@glo@name}{\@glo@label}{sortvalue}{\@glo@sort}%
2534 \letcs\@glo@sort{glo@\@glo@label @sortvalue}%
2535 \gls@assign@field{\@glo@name}{\@glo@label}{text}{\@glo@text}%
2536 \expandafter\gls@assign@field\expandafter
2537     {\csname glo@\@glo@label @text\endcsname\glspluralsuffix}%
2538     {\@glo@label}{plural}{\@glo@plural}%
2539 \expandafter\gls@assign@field\expandafter
2540     {\csname glo@\@glo@label @text\endcsname}%
2541     {\@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.

```
2542 \ifx\@glo@first\@gls@default@value
2543     \expandafter\gls@assign@field\expandafter
2544         {\csname glo@\@glo@label @plural\endcsname}%
2545         {\@glo@label}{firstpl}{\@glo@firstplural}%
2546 \else
2547     \expandafter\gls@assign@field\expandafter
```

```

2548     {\csname glo@\@glo@label @first\endcsname\glspluralsuffix}%
2549     {\@glo@label}{firstpl}{\@glo@firstplural}%
2550 \fi

2551 \ifcsundef{@glo@type@\@glo@type @counter}%
2552 {%
2553     \def@glo@defaultcounter{\glscounter}%
2554 }%
2555 {%
2556     \letcs@glo@defaultcounter{@glo@type@\@glo@type @counter}%
2557 }%
2558 \gls@assign@field{\@glo@defaultcounter}{\@glo@label}{counter}{\@glo@counter}%
2559 \gls@assign@field{}{\@glo@label}{useri}{\@glo@useri}%
2560 \gls@assign@field{}{\@glo@label}{userii}{\@glo@userii}%
2561 \gls@assign@field{}{\@glo@label}{useriii}{\@glo@useriii}%
2562 \gls@assign@field{}{\@glo@label}{useriv}{\@glo@useriv}%
2563 \gls@assign@field{}{\@glo@label}{userv}{\@glo@userv}%
2564 \gls@assign@field{}{\@glo@label}{uservi}{\@glo@uservi}%
2565 \gls@assign@field{}{\@glo@label}{short}{\@glo@short}%
2566 \gls@assign@field{}{\@glo@label}{shortpl}{\@glo@shortpl}%
2567 \gls@assign@field{}{\@glo@label}{long}{\@glo@long}%
2568 \gls@assign@field{}{\@glo@label}{longpl}{\@glo@longpl}%
2569 \ifx@glo@name@glsnoname
2570     \glsnoname
2571     \let@glo@name@gls@default@value
2572 \fi
2573 \gls@assign@field{}{\@glo@label}{name}{\@glo@name}%

```

Set default numberlist if not defined:

```

2574 \ifcsundef{glo@\@glo@label @numberlist}%
2575 {%
2576     \csxdef{glo@\@glo@label @numberlist}{%
2577         \noexpand@gls@missingnumberlist{\@glo@label}}%
2578 }%
2579 {}%

```

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

```

2580 \@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.

```

2581 \def@glo@@desc{\@glo@first}%
2582 \ifx@glo@desc@glo@@desc
2583     \let@glo@desc@glo@first
2584 \fi
2585 \ifx@glo@desc@glsnodesc
2586     \glsnodesc
2587     \let@glo@desc@gls@default@value
2588 \fi
2589 \gls@assign@desc{\@glo@label}%

```


Set the sort key for this entry:

```
2590 \@gls@defsort{\@glo@type}{\@glo@label}%
2591 \def\@glo@symbol{\@glo@text}%
2592 \ifx\@glo@symbol\@glo@symbol
2593   \let\@glo@symbol\@glo@text
2594 \fi
2595 \gls@assign@field{\relax}{\@glo@label}{symbol}{\@glo@symbol}%
2596 \expandafter
2597   \gls@assign@field\expandafter
2598   {\csname glo@\@glo@label @symbol\endcsname}
2599   {\@glo@label}{symbolplural}{\@glo@symbolplural}%
```

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

```
2600 \expandafter\xdef\csname glo@\@glo@label @flagfalse\endcsname{%
2601   \noexpand\global
2602     \noexpand\let\expandafter\noexpand
2603     \csname ifglo@\@glo@label @flag\endcsname\noexpand\iffalse
2604 }%
2605 \expandafter\xdef\csname glo@\@glo@label @flagtrue\endcsname{%
2606   \noexpand\global
2607     \noexpand\let\expandafter\noexpand
2608     \csname ifglo@\@glo@label @flag\endcsname\noexpand\iftrue
2609 }%
2610 \csname glo@\@glo@label @flagfalse\endcsname
```

Sort out any cross-referencing if required.

```
2611 \@glo@autosee
```

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

```
2612 \ifignoredglossary\@glo@type
2613 {%
2614   \csdef{glo@\@glo@label @index}{}%
2615 }
2616 {%
2617   \do@glo@storeentry{\@glo@label}%
2618 }%
```

Define entry counters if enabled:

```
2619 \@newglossaryentry@defcounters
```

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

```
2620 \@newglossaryentryposthook
2621 }
```

\@glo@autosee Automatically implement \glssee.

```
2622 \newcommand*{\@glo@autosee}{%
2623   \ifdefined\@glo@see}%
2624   {%
2625     \protected@edef\do@glssee{%
```

```

2626     \noexpand\@gls@fixbraces\noexpand\@glo@list\@glo@see\noexpand\@nil
2627     \noexpand\expandafter\noexpand\@glssee\noexpand\@glo@list{\@glo@label}}%
2628     \@do@glssee
2629 }%
2630 \@glo@autoseehook
2631 }%

```

glo@autoseehook

```
2632 \newcommand*\@glo@autoseehook{}
```

aryentryprehook Allow extra information to be added to glossary entries:

```
2633 \newcommand*\@newglossaryentryprehook{}
```

ryentryposthook Allow extra information to be added to glossary entries:

```
2634 \newcommand*\@newglossaryentryposthook{}
```

try@defcounters

```
2635 \newcommand*\@newglossaryentry@defcounters{}
```

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

```

2636 \newcommand*\glsmoveentry [2]{%
2637   \edef\@glo@thislabel{\glsdetoklabel{#1}}%
2638   \edef\@glo@type{\csname glo@\@glo@thislabel @type\endcsname}%
2639   \def\@glo@list{,%}
2640   \for@gl@entries[\@glo@type]{\@glo@label}%
2641   {%
2642     \ifdefequal\@glo@thislabel\@glo@label
2643       {\eappto\@glo@list{\@glo@label,%}}%
2644     }%
2645     \cslet{glolist@\@glo@type}{\@glo@list}%
2646     \csdef{glo@\@glo@thislabel @type}{#2}%
2647 }

```

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

```

2648 \ifglxindy
2649   \newcommand*\@glossaryentryfield{\string\@glossentry}
2650 \else
2651   \newcommand*\@glossaryentryfield{\string\glossentry}
2652 \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.)

```

2653 \ifglxindy
2654   \newcommand*\@glossarysubentryfield{%
2655     \string\@subglossentry}

```

```

2656 \else
2657   \newcommand*{\@glossarysubentryfield}{%
2658     \string\subglossentry}
2659 \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.)

```
2660 \newcommand{\@glo@storeentry}[1]{%
```

Escape makeindex/xindy special characters in the label:

```

2661   \edef\@glo@esclabel{#1}%
2662   \@gls@checkmkidxchars\@glo@esclabel

```

Get the sort string and escape any special characters

```

2663   \protected@edef\@glo@sort{\csname glo@#1@sort\endcsname}%
2664   \@gls@checkmkidxchars\@glo@sort

```

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

```
2665   \@gls@checkmkidxchars\@glo@prefix
```

Get the parent, if one exists

```
2666   \edef\@glo@parent{\csname glo@#1@parent\endcsname}%
```

Write the information to the glossary file.

```
2667   \ifglsxindy
```

Store using xindy syntax.

```
2668   \ifx\@glo@parent\@empty
```

Entry doesn't have a parent

```

2669     \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2670       (\string"\@glo@sort\string" %
2671       \string"\@glo@prefix\@glossarysubentryfield{\@glo@esclabel}\string") %
2672       }%
2673     \else

```

Entry has a parent

```

2674     \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2675       \csname glo@\@glo@parent @index\endcsname
2676       (\string"\@glo@sort\string" %
2677       \string"\@glo@prefix\@glossarysubentryfield
2678       {\csname glo@#1@level\endcsname}{\@glo@esclabel}\string") %
2679     }%
2680   \fi
2681 \else

```

Store using makeindex syntax.

```
2682 \ifx\@glo@parent\@empty
    Sanitize \@glo@prefix
2683 \@onelevel@sanitize\@glo@prefix
    Entry doesn't have a parent
2684 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2685 \@glo@sort\@gls@actualchar\@glo@prefix
2686 \@glossaryentryfield{\@glo@esclabel}%
2687 }%
2688 \else
    Entry has a parent
2689 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2690 \csname glo@\@glo@parent @index\endcsname\@gls@levelchar
2691 \@glo@sort\@gls@actualchar\@glo@prefix
2692 \@glossarysubentryfield
2693 {\csname glo@#1@level\endcsname}\@glo@esclabel}%
2694 }%
2695 \fi
2696 \fi
2697 }
```

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`

```
2698 \AtBeginDocument{%
2699 \@ifpackageloaded{amsmath}%
2700 {\let\gls@ifnotmeasuring\@gls@ifnotmeasuring}%
2701 }%
2702 }
2703 \newcommand*\@gls@ifnotmeasuring[1]{%
2704 \ifmeasuring@
2705 \else
2706 #1%
2707 \fi
2708 }
2709 \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.

```
2710 \def\@gls@patchtabularx#1\hbox#2#3!!{%
```

```

2711 \def\TX@trial##1{##1\hbox{\let\glsunset@gobble#2}#3}%
2712 }
2713 \newcommand*\glspatchtabularx{%
2714 \ifdef\TX@trial
2715 {%
2716 \expandafter@gls@patchtabularx\TX@trial{##1}!!%
2717 \let\glspatchtabularx\relax
2718 }%
2719 {}}%
2720 }

```

`\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.

```

2721 \newcommand*\glsreset}[1]{%
2722 \gls@ifnotmeasuring
2723 {%
2724 \glsdoifexists{#1}%
2725 {%
2726 \@glsreset{#1}%
2727 }%
2728 }%
2729 }

```

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

```

2730 \newcommand*\glslocalreset}[1]{%
2731 \gls@ifnotmeasuring
2732 {%
2733 \glsdoifexists{#1}%
2734 {%
2735 \@glslocalreset{#1}%
2736 }%
2737 }%
2738 }

```

`\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.

```

2739 \newcommand*\glsunset}[1]{%
2740 \gls@ifnotmeasuring
2741 {%
2742 \glsdoifexists{#1}%
2743 {%
2744 \@glsunset{#1}%
2745 }%
2746 }%
2747 }

```

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

```

2748 \newcommand*\glslocalunset}[1]{%

```

```

2749 \gls@ifnotmeasuring
2750 {%
2751   \glsdoifexists{#1}%
2752   {%
2753     \glslocalunset{#1}%
2754   }%
2755 }%
2756 }

```

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

```
2757 \newcommand*{\glslocalunset}{\@@glslocalunset}
```

`@@glslocalunset` Local unset without checks.

```

2758 \newcommand*{\@@glslocalunset}[1]{%
2759   \expandafter\let\csname ifglo@glsdetoklabel{#1}@flag\endcsname\iftrue
2760 }

```

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

```
2761 \newcommand*{\glsunset}{\@@glsunset}
```

`\@@glsunset` Global unset without checks.

```

2762 \newcommand*{\@@glsunset}[1]{%
2763   \expandafter\global\csname glo@glsdetoklabel{#1}@flagtrue\endcsname
2764 }

```

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

```
2765 \newcommand*{\glslocalreset}{\@@glslocalreset}
```

`@@glslocalreset` Local reset without checks.

```

2766 \newcommand*{\@@glslocalreset}[1]{%
2767   \expandafter\let\csname ifglo@glsdetoklabel{#1}@flag\endcsname\iffalse
2768 }

```

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

```
2769 \newcommand*{\glsreset}{\@@glsreset}
```

`\@@glsreset` Global reset without checks.

```

2770 \newcommand*{\@@glsreset}[1]{%
2771   \expandafter\global\csname glo@glsdetoklabel{#1}@flagfalse\endcsname
2772 }

```

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

`\glsresetall`

```
2773 \newcommand*{\glsresetall}[1][\@glo@types]{%
2774   \forallglsentries[#1]{\@glsentry}%
2775   {%
2776     \glsreset{\@glsentry}%
2777   }%
2778 }
```

As above, but with only a local effect:

`lslocalresetall`

```
2779 \newcommand*{\glslocalresetall}[1][\@glo@types]{%
2780   \forallglsentries[#1]{\@glsentry}%
2781   {%
2782     \glslocalreset{\@glsentry}%
2783   }%
2784 }
```

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

`\glsunsetall`

```
2785 \newcommand*{\glsunsetall}[1][\@glo@types]{%
2786   \forallglsentries[#1]{\@glsentry}%
2787   {%
2788     \glsunset{\@glsentry}%
2789   }%
2790 }
```

As above, but with only a local effect:

`lslocalunsetall`

```
2791 \newcommand*{\glslocalunsetall}[1][\@glo@types]{%
2792   \forallglsentries[#1]{\@glsentry}%
2793   {%
2794     \glslocalunset{\@glsentry}%
2795   }%
2796 }
```

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

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

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

```
2797 \newcommand*{\@@newglossaryentry@defcounters}{%
2798   \csdef{glo@\@glo@label @currcount}{0}%
2799   \csdef{glo@\@glo@label @prevcount}{0}%
2800 }
```

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

```
2801 \newcommand*{\glsenableentrycount}{%
  Enable new entry fields.
2802   \let\@newglossaryentry@defcounters\@newglossaryentry@defcounters
  Disable \newglossaryentry in the document environment.
2803   \renewcommand*{\gls@defdocnewglossaryentry}{%
2804     \renewcommand*newglossaryentry[2]{%
2805       \PackageError{glossaries}{\string\newglossaryentry\space
2806         may only be used in the preamble when entry counting has
2807         been activated}{If you use \string\glsenableentrycount\space
2808         you must place all entry definitions in the preamble not in
2809         the document environment}%
2810     }%
2811   }%
```

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

```
2812 \newcommand*{\glsentrycurrcount}[1]{%
2813   \ifcsundef{glo@\glsdetoklabel{##1}@currcount}%
2814     {0}{\@gls@entry@field{##1}{currcount}}%
2815   }%
2816 \newcommand*{\glsentryprevcount}[1]{%
2817   \ifcsundef{glo@\glsdetoklabel{##1}@prevcount}%
2818     {0}{\@gls@entry@field{##1}{prevcount}}%
2819   }%
```

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

```
2820 \renewcommand*{\@glsunset}[1]{%
2821   \@glsunset{##1}%
2822   \@gls@increment@currcount{##1}%
2823   }%
2824 \renewcommand*{\@glslocalunset}[1]{%
2825   \@glslocalunset{##1}%
2826   \@gls@local@increment@currcount{##1}%
2827   }%
2828 \renewcommand*{\@glsreset}[1]{%
2829   \@glsreset{##1}%
2830   \csgdef{glo@\glsdetoklabel{##1}@currcount}{0}%
2831   }%
2832 \renewcommand*{\@glslocalreset}[1]{%
2833   \@glslocalreset{##1}%
2834   \csdef{glo@\glsdetoklabel{##1}@currcount}{0}%
2835   }%
```


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.)

```

2836 \def\@cgl@s@##1##2[##3]{%
2837   \ifnum\gl@sentryprevcount{##2}=1\relax
2838     \cgl@sformat{##2}{##3}%
2839     \gl@sunset{##2}%
2840   \else
2841     \@cgl@s@{##1}{##2}[##3]%
2842   \fi
2843 }%
```

Similarly for the analogous commands. No case change plural:

```

2844 \def\@cgl@sp@##1##2[##3]{%
2845   \ifnum\gl@sentryprevcount{##2}=1\relax
2846     \cgl@sp@format{##2}{##3}%
2847     \gl@sunset{##2}%
2848   \else
2849     \@cgl@sp@{##1}{##2}[##3]%
2850   \fi
2851 }%
```

First letter uppercase singular:

```

2852 \def\@cGl@s@##1##2[##3]{%
2853   \ifnum\gl@sentryprevcount{##2}=1\relax
2854     \cGl@sformat{##2}{##3}%
2855     \gl@sunset{##2}%
2856   \else
2857     \@cGl@s@{##1}{##2}[##3]%
2858   \fi
2859 }%
```

First letter uppercase plural:

```

2860 \def\@cGl@sp@##1##2[##3]{%
2861   \ifnum\gl@sentryprevcount{##2}=1\relax
2862     \cGl@sp@format{##2}{##3}%
2863     \gl@sunset{##2}%
2864   \else
2865     \@cGl@sp@{##1}{##2}[##3]%
2866   \fi
2867 }%
```

Write information to aux file at the end of the document

```

2868 \AtEndDocument{\@gls@write@entrycounts}%
```

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

```

2869 \renewcommand*{\@gls@entry@count}[2]{%
2870   \csgdef{glo@glsdetoklabel{##1}@prevcount}{##2}%
2871 }%
```

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

```

2872 \let\glsenableentrycount\relax
2873 }
2874 \@onlypreamble\glsenableentrycount

```

ement@currcount

```

2875 \newcommand*{\@gls@increment@currcount}[1]{%
2876 \csxdef{glo@glsdetoklabel{#1}@currcount}{%
2877 \number\numexpr\glsentrycurrcount{#1}+1}%
2878 }

```

ement@currcount

```

2879 \newcommand*{\@gls@local@increment@currcount}[1]{%
2880 \csedef{glo@glsdetoklabel{#1}@currcount}{%
2881 \number\numexpr\glsentrycurrcount{#1}+1}%
2882 }

```

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.)

```

2883 \newcommand*{\@gls@write@entrycounts}{%
2884 \immediate\write\@auxout
2885 {\string\providecommand*\string\@gls@entry@count}[2]{}}%
2886 \forallglsentries{\@glsentry}{%
2887 \ifglsused{\@glsentry}%
2888 {\immediate\write\@auxout
2889 {\string\@gls@entry@count{\@glsentry}{\glsentrycurrcount{\@glsentry}}}}%
2890 {}%
2891 }%
2892 }

```

gls@entry@count

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

```

2893 \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.)

```

2894 \newrobustcmd*{\cgl}{\@gls@hyp@opt\@cgl}

```

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

```

2895 \newcommand*{\@cgl}[2][ ]{%
2896 \new@ifnextchar[{\@cgl@{#1}{#2}}{\@cgl@{#1}{#2}[ ]}%
2897 }

```

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

```

2898 \def\@cgl@#1#2[#3]{%
2899 \GlossariesWarning{\string\cgl\space is defaulting to

```

```

2900 \string\gls\space since you haven't enabled entry counting}%
2901 \@gls@{#1}{#2}[#3]%
2902 }

```

`\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.

```

2903 \newcommand*{\cglformat}[2]{%
2904 \ifglshaslong{#1}{\glentrylong{#1}}{\glentryfirst{#1}}#2%
2905 }

```

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

```

2906 \newrobustcmd*{\cGls}{\@gls@hyp@opt\@cGls}

```

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

```

2907 \newcommand*{\@cGls}[2] []{%
2908 \new@ifnextchar[{\@cGls@{#1}{#2}}{\@cGls@{#1}{#2}[]}%
2909 }

```

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

```

2910 \def\@cGls@#1#2[#3]{%
2911 \GlossariesWarning{\string\cGls\space is defaulting to
2912 \string\Gls\space since you haven't enabled entry counting}%
2913 \@Gls@{#1}{#2}[#3]%
2914 }

```

`\cGlsformat` 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.

```

2915 \newcommand*{\cGlsformat}[2]{%
2916 \ifglshaslong{#1}{\Glentrylong{#1}}{\Glentryfirst{#1}}#2%
2917 }

```

`\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.)

```

2918 \newrobustcmd*{\cglsp1}{\@gls@hyp@opt\@cglsp1}

```

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

```

2919 \newcommand*{\@cglsp1}[2] []{%
2920 \new@ifnextchar[{\@cglsp1@{#1}{#2}}{\@cglsp1@{#1}{#2}[]}%
2921 }

```

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

```

2922 \def\@cglsp1@#1#2[#3]{%
2923 \GlossariesWarning{\string\cglsp1\space is defaulting to
2924 \string\glsp1\space since you haven't enabled entry counting}%
2925 \@glsp1@{#1}{#2}[#3]%
2926 }

```

`\cglspformat` 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.

```
2927 \newcommand*{\cglspformat}[2]{%
2928   \ifglshaslong{#1}{\glentrylongpl{#1}}{\glentryfirstplural{#1}}#2%
2929 }
```

`\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.)

```
2930 \newrobustcmd*{\cGlsp1}{\@gls@hyp@opt\@cGlsp1}
```

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

```
2931 \newcommand*{\@cGlsp1}[2][ ]{%
2932   \new@ifnextchar[{\@cGlsp1@{#1}{#2}}{\@cGlsp1@{#1}{#2}[ ]}%
2933 }
```

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

```
2934 \def\@cGlsp1@#1#2[#3]{%
2935   \GlossariesWarning{\string\cGlsp1\space is defaulting to
2936     \string\Glsp1\space since you haven't enabled entry counting}%
2937   \@Glsp1@{#1}{#2}[#3]%
2938 }
```

`\cGlspformat` 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.

```
2939 \newcommand*{\cGlspformat}[2]{%
2940   \ifglshaslong{#1}{\glentrylongpl{#1}}{\glentryfirstplural{#1}}#2%
2941 }
```

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`, `\glsp1` and uppercase variants or `\glsadd` and `\glsaddall` will appear in the glossary). The mandatory argument is the filename (with or without `.tex` extension).

`\loadglsentries`

```
2942 \newcommand*{\loadglsentries}[2][\@gls@default]{%
```

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

```

2943 \let\@gls@default\glsdefaultttype
2944 \def\glsdefaultttype{#1}\input{#2}%
2945 \let\glsdefaultttype\@gls@default
2946 }

```

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

```

2947 \@onlypreamble{\loadglsentries}

```

1.11 Using glossary entries in the text

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

`\glstextformat`

```

2948 \newcommand*{\glstextformat}[1]{#1}

```

`\glsentryfmt`

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

```

2949 \newcommand*{\glsentryfmt}{%
2950   \@@gls@default@entryfmt\glsdisplayfirst\glsdisplay
2951 }

```

Format that provides backwards compatibility:

```

2952 \newcommand*{\@@gls@default@entryfmt}[2]{%
2953   \ifdefempty\glscustomtext
2954   {%
2955     \glsifplural
2956     {%

```

Plural form

```

2957     \glscapscase
2958     {%

```

Don't adjust case

```

2959     \ifglsused\glslabel
2960     {%

```

Subsequent use

```

2961     #2{\glsentryplural{\glslabel}}%
2962     {\glsentrydescplural{\glslabel}}%
2963     {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2964     }%
2965     {%

```

First use

```
2966      #1{\glsentryfirstplural{\glslabel}}%
2967      {\glsentrydescplural{\glslabel}}%
2968      {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2969      }%
2970      }%
2971      {%
```

Make first letter upper case

```
2972      \ifglsused\glslabel
2973      {%
```

Subsequent use. (Expansion was used in version 3.07 and below in case the name wasn't the first thing to be displayed, but now the user can sort out the upper casing in `\defglsentryfmt`, which avoids the issues caused by fragile commands.)

```
2974      \ifbool{glscompatible-3.07}%
2975      {%
2976      \protected@edef\@glo@etext{%
2977      #2{\glsentryplural{\glslabel}}%
2978      {\glsentrydescplural{\glslabel}}%
2979      {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2980      \xmakefirstuc\@glo@etext
2981      }%
2982      {%
2983      #2{\Glsentryplural{\glslabel}}%
2984      {\Glsentrydescplural{\glslabel}}%
2985      {\Glsentrysymbolplural{\glslabel}}{\glsinsert}%
2986      }%
2987      }%
2988      {%
```

First use

```
2989      \ifbool{glscompatible-3.07}%
2990      {%
2991      \protected@edef\@glo@etext{%
2992      #1{\glsentryfirstplural{\glslabel}}%
2993      {\glsentrydescplural{\glslabel}}%
2994      {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2995      \xmakefirstuc\@glo@etext
2996      }%
2997      {%
2998      #1{\Glsentryfirstplural{\glslabel}}%
2999      {\Glsentrydescplural{\glslabel}}%
3000      {\Glsentrysymbolplural{\glslabel}}{\glsinsert}%
3001      }%
3002      }%
3003      }%
3004      {%
```

Make all upper case

3005 \ifglsused\glslabel
3006 {%

Subsequent use

3007 \mfirstucMakeUppercase{#2{\glsentryplural{\glslabel}}}%
3008 {\glsentrydescplural{\glslabel}}%
3009 {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
3010 }%
3011 {%

First use

3012 \mfirstucMakeUppercase{#1{\glsentryfirstplural{\glslabel}}}%
3013 {\glsentrydescplural{\glslabel}}%
3014 {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
3015 }%
3016 }%
3017 }%
3018 {%

Singular form

3019 \glscapscase
3020 {%

Don't adjust case

3021 \ifglsused\glslabel
3022 {%

Subsequent use

3023 #2{\glsentrytext{\glslabel}}%
3024 {\glsentrydesc{\glslabel}}%
3025 {\glsentrysymbol{\glslabel}}{\glsinsert}}%
3026 }%
3027 {%

First use

3028 #1{\glsentryfirst{\glslabel}}%
3029 {\glsentrydesc{\glslabel}}%
3030 {\glsentrysymbol{\glslabel}}{\glsinsert}}%
3031 }%
3032 }%
3033 {%

Make first letter upper case

3034 \ifglsused\glslabel
3035 {%

Subsequent use

3036 \ifbool{glscompatible-3.07}%
3037 {%
3038 \protected@edef\@glo@etext{%
3039 #2{\glsentrytext{\glslabel}}%
3040 {\glsentrydesc{\glslabel}}%

```

3041         {\glsentrysymbol{\glslabel}}{\glsinsert}}%
3042     \xmakefirstuc\@glo@etext
3043 }%
3044 {%
3045     #2{\Glsentrytext{\glslabel}}%
3046     {\glsentrydesc{\glslabel}}%
3047     {\glsentrysymbol{\glslabel}}{\glsinsert}}%
3048 }%
3049 }%
3050 {%

```

First use

```

3051     \ifbool{glscompatible-3.07}%
3052     {%
3053         \protected@edef\@glo@etext{%
3054             #1{\Glsentryfirst{\glslabel}}%
3055             {\glsentrydesc{\glslabel}}%
3056             {\glsentrysymbol{\glslabel}}{\glsinsert}}%
3057         \xmakefirstuc\@glo@etext
3058     }%
3059     {%
3060         #1{\Glsentryfirst{\glslabel}}%
3061         {\glsentrydesc{\glslabel}}%
3062         {\glsentrysymbol{\glslabel}}{\glsinsert}}%
3063     }%
3064 }%
3065 }%
3066 {%

```

Make all upper case

```

3067     \ifglsused\glslabel
3068     {%

```

Subsequent use

```

3069         \mfirstucMakeUppercase{#2{\glsentrytext{\glslabel}}%
3070         {\glsentrydesc{\glslabel}}%
3071         {\glsentrysymbol{\glslabel}}{\glsinsert}}%
3072     }%
3073     {%

```

First use

```

3074         \mfirstucMakeUppercase{#1{\glsentryfirst{\glslabel}}%
3075         {\glsentrydesc{\glslabel}}%
3076         {\glsentrysymbol{\glslabel}}{\glsinsert}}%
3077     }%
3078 }%
3079 }%
3080 }%
3081 {%

```

Custom text provided in \glsdisp


```

3082   \ifglused{\glslabel}%
3083   {%
      Subsequent use
3084       #2{\glscustomtext}%
3085       {\glentrydesc{\glslabel}}%
3086       {\glentrysymbol{\glslabel}}{}%
3087   }%
3088   {%
      First use
3089       #1{\glscustomtext}%
3090       {\glentrydesc{\glslabel}}%
3091       {\glentrysymbol{\glslabel}}{}%
3092   }%
3093 }%
3094 }

```

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

```

3095 \newcommand*{\glsgenentryfmt}{%
3096   \ifdefempty\glscustomtext
3097   {%
3098     \glrifplural
3099   {%

```

Plural form

```

3100     \glscapscale
3101     {%

```

Don't adjust case

```

3102     \ifglused\glslabel
3103     {%

```

Subsequent use

```

3104     \glentryplural{\glslabel}\glinsert
3105   }%
3106   {%

```

First use

```

3107     \glentryfirstplural{\glslabel}\glinsert
3108   }%
3109 }%
3110 }%

```

Make first letter upper case

```

3111     \ifglused\glslabel
3112     {%

```

Subsequent use.

```

3113     \Glentryplural{\glslabel}\glinsert
3114   }%
3115   {%

```

First use

```
3116      \Glsentryfirstplural{\glslabel}\glsinsert
3117      }%
3118      }%
3119      {%
```

Make all upper case

```
3120      \ifglsused\glslabel
3121      {%
```

Subsequent use

```
3122      \mfirstucMakeUppercase
3123      {\glsentryplural{\glslabel}\glsinsert}%
3124      }%
3125      {%
```

First use

```
3126      \mfirstucMakeUppercase
3127      {\glsentryfirstplural{\glslabel}\glsinsert}%
3128      }%
3129      }%
3130      }%
3131      {%
```

Singular form

```
3132      \glscapscale
3133      {%
```

Don't adjust case

```
3134      \ifglsused\glslabel
3135      {%
```

Subsequent use

```
3136      \glsentrytext{\glslabel}\glsinsert
3137      }%
3138      {%
```

First use

```
3139      \glsentryfirst{\glslabel}\glsinsert
3140      }%
3141      }%
3142      {%
```

Make first letter upper case

```
3143      \ifglsused\glslabel
3144      {%
```

Subsequent use

```
3145      \Glsentrytext{\glslabel}\glsinsert
3146      }%
3147      {%
```

First use

```
3148      \Glsentryfirst{\glslabel}\glsinsert
3149      }%
3150    }%
3151    {%
```

Make all upper case

```
3152      \ifglsused\glslabel
3153      {%
```

Subsequent use

```
3154      \mfirstucMakeUppercase{\glsentrytext{\glslabel}\glsinsert}%
3155      }%
3156    {%
```

First use

```
3157      \mfirstucMakeUppercase{\glsentryfirst{\glslabel}\glsinsert}%
3158      }%
3159    }%
3160  }%
3161 }%
3162 {%
```

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

```
3163   \glscustomtext\glsinsert
3164   }%
3165 }
```

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

```
3166 \newcommand*{\glsngenacfmt}{%
3167   \ifdefempty\glscustomtext
3168   {%
3169     \ifglsused\glslabel
3170     {%
```

Subsequent use:

```
3171     \glsifplural
3172     {%
```

Subsequent plural form:

```
3173     \glscapscase
3174     {%
```

Subsequent plural form, don't adjust case:

```
3175     \acronymfont{\glsentryshortpl{\glslabel}}\glsinsert
3176     }%
3177     {%
```

Subsequent plural form, make first letter upper case:

```
3178     \acronymfont{\Glsentryshortpl{\glslabel}}\glsinsert
3179     }%
3180     {%
```

Subsequent plural form, all caps:

```
3181      \mfirstucMakeUppercase
3182      {\acronymfont{\glsentryshortpl{\glslabel}}\glsinsert}%
3183      }%
3184      }%
3185      {%
```

Subsequent singular form

```
3186      \glscapscase
3187      {%
```

Subsequent singular form, don't adjust case:

```
3188      \acronymfont{\glsentryshort{\glslabel}}\glsinsert
3189      }%
3190      {%
```

Subsequent singular form, make first letter upper case:

```
3191      \acronymfont{\Glsentryshort{\glslabel}}\glsinsert
3192      }%
3193      {%
```

Subsequent singular form, all caps:

```
3194      \mfirstucMakeUppercase
3195      {\acronymfont{\glsentryshort{\glslabel}}\glsinsert}%
3196      }%
3197      }%
3198      }%
3199      {%
```

First use:

```
3200      \glsifplural
3201      {%
```

First use plural form:

```
3202      \glscapscase
3203      {%
```

First use plural form, don't adjust case:

```
3204      \genplacrfullformat{\glslabel}{\glsinsert}%
3205      }%
3206      {%
```

First use plural form, make first letter upper case:

```
3207      \Genplacrfullformat{\glslabel}{\glsinsert}%
3208      }%
3209      {%
```

First use plural form, all caps:

```
3210      \mfirstucMakeUppercase
3211      {\genplacrfullformat{\glslabel}{\glsinsert}}%
3212      }%
3213      }%
3214      {%
```

First use singular form

```
3215     \glscapscase
3216     {%
```

First use singular form, don't adjust case:

```
3217     \genacrfullformat{\glslabel}{\glsinsert}%
3218     }%
3219     {%
```

First use singular form, make first letter upper case:

```
3220     \Genacrfullformat{\glslabel}{\glsinsert}%
3221     }%
3222     {%
```

First use singular form, all caps:

```
3223     \mfirstucMakeUppercase
3224     {\genacrfullformat{\glslabel}{\glsinsert}}%
3225     }%
3226     }%
3227     }%
3228     }%
3229     {%
```

User supplied text.

```
3230     \glscustomtext
3231     }%
3232 }
```

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

The full format used by \gls`genacfmt` (singular).

```
3233 \newcommand*{\genacrfullformat}[2]{%
3234   \glsentrylong{#1}#2\space
3235   (\protect\firstacronymfont{\glsentryshort{#1}})}%
3236 }
```

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

As above but makes the first letter upper case.

```
3237 \newcommand*{\Genacrfullformat}[2]{%
3238   \protected@edef\gls@text{\genacrfullformat{#1}{#2}}%
3239   \xmakefirstuc\gls@text
3240 }
```

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

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

```
3241 \newcommand*{\genplacrfullformat}[2]{%
3242   \glstrylongpl{#1}#2\space
3243   (\protect\firstacronymfont{\glstryshortpl{#1}})%
3244 }
```

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

As above but makes the first letter upper case.

```
3245 \newcommand*{\Genplacrfullformat}[2]{%
3246   \protected@edef\gls@text{\genplacrfullformat{#1}{#2}}%
3247   \xmakefirstuc\gls@text
3248 }
```

`\glsdisplayfirst` Deprecated. Kept for backward compatibility.

```
3249 \newcommand*{\glsdisplayfirst}[4]{#1#4}
```

`\glsdisplay` Deprecated. Kept for backward compatibility.

```
3250 \newcommand*{\glsdisplay}[4]{#1#4}
```

`\defglsdisplay` Deprecated. Kept for backward compatibility.

```
3251 \newcommand*{\defglsdisplay}[2][\glsdefaulttype]{%
3252   \GlossariesWarning{\string\defglsdisplay\space is now obsolete.^^J
3253   Use \string\defglsentryfmt\space instead}%
3254   \expandafter\def\csname gls@#1@display\endcsname##1##2##3##4{#2}%
3255   \edef\@gls@doentrydef{%
3256     \noexpand\defglsentryfmt [#1]{%
3257       \noexpand\ifcsdef{gls@#1@displayfirst}%
3258       {%
3259         \noexpand\@gls@default@entryfmt
3260         {\noexpand\csuse{gls@#1@displayfirst}}%
3261         {\noexpand\csuse{gls@#1@display}}%
3262       }%
3263       {%
3264         \noexpand\@gls@default@entryfmt
3265         {\noexpand\glsdisplayfirst}%
3266         {\noexpand\csuse{gls@#1@display}}%
3267       }%
3268     }%
3269   }%
3270   \@gls@doentrydef
3271 }
```

`\glsdisplayfirst` Deprecated. Kept for backward compatibility.

```
3272 \newcommand*{\defglsdisplayfirst}[2][\glsdefaulttype]{%
3273   \GlossariesWarning{\string\defglsdisplayfirst\space is now obsolete.^^J
3274   Use \string\defglsentryfmt\space instead}%
3275 }
```

```

3275 \expandafter\def\csname gls@#1@displayfirst\endcsname##1##2##3##4{#2}%
3276 \edef\@gls@doentrydef{%
3277   \noexpand\defglsentryfmt [#1] {%
3278     \noexpand\ifcsdef{gls@#1@display}%
3279     {%
3280       \noexpand\@@gls@default@entryfmt
3281       {\noexpand\csuse{gls@#1@displayfirst}}}%
3282       {\noexpand\csuse{gls@#1@display}}}%
3283     }%
3284     {%

3285     \noexpand\@@gls@default@entryfmt
3286     {\noexpand\csuse{gls@#1@displayfirst}}}%
3287     {\noexpand\glsdisplay}%
3288     }%
3289   }%
3290 }%
3291 \@gls@doentrydef
3292 }

```

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 `\defglsentryfmt`). 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.

```

3293 \define@key{glslink}{counter}{%
3294   \ifcsundef{c@#1}%
3295   {%
3296     \PackageError{glossaries}%
3297     {There is no counter called '#1'}%
3298     {%
3299       The counter key should have the name of a valid counter
3300       as its value%
3301     }%
3302   }%
3303   {%
3304     \def\@gls@counter{#1}%
3305   }%
3306 }

```

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.

```
3307 \define@key{glslink}{format}{%
3308   \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.

```
3309 \define@boolkey{glslink}{hyper}[true]{}
```

Initialise hyper key.

```
3310 \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.

```
3311 \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.

```
3312 \newcommand*{\glslinkvar}[3]{#1}
```

`\glsifhyper` Now deprecated.

```
3313 \newcommand*{\glsifhyper}[2]{%
3314   \glslinkvar{#1}{#2}{#1}%
3315   \GlossariesWarning{\string\glsifhyper\space is deprecated. Did
3316     you mean \string\glsifhyperon\space or \string\glslinkvar?}%
3317 }
```

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

```
3318 \newcommand*{\@gls@hyp@opt}[1]{%
3319   \let\glslinkvar\@firstofthree
3320   \let\@gls@hyp@opt@cs#1\relax
3321   \@ifstar{\s@gls@hyp@opt}%
3322   {\@ifnextchar+{\@firstoftwo{\p@gls@hyp@opt}}{#1}}%
3323 }
```

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

```
3324 \newcommand*{\s@gls@hyp@opt}[1] []{%
```



```

3325 \let\glslinkvar\@secondofthree
3326 \@gls@hyp@opt@cs[hyper=false,#1]}

```

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

```

3327 \newcommand*{\p@gls@hyp@opt}[1] [] {%
3328 \let\glslinkvar\@thirdofthree
3329 \@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`

```

3330 \newrobustcmd*{\glslink}{%
3331 \@gls@hyp@opt\@gls@link
3332 }

```

`\@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.

```

3333 \newcommand*{\@gls@link}[3] [] {%
3334 \glsdoifexistsordo{#2}%
3335 {%
3336 \let\do@gls@link@checkfirsthyper\relax
3337 \@gls@link[#1]{#2}{#3}%
3338 }%

```

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

```

3339 \glstextformat{#3}%
3340 }%

```

```

3341 \glspostlinkhook
3342 }

```

`glspostlinkhook`

```

3343 \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.

```
3344 \newcommand*{\@gls@link@checkfirsthyper}{%
3345   \ifglsused{\glslabel}%
3346   {%
3347   }%
3348   {%
3349     \gls@checkisacronymlist\glstype
3350     \ifglshyperfirst
3351     \ifglsisacronymlist
3352     \ifglsacrfootnote
3353     \KV@glslink@hyperfalse
3354     \fi
3355     \fi
3356     \else
3357     \KV@glslink@hyperfalse
3358     \fi
3359   }%
```

Allow user to hook into this

```
3360 \glslinkcheckfirsthyperhook
3361 }
```

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

```
3362 \newcommand*{\glslinkcheckfirsthyperhook}{}
```

linkpostsetkeys

```
3363 \newcommand*{\glslinkpostsetkeys}{}
```

\glsifhyperon Check the value of the hyper key:

```
3364 \newcommand{\glsifhyperon}[2]{\ifKV@glslink@hyper#1\else#2\fi}
```

ablehyperinlist Disable hyperlink if in the “nohyper” list.

```
3365 \newcommand*{\do@glsdisablehyperinlist}{%
3366   \expandafter\DTLifinlist\expandafter{\glstype}{\@gls@nohyperlist}%
3367   {\KV@glslink@hyperfalse}}%
3368 }
```

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

```
3369 \newcommand*{\@gls@setdefault@glslink@opts}{}
```

\@gls@link

```
3370 \def\@gls@link[#1]#2#3{%
```

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

```
3371   \leavevmode
3372   \edef\glslabel{\glsdetoklabel{#2}}%
```

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

```
3373 \def@gls@link@opts{#1}%
3374 \let@gls@link@label\glslabel
3375 \def@glsnumberformat{glsnumberformat}%
3376 \edef@gls@counter{\csname glo@\glslabel @counter\endcsname}%
```

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

```
3377 \edef@glstype{\csname glo@\glslabel @type\endcsname}%
```

Save original setting

```
3378 \let\org@ifKV@glslink@hyper\ifKV@glslink@hyper
```

Set defaults:

```
3379 \@gls@setdefault@glslink@opts
```

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

```
3380 \do@gl:disablehyperinlist
```

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

```
3381 \do@gls@link@checkfirsthyper
3382 \setkeys{glslink}{#1}%
```

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

```
3383 \glslinkpostsetkeys
```

Store the entry’s counter in \theglsentrycounter

```
3384 \@gls@saveentrycounter
```

Define sort key if necessary:

```
3385 \@gls@setsort{\glslabel}%
```

(De-tok’ing done by \@do@wrglossary)

```
3386 \@do@wrglossary{#2}%
3387 \ifKV@glslink@hyper
3388 \@glslink{\glolinkprefix\glslabel}{\glstextformat{#3}}%
3389 \else
3390 \glsdonohyperlink{\glolinkprefix\glslabel}{\glstextformat{#3}}%
3391 \fi
```

Restore original setting

```
3392 \let\ifKV@glslink@hyper\org@ifKV@glslink@hyper
3393 }
```

\glolinkprefix

```
3394 \newcommand*{\glolinkprefix}{glo:}
```

glsentrycounter Set default value of entry counter

```
3395 \def\glsentrycounter{\glscounter}%
```

aveentrycounter Need to check if using equation counter in align environment:

```
3396 \newcommand*{\@gls@saveentrycounter}{%
3397 \def\@gls@Hcounter{}}%
```

Are we using equation counter?

```
3398 \ifthenelse{\equal{\@gls@counter}{equation}}{%
3399 {
```

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

```
3400 \ifcsundef{xatlevel@}%
3401 {%
3402 \edef\theglsentrycounter{\expandafter\noexpand
3403 \csname the\@gls@counter\endcsname}%
3404 }%
3405 {%
3406 \ifx\xatlevel@\@empty
3407 \edef\theglsentrycounter{\expandafter\noexpand
3408 \csname the\@gls@counter\endcsname}%
3409 \else
3410 \savecounters@
3411 \advance\c@equation by 1\relax
3412 \edef\theglsentrycounter{\csname the\@gls@counter\endcsname}%
```

Check if hyperref version of this counter

```
3413 \ifcsundef{theH\@gls@counter}%
3414 {%
3415 \def\@gls@Hcounter{\theglsentrycounter}%
3416 }%
3417 {%
3418 \def\@gls@Hcounter{\csname theH\@gls@counter\endcsname}%
3419 }%
3420 \protected@edef\theHglsentrycounter{\@gls@Hcounter}%
3421 \restorecounters@
3422 \fi
3423 }%
3424 }%
3425 {%
```

Not using equation counter so no special measures:

```
3426 \edef\theglsentrycounter{\expandafter\noexpand
3427 \csname the\@gls@counter\endcsname}%
3428 }%
```

Check if hyperref version of this counter

```
3429 \ifx\@gls@Hcounter\@empty
3430 \ifcsundef{theH\@gls@counter}%
3431 {%
3432 \def\theHglsentrycounter{\theglsentrycounter}%
3433 }%
3434 {%
```

```

3435     \protected@edef\theHglentrycounter{\expandafter\noexpand
3436     \csname theH\@gls@counter\endcsname}%
3437   }%
3438 \fi
3439 }

```

`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.

```

3440 \def\@set@glo@numformat#1#2#3#4{%
3441   \expandafter\@glo@check@mkidxrangear#3\@nil
3442   \protected@edef#1{%
3443     \@glo@prefix setentrycounter[#4]{#2}%
3444     \expandafter\string\csname\@glo@suffix\endcsname
3445   }%
3446   \@gls@checkmkidxchars#1%
3447 }

```

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.

```

3448 \def\@glo@check@mkidxrangear#1#2\@nil{%
3449 \if#1(\relax
3450   \def\@glo@prefix{(%}
3451   \if\relax#2\relax
3452     \def\@glo@suffix{glsnumberformat}%
3453   \else
3454     \def\@glo@suffix{#2}%
3455   \fi
3456 \else
3457   \if#1)\relax
3458     \def\@glo@prefix{)}%
3459     \if\relax#2\relax
3460       \def\@glo@suffix{glsnumberformat}%
3461     \else
3462       \def\@glo@suffix{#2}%
3463     \fi
3464   \else
3465     \def\@glo@prefix{}\def\@glo@suffix{#1#2}%
3466   \fi
3467 \fi}

```

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

```

3468 \newcommand*\@gls@escbsdq[1]{%
3469   \def\@gls@checkedmkidx{%
3470     \let\gls@xdystring=#1\relax

```

```

3471 \@onelevel@sanitize\gls@xdystring
3472 \edef\do@gls@xdycheckbackslash{%
3473   \noexpand\@gls@xdycheckbackslash\gls@xdystring\noexpand\@nil
3474   \@backslashchar\@backslashchar\noexpand\null}%
3475 \do@gls@xdycheckbackslash
3476 \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdystring}%
3477 \def\@gls@checkedmkidx{%
3478 \expandafter\@gls@xdycheckquote\gls@xdystring\@nil""\null
3479 \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdystring}%

```

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

```

3480 \@for\@gls@tmp:=\gls@protected@pagefmts\do
3481 {%
3482   \edef\@gls@sanitized@tmp{\expandafter\@gobble\string\\ \expandonce\@gls@tmp}%
3483   \@onelevel@sanitize\@gls@sanitized@tmp
3484   \edef\gls@dostsubst{%
3485     \noexpand\DTLsubstituteall\noexpand\gls@xdystring
3486     {\@gls@sanitized@tmp}{\expandonce\@gls@tmp}%
3487   }%
3488   \gls@dostsubst
3489 }%

```

Assign to required control sequence

```

3490 \let#1=\gls@xdystring
3491 }

```

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

checkmkidxchars

```

3492 \newcommand{\@gls@checkmkidxchars}[1]{%
3493   \ifglxindy
3494     \@gls@escbsdq{#1}%
3495   \else
3496     \def\@gls@checkedmkidx{%
3497       \expandafter\@gls@checkquote#1\@nil""\null
3498       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3499     \def\@gls@checkedmkidx{%
3500       \expandafter\@gls@checkescquote#1\@nil\""\null
3501       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3502     \def\@gls@checkedmkidx{%
3503       \expandafter\@gls@checkescactual#1\@nil\?\?\null
3504       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3505     \def\@gls@checkedmkidx{%
3506       \expandafter\@gls@checkactual#1\@nil??\null
3507       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3508     \def\@gls@checkedmkidx{%
3509       \expandafter\@gls@checkbar#1\@nil||\null
3510       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3511     \def\@gls@checkedmkidx{%

```

```

3512 \expandafter\@gls@checkescbar#1\@nil\|\|\null
3513 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3514 \def\@gls@checkedmkidx{}%
3515 \expandafter\@gls@checklevel#1\@nil!!\null
3516 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3517 \fi
3518 }

```

Update the control sequence and strip trailing \@nil:

s@updatechecked

```
3519 \def\@gls@updatechecked#1\@nil#2{\def#2{#1}}
```

\@gls@tmpb Define temporary token

```
3520 \newtoks\@gls@tmpb
```

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

```

3521 \def\@gls@checkquote#1"#2"#3\null{%
3522 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3523 \toks@={#1}%
3524 \ifx\null#2\null
3525 \ifx\null#3\null
3526 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3527 \def\@gls@checkquote{\relax}%
3528 \else
3529 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3530 \@gls@quotechar\@gls@quotechar\@gls@quotechar\@gls@quotechar}%
3531 \def\@gls@checkquote{\@gls@checkquote#3\null}%
3532 \fi
3533 \else
3534 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3535 \@gls@quotechar\@gls@quotechar}%
3536 \ifx\null#3\null
3537 \def\@gls@checkquote{\@gls@checkquote#2""\null}%
3538 \else
3539 \def\@gls@checkquote{\@gls@checkquote#2"#3\null}%
3540 \fi
3541 \fi
3542 \@gls@checkquote
3543 }

```

s@checkescquote Do the same for \":

```

3544 \def\@gls@checkescquote#1\"#2\"#3\null{%
3545 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3546 \toks@={#1}%
3547 \ifx\null#2\null
3548 \ifx\null#3\null
3549 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3550 \def\@gls@checkescquote{\relax}%

```

```

3551 \else
3552 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3553 \@gls@quotechar\string\\"\@gls@quotechar
3554 \@gls@quotechar\string\\"\@gls@quotechar}%
3555 \def\@@gls@checkescquote{\@gls@checkescquote#3\null}%
3556 \fi
3557 \else
3558 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3559 \@gls@quotechar\string\\"\@gls@quotechar}%
3560 \ifx\null#3\null
3561 \def\@@gls@checkescquote{\@gls@checkescquote#2\\"\null}%
3562 \else
3563 \def\@@gls@checkescquote{\@gls@checkescquote#2\#"#3\null}%
3564 \fi
3565 \fi
3566 \@@gls@checkescquote
3567 }

```

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

```

3568 \def\@gls@checkescactual#1\?#2\?#3\null{%
3569 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3570 \toks@={#1}%
3571 \ifx\null#2\null
3572 \ifx\null#3\null
3573 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3574 \def\@@gls@checkescactual{\relax}%
3575 \else
3576 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3577 \@gls@quotechar\string\\"\@gls@actualchar
3578 \@gls@quotechar\string\\"\@gls@actualchar}%
3579 \def\@@gls@checkescactual{\@gls@checkescactual#3\null}%
3580 \fi
3581 \else
3582 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3583 \@gls@quotechar\string\\"\@gls@actualchar}%
3584 \ifx\null#3\null
3585 \def\@@gls@checkescactual{\@gls@checkescactual#2\?\?\null}%
3586 \else
3587 \def\@@gls@checkescactual{\@gls@checkescactual#2\?#3\null}%
3588 \fi
3589 \fi
3590 \@@gls@checkescactual
3591 }

```

`gls@checkescbar` Similarly for \|:

```

3592 \def\@gls@checkescbar#1\|#2\|#3\null{%
3593 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3594 \toks@={#1}%
3595 \ifx\null#2\null

```



```

3596 \ifx\null#3\null
3597 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3598 \def\@gls@checkesbar{\relax}%
3599 \else
3600 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3601 \@gls@quotechar\string\\"\@gls@encapchar
3602 \@gls@quotechar\string\\"\@gls@encapchar}%
3603 \def\@gls@checkesbar{\@gls@checkesbar#3\null}%
3604 \fi
3605 \else
3606 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3607 \@gls@quotechar\string\\"\@gls@encapchar}%
3608 \ifx\null#3\null
3609 \def\@gls@checkesbar{\@gls@checkesbar#2\|\|\null}%
3610 \else
3611 \def\@gls@checkesbar{\@gls@checkesbar#2\|#3\null}%
3612 \fi
3613 \fi
3614 \@gls@checkesbar
3615 }

```

s@checkeslevel Similarly for \!:

```

3616 \def\@gls@checkeslevel#1\!#2\!#3\null{%
3617 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3618 \toks@={#1}%
3619 \ifx\null#2\null
3620 \ifx\null#3\null
3621 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3622 \def\@gls@checkeslevel{\relax}%
3623 \else
3624 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3625 \@gls@quotechar\string\\"\@gls@levelchar
3626 \@gls@quotechar\string\\"\@gls@levelchar}%
3627 \def\@gls@checkeslevel{\@gls@checkeslevel#3\null}%
3628 \fi
3629 \else
3630 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3631 \@gls@quotechar\string\\"\@gls@levelchar}%
3632 \ifx\null#3\null
3633 \def\@gls@checkeslevel{\@gls@checkeslevel#2\!\!\null}%
3634 \else
3635 \def\@gls@checkeslevel{\@gls@checkeslevel#2\!#3\null}%
3636 \fi
3637 \fi
3638 \@gls@checkeslevel
3639 }

```

\@gls@checkbar and for |:

```

3640 \def\@gls@checkbar#1|#2|#3\null{%

```

```

3641 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3642 \toks@={#1}%
3643 \ifx\null#2\null
3644 \ifx\null#3\null
3645 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3646 \def\@gls@checkbar{\relax}%
3647 \else
3648 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3649 \@gls@quotechar\@gls@encapchar\@gls@quotechar\@gls@encapchar}%
3650 \def\@gls@checkbar{\@gls@checkbar#3\null}%
3651 \fi
3652 \else
3653 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3654 \@gls@quotechar\@gls@encapchar}%
3655 \ifx\null#3\null
3656 \def\@gls@checkbar{\@gls@checkbar#2||\null}%
3657 \else
3658 \def\@gls@checkbar{\@gls@checkbar#2|#3\null}%
3659 \fi
3660 \fi
3661 \@gls@checkbar
3662 }

```

@gls@checklevel and for !:

```

3663 \def\@gls@checklevel#1!#2!#3\null{%
3664 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3665 \toks@={#1}%
3666 \ifx\null#2\null
3667 \ifx\null#3\null
3668 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3669 \def\@gls@checklevel{\relax}%
3670 \else
3671 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3672 \@gls@quotechar\@gls@levelchar\@gls@quotechar\@gls@levelchar}%
3673 \def\@gls@checklevel{\@gls@checklevel#3\null}%
3674 \fi
3675 \else
3676 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3677 \@gls@quotechar\@gls@levelchar}%
3678 \ifx\null#3\null
3679 \def\@gls@checklevel{\@gls@checklevel#2!!\null}%
3680 \else
3681 \def\@gls@checklevel{\@gls@checklevel#2!#3\null}%
3682 \fi
3683 \fi
3684 \@gls@checklevel
3685 }

```

gls@checkactual and for ?:

```

3686 \def\@gls@checkactual#1?#2?#3\null{%
3687 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3688 \toks@={#1}%
3689 \ifx\null#2\null
3690 \ifx\null#3\null
3691 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3692 \def\@gls@checkactual{\relax}%
3693 \else
3694 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3695 \@gls@quotechar\@gls@actualchar\@gls@quotechar\@gls@actualchar}%
3696 \def\@gls@checkactual{\@gls@checkactual#3\null}%
3697 \fi
3698 \else
3699 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3700 \@gls@quotechar\@gls@actualchar}%
3701 \ifx\null#3\null
3702 \def\@gls@checkactual{\@gls@checkactual#2??\null}%
3703 \else
3704 \def\@gls@checkactual{\@gls@checkactual#2?#3\null}%
3705 \fi
3706 \fi
3707 \@gls@checkactual
3708 }

```

s@xdycheckquote As before but for use with xindy

```

3709 \def\@gls@xdycheckquote#1"#2"#3\null{%
3710 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3711 \toks@={#1}%
3712 \ifx\null#2\null
3713 \ifx\null#3\null
3714 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3715 \def\@gls@xdycheckquote{\relax}%
3716 \else
3717 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3718 \string\#\string\}%
3719 \def\@gls@xdycheckquote{\@gls@xdycheckquote#3\null}%
3720 \fi
3721 \else
3722 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3723 \string\}%
3724 \ifx\null#3\null
3725 \def\@gls@xdycheckquote{\@gls@xdycheckquote#2""\null}%
3726 \else
3727 \def\@gls@xdycheckquote{\@gls@xdycheckquote#2"#3\null}%
3728 \fi
3729 \fi
3730 \@gls@xdycheckquote
3731 }

```

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

```
3732 \edef\def@gls@xdycheckbackslash{%
3733 \noexpand\def\noexpand@gls@xdycheckbackslash##1\@backslashchar
3734 ##2\@backslashchar##3\noexpand\null{%
3735 \noexpand@gls@tmpb=\noexpand\expandafter
3736 {\noexpand@gls@checkedmkidx}%
3737 \noexpand\toks@={##1}%
3738 \noexpand\ifx\noexpand\null##2\noexpand\null
3739 \noexpand\ifx\noexpand\null##3\noexpand\null
3740 \noexpand\edef\noexpand@gls@checkedmkidx{%
3741 \noexpand\the\noexpand@gls@tmpb\noexpand\the\noexpand\toks@}%
3742 \noexpand\def\noexpand\@gls@xdycheckbackslash{\relax}%
3743 \noexpand\else
3744 \noexpand\edef\noexpand@gls@checkedmkidx{%
3745 \noexpand\the\noexpand@gls@tmpb\noexpand\the\noexpand\toks@
3746 \@backslashchar\@backslashchar\@backslashchar\@backslashchar}%
3747 \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3748 \noexpand@gls@xdycheckbackslash##3\noexpand\null}%
3749 \noexpand\fi
3750 \noexpand\else
3751 \noexpand\edef\noexpand@gls@checkedmkidx{%
3752 \noexpand\the\noexpand@gls@tmpb\noexpand\the\noexpand\toks@
3753 \@backslashchar\@backslashchar}%
3754 \noexpand\ifx\noexpand\null##3\noexpand\null
3755 \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3756 \noexpand@gls@xdycheckbackslash##2\@backslashchar
3757 \@backslashchar\noexpand\null}%
3758 \noexpand\else
3759 \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3760 \noexpand@gls@xdycheckbackslash##2\@backslashchar
3761 ##3\noexpand\null}%
3762 \noexpand\fi
3763 \noexpand\fi
3764 \noexpand\@gls@xdycheckbackslash
3765 }%
3766 }
```

Now go ahead and define \@gls@xdycheckbackslash

```
3767 \def@gls@xdycheckbackslash
```

lstdohypertarget

```
3768 \newlength@gls@tmplen
3769 \newcommand*{\glsdohypertarget}[2]{%
3770 \@gls@showtarget{#1}%
3771 \settoheight@gls@tmplen{#2}%
3772 \raisebox@gls@tmplen{\hypertarget{#1}{}}#2%
3773 }
```

\glsdohyperlink

```

3774 \newcommand*\glsdohyperlink}[2]{%
3775 \@glsshowtarget{#1}%
3776 \hyperlink{#1}{#2}%
3777 }

```

`\glsdonohyperlink`

```

3778 \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`.

```

3779 \ifcsundef{hyperlink}%
3780 {%
3781 \let\@glslink\glsdonohyperlink
3782 }%
3783 {%
3784 \let\@glslink\glsdohyperlink
3785 }

```

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

```

3786 \ifcsundef{hypertarget}%
3787 {%
3788 \let\@glstarget\@secondoftwo
3789 }%
3790 {%
3791 \let\@glstarget\glsdohypertarget
3792 }

```

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

`\glsdisablehyper`

```

3793 \newcommand{\glsdisablehyper}{%
3794 \KV@glslink@hyperfalse
3795 \let\@glslink\glsdonohyperlink
3796 \let\@glstarget\@secondoftwo
3797 }

```

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

`\glsenablehyper`

```

3798 \newcommand{\glsenablehyper}{%
3799 \KV@glslink@hypertrue
3800 \let\@glslink\glsdohyperlink
3801 \let\@glstarget\glsdohypertarget
3802 }

```

Provide some convenience commands if not already defined:

```

3803 \providecommand{\@firstofthree}[3]{#1}
3804 \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`

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

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

`\@gls`

```
3806 \newcommand*{\@gls}[2] [] {%
3807   \new@ifnextchar[{\@gls@{#1}{#2}}{\@gls@{#1}{#2} []}]%
3808 }
```

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

```
3809 \def\@gls@#1#2[#3]{%
3810   \glsdoifexists{#2}%
3811   {%
3812     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
3813     \let\glsifplural\@secondoftwo
3814     \let\glsapscase\@firstofthree
3815     \let\glscustomtext\@empty
3816     \def\glsinsert{#3}%
```

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

```
3817   \def\@gls@text{\csname gls@\gls@type @entryfmt\endcsname}%
```

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

```
3818   \@gls@link[#1]{#2}{\@gls@text}%
```

Indicate that this entry has now been used

```
3819   \ifKV@glslink@local
3820     \glslocalunset{#2}%
3821   \else
3822     \glsunset{#2}%
```

```

3823   \fi
3824 }%

3825 \glspostlinkhook
3826 }

```

`\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`

```

3827 \newrobustcmd*{\Gls}{\@gls@hyp@opt\@Gls}

```

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

```

3828 \newcommand*{\@Gls}[2][\@Gls@{#1}{#2}]{\@Gls@{#1}{#2}[]}%
3829 \new@ifnextchar[\@Gls@{#1}{#2}]{\@Gls@{#1}{#2}[]}%
3830 }

```

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

```

3831 \def\@Gls@#1#2[#3]{%
3832   \glsdoifexists{#2}%
3833   {%
3834     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3835     \let\glsifplural\@secondoftwo
3836     \let\glsupcase\@secondofthree
3837     \let\glscustomtext\@empty
3838     \def\glsinsert{#3}%

```

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

```

3839   \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.

```

3840   \@gls@link[#1]{#2}{\@glo@text}%

```

Indicate that this entry has now been used

```

3841   \ifKV@glslink@local
3842     \glslocalunset{#2}%
3843   \else
3844     \glsunset{#2}%
3845   \fi
3846 }%

```

```

3847 \glspostlinkhook
3848 }

```

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

`\GLS`

```
3849 \newrobustcmd*{\GLS}{\@gls@hyp@opt\@GLS}
```

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

```
3850 \newcommand*{\@GLS}[2] [] {%
3851   \new@ifnextchar[{\@GLS@{#1}{#2}}{\@GLS@{#1}{#2} []}%
3852 }
```

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

```
3853 \def\@GLS@#1#2[#3] {%
3854   \glsdoifexists{#2}%
3855   {%
3856     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
3857     \let\glsifplural\@secondoftwo
3858     \let\glsapscase\@thirdofthree
3859     \let\glscustomtext\@empty
3860     \def\glsinsert{#3}%
```

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

```
3861   \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.

```
3862   \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3863   \ifKV@gls@link@local
3864     \glslocalunset{#2}%
3865   \else
3866     \glsunset{#2}%
3867   \fi
3868 }%
```

```
3869 \glspostlinkhook
3870 }
```

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

`\glspl`

```
3871 \newrobustcmd*{\glspl}{\@gls@hyp@opt\@glspl}
```

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

```
3872 \newcommand*{\@glspl}[2] [] {%
3873   \new@ifnextchar[{\@glspl@{#1}{#2}}{\@glspl@{#1}{#2} []}%
3874 }
```


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

```
3875 \def\@glspl@#1#2[#3]{%
3876   \glsdoifexists{#2}%
3877   {%
3878     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3879     \let\glsifplural\@firstoftwo
3880     \let\glsapscase\@firstofthree
3881     \let\glscustomtext\@empty
3882     \def\glsinsert{#3}%
```

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

```
3883   \def\@glo@text{\csname gls@\gls@type @entryfmt\endcsname}%
```

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

```
3884   \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3885   \ifKV@gls@link@local
3886     \glslocalunset{#2}%
3887   \else
3888     \glsunset{#2}%
3889   \fi
3890 }%
```

```
3891 \glspostlinkhook
3892 }
```

`\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`

```
3893 \newrobustcmd*{\Glspl}{\@gls@hyp@opt\@Glspl}
```

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

```
3894 \newcommand*{\@Glspl}[2][ ]{%
3895   \new@ifnextchar[{\@Glspl@{#1}{#2}}{\@Glspl@{#1}{#2} [ ]}%
3896 }
```

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

```
3897 \def\@Glspl@#1#2[#3]{%
3898   \glsdoifexists{#2}%
3899   {%
3900     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3901     \let\glsifplural\@firstoftwo
3902     \let\glsapscase\@secondofthree
3903     \let\glscustomtext\@empty
3904     \def\glsinsert{#3}%
```

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

```

3905 \def\@glo@text{\csname gls@\gls@type @entryfmt\endcsname}%
Call \@gls@link. If footnote package option has been used and the glossary type is
\acronymtype, suppress hyperlink for first use. Likewise if the hyperfirst=false package op-
tion is used.
3906 \@gls@link[#1]{#2}{\@glo@text}%
Indicate that this entry has now been used
3907 \ifKV@gls@link@local
3908 \glslocalunset{#2}%
3909 \else
3910 \glsunset{#2}%
3911 \fi
3912 }%

3913 \gls@postlinkhook
3914 }

```

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

`\GLSp1`

```

3915 \newrobustcmd*{\GLSp1}{\@gls@hyp@opt\@GLSp1}

```

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

```

3916 \newcommand*{\@GLSp1}[2] [] {%
3917 \new@ifnextchar[{\@GLSp1@{#1}{#2}}{\@GLSp1@{#1}{#2} []}]%
3918 }

```

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

```

3919 \def\@GLSp1@#1#2[#3] {%
3920 \glsdoifexists{#2}%
3921 {%
3922 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3923 \let\glsifplural\@firstoftwo
3924 \let\gls@caps@case\@thirdofthree
3925 \let\gls@custom@text\@empty
3926 \def\glsinsert{#3}%

```

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

```

3927 \def\@glo@text{\csname gls@\gls@type @entryfmt\endcsname}%
Call \@gls@link. If footnote package option has been used and the glossary type is
\acronymtype, suppress hyperlink for first use. Likewise if the hyperfirst=false package op-
tion is used.
3928 \@gls@link[#1]{#2}{\@glo@text}%

```

Indicate that this entry has now been used

```
3929 \ifKV@glslink@local
3930 \glslocalunset{#2}%
3931 \else
3932 \glsunset{#2}%
3933 \fi
3934 }%

3935 \glspostlinkhook
3936 }
```

`\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:

```
3937 \newrobustcmd*{\glsdisp}{\@gls@hyp@opt\@glsdisp}
```

Defined the un-starred form.

`\@glsdisp`

```
3938 \newcommand*{\@glsdisp}[3] [] {%
3939 \glsdoifexists{#2}{%

3940 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3941 \let\glsifplural\@secondoftwo
3942 \let\glsapscase\@firstofthree
3943 \def\glscustomtext{#3}%
3944 \def\glsinsert{}}%
```

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

```
3945 \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.

```
3946 \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3947 \ifKV@glslink@local
3948 \glslocalunset{#2}%
3949 \else
3950 \glsunset{#2}%
3951 \fi
3952 }%

3953 \glspostlinkhook
3954 }
```

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

```
3955 \newcommand*{\@gls@link@nocheckfirsthyper}{}
```

@gls@field@link

```
3956 \newcommand{\@gls@field@link}[3]{%
3957   \glsdoifexists{#2}%
3958   {%
3959     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
3960     \@gls@link[#1]{#2}{#3}%
3961   }%

3962   \glspostlinkhook
3963 }
```

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

`\gls@text`

```
3964 \newrobustcmd*{\gls@text}{\@gls@hyp@opt\@gls@text}
```

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

```
3965 \newcommand*{\@gls@text}[2] [] {%
3966   \new@ifnextchar[{\@gls@text@{#1}{#2}}{\@gls@text@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3967 \def\@gls@text@#1#2[#3] {%
3968   \@gls@field@link{#1}{#2}{\gls@entrytext{#2}#3}%
3969 }
```

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

`\GLStext`

```
3970 \newrobustcmd*{\GLStext}{\@gls@hyp@opt\@GLStext}
```

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

```
3971 \newcommand*{\@GLStext}[2] [] {%
3972   \new@ifnextchar[{\@GLStext@{#1}{#2}}{\@GLStext@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3973 \def\@GLStext@#1#2[#3] {%
3974   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\gls@entrytext{#2}#3}}%
3975 }
```

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

`\Glstext`

```
3976 \newrobustcmd*{\Glstext}{\@gls@hyp@opt\@Glstext}
```

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

```
3977 \newcommand*{\@Glstext}[2] [] {%
3978   \new@ifnextchar[{\@Glstext@{#1}{#2}}{\@Glstext@{#1}{#2} [] ]}}
```

Read in the final optional argument:

```
3979 \def\@Glstext@#1#2[#3] {%
3980   \@gls@field@link{#1}{#2}{\Glsentrytext{#2}#3}%
3981 }
```

`\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`

```
3982 \newrobustcmd*{\glsfirst}{\@gls@hyp@opt\@glsfirst}
```

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

```
3983 \newcommand*{\@glsfirst}[2] [] {%
3984   \new@ifnextchar[{\@glsfirst@{#1}{#2}}{\@glsfirst@{#1}{#2} [] ]}}
```

Read in the final optional argument:

```
3985 \def\@glsfirst@#1#2[#3] {%
3986   \@gls@field@link{#1}{#2}{\glsentryfirst{#2}#3}%
3987 }
```

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

`\Glsfirst`

```
3988 \newrobustcmd*{\Glsfirst}{\@gls@hyp@opt\@Glsfirst}
```

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

```
3989 \newcommand*{\@Glsfirst}[2] [] {%
3990   \new@ifnextchar[{\@Glsfirst@{#1}{#2}}{\@Glsfirst@{#1}{#2} [] ]}}
```

Read in the final optional argument:

```
3991 \def\@Glsfirst@#1#2[#3] {%
3992   \@gls@field@link{#1}{#2}{\Glsentryfirst{#2}#3}%
3993 }
```

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

`\GLSfirst`

```
3994 \newrobustcmd*{\GLSfirst}{\@gls@hyp@opt\@GLSfirst}
```

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

```
3995 \newcommand*{\@GLSfirst}[2] [] {%
3996   \new@ifnextchar[{\@GLSfirst@{#1}{#2}}{\@GLSfirst@{#1}{#2} [] ]}}
```

Read in the final optional argument:

```
3997 \def\@GLSfirst@#1#2[#3] {%
3998   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryfirst{#2}#3}}%
3999 }
```

`\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`

```
4000 \newrobustcmd*{\glsplural}{\@gls@hyp@opt\@glsplural}
```

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

```
4001 \newcommand*{\@glsplural}[2] [] {%
```

```
4002 \new@ifnextchar[{\@glsplural@{#1}{#2}}{\@glsplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4003 \def\@glsplural@#1#2[#3] {%
```

```
4004 \@gls@field@link{#1}{#2}{\glsentryplural{#2}#3}%
```

```
4005 }
```

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

`\Glsplural`

```
4006 \newrobustcmd*{\Glsplural}{\@gls@hyp@opt\@Glsplural}
```

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

```
4007 \newcommand*{\@Glsplural}[2] [] {%
```

```
4008 \new@ifnextchar[{\@Glsplural@{#1}{#2}}{\@Glsplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4009 \def\@Glsplural@#1#2[#3] {%
```

```
4010 \@gls@field@link{#1}{#2}{\Glsentryplural{#2}#3}%
```

```
4011 }
```

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

`\GLSplural`

```
4012 \newrobustcmd*{\GLSplural}{\@gls@hyp@opt\@GLSplural}
```

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

```
4013 \newcommand*{\@GLSplural}[2] [] {%
```

```
4014 \new@ifnextchar[{\@GLSplural@{#1}{#2}}{\@GLSplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4015 \def\@GLSplural@#1#2[#3] {%
```

```
4016 \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryplural{#2}#3}}%
```

```
4017 }
```

`\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`

```
4018 \newrobustcmd*{\glsfirstplural}{\@gls@hyp@opt\@glsfirstplural}
```

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

```
4019 \newcommand*{\@glsfirstplural}[2] [] {%
```

```
4020 \new@ifnextchar[{\@glsfirstplural@{#1}{#2}}{\@glsfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4021 \def\@glsfirstplural@#1#2[#3] {%
```

```
4022 \@gls@field@link{#1}{#2}{\glsentryfirstplural{#2}#3}%
```

```
4023 }
```

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

`\Glsfirstplural`

```
4024 \newrobustcmd*{\Glsfirstplural}{\@gls@hyp@opt\@Glsfirstplural}
```

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

```
4025 \newcommand*{\@Glsfirstplural}[2] [] {%
```

```
4026 \new@ifnextchar[{\@Glsfirstplural@{#1}{#2}}{\@Glsfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4027 \def\@Glsfirstplural@#1#2[#3] {%
```

```
4028 \@gls@field@link{#1}{#2}{\glsentryfirstplural{#2}#3}%
```

```
4029 }
```

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

`\GLSfirstplural`

```
4030 \newrobustcmd*{\GLSfirstplural}{\@gls@hyp@opt\@GLSfirstplural}
```

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

```
4031 \newcommand*{\@GLSfirstplural}[2] [] {%
```

```
4032 \new@ifnextchar[{\@GLSfirstplural@{#1}{#2}}{\@GLSfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4033 \def\@GLSfirstplural@#1#2[#3] {%
```

```
4034 \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryfirstplural{#2}#3}}%
```

```
4035 }
```

`\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`

```
4036 \newrobustcmd*{\glsname}{\@gls@hyp@opt\@glsname}
```

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

```
4037 \newcommand*{\@glsname}[2] [] {%
```

```
4038 \new@ifnextchar[{\@glsname@{#1}{#2}}{\@glsname@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4039 \def\@glsname@#1#2[#3] {%
```

```
4040 \@gls@field@link{#1}{#2}{\glsentryname{#2}#3}%
```

```
4041 }
```

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

`\Glsname`

```
4042 \newrobustcmd*{\Glsname}{\@gls@hyp@opt\@Glsname}
```

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

```
4043 \newcommand*{\@Glsname}[2] [] {%
```

```
4044 \new@ifnextchar[{\@Glsname@{#1}{#2}}{\@Glsname@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4045 \def\@Glsname@#1#2[#3]{%
4046 \@gls@field@link{#1}{#2}{\Glsentryname{#2}#3}%
4047 }
```

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

\GLSname

```
4048 \newrobustcmd*{\GLSname}{\@gls@hyp@opt\@GLSname}
```

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

```
4049 \newcommand*{\@GLSname}[2] [] {%
4050 \new@ifnextchar[{\@GLSname@{#1}{#2}}{\@GLSname@{#1}{#2} []]}
```

Read in the final optional argument:

```
4051 \def\@GLSname@#1#2[#3]{%
4052 \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentryname{#2}#3}}%
4053 }
```

\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

```
4054 \newrobustcmd*{\glsdesc}{\@gls@hyp@opt\@glsdesc}
```

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

```
4055 \newcommand*{\@glsdesc}[2] [] {%
4056 \new@ifnextchar[{\@glsdesc@{#1}{#2}}{\@glsdesc@{#1}{#2} []]}
```

Read in the final optional argument:

```
4057 \def\@glsdesc@#1#2[#3]{%
4058 \@gls@field@link{#1}{#2}{\Glsentrydesc{#2}#3}%
4059 }
```

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

\Glsdesc

```
4060 \newrobustcmd*{\Glsdesc}{\@gls@hyp@opt\@Glsdesc}
```

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

```
4061 \newcommand*{\@Glsdesc}[2] [] {%
4062 \new@ifnextchar[{\@Glsdesc@{#1}{#2}}{\@Glsdesc@{#1}{#2} []]}
```

Read in the final optional argument:

```
4063 \def\@Glsdesc@#1#2[#3]{%
4064 \@gls@field@link{#1}{#2}{\Glsentrydesc{#2}#3}%
4065 }
```

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

\GLSdesc

```
4066 \newrobustcmd*{\GLSdesc}{\@gls@hyp@opt\@GLSdesc}
```


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

```
4067 \newcommand*{\@GLSdesc}[2] [] {%
4068   \new@ifnextchar [{\@GLSdesc@{#1}{#2}}{\@GLSdesc@{#1}{#2} []]}
```

Read in the final optional argument:

```
4069 \def\@GLSdesc@#1#2[#3] {%
4070   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrydesc{#2}#3}}%
4071 }
```

`\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`

```
4072 \newrobustcmd*{\glsdescplural}{\@gls@hyp@opt\@glsdescplural}
```

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

```
4073 \newcommand*{\@glsdescplural}[2] [] {%
4074   \new@ifnextchar [{\@glsdescplural@{#1}{#2}}{\@glsdescplural@{#1}{#2} []]}
```

Read in the final optional argument:

```
4075 \def\@glsdescplural@#1#2[#3] {%
4076   \@gls@field@link{#1}{#2}{\glsentrydescplural{#2}#3}}%
4077 }
```

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

`\Glsdescplural`

```
4078 \newrobustcmd*{\Glsdescplural}{\@gls@hyp@opt\@Glsdescplural}
```

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

```
4079 \newcommand*{\@Glsdescplural}[2] [] {%
4080   \new@ifnextchar [{\@Glsdescplural@{#1}{#2}}{\@Glsdescplural@{#1}{#2} []]}
```

Read in the final optional argument:

```
4081 \def\@Glsdescplural@#1#2[#3] {%
4082   \@gls@field@link{#1}{#2}{\Glsentrydescplural{#2}#3}}%
4083 }
```

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

`\GLSdescplural`

```
4084 \newrobustcmd*{\GLSdescplural}{\@gls@hyp@opt\@GLSdescplural}
```

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

```
4085 \newcommand*{\@GLSdescplural}[2] [] {%
4086   \new@ifnextchar [{\@GLSdescplural@{#1}{#2}}{\@GLSdescplural@{#1}{#2} []]}
```

Read in the final optional argument:

```
4087 \def\@GLSdescplural@#1#2[#3] {%
4088   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrydescplural{#2}#3}}%
4089 }
```

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

`\glssymbol`

```
4090 \newrobustcmd*{\glssymbol}{\@gls@hyp@opt\@glssymbol}
```

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

```
4091 \newcommand*{\@glssymbol}[2] [] {%
```

```
4092   \new@ifnextchar[{\@glssymbol@{#1}{#2}}{\@glssymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4093 \def\@glssymbol@#1#2[#3] {%
```

```
4094   \@gls@field@link{#1}{#2}{\glstentrysymbol{#2}#3}%
```

```
4095 }
```

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

`\Glssymbol`

```
4096 \newrobustcmd*{\Glssymbol}{\@gls@hyp@opt\@Glssymbol}
```

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

```
4097 \newcommand*{\@Glssymbol}[2] [] {%
```

```
4098   \new@ifnextchar[{\@Glssymbol@{#1}{#2}}{\@Glssymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4099 \def\@Glssymbol@#1#2[#3] {%
```

```
4100   \@gls@field@link{#1}{#2}{\glstentrysymbol{#2}#3}%
```

```
4101 }
```

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

`\GLSsymbol`

```
4102 \newrobustcmd*{\GLSsymbol}{\@gls@hyp@opt\@GLSsymbol}
```

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

```
4103 \newcommand*{\@GLSsymbol}[2] [] {%
```

```
4104   \new@ifnextchar[{\@GLSsymbol@{#1}{#2}}{\@GLSsymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4105 \def\@GLSsymbol@#1#2[#3] {%
```

```
4106   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glstentrysymbol{#2}#3}}%
```

```
4107 }
```

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

`glsymbolplural`

```
4108 \newrobustcmd*{\glsymbolplural}{\@gls@hyp@opt\@glsymbolplural}
```

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

```
4109 \newcommand*{\@glsymbolplural}[2] [] {%
```

```
4110   \new@ifnextchar[{\@glsymbolplural@{#1}{#2}}{\@glsymbolplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4111 \def\@glssymbolplural@#1#2[#3]{%
4112 \@gls@field@link{#1}{#2}{\glstryymbolplural{#2}#3}%
4113 }
```

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

`Glsymbolplural`

```
4114 \newrobustcmd*{\Glsymbolplural}{\@gls@hyp@opt\@Glsymbolplural}
```

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

```
4115 \newcommand*{\@Glsymbolplural}[2] [] {%
4116 \new@ifnextchar[{\@Glsymbolplural@{#1}{#2}}{\@Glsymbolplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4117 \def\@Glsymbolplural@#1#2[#3]{%
4118 \@gls@field@link{#1}{#2}{\glstryymbolplural{#2}#3}%
4119 }
```

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

`GLSsymbolplural`

```
4120 \newrobustcmd*{\GLSsymbolplural}{\@gls@hyp@opt\@GLSsymbolplural}
```

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

```
4121 \newcommand*{\@GLSsymbolplural}[2] [] {%
4122 \new@ifnextchar[{\@GLSsymbolplural@{#1}{#2}}{\@GLSsymbolplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4123 \def\@GLSsymbolplural@#1#2[#3]{%
4124 \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glstryymbolplural{#2}#3}}%
4125 }
```

`\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`

```
4126 \newrobustcmd*{\glsuseri}{\@gls@hyp@opt\@glsuseri}
```

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

```
4127 \newcommand*{\@glsuseri}[2] [] {%
4128 \new@ifnextchar[{\@glsuseri@{#1}{#2}}{\@glsuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4129 \def\@glsuseri@#1#2[#3]{%
4130 \@gls@field@link{#1}{#2}{\glstryuseri{#2}#3}%
4131 }
```

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

`\Glsuseri`

```
4132 \newrobustcmd*{\Glsuseri}{\@gls@hyp@opt\@Glsuseri}
```

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

```
4133 \newcommand*{\@Glsuseri}[2][\@Glsuseri]
```

```
4134 \new@ifnextchar[{\@Glsuseri@{#1}{#2}}{\@Glsuseri@{#1}{#2}[]}]
```

Read in the final optional argument:

```
4135 \def\@Glsuseri@#1#2[#3]{%
```

```
4136 \@gls@field@link{#1}{#2}{\Glsentryuseri{#2}#3}%
```

```
4137 }
```

`\GLSuseri` behaves like `\glsuseri` except that the link text is converted to uppercase.

`\GLSuseri`

```
4138 \newrobustcmd*{\GLSuseri}{\@gls@hyp@opt\@GLSuseri}
```

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

```
4139 \newcommand*{\@GLSuseri}[2][\@GLSuseri]
```

```
4140 \new@ifnextchar[{\@GLSuseri@{#1}{#2}}{\@GLSuseri@{#1}{#2}[]}]
```

Read in the final optional argument:

```
4141 \def\@GLSuseri@#1#2[#3]{%
```

```
4142 \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseri{#2}#3}_%
```

```
4143 }
```

`\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`

```
4144 \newrobustcmd*{\glsuserii}{\@gls@hyp@opt\@glsuserii}
```

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

```
4145 \newcommand*{\@glsuserii}[2][\@glsuserii]
```

```
4146 \new@ifnextchar[{\@glsuserii@{#1}{#2}}{\@glsuserii@{#1}{#2}[]}]
```

Read in the final optional argument:

```
4147 \def\@glsuserii@#1#2[#3]{%
```

```
4148 \@gls@field@link{#1}{#2}{\glsentryuserii{#2}#3}_%
```

```
4149 }
```

`\Glsuserii` behaves like `\glsuserii` except that the first letter is converted to uppercase.

`\Glsuserii`

```
4150 \newrobustcmd*{\Glsuserii}{\@gls@hyp@opt\@Glsuserii}
```

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

```
4151 \newcommand*{\@Glsuserii}[2][\@Glsuserii]
```

```
4152 \new@ifnextchar[{\@Glsuserii@{#1}{#2}}{\@Glsuserii@{#1}{#2}[]}]
```

Read in the final optional argument:

```
4153 \def\@Glsuserii@#1#2[#3]{%
```

```
4154 \@gls@field@link{#1}{#2}{\Glsentryuserii{#2}#3}_%
```

```
4155 }
```

`\GLSuserii` behaves like `\glsuserii` except that the link text is converted to uppercase.

`\GLSuserii`

```
4156 \newrobustcmd*{\GLSuserii}{\@gls@hyp@opt\@GLSuserii}
```

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

```
4157 \newcommand*{\@GLSuserii}[2][\@GLSuserii@#1]{\@GLSuserii@#1}{\@GLSuserii@#1}{\@GLSuserii@#1}{\@GLSuserii@#1}{\@GLSuserii@#1}{\@GLSuserii@#1}}
```

```
4158 \new@ifnextchar[{\@GLSuserii@#1}{\@GLSuserii@#1}{\@GLSuserii@#1}{\@GLSuserii@#1}{\@GLSuserii@#1}{\@GLSuserii@#1}}
```

Read in the final optional argument:

```
4159 \def\@GLSuserii@#1#2[#3]{\@GLSuserii@#1}{\@GLSuserii@#1}{\@GLSuserii@#1}{\@GLSuserii@#1}{\@GLSuserii@#1}{\@GLSuserii@#1}}
```

```
4160 \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuserii{#2}#3}}%
```

```
4161 }
```

`\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`

```
4162 \newrobustcmd*{\glsuseriii}{\@gls@hyp@opt\@glsuseriii}
```

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

```
4163 \newcommand*{\@glsuseriii}[2][\@glsuseriii@#1]{\@glsuseriii@#1}{\@glsuseriii@#1}{\@glsuseriii@#1}{\@glsuseriii@#1}{\@glsuseriii@#1}{\@glsuseriii@#1}}
```

```
4164 \new@ifnextchar[{\@glsuseriii@#1}{\@glsuseriii@#1}{\@glsuseriii@#1}{\@glsuseriii@#1}{\@glsuseriii@#1}{\@glsuseriii@#1}}
```

Read in the final optional argument:

```
4165 \def\@glsuseriii@#1#2[#3]{\@glsuseriii@#1}{\@glsuseriii@#1}{\@glsuseriii@#1}{\@glsuseriii@#1}{\@glsuseriii@#1}{\@glsuseriii@#1}}
```

```
4166 \@gls@field@link{#1}{#2}{\glsentryuseriii{#2}#3}}%
```

```
4167 }
```

`\Glsuseriii` behaves like `\glsuseriii` except that the first letter is converted to uppercase.

`\Glsuseriii`

```
4168 \newrobustcmd*{\Glsuseriii}{\@gls@hyp@opt\@Glsuseriii}
```

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

```
4169 \newcommand*{\@Glsuseriii}[2][\@Glsuseriii@#1]{\@Glsuseriii@#1}{\@Glsuseriii@#1}{\@Glsuseriii@#1}{\@Glsuseriii@#1}{\@Glsuseriii@#1}{\@Glsuseriii@#1}}
```

```
4170 \new@ifnextchar[{\@Glsuseriii@#1}{\@Glsuseriii@#1}{\@Glsuseriii@#1}{\@Glsuseriii@#1}{\@Glsuseriii@#1}{\@Glsuseriii@#1}}
```

Read in the final optional argument:

```
4171 \def\@Glsuseriii@#1#2[#3]{\@Glsuseriii@#1}{\@Glsuseriii@#1}{\@Glsuseriii@#1}{\@Glsuseriii@#1}{\@Glsuseriii@#1}{\@Glsuseriii@#1}}
```

```
4172 \@gls@field@link{#1}{#2}{\Glsentryuseriii{#2}#3}}%
```

```
4173 }
```

`\GLSuseriii` behaves like `\glsuseriii` except that the link text is converted to uppercase.

`\GLSuseriii`

```
4174 \newrobustcmd*{\GLSuseriii}{\@gls@hyp@opt\@GLSuseriii}
```

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

```
4175 \newcommand*{\@GLSuseriii}[2][\@GLSuseriii@#1]{\@GLSuseriii@#1}{\@GLSuseriii@#1}{\@GLSuseriii@#1}{\@GLSuseriii@#1}{\@GLSuseriii@#1}{\@GLSuseriii@#1}}
```

```
4176 \new@ifnextchar[{\@GLSuseriii@#1}{\@GLSuseriii@#1}{\@GLSuseriii@#1}{\@GLSuseriii@#1}{\@GLSuseriii@#1}{\@GLSuseriii@#1}}
```

Read in the final optional argument:

```
4177 \def\@GLSuseriii@#1#2[#3]{%
4178 \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseriii{#2}#3}}%
4179 }
```

`\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`

```
4180 \newrobustcmd*{\glsuseriv}{\@gls@hyp@opt\@glsuseriv}
```

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

```
4181 \newcommand*{\@glsuseriv}[2][]{%
4182 \new@ifnextchar[{\@glsuseriv@{#1}{#2}}{\@glsuseriv@{#1}{#2}[]}}
```

Read in the final optional argument:

```
4183 \def\@glsuseriv@#1#2[#3]{%
4184 \@gls@field@link{#1}{#2}{\glsentryuseriv{#2}#3}%
4185 }
```

`\Glsuseriv` behaves like `\glsuseriv` except that the first letter is converted to uppercase.

`\Glsuseriv`

```
4186 \newrobustcmd*{\Glsuseriv}{\@gls@hyp@opt\@Glsuseriv}
```

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

```
4187 \newcommand*{\@Glsuseriv}[2][]{%
4188 \new@ifnextchar[{\@Glsuseriv@{#1}{#2}}{\@Glsuseriv@{#1}{#2}[]}}
```

Read in the final optional argument:

```
4189 \def\@Glsuseriv@#1#2[#3]{%
4190 \@gls@field@link{#1}{#2}{\Glsentryuseriv{#2}#3}%
4191 }
```

`\GLSuseriv` behaves like `\glsuseriv` except that the link text is converted to uppercase.

`\GLSuseriv`

```
4192 \newrobustcmd*{\GLSuseriv}{\@gls@hyp@opt\@GLSuseriv}
```

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

```
4193 \newcommand*{\@GLSuseriv}[2][]{%
4194 \new@ifnextchar[{\@GLSuseriv@{#1}{#2}}{\@GLSuseriv@{#1}{#2}[]}}
```

Read in the final optional argument:

```
4195 \def\@GLSuseriv@#1#2[#3]{%
4196 \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseriv{#2}#3}}%
4197 }
```

`\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`

```
4198 \newrobustcmd*{\glsuserv}{\@gls@hyp@opt\@glsuserv}
```

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

```
4199 \newcommand*{\@glsuserv}[2] [] {%
4200   \new@ifnextchar [{\@glsuserv@{#1}{#2}}{\@glsuserv@{#1}{#2} []}}
```

Read in the final optional argument:

```
4201 \def\@glsuserv@#1#2[#3] {%
4202   \@gls@field@link{#1}{#2}{\glsentryuserv{#2}#3}%
4203 }
```

`\Glsuserv` behaves like `\glsuserv` except that the first letter is converted to uppercase.

`\Glsuserv`

```
4204 \newrobustcmd*{\Glsuserv}{\@gls@hyp@opt\@Glsuserv}
```

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

```
4205 \newcommand*{\@GLSuserv}[2] [] {%
4206   \new@ifnextchar [{\@GLSuserv@{#1}{#2}}{\@GLSuserv@{#1}{#2} []}}
```

Read in the final optional argument:

```
4207 \def\@GLSuserv@#1#2[#3] {%
4208   \@gls@field@link{#1}{#2}{\Glsentryuserv{#2}#3}%
4209 }
```

`\GLSuserv` behaves like `\glsuserv` except that the link text is converted to uppercase.

`\GLSuserv`

```
4210 \newrobustcmd*{\GLSuserv}{\@gls@hyp@opt\@GLSuserv}
```

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

```
4211 \newcommand*{\@GLSserv}[2] [] {%
4212   \new@ifnextchar [{\@GLSserv@{#1}{#2}}{\@GLSserv@{#1}{#2} []}}
```

Read in the final optional argument:

```
4213 \def\@GLSserv@#1#2[#3] {%
4214   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuserv{#2}#3}}%
4215 }
```

`\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`

```
4216 \newrobustcmd*{\glsuservi}{\@gls@hyp@opt\@glsuservi}
```

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

```
4217 \newcommand*{\@glsuservi}[2] [] {%
4218   \new@ifnextchar [{\@glsuservi@{#1}{#2}}{\@glsuservi@{#1}{#2} []}}
```

Read in the final optional argument:

```
4219 \def\@glsuservi@#1#2[#3] {%
4220   \@gls@field@link{#1}{#2}{\glsentryuservi{#2}#3}%
4221 }
```

`\Glsuservi` behaves like `\glsuservi` except that the first letter is converted to uppercase.

`\Glsuservi`

```
4222 \newrobustcmd*{\Glsuservi}{\@gls@hyp@opt\@Glsuservi}
```

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

```
4223 \newcommand*{\@Glsuservi}[2][\@Glsuservi]
```

```
4224 \new@ifnextchar[{\@Glsuservi@{#1}{#2}}{\@Glsuservi@{#1}{#2}[]}]
```

Read in the final optional argument:

```
4225 \def\@Glsuservi@#1#2[#3]{%
```

```
4226 \@gls@field@link{#1}{#2}{\Glsentryuservi{#2}#3}%
```

```
4227 }
```

`\GLSuservi` behaves like `\glsuservi` except that the link text is converted to uppercase.

`\GLSuservi`

```
4228 \newrobustcmd*{\GLSuservi}{\@gls@hyp@opt\@GLSuservi}
```

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

```
4229 \newcommand*{\@GLSuservi}[2][\@GLSuservi]
```

```
4230 \new@ifnextchar[{\@GLSuservi@{#1}{#2}}{\@GLSuservi@{#1}{#2}[]}]
```

Read in the final optional argument:

```
4231 \def\@GLSuservi@#1#2[#3]{%
```

```
4232 \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuservi{#2}#3}}%
```

```
4233 }
```

Now deal with acronym related keys. First the short form:

`\acrshort`

```
4234 \newrobustcmd*{\acrshort}{\@gls@hyp@opt\@ns@acrshort}
```

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

```
4235 \newcommand*{\@ns@acrshort}[2][\@ns@acrshort]
```

```
4236 \new@ifnextchar[{\@acrshort{#1}{#2}}{\@acrshort{#1}{#2}[]}]
```

```
4237 }
```

Read in the final optional argument:

```
4238 \def\@acrshort#1#2[#3]{%
```

```
4239 \glsdoifexists{#2}%
```

```
4240 {%
```

```
4241 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
```

```
4242 \let\glsifplural\@secondoftwo
```

```
4243 \let\glsapscase\@firstofthree
```

```
4244 \let\glsinsert\@empty
```

```
4245 \def\glscustomtext{%
```

```
4246 \acronymfont{\glsentryshort{#2}}#3%
```

```
4247 }%
```

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

```
4248 \@gls@link[#1]{#2}{\csname gls@\glstyp @entryfmt\endcsname}%
```

```
4249 }
```



```
4250 \glspostlinkhook
4251 }
```

\Acrshort

```
4252 \newrobustcmd*{\Acrshort}{\@gls@hyp@opt\ns@Acrshort}
```

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

```
4253 \newcommand*{\ns@Acrshort}[2] [] {%
4254   \new@ifnextchar[{\@Acrshort{#1}{#2}}{\@Acrshort{#1}{#2} []}]%
4255 }
```

Read in the final optional argument:

```
4256 \def\@Acrshort#1#2[#3] {%
4257   \glsdoifexists{#2}%
4258   {%
4259     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
4260     \def\glslabel{#2}%
4261     \let\glsifplural\@secondoftwo
4262     \let\glscapscase\@secondofthree
4263     \let\glsinsert\@empty
4264     \def\glscustomtext{%
4265       \acronymfont{\Glsentryshort{#2}}#3%
4266     }%

```

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

```
4267   \@gls@link[#1]{#2}{\cename gls@\gls@type @entryfmt\endcename}%
4268   }%
4269   \glspostlinkhook
4270 }
```

\ACRshort

```
4271 \newrobustcmd*{\ACRshort}{\@gls@hyp@opt\ns@ACRshort}
```

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

```
4272 \newcommand*{\ns@ACRshort}[2] [] {%
4273   \new@ifnextchar[{\@ACRshort{#1}{#2}}{\@ACRshort{#1}{#2} []}]%
4274 }
```

Read in the final optional argument:

```
4275 \def\@ACRshort#1#2[#3] {%
4276   \glsdoifexists{#2}%
4277   {%
4278     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

```

```

4279 \def\glslabel{#2}%
4280 \let\glsifplural\@secondoftwo
4281 \let\glsifcaps\@thirdofthree
4282 \let\glsinsert\@empty
4283 \def\glscustomtext{%
4284     \mfirstucMakeUppercase{\acronymfont{\glsentryshort{#2}}#3}%
4285 }%

```

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

```

4286 \@gls@link[#1]{#2}{\csname gls@\glstyp @entryfmt\endcsname}%
4287 }%

4288 \glspostlinkhook
4289 }

```

Short plural:

`\acrshortpl`

```

4290 \newrobustcmd*{\acrshortpl}{\@gls@hyp@opt\ns@acrshortpl}

```

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

```

4291 \newcommand*{\ns@acrshortpl}[2] [] {%
4292     \new@ifnextchar[{\@acrshortpl{#1}{#2}}{\@acrshortpl{#1}{#2} []}%
4293 }

```

Read in the final optional argument:

```

4294 \def\@acrshortpl#1#2[#3]{%
4295     \glsdoifexists{#2}%
4296     {%

4297     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

4298     \def\glslabel{#2}%
4299     \let\glsifplural\@firstoftwo
4300     \let\glsifcaps\@firstofthree
4301     \let\glsinsert\@empty
4302     \def\glscustomtext{%
4303         \acronymfont{\glsentryshortpl{#2}}#3%
4304     }%

```

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

```

4305     \@gls@link[#1]{#2}{\csname gls@\glstyp @entryfmt\endcsname}%
4306 }%

4307 \glspostlinkhook
4308 }

```

`\Acrshortpl`

```

4309 \newrobustcmd*{\Acrshortpl}{\@gls@hyp@opt\ns@Acrshortpl}

```

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

```
4310 \newcommand*{\ns@Acrshortpl}[2] [] {%
4311   \new@ifnextchar[{\@Acrshortpl{#1}{#2}}{\@Acrshortpl{#1}{#2} []}%
4312 }
```

Read in the final optional argument:

```
4313 \def\@Acrshortpl#1#2[#3] {%
4314   \glsdoifexists{#2}%
4315   {%
4316     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
4317     \def\glslabel{#2}%
4318     \let\glsifplural\@firstoftwo
4319     \let\glscapscase\@secondofthree
4320     \let\glsinsert\@empty
4321     \def\glscustomtext{%
4322       \acronymfont{\Glsentryshortpl{#2}}#3%
4323     }%
4324   }%
4325 }
```

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

```
4324   \@gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
4325   }%
4326 \glspostlinkhook
4327 }
```

`\ACRshortpl`

```
4328 \newrobustcmd*{\ACRshortpl}{\@gls@hyp@opt\ns@ACRshortpl}
```

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

```
4329 \newcommand*{\ns@ACRshortpl}[2] [] {%
4330   \new@ifnextchar[{\@ACRshortpl{#1}{#2}}{\@ACRshortpl{#1}{#2} []}%
4331 }
```

Read in the final optional argument:

```
4332 \def\@ACRshortpl#1#2[#3] {%
4333   \glsdoifexists{#2}%
4334   {%
4335     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
4336     \def\glslabel{#2}%
4337     \let\glsifplural\@firstoftwo
4338     \let\glsapspace\@thirdofthree
4339     \let\glsinsert\@empty
4340     \def\glscustomtext{%
4341       \mfirstucMakeUppercase{\acronymfont{\Glsentryshortpl{#2}}#3}%
4342     }%
4343   }%
4344 }
```

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

```
4343 \gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%  
4344 }%  
  
4345 \glspostlinkhook  
4346 }
```

`\acrlong`

```
4347 \newrobustcmd*{\acrlong}{\@gls@hyp@opt\ns@acrlong}
```

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

```
4348 \newcommand*{\ns@acrlong}[2][{}]{%  
4349 \new@ifnextchar[{\@acrlong{#1}{#2}}{\@acrlong{#1}{#2}[]}%  
4350 }
```

Read in the final optional argument:

```
4351 \def\@acrlong#1#2[#3]{%  
4352 \glsdoifexists{#2}%  
4353 {%  
  
4354 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper  
  
4355 \def\glslabel{#2}%  
4356 \let\glsifplural\@secondoftwo  
4357 \let\glscapscase\@firstofthree  
4358 \let\glsinsert\@empty
```

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

```
4359 \def\glscustomtext{%  
4360 \glsentrylong{#2}#3%  
4361 }%
```

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

```
4362 \gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%  
4363 }%  
  
4364 \glspostlinkhook  
4365 }
```

`\Acrlong`

```
4366 \newrobustcmd*{\Acrlong}{\@gls@hyp@opt\ns@Acrlong}
```

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

```
4367 \newcommand*{\ns@Acrlong}[2][{}]{%  
4368 \new@ifnextchar[{\@Acrlong{#1}{#2}}{\@Acrlong{#1}{#2}[]}%  
4369 }
```

Read in the final optional argument:

```
4370 \def\@Acrlong#1#2[#3]{%  
4371 \glsdoifexists{#2}%  
4372 {%
```

```

4373 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4374 \def\glslabel{#2}%
4375 \let\gl@sifplural\@secondoftwo
4376 \let\glscapscase\@secondofthree
4377 \let\glinsert\@empty

```

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

```

4378 \def\glscustomtext{%
4379 \glentrylong{#2}#3%
4380 }%

```

Call \@gl@link. Note that \@gl@link sets \glstype.

```

4381 \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4382 }%
4383 \glspostlinkhook
4384 }

```

\ACRlong

```

4385 \newrobustcmd*{\ACRlong}{\@gl@hyp@opt\ns@ACRlong}

```

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

```

4386 \newcommand*{\ns@ACRlong}[2][{}]{%
4387 \new@ifnextchar[{\@ACRlong{#1}{#2}}{\@ACRlong{#1}{#2}[]}%
4388 }

```

Read in the final optional argument:

```

4389 \def\@ACRlong#1#2[#3]{%
4390 \gl@dofexists{#2}%
4391 {%

```

```

4392 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4393 \def\glslabel{#2}%
4394 \let\gl@sifplural\@secondoftwo
4395 \let\glscapscase\@thirdofthree
4396 \let\glinsert\@empty

```

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

```

4397 \def\glscustomtext{%
4398 \mfirstucMakeUppercase{\glentrylong{#2}#3}%
4399 }%

```

Call \@gl@link. Note that \@gl@link sets \glstype.

```

4400 \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4401 }%
4402 \glspostlinkhook
4403 }

```

Short plural:

`\acrlongpl`

```
4404 \newrobustcmd*{\acrlongpl}{\@gls@hyp@opt\ns@acrlongpl}
```

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

```
4405 \newcommand*{\ns@acrlongpl}[2][\%]
```

```
4406 \new@ifnextchar[{\@acrlongpl{#1}{#2}}{\@acrlongpl{#1}{#2}[]}%
```

```
4407 }
```

Read in the final optional argument:

```
4408 \def\@acrlongpl#1#2[#3]{%
```

```
4409 \glsdoifexists{#2}%
```

```
4410 {%
```

```
4411 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
```

```
4412 \def\glslabel{#2}%
```

```
4413 \let\glsifplural\@firstoftwo
```

```
4414 \let\glsapscase\@firstofthree
```

```
4415 \let\glsinsert\@empty
```

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

```
4416 \def\glscustomtext{%
```

```
4417 \glsentrylongpl{#2}#3%
```

```
4418 }%
```

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

```
4419 \@gls@link[#1]{#2}{\csname gls@\glsstyle @entryfmt\endcsname}%
```

```
4420 }%
```

```
4421 \glspostlinkhook
```

```
4422 }
```

`\Acrlongpl`

```
4423 \newrobustcmd*{\Acrlongpl}{\@gls@hyp@opt\ns@Acrlongpl}
```

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

```
4424 \newcommand*{\ns@Acrlongpl}[2][\%]
```

```
4425 \new@ifnextchar[{\@Acrlongpl{#1}{#2}}{\@Acrlongpl{#1}{#2}[]}%
```

```
4426 }
```

Read in the final optional argument:

```
4427 \def\@Acrlongpl#1#2[#3]{%
```

```
4428 \glsdoifexists{#2}%
```

```
4429 {%
```

```
4430 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
```

```

4431 \def\glslabel{#2}%
4432 \let\glsifplural\@firstoftwo
4433 \let\gls caps case\@secondofthree
4434 \let\glsinsert\@empty

```

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

```

4435 \def\gls customtext{%
4436 \Glsentrylongpl{#2}#3%
4437 }%

```

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

```

4438 \@gls@link[#1]{#2}{\csname gls@\gls type @entryfmt\endcsname}%
4439 }%

```

```

4440 \gls post link hook
4441 }

```

\ACRlongpl

```

4442 \newrobustcmd*{\ACRlongpl}{\@gls@hyp@opt\@ns@ACRlongpl}

```

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

```

4443 \newcommand*{\@ns@ACRlongpl}[2][ ]{%
4444 \new@ifnextchar[{\@ACRlongpl{#1}{#2}}{\@ACRlongpl{#1}{#2}[]}%
4445 }

```

Read in the final optional argument:

```

4446 \def\@ACRlongpl#1#2[#3]{%
4447 \glsdoifexists{#2}%
4448 {%

```

```

4449 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

```

```

4450 \def\glslabel{#2}%
4451 \let\glsifplural\@firstoftwo
4452 \let\gls caps case\@thirdofthree
4453 \let\glsinsert\@empty

```

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

```

4454 \def\gls customtext{%
4455 \mfirstucMakeUppercase{\glsentrylongpl{#2}#3}%
4456 }%

```

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

```

4457 \@gls@link[#1]{#2}{\csname gls@\gls type @entryfmt\endcsname}%
4458 }%

```

```

4459 \gls post link hook
4460 }

```

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{\langle label \rangle}{\langle field \rangle}
```

```
4461 \newcommand*{\@gls@entry@field}[2]{%
4462   \csname glo@glsdetoklabel{#1}@#2\endcsname
4463 }
```

`glsletentryfield`

```
\glsletentryfield{\langle cs \rangle}{\langle label \rangle}{\langle field \rangle}
```

```
4464 \newcommand*{\glsletentryfield}[3]{%
4465   \letcs{#1}{glo@glsdetoklabel{#2}@#3}%
4466 }
```

`Gls@entry@field` Generic first letter uppercase version.

```
\@Gls@entry@field{\langle label \rangle}{\langle field \rangle}
```

```
4467 \newcommand*{\@Gls@entry@field}[2]{%
4468   \glsdoifexistsordo{#1}%
4469   {%
4470     \letcs{\@glo@text}{glo@glsdetoklabel{#1}@#2}%
4471     \ifdef{\@glo@text
4472       {%
4473         \xmakefirstuc{\@glo@text}%
4474       }%
4475     }%
4476     ??\PackageError{glossaries}{The field ‘#2’ doesn’t exist for glossary
4477     entry ‘\glsdetoklabel{#1}’}{Check you have correctly spelt the entry
4478     label and the field name}%
4479   }%
4480 }%
4481 {%
4482   ???%
4483 }%
4484 }
```

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`

```
4485 \newcommand*{\glsentryname}[1]{\@gls@entry@field{#1}{name}}
```

`\Glsentryname`

```
4486 \newrobustcmd*{\Glsentryname}[1]{%
4487   \@Gls@entryname{#1}%
4488 }
```

`\@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:

```
4489 \newcommand*{\@Gls@entryname}[1]{%
4490   \@Gls@entry@field{#1}{name}%
4491 }
```

`\ls@acentryname` Now the behaviour when `\setacronymstyle` is used:

```
4492 \newcommand*{\@Gls@acentryname}[1]{%
4493   \ifglshaslong{#1}%
4494   {%
4495     \letcs\@glo@text{glo@\glsdetoklabel{#1}@name}%
4496     \expandafter\@gls@getbody\@glo@text{}\@nil
4497     \expandafter\ifx\@gls@body\glsentrylong\relax
4498       \expandafter\Glsentrylong\@gls@rest
4499     \else
4500       \expandafter\ifx\@gls@body\glsentryshort\relax
4501         \expandafter\Glsentryshort\@gls@rest
4502       \else
4503         \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.

```
4504     {%
4505       \let\glsentryshort\Glsentryshort
4506       \@glo@text
4507     }%
4508   \else
4509     \xmakefirstuc{\@glo@text}%
4510   \fi
4511 \fi
4512 \fi
4513 }%
4514 }
```

Not an acronym

```
4515   \@Gls@entry@field{#1}{name}%
4516 }%
4517 }
```

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`

```
4518 \newcommand*\glsentrydesc}[1]{\@gls@entry@field{#1}{desc}}
```

`\Glsentrydesc`

```
4519 \newrobustcmd*\Glsentrydesc}[1]{%
4520 \@Gls@entry@field{#1}{desc}%
4521 }
```

Plural form:

`entrydescplural`

```
4522 \newcommand*\glsentrydescplural}[1]{%
4523 \@gls@entry@field{#1}{descplural}%
4524 }
```

`entrydescplural`

```
4525 \newrobustcmd*\Glsentrydescplural}[1]{%
4526 \@Gls@entry@field{#1}{descplural}%
4527 }
```

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`

```
4528 \newcommand*\glsentrytext}[1]{\@gls@entry@field{#1}{text}}
```

`\Glsentrytext`

```
4529 \newrobustcmd*\Glsentrytext}[1]{%
4530 \@Gls@entry@field{#1}{text}%
4531 }
```

Get the plural form:

`\glsentryplural`

```
4532 \newcommand*\glsentryplural}[1]{%
4533 \@gls@entry@field{#1}{plural}%
4534 }
```

`\Glsentryplural`

```
4535 \newrobustcmd*\Glsentryplural}[1]{%
4536 \@Gls@entry@field{#1}{plural}%
4537 }
```

Get the symbol associated with this entry. The argument is the label associated with the entry.

`\glsentrysymbol`

```
4538 \newcommand*{\glsentrysymbol}[1]{%
4539   \@gls@entry@field{#1}{symbol}%
4540 }
```

`\Glsentrysymbol`

```
4541 \newrobustcmd*{\Glsentrysymbol}[1]{%
4542   \@Gls@entry@field{#1}{symbol}%
4543 }
```

Plural form:

`trysymbolplural`

```
4544 \newcommand*{\glsentrysymbolplural}[1]{%
4545   \@gls@entry@field{#1}{symbolplural}%
4546 }
```

`trysymbolplural`

```
4547 \newrobustcmd*{\Glsentrysymbolplural}[1]{%
4548   \@Gls@entry@field{#1}{symbolplural}%
4549 }
```

Get the entry text to be used when the entry is first used in the document (as specified by the first key when the entry was defined).

`\glsentryfirst`

```
4550 \newcommand*{\glsentryfirst}[1]{%
4551   \@gls@entry@field{#1}{first}%
4552 }
```

`\Glsentryfirst`

```
4553 \newrobustcmd*{\Glsentryfirst}[1]{%
4554   \@Gls@entry@field{#1}{first}%
4555 }
```

Get the plural form (as specified by the firstplural key when the entry was defined).

`ntryfirstplural`

```
4556 \newcommand*{\glsentryfirstplural}[1]{%
4557   \@gls@entry@field{#1}{firstpl}%
4558 }
```

`ntryfirstplural`

```
4559 \newrobustcmd*{\Glsentryfirstplural}[1]{%
4560   \@Gls@entry@field{#1}{firstpl}%
4561 }
```

sentrytitlecase

```
4562 \newrobustcmd*{\@glsentrytitlecase}[2]{%
4563   \glsfieldfetch{#1}{#2}{\@gls@value}%
4564   \xcapitalisewords{\@gls@value}%
4565 }
4566 \ifdef\texorpdfstring
4567 {
4568   \newcommand*{\glsentrytitlecase}[2]{%
4569     \texorpdfstring
4570     {\@glsentrytitlecase{#1}{#2}}%
4571     {\@gls@entry@field{#1}{#2}}%
4572   }
4573 }
4574 {
4575   \newcommand*{\glsentrytitlecase}[2]{\@glsentrytitlecase{#1}{#2}}
4576 }
```

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

\glsentrytype

```
4577 \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

```
4578 \newcommand*{\glsentrysort}[1]{%
4579   \@gls@entry@field{#1}{sort}%
4580 }
```

\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.

```
4581 \newcommand*{\glsentryuseri}[1]{%
4582   \@gls@entry@field{#1}{useri}%
4583 }
```

\Glsentryuseri

```
4584 \newrobustcmd*{\Glsentryuseri}[1]{%
4585   \@Gls@entry@field{#1}{useri}%
4586 }
```

\glsentryuserii Get the second user key (as specified by the user2 when the entry was defined). The argument is the label associated with the entry.

```
4587 \newcommand*{\glsentryuserii}[1]{%
4588   \@gls@entry@field{#1}{userii}%
4589 }
```

`\Glsentryuserii`

```
4590 \newrobustcmd*{\Glsentryuserii}[1]{%
4591   \@Gls@entry@field{#1}{userii}%
4592 }
```

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

```
4593 \newcommand*{\glsentryuseriii}[1]{%
4594   \@Gls@entry@field{#1}{useriii}%
4595 }
```

`Glsentryuseriii`

```
4596 \newrobustcmd*{\Glsentryuseriii}[1]{%
4597   \@Gls@entry@field{#1}{useriii}%
4598 }
```

`\glsentryuseriv` Get the fourth user key (as specified by the `user4` when the entry was defined). The argument is the label associated with the entry.

```
4599 \newcommand*{\glsentryuseriv}[1]{%
4600   \@Gls@entry@field{#1}{useriv}%
4601 }
```

`\Glsentryuseriv`

```
4602 \newrobustcmd*{\Glsentryuseriv}[1]{%
4603   \@Gls@entry@field{#1}{useriv}%
4604 }
```

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

```
4605 \newcommand*{\glsentryuserv}[1]{%
4606   \@Gls@entry@field{#1}{userv}%
4607 }
```

`\Glsentryuserv`

```
4608 \newrobustcmd*{\Glsentryuserv}[1]{%
4609   \@Gls@entry@field{#1}{userv}%
4610 }
```

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

```
4611 \newcommand*{\glsentryuservi}[1]{%
4612   \@Gls@entry@field{#1}{uservi}%
4613 }
```

`\Glsentryuservi`

```
4614 \newrobustcmd*{\Glsentryuservi}[1]{%
4615   \@Gls@entry@field{#1}{uservi}%
4616 }
```

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

```
4617 \newcommand*{\glsentryshort}[1]{\@gls@entry@field{#1}{short}}
```

`\Glsentryshort`

```
4618 \newrobustcmd*{\Glsentryshort}[1]{%
4619   \@Gls@entry@field{#1}{short}%
4620 }
```

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

```
4621 \newcommand*{\glsentryshortpl}[1]{\@gls@entry@field{#1}{shortpl}}
```

`\Glsentryshortpl`

```
4622 \newrobustcmd*{\Glsentryshortpl}[1]{%
4623   \@Gls@entry@field{#1}{shortpl}%
4624 }
```

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

```
4625 \newcommand*{\glsentrylong}[1]{\@gls@entry@field{#1}{long}}
```

`\Glsentrylong`

```
4626 \newrobustcmd*{\Glsentrylong}[1]{%
4627   \@Gls@entry@field{#1}{long}%
4628 }
```

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

```
4629 \newcommand*{\glsentrylongpl}[1]{\@gls@entry@field{#1}{longpl}}
```

`\Glsentrylongpl`

```
4630 \newrobustcmd*{\Glsentrylongpl}[1]{%
4631   \@Gls@entry@field{#1}{longpl}%
4632 }
```

Short cut macros to access full form:

`\glsentryfull`

```
4633 \newcommand*{\glsentryfull}[1]{%
4634   \acrfullformat{\glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%
4635 }
```

`\Glsentryfull`

```
4636 \newrobustcmd*{\Glsentryfull}[1]{%
4637   \acrfullformat{\Glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%
4638 }
```

`\glsentryfullpl`

```
4639 \newcommand*\glsentryfullpl}[1]{%
4640   \acrfullformat{\glsentrylongpl{#1}}{\acronymfont{\glsentryshortpl{#1}}}%
4641 }
```

`\Glsentryfullpl`

```
4642 \newrobustcmd*\Glsentryfullpl}[1]{%
4643   \acrfullformat{\Glsentrylongpl{#1}}{\acronymfont{\glsentryshortpl{#1}}}%
4644 }
```

`entrynumberlist` Displays the number list as is.

```
4645 \newcommand*\glsentrynumberlist}[1]{%
4646   \glsdoifexists{#1}%
4647   {%
4648     \@gls@entry@field{#1}{numberlist}%
4649   }%
4650 }
```

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

```
4651 \@ifpackageloaded{hyperref} {%
4652   \newcommand*\glsdisplaynumberlist}[1]{%
4653     \GlossariesWarning
4654     {%
4655       \string\glsdisplaynumberlist\space
4656       doesn't work with hyperref.^^JUsing
4657       \string\glsentrynumberlist\space instead%
4658     }%
4659     \glsentrynumberlist{#1}%
4660   }%
4661 }%
4662 {%
4663   \newcommand*\glsdisplaynumberlist}[1]{%
4664     \glsdoifexists{#1}%
4665     {%
4666       \bgroup
4667
4668       \edef\@glo@label{\glsdetoklabel{#1}}%
4669       \let\@org@glsnumberformat\glsnumberformat
4670       \def\glsnumberformat##1{##1}%
4671       \protected@edef\the@numberlist{%
4672         \csname glo@\@glo@label @numberlist\endcsname}%
4673       \def\@gls@numlist@sep{%
4674         \def\@gls@numlist@nextsep{%
4675           \def\@gls@thislist{%
4676             \def\@gls@donext@def{%
4677               \renewcommand\do[1]{%
4678                 \protected@edef\@gls@thislist{%
4679                   \@gls@thislist
```

```

4680         \noexpand\@gls@numlist@sep
4681         ##1%
4682     }%
4683     \let\@gls@numlist@sep\@gls@numlist@nextsep
4684     \def\@gls@numlist@nextsep{\glsnumlistsep}%
4685     \@gls@donext@def
4686     \def\@gls@donext@def{%
4687         \def\@gls@numlist@lastsep{\glsnumlistlastsep}%
4688     }%
4689 }%
4690 \expandafter \glsnumlistparser \expandafter{\the@numberlist}%
4691 \let\@gls@numlist@sep\@gls@numlist@lastsep
4692 \@gls@thislist
4693 \egroup
4694 }%
4695 }
4696 }

```

`\glsnumlistsep`

```
4697 \newcommand*{\glsnumlistsep}{, }
```

`\glsnumlistlastsep`

```
4698 \newcommand*{\glsnumlistlastsep}{ \& }
```

`\gls hyperlink`

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

```

4699 \newcommand*{\gls hyperlink}[2][\glsentrytext{\@glo@label}]{%
4700 \def\@glo@label{#2}%
4701 \@gls link{\glo 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`:

```

4702 \define@key{glossadd}{counter}{\def\@gls@counter{#1}}
4703 \define@key{glossadd}{format}{\def\@glsnumberformat{#1}}

```

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

```
4704 \define@key{glossadd}{types}{\def\@glo@type{#1}}
```

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

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

`\glsadd`

```
4705 \newrobustcmd*{\glsadd}[2] [] {%
```

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

```
4706 \@gls@adjustmode
```

```
4707 \glsdoifexists{#2}%
```

```
4708 {%
```

```
4709 \def\@glsnumberformat{glsnumberformat}%
```

```
4710 \edef\@gls@counter{\csname glo@glstoklabel{#2}@counter\endcsname}%
```

```
4711 \setkeys{glossadd}{#1}%
```

Store the entry's counter in `\theglsentrycounter`

```
4712 \@gls@saveentrycounter
```

Define sort key if necessary:

```
4713 \@gls@setsort{#2}%
```

This should use `\@do@wrglossary` rather than `\do@wrglossary` since the whole point of `\glsadd` is to add a line to the glossary.

```
4714 \@do@wrglossary{#2}%
```

```
4715 }%
```

```
4716 }
```

`@gls@adjustmode`

```
4717 \newcommand*{\@gls@adjustmode}{}
```

```
4718 \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`

```
4719 \newrobustcmd*{\glsaddall}[1] [] {%
```

```
4720 \edef\@glo@type{\@glo@types}%
```

```
4721 \setkeys{glossadd}{#1}%
```

```
4722 \forallglsentries[\@glo@type]{\@glo@entry}{%
```

```
4723 \glsadd[#1]{\@glo@entry}%
```

```
4724 }%
```

```
4725 }
```

`\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.

```
4726 \newrobustcmd*{\glsaddallunused}[1] [\@glo@types] {%
```

```

4727 \forallglsentries[#1]{\@glo@entry}%
4728 {%
4729     \ifglsused{\@glo@entry}{\glsadd[format=glsignore]{\@glo@entry}}%
4730 }%
4731 }

```

`\glsignore`

```
4732 \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 `\glsymbolsgroupname` replaces `glssymbols` and `\glsnumbersgroupname` replaces `glsnumbers`) using the command `\glsgetgrouptitle` which is defined in `.` This is done to prevent any problem characters in `\glsymbolsgroupname` and `\glsnumbersgroupname` from breaking hyperlinks.

`\glsopenbrace` Define `\glsopenbrace` to make it easier to write an opening brace to a file.

```
4733 \edef\glsopenbrace{\expandafter\@gobble\string\{}
```

`\glsclosebrace` Define `\glsclosebrace` to make it easier to write an opening brace to a file.

```
4734 \edef\glsclosebrace{\expandafter\@gobble\string\}}
```

`\glsbackslash` Define `\glsbackslash` to make it easier to write a backslash to a file.

```
4735 \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.

```
4736 \edef\glsquote#1{\string"#1\string"}
```

`\glspercentchar` Define `\glspercentchar` to make it easier to write a percent character to a file.

```
4737 \edef\glspercentchar{\expandafter\@gobble\string\%}
```

`\glstildechar` Define `\glstildechar` to make it easier to write a tilde character to a file.

```
4738 \edef\glstildechar{\string~}
```

`\@glsfirstletter` Define the first letter to come after the digits 0,...,9. Only required for xindy.

```
4739 \ifglxsindy
4740 \newcommand*{\@glsfirstletter}{A}
4741 \fi
```

`\@glsfirstletterAfterDigits` Sets the first letter to come after the digits 0,...,9. The starred version sanitizes.

```
4742 \newcommand*{\@GlsSetXdyFirstLetterAfterDigits}{%
4743 \ifstar\s@GlsSetXdyFirstLetterAfterDigits\@GlsSetXdyFirstLetterAfterDigits}
4744 \ifglxsindy
4745 \newcommand*{\@GlsSetXdyFirstLetterAfterDigits}[1]{%
4746 \renewcommand*{\@glsfirstletter}{#1}}
4747 \newcommand*{\s@GlsSetXdyFirstLetterAfterDigits}[1]{%
4748 \renewcommand*{\@glsfirstletter}{#1}%
4749 \@onelevel@sanitize\@glsfirstletter
4750 }
4751 \else
4752 \newcommand*{\@GlsSetXdyFirstLetterAfterDigits}[1]{%
4753 \glsnoindywarning\GlsSetXdyFirstLetterAfterDigits}
4754 \newcommand*{\s@GlsSetXdyFirstLetterAfterDigits}{%
4755 \@GlsSetXdyFirstLetterAfterDigits
4756 }
4757 \fi
```

`\@numbergrouporder` Specifies the order of the number group.

```
4758 \ifglxsindy
4759 \newcommand*{\@xdynumbergrouporder}{:before \string"\@glsfirstletter\string"}
4760 \fi
```

`\@numberGroupOrder` Sets the relative location of the number group. The starred version sanitizes.

```
4761 \newcommand*{\@GlsSetXdyNumberGroupOrder}[1]{%
4762 \ifstar\s@GlsSetXdyNumberGroupOrder\@GlsSetXdyNumberGroupOrder
4763 }
4764 \ifglxsindy
4765 \newcommand*{\@GlsSetXdyNumberGroupOrder}[1]{%
4766 \renewcommand*{\@xdynumbergrouporder}{#1}%
4767 }
4768 \newcommand*{\s@GlsSetXdyNumberGroupOrder}[1]{%
4769 \renewcommand*{\@xdynumbergrouporder}{#1}%
4770 \@onelevel@sanitize\@xdynumbergrouporder
4771 }
4772 \else
4773 \newcommand*{\@GlsSetXdyNumberGroupOrder}[1]{%
4774 \glsnoindywarning\GlsSetXdyNumberGroupOrder}
4775 \newcommand*{\s@GlsSetXdyNumberGroupOrder}{%
4776 \@GlsSetXdyNumberGroupOrder}
4777 \fi
```

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

```
4778 \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.

```
4779 \ifglsxindy
4780 \newcommand*\GlsSetXdyMinRangeLength[1]{%
4781 \renewcommand*\@glsminrange{#1}}
4782 \else
4783 \newcommand*\GlsSetXdyMinRangeLength[1]{%
4784 \glsnoxindywarning\GlsSetXdyMinRangeLength}
4785 \fi
```

`\writeist`

```
4786 \ifglsxindy
Code to use if xindy is required.
4787 \def\writeist{%
Define write register if not already defined
4788 \ifundef{\glswrite}{\newwrite\glswrite}{}%
Update attributes list
4789 \@gls@addpredefinedattributes
Open the file.
4790 \openout\glswrite=\istfilename
Write header comment at the start of the file
4791 \write\glswrite{;; xindy style file created by the glossaries
4792 package}%
4793 \write\glswrite{;; for document '\jobname' on
4794 \the\year-\the\month-\the\day}%
Specify the required styles
4795 \write\glswrite{^^J; required styles^^J}
4796 \@for\@xdystyle:=\@xdyrequiredstyles\do{%
4797 \ifx\@xdystyle\@empty
4798 \else
4799 \protected@write\glswrite}{(require
4800 \string"\@xdystyle.xdy\string")}%
4801 \fi
4802 }%
List the allowed attributes (possible values used by the format key)
4803 \write\glswrite{^^J%
4804 ; list of allowed attributes (number formats)^^J}%
4805 \write\glswrite{(define-attributes ((\@xdyattributes)))}%
Define any additional alphabets
4806 \write\glswrite{^^J; user defined alphabets^^J}%
4807 \write\glswrite{\@xdyuseralphabets}%
Define location classes.
4808 \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.

```
4809 \@for\@gls@classI:=\@gls@xdy@locationlist\do{%
```

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

```
4810 \protected@write\glswrite{}{(define-location-class
4811 \string"\@gls@classI\string"^^J\space\space\space
4812 (
4813 :sep "{}"
4814 \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
4815 :sep "}"
4816 )
4817 ^^J\space\space\space
4818 :min-range-length \@glsminrange^^J%
4819 )
4820 }%
```

Nested iteration over all classes:

```
4821 {%
4822 \@for\@gls@classII:=\@gls@xdy@locationlist\do{%
4823 \protected@write\glswrite{}{(define-location-class
4824 \string"\@gls@classII-\@gls@classI\string"
4825 ^^J\space\space\space
4826 (
4827 :sep "{"
4828 \csname @gls@xdy@Lclass@\@gls@classII\endcsname\space
4829 :sep "{}"
4830 \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
4831 :sep "}"
4832 )
4833 ^^J\space\space\space
4834 :min-range-length \@glsminrange^^J%
4835 )
4836 }%
4837 }%
4838 }%
4839 }%
```

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

```
4840 \write\glswrite{^^J; user defined location classes}%
4841 \write\glswrite{\@xdyuserlocationdefs}%
```

Cross-reference class. (The unverified option is used as the cross-references are supplied using the list of labels along with the optional argument for $\backslash\text{glsseeformat}$ which xindy won't recognise.)

```
4842 \write\glswrite{^^J; define cross-reference class^^J}%
4843 \write\glswrite{(define-crossref-class \string"see\string"
4844 :unverified )}%
```

Define how cross-references should be displayed. This adds an empty set of braces after the cross-referencing information allowing for the final argument of $\backslash\text{glsseeformat}$ which

gets ignored. (When using `makeindex` this final argument contains the location information which is not required.)

```
4845 \write\glswrite{(markup-crossref-list
4846   :class \string"see\string"^^J\space\space\space
4847   :open \string"\string\glssseeformat\string"
4848   :close \string"{}\string")}%
```

Provide hook to write extra material here (used by `glossaries-extra` to define a `seealso` class).

```
4849 \@xdycrossrefhook
```

List the order to sort the classes.

```
4850 \write\glswrite{^^J; define the order of the location classes}%
4851 \write\glswrite{(define-location-class-order
4852   (\@xdylocationclassorder))}%
```

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

```
4853 \write\glswrite{^^J; define the glossary markup^^J}%

4854 \write\glswrite{(markup-index^^J\space\space\space
4855   :open \string"\string
4856   \glossarysection[\string\glossarytoctitle]{\string
4857   \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`)

```
4858 \@for\@this@ctr:=\@xdycounters\do{%
4859   {%
4860     \@for\@this@attr:=\@xdyattributelist\do{%
4861       \protected\write\glswrite{}{\string\providecommand*%
4862         \expandafter\string
4863         \csname glsX\@this@ctr X\@this@attr\endcsname[2]%
4864         {%
4865           \string\setentrycounter
4866           [\expandafter\@gobble\string\#1]{\@this@ctr}%
4867           \expandafter\string
4868           \csname\@this@attr\endcsname
4869           {\expandafter\@gobble\string\#2}%
4870         }%
4871       }%
4872     }%
4873   }%
4874 }%
```

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

```
4875 \write\glswrite{%
4876   \string\begin
4877   {theglossary}\string\glossaryheader\glstildechar n\string" ^^J\space
4878   \space\space:close \string"\glspersentchar\glstildechar n\string
4879   \end{theglossary}\string\glossarypostamble
4880   \glstildechar n\string" ^^J\space\space\space
4881   :tree}}%
```

Specify what to put between letter groups

```
4882 \write\glswrite{(markup-letter-group-list
4883 :sep \string\string\glsgroupskip\glstildechar n\string)}}%
```

Specify what to put between entries

```
4884 \write\glswrite{(markup-indexentry
4885 :open \string\string\relax \string\glresetentrylist
4886 \glstildechar n\string)}}%
```

Specify how to format entries

```
4887 \write\glswrite{(markup-locclass-list :open
4888 \string\glsoopenbrace\string\glossaryentrynumbers
4889 \glsoopenbrace\string\relax\space \string^^J\space\space\space
4890 :sep \string", \string"
4891 :close \string\glsclosebrace\glsclosebrace\string)}}%
```

Specify how to separate location numbers

```
4892 \write\glswrite{(markup-locref-list
4893 :sep \string\string\delimN\space\string)}}%
```

Specify how to indicate location ranges

```
4894 \write\glswrite{(markup-range
4895 :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.

```
4896 \@onelevel@sanitize\gls@suffixF
4897 \@onelevel@sanitize\gls@suffixFF
4898 \ifx\gls@suffixF\@empty
4899 \else
4900 \write\glswrite{(markup-range
4901 :close "\gls@suffixF" :length 1 :ignore-end)}}%
4902 \fi
4903 \ifx\gls@suffixFF\@empty
4904 \else
4905 \write\glswrite{(markup-range
4906 :close "\gls@suffixFF" :length 2 :ignore-end)}}%
4907 \fi
```

Specify how to format locations.

```
4908 \write\glswrite{^^J; define format to use for locations^^J}%
4909 \write\glswrite{\@xdylocref}%
```

Specify how to separate letter groups.

```
4910 \write\glswrite{^^J; define letter group list format^^J}%
4911 \write\glswrite{(markup-letter-group-list
4912 :sep \string\string\glsgroupskip\glstildechar n\string)}}%
```

Define letter group headings.

```
4913 \write\glswrite{^^J; letter group headings^^J}%
4914 \write\glswrite{(markup-letter-group
```

```

4915      :open-head \string"\string\glsgroupheading
4916      \glsopenbrace\string"^^J\space\space\space
4917      :close-head \string"\glsclosebrace\string")}%

```

Define additional letter groups.

```

4918      \write\glswrite{^^J; additional letter groups^^J}%
4919      \write\glswrite{\@xdylettergroups}%

```

Define additional sort rules

```

4920      \write\glswrite{^^J; additional sort rules^^J}
4921      \write\glswrite{\@xdysortrules}%

```

Hook for any additional information:

```

4922      \@gls@writeisthook

```

Close the style file

```

4923      \closeout\glswrite

```

Suppress any further calls.

```

4924      \let\writeist\relax
4925      }
4926 \else

```

Code to use if makeindex is required.

```

4927 \edef\@gls@actualchar{\string?}
4928 \edef\@gls@encapchar{\string|}
4929 \edef\@gls@levelchar{\string!}
4930 \edef\@gls@quotechar{\string"}%
4931 \let\GlsSetQuote\gls@nosetquote
4932 \def\writeist{\relax
4933 \ifundef{\glswrite}{\newwrite\glswrite}{}\relax
4934 \openout\glswrite=\istfilename
4935 \write\glswrite{\glspercentchar\space makeindex style file
4936 created by the glossaries package}
4937 \write\glswrite{\glspercentchar\space for document
4938 ' \jobname' on \the\year-\the\month-\the\day}
4939 \write\glswrite{actual '@gls@actualchar'}
4940 \write\glswrite{encap '@gls@encapchar'}
4941 \write\glswrite{level '@gls@levelchar'}
4942 \write\glswrite{quote '@gls@quotechar'}
4943 \write\glswrite{keyword \string"\string\glossaryentry\string"}
4944 \write\glswrite{preamble \string"\string\glossarysection[\string
4945 \glossarytoctitle]{\string\glossarytitle}\string
4946 \glossarypreamble\string\n\string\begin{theglossary}\string
4947 \glossaryheader\string\n\string"}
4948 \write\glswrite{postamble \string"\string%\string\n\string
4949 \end{theglossary}\string\glossarypostamble\string\n
4950 \string"}
4951 \write\glswrite{group_skip \string"\string\glsgroupskip\string\n
4952 \string"}
4953 \write\glswrite{item_0 \string"\string%\string\n\string"}
4954 \write\glswrite{item_1 \string"\string%\string\n\string"}

```



```

4955 \write\glswrite{item_2 \string"\string%\string\n\string"}
4956 \write\glswrite{item_01 \string"\string%\string\n\string"}
4957 \write\glswrite{item_x1
4958 \string"\string\relax \string\glsresetentrylist\string\n
4959 \string"}
4960 \write\glswrite{item_12 \string"\string%\string\n\string"}
4961 \write\glswrite{item_x2
4962 \string"\string\relax \string\glsresetentrylist\string\n
4963 \string"}

4964 \write\glswrite{delim_0 \string"\string\{\string
4965 \glossaryentrynumbers\string\{\string\relax \string"}
4966 \write\glswrite{delim_1 \string"\string\{\string
4967 \glossaryentrynumbers\string\{\string\relax \string"}
4968 \write\glswrite{delim_2 \string"\string\{\string
4969 \glossaryentrynumbers\string\{\string\relax \string"}
4970 \write\glswrite{delim_t \string"\string\}\string}\string"}
4971 \write\glswrite{delim_n \string"\string\delimN \string"}
4972 \write\glswrite{delim_r \string"\string\delimR \string"}
4973 \write\glswrite{headings_flag 1}
4974 \write\glswrite{heading_prefix
4975 \string"\string\glsgroupheading\string\{\string"}
4976 \write\glswrite{heading_suffix
4977 \string"\string}\string\relax
4978 \string\glsresetentrylist \string"}
4979 \write\glswrite{symhead_positive \string"glssymbols\string"}
4980 \write\glswrite{numhead_positive \string"glslnumbers\string"}
4981 \write\glswrite{page_compositor \string"glscpositor\string"}
4982 \@gls@escbsdq\gls@suffixF
4983 \@gls@escbsdq\gls@suffixFF
4984 \ifx\gls@suffixF\@empty
4985 \else
4986 \write\glswrite{suffix_2p \string"\gls@suffixF\string"}
4987 \fi
4988 \ifx\gls@suffixFF\@empty
4989 \else
4990 \write\glswrite{suffix_3p \string"\gls@suffixFF\string"}
4991 \fi

```

Hook for any additional information:

```
4992 \@gls@writeisthook
```

Close the file and disable \writeist.

```

4993 \closeout\glswrite
4994 \let\writeist\relax
4995 }
4996 \fi

```

SetWriteIstHook Allow user to append information to the style file.

```

4997 \newcommand*\GlsSetWriteIstHook}[1]{\renewcommand*\@gls@writeisthook}{#1}}
4998 \@onlypremakeg\GlsSetWriteIstHook

```

ls@writeisthook

```
4999 \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.

```
5000 \ifglxsindy
```

```
5001 \newcommand*{\GlsSetQuote}[1]{\glsnomakeindexwarning\GlsSetQuote}
```

```
5002 \newcommand*{\gls@nosetquote}[1]{\glsnomakeindexwarning\GlsSetQuote}
```

```
5003 \else
```

```
5004 \newcommand*{\GlsSetQuote}[1]{\edef\@gls@quotechar{\string#1}%
```

If German is in use, set the extra makeindex option so makeglossaries can pick it up.

```
5005 \ifpackageloaded{tracklang}%
```

```
5006 {%
```

```
5007 \IfTrackedLanguage{german}%
```

```
5008 {%
```

```
5009 \def\@gls@extramakeindexopts{-g}%
```

```
5010 }%
```

```
5011 {}%
```

```
5012 }%
```

```
5013 {}%
```

Need to redefine `\@gls@checkquote`

```
5014 \edef\@gls@docheckquotedef{%
```

```
5015 \noexpand\def\noexpand\@gls@checkquote####1#1####2#1####3\noexpand\null{%
```

```
5016 \noexpand\@gls@tmpb=\noexpand\expandafter{\noexpand\@gls@checkedmkidx}%
```

```
5017 \noexpand\toks@={####1}%
```

```
5018 \noexpand\ifx\noexpand\null####2\noexpand\null
```

```
5019 \noexpand\ifx\noexpand\null####3\noexpand\null
```

```
5020 \noexpand\edef\noexpand\@gls@checkedmkidx{%
```

```
5021 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
```

```
5022 \noexpand\def\noexpand\@gls@checkquote{\noexpand\relax}%
```

```
5023 \noexpand\else
```

```
5024 \noexpand\edef\noexpand\@gls@checkedmkidx{%
```

```
5025 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
```

```
5026 \noexpand\@gls@quotechar\noexpand\@gls@quotechar
```

```
5027 \noexpand\@gls@quotechar\noexpand\@gls@quotechar}%
```

```
5028 \noexpand\def\noexpand\@gls@checkquote{%
```

```
5029 \noexpand\@gls@checkquote####3\noexpand\null}%
```

```
5030 \noexpand\fi
```

```
5031 \noexpand\else
```

```
5032 \noexpand\edef\noexpand\@gls@checkedmkidx{%
```

```
5033 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
```

```
5034 \noexpand\@gls@quotechar\noexpand\@gls@quotechar}%
```

```
5035 \noexpand\ifx\noexpand\null####3\noexpand\null
```

```
5036 \noexpand\def\noexpand\@gls@checkquote{%
```

```
5037 \noexpand\@gls@checkquote####2#1#1\noexpand\null}%
```

```
5038 \noexpand\else
```

```
5039 \noexpand\def\noexpand\@gls@checkquote{%
```

```
5040 \noexpand\@gls@checkquote####2#1#1####3\noexpand\null}%
```

```

5041     \noexpand\fi
5042     \noexpand\fi
5043     \noexpand\@gls@checkquote
5044 }%
5045 }%
5046 \@gls@docheckquotedef
5047 \edef\@gls@docheckquotedef{%
5048     \noexpand\renewcommand{\noexpand\@gls@checkmkidxchars}[1]{%
5049     \noexpand\def\noexpand\@gls@checkedmkidx{%
5050     \noexpand\expandafter\noexpand\@gls@checkquote####1\noexpand\@nil
5051     #1#1\noexpand\null
5052     \noexpand\expandafter\noexpand\@gls@updatechecked
5053     \noexpand\@gls@checkedmkidx{####1}%
5054     \noexpand\def\noexpand\@gls@checkedmkidx{%
5055     \noexpand\expandafter\noexpand\@gls@checkescquote####1\noexpand\@nil
5056     \expandonce{\csname#1\endcsname}\expandonce{\csname#1\endcsname}%
5057     \noexpand\null
5058     \noexpand\expandafter\noexpand\@gls@updatechecked
5059     \noexpand\@gls@checkedmkidx{####1}%
5060     \noexpand\def\noexpand\@gls@checkedmkidx{%
5061     \noexpand\expandafter\noexpand\@gls@checkescactual####1\noexpand\@nil
5062     \noexpand\?\noexpand\?\noexpand\null
5063     \noexpand\expandafter\noexpand\@gls@updatechecked
5064     \noexpand\@gls@checkedmkidx{####1}%
5065     \noexpand\def\noexpand\@gls@checkedmkidx{%
5066     \noexpand\expandafter\noexpand\@gls@checkactual####1\noexpand\@nil
5067     \noexpand?\noexpand?\noexpand\null
5068     \noexpand\expandafter\noexpand\@gls@updatechecked
5069     \noexpand\@gls@checkedmkidx{####1}%
5070     \noexpand\def\noexpand\@gls@checkedmkidx{%
5071     \noexpand\expandafter\noexpand\@gls@checkbar####1\noexpand\@nil
5072     \noexpand|\noexpand|\noexpand\null
5073     \noexpand\expandafter\noexpand\@gls@updatechecked
5074     \noexpand\@gls@checkedmkidx{####1}%
5075     \noexpand\def\noexpand\@gls@checkedmkidx{%
5076     \noexpand\expandafter\noexpand\@gls@checkesbar####1\noexpand\@nil
5077     \noexpand\|\noexpand\|\noexpand\null
5078     \noexpand\expandafter\noexpand\@gls@updatechecked
5079     \noexpand\@gls@checkedmkidx{####1}%
5080     \noexpand\def\noexpand\@gls@checkedmkidx{%
5081     \noexpand\expandafter\noexpand\@gls@checklevel####1\noexpand\@nil
5082     \noexpand!\noexpand!\noexpand\null
5083     \noexpand\expandafter\noexpand\@gls@updatechecked
5084     \noexpand\@gls@checkedmkidx{####1}%
5085 }%
5086 }%
5087 \@gls@docheckquotedef
5088 \edef\@gls@docheckquotedef{%
5089     \noexpand\def\noexpand\@gls@checkescquote####1%

```

```

5090 \expandonce{\csname#1\endcsname}###2\expandonce{\csname#1\endcsname}%
5091 ###3\noexpand\null{%
5092 \noexpand\@gls@tmpb=\noexpand\expandafter{\noexpand\@gls@checkedmkidx}%
5093 \noexpand\toks@={###1}%
5094 \noexpand\ifx\noexpand\null###2\noexpand\null
5095 \noexpand\ifx\noexpand\null###3\noexpand\null
5096 \noexpand\edef\noexpand\@gls@checkedmkidx{%
5097 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
5098 \noexpand\def\noexpand\@gls@checkescquote{\noexpand\relax}%
5099 \noexpand\else
5100 \noexpand\edef\noexpand\@gls@checkedmkidx{%
5101 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
5102 \noexpand\@gls@quotechar\noexpand\string\expandonce{%
5103 \csname#1\endcsname}\noexpand\@gls@quotechar
5104 \noexpand\@gls@quotechar\noexpand\string\expandonce{%
5105 \csname#1\endcsname}\noexpand\@gls@quotechar}%
5106 \noexpand\def\noexpand\@gls@checkescquote{%
5107 \noexpand\@gls@checkescquote###3\noexpand\null}%
5108 \noexpand\fi
5109 \noexpand\else
5110 \noexpand\edef\noexpand\@gls@checkedmkidx{%
5111 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
5112 \noexpand\@gls@quotechar\noexpand\string
5113 \expandonce{\csname#1\endcsname}\noexpand\@gls@quotechar}%
5114 \noexpand\ifx\noexpand\null###3\noexpand\null
5115 \noexpand\def\noexpand\@gls@checkescquote{%
5116 \noexpand\@gls@checkescquote###2\expandonce{\csname#1\endcsname}%
5117 \expandonce{\csname#1\endcsname}\noexpand\null}%
5118 \noexpand\else
5119 \noexpand\def\noexpand\@gls@checkescquote{%
5120 \noexpand\@gls@checkescquote###2\expandonce{\csname#1\endcsname}%
5121 ###3\noexpand\null}%
5122 \noexpand\fi
5123 \noexpand\fi
5124 \noexpand\@gls@checkescquote
5125 }%
5126 }%
5127 \@gls@docheckquotedef
5128 }
5129 \newcommand*{\gls@nosetquote}[1]{\PackageError{glossaries}%
5130 {\string\GlsSetQuote\space not permitted here}%
5131 {Move \string\GlsSetQuote\space earlier in the preamble, as
5132 soon as possible after glossaries.sty has been loaded}}
5133 \fi

```

ramakeindexopts

```
5134 \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`

```
5135 \newcommand{\noist}{%
```

```
    Update attributes list
```

```
5136 \@gls@addpredefinedattributes
```

```
5137 \let\writeist\relax
```

```
5138 }
```

`\@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`

```
5139 \newcommand*{\@makeglossary}[1]{%
```

```
5140 \ifglossaryexists{#1}%
```

```
5141 {%
```

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

```
5142 \ifglssavewrites
```

```
5143 \expandafter\newtoks\csname glo@#1@filetok\endcsname
```

```
5144 \else
```

```
5145 \expandafter\newwrite\csname glo@#1@file\endcsname
```

```
5146 \expandafter\@glsopenfile\csname glo@#1@file\endcsname{#1}%
```

```
5147 \fi
```

```
5148 \@gls@renewglossary
```

```
5149 \writeist
```

```
5150 }%
```

```
5151 {%
```

```
5152 \PackageError{glossaries}%
```

```
5153 {Glossary type ‘#1’ not defined}%
```

```
5154 {New glossaries must be defined before using \string\makeglossaries}%
```

```
5155 }%
```

```
5156 }
```

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

```
5157 \newcommand*{\@glsopenfile}[2]{%
```

```
5158 \immediate\openout#1=\jobname.\csname @glotype@#2@out\endcsname
```

```
5159 \PackageInfo{glossaries}{Writing glossary file
```

```
5160 \jobname.\csname @glotype@#2@out\endcsname}%
```

```
5161 }
```

`\@closegls`

```

5162 \newcommand*{\@closegls}[1]{%
5163   \closeout\csname glo@#1@file\endcsname
5164 }

```

\@gls@automake

```

5165 \ifglsxindy
5166 \newcommand*{\@gls@automake}[1]{%
5167   \ifglossaryexists{#1}
5168   {%
5169     \@closegls{#1}%
5170     \ifdefstring{\glsorder}{letter}%
5171     {\def\@gls@order{-M ord/letorder }}%
5172     {\let\@gls@order\@empty}%
5173     \ifcsundef{\xdy@#1@language}%
5174     {\let\@gls@langmod\@xdy@main@language}%
5175     {\letcs\@gls@langmod{\xdy@#1@language}}%
5176     \edef\@gls@dothiswrite{\noexpand\write18{xindy
5177       -I xindy
5178       \@gls@order
5179       -L \@gls@langmod\space
5180       -M \gls@istfilename\space
5181       -C \gls@codepage\space
5182       -t \jobname.\csuse{@glotype@#1@log}
5183       -o \jobname.\csuse{@glotype@#1@in}
5184       \jobname.\csuse{@glotype@#1@out}}}%
5185     }%
5186     \@gls@dothiswrite
5187   }%
5188   {%
5189     \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
5190   }%
5191 }
5192 \else
5193 \newcommand*{\@gls@automake}[1]{%
5194   \ifglossaryexists{#1}
5195   {%
5196     \@closegls{#1}%
5197     \ifdefstring{\glsorder}{letter}%
5198     {\def\@gls@order{-l }}%
5199     {\let\@gls@order\@empty}%
5200     \edef\@gls@dothiswrite{\noexpand\write18{makeindex \@gls@order
5201       -s \istfilename\space
5202       -t \jobname.\csuse{@glotype@#1@log}
5203       -o \jobname.\csuse{@glotype@#1@in}
5204       \jobname.\csuse{@glotype@#1@out}}}%
5205     }%
5206     \@gls@dothiswrite
5207   }%
5208   {%

```

```

5209     \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
5210 }%
5211 }
5212 \fi

```

omake@immediate

```

5213 \ifglxsindy
5214 \newcommand*{\@gls@automake@immediate}[1]{%
5215   \ifglossaryexists{#1}
5216   {%
5217     \IfFileExists{\jobname.\csuse{@glotype@#1@out}}%
5218     {%
5219       \ifdefstring{\glsorder}{letter}%
5220       {\def\@gls@order{-M ord/letorder }}%
5221       {\let\@gls@order\@empty}%
5222       \ifcsundef{@xdy@#1@language}%
5223       {\let\@gls@langmod\@xdy@main@language}%
5224       {\letcs\@gls@langmod{@xdy@#1@language}}%
5225       \edef\@gls@dothiswrite{\noexpand\immediate\noexpand\write18{xindy
5226         -I xindy
5227         \@gls@order
5228         -L \@gls@langmod\space
5229         -M \gls@istfilebase\space
5230         -C \gls@codepage\space
5231         -t \jobname.\csuse{@glotype@#1@log}
5232         -o \jobname.\csuse{@glotype@#1@in}
5233         \jobname.\csuse{@glotype@#1@out}}%
5234       }%
5235       \@gls@dothiswrite
5236     }%
5237     {\GlossariesWarning{can't automake '#1': \jobname.\csuse{@glotype@#1@out}
5238       doesn't exist. Rerun may be required}}%
5239   }%
5240   {%
5241     \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
5242   }%
5243 }
5244 \else
5245 \newcommand*{\@gls@automake@immediate}[1]{%
5246   \ifglossaryexists{#1}
5247   {%
5248     \IfFileExists{\jobname.\csuse{@glotype@#1@out}}%
5249     {%
5250       \ifdefstring{\glsorder}{letter}%
5251       {\def\@gls@order{-l }}%
5252       {\let\@gls@order\@empty}%
5253       \edef\@gls@dothiswrite{\noexpand\immediate\noexpand\write18{makeindex \@gls@order
5254         -s \istfilename\space
5255         -t \jobname.\csuse{@glotype@#1@log}

```

```

5256         -o \jobname.\csuse{@glotype@#1@in}
5257         \jobname.\csuse{@glotype@#1@out}}%
5258     }%
5259     \@gls@dothiswrite
5260 }%
5261 {\GlossariesWarning{can't automake '#1': \jobname.\csuse{@glotype@#1@out}
5262     doesn't exist. Rerun may be required}}%
5263 }%
5264 {%
5265     \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
5266 }%
5267 }
5268 \fi

```

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

```

5269 \newcommand*{\@warn@nomakeglossaries}{}
    Only use this if warning if \printglossary has been used without \makeglossaries
5270 \newcommand*{\@warn@nomakeglossaries}{\@warn@nomakeglossaries}

```

`\make@immediate`

```

5271 \newcommand{\@gls@@automake@immediate}{%
5272     \ifnum\gls@automake@nr=2\relax
5273     \@for\@gls@type:=\@glo@types\do{%
5274         \ifdefempty{\@gls@type}{}%
5275         {\@gls@automake@immediate{\@gls@type}}%
5276     }%
5277     \glsautomakefalse
5278     \renewcommand*{\@gls@doautomake}{}%
5279 \fi
5280 }

```

`\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`

```

5281 \newcommand*{\makeglossaries}{%
    If automake=immediate setting is on, use the shell escape now.
5282     \@gls@@automake@immediate
    Define the write used for style file also used for all other output files if savewrites=true.
5283     \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.
5284     \protected@write\@auxout{}{\string\providecommand\string\@glsorder[1]{}
5285     \protected@write\@auxout{}{\string\providecommand\string\@istfilename[1]{}

```


If `\@gls@extramakeindexopts` has been defined, write it:

```
5286 \ifundef\@gls@extramakeindexopts
5287 {}%
5288 {%
5289   \protected@write\@auxout{}\string\providecommand
5290     \string\@gls@extramakeindexopts[1]{}}
5291   \protected@write\@auxout{}\string\@gls@extramakeindexopts
5292     {\@gls@extramakeindexopts}}%
5293 }%
```

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

```
5294 \protected@write\@auxout{}\string\@istfilename{\istfilename}}%
5295 \protected@write\@auxout{}\string\@glsorder{\glsorder}}
```

Iterate through each glossary type and activate it.

```
5296 \@for\@glo@type:=\@glo@types\do{%
5297   \ifthenelse{equal{\@glo@type}{}}{}}{%
5298   \@makeglossary{\@glo@type}}%
5299 }%
```

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

```
5300 \renewcommand*\newglossary[4] []{%
5301 \PackageError{glossaries}{New glossaries
5302 must be created before \string\makeglossaries}{You need
5303 to move \string\makeglossaries\space after all your
5304 \string\newglossary\space commands}}%
```

Any subsequent instances of this command should have no effect. The deprecated `\makeglossary` is not redefined here as it either implements `\makeglossaries` or has been restored to its original definition (in which case it shouldn't be changed).

```
5305 \let\@makeglossary\relax
5306 \let\makeglossaries\relax
```

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

```
5307 \@disable@onlypremakeg
```

Allow see key:

```
5308 \let\gls@checkseeallowed\relax
```

Suppress warning about no `\makeglossaries`

```
5309 \let\warn@nomakeglossaries\relax
```

Activate warning about missing `\printglossary`

```
5310 \def\warn@noprintglossary{%
5311   \ifdefstring{\@glo@types}{,}%
5312   {%
5313     \GlossariesWarningNoLine{No glossaries have been defined}}%
5314   }%
5315   {%
5316     \GlossariesWarningNoLine{No \string\printglossary\space
5317       or \string\printglossaries\space
5318       found. ^^J(Remove \string\makeglossaries\space if you
```

```

5319         don't want any glossaries.) ^^JThis document will not
5320         have a glossary}%
5321     }%
5322 }%

```

Declare list parser for `\glsdisplaynumberlist`

```

5323 \ifglssavenumberlist
5324   \edef\@gls@dodolistparser{\noexpand\DeclareListParser
5325     {\noexpand\glsnumlistparser}{\delimN}}}%
5326   \@gls@dodolistparser
5327 \fi

```

Prevent user from also using `\makenoidxglossaries`

```

5328 \let\makenoidxglossaries\@no@makeglossaries

```

Prohibit sort key in printgloss family:

```

5329 \renewcommand*{\@printgloss@setsort}{%
5330   \let\@glo@assign@sortkey\@glo@no@assign@sortkey
5331 }%

```

Check the automake setting:

```

5332 \ifglsautomake
5333   \renewcommand*{\@gls@doautomake}{%
5334     \@for\@gls@type:=\@glo@types\do{%
5335       \ifdefempty{\@gls@type}{}%
5336       {\@gls@automake{\@gls@type}}}%
5337   }%
5338 }%
5339 \fi

```

Check the sort setting:

```

5340 \@glo@check@sortallowed\makeglossaries
5341 }

```

Must occur in the preamble:

```

5342 \@onlypreamble{\makeglossaries}

```

`\glswrite` The definition of `\glswrite` has now been moved to `\makeglossaries` so that it's only defined if needed.

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

```

5343 \AtEndDocument{%
5344   \warn@nomakeglossaries
5345   \warn@noprintglossary
5346 }

```

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

```

5347 \newcommand*{\makenoidxglossaries}{%

```

Redefine empty glossary warning:

```
5348 \renewcommand{\@gls@noref@warn}[1]{%
5349   \GlossariesWarning{Empty glossary for
5350   \string\printnoidxglossary[type={##1}].
5351   Rerun may be required (or you may have forgotten to use
5352   commands like \string\gls)}%
5353 }%
```

Don't escape makeindex/xindy characters:

```
5354 \let\@gls@checkmkidxchars\@gobble
```

Don't escape locations:

```
5355 \glsesclocationsfalse
```

Write glossary information to aux instead of glossary files

```
5356 \let\@do@wrglossary\gls@noidxglossary
```

Switch on group headings that use the character code:

```
5357 \let\@gls@getgrouptitle\@gls@noidx@getgrouptitle
```

Allow see key:

```
5358 \let\gls@checkseeallowed\relax
```

Redefine cross-referencing macro:

```
5359 \renewcommand{\@do@seeglossary}[2]{%
5360   \edef\@gls@label{\glsdetoklabel{##1}}%
5361   \protected@write\@auxout{}{%
5362     \string\@gls@reference
5363     {\csname glo@\@gls@label @type\endcsname}%
5364     {\@gls@label}%
5365     {%
5366       \string\glsseeformat##2}%
5367     }%
5368   }%
5369 }%
```

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.

```
5370 \AtBeginDocument
5371 {%
5372   \write\@auxout{\string\providecommand\string\@gls@reference[3]{}}%
5373 }%
```

Change warning about no glossaries

```
5374 \def\warn@noprntglossary{%
5375   \GlossariesWarningNoLine{No \string\printnoidxglossary\space
5376   or \string\printnoidxglossaries ^^J
5377   found. (Remove \string\makenoidxglossaries\space if you
5378   don't want any glossaries.)^^JThis document will not have a glossary}%
5379 }%
```

Suppress warning about no \makeglossaries

```
5380 \let\warn@nomakeglossaries\relax
```

```

Prevent user from also using \makeglossaries
5381 \let\makeglossaries\@no@makeglossaries

Allow sort key in printgloss family:
5382 \renewcommand*{\@printgloss@setsort}{%
5383 \let\@glo@assign@sortkey\@glo@assign@sortkey

Initialise default sort order:
5384 \def\@glo@sorttype{\@glo@default@sorttype}%
5385 }%

All entries must be defined in the preamble:
5386 \renewcommand*\new@glossaryentry[2]{%
5387 \PackageError{glossaries}{Glossary entries must be
5388 defined in the preamble^^Jwhen you use
5389 \string\makenoidxglossaries}%
5390 {Either move your definitions to the preamble or use
5391 \string\makeglossaries}%
5392 }%

Redefine \glsentrynumberlist
5393 \renewcommand*{\glsentrynumberlist}[1]{%
5394 \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
5395 \ifdef\@gls@loclist
5396 {%
5397 \glsnoidxloclist{\@gls@loclist}%
5398 }%
5399 {%
5400 ??\glsdoifexists{##1}%
5401 {%
5402 \GlossariesWarning{Missing location list for ‘##1’. Either
5403 a rerun is required or you haven’t referenced the entry}%
5404 }%
5405 }%
5406 }%

Redefine \glsdisplaynumberlist
5407 \renewcommand*{\glsdisplaynumberlist}[1]{%
5408 \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
5409 \ifdef\@gls@loclist
5410 {%
5411 \def\@gls@noidxloclist@sep{%
5412 \def\@gls@noidxloclist@sep{%
5413 \def\@gls@noidxloclist@sep{%
5414 \glsnumlistsep
5415 }%
5416 \def\@gls@noidxloclist@finalsep{\glsnumlistlastsep}%
5417 }%
5418 }%
5419 \def\@gls@noidxloclist@finalsep{}%
5420 \def\@gls@noidxloclist@prev{}%

```

```

5421     \forlistloop{\glsnoidxdisplayloclisthandler}{\@gls@loclist}%
5422     \@gls@noidxloclist@finalsep
5423     \@gls@noidxloclist@prev
5424   }%
5425   {%
5426     ??\glsdoifexists{##1}%
5427     {%
5428       \GlossariesWarning{Missing location list for ‘##1’. Either
5429         a rerun is required or you haven’t referenced the entry}%
5430     }%
5431   }%
5432 }%

```

Provide a generic way of iterating through the number list:

```

5433 \renewcommand*\glsnumberlistloop}[3]{%
5434   \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
5435   \let\@gls@org@glsnoidxdisplayloc\glsnoidxdisplayloc
5436   \let\@gls@org@glsseeformat\glsseeformat
5437   \let\glsnoidxdisplayloc##2\relax
5438   \let\glsseeformat##3\relax
5439   \ifdef\@gls@loclist
5440   {%
5441     \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
5442   }%
5443   {%
5444     ??\glsdoifexists{##1}%
5445     {%
5446       \GlossariesWarning{Missing location list for ‘##1’. Either
5447         a rerun is required or you haven’t referenced the entry}%
5448     }%
5449   }%
5450   \let\glsnoidxdisplayloc\@gls@org@glsnoidxdisplayloc
5451   \let\glsseeformat\@gls@org@glsseeformat
5452 }%

```

Modify sanitize sort function

```

5453 \let\@gls@sanitizesort\@gls@noidx@sanitizesort
5454 \let\@gls@nosanitizesort\@gls@noidx@nosanitizesort
5455 \@gls@noidx@setsanitizesort

```

Check sort option allowed.

```

5456 \@glo@check@sortallowed\makenoidxglossaries
5457 }

```

Preamble-only command:

```

5458 \@onlypreamble{\makenoidxglossaries}

```

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

```

5459 \newcommand*\glsnumberlistloop}[2]{%
5460   \PackageError{glossaries}{\string\glsnumberlistloop\space
5461     only works with \string\makenoidxglossaries}{}%
5462 }

```

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

```

5463 \newcommand*\glsnoidxnumberlistloophandler}[1]{%
5464   #1%
5465 }

```

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

```

5466 \newcommand*\@no@makeglossaries{%
5467   \PackageError{glossaries}{You can't use both
5468     \string\makeglossaries\space and \string\makenoidxglossaries}%
5469   {Either use one or other (or none) of those commands but not both
5470     together.}%
5471 }

```

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

```

5472 \newcommand{\@gls@noref@warn}[1]{%
5473   \GlossariesWarning{\string\makenoidxglossaries\space
5474     is required to make \string\printnoidxglossary[type={#1}] work}%
5475 }

```

1.14 Writing information to associated files

`s@noidxglossary` Write the glossary information to the aux file (for the 'noidx' method):

```

5476 \newcommand*\@gls@noidxglossary{%
5477   \protected@write\@auxout{}{%
5478     \string\@gls@reference
5479     {\csname glo@\@gls@label @type\endcsname}%
5480     {\@gls@label}%
5481     {\string\glsnoidxdisplayloc
5482       {\@glo@counterprefix}%
5483       {\@gls@counter}%
5484       {\@glsnumberformat}%
5485       {\@glslocref}%
5486     }%
5487   }%
5488 }

```

`\istfile` Deprecated.

```

5489 \providecommand\istfile{\glswrite}

```

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

```

5490 \AtEndDocument{%

```

```
5491 \glswritefiles
5492 }
```

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

```
5493 \newcommand*{\@glswritefiles}{%
```

Iterate through all the glossaries.

```
5494 \forallglossaries{\@glo@type}{%
```

Check for empty glossaries (patch provided by Patrick Häcker)

```
5495 \ifcsundef{glo@\@glo@type @filetok}%
```

```
5496 {%
```

```
5497 \def\gls@tmp{}
```

```
5498 }%
```

```
5499 {%
```

```
5500 \edef\gls@tmp{\expandafter\the
```

```
5501 \csname glo@\@glo@type @filetok\endcsname}%
```

```
5502 }%
```

```
5503 \ifx\gls@tmp\@empty
```

```
5504 \ifx\@glo@type\glsdefaulttype
```

```
5505 \GlossariesWarningNoLine{Glossary ‘\@glo@type’ has no
```

```
5506 entries.^^JRemember to use package option ‘nomain’ if
```

```
5507 you
```

```
5508 \don’t want to^^Juse the main glossary}%
```

```
5509 \else
```

```
5510 \GlossariesWarningNoLine{Glossary ‘\@glo@type’ has no
```

```
5511 entries}%
```

```
5512 \fi
```

```
5513 \else
```

```
5514 \@glsopenfile{\glswrite}{\@glo@type}%
```

```
5515 \immediate\write\glswrite{%
```

```
5516 \expandafter\the
```

```
5517 \csname glo@\@glo@type @filetok\endcsname}%
```

```
5518 \immediate\closeout\glswrite
```

```
5519 \fi
```

```
5520 }%
```

```
5521 }
```

As from v4.10, the `\glossary` command isn't 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).

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

`\gls@glossary`

```
5522 \newcommand*{\gls@glossary}[1]{%
```

```
5523 \@gls@glossary{#1}%
```

```
5524 }
```

```
\@gls@glossary
```

<code>\@gls@glossary{<type>}{<indexing info>}</code>
--

(In v4.10, \@glossary was redefined to \@gls@glossary to avoid conflict with other packages.) Initially define internal \@gls@glossary to ignore its argument. Indexing will be enabled when \@gls@glossary is redefined by \@makeglossary.

This command was originally defined to do \@index{<indexing info>} so that it behaved much like \index. The definition was then changed to use \index as memoir changes the definition of \@index. (Thanks to Dan Luecking for pointing this out.)

However, if normal indexing is enabled (for example with \makeindex) but no glossary lists are required (so \@makeglossary isn't used), then \index will cause a problem here. The \@index trick allows for special characters within <indexing info> (so you can do, for example, \index{%@\%}), and the original design of \@glossary here was actually a legacy from the old glossary package. With the glossaries package, the indexing information supplied in the second argument is more constrained and just consists of the sort value (given by the sort key), the actual value (given by \glossentry{<label>} or \subglossentry{<level>}{<label>}), and the format. This means that there's no need to worry about special characters appearing in the second argument as they can't be in the label or sort value. (If they are in the sort value then the category code would've needed to be changed when the entry was defined or \glspercentchar would be needed with the sort sanitization switched off.) This means that it's safe to simply ignore the second argument.

```
5525 \newcommand*{\@gls@glossary}[2]{%
5526   \ifgls@debug
5527     \PackageInfo{glossaries}{wrglossary(#1)(#2)}%
5528   \fi
5529 }
```

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.

```
@renewglossary
```

```
5530 \newcommand{\@gls@renewglossary}{%
5531   \gdef\@gls@glossary##1{\@bsphack\begingroup\gls@wrglossary{##1}}%
5532   \let\@gls@renewglossary\@empty
5533 }
```

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
```

```
5534 \newcommand*{\gls@wrglossary}[2]{%
5535   \ifglssavewrites
5536     \protected@edef\@gls@tmp{\the\csname glo@#1@filetok\endcsname#2}%
5537     \expandafter\global\expandafter\csname glo@#1@filetok\endcsname
5538       \expandafter{\@gls@tmp^^J}%
5539   \else
```



```

5540 \ifcsdef{glo@#1@file}%
5541 {%
5542   \expandafter\protected@write\csname glo@#1@file\endcsname{%
5543     \gls@disablepagerefexpansion}{#2}%
5544   }%
5545   {%
5546     \ifignoredglossary{#1}{}%
5547     {%
5548       \GlossariesWarning{No file defined for glossary ‘#1’}%
5549     }%
5550   }%
5551 \fi
5552 \endgroup\@esphack
5553 }

```

\do@wrglossary

```

5554 \newcommand*\do@wrglossary}[1]{%
5555   \glswriteentry{#1}{\do@wrglossary{#1}}%
5556 }

```

\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.

```

5557 \newcommand*\glswriteentry}[2]{%
5558   \ifglsindexonlyfirst
5559     \ifglsused{#1}{#2}%
5560   \else
5561     #2%
5562   \fi
5563 }

```

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

```

5564 \newcommand{\gls@protected@pagefmts}{\gls@numberpage,\gls@alphpage,%
5565   \gls@Alphpage,\gls@romanpage,\gls@Romanpage,\gls@arabicpage}

```

pagerefexpansion

```

5566 \newcommand*\gls@disablepagerefexpansion){%
5567   \@for\@gls@this:=\gls@protected@pagefmts\do
5568   {%
5569     \expandafter\let\@gls@this\relax
5570   }%
5571 }

```

\gls@alphpage

```

5572 \newcommand*\gls@alphpage}{\@alph\c@page}

```

\gls@Alphpage

```

5573 \newcommand*\gls@Alphpage}{\@Alph\c@page}

```

`\gls@numberpage`

```
5574 \newcommand*{\gls@numberpage}{\number\c@page}
```

`\gls@arabicpage`

```
5575 \newcommand*{\gls@arabicpage}{\@arabic\c@page}
```

`\gls@romanpage`

```
5576 \newcommand*{\gls@romanpage}{\romannumeral\c@page}
```

`\gls@Romanpage`

```
5577 \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).

```
5578 \newcommand*{\glsaddprotectedpagefmt}[1]{%
5579   \eappto\gls@protected@pagefmts{,\expandonce{\csname gls#1page\endcsname}}%
5580   \csedef{gls#1page}{\expandonce{\csname#1\endcsname}\noexpand\c@page}%
5581   \eappto\@wrglossarynumberhook{%
5582     \noexpand\let\expandonce{\csname org@gls#1\endcsname}%
5583     \expandonce{\csname#1\endcsname}%
5584     \noexpand\def\expandonce{\csname#1\endcsname}{%
5585       \noexpand\@wrglossary@pageformat
5586       \expandonce{\csname gls#1page\endcsname}%
5587       \expandonce{\csname org@gls#1\endcsname}%
5588     }%
5589   }%
5590 }
```

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

```
5591 \newcommand*\@wrglossarynumberhook{}
```

`sary@pageformat`

```
5592 \newcommand{\@wrglossary@pageformat}[3]{%
5593   \ifx#3\c@page #1\else #2#3\fi
5594 }
```

`@do@wrglossary` Write the glossary entry in the appropriate format.

```
5595 \newcommand*{\@do@wrglossary}[1]{%
5596   \ifglsclocations
5597   \@do@esc@wrglossary{#1}%
5598   \else
5599   \@do@noesc@wrglossary{#1}%
5600   \fi
5601 }
```

`\noesc@wrglossary` Write the glossary entry in the appropriate format. The locations don't need to be pre-processed before writing the information to the glossary file, but the prefix still needs to be found.

```
5602 \newcommand*{\@do@noesc@wrglossary}[1]{%
```

Don't fully expand yet.

```
5603 \expandafter\def\expandafter\@glslocref\expandafter{\theglsentrycounter}%
```

```
5604 \expandafter\def\expandafter\@glsHlocref\expandafter{\theHglentrycounter}%
```

Find the prefix if `\@glsHlocref` and `\@glslocref` aren't the same.

```
5605 \ifx\@glsHlocref\@glslocref
```

```
5606 \def\@glo@counterprefix{}
```

```
5607 \else
```

The value of the counter isn't important here as it's the prefix that's of interest. (`\c@page` will have the same value in both `\theglsentrycounter` and `\theHglentrycounter` at this point, even if it hasn't been updated yet. The page number is not expected to occur in the prefix.)

```
5608 \protected@edef\@do@gls@getcounterprefix{\noexpand\@gls@getcounterprefix
```

```
5609 {\@glslocref}{\@glsHlocref}%
```

```
5610 }%
```

```
5611 \@do@gls@getcounterprefix
```

```
5612 \fi
```

De-tok label if required.

```
5613 \edef\@gls@label{\glsdetoklabel{#1}}%
```

Write the information to file:

```
5614 \@do@@wrglossary
```

```
5615 }
```

`\owprimitivemods` Conditional to determine whether or not `\@do@esc@wrglossary` should be allowed to temporarily redefine `\the` and `\number`.

```
5616 \newif\ifglswrallowprimitivemods
```

```
5617 \glswrallowprimitivemodstrue
```

`\@esc@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. This is far more complicated with `xindy` than with other indexing methods. There are two necessary but conflicting requirements with `xindy`:

1. all backslashes in the location must be escaped;
2. `\c@page` can't be prematurely expanded.

(With `makeindex` there's the remote possibility that the page compositor is a `makeindex` special character, so that would also need to be escaped.)

For example, suppose `\thepage` is defined as

```
\renewcommand{\thepage}{\tally{page}}
```

```
\newcommand{\tally}[1]{\tallynum{\expandafter\the\c@name c@#1\endcsname}}
```

where `\tallynum` is a robust command that takes a number as its argument. With all indexing methods other than `xindy`, a deferred write with `\thepage` as the location will expand to `\tallynum{<n>}` where `<n>` is the page number. Since the write is deferred, the page number is correct. (`makeindex` won't accept this location format, but `\makenoidxglossaries` and `bib2gls` are quite happy with it.) Unfortunately, this fails with `xindy` because `xindy` interprets this location as `tallynum{<n>}` because `\t` represents a the character "t". The location must be written as `\\tallynum{<n>}`.

This means that the location `\tally{page}` must be expanded and then the backslashes must be doubled. Unfortunately `\c@page` mustn't be expanded until the deferred write is performed, so the location actually needs to be expanded to `\tallynum{\the\c@page}` but the backslashes in `\the\c@page` mustn't be escaped. All other backslashes must be escaped. (In this case, only the backslash in `\tallynum` but the location format may include other control sequences.) The code below works on the assumption that commands like `\tally` are defined in the form

```
\newcommand{\tally}[1]{\tallynum{\expandafter\the\csname c@#1\endcsname}}
```

(note the use of `\expandafter` and `\name`) or in the form

```
\newcommand{\tally}[1]{\tallynum{\arabic{#1}}}
```

In the second case, `\arabic` is one of the known commands that's temporarily adjusted to prevent `\c@page` from being prematurely expanded. In the first case, `\the` is temporarily modified (unless `\glswrallowprimitivemodsfalse`) to check if it's followed by `\c@page`. The `\expandafter` ensures that it is. If `\tally` is defined in another way that hides `\c@page` for example using `\the\value{#1}` then the process fails.

With `makeindex`, `\tallynum` needs to expand to just the decimal number while writing the location to the glossary file, otherwise `makeindex` will reject it. This can be done by defining `\glstallypage` so that `\tally` can locally be set to `\arabic` while expansion is occurring. Again, `\c@page` must be protected from expansion until the deferred write occurs.

The expansion before the write occurs also allows the hyper prefix to be determined where `\theH<counter>` is defined in the form `<prefix>.\the<counter>`. It's possible (although again unlikely) that a `makeindex` character might occur in the prefix, which therefore needs escaping. The prefix is passed as the optional argument of `\setentrycounter` which is needed by commands like `\glshypernumber` to create a hyperlink for a given counter (like `\hyperpage` but for an arbitrary counter).

```
5618 \newcommand*{\@@do@esc@wrglossary}[1]{% please read documented code!
5619 \begingroup
```

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

```
5620 \let\gls@orgthe\the
5621 \let\gls@orgnumber\number
5622 \let\gls@orgarabic\@arabic
5623 \let\gls@orgromannumeral\romannumeral
5624 \let\gls@orgalph\@alph
5625 \let\gls@orgAlph\@Alph
```

```
5626 \let\gls@orgRoman\@Roman
```

Redefine:

```
5627 \ifglswrallowprimitivemods
```

The redefinition of `\the` to use `\expandafter` solves the problem of `\the\csname c@<counter>\endcsname` but is only a partial solution to the problem of `\the\value`. With `\value`, `\c@page` is too deeply hidden and will be expanded too soon, but at least there won't be an error.

```
5628 \def\gls@the##1{%
5629 \ifx##1\c@page \gls@numberpage\else\gls@orgthe##1\fi}%
5630 \def\the{\expandafter\gls@the}%
5631 \def\gls@number##1{%
5632 \ifx##1\c@page \gls@numberpage\else\gls@orgnumber##1\fi}%
5633 \def\number{\expandafter\gls@number}%
5634 \fi
5635 \def\@arabic##1{%
5636 \ifx##1\c@page \gls@arabicpage\else\gls@orgarabic##1\fi}%
5637 \def\romannumeral##1{%
5638 \ifx##1\c@page \gls@romanpage\else\gls@orgromannumeral##1\fi}%
5639 \def\@Roman##1{%
5640 \ifx##1\c@page \gls@Romanpage\else\gls@orgRoman##1\fi}%
5641 \def\@alph##1{%
5642 \ifx##1\c@page \gls@alphpage\else\gls@orgalph##1\fi}%
5643 \def\@Alph##1{%
5644 \ifx##1\c@page \gls@Alphpage\else\gls@orgAlph##1\fi}%
```

Add hook to allow for other number formats:

```
5645 \@wrglossarynumberhook
```

Prevent expansion:

```
5646 \gls@disablepagerefexpansion
```

Now store location in `\@glslocref`:

```
5647 \protected@xdef\@glslocref{\theHglentrycounter}%
5648 \endgroup
```

Escape any special characters. It's possible that with `makeindex` the separator might be a `makeindex` special character. Although not likely, it still needs to be taken into account.

```
5649 \@gls@checkmkidxchars\@glslocref
```

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

```
5650 \expandafter\ifx\theHglentrycounter\theHglentrycounter\relax
5651 \def\@glo@counterprefix{}%
5652 \else
5653 \protected@edef\@glsHlocref{\theHglentrycounter}%
5654 \@gls@checkmkidxchars\@glsHlocref
5655 \edef\@do@gls@getcounterprefix{\noexpand\@gls@getcounterprefix
5656 {\@glslocref}{\@glsHlocref}%
5657 }%
5658 \@do@gls@getcounterprefix
5659 \fi
```

De-tok label if required

```
5660 \edef\@gls@label{\glsdetoklabel{#1}}%
```

Write the information to file:

```
5661 \@do@wrglossary
```

```
5662 }
```

@do@wrglossary

```
5663 \newcommand*{\@do@wrglossary}{%
```

Determine whether to use xindy or makeindex syntax

```
5664 \ifglsxindy
```

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

```
5665 \expandafter\@glo@check@mkidxrangear\@glsnumberformat\@nil
```

```
5666 \def\@glo@range{}%
```

```
5667 \expandafter\if\@glo@prefix(\relax
```

```
5668 \def\@glo@range{:open-range}%
```

```
5669 \else
```

```
5670 \expandafter\if\@glo@prefix)\relax
```

```
5671 \def\@glo@range{:close-range}%
```

```
5672 \fi
```

```
5673 \fi
```

Write to the glossary file using xindy syntax.

```
5674 \gls@glossary{\csname glo@\@gls@label @type\endcsname}{%
```

```
5675 (indexentry :tkey (\csname glo@\@gls@label @index\endcsname)
```

```
5676 :locref \string"\@glo@counterprefix}{\@glslocref}\string" %
```

```
5677 :attr \string"\@gls@counter\@glo@suffix\string"
```

```
5678 \@glo@range
```

```
5679 )
```

```
5680 }%
```

```
5681 \else
```

Convert the format information into the format required for makeindex

```
5682 \@set@glo@numformat{\@glo@numfmt}{\@gls@counter}{\@glsnumberformat}%
```

```
5683 {\@glo@counterprefix}%
```

Write to the glossary file using makeindex syntax.

```
5684 \gls@glossary{\csname glo@\@gls@label @type\endcsname}{%
```

```
5685 \string@glossaryentry{\csname glo@\@gls@label @index\endcsname
```

```
5686 \@gls@encapchar\@glo@numfmt}{\@glslocref}}%
```

```
5687 \fi
```

```
5688 }
```

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.)

```

5689 \newcommand*\@gls@getcounterprefix[2]{%
5690   \edef\@gls@thisloc{#1}\edef\@gls@thisHloc{#2}%
5691   \ifx\@gls@thisloc\@gls@thisHloc
5692     \def\@glo@counterprefix{}%
5693   \else
5694     \def\@gls@get@counterprefix##1.#1##2\end@getprefix{%
5695       \def\@glo@tmp{##2}%
5696       \ifx\@glo@tmp\@empty
5697         \def\@glo@counterprefix{}%
5698       \else
5699         \def\@glo@counterprefix{##1}%
5700       \fi
5701     }%
5702   \@gls@get@counterprefix#2.#1\end@getprefix

```

Warn if no prefix can be formed.

```

5703   \ifx\@glo@counterprefix\@empty
5704     \GlossariesWarning{Hyper target ‘#2’ can’t be formed by
5705       prefixing^^Jlocation ‘#1’. You need to modify the
5706       definition of \string\theH\@gls@counter^^Jotherwise you
5707       will get the warning: “name{\@gls@counter.#1}’ has been^^J
5708       referenced but does not exist”}%
5709   \fi
5710 \fi
5711 }

```

1.15 Glossary Entry Cross-References

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

```

5712 \newcommand{\@do@seeglossary}[2]{%
5713   \def\@gls@xref{#2}%
5714   \@onelevel@sanitize\@gls@xref
5715   \@gls@checkmkidxchars\@gls@xref
5716   \ifglsxindy
5717     \gls@glossary{\csname glo@#1@type\endcsname}{%
5718       (indexentry
5719         :tkey (\csname glo@#1@index\endcsname)
5720         :xref (\string"\@gls@xref\string")
5721         :attr \string"see\string"
5722       )
5723     }%
5724   \else
5725     \gls@glossary{\csname glo@#1@type\endcsname}{%
5726       \string\glossaryentry{\csname glo@#1@index\endcsname
5727         \@gls@encapchar glsseeformat\@gls@xref}{Z}}%
5728   \fi

```

5729 }

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

```
5730 \def\@gls@fixbraces#1#2#3\@nil{%
5731   \ifx#2[\relax
5732     \@gls@fixbraces#1#2#3\@end@fixbraces
5733   \else
5734     \def#1{{#2#3}}%
5735   \fi
5736 }
```

`@@gls@fixbraces`

```
5737 \def\@@gls@fixbraces#1[#2]#3\@end@fixbraces{%
5738   \def#1{[#2]{#3}}%
5739 }
```

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

```
5740 \DeclareRobustCommand*\glssee[3][\seename]{%
5741   \@do@seeglossary{#2}{#1}{#3}}
5742 \newcommand*\@glssee[3][\seename]{%
5743   \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.

```
5744 \DeclareRobustCommand*\glsseeformat[3][\seename]{%
5745   \emph{#1} \glsseelist{#2}}
```

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

```
5746 \DeclareRobustCommand*\glsseelist[1]{%
```

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

```
5747   \let\@gls@dolast\relax
```

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

```
5748   \let\@gls@donext\relax
```

Iterate through the labels

```
5749   \@for\@gls@thislabel:=#1\do{%
```

Check if on last iteration of loop

```
5750     \ifx\@xfor@nextelement\@nnil
```

```
5751       \@gls@dolast
```

```
5752     \else
```

```
5753       \@gls@donext
```

```
5754     \fi
```

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

```
5755     \expandafter\glsseeitem\expandafter{\@gls@thislabel}%
```


Update separators

```
5756 \let\@gls@dolast\glsseelastsep
5757 \let\@gls@donext\glsseesep
5758 }%
5759 }
```

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

```
5760 \newcommand*{\glsseelastsep}{\space\andname\space}
```

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

```
5761 \newcommand*{\glsseesep}{, }
```

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

```
5762 \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.)

```
5763 \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.

```
5764 \newcommand*{\gls@save@numberlist}[1]{%
5765 \ifglssavenumberlist
5766 \toks@{#1}%
5767 \edef\@do@writeaux@info{%
5768 \noexpand\csgdef{glo@\glscurrententrylabel @numberlist}{\the\toks@}%
5769 }%
5770 \@onelevel@sanitize\@do@writeaux@info
5771 \protected@write\@auxout{}{\@do@writeaux@info}%
5772 \fi
5773 }
```

`\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.)

```
5774 \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.

```
5775 \ifcsundef{printglossary}{}%
5776 {%
```

If `\printglossary` is already defined, issue a warning and undefine it.

```
5777 \@gls@warnonglossdefined
5778 \undef\printglossary
5779 }
```

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

```
5780 \newcommand*{\printglossary}[1][type=\glsdefaulttype]{%
5781 \@printglossary{#1}{\@print@glossary}%
5782 }
```

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`

```
5783 \newcommand*{\printglossaries}{%
5784 \forallglossaries{\@glo@type}{\printglossary[type=\@glo@type]}%
5785 }
```

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

```
5786 \newcommand*{\printnoidxglossary}[1][type=\glsdefaulttype]{%
5787 \@printglossary{#1}{\@print@noidx@glossary}%
5788 }
```

`printnoidxglossaries` Analogous to `\printglossaries`

```
5789 \newcommand*{\printnoidxglossaries}{%
5790 \forallglossaries{\@glo@type}{\printnoidxglossary[type=\@glo@type]}%
5791 }
```

`printgloss@setsort` Initialise to do nothing.

```
5792 \newcommand*{\@printgloss@setsort}{}
```

`preglossaryhook`

```
5793 \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.

```
5794 \newcommand{\@printglossary}[2]{%
```

Set up defaults.

```
5795 \def\@glo@type{\glsdefaulttype}%
5796 \def\glossarytitle{\csname @glo@type@\@glo@type @title\endcsname}%
```

```

5797 \def\glossarytoctitle{\glossarytitle}%
5798 \let\org@glossarytitle\glossarytitle

5799 \def\@glossarystyle{%
5800   \ifx\@glossary@default@style\relax
5801     \GlossariesWarning{No default glossary style provided \MessageBreak
5802       for the glossary '@glo@type'. \MessageBreak
5803       Using deprecated fallback. \MessageBreak
5804       To fix this set the style with \MessageBreak
5805       \string\setglossarystyle\space or use the \MessageBreak
5806       style key=value option}%
5807   \fi
5808 }%
5809 \def\gls\dotocitle{\glssettocitle{\@glo@type}}%

Store current value of \glossaryentrynumbers. (This may be changed via the optional ar-
gument)
5810 \let\@org@glossaryentrynumbers\glossaryentrynumbers

Localise the effects of the optional argument
5811 \bgroup

Activate or deactivate sort key:
5812 \@printgloss@setsort

Determine settings specified in the optional argument.
5813 \setkeys{printgloss}{#1}%

Does the glossary exist?
5814 \ifglossaryexists{\@glo@type}%
5815 {%

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)
5816 \ifx\glossarytitle\org@glossarytitle
5817 \else
5818 \expandafter\let\csname @glo@type@\@glo@type @title\endcsname
5819 \glossarytitle
5820 \fi

Allow a high-level user command to indicate the current glossary
5821 \let\currentglossary\@glo@type

Enable individual number lists to be suppressed.
5822 \let\org@glossaryentrynumbers\glossaryentrynumbers
5823 \let\glsnonextpages\@glsnonextpages

Enable individual number list to be activated:
5824 \let\glsnextpages\@glsnextpages

Enable suppression of description terminators.
5825 \let\nopostdesc\@nopostdesc

```

Set up the entry for the TOC

```
5826 \gls@dotocitle
```

Set the glossary style

```
5827 \@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):

```
5828 \let\gls@org@glossaryentryfield\glossentry
5829 \let\gls@org@glossarysubentryfield\subglossentry
5830 \renewcommand{\glossentry}[1]{%
5831   \xdef\glscurrententrylabel{\glsdetoklabel{##1}}%
5832   \gls@org@glossaryentryfield{##1}%
5833 }%
5834 \renewcommand{\subglossentry}[2]{%
5835   \xdef\glscurrententrylabel{\glsdetoklabel{##2}}%
5836   \gls@org@glossarysubentryfield{##1}{##2}%
5837 }%
```

```
5838 \@gls@preglossaryhook
```

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

```
5839 #2%
5840 }%
5841 {\GlossariesWarning{Glossary ‘\@glo@type’ doesn’t exist}}%
```

End the current scope

```
5842 \egroup
```

Reset `\glossaryentrynumbers`

```
5843 \global\let\glossaryentrynumbers\@org@glossaryentrynumbers
```

Suppress warning about no `\printglossary`

```
5844 \global\let\warn@noprntglossary\relax
5845 }
```

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

```
5846 \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.)

```
5847 \makeatletter
```

Input the glossary file, if it exists.

```
5848 \@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.

```
5849 \IfFileExists{\jobname.\csname @glo@type\@glo@type @in\endcsname}%
5850 {}%
5851 {\null}%
```

If xindy is being used, need to write the language dependent information to the .aux file for makeglossaries.

```
5852 \ifglxindy
5853   \ifcsundef{@xdy@\@glo@type @language}%
5854   {%
5855     \edef\@do@auxoutstuff{%
5856       \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.

```
5857         \noexpand\immediate\noexpand\write\@auxout{%
5858           \string\providecommand\string\@xdylanguage[2]{}}%
5859         \noexpand\immediate\noexpand\write\@auxout{%
5860           \string\@xdylanguage{\@glo@type}{\@xdy@main@language}}%
5861       }%
5862     }%
5863   }%
5864   {%
5865     \edef\@do@auxoutstuff{%
5866       \noexpand\AtEndDocument{%
5867         \noexpand\immediate\noexpand\write\@auxout{%
5868           \string\providecommand\string\@xdylanguage[2]{}}%
5869         \noexpand\immediate\noexpand\write\@auxout{%
5870           \string\@xdylanguage{\@glo@type}{\csname @xdy@\@glo@type
5871             @language\endcsname}}%
5872       }%
5873     }%
5874   }%
5875   \@do@auxoutstuff
5876   \edef\@do@auxoutstuff{%
5877     \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.

```
5878     \noexpand\immediate\noexpand\write\@auxout{%
5879       \string\providecommand\string\@gls@codepage[2]{}}%
5880     \noexpand\immediate\noexpand\write\@auxout{%
5881       \string\@gls@codepage{\@glo@type}{\@gls@codepage}}%
5882   }%
5883 }%
5884 \@do@auxoutstuff
5885 \fi
```

Activate warning if \makeglossaries hasn't been used.

```
5886 \renewcommand*{\@warn@nomakeglossaries}{%
5887   \GlossariesWarningNoLine{\string\makeglossaries\space
5888     hasn't been used,^^Jthe glossaries will not be updated}%
5889 }%
5890 }
```

The sort macros all have the syntax:

```
\@glo@sortmacro@<order>{<type>}
```

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

`glo@sortentries`

```
5891 \newcommand*{\@glo@sortentries}[2]{%
5892   \glosortentrieswarning
5893   \def\@glo@sortinglist{}%
5894   \def\@glo@sortinghandler{#1}%
5895   \edef\@glo@type{#2}%
5896   \forlistcsloop{\@glo@do@sortentries}{@glsref@#2}%
5897   \csdef{@glsref@#2}{}%
5898   \@for\@this@label:=\@glo@sortinglist\do{%
```

Has this entry already been added?

```
5899   \xifinlistcs{\@this@label}{@glsref@#2}%
5900   {}%
5901   {%
5902   \listcsxadd{@glsref@#2}{\@this@label}%
5903   }%
5904   \ifcsdef{@glo@sortingchildren@ \@this@label}%
5905   {%
5906   \@glo@addchildren{#2}{\@this@label}%
5907   }%
5908   {}%
5909   }%
5910 }
```

`@glo@addchildren`

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

```
5911 \newcommand*{\@glo@addchildren}[2]{%
```

Scope to allow nesting.

```
5912   \bgroup
5913   \letcs{\@glo@childlist}{@glo@sortingchildren@#2}%
5914   \@for\@this@childlabel:=\@glo@childlist\do
5915   {%
```

Check this label hasn't already been added.

```
5916   \xifinlistcs{\@this@childlabel}{@glsref@#1}%
5917   {}%
5918   {%
5919   \listcsxadd{@glsref@#1}{\@this@childlabel}%
5920   }%
```

Does this child have children?

```
5921     \ifcsdef{@glo@sortingchildren@\@this@childlabel}%  
5922     {%  
5923     \@glo@addchildren{#1}\@this@childlabel}%  
5924     }%  
5925     {%  
5926     }%  
5927     }%  
5928 \egroup  
5929 }
```

@do@sortentries

```
5930 \newcommand*{\@glo@do@sortentries}[1]{%  
5931 \ifglshasparent{#1}%  
5932 {%
```

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

```
5933 \edef\@glo@parent{\csuse{glo@glstdetoklabel{#1}@parent}}%  
5934 \ifcsundef{@glo@sortingchildren@\@glo@parent}%  
5935 {%  
5936 \csdef{@glo@sortingchildren@\@glo@parent}{}%  
5937 }%  
5938 {}%  
5939 \expandafter\@glo@sortedinsert  
5940 \csname @glo@sortingchildren@\@glo@parent\endcsname{#1}%
```

Has the parent been added?

```
5941 \xifinlistcs{\@glo@parent}{@glstoklabel{#1}@parent}%  
5942 {%
```

Yes, it has so do nothing.

```
5943 }%  
5944 {%
```

No, it hasn't so add it now.

```
5945 \expandafter\@glo@do@sortentries\expandafter{\@glo@parent}%  
5946 }%  
5947 }%  
5948 {%  
5949 \@glo@sortedinsert{\@glo@sortinglist}{#1}%  
5950 }%  
5951 }
```

glo@sortedinsert

```
\@glo@sortedinsert{\@glo@sortinglist}{#1}
```

Insert into list.

```
5952 \newcommand*{\@glo@sortedinsert}[2]{%  
5953 \dtl@insertinto{#2}{#1}\@glo@sortinghandler}%  
5954 }%
```

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 $\#2$).

`sorthandler@word`

```
5955 \newcommand*{\@glo@sorthandler@word}[2]{%
5956   \letcs\@gls@sort@A{glo@glstdetoklabel{#1}@sort}%
5957   \letcs\@gls@sort@B{glo@glstdetoklabel{#2}@sort}%
5958   \edef\glo@do@compare{%
5959     \noexpand\dtlwordindexcompare{\noexpand\dtl@sortresult}%
5960     {\expandonce\@gls@sort@B}%
5961     {\expandonce\@gls@sort@A}%
5962   }%
5963   \glo@do@compare
5964 }
```

`thandler@letter`

```
5965 \newcommand*{\@glo@sorthandler@letter}[2]{%
5966   \letcs\@gls@sort@A{glo@glstdetoklabel{#1}@sort}%
5967   \letcs\@gls@sort@B{glo@glstdetoklabel{#2}@sort}%
5968   \edef\glo@do@compare{%
5969     \noexpand\dtlletterindexcompare{\noexpand\dtl@sortresult}%
5970     {\expandonce\@gls@sort@B}%
5971     {\expandonce\@gls@sort@A}%
5972   }%
5973   \glo@do@compare
5974 }
```

`sorthandler@case` Case-sensitive sort.

```
5975 \newcommand*{\@glo@sorthandler@case}[2]{%
5976   \letcs\@gls@sort@A{glo@glstdetoklabel{#1}@sort}%
5977   \letcs\@gls@sort@B{glo@glstdetoklabel{#2}@sort}%
5978   \edef\glo@do@compare{%
5979     \noexpand\dtlcompare{\noexpand\dtl@sortresult}%
5980     {\expandonce\@gls@sort@B}%
5981     {\expandonce\@gls@sort@A}%
5982   }%
5983   \glo@do@compare
5984 }
```

`thandler@nocase` Case-insensitive sort.

```
5985 \newcommand*{\@glo@sorthandler@nocase}[2]{%
5986   \letcs\@gls@sort@A{glo@glstdetoklabel{#1}@sort}%
5987   \letcs\@gls@sort@B{glo@glstdetoklabel{#2}@sort}%
5988   \edef\glo@do@compare{%
5989     \noexpand\dtlicompare{\noexpand\dtl@sortresult}%
5990     {\expandonce\@gls@sort@B}%
5991     {\expandonce\@gls@sort@A}%
5992   }%
```



```
5993 \glo@do@compare
5994 }
```

@sortmacro@word Sort macro for ‘word’

```
5995 \newcommand*{\@glo@sortmacro@word}[1]{%
5996 \ifdefstring{\@glo@default@sorttype}{standard}%
5997 {%
5998 \@glo@sortentries{\@glo@sorthandler@word}{#1}%
5999 }%
6000 {%
6001 \PackageError{glossaries}{Conflicting sort options:^^J
6002 \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
6003 \string\printnoidxglossary[sort=word]}{}}%
6004 }%
6005 }
```

ortmacro@letter Sort macro for ‘letter’

```
6006 \newcommand*{\@glo@sortmacro@letter}[1]{%
6007 \ifdefstring{\@glo@default@sorttype}{standard}%
6008 {%
6009 \@glo@sortentries{\@glo@sorthandler@letter}{#1}%
6010 }%
6011 {%
6012 \PackageError{glossaries}{Conflicting sort options:^^J
6013 \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
6014 \string\printnoidxglossary[sort=letter]}{}}%
6015 }%
6016 }
```

macro@standard Sort macro for ‘standard’. (Use either ‘word’ or ‘letter’ order.)

```
6017 \newcommand*{\@glo@sortmacro@standard}[1]{%
6018 \ifdefstring{\@glo@default@sorttype}{standard}%
6019 {%
6020 \ifcsdef{\@glo@sorthandler@\glsorder}%
6021 {%
6022 \@glo@sortentries{\csuse{\@glo@sorthandler@\glsorder}}{#1}%
6023 }%
6024 {%
6025 \PackageError{glossaries}{Unknown sort handler ‘\glsorder’}{}}%
6026 }%
6027 }%
6028 {%
6029 \PackageError{glossaries}{Conflicting sort options:^^J
6030 \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
6031 \string\printnoidxglossary[sort=standard]}{}}%
6032 }%
6033 }
```

@sortmacro@case Sort macro for ‘case’

```

6034 \newcommand*\@glo@sortmacro@case}[1]{%
6035 \ifdefstring{\@glo@default@sorttype}{standard}%
6036 {%
6037   \@glo@sortentries{\@glo@sorthandler@case}{#1}%
6038 }%
6039 {%
6040   \PackageError{glossaries}{Conflicting sort options:^^J
6041     \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
6042     \string\printnoidxglossary[sort=case]}{}}%
6043 }%
6044 }

```

ortmacro@nocase Sort macro for 'nocase'

```

6045 \newcommand*\@glo@sortmacro@nocase}[1]{%
6046 \ifdefstring{\@glo@default@sorttype}{standard}%
6047 {%
6048   \@glo@sortentries{\@glo@sorthandler@nocase}{#1}%
6049 }%
6050 {%
6051   \PackageError{glossaries}{Conflicting sort options:^^J
6052     \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
6053     \string\printnoidxglossary[sort=nocase]}{}}%
6054 }%
6055 }

```

o@sortmacro@def Sort macro for 'def'. The order of definition is given in \glo@list@{*type*}.

```

6056 \newcommand*\@glo@sortmacro@def}[1]{%
6057 \def\@glo@sortinglist{}%
6058 \for@gl@sentries[#1]{\@gl@thislabel}%
6059 {%
6060   \xifinlistcs{\@gl@thislabel}{\@gl@ref@#1}%
6061   {%
6062     \list@add{\@glo@sortinglist}{\@gl@thislabel}%
6063   }%
6064   {%

```

Hasn't been referenced.

```

6065   }%
6066 }%
6067 \cslet{\@gl@ref@#1}{\@glo@sortinglist}%
6068 }

```

ortmacro@def@do This won't include parent entries that haven't been referenced.

```

6069 \newcommand*\@glo@sortmacro@def@do}[1]{%
6070 \ifinlistcs{#1}{\@gl@ref@\@glo@type}%
6071 }%
6072 {%
6073   \list@csadd{\@gl@ref@\@glo@type}{#1}%
6074 }%

```

```

6075 \ifcsdef{@glo@sortingchildren@#1}%
6076 {%
6077   \@glo@addchildren{\@glo@type}{#1}%
6078 }%
6079 {}%
6080 }

```

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

```
6081 \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.

```

6082 \newcommand*{\@print@noidx@glossary}{%
6083   \ifcsdef{@glsref@\@glo@type}%
6084   {%

```

Sort the entries:

```

6085   \ifcsdef{@glo@sortmacro@\@glo@sorttype}%
6086   {%
6087     \csuse{@glo@sortmacro@\@glo@sorttype}{\@glo@type}%
6088   }%
6089   {%
6090     \PackageError{glossaries}{Unknown sort handler ‘\@glo@sorttype’}{}%
6091   }%

```

Do the glossary heading and preamble

```

6092   \glossarysection[\glossarytoctitle]{\glossarytitle}%
6093   \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.

```

6094   \def\@gls@currentlettergroup{%
6095     \begin{theglossary}%
6096     \glossaryheader
6097     \glsresetentrylist

```

Iterate through the entries.

```
6098   \forlistcsloop{\@gls@noidx@do}{@glsref@\@glo@type}%
```

Finally end the glossary and do the postamble:

```

6099   \end{theglossary}%
6100   \glossarypostamble
6101 }%
6102 {%

```

```

6103 \@gls@noref@warn{\@glo@type}%
6104 }%
6105 }

```

`\glo@grabfirst`

```

6106 \def\glo@grabfirst#1#2\@nil{%
6107 \def\@gls@firsttok{#1}%
6108 \ifdefempty\@gls@firsttok
6109 {%
6110 \def\@glo@thislettergrp{0}%
6111 }%
6112 {%

```

Sanitize it:

```

6113 \@onelevel@sanitize\@gls@firsttok

```

Fetch the first letter:

```

6114 \expandafter\@glo@grabfirst\@gls@firsttok{}{}\@nil
6115 }%
6116 }

```

`\@glo@grabfirst`

```

6117 \def\@glo@grabfirst#1#2\@nil{%
6118 \ifdefempty\@glo@thislettergrp
6119 {%
6120 \def\@glo@thislettergrp{glssymbols}%
6121 }%
6122 {%
6123 \count@=\uccode‘#1\relax
6124 \ifnum\count@=0\relax
6125 \def\@glo@thislettergrp{glssymbols}%
6126 \else
6127 \ifdefstring\@glo@sorttype{case}%
6128 {%
6129 \count@=‘#1\relax
6130 }%
6131 {%
6132 }%
6133 \edef\@glo@thislettergrp{\the\count@}%
6134 \fi
6135 }%
6136 }

```

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

```

6137 \newcommand{\@gls@noidx@do}[1]{%

```

Get this entry's location list

```

6138 \global\letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%

```

Does this entry have a parent?

```
6139 \ifglshasparent{#1}%  
6140 {%
```

Has a parent.

```
6141 \gls@level=\csuse{glo@glsdetoklabel{#1}@level}\relax  
6142 \ifdefvoid{\@gls@loclist}  
6143 {%  
6144 \subglossentry{\gls@level}{#1}{}%  
6145 }%  
6146 {%  
6147 \subglossentry{\gls@level}{#1}%  
6148 {%  
6149 \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%  
6150 }%  
6151 }%  
6152 }%  
6153 {%
```

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

```
6154 \letcs{\@gls@sort}{glo@glsdetoklabel{#1}@sort}%
```

Fetch the first letter:

```
6155 \expandafter\glo@grabfirst\@gls@sort{ }\@nil  
6156 \ifdefequal{\@glo@thislettergrp}{\@gls@currentlettergroup}%  
6157 {}%  
6158 {%
```

Do the group header:

```
6159 \ifdefempty{\@gls@currentlettergroup}{}%  
6160 {%
```

The group skip may start a new scope, so make a global assignment.

```
6161 \global\let\@glo@thislettergrp\@glo@thislettergrp  
6162 \glsgroupskip  
6163 }%  
6164 \glsgroupheading{\@glo@thislettergrp}%  
6165 }%  
  
6166 \global\let\@gls@currentlettergroup\@glo@thislettergrp
```

Do this entry:

```
6167 \ifdefvoid{\@gls@loclist}  
6168 {%  
6169 \glossentry{#1}{}%  
6170 }%  
6171 {%  
6172 \glossentry{#1}%  
6173 {%  
6174 \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%  
6175 }%  
6176 }%
```

```
6177 }%
6178 }
```

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

Display location list.

```
6179 \newcommand*{\glsnoidxloclist}[1]{%
6180   \def\@gls@noidxloclist@sep{}%
6181   \def\@gls@noidxloclist@prev{}%
6182   \forlistloop{\glsnoidxloclisthandler}{#1}%
6183 }
```

`xloclisthandler` Handler for location list iterator.

```
6184 \newcommand*{\glsnoidxloclisthandler}[1]{%
6185   \ifdefstring{\@gls@noidxloclist@prev}{#1}%
6186   {%
```

Same as previous location so skip.

```
6187 }%
6188 {%
6189   \@gls@noidxloclist@sep
6190   #1%
6191   \def\@gls@noidxloclist@sep{\delimN}%
6192   \def\@gls@noidxloclist@prev{#1}%
6193 }%
6194 }
```

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

```
6195 \newcommand*{\glsnoidxdisplayloclisthandler}[1]{%
6196   \ifdefstring{\@gls@noidxloclist@prev}{#1}%
6197   {%
```

Same as previous location so skip.

```
6198 }%
6199 {%
6200   \@gls@noidxloclist@sep
6201   \@gls@noidxloclist@prev
6202   \def\@gls@noidxloclist@prev{#1}%
6203 }%
6204 }
```

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

Display a location in the location list.

```
6205 \newcommand*{\glsnoidxdisplayloc}[4]{%
6206   \setentrycounter[#1]{#2}%
6207   \csuse{#3}{#4}%
6208 }
```

`\@gls@reference`

`\@gls@reference{<type>}{<label>}{<loc>}`

Identifies that a reference has been used (for use in the aux file). All entries must be defined in the preamble.

```
6209 \newcommand*{\@gls@reference}[3]{%
```

Add to label list

```
6210 \glsdoifexistsorwarn{#2}%
6211 {%
6212 \ifcsundef{@glsref@#1}{\csgdef{@glsref@#1}{}}{}%
6213 \ifinlistcs{#2}{@glsref@#1}%
6214 }{%
6215 {\listcsgadd{@glsref@#1}{#2}}%
```

Add to location list

```
6216 \ifcsundef{glo@\glsdetoklabel{#2}@loclist}%
6217 {\csgdef{glo@\glsdetoklabel{#2}@loclist}{}}%
6218 }{%
6219 \listcsgadd{glo@\glsdetoklabel{#2}@loclist}{#3}%
6220 }%
6221 }
```

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

```
6222 \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`.

```
6223 \define@key{printgloss}{title}{%
6224 \def\glossarytitle{#1}%
6225 \let\gls@dotoc@title\relax
6226 }
```

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

```
6227 \define@key{printgloss}{toctitle}{%
6228 \def\glossarytoctitle{#1}%
6229 \let\gls@dotoc@title\relax
6230 }
```

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

```
6231 \define@key{printgloss}{style}{%
6232 \ifcsundef{@glsstyle@#1}%
6233 {%
6234 \PackageError{glossaries}%
6235 {Glossary style ‘#1’ undefined}{}%
6236 }%
6237 {%
6238 \def\@glossarystyle{\setglossentrycompatibility
6239 \csname @glsstyle@#1\endcsname}%
6240 }%
6241 }
```

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

```

6242 \define@choicekey{printgloss}{numberedsection}%
6243  [\gls@numberedsection@val\gls@numberedsection@nr]%
6244  {false,nolabel,autolabel,nameref}[nolabel]%
6245  {%
6246   \ifcase\gls@numberedsection@nr\relax
6247     \renewcommand*{\@@glossarysecstar}{*}%
6248     \renewcommand*{\@@glossaryseclabel}{}%
6249   \or
6250     \renewcommand*{\@@glossarysecstar}{}%
6251     \renewcommand*{\@@glossaryseclabel}{}%
6252   \or
6253     \renewcommand*{\@@glossarysecstar}{}%
6254     \renewcommand*{\@@glossaryseclabel}{\label{\glsautoprefix\@glo@type}}%
6255   \or
6256     \renewcommand*{\@@glossarysecstar}{*}%
6257     \renewcommand*{\@@glossaryseclabel}{%
6258       \protected@edef\@currentlabelname{\glossarytoctitle}%
6259       \label{\glsautoprefix\@glo@type}}%
6260   \fi
6261 }

```

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

```

6262 \define@choicekey{printgloss}{nogroupskip}{true,false}[true]{%
6263  \csuse{glsnogroupskip#1}%
6264 }

```

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

```

6265 \define@choicekey{printgloss}{nopostdot}{true,false}[true]{%
6266  \csuse{glsnopostdot#1}%
6267 }

```

`CounterLabelPrefix` Make it easier to redefine the label prefix.

```

6268 \newcommand*{\GlsEntryCounterLabelPrefix}{glsentry-}

```

The conditionals have been moved inside the appropriate commands to make it easier for the user to redefine them in the preamble and selectively switch the counter display on and off. Previously the helper commands were redefined by the `entrycounter` option, which would counteract any earlier customisation.

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

```

6269 \define@choicekey{printgloss}{entrycounter}{true,false}[true]{%
6270  \csuse{glsentrycounter#1}%
6271  \@gls@define@glossaryentrycounter
6272 }

```

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.


```

6273 \define@choicekey{printgloss}{subentrycounter}{true,false}[true]{%
6274   \csuse{glssubentrycounter#1}%
6275   \@gls@define@glossarysubentrycounter
6276 }

```

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

```

6277 \define@boolkey{printgloss}[gls]{nonnumberlist}[true]{%
6278   \ifglsnonnumberlist
6279     \def\glossaryentrynumbers##1{}}%
6280 \else
6281   \def\glossaryentrynumbers##1{##1}%
6282 \fi}

```

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

```

6283 \define@key{printgloss}{sort}{\@glo@assign@sortkey{#1}}

```

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

```

6284 \newcommand*\@glo@no@assign@sortkey}[1]{%
6285   \PackageError{glossaries}{'sort' key not permitted with
6286     \string\printglossary}%
6287   {The 'sort' key may only be used with \string\printnoidxglossary}%
6288 }

```

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

```

6289 \newcommand*\@glo@assign@sortkey}[1]{%
6290   \def\@glo@sorttype{#1}%
6291 }

```

`@glsnonextpages` Suppresses the next number list only. Global assignments required as it may not occur in the same level of grouping as the next numberlist. (For example, if `\glsnonextpages` is placed in the entry's description and 3 column tabular style glossary is used.) `\org@glossaryentrynumbers` needs to be set at the start of each glossary, in the event that `\glossaryentrynumber` is re-defined.

```

6292 \newcommand*\@glsnonextpages}{%
6293   \gdef\glossaryentrynumbers##1{%
6294     \glsresetentrylist
6295   }%
6296 }

```

`\@glsnextpages` Activate the next number list only. Global assignments required as it may not occur in the same level of grouping as the next numberlist. (For example, if `\glsnextpages` is placed in the entry's description and 3 column tabular style glossary is used.) `\org@glossaryentrynumbers` needs to be set at the start of each glossary, in the event that `\glossaryentrynumber` is re-defined.

```

6297 \newcommand*\@glsnextpages}{%
6298   \gdef\glossaryentrynumbers##1{%
6299     ##1\glsresetentrylist}}

```

sresetentrylist Resets \glossaryentrynumbers
6300 \newcommand*{\glsresetentrylist}{%
6301 \global\let\glossaryentrynumbers\org@glossaryentrynumbers}

\glsnonextpages Outside of \printglossary this does nothing.
6302 \newcommand*{\glsnonextpages}{}

\glsnextpages Outside of \printglossary this does nothing.
6303 \newcommand*{\glsnextpages}{}

Process entrycounter and then subentrycounter options (this ensures the sub-counter can pick up the main counter as the master if required):

6304 \@gls@define@glossaryentrycounter
6305 \@gls@define@glossarysubentrycounter

subentrycounter Resets the glossarysubentry counter.
6306 \newcommand*{\glsresetsubentrycounter}{%
6307 \ifglssubentrycounter
6308 \setcounter{glossarysubentry}{0}%
6309 \fi
6310 }

subentrycounter Resets the glossaryentry counter.
6311 \newcommand*{\glsresetentrycounter}{%
6312 \ifglseentrycounter
6313 \setcounter{glossaryentry}{0}%
6314 \fi
6315 }

\glsstepentry Advance the glossaryentry counter if in use. The argument is the label associated with the entry.
6316 \newcommand*{\glsstepentry}[1]{%
6317 \ifglseentrycounter
6318 \refstepcounter{glossaryentry}%
6319 \label{\GlsEntryCounterLabelPrefix\glsdetoklabel{#1}}%
6320 \fi
6321 }

glsstepsubentry Advance the glossarysubentry counter if in use. The argument is the label associated with the subentry.
6322 \newcommand*{\glsstepsubentry}[1]{%
6323 \ifglssubentrycounter
6324 \edef\currentglssubentry{\glsdetoklabel{#1}}%
6325 \refstepcounter{glossarysubentry}%
6326 \label{\GlsEntryCounterLabelPrefix\currentglssubentry}%
6327 \fi
6328 }

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

```
6329 \newcommand*{\glsrefentry}[1]{%
6330   \ifglentrycounter
6331     \ref{\GlsEntryCounterLabelPrefix\glsdetoklabel{#1}}%
6332   \else
6333     \ifglssubentrycounter
6334       \ref{\GlsEntryCounterLabelPrefix\glsdetoklabel{#1}}%
6335     \else
6336       \gls{#1}%
6337     \fi
6338   \fi
6339 }
```

`entrycounterlabel` Defines how to display the glossaryentry counter.

```
6340 \newcommand*{\glentrycounterlabel}{%
6341   \ifglentrycounter
6342     \theglossaryentry.\space
6343   \fi
6344 }
```

`entrycounterlabel` Defines how to display the glossarysubentry counter.

```
6345 \newcommand*{\glssubentrycounterlabel}{%
6346   \ifglssubentrycounter
6347     \theglossarysubentry)\space
6348   \fi
6349 }
```

`\glentryitem` Step and display glossaryentry counter, if appropriate.

```
6350 \newcommand*{\glentryitem}[1]{%
6351   \ifglentrycounter
6352     \glsstepentry{#1}\glentrycounterlabel
6353   \else
6354     \glsresetsubentrycounter
6355   \fi
6356 }
```

`glssubentryitem` Step and display glossarysubentry counter, if appropriate.

```
6357 \newcommand*{\glssubentryitem}[1]{%
6358   \ifglssubentrycounter
6359     \glsstepsubentry{#1}\glssubentrycounterlabel
6360   \fi
6361 }
```

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

```
6362 \ifcsundef{theglossary}%
6363 {%
6364   \newenvironment{theglossary}{-}{-}%
6365 }
```

```

6365 }%
6366 {%
6367 \@gls@warnontheglossdefined
6368 \renewenvironment{theglossary}{-}{-}%
6369 }

```

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`

```
6370 \newcommand*{\glossaryheader}{}
```

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

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

```
6371 \newcommand*{\glstarget}[2]{\@glstarget{\glo@linkprefix#1}{#2}}
```

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

`\compatibleglossentry`

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

```

6372 \providecommand*{\compatibleglossentry}[2]{%
6373 \toks@{#2}%
6374 \protected@edef\@do@glossentry{\noexpand\glossaryentryfield{#1}%
6375 \noexpand\glsnamefont
6376 \expandafter\expandonce\csname glo@#1@name\endcsname}}%
6377 \expandafter\expandonce\csname glo@#1@desc\endcsname}%
6378 \expandafter\expandonce\csname glo@#1@symbol\endcsname}%
6379 \the\toks@}%
6380 }%
6381 \@do@glossentry
6382 }

```

`\glossentryname`

```

6383 \newcommand*{\glossentryname}[1]{%
6384 \glsdoifexistsorwarn{#1}%
6385 {%
6386 \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%

```

```

6387 \expandafter\glsnamefont\expandafter{\glo@name}%
6388 }%
6389 }

```

\Glossentryname

```

6390 \newcommand*{\Glossentryname}[1]{%
6391 \glsdoifexistsorwarn{#1}%
6392 {%
6393 \glsnamefont{\Glsentryname{#1}}%
6394 }%
6395 }

```

\glossentrydesc

```

6396 \newcommand*{\glossentrydesc}[1]{%
6397 \glsdoifexistsorwarn{#1}%
6398 {%
6399 \glsentrydesc{#1}%
6400 }%
6401 }

```

\Glossentrydesc

```

6402 \newcommand*{\Glossentrydesc}[1]{%
6403 \glsdoifexistsorwarn{#1}%
6404 {%
6405 \Glsentrydesc{#1}%
6406 }%
6407 }

```

lossentrysymbol

```

6408 \newcommand*{\glossentrysymbol}[1]{%
6409 \glsdoifexistsorwarn{#1}%
6410 {%
6411 \glsentrysymbol{#1}%
6412 }%
6413 }

```

lossentrysymbol

```

6414 \newcommand*{\Glossentrysymbol}[1]{%
6415 \glsdoifexistsorwarn{#1}%
6416 {%
6417 \Glsentrysymbol{#1}%
6418 }%
6419 }

```

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

```

6420 \providecommand*{\compatiblesubglossentry}[3]{%

```

```

6421 \toks@{#3}%
6422 \protected@edef\@do@subglossentry{\noexpand\glossarysubentryfield{\number#1}%
6423 {#2}%
6424   {\noexpand\glsnamefont
6425     {\expandafter\expandonce\csname glo@#2@name\endcsname}}}%
6426   {\expandafter\expandonce\csname glo@#2@desc\endcsname}%
6427   {\expandafter\expandonce\csname glo@#2@symbol\endcsname}%
6428   {\the\toks@}%
6429 }%
6430 \@do@subglossentry
6431 }

```

rycompatibility

```

6432 \newcommand*\setglossentrycompatibility{%
6433   \let\glossentry\compatibleglossentry
6434   \let\subglossentry\compatiblesubglossentry
6435 }
6436 \setglossentrycompatibility

```

glossaryentryfield

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

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

```

6437 \newcommand{\glossaryentryfield}[5]{%
6438   \GlossariesWarning
6439   {Deprecated use of \string\glossaryentryfield.^^J
6440     I recommend you change to \string\glossentry.^^J
6441     If you've just upgraded, try removing your gls auxiliary
6442     files^^J and recompile}%
6443   \noindent\textbf{\glstarget{#1}{#2}} #4 #3. #5\par}

```

glossarysubentryfield

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

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

```

6444 \newcommand*\glossarysubentryfield}[6]{%
6445   \GlossariesWarning
6446   {Deprecated use of \string\glossarysubentryfield.^^J
6447     I recommend you change to \string\subglossentry.^^J
6448     If you've just upgraded, try removing your gls auxiliary
6449     files^^J and recompile}%
6450   \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`

```
6451 \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`

```
6452 \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.

`\glsgetgrouptitle`

```
6453 \newcommand*{\glsgetgrouptitle}[1]{%
6454   \@gls@getgrouptitle{#1}{\@gls@grptitle}%
6455   \@gls@grptitle
6456 }
```

`\@gls@getgrouptitle`

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

```
6457 \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.

```

6458 \dtl@ifsingle{#1}%
6459 {%
6460 \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}}%
6461 }%
6462 {%
6463 \ifboolexpr{test{\ifstrequal{#1}{glssymbols}}
6464             or test{\ifstrequal{#1}{glsnumbers}}}%
6465 {%
6466 \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}}%
6467 }%
6468 {%
6469 \def#2{#1}%
6470 }%
6471 }%
6472 }

```

`x@getgrouptitle` Version for the no-indexing app option:

```

6473 \newcommand*{\@gls@noidx@getgrouptitle}[2]{%
6474 \DTLifint{#1}%
6475 {\edef#2{\char#1\relax}}%
6476 {%
6477 \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}}%
6478 }%
6479 }

```

`\glsgetgrouplabel{<title>}`

This command does the reverse to the previous command. The argument is the group title, and it produces the group label. Note that if you redefine `\glsgetgrouptitle`, you will also need to redefine `\glsgetgrouplabel`.

`lsgetgrouplabel`

```

6480 \newcommand*{\glsgetgrouplabel}[1]{%
6481 \ifthenelse{\equal{#1}{\glssymbolsgroupname}}{\glssymbols}{%
6482 \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`

```

6483 \newcommand*{\setentrycounter}[2][ ]{%
6484 \def\@glo@counterprefix{#1}%
6485 \ifx\@glo@counterprefix\empty
6486 \def\@glo@counterprefix{.}%
6487 \else

```



```

6488   \def\@glo@counterprefix{.#1.}%
6489   \fi
6490   \def\glsentrycounter{#2}%
6491 }

```

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

`etglossarystyle`

```

6492 \newcommand*\setglossarystyle}[1]{%
6493   \ifcsundef{@glsstyle@#1}%
6494   {%
6495     \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
6496   }%
6497   {%
6498     \csname @glsstyle@#1\endcsname
6499   }%

```

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

```

6500 \ifx\@glossary@default@style\relax
6501   \protected@edef\@glossary@default@style{#1}%
6502 \fi
6503 }

```

`\glossarystyle`

```

6504 \newcommand*\glossarystyle}[1]{%
6505   \ifcsundef{@glsstyle@#1}%
6506   {%
6507     \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
6508   }%
6509   {%
6510     \GlossariesWarning
6511     {Deprecated command \string\glossarystyle.^~J
6512     I recommend you switch to \string\setglossarystyle\space unless
6513     you want to maintain backward compatibility}%
6514     \setglossentrycompatibility
6515     \csname @glsstyle@#1\endcsname

6516     \ifcsdef{@glscompstyle@#1}%
6517     {\setglossentrycompatibility\csuse{@glscompstyle@#1}}%
6518     }%
6519   }%

```

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

```

6520 \ifx\@glossary@default@style\relax
6521   \protected@edef\@glossary@default@style{#1}%
6522 \fi
6523 }

```

`ewglossarystyle` 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.

```
6524 \newcommand{\newglossarystyle}[2]{%
6525   \ifcsundef{@glsstyle@#1}%
6526   {%
6527     \expandafter\def\csname @glsstyle@#1\endcsname{#2}%
6528   }%
6529   {%
6530     \PackageError{glossaries}{Glossary style ‘#1’ is already defined}{}%
6531   }%
6532 }
```

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

```
6533 \newcommand{\renewglossarystyle}[2]{%
6534   \ifcsundef{@glsstyle@#1}%
6535   {%
6536     \PackageError{glossaries}{Glossary style ‘#1’ isn’t already defined}{}%
6537   }%
6538   {%
6539     \csdef{@glsstyle@#1}{#2}%
6540   }%
6541 }
```

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`

```
6542 \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 `\glslinknumber`. 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`

```
6543 \ifcsundef{hyperlink}%
6544 {%
6545   \def\glshypernumber#1{#1}%
6546 }%
6547 {%
6548   \def\glshypernumber#1{\@glshypernumber#1\nohyperpage{}}\@nil}
6549 }
```

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

```
6550 \def\@glshypernumber#1\nohyperpage#2#3\@nil{%
6551   \ifx\#1\%
6552   \else
6553     \@delimR#1\delimR\delimR\%
6554   \fi
6555   \ifx\#2\%
6556   \else
6557     #2%
6558   \fi
6559   \ifx\#3\%
6560   \else
6561     \@glshypernumber#3\@nil
6562   \fi
6563 }
```

`\@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`

```
6564 \def\@delimR#1\delimR #2\delimR #3\%
6565 \ifx\#2\%
6566   \@delimN{#1}%
6567 \else
6568   \@gls@numberlink{#1}\delimR\@gls@numberlink{#2}%
6569 \fi}
```

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

`\@delimN`

```
6570 \def\@delimN#1{\@delimN#1\delimN \delimN\%
6571 \def\@delimN#1\delimN #2\delimN#3\%
6572 \ifx\#3\%
6573   \@gls@numberlink{#1}%
6574 \else
6575   \@gls@numberlink{#1}\delimN\@gls@numberlink{#2}%
6576 \fi
6577 }
```

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

```

6578 \def\@gls@numberlink#1{%
6579 \begingroup
6580 \toks@={}%
6581 \@gls@removespaces#1 \@nil
6582 \endgroup}

6583 \def\@gls@removespaces#1 #2\@nil{%
6584 \toks@=\expandafter{\the\toks@#1}%
6585 \ifx\#2\%
6586 \edef\x{\the\toks@}%
6587 \ifx\x\empty
6588 \else

6589 \hyperlink{\glsentrycounter\@glo@counterprefix\the\toks@}%
6590 {\the\toks@}%
6591 \fi
6592 \else
6593 \@gls@ReturnAfterFi{%
6594 \@gls@removespaces#2\@nil
6595 }%
6596 \fi
6597 }
6598 \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

```
6599 \newcommand*\hyperrm[1]{\textrm{\glsnumber{#1}}}
```

\hypersf

```
6600 \newcommand*\hypersf[1]{\textsf{\glsnumber{#1}}}
```

\hypertt

```
6601 \newcommand*\hypertt[1]{\texttt{\glsnumber{#1}}}
```

\hyperbf

```
6602 \newcommand*\hyperbf[1]{\textbf{\glsnumber{#1}}}
```

\hypermd

```
6603 \newcommand*\hypermd[1]{\textmd{\glsnumber{#1}}}
```

\hyperit

```
6604 \newcommand*\hyperit[1]{\textit{\glsnumber{#1}}}
```

\hypersl

```
6605 \newcommand*\hypersl[1]{\textsl{\glsnumber{#1}}}
```

`\hyperup`

```
6606 \newcommand*{\hyperup}[1]{\textup{\glshypernumber{#1}}}
```

`\hypersc`

```
6607 \newcommand*{\hypersc}[1]{\textsc{\glshypernumber{#1}}}
```

`\hyperemph`

```
6608 \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.

```
6609 \newcommand{\oldacronym}[4] [\gls@label]{%
6610   \def\gls@label{#2}%
6611   \newacronym[#4]{#1}{#2}{#3}%
6612   \ifcsundef{xspace}%
6613   {%
6614     \expandafter\edef\csname#1\endcsname{%
6615       \noexpand\@ifstar{\noexpand\Gls{#1}}{\noexpand\gls{#1}}%
6616     }%
6617   }%
6618   {%
6619     \expandafter\edef\csname#1\endcsname{%
6620       \noexpand\@ifstar{\noexpand\Gls{#1}\noexpand\xspace}{%
6621         \noexpand\gls{#1}\noexpand\xspace}%
6622     }%
6623   }%
6624 }
```

```
\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`

```
6625 \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 needed to cancel it out.

```
6626 \newcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}
```

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

`\glstextup`

```
6627 \newrobustcmd*{\glstextup}[1]{\ifdef\textulc{\textulc{#1}}{\textup{#1}}}
```

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

`\glsshortkey`

```
6628 \newcommand*{\glsshortkey}{short}
```

`\glsshortpluralkey`

```
6629 \newcommand*{\glsshortpluralkey}{shortplural}
```

`\glslongkey`

```
6630 \newcommand*{\glslongkey}{long}
```

`\glslongpluralkey`

```
6631 \newcommand*{\glslongpluralkey}{longplural}
```

`\acrfull` Full form of the acronym.

```
6632 \newrobustcmd*{\acrfull}{\@gls@hyp@opt\ns@acrfull}
```

```
6633 \newcommand*\ns@acrfull[2] [] {%
```

```
6634 \new@ifnextchar[{\@acrfull{#1}{#2}}%
```

```
6635 {\@acrfull{#1}{#2} []}%
```

```
6636 }
```

`\@acrfull` Low-level macro:

```
6637 \def\@acrfull#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6638 \acrfullfmt{#1}{#2}{#3}%  
6639 }
```

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

`\acrfullfmt` No case change full format.

```
6640 \newcommand*\acrfullfmt}[3]{%  
6641 \acrlinkfullformat{\@acrlong}{\@acrshort}{#1}{#2}{#3}%  
6642 }
```

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

```
6643 \newcommand{\acrlinkfullformat}[5]{%  
6644 \acrfullformat{#1{#3}{#4}[#5]}{#2{#3}{#4}[]}%  
6645 }
```

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

```
6646 \newcommand{\acrfullformat}[2]{#1\glsspace(#2)}
```

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

```
6647 \newrobustcmd{\glsspace}{\space}
```

Default format for full acronym

`\Acrfull`

```
6648 \newrobustcmd*\Acrfull{\@gls@hyp@opt\ns@Acrfull}
```

```
6649 \newcommand*\ns@Acrfull[2][]{%  
6650 \new@ifnextchar[{\@Acrfull{#1}{#2}}%  
6651 {\@Acrfull{#1}{#2}[]}%  
6652 }
```

Low-level macro:

```
6653 \def\@Acrfull#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6654 \Acrfullfmt{#1}{#2}{#3}%  
6655 }
```

`\Acrfullfmt` First letter upper case full format.

```
6656 \newcommand*\Acrfullfmt}[3]{%  
6657 \acrlinkfullformat{\@Acrlong}{\@acrshort}{#1}{#2}{#3}%  
6658 }
```

`\ACRfull`

```
6659 \newrobustcmd*{\ACRfull}{\@gls@hyp@opt\ns@ACRfull}
```

```
6660 \newcommand*\ns@ACRfull[2] [] {%
6661   \new@ifnextchar[{\@ACRfull{#1}{#2}}%
6662     {\@ACRfull{#1}{#2} []}%
6663 }
```

Low-level macro:

```
6664 \def\@ACRfull#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6665   \ACRfullfmt{#1}{#2}{#3}%
6666 }
```

`\ACRfullfmt` All upper case full format.

```
6667 \newcommand*{\ACRfullfmt}[3]{%
6668   \acrlinkfullformat{\@ACRlong}{\@ACRshort}{#1}{#2}{#3}%
6669 }
```

Plural:

`\acrfullpl`

```
6670 \newrobustcmd*{\acrfullpl}{\@gls@hyp@opt\ns@acrfullpl}
```

```
6671 \newcommand*\ns@acrfullpl[2] [] {%
6672   \new@ifnextchar[{\@acrfullpl{#1}{#2}}%
6673     {\@acrfullpl{#1}{#2} []}%
6674 }
```

Low-level macro:

```
6675 \def\@acrfullpl#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6676   \acrfullplfmt{#1}{#2}{#3}%
6677 }
```

`\acrfullplfmt` No case change plural full format.

```
6678 \newcommand*{\acrfullplfmt}[3]{%
6679   \acrlinkfullformat{\@acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%
6680 }
```

`\Acrfullpl`

```
6681 \newrobustcmd*{\Acrfullpl}{\@gls@hyp@opt\ns@Acrfullpl}
```

```
6682 \newcommand*\ns@Acrfullpl[2] [] {%
6683   \new@ifnextchar[{\@Acrfullpl{#1}{#2}}%
6684     {\@Acrfullpl{#1}{#2} []}%
6685 }
```


Low-level macro:

```
6686 \def\@Acrfullpl#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6687 \Acrfullplfmt{#1}{#2}{#3}%
6688 }
```

`\Acrfullplfmt` First letter upper case plural full format.

```
6689 \newcommand*\Acrfullplfmt}[3]{%
6690 \acrlinkfullformat{\@Acrlongpl}\@acrshortpl}{#1}{#2}{#3}%
6691 }
```

`\ACRfullpl`

```
6692 \newrobustcmd*\ACRfullpl{\@gls@hyp@opt\ns@ACRfullpl}

6693 \newcommand*\ns@ACRfullpl[2][ ]{%
6694 \new@ifnextchar[{\@ACRfullpl{#1}{#2}}%
6695 {\@ACRfullpl{#1}{#2}[ ]}%
6696 }
```

Low-level macro:

```
6697 \def\@ACRfullpl#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6698 \ACRfullplfmt{#1}{#2}{#3}%
6699 }
```

`\ACRfullplfmt` All upper case plural full format.

```
6700 \newcommand*\ACRfullplfmt}[3]{%
6701 \acrlinkfullformat{\@ACRlongpl}\@ACRshortpl}{#1}{#2}{#3}%
6702 }
```

1.18 Predefined acronym styles

`\acronymfont` This is only used with the additional acronym styles:

```
6703 \newcommand{\acronymfont}[1]{#1}
```

`\firstacronymfont` This is only used with the additional acronym styles:

```
6704 \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.

```
6705 \newcommand*\acrnameformat}[2]{\acronymfont{#1}}
```

Define some tokens used by `\newacronym`:

`\glskeylisttok`

```
6706 \newtoks\glskeylisttok
```

```

\glslabeltok
6707 \newtoks\glslabeltok

\glsshorttok
6708 \newtoks\glsshorttok

\glslongtok
6709 \newtoks\glslongtok

\newacronymhook  Provide a hook for \newacronym:
6710 \newcommand*{\newacronymhook}{}

genericNewAcronym  New improved version of setting the acronym style.
6711 \newcommand*{\SetGenericNewAcronym}{%
    Change the behaviour of \Glsentryname to workaround expansion issues that cause a problem for \makefirstuc
6712 \let\@Gls@entryname\@Gls@acentryname
    Change the way acronyms are defined:
6713 \renewcommand{\newacronym}[4][\%
6714 \ifdefempty{\@glsacronymlists}%
6715 {%
6716 \def\@glo@type{\acronymtype}%
6717 \setkeys{glossentry}{##1}%
6718 \DeclareAcronymList{\@glo@type}%
6719 }%
6720 }%
6721 \glskeylisttok{##1}%
6722 \glslabeltok{##2}%
6723 \glsshorttok{##3}%
6724 \glslongtok{##4}%
6725 \newacronymhook
6726 \protected@edef\@do@newglossaryentry{%
6727 \noexpand\newglossaryentry{\the\glslabeltok}%
6728 {%
6729 type=\acronymtype,%
6730 name={\expandonce{\acronymentry{##2}}},%
6731 sort={\acronymssort{\the\glsshorttok}{\the\glslongtok}},%
6732 text={\the\glsshorttok},%
6733 short={\the\glsshorttok},%
6734 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6735 long={\the\glslongtok},%
6736 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6737 \GenericAcronymFields,%
6738 \the\glskeylisttok
6739 }%
6740 }%
6741 \@do@newglossaryentry
6742 }%

```

Make sure that `\acrfull` etc reflects the new style:

```
6743 \renewcommand*{\acrfullfmt}[3]{%
6744   \glslink[##1]{##2}{\genacrfullformat{##2}{##3}}}%
6745 \renewcommand*{\Acrfullfmt}[3]{%
6746   \glslink[##1]{##2}{\Genacrfullformat{##2}{##3}}}%
6747 \renewcommand*{\ACRfullfmt}[3]{%
6748   \glslink[##1]{##2}{%
6749     \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}}}%
6750 \renewcommand*{\acrfullplfmt}[3]{%
6751   \glslink[##1]{##2}{\genplacrfullformat{##2}{##3}}}%
6752 \renewcommand*{\Acrfullplfmt}[3]{%
6753   \glslink[##1]{##2}{\Genplacrfullformat{##2}{##3}}}%
6754 \renewcommand*{\ACRfullplfmt}[3]{%
6755   \glslink[##1]{##2}{%
6756     \mfirstucMakeUppercase{\genplacrfullformat{##2}{##3}}}}}%

```

Make sure that `\glstentryfull` etc reflects the new style:

```
6757 \renewcommand*{\glstentryfull}[1]{\genacrfullformat{##1}{}}%
6758 \renewcommand*{\Glstentryfull}[1]{\Genacrfullformat{##1}{}}%
6759 \renewcommand*{\glstentryfullpl}[1]{\genplacrfullformat{##1}{}}%
6760 \renewcommand*{\Glstentryfullpl}[1]{\Genplacrfullformat{##1}{}}%
6761 }

```

`\GenericAcronymFields` Fields used by `\SetGenericNewAcronym` that can be changed by the acronym style.

```
6762 \newcommand*{\GenericAcronymFields}{description={\the\glslongtok}}
```

`\acronymentry` `\acronymentry{<label>}`

Display style for the name field in the list of acronyms.

```
6763 \newcommand*{\acronymentry}[1]{\acronymfont{\glstentryshort{##1}}}
```

`\acronymsort` `\acronymsort{<short>}{<long>}`

Default sort format for acronyms.

```
6764 \newcommand*{\acronymsort}[2]{##1}
```

`\setacronymstyle` `\setacronymstyle{<style name>}`

```
6765 \newcommand*{\setacronymstyle}[1]{%
6766   \ifcsundef{@glsacr@dispstyle@##1}
6767   {%
6768     \PackageError{glossaries}{Undefined acronym style ‘##1’}{%
6769   }}%
6770   {%

```

```

6771 \ifdefempty{\@glsacronymlists}%
6772 {%
6773   \DeclareAcronymList{\acronymtype}%
6774   }%
6775   {}%
6776   \SetGenericNewAcronym
6777   \GlsUseAcrStyleDefs{#1}%
6778   \@for\@gls@type:=\@glsacronymlists\do{%
6779     \defglsentryfmt[\@gls@type]{\GlsUseAcrEntryDispStyle{#1}}%
6780   }%
6781 }%
6782 }

```

`\newacronymstyle` \newacronymstyle{<style name>}{<entry format definition>}{<display definitions>}

Defines a new acronym style called *<style name>*.

```

6783 \newcommand*{\newacronymstyle}[3]{%
6784   \ifcsdef{@glsacr@dispstyle@#1}%
6785   {%
6786     \PackageError{glossaries}{Acronym style ‘#1’ already exists}{}%
6787   }%
6788   {%
6789     \csdef{@glsacr@dispstyle@#1}{#2}%
6790     \csdef{@glsacr@styledefs@#1}{#3}%
6791   }%
6792 }

```

`\renewacronymstyle` Redefines the given acronym style.

```

6793 \newcommand*{\renewacronymstyle}[3]{%
6794   \ifcsdef{@glsacr@dispstyle@#1}%
6795   {%
6796     \csdef{@glsacr@dispstyle@#1}{#2}%
6797     \csdef{@glsacr@styledefs@#1}{#3}%
6798   }%
6799   {%
6800     \PackageError{glossaries}{Acronym style ‘#1’ doesn’t exist}{}%
6801   }%
6802 }

```

`\rEntryDispStyle`

```

6803 \newcommand*{\GlsUseAcrEntryDispStyle}[1]{\csuse{@glsacr@dispstyle@#1}}

```

`\UseAcrStyleDefs`

```

6804 \newcommand*{\GlsUseAcrStyleDefs}[1]{\csuse{@glsacr@styledefs@#1}}

```

Predefined acronym styles:

long-short *<long>* (*<short>*) acronym style.

```
6805 \newacronymstyle{long-short}%
6806 {%
```

Check for long form in case this is a mixed glossary.

```
6807 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6808 }%
6809 {%
6810 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6811 \renewcommand*{\genacrfullformat}[2]{%
6812 \glsentrylong{##1}##2\space
6813 (\protect\firstacronymfont{\glsentryshort{##1}})%
6814 }%
6815 \renewcommand*{\Genacrfullformat}[2]{%
6816 \Glsentrylong{##1}##2\space
6817 (\protect\firstacronymfont{\glsentryshort{##1}})%
6818 }%
6819 \renewcommand*{\genplacrfullformat}[2]{%
6820 \glsentrylongpl{##1}##2\space
6821 (\protect\firstacronymfont{\glsentryshortpl{##1}})%
6822 }%
6823 \renewcommand*{\Genplacrfullformat}[2]{%
6824 \Glsentrylongpl{##1}##2\space
6825 (\protect\firstacronymfont{\glsentryshortpl{##1}})%
6826 }%
6827 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6828 \renewcommand*{\acronymsort}[2]{##1}%
6829 \renewcommand*{\acronymfont}[1]{##1}%
6830 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
6831 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6832 }
```

long-sp-short Similar to the previous style but allows the space between the long and short form to be customized.

```
6833 \newacronymstyle{long-sp-short}%
6834 {%
```

Check for long form in case this is a mixed glossary.

```
6835 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6836 }%
6837 {%
6838 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6839 \renewcommand*{\genacrfullformat}[2]{%
6840 \glsentrylong{##1}##2\glsacspace{##1}%
6841 (\protect\firstacronymfont{\glsentryshort{##1}})%
6842 }%
6843 \renewcommand*{\Genacrfullformat}[2]{%
6844 \Glsentrylong{##1}##2\glsacspace{##1}%
6845 (\protect\firstacronymfont{\glsentryshort{##1}})%
6846 }%
```

```

6847 \renewcommand*\genplacrfullformat}[2]{%
6848 \glsentrylongpl{##1}##2\glsacspace{##1}%
6849 (\protect\firstacronymfont{\glsentryshortpl{##1}})%
6850 }%
6851 \renewcommand*\Genplacrfullformat}[2]{%
6852 \Glsentrylongpl{##1}##2\glsacspace{##1}%
6853 (\protect\firstacronymfont{\glsentryshortpl{##1}})%
6854 }%
6855 \renewcommand*\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6856 \renewcommand*\acronymsort}[2]{##1}%
6857 \renewcommand*\acronymfont}[1]{##1}%
6858 \renewcommand*\firstacronymfont}[1]{\acronymfont{##1}}%
6859 \renewcommand*\acrpluralsuffix{\glspluralsuffix}%
6860 }

```

`\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.

```

6861 \newcommand*\glsacspace}[1]{%
6862 \settowidth{\dimen@}{(\firstacronymfont{\glsentryshort{##1}})}%
6863 \ifdim\dimen@<3em~\else\space\fi
6864 }

```

`short-long` (*short*) (*long*) acronym style.

```

6865 \newacronymstyle{short-long}%
6866 {%

```

Check for long form in case this is a mixed glossary.

```

6867 \ifglshaslong{\glslabel}{\glsngenacfmt}{\glsngenentryfmt}%
6868 }%
6869 {%
6870 \renewcommand*\GenericAcronymFields{description={\the\glslongtok}}%
6871 \renewcommand*\genacrfullformat}[2]{%
6872 \protect\firstacronymfont{\glsentryshort{##1}}##2\space
6873 (\glsentrylong{##1})%
6874 }%
6875 \renewcommand*\Genacrfullformat}[2]{%
6876 \protect\firstacronymfont{\Glsentryshort{##1}}##2\space
6877 (\glsentrylong{##1})%
6878 }%
6879 \renewcommand*\genplacrfullformat}[2]{%
6880 \protect\firstacronymfont{\glsentryshortpl{##1}}##2\space
6881 (\glsentrylongpl{##1})%
6882 }%
6883 \renewcommand*\Genplacrfullformat}[2]{%
6884 \protect\firstacronymfont{\Glsentryshortpl{##1}}##2\space
6885 (\glsentrylongpl{##1})%
6886 }%

6887 \renewcommand*\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6888 \renewcommand*\acronymsort}[2]{##1}%

```

```

6889 \renewcommand*{\acronymfont}[1]{##1}%
6890 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
6891 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6892 }

```

long-sc-short *(long)* (\textsc{*(short)*}) acronym style.

```

6893 \newacronymstyle{long-sc-short}%
6894 {%
6895 \GlsUseAcrEntryDispStyle{long-short}%
6896 }%
6897 {%
6898 \GlsUseAcrStyleDefs{long-short}%
6899 \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6900 \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6901 }

```

long-sm-short *(long)* (\textsmaller{*(short)*}) acronym style.

```

6902 \newacronymstyle{long-sm-short}%
6903 {%
6904 \GlsUseAcrEntryDispStyle{long-short}%
6905 }%
6906 {%
6907 \GlsUseAcrStyleDefs{long-short}%
6908 \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6909 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6910 }

```

sc-short-long *(short)* (\textsc{*(long)*}) acronym style.

```

6911 \newacronymstyle{sc-short-long}%
6912 {%
6913 \GlsUseAcrEntryDispStyle{short-long}%
6914 }%
6915 {%
6916 \GlsUseAcrStyleDefs{short-long}%
6917 \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6918 \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6919 }

```

sm-short-long *(short)* (\textsmaller{*(long)*}) acronym style.

```

6920 \newacronymstyle{sm-short-long}%
6921 {%
6922 \GlsUseAcrEntryDispStyle{short-long}%
6923 }%
6924 {%
6925 \GlsUseAcrStyleDefs{short-long}%
6926 \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6927 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6928 }

```

long-short-desc *<long>* (*{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```
6929 \newacronymstyle{long-short-desc}%
6930 {%
6931   \GlsUseAcrEntryDispStyle{long-short}%
6932 }%
6933 {%
6934   \GlsUseAcrStyleDefs{long-short}%
6935   \renewcommand*{\GenericAcronymFields}{}%
6936   \renewcommand*{\acronymsort}[2]{##2}%
6937   \renewcommand*{\acronymentry}[1]{%
6938     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6939 }
```

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`.

```
6940 \newacronymstyle{long-sp-short-desc}%
6941 {%
6942   \GlsUseAcrEntryDispStyle{long-sp-short}%
6943 }%
6944 {%
6945   \GlsUseAcrStyleDefs{long-sp-short}%
6946   \renewcommand*{\GenericAcronymFields}{}%
6947   \renewcommand*{\acronymsort}[2]{##2}%
6948   \renewcommand*{\acronymentry}[1]{%
6949     \glentrylong{##1}\glsacspace{##1}(\acronymfont{\glentryshort{##1}})}%
6950 }
```

g-sc-short-desc *<long>* (`\textsc{<short>}`) acronym style that has an accompanying description (which the user needs to supply).

```
6951 \newacronymstyle{long-sc-short-desc}%
6952 {%
6953   \GlsUseAcrEntryDispStyle{long-sc-short}%
6954 }%
6955 {%
6956   \GlsUseAcrStyleDefs{long-sc-short}%
6957   \renewcommand*{\GenericAcronymFields}{}%
6958   \renewcommand*{\acronymsort}[2]{##2}%
6959   \renewcommand*{\acronymentry}[1]{%
6960     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6961 }
```

g-sm-short-desc *<long>* (`\textsmaller{<short>}`) acronym style that has an accompanying description (which the user needs to supply).

```
6962 \newacronymstyle{long-sm-short-desc}%
6963 {%
6964   \GlsUseAcrEntryDispStyle{long-sm-short}%
6965 }%
```



```

6966 {%
6967 \GlsUseAcrStyleDefs{long-sm-short}%
6968 \renewcommand*\GenericAcronymFields{}%
6969 \renewcommand*\acronymsort}[2]{##2}%
6970 \renewcommand*\acronymentry}[1]{%
6971 \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6972 }

```

short-long-desc *<short>* (*<long>*) acronym style that has an accompanying description (which the user needs to supply).

```

6973 \newacronymstyle{short-long-desc}%
6974 {%
6975 \GlsUseAcrEntryDispStyle{short-long}%
6976 }%
6977 {%
6978 \GlsUseAcrStyleDefs{short-long}%
6979 \renewcommand*\GenericAcronymFields{}%
6980 \renewcommand*\acronymsort}[2]{##2}%
6981 \renewcommand*\acronymentry}[1]{%
6982 \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6983 }

```

short-long-desc *<long>* (*\textsc{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

6984 \newacronymstyle{sc-short-long-desc}%
6985 {%
6986 \GlsUseAcrEntryDispStyle{sc-short-long}%
6987 }%
6988 {%
6989 \GlsUseAcrStyleDefs{sc-short-long}%
6990 \renewcommand*\GenericAcronymFields{}%
6991 \renewcommand*\acronymsort}[2]{##2}%
6992 \renewcommand*\acronymentry}[1]{%
6993 \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6994 }

```

short-long-desc *<long>* (*\textsmaller{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

6995 \newacronymstyle{sm-short-long-desc}%
6996 {%
6997 \GlsUseAcrEntryDispStyle{sm-short-long}%
6998 }%
6999 {%
7000 \GlsUseAcrStyleDefs{sm-short-long}%
7001 \renewcommand*\GenericAcronymFields{}%
7002 \renewcommand*\acronymsort}[2]{##2}%
7003 \renewcommand*\acronymentry}[1]{%
7004 \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
7005 }

```

dua *<long>* only acronym style.

```
7006 \newacronymstyle{dua}%  
7007 {%
```

Check for long form in case this is a mixed glossary.

```
7008 \ifdefempty\glscustomtext  
7009 {%  
7010 \ifglshaslong{\glslabel}%  
7011 {%  
7012 \glsifplural  
7013 {%
```

Plural form:

```
7014 \glscapscase  
7015 {%
```

Plural form, don't adjust case:

```
7016 \glstentrylongpl{\glslabel}\glsinsert  
7017 }%  
7018 {%
```

Plural form, make first letter upper case:

```
7019 \Glstentrylongpl{\glslabel}\glsinsert  
7020 }%  
7021 {%
```

Plural form, all caps:

```
7022 \mfirstucMakeUppercase  
7023 {\glstentrylongpl{\glslabel}\glsinsert}%  
7024 }%  
7025 }%  
7026 {%
```

Singular form

```
7027 \glscapscase  
7028 {%
```

Singular form, don't adjust case:

```
7029 \glstentrylong{\glslabel}\glsinsert  
7030 }%  
7031 {%
```

Subsequent singular form, make first letter upper case:

```
7032 \Glstentrylong{\glslabel}\glsinsert  
7033 }%  
7034 {%
```

Subsequent singular form, all caps:

```
7035 \mfirstucMakeUppercase  
7036 {\glstentrylong{\glslabel}\glsinsert}%  
7037 }%  
7038 }%  
7039 }%  
7040 {%
```

Not an acronym:

```
7041     \glsgenentryfmt
7042     }%
7043 }%
7044 {\glscustomtext\glsinsert}%
7045 }%
7046 {%
7047 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%

7048 \renewcommand*{\acrfullfmt}[3]{%
7049   \glslink[##1]{##2}{\glsentrylong{##2}##3\space
7050   (\acronymfont{\glsentryshort{##2}})}}%
7051 \renewcommand*{\Acrfullfmt}[3]{%
7052   \glslink[##1]{##2}{\Glsentrylong{##2}##3\space
7053   (\acronymfont{\glsentryshort{##2}})}}%
7054 \renewcommand*{\ACRfullfmt}[3]{%
7055   \glslink[##1]{##2}{%
7056   \mfirstucMakeUppercase{\glsentrylong{##2}##3\space
7057   (\acronymfont{\glsentryshort{##2}})}}}%

7058 \renewcommand*{\acrfullplfmt}[3]{%
7059   \glslink[##1]{##2}{\glsentrylongpl{##2}##3\space
7060   (\acronymfont{\glsentryshortpl{##2}})}}%

7061 \renewcommand*{\Acrfullplfmt}[3]{%
7062   \glslink[##1]{##2}{\Glsentrylongpl{##2}##3\space
7063   (\acronymfont{\glsentryshortpl{##2}})}}%
7064 \renewcommand*{\ACRfullplfmt}[3]{%
7065   \glslink[##1]{##2}{%
7066   \mfirstucMakeUppercase{\glsentrylongpl{##2}##3\space
7067   (\acronymfont{\glsentryshortpl{##2}})}}}%
7068 \renewcommand*{\glsentryfull}[1]{%
7069   \glsentrylong{##1}\space(\acronymfont{\glsentryshort{##1}})%
7070 }%
7071 \renewcommand*{\Glsentryfull}[1]{%
7072   \Glsentrylong{##1}\space(\acronymfont{\glsentryshort{##1}})%
7073 }%
7074 \renewcommand*{\glsentryfullpl}[1]{%
7075   \glsentrylongpl{##1}\space(\acronymfont{\glsentryshortpl{##1}})%
7076 }%
7077 \renewcommand*{\Glsentryfullpl}[1]{%
7078   \Glsentrylongpl{##1}\space(\acronymfont{\glsentryshortpl{##1}})%
7079 }%
7080 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
7081 \renewcommand*{\acronymsort}[2]{##1}%
7082 \renewcommand*{\acronymfont}[1]{##1}%
7083 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
7084 }
```

dua-desc *<long>* only acronym style with user-supplied description.

```
7085 \newacronymstyle{dua-desc}%
7086 {%
7087   \GlsUseAcrEntryDispStyle{dua}%
7088 }%
7089 {%
7090   \GlsUseAcrStyleDefs{dua}%
7091   \renewcommand*{\GenericAcronymFields}{}%

7092   \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentrylong{##1}}}%
7093   \renewcommand*{\acronymsort}[2]{##2}%
7094 }%
```

footnote *<short>*\footnote{*<long>*} acronym style.

```
7095 \newacronymstyle{footnote}%
7096 {%

  Check for long form in case this is a mixed glossary.
7097   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
7098 }%
7099 {%
7100   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
```

Need to ensure hyperlinks are switched off on first use:

```
7101 \glshyperfirstfalse
7102 \renewcommand*{\genacrfullformat}[2]{%
7103   \protect\firstacronymfont{\glsentryshort{##1}}##2%
7104   \protect\footnote{\glsentrylong{##1}}%
7105 }%
7106 \renewcommand*{\Genacrfullformat}[2]{%
7107   \firstacronymfont{\Glsentryshort{##1}}##2%
7108   \protect\footnote{\glsentrylong{##1}}%
7109 }%
7110 \renewcommand*{\genplacrfullformat}[2]{%
7111   \protect\firstacronymfont{\glsentryshortpl{##1}}##2%
7112   \protect\footnote{\glsentrylongpl{##1}}%
7113 }%
7114 \renewcommand*{\Genplacrfullformat}[2]{%
7115   \protect\firstacronymfont{\Glsentryshortpl{##1}}##2%
7116   \protect\footnote{\glsentrylongpl{##1}}%
7117 }%
7118 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
7119 \renewcommand*{\acronymsort}[2]{##1}%
7120 \renewcommand*{\acronymfont}[1]{##1}%
7121 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%

  Don't use footnotes for \acrfull:
```

```
7122 \renewcommand*{\acrfullfmt}[3]{%
7123   \glslink[##1]{##2}{\acronymfont{\glsentryshort{##2}}##3\space
7124   (\glsentrylong{##2})}%
```

```

7125 \renewcommand*{\Acrfullfmt}[3]{%
7126   \glslink[##1]{##2}{\acronymfont{\Glsentryshort{##2}}##3\space
7127   (\glsentrylong{##2})}%
7128 \renewcommand*{\ACRfullfmt}[3]{%
7129   \glslink[##1]{##2}{%
7130     \mfirstucMakeUppercase{\acronymfont{\glsentryshort{##2}}##3\space
7131     (\glsentrylong{##2})}}}%
7132 \renewcommand*{\acrfullplfmt}[3]{%
7133   \glslink[##1]{##2}{\acronymfont{\glsentryshortpl{##2}}##3\space
7134   (\glsentrylongpl{##2})}%
7135 \renewcommand*{\Acrfullplfmt}[3]{%
7136   \glslink[##1]{##2}{\acronymfont{\Glsentryshortpl{##2}}##3\space
7137   (\glsentrylongpl{##2})}%
7138 \renewcommand*{\ACRfullplfmt}[3]{%
7139   \glslink[##1]{##2}{%
7140     \mfirstucMakeUppercase{\acronymfont{\glsentryshortpl{##2}}##3\space
7141     (\glsentrylongpl{##2})}}}%

```

Similarly for \glsentryfull etc:

```

7142 \renewcommand*{\glsentryfull}[1]{%
7143   \acronymfont{\glsentryshort{##1}}\space(\glsentrylong{##1})}%
7144 \renewcommand*{\Glsentryfull}[1]{%
7145   \acronymfont{\Glsentryshort{##1}}\space(\glsentrylong{##1})}%
7146 \renewcommand*{\glsentryfullpl}[1]{%
7147   \acronymfont{\glsentryshortpl{##1}}\space(\glsentrylongpl{##1})}%
7148 \renewcommand*{\Glsentryfullpl}[1]{%
7149   \acronymfont{\Glsentryshortpl{##1}}\space(\glsentrylongpl{##1})}%
7150 }

```

footnote-sc \textsc{<short>}\footnote{<long>} acronym style.

```

7151 \newacronymstyle{footnote-sc}%
7152 {%
7153   \GlsUseAcrEntryDispStyle{footnote}%
7154 }%
7155 {%
7156   \GlsUseAcrStyleDefs{footnote}%
7157   \renewcommand{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}
7158   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
7159   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7160 }%

```

footnote-sm \textsmaller{<short>}\footnote{<long>} acronym style.

```

7161 \newacronymstyle{footnote-sm}%
7162 {%
7163   \GlsUseAcrEntryDispStyle{footnote}%
7164 }%
7165 {%
7166   \GlsUseAcrStyleDefs{footnote}%
7167   \renewcommand{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}
7168   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%

```

```
7169 \renewcommand*\acrpluralsuffix{\glsacrpluralsuffix}%
7170 }%
```

footnote-desc *<short>*\footnote{*<long>*} acronym style that has an accompanying description (which the user needs to supply).

```
7171 \newacronymstyle{footnote-desc}%
7172 {%
7173 \GlsUseAcrEntryDispStyle{footnote}%
7174 }%
7175 {%
7176 \GlsUseAcrStyleDefs{footnote}%
7177 \renewcommand*\GenericAcronymFields{}%
7178 \renewcommand*\acronymsort}[2]{##2}%
7179 \renewcommand*\acronymentry}[1]{%
7180 \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
7181 }
```

ootnote-sc-desc \textsc{*<short>*}\footnote{*<long>*} acronym style that has an accompanying description (which the user needs to supply).

```
7182 \newacronymstyle{footnote-sc-desc}%
7183 {%
7184 \GlsUseAcrEntryDispStyle{footnote-sc}%
7185 }%
7186 {%
7187 \GlsUseAcrStyleDefs{footnote-sc}%
7188 \renewcommand*\GenericAcronymFields{}%
7189 \renewcommand*\acronymsort}[2]{##2}%
7190 \renewcommand*\acronymentry}[1]{%
7191 \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
7192 }
```

ootnote-sm-desc \textsmaller{*<short>*}\footnote{*<long>*} acronym style that has an accompanying description (which the user needs to supply).

```
7193 \newacronymstyle{footnote-sm-desc}%
7194 {%
7195 \GlsUseAcrEntryDispStyle{footnote-sm}%
7196 }%
7197 {%
7198 \GlsUseAcrStyleDefs{footnote-sm}%
7199 \renewcommand*\GenericAcronymFields{}%
7200 \renewcommand*\acronymsort}[2]{##2}%
7201 \renewcommand*\acronymentry}[1]{%
7202 \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
7203 }
```

AcronymSynonyms

```
7204 \newcommand*\DefineAcronymSynonyms{%
```

Short form

`\acs`
7205 `\let\acs\acrshort`
First letter uppercase short form

`\Acs`
7206 `\let\Acs\Acrshort`
Plural short form

`\acsp`
7207 `\let\acsp\acrshortpl`
First letter uppercase plural short form

`\Acsp`
7208 `\let\Acsp\Acrshortpl`
Long form

`\acl`
7209 `\let\acl\aclong`
Plural long form

`\aclp`
7210 `\let\aclp\aclongpl`
First letter upper case long form

`\Acl`
7211 `\let\Acl\Aclong`
First letter upper case plural long form

`\Aclp`
7212 `\let\Aclp\Aclongpl`
Full form

`\acf`
7213 `\let\acf\acrfull`
Plural full form

`\acfp`
7214 `\let\acfp\acrfullpl`
First letter upper case full form

`\Acf`
7215 `\let\Acf\Acrfull`

First letter upper case plural full form

`\Acfp`

```
7216 \let\Acfp\Acrfullpl
```

Standard form

`\ac`

```
7217 \let\ac\gls
```

First upper case standard form

`\Ac`

```
7218 \let\Ac\Gls
```

Standard plural form

`\acp`

```
7219 \let\acp\glspl
```

Standard first letter upper case plural form

`\Acp`

```
7220 \let\Acp\Glspl
```

```
7221 }
```

Define synonyms if required

```
7222 \ifglsacrshortcuts
```

```
7223 \DefineAcronymSynonyms
```

```
7224 \fi
```

These commands for setting the style are now deprecated but are kept for backward compatibility.

`\glsAcronymDisplayStyle` Sets the default acronym display style for given glossary.

```
7225 \newcommand*{\SetDefaultAcronymDisplayStyle}[1]{%
```

```
7226 \defglsentryfmt[#1]{\glsentryfmt}%
```

```
7227 }
```

`\glsNewAcronymDef` Sets up the acronym definition for the default style. The information is provided by the tokens `\glslabeltok`, `\glsshorttok`, `\glslongtok` and `\glskeylisttok`.

```
7228 \newcommand*{\DefaultNewAcronymDef}{%
```

```
7229 \edef\@do@newglossaryentry{%
```

```
7230 \noexpand\newglossaryentry{\the\glslabeltok}%
```

```
7231 {%
```

```
7232 type=\acronymtype,%
```

```
7233 name={\the\glsshorttok},%
```

```
7234 sort={\the\glsshorttok},%
```

```
7235 text={\the\glsshorttok},%
```

```
7236 first={\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
```

```
7237 plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
```



```

7238     firstplural={\acrfullformat{\noexpand\expandonce\noexpand\@glo@longpl}%
7239                               {\noexpand\expandonce\noexpand\@glo@shortpl}},%
7240     short={\the\glsshorttok},%
7241     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7242     long={\the\glslongtok},%
7243     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7244     description={\the\glslongtok},%
7245     descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%

```

Remaining options specified by the user:

```

7246     \the\glskeylisttok
7247     }%
7248     }%
7249     \let\@org@gls@assign@firstpl\gls@assign@firstpl
7250     \let\@org@gls@assign@plural\gls@assign@plural
7251     \let\@org@gls@assign@descplural\gls@assign@descplural
7252     \def\gls@assign@firstpl##1##2{%
7253       \@gls@expand@field{##1}{firstpl}{##2}%
7254     }%
7255     \def\gls@assign@plural##1##2{%
7256       \@gls@expand@field{##1}{plural}{##2}%
7257     }%
7258     \def\gls@assign@descplural##1##2{%
7259       \@gls@expand@field{##1}{descplural}{##2}%
7260     }%
7261     \do@newglossaryentry
7262     \let\gls@assign@firstpl\@org@gls@assign@firstpl
7263     \let\gls@assign@plural\@org@gls@assign@plural
7264     \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7265 }

```

`\ultAcronymStyle` Set up the default acronym style:

```

7266 \newcommand*{\SetDefaultAcronymStyle}{%

```

Set the display style:

```

7267   \@for\@gls@type:=\@gls@acronymlists\do{%
7268     \SetDefaultAcronymDisplayStyle{\@gls@type}%
7269   }%

```

Set up the definition of `\newacronym`:

```

7270 \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).

```

7271   \ifx\@gls@acronymlists\@empty
7272     \def\@glo@type{\acronymtype}%
7273     \setkeys{glossentry}{##1}%
7274     \DeclareAcronymList{\@glo@type}%
7275     \SetDefaultAcronymDisplayStyle{\@glo@type}%
7276   \fi
7277   \glskeylisttok{##1}%

```

```

7278 \glslabeltok{##2}%
7279 \glsshorttok{##3}%
7280 \glslongtok{##4}%
7281 \newacronymhook
7282 \DefaultNewAcronymDef
7283 }%
7284 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
7285 }

```

`\acrfootnote` Used by the footnote acronym styles.

```
7286 \newcommand*{\acrfootnote}[3]{\acrlinkfootnote{#1}{#2}{#3}}
```

`acrlinkfootnote`

```

7287 \newcommand*{\acrlinkfootnote}[3]{%
7288 \footnote{\glslink[#1]{#2}{#3}}%
7289 }

```

`acrnolinkfootnote`

```

7290 \newcommand*{\acrnolinkfootnote}[3]{%
7291 \footnote{#3}%
7292 }

```

`acronymDisplayStyle` Sets the acronym display style for given glossary for the description and footnote combination.

```

7293 \newcommand*{\SetDescriptionFootnoteAcronymDisplayStyle}[1]{%
7294 \defglsentryfmt[#1]{%
7295 \ifdefempty\glscustomtext
7296 {%
7297 \ifglsused{\glslabel}%
7298 {%
7299 \acronymfont{\glsgenentryfmt}%
7300 }%
7301 {%
7302 \firstacronymfont{\glsgenentryfmt}%
7303 \ifgls hassymbol{\glslabel}%
7304 {%
7305 \expandafter\protect\expandafter\acrfootnote\expandafter
7306 {\@gls@link@opts}{\@gls@link@label}%
7307 {%
7308 \glsifplural
7309 {\glsentrysymbolplural{\glslabel}}%
7310 {\glsentrysymbol{\glslabel}}%
7311 }%
7312 }%
7313 }%
7314 }%
7315 {\glscustomtext\glsinsert}%
7316 }%
7317 }

```

teNewAcronymDef

```
7318 \newcommand*{\DescriptionFootnoteNewAcronymDef}{%
7319 \edef\@do@newglossaryentry{%
7320 \noexpand\newglossaryentry{\the\glslabeltok}%
7321 {%
7322 type=\acronymtype,%
7323 name={\noexpand\acronymfont{\the\glsshorttok}},%
7324 sort={\the\glsshorttok},%
7325 first={\the\glsshorttok},%
7326 firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7327 text={\the\glsshorttok},%
7328 plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7329 short={\the\glsshorttok},%
7330 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7331 long={\the\glslongtok},%
7332 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7333 symbol={\the\glslongtok},%
7334 symbolplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7335 \the\glskeylisttok
7336 }%
7337 }%
7338 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7339 \let\@org@gls@assign@plural\gls@assign@plural
7340 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7341 \def\gls@assign@firstpl##1##2{%
7342 \@gls@expand@field{##1}{firstpl}{##2}%
7343 }%
7344 \def\gls@assign@plural##1##2{%
7345 \@gls@expand@field{##1}{plural}{##2}%
7346 }%
7347 \def\gls@assign@symbolplural##1##2{%
7348 \@gls@expand@field{##1}{symbolplural}{##2}%
7349 }%
7350 \@do@newglossaryentry
7351 \let\gls@assign@plural\@org@gls@assign@plural
7352 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7353 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7354 }
```

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.

```
7355 \newcommand*{\SetDescriptionFootnoteAcronymStyle}{%
7356 \renewcommand{\newacronym}[4][ ]{%
7357 \ifx\@glsacronymlists\@empty
7358 \def\@glo@type{\acronymtype}%
7359 \setkeys{glossentry}{##1}%
7360 \DeclareAcronymList{\@glo@type}%
```

```

7361     \SetDescriptionFootnoteAcronymDisplayStyle{\@glo@type}%
7362     \fi
7363     \glskeylisttok{##1}%
7364     \glslabeltok{##2}%
7365     \glsshorttok{##3}%
7366     \glslongtok{##4}%
7367     \newacronymhook
7368     \DescriptionFootnoteNewAcronymDef
7369 }%

```

If footnote package option is specified, set the first use to append the long form (stored in symbol) as a footnote.

```

7370 \@for\@gls@type:=\@glsacronymlists\do{%
7371     \SetDescriptionFootnoteAcronymDisplayStyle{\@gls@type}%
7372 }%

```

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.

```

7373 \ifglsacrsmallcaps
7374     \renewcommand*\acronymfont}[1]{\textsc{##1}}%
7375     \renewcommand*\acrpluralsuffix{\glsupacrpluralsuffix}%
7376 \else
7377     \ifglsacrsmaller
7378         \renewcommand*\acronymfont}[1]{\textsmaller{##1}}%
7379     \fi
7380 \fi

```

Check for package option clash

```

7381 \ifglsacrdua
7382     \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’
7383     can’t both be set}{}%
7384 \fi
7385 }%

```

`nymDisplayStyle` Sets the acronym display style for given glossary with description and dua combination.

```

7386 \newcommand*\SetDescriptionDUAAcronymDisplayStyle}[1]{%
7387     \defglsentryfmt[##1]{\glsgenentryfmt}%
7388 }

```

`UANewAcronymDef`

```

7389 \newcommand*\DescriptionDUANewAcronymDef}{%
7390     \edef\@do@newglossaryentry{%
7391         \noexpand\newglossaryentry{\the\glslabeltok}%
7392         {%
7393             type=\acronymtype,%
7394             name={\the\glslongtok},%
7395             sort={\the\glslongtok},%
7396             text={\the\glslongtok},%
7397             first={\the\glslongtok},%

```

```

7398 plural={\noexpand\expandonce\noexpand\@glo@longpl},%
7399 firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7400 short={\the\glsshorttok},%
7401 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7402 long={\the\glslongtok},%
7403 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7404 symbol={\the\glsshorttok},%
7405 symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7406 \the\glskeylisttok
7407 }%
7408 }%
7409 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7410 \let\@org@gls@assign@plural\gls@assign@plural
7411 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7412 \def\gls@assign@firstpl##1##2{%
7413 \@@gls@expand@field{##1}{firstpl}{##2}%
7414 }%
7415 \def\gls@assign@plural##1##2{%
7416 \@@gls@expand@field{##1}{plural}{##2}%
7417 }%
7418 \def\gls@assign@symbolplural##1##2{%
7419 \@@gls@expand@field{##1}{symbolplural}{##2}%
7420 }%
7421 \@do@newglossaryentry
7422 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7423 \let\gls@assign@plural\@org@gls@assign@plural
7424 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7425 }

```

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.

```

7426 \newcommand*\SetDescriptionDUAACronymStyle{%
7427 \ifglsacrsmallcaps
7428 \PackageError{glossaries}{Option clash: 'smallcaps' and 'dua'
7429 can't both be set}{}%
7430 \else
7431 \ifglsacrsmaller
7432 \PackageError{glossaries}{Option clash: 'smaller' and 'dua'
7433 can't both be set}{}%
7434 \fi
7435 \fi
7436 \renewcommand{\newacronym}[4] []{%
7437 \ifx\@glsacronymlists\@empty
7438 \def\@glo@type{\acronymtype}%
7439 \setkeys{glossentry}{##1}%
7440 \DeclareAcronymList{\@glo@type}%
7441 \SetDescriptionDUAACronymDisplayStyle{\@glo@type}%
7442 \fi

```

```

7443 \glskeylisttok{##1}%
7444 \glslabeltok{##2}%
7445 \glsshorttok{##3}%
7446 \glslongtok{##4}%
7447 \newacronymhook
7448 \DescriptionDUANewAcronymDef
7449 }%

```

Set display.

```

7450 \@for\@gls@type:=\@glsacronymlists\do{%
7451 \SetDescriptionDUAAcronymDisplayStyle{\@gls@type}%
7452 }%
7453 }%

```

`\acronymDisplayStyle` Sets the acronym display style for given glossary using the description setting (but not footnote or dua).

```

7454 \newcommand*{\SetDescriptionAcronymDisplayStyle}[1]{%
7455 \defglsentryfmt[#1]{%

7456 \ifdefempty\glscustomtext
7457 {%
7458 \ifglsused{\glslabel}%
7459 {%

```

Move the inserted text outside of `\acronymfont`

```

7460 \let\gls@org@insert\glsinsert
7461 \let\glsinsert\@empty
7462 \acronymfont{\glsgenentryfmt}\gls@org@insert
7463 }%
7464 {%
7465 \glsgenentryfmt
7466 \ifgls hassymbol{\glslabel}%
7467 {%
7468 \glsifplural
7469 {%
7470 \def\@glo@symbol{\glsentrysymbolplural{\glslabel}}%
7471 }%
7472 {%
7473 \def\@glo@symbol{\glsentrysymbol{\glslabel}}%
7474 }%
7475 \space(\protect\firstacronymfont
7476 {\gls caps case
7477 {\@glo@symbol}
7478 {\@glo@symbol}
7479 {\mfirstucMakeUppercase{\@glo@symbol}}})%
7480 }%
7481 {}%
7482 }%
7483 }%
7484 {\glscustomtext\glsinsert}%

```

```
7485 }%
7486 }
```

onNewAcronymDef

```
7487 \newcommand*{\DescriptionNewAcronymDef}{%
7488 \edef\@do@newglossaryentry{%
7489 \noexpand\newglossaryentry{\the\glslabeltok}%
7490 {%
7491 type=\acronymtype,%
7492 name={\noexpand
7493 \acronymformat{\the\glsshorttok}{\the\glslongtok}},%
7494 sort={\the\glsshorttok},%
7495 first={\the\glslongtok},%
7496 firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7497 text={\the\glsshorttok},%
7498 plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7499 short={\the\glsshorttok},%
7500 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7501 long={\the\glslongtok},%
7502 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7503 symbol={\noexpand\@glo@text},%
7504 symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7505 \the\glskeylisttok}%
7506 }%
7507 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7508 \let\@org@gls@assign@plural\gls@assign@plural
7509 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7510 \def\gls@assign@firstpl##1##2{%
7511 \@@gls@expand@field{##1}{firstpl}{##2}%
7512 }%
7513 \def\gls@assign@plural##1##2{%
7514 \@@gls@expand@field{##1}{plural}{##2}%
7515 }%
7516 \def\gls@assign@symbolplural##1##2{%
7517 \@@gls@expand@field{##1}{symbolplural}{##2}%
7518 }%
7519 \@do@newglossaryentry
7520 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7521 \let\gls@assign@plural\@org@gls@assign@plural
7522 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7523 }
```

ionAcronymStyle Option description is used, but not dua or footnote. Store long form in first key and short form in text and symbol key. The name is stored using \acronymformat to allow the user to override the way the name is displayed in the list of acronyms.

```
7524 \newcommand*{\SetDescriptionAcronymStyle}{%
7525 \renewcommand{\newacronym}[4][ ]{%
7526 \ifx\@glsacronymlists\@empty
7527 \def\@glo@type{\acronymtype}%
7528 }
```

```

7528     \setkeys{glossentry}{##1}%
7529     \DeclareAcronymList{\@glo@type}%
7530     \SetDescriptionAcronymDisplayStyle{\@glo@type}%
7531     \fi
7532     \glskeylisttok{##1}%
7533     \glslabeltok{##2}%
7534     \glsshorttok{##3}%
7535     \glslongtok{##4}%
7536     \newacronymhook
7537     \DescriptionNewAcronymDef
7538 }%

```

Set display.

```

7539 \@for\@gls@type:=\@glsacronymlists\do{%
7540   \SetDescriptionAcronymDisplayStyle{\@gls@type}%
7541 }%

```

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.

```

7542 \ifglsacrsmallcaps
7543   \renewcommand{\acronymfont}[1]{\textsc{##1}}
7544   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7545 \else
7546   \ifglsacrsmaller
7547     \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
7548   \fi
7549 \fi
7550 }%

```

`\acronymDisplayStyle` Sets the acronym display style for given glossary with footnote setting (but not description or dua).

```

7551 \newcommand*{\SetFootnoteAcronymDisplayStyle}[1]{%
7552   \defglsentryfmt[#1]{%

```

```

7553     \ifdefempty\glscustomtext
7554     {%

```

Move the inserted text outside of `\acronymfont`

```

7555     \let\gls@org@insert\glsinsert
7556     \let\glsinsert\@empty
7557     \ifglsused{\glslabel}%
7558     {%
7559       \acronymfont{\glsentryfmt}\gls@org@insert
7560     }%
7561     {%
7562       \firstacronymfont{\glsentryfmt}\gls@org@insert
7563       \ifglsahaslong{\glslabel}%
7564       {%
7565         \expandafter\protect\expandafter\acrfootnote\expandafter
7566         {\@gls@link@opts}{\@gls@link@label}%

```



```

7567         {%
7568         \glsifplural
7569         {\glsentrylongpl{\glslabel}}%
7570         {\glsentrylong{\glslabel}}%
7571         }%
7572     }%

7573     {%
7574     }%
7575 }%
7576 {\glscustomtext\glsinsert}%
7577 }%
7578 }

```

teNewAcronymDef

```

7579 \newcommand*{\FootnoteNewAcronymDef}{%
7580 \edef\@do@newglossaryentry{%
7581 \noexpand\newglossaryentry{\the\glslabeltok}%
7582 {%
7583     type=\acronymtype,%
7584     name={\noexpand\acronymfont{\the\glsshorttok}},%
7585     sort={\the\glsshorttok},%
7586     text={\the\glsshorttok},%
7587     plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7588     first={\the\glsshorttok},%
7589     firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7590     short={\the\glsshorttok},%
7591     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7592     long={\the\glslongtok},%
7593     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7594     description={\the\glslongtok},%
7595     descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7596     \the\glskeylisttok
7597 }%
7598 }%
7599 \let\@org@gls@assign@plural\gls@assign@plural
7600 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7601 \let\@org@gls@assign@descplural\gls@assign@descplural
7602 \def\gls@assign@firstpl##1##2{%
7603     \@gls@expand@field{##1}{firstpl}{##2}%
7604 }%
7605 \def\gls@assign@plural##1##2{%
7606     \@gls@expand@field{##1}{plural}{##2}%
7607 }%
7608 \def\gls@assign@descplural##1##2{%
7609     \@gls@expand@field{##1}{descplural}{##2}%
7610 }%
7611 \@do@newglossaryentry
7612 \let\gls@assign@plural\@org@gls@assign@plural
7613 \let\gls@assign@firstpl\@org@gls@assign@firstpl

```

```

7614 \let\gls@assign@descplural\@org@gls@assign@descplural
7615 }

```

`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.

```

7616 \newcommand*{\SetFootnoteAcronymStyle}{%
7617   \renewcommand{\newacronym}[4][]{%
7618     \ifx\@glsacronymlists\@empty
7619       \def\@glo@type{\acronymtype}%
7620       \setkeys{glossentry}{##1}%
7621       \DeclareAcronymList{\@glo@type}%
7622       \SetFootnoteAcronymDisplayStyle{\@glo@type}%
7623     \fi
7624     \glskeylisttok{##1}%
7625     \glslabeltok{##2}%
7626     \glsshorttok{##3}%
7627     \glslongtok{##4}%
7628     \newacronymhook
7629     \FootnoteNewAcronymDef
7630   }%

```

Set display

```

7631 \@for\@gls@type:=\@glsacronymlists\do{%
7632   \SetFootnoteAcronymDisplayStyle{\@gls@type}%
7633 }%

```

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.

```

7634 \ifglsacrsmallcaps
7635   \renewcommand*{\acronymfont}[1]{\textsc{##1}}%
7636   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7637 \else
7638   \ifglsacrsmaller
7639     \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
7640   \fi
7641 \fi

```

Check for option clash

```

7642 \ifglsacrdua
7643   \PackageError{glossaries}{Option clash: 'footnote' and 'dua'
7644     can't both be set}{}%
7645 \fi
7646 }%

```

`parenifnotempty` Do a space followed by the argument if the argument doesn't expand to empty or `\relax`. If argument isn't empty (or `\relax`), apply the macro to it given in the second argument.

```

7647 \DeclareRobustCommand*{\glsdoparenifnotempty}[2]{%
7648   \protected@edef\gls@tmp{##1}%
7649   \ifdefempty\gls@tmp
7650     {}%

```

```

7651  {%
7652    \ifx\gls@tmp\@gls@default@value
7653    \else
7654      \space (#2{#1})%
7655    \fi
7656  }%
7657 }

```

`\gls@acronymDisplayStyle` Sets the acronym display style for given glossary where neither footnote nor description is required, but smallcaps or smaller specified.

```

7658 \newcommand*{\SetSmallAcronymDisplayStyle}[1]{%
7659   \defglsentryfmt[#1]{%

```

```

7660     \ifdefempty\gls@customtext
7661     {%

```

Move the inserted text outside of `\acronymfont`

```

7662       \let\gls@org@insert\glsinsert
7663       \let\glsinsert\@empty
7664       \ifglsused{\glslabel}%
7665       {%
7666         \acronymfont{\gls@genentryfmt}\gls@org@insert
7667       }%
7668       {%
7669         \gls@genentryfmt
7670         \ifgls@hassymbol{\glslabel}%
7671         {%
7672           \gls@ifplural
7673           {%
7674             \def\@glo@symbol{\gls@entrysymbolplural{\glslabel}}%
7675           }%
7676           {%
7677             \def\@glo@symbol{\gls@entrysymbol{\glslabel}}%
7678           }%
7679           \space
7680           (\gls@scapscase
7681            {\firstacronymfont{\@glo@symbol}}%
7682            {\firstacronymfont{\@glo@symbol}}%
7683            {\firstacronymfont{\mfirstucMakeUppercase{\@glo@symbol}}})%
7684         }%
7685       }%
7686     }%
7687   }%
7688   {\gls@customtext\glsinsert}%
7689 }%
7690 }

```

`\SmallNewAcronymDef`

```

7691 \newcommand*{\SmallNewAcronymDef}{%

```

```

7692 \edef\@do@newglossaryentry{%
7693   \noexpand\newglossaryentry{\the\glslabeltok}%
7694   {%
7695     type=\acronymtype,%
7696     name={\noexpand\acronymfont{\the\glsshorttok}},%
7697     sort={\the\glsshorttok},%
7698     text={\the\glsshorttok},%

Default to the short plural.
7699     plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7700     first={\the\glslongtok},%

Default to the long plural.
7701     firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7702     short={\the\glsshorttok},%
7703     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7704     long={\the\glslongtok},%
7705     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7706     description={\noexpand\@glo@first},%
7707     descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7708     symbol={\the\glsshorttok},%

Default to the short plural.
7709     symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7710     \the\glskeylisttok
7711   }%
7712 }%
7713 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7714 \let\@org@gls@assign@plural\gls@assign@plural
7715 \let\@org@gls@assign@descplural\gls@assign@descplural
7716 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7717 \def\gls@assign@firstpl##1##2{%
7718   \@@gls@expand@field{##1}{firstpl}{##2}%
7719 }%
7720 \def\gls@assign@plural##1##2{%
7721   \@@gls@expand@field{##1}{plural}{##2}%
7722 }%
7723 \def\gls@assign@descplural##1##2{%
7724   \@@gls@expand@field{##1}{descplural}{##2}%
7725 }%
7726 \def\gls@assign@symbolplural##1##2{%
7727   \@@gls@expand@field{##1}{symbolplural}{##2}%
7728 }%
7729 \@do@newglossaryentry
7730 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7731 \let\gls@assign@plural\@org@gls@assign@plural
7732 \let\gls@assign@descplural\@org@gls@assign@descplural
7733 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7734 }

```

`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.

```
7735 \newcommand*{\SetSmallAcronymStyle}{%
7736   \renewcommand{\newacronym}[4] []{%
7737     \ifx\@glsacronymlists\@empty
7738       \def\@glo@type{\acronymtype}%
7739       \setkeys{glossentry}{##1}%
7740       \DeclareAcronymList{\@glo@type}%
7741       \SetSmallAcronymDisplayStyle{\@glo@type}%
7742     \fi
7743     \glskeylisttok{##1}%
7744     \glslabeltok{##2}%
7745     \glsshorttok{##3}%
7746     \glslongtok{##4}%
7747     \newacronymhook
7748     \SmallNewAcronymDef
7749   }%
```

Change the display since first only contains long form.

```
7750 \@for\@gls@type:=\@glsacronymlists\do{%
7751   \SetSmallAcronymDisplayStyle{\@gls@type}%
7752 }%
```

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.

```
7753 \ifglsacrsmallcaps
7754   \renewcommand*{\acronymfont}[1]{\textsc{##1}}
7755   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7756 \else
7757   \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}
7758 \fi
```

check for option clash

```
7759 \ifglsacrdua
7760   \ifglsacrsmallcaps
7761     \PackageError{glossaries}{Option clash: ‘smallcaps’ and ‘dua’
7762     can’t both be set}{}%
7763   \else
7764     \PackageError{glossaries}{Option clash: ‘smaller’ and ‘dua’
7765     can’t both be set}{}%
7766   \fi
7767 \fi
7768 }%
```

`DUADisplayStyle` Sets the acronym display style for given glossary with dua setting.

```
7769 \newcommand*{\SetDUADisplayStyle}[1]{%
7770   \defglsentryfmt[#1]{\glsentryfmt}%
7771 }
```

`UANewAcronymDef`

```
7772 \newcommand*{\DUANewAcronymDef}{%
```

```

7773 \edef\@do@newglossaryentry{%
7774   \noexpand\newglossaryentry{\the\glslabeltok}%
7775   {%
7776     type=\acronymtype,%
7777     name={\the\glsshorttok},%
7778     text={\the\glslongtok},%
7779     first={\the\glslongtok},%
7780     plural={\noexpand\expandonce\noexpand\@glo@longpl},%
7781     firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7782     short={\the\glsshorttok},%
7783     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7784     long={\the\glslongtok},%
7785     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7786     description={\the\glslongtok},%
7787     descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7788     symbol={\the\glsshorttok},%
7789     symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7790     \the\glskeylisttok
7791   }%
7792 }%
7793 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7794 \let\@org@gls@assign@plural\gls@assign@plural
7795 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7796 \let\@org@gls@assign@descplural\gls@assign@descplural
7797 \def\gls@assign@firstpl##1##2{%
7798   \@gls@expand@field{##1}{firstpl}{##2}%
7799 }%
7800 \def\gls@assign@plural##1##2{%
7801   \@gls@expand@field{##1}{plural}{##2}%
7802 }%
7803 \def\gls@assign@symbolplural##1##2{%
7804   \@gls@expand@field{##1}{symbolplural}{##2}%
7805 }%
7806 \def\gls@assign@descplural##1##2{%
7807   \@gls@expand@field{##1}{descplural}{##2}%
7808 }%
7809 \@do@newglossaryentry
7810 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7811 \let\gls@assign@plural\@org@gls@assign@plural
7812 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7813 \let\gls@assign@descplural\@org@gls@assign@descplural
7814 }

```

\SetDUASyle Always expand acronyms.

```

7815 \newcommand*\SetDUASyle{%
7816   \renewcommand{\newacronym}[4] []{%
7817     \ifx\@glsacronymlists\empty
7818       \def\@glo@type{\acronymtype}%
7819       \setkeys{glossentry}{##1}%

```

```

7820     \DeclareAcronymList{\@glo@type}%
7821     \SetDUADisplayStyle{\@glo@type}%
7822     \fi
7823     \glskeylisttok{##1}%
7824     \glslabeltok{##2}%
7825     \glsshorttok{##3}%
7826     \glslongtok{##4}%
7827     \newacronymhook
7828     \DUANewAcronymDef
7829 }%
```

Set the display

```

7830 \@for\@gls@type:=\@glsacronymlists\do{%
7831   \SetDUADisplayStyle{\@gls@type}%
7832 }%
7833 }
```

SetAcronymStyle

```

7834 \newcommand*\SetAcronymStyle{%
7835   \SetDefaultAcronymStyle
7836   \ifglsacrdescription
7837   \ifglsacrfootnote
7838     \SetDescriptionFootnoteAcronymStyle
7839   \else
7840     \ifglsacrdua
7841       \SetDescriptionDUAAcronymStyle
7842     \else
7843       \SetDescriptionAcronymStyle
7844     \fi
7845   \fi
7846 \else
7847   \ifglsacrfootnote
7848     \SetFootnoteAcronymStyle
7849   \else
7850     \ifthenelse{\boolean{glsacrsmalldcaps}\OR
7851       \boolean{glsacrsmalld}}{%
7852       {%
7853         \SetSmallAcronymStyle
7854       }%
7855     }%
7856     \ifglsacrdua
7857       \SetDUASyle
7858     \fi
7859   }%
7860 \fi
7861 \fi
7862 }
```

Set the acronym style according to the package options

```

7863 \SetAcronymStyle
```

Allow user to define their own custom acronyms. (For compatibility with versions before v3.0, the short form is stored in the user1 key, the plural short form is stored in the user2 key, the long form is stored in the user3 key and the plural long form is stored in the user4 key.) Defaults to displaying only the acronym with the long form as the description.

`\SetCustomDisplayStyle` Sets the acronym display style.

```
7864 \newcommand*\SetCustomDisplayStyle}[1]{%
7865   \def\glsentryfmt[#1]{\glsentryfmt}%
7866 }
```

`\CustomAcronymFields`

```
7867 \newcommand*\CustomAcronymFields{%
7868   name={\the\glsshorttok},%
7869   description={\the\glslongtok},%
7870   first={\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
7871   firstplural={\acrfullformat
7872     {\noexpand\glsentrylongpl{\the\glslabeltok}}%
7873     {\noexpand\glsentryshortpl{\the\glslabeltok}}},%

7874   text={\the\glsshorttok},%
7875   plural={\the\glsshorttok\noexpand\acrpluralsuffix}%
7876 }
```

`\CustomNewAcronymDef`

```
7877 \newcommand*\CustomNewAcronymDef{%
7878   \protected@edef\do@newglossaryentry{%
7879     \noexpand\newglossaryentry{\the\glslabeltok}%
7880     {%
7881       type=\acronymtype,%
7882       short={\the\glsshorttok},%
7883       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7884       long={\the\glslongtok},%
7885       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7886       user1={\the\glsshorttok},%
7887       user2={\the\glsshorttok\noexpand\acrpluralsuffix},%
7888       user3={\the\glslongtok},%
7889       user4={\the\glslongtok\noexpand\acrpluralsuffix},%
7890       \CustomAcronymFields,%
7891       \the\glskeylisttok
7892     }%
7893   }%
7894   \@do@newglossaryentry
7895 }
```

`\SetCustomStyle`

```
7896 \newcommand*\SetCustomStyle{%
7897   \renewcommand{\newacronym}[4][ ]{%
7898     \ifx\@glsacronymlists@empty
7899       \def\@glo@type{\acronymtype}%
7900     }
```



```

7900     \setkeys{glossentry}{##1}%
7901     \DeclareAcronymList{\@glo@type}%
7902     \SetCustomDisplayStyle{\@glo@type}%
7903     \fi
7904     \glskeylisttok{##1}%
7905     \glslabeltok{##2}%
7906     \glsshorttok{##3}%
7907     \glslongtok{##4}%
7908     \newacronymhook
7909     \CustomNewAcronymDef
7910 }%

Set the display
7911 \@for\@gls@type:=\@gls@acronymlists\do{%
7912     \SetCustomDisplayStyle{\@gls@type}%
7913 }%
7914 }

```

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.

```
7915 \RequirePackage{glossary-hypernav}
```

The styles that use list-like environments. These are not loaded if the `nolist` option is used:

```
7916 \@gls@loadlist
```

The styles that use the `longtable` environment. These are not loaded if the `nolong` package option is used.

```
7917 \@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.

```
7918 \@gls@loadsuper
```

The tree-like styles. These are not loaded if the `notree` package option is used.

```
7919 \@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.

```

7920 \ifx\@glossary@default@style\relax
7921 \else
7922   \setglossarystyle{\@glossary@default@style}
7923 \fi

```

1.20 Debugging Commands

`\showgloparent` `\showgloparent{<label>}`

```
7924 \newcommand*{\showgloparent}[1]{%
7925   \expandafter\show\csname glo@glstetoklabel{#1}@parent\endcsname
7926 }
```

`\showglolevel` `\showglolevel{<label>}`

```
7927 \newcommand*{\showglolevel}[1]{%
7928   \expandafter\show\csname glo@glstetoklabel{#1}@level\endcsname
7929 }
```

`\showglotext` `\showglotext{<label>}`

```
7930 \newcommand*{\showglotext}[1]{%
7931   \expandafter\show\csname glo@glstetoklabel{#1}@text\endcsname
7932 }
```

`\showgloplural` `\showgloplural{<label>}`

```
7933 \newcommand*{\showgloplural}[1]{%
7934   \expandafter\show\csname glo@glstetoklabel{#1}@plural\endcsname
7935 }
```

`\showglofirst` `\showglofirst{<label>}`

```
7936 \newcommand*{\showglofirst}[1]{%
7937   \expandafter\show\csname glo@glstetoklabel{#1}@first\endcsname
7938 }
```

`\showglofirstpl` `\showglofirstpl{<label>}`

```
7939 \newcommand*{\showglofirstpl}[1]{%
7940   \expandafter\show\csname glo@glstetoklabel{#1}@firstpl\endcsname
7941 }
```

`\showglotype` `\showglotype{<label>}`

```
7942 \newcommand*{\showglotype}[1]{%
7943   \expandafter\show\csname glo@glstdetoklabel{#1}@type\endcsname
7944 }
```

`\showglocounter` `\showglocounter{<label>}`

```
7945 \newcommand*{\showglocounter}[1]{%
7946   \expandafter\show\csname glo@glstdetoklabel{#1}@counter\endcsname
7947 }
```

`\showglouser` `\showglouser{<label>}`

```
7948 \newcommand*{\showglouser}[1]{%
7949   \expandafter\show\csname glo@glstdetoklabel{#1}@user\endcsname
7950 }
```

`\showglouserii` `\showglouserii{<label>}`

```
7951 \newcommand*{\showglouserii}[1]{%
7952   \expandafter\show\csname glo@glstdetoklabel{#1}@userii\endcsname
7953 }
```

`\showglouseriii` `\showglouseriii{<label>}`

```
7954 \newcommand*{\showglouseriii}[1]{%
7955   \expandafter\show\csname glo@glstdetoklabel{#1}@useriii\endcsname
7956 }
```

`\showglouseriv` `\showglouseriv{<label>}`

```
7957 \newcommand*{\showglouseriv}[1]{%
7958   \expandafter\show\csname glo@glstdetoklabel{#1}@useriv\endcsname
7959 }
```

\showglouserv \showglouserv{<label>}

```
7960 \newcommand*{\showglouserv}[1]{%
7961 \expandafter\show\csname glo@glstdetoklabel{#1}@user\endcsname
7962 }
```

\showglouservi \showglouservi{<label>}

```
7963 \newcommand*{\showglouservi}[1]{%
7964 \expandafter\show\csname glo@glstdetoklabel{#1}@uservi\endcsname
7965 }
```

\showgloname \showgloname{<label>}

```
7966 \newcommand*{\showgloname}[1]{%
7967 \expandafter\show\csname glo@glstdetoklabel{#1}@name\endcsname
7968 }
```

\showglodesc \showglodesc{<label>}

```
7969 \newcommand*{\showglodesc}[1]{%
7970 \expandafter\show\csname glo@glstdetoklabel{#1}@desc\endcsname
7971 }
```

showglodescplural \showglodescplural{<label>}

```
7972 \newcommand*{\showglodescplural}[1]{%
7973 \expandafter\show\csname glo@glstdetoklabel{#1}@descplural\endcsname
7974 }
```

\showglosort \showglosort{<label>}

```
7975 \newcommand*{\showglosort}[1]{%
7976 \expandafter\show\csname glo@glstdetoklabel{#1}@sort\endcsname
7977 }
```

`\showglosymbol` `\showglosymbol{<label>}`

```
7978 \newcommand*{\showglosymbol}[1]{%
7979   \expandafter\show\csname glo@glstetoklabel{#1}@symbol\endcsname
7980 }
```

`glosymbolplural` `\showglosymbolplural{<label>}`

```
7981 \newcommand*{\showglosymbolplural}[1]{%
7982   \expandafter\show\csname glo@glstetoklabel{#1}@symbolplural\endcsname
7983 }
```

`\showgloshort` `\showgloshort{<label>}`

```
7984 \newcommand*{\showgloshort}[1]{%
7985   \expandafter\show\csname glo@glstetoklabel{#1}@short\endcsname
7986 }
```

`\showglolong` `\showglolong{<label>}`

```
7987 \newcommand*{\showglolong}[1]{%
7988   \expandafter\show\csname glo@glstetoklabel{#1}@long\endcsname
7989 }
```

`\showgloindex` `\showgloindex{<label>}`

```
7990 \newcommand*{\showgloindex}[1]{%
7991   \expandafter\show\csname glo@glstetoklabel{#1}@index\endcsname
7992 }
```

`\showgloflag` `\showgloflag{<label>}`

```
7993 \newcommand*{\showgloflag}[1]{%
7994   \expandafter\show\csname ifglo@glstetoklabel{#1}@flag\endcsname
7995 }
```

`\showgloclist` `\showgloclist{<label>}`

```
7996 \newcommand*{\showgloclist}[1]{%
7997   \expandafter\show\csname glo@glstetoklabel{#1}@loclist\endcsname
7998 }
```

`\showglofield` `\showglofield{<label>}{<field>}`

```
7999 \newcommand*{\showglofield}[2]{%
8000   \csshow{glo@glstetoklabel{#1}@#2}%
8001 }
```

`showacronymlists` `\showacronymlists`

Show list of glossaries that have been flagged as a list of acronyms.

```
8002 \newcommand*{\showacronymlists}{%
8003   \show@glstetokacronymlists
8004 }
```

`\showglossaries` `\showglossaries`

Show list of defined glossaries.

```
8005 \newcommand*{\showglossaries}{%
8006   \show@glo@types
8007 }
```

`\showglossaryin` `\showglossaryin{<glossary-label>}`

Show the 'in' extension for the given glossary.

```
8008 \newcommand*{\showglossaryin}[1]{%
8009   \expandafter\show\csname @glo@type@#1@in\endcsname
8010 }
```

`\showglossaryout` `\showglossaryout{<glossary-label>}`

Show the 'out' extension for the given glossary.

```
8011 \newcommand*{\showglossaryout}[1]{%
8012   \expandafter\show\csname @glo@type@#1@out\endcsname
8013 }
```

showglossarytitle

```
\showglossarytitle{<glossary-label>}
```

Show the title for the given glossary.

```
8014 \newcommand*{\showglossarytitle}[1]{%
8015   \expandafter\show\csname @glotype@#1@title\endcsname
8016 }
```

showglossarycounter

```
\showglossarycounter{<glossary-label>}
```

Show the counter for the given glossary.

```
8017 \newcommand*{\showglossarycounter}[1]{%
8018   \expandafter\show\csname @glotype@#1@counter\endcsname
8019 }
```

showglossaryentries

```
\showglossaryentries{<glossary-label>}
```

Show the list of entry labels for the given glossary.

```
8020 \newcommand*{\showglossaryentries}[1]{%
8021   \expandafter\show\csname glolist@#1\endcsname
8022 }
```

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.

```
8023 \csname ifglcompatible-2.07\endcsname
8024   \RequirePackage{glossaries-compatible-207}
8025 \fi
```

2 Prefix Support (glossaries-prefix Code)

This package provides a means of adding prefixes to your glossary entries. For example, you may want to use “`a \gls{<label>}`” on first use but use “`an \gls{<label>}`” on subsequent use.

```
8026 \NeedsTeXFormat{LaTeX2e}
```

```
8027 \ProvidesPackage{glossaries-prefix}[2019/01/06 v4.42 (NLCT)]
```

Pass all options to glossaries:

```
8028 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
8029 \ProcessOptions
```

Load glossaries:

```
8030 \RequirePackage{glossaries}
```

Add the new keys:

```
8031 \define@key{glossentry}{prefixfirst}{\def\@glo@entryprefixfirst{#1}}%
```

```
8032 \define@key{glossentry}{prefixfirstplural}{\def\@glo@entryprefixfirstplural{#1}}%
```

```
8033 \define@key{glossentry}{prefix}{\def\@glo@entryprefix{#1}}%
```

```
8034 \define@key{glossentry}{prefixplural}{\def\@glo@entryprefixplural{#1}}%
```

Add them to `\@gls@keymap`:

```
8035 \appto\@gls@keymap{,%
```

```
8036   {prefixfirst}{prefixfirst},%
```

```
8037   {prefixfirstplural}{prefixfirstplural},%
```

```
8038   {prefix}{prefix},%
```

```
8039   {prefixplural}{prefixplural}}%
```

```
8040 }
```

Set the default values:

```
8041 \appto\@newglossaryentryprehook{%
```

```
8042   \def\@glo@entryprefix{}}%
```

```
8043   \def\@glo@entryprefixplural{}}%
```

```
8044   \let\@glo@entryprefixfirst\@gls@default@value
```

```
8045   \let\@glo@entryprefixfirstplural\@gls@default@value
```

```
8046 }
```

Set the assignment code:

```
8047 \appto\@newglossaryentryposthook{%
```

```
8048   \gls@assign@field{ }\@glo@label}{prefix}{\@glo@entryprefix}}%
```

```
8049   \gls@assign@field{ }\@glo@label}{prefixplural}{\@glo@entryprefixplural}}%
```

If `prefixfirst` has not been supplied, make it the same as `prefix`.

```
8050 \expandafter\gls@assign@field\expandafter
```

```
8051   {\csname glo@\@glo@label @prefix\endcsname}{\@glo@label}{prefixfirst}}%
```

```
8052   {\@glo@entryprefixfirst}}%
```


If prefixfirstplural has not been supplied, make it the same as prefixplural.

```
8053 \expandafter\gls@assign@field\expandafter
8054   {\csname glo@\glo@label @prefixplural\endcsname}{\@glo@label}%
8055   {prefixfirstplural}{\@glo@entryprefixfirstplural}%
8056 }
```

Define commands to access these fields:

entryprefixfirst

```
8057 \newcommand*\glsentryprefixfirst[1]{\csuse{glo@#1@prefixfirst}}
```

entryprefixfirstplural

```
8058 \newcommand*\glsentryprefixfirstplural[1]{\csuse{glo@#1@prefixfirstplural}}
```

\glsentryprefix

```
8059 \newcommand*\glsentryprefix[1]{\csuse{glo@#1@prefix}}
```

entryprefixplural

```
8060 \newcommand*\glsentryprefixplural[1]{\csuse{glo@#1@prefixplural}}
```

Now for the initial upper case variants:

entryprefixfirst

```
8061 \newrobustcmd*\Glsentryprefixfirst[1]{%
8062   \protected@edef\@glo@text{\csname glo@#1@prefixfirst\endcsname}%
8063   \xmakefirstuc\@glo@text
8064 }
```

entryprefixfirstplural

```
8065 \newrobustcmd*\Glsentryprefixfirstplural[1]{%
8066   \protected@edef\@glo@text{\csname glo@#1@prefixfirstplural\endcsname}%
8067   \xmakefirstuc\@glo@text
8068 }
```

\Glsentryprefix

```
8069 \newrobustcmd*\Glsentryprefix[1]{%
8070   \protected@edef\@glo@text{\csname glo@#1@prefix\endcsname}%
8071   \xmakefirstuc\@glo@text
8072 }
```

entryprefixplural

```
8073 \newrobustcmd*\Glsentryprefixplural[1]{%
8074   \protected@edef\@glo@text{\csname glo@#1@prefixplural\endcsname}%
8075   \xmakefirstuc\@glo@text
8076 }
```

Define commands to determine if the prefix keys have been set:

`\ifglshasprefix`

```
8077 \newcommand*{\ifglshasprefix}[3]{%
8078   \ifcseempty{glo@#1@prefix}%
8079   {#3}%
8080   {#2}%
8081 }
```

`hasprefixplural`

```
8082 \newcommand*{\ifglshasprefixplural}[3]{%
8083   \ifcseempty{glo@#1@prefixplural}%
8084   {#3}%
8085   {#2}%
8086 }
```

`shasprefixfirst`

```
8087 \newcommand*{\ifglshasprefixfirst}[3]{%
8088   \ifcseempty{glo@#1@prefixfirst}%
8089   {#3}%
8090   {#2}%
8091 }
```

`efixfirstplural`

```
8092 \newcommand*{\ifglshasprefixfirstplural}[3]{%
8093   \ifcseempty{glo@#1@prefixfirstplural}%
8094   {#3}%
8095   {#2}%
8096 }
```

Define commands that insert the prefix before commands like `\gls`:

`\pgls`

```
8097 \newrobustcmd{\pgls}{\@gls@hyp@opt\@pgls}
```

`\@pgls` Unstarred version.

```
8098 \newcommand*{\@pgls}[2][ ]{%
8099   \new@ifnextchar[%
8100   {\@pgls@{#1}{#2}}%
8101   {\@pgls@{#1}{#2}[ ]}%
8102 }
```

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

```
8103 \def\@pgls@#1#2[#3]{%
8104   \glsdoifexists{#2}%
8105   {%
8106     \ifglsused{#2}%
8107     {%
8108       \glsentryprefix{#2}%
8109     }%

```

```

8110   {%
8111     \glsentryprefixfirst{#2}%
8112   }%
8113   \@gls@{#1}{#2}[#3]%
8114 }%
8115 }

```

Similarly for the plural version:

```

\pglsp1
8116 \newrobustcmd{\pglsp1}{\@gls@hyp@opt\@pglsp1}

```

\@pglsp1 Unstarred version.

```

8117 \newcommand*{\@pglsp1}[2][ ]{%
8118   \new@ifnextchar[%
8119     {\@pglsp1@{#1}{#2}}%
8120     {\@pglsp1@{#1}{#2}[ ]}%
8121 }

```

\@pglsp1@ Read in the final optional argument:

```

8122 \def\@pglsp1@#1#2[#3]{%
8123   \glsdoifexists{#2}%
8124   {%
8125     \ifglsused{#2}%
8126     {%
8127       \glsentryprefixplural{#2}%
8128     }%
8129     {%
8130       \glsentryprefixfirstplural{#2}%
8131     }%
8132     \@glspl@{#1}{#2}[#3]%
8133   }%
8134 }

```

Now for the first letter upper case versions:

```

\Pgls
8135 \newrobustcmd{\Pgls}{\@gls@hyp@opt\@Pgls}

```

\@Pgls Unstarred version.

```

8136 \newcommand*{\@Pgls}[2][ ]{%
8137   \new@ifnextchar[%
8138     {\@Pgls@{#1}{#2}}%
8139     {\@Pgls@{#1}{#2}[ ]}%
8140 }

```

\@Pgls@ Read in the final optional argument:

```

8141 \def\@Pgls@#1#2[#3]{%

```

```

8142 \glsdoifexists{#2}%
8143 {%
8144   \ifglsused{#2}%
8145   {%
8146     \ifglsasprefix{#2}%
8147     {%
8148       \Glsentryprefix{#2}%
8149       \@gls@{#1}{#2}[#3]%
8150     }%
8151     {\@Gls@{#1}{#2}[#3]}%
8152   }%
8153   {%
8154     \ifglsasprefixfirst{#2}%
8155     {%
8156       \Glsentryprefixfirst{#2}%
8157       \@gls@{#1}{#2}[#3]%
8158     }%
8159     {\@Gls@{#1}{#2}[#3]}%
8160   }%
8161 }%
8162 }

```

Similarly for the plural version:

`\Pglsp1`

```
8163 \newrobustcmd{\Pglsp1}{\@gls@hyp@opt\@Pglsp1}
```

`\@Pglsp1` Unstarred version.

```

8164 \newcommand*{\@Pglsp1}[2] [] {%
8165   \new@ifnextchar [%
8166   {\@Pglsp1@{#1}{#2}}%
8167   {\@Pglsp1@{#1}{#2} []}%
8168 }

```

`\@Pglsp1@` Read in the final optional argument:

```

8169 \def\@Pglsp1@#1#2[#3] {%
8170   \glsdoifexists{#2}%
8171   {%
8172     \ifglsused{#2}%
8173     {%
8174       \ifglsasprefixplural{#2}%
8175       {%
8176         \Glsentryprefixplural{#2}%
8177         \@glspl@{#1}{#2}[#3]%
8178       }%
8179       {\@Glspl@{#1}{#2}[#3]}%
8180     }%
8181     {%
8182       \ifglsasprefixfirstplural{#2}%

```

```

8183     {%
8184         \Glsentryprefixfirstplural{#2}%
8185         \@glspl@{#1}{#2}[#3]%
8186     }%
8187     {\@Glspl@{#1}{#2}[#3]}%
8188 }%
8189 }%
8190 }

```

Finally the all upper case versions:

\PGLS

```
8191 \newrobustcmd{\PGLS}{\@gls@hyp@opt\PGLS}
```

\@PGLS Unstarred version.

```

8192 \newcommand*{\@PGLS}[2][ ]{%
8193     \new@ifnextchar[%
8194     {\@PGLS@{#1}{#2}}%
8195     {\@PGLS@{#1}{#2}[ ]}%
8196 }

```

\@PGLS@ Read in the final optional argument:

```

8197 \def\@PGLS@#1#2[#3]{%
8198     \glsdoifexists{#2}%
8199     {%
8200         \ifglsused{#2}%
8201         {%
8202             \mfirstucMakeUppercase{\glsentryprefix{#2}}%
8203         }%
8204         {%
8205             \mfirstucMakeUppercase{\glsentryprefixfirst{#2}}%
8206         }%
8207         \@GLS@{#1}{#2}[#3]%
8208     }%
8209 }

```

Plural version:

\PGLSp1

```
8210 \newrobustcmd{\PGLSp1}{\@gls@hyp@opt\PGLSp1}
```

\@PGLSp1 Unstarred version.

```

8211 \newcommand*{\@PGLSp1}[2][ ]{%
8212     \new@ifnextchar[%
8213     {\@PGLSp1@{#1}{#2}}%
8214     {\@PGLSp1@{#1}{#2}[ ]}%
8215 }

```

\@PGLSp1@ Read in the final optional argument:

```
8216 \def\@PGLSp1@#1#2[#3]{%
8217   \glsdoifexists{#2}%
8218   {%
8219     \ifglsused{#2}%
8220     {%
8221       \mfirstucMakeUppercase{\glsentryprefixplural{#2}}%
8222     }%
8223     {%
8224       \mfirstucMakeUppercase{\glsentryprefixfirstplural{#2}}%
8225     }%
8226     \@GLSp1@{#1}{#2}[#3]%
8227   }%
8228 }
```

3 Glossary Styles

3.1 Glossary hyper-navigation definitions (glossary-hypernav package)

Package Definition:

```
8229 \ProvidesPackage{glossary-hypernav}[2019/01/06 v4.42 (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`

```
8230 \newcommand*{\glsnavhyperlink}[3][\@glo@type]{%
8231   \edef\gls@grplabel{#2}\protected@edef\gls@grptitle{#3}%
8232   \@glslink{\glsnavhyperlinkname{#1}{#2}}{#3}}
```

`navhyperlinkname` Expands to the hypertarget name. The first argument is the glossary type. The second argument is the group label.

```
8233 \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.

`navhypertarget`

```
8234 \newcommand*{\glsnavhypertarget}[3][\@glo@type]{%
8235   \@glsnavhypertarget{#1}{#2}{#3}%
8236 }
```

The actual code is now in an internal command that doesn't have an optional argument, which makes it easier to save and restore the original behaviour.

`navhypertarget`

```
8237 \newcommand*{\@glsnavhypertarget}[3]{%
```

Add this group to the aux file for re-run check.

```
8238 \protected@write\auxout-{}{\string\@gls@hypergroup{#1}{#2}}%
```

Add the target.

```
8239 \@gls@target{\glsnavhyperlinkname{#1}{#2}}{#3}%
```

Check list of known groups to determine if a re-run is required.

```
8240 \expandafter\let
```

```
8241 \expandafter\@gls@list\csname @gls@hypergroup@list@#1\endcsname
```

Iterate through list and terminate loop if this group is found.

```
8242 \@for\@gls@elem:=\@gls@list\do{%
```

```
8243 \ifthenelse{\equal{\@gls@elem}{#2}}{\@endfortrue}{}}%
```

Check if list terminated prematurely.

```
8244 \if@endfor
```

```
8245 \else
```

This group was not included in the list, so issue a warning.

```
8246 \GlossariesWarningNoLine{Navigation panel
```

```
8247 for glossary type ‘#1’^^Jmissing group ‘#2’}%
```

```
8248 \gdef\gls@hypergroup@rerun{%
```

```
8249 \GlossariesWarningNoLine{Navigation panel
```

```
8250 has changed. Rerun LaTeX}}%
```

```
8251 \fi
```

```
8252 }
```

`hypergroup@rerun` Give a warning at the end if re-run required

```
8253 \let\gls@hypergroup@rerun\relax
```

```
8254 \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.

```
8255 \newcommand*{\@gls@hypergroup}[2]{%
```

```
8256 \@ifundefined{@gls@hypergroup@list@#1}{%
```

```
8257 \expandafter\xdef\csname @gls@hypergroup@list@#1\endcsname{#2}}%
```

```
8258 }{%
```

```
8259 \expandafter\let\expandafter\@gls@tmp
```

```
8260 \csname @gls@hypergroup@list@#1\endcsname
```

```
8261 \expandafter\xdef\csname @gls@hypergroup@list@#1\endcsname{%
```

```
8262 \@gls@tmp,#2}}%
```

```
8263 }%
```

```
8264 }
```

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. (In earlier versions 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. Version 1.14 changed this to only use labels for groups that are present.) Now for the whole navigation bit:

`\glsnavigation`

```
8265 \newcommand*{\glsnavigation}{%
8266   \def\@gls@between{}%
8267   \ifcsundef{@gls@hypergrouplist@\@glo@type}%
8268     {%
8269       \def\@gls@list{}%
8270     }%
8271     {%
8272       \expandafter\let\expandafter\@gls@list
8273       \csname @gls@hypergrouplist@\@glo@type\endcsname
8274     }%
8275     \@for\@gls@tmp:=\@gls@list\do{%
8276       \@gls@between

8277       \@gls@getgrouptitle{\@gls@tmp}{\@gls@grptitle}%
8278       \glsnavhyperlink{\@gls@tmp}{\@gls@grptitle}%
8279       \let\@gls@between\glshypernavsep
8280     }%
8281 }
```

`\glshypernavsep` Separator for the hyper navigation bar.

```
8282 \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`

```
8283 \newcommand*{\glssymbolnav}{%
8284   \glsnavhyperlink{glssymbols}{\glsgetgrouptitle{glssymbols}}%
8285   \glshypernavsep
8286   \glsnavhyperlink{glsnumbers}{\glsgetgrouptitle{glsnumbers}}%
8287   \glshypernavsep
8288 }
```

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.

```
8289 \ProvidesPackage{glossary-inline}[2019/01/06 v4.42 (NLCT)]
```

`inline` Define the inline style.

```
8290 \newglossarystyle{inline}{%
```

Start of glossary sets up first empty separator between entries. (This is then changed by `\glossentry`)

```
8291   \renewenvironment{theglossary}%
8292     {%
```

```

8293     \def\gls@inlinesep{}%
8294     \def\gls@inlinesubsep{}%
8295     \def\gls@inlinepostchild{}%
8296     }%
8297     {\glspostinline}%

```

No header:

```
8298 \renewcommand*\glossaryheader{}%
```

No group headings (if heading is required, add `\glsinlinedopostchild` to start definition in case heading follows a child entry):

```
8299 \renewcommand*\glsgroupheading}[1]{}%
```

Just display separator followed by name and description:

```

8300 \renewcommand{\glossentry}[2]{%
8301   \glsinlinedopostchild
8302   \gls@inlinesep
8303   \glsentryitem{##1}%
8304   \glsinlinenameformat{##1}{%
8305     \glossentryname{##1}%
8306   }%
8307   \ifglsdescsuppressed{##1}%
8308   {%
8309     \glsinlineemptydescformat
8310     {%
8311       \glossentrysymbol{##1}%
8312     }%
8313     {%
8314       ##2%
8315     }%
8316   }%
8317   {%
8318     \ifglshasdesc{##1}%
8319     {\glsinlinedescformat{\glossentrydesc{##1}}{\glossentrysymbol{##1}}{##2}}%
8320     {\glsinlineemptydescformat{\glossentrysymbol{##1}}{##2}}%
8321   }%
8322   \ifglshaschildren{##1}%
8323   {%
8324     \glsresetsubentrycounter
8325     \glsinlineparentchildseparator
8326     \def\gls@inlinesubsep{}%
8327     \def\gls@inlinepostchild{\glsinlinepostchild}%
8328   }%
8329   }%
8330   \def\gls@inlinesep{\glsinlineseparator}%
8331 }%

```

Sub-entries display description:

```

8332 \renewcommand{\subglossentry}[3]{%
8333   \gls@inlinesubsep%
8334   \glsinlinesubnameformat{##2}{%

```

```

8335     \glossentryname{##2}}%
8336     \glsentryitem{##2}%
8337     \glsinlinesubdescformat{\glossentrydesc{##2}}{\glossentrysymbol{##2}}{##3}%
8338     \def\gls@inlinesubsep{\glsinlinesubseparator}%
8339 }%

```

Nothing special between groups:

```

8340 \renewcommand*\glsgroupskip{}%
8341 }

```

linedopostchild

```

8342 \newcommand*\glsinlinedopostchild{%
8343     \gls@inlinepostchild
8344     \def\gls@inlinepostchild{}%
8345 }

```

inlineseparator Separator to use between entries.

```

8346 \newcommand*\glsinlineseparator{;\space}

```

inlinesubseparator Separator to use between sub-entries.

```

8347 \newcommand*\glsinlinesubseparator{,\space}

```

parentchildseparator Separator to use between parent and children.

```

8348 \newcommand*\glsinlineparentchildseparator{: \space}

```

inlinepostchild Hook to use between child and next entry

```

8349 \newcommand*\glsinlinepostchild{}

```

\glspostinline Terminator for inline glossary.

```

8350 \newcommand*\glspostinline{\glspostdescription\space}

```

inlinenameformat Formats the name of the entry (first argument label, second argument name):

```

8351 \newcommand*\glsinlinenameformat}[2]{\glstarget{#1}{#2}}

```

inlinedescformat Formats the entry's description, symbol and location list:

```

8352 \newcommand*\glsinlinedescformat}[3]{\space#1}

```

emptydescformat Formats the entry's symbol and location list when the description is empty:

```

8353 \newcommand*\glsinlineemptydescformat}[2]{}

```

inlinesubnameformat Formats the name of the subentry (first argument label, second argument name):

```

8354 \newcommand*\glsinlinesubnameformat}[2]{\glstarget{#1}{}}

```

inlinesubdescformat Formats the subentry's description, symbol and location list:

```

8355 \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.

```
8356 \ProvidesPackage{glossary-list}[2019/01/06 v4.42 (NLCT)]
```

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```
8357 \providecommand{\indexspace}{%
8358   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
8359 }
```

`tgroupheaderfmt` Provide a way of adjusting the format of the group headings.

```
8360 \newcommand*{\glslistgroupheaderfmt}[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.

```
8361 \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.

```
8362 \newglossarystyle{list}{%
```

Use description environment:

```
8363 \renewenvironment{theglossary}%
8364   {\begin{description}}{\end{description}}%
```

No header at the start of the environment:

```
8365 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8366 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries start a new item in the list:

```
8367 \renewcommand*{\glossentry}[2]{%
8368   \item[\glsentryitem{##1}]%
8369     \glstarget{##1}{\glossentryname{##1}}]
8370   \glossentrydesc{##1}\glspostdescription\space ##2}%
```

Sub-entries continue on the same line:

```
8371 \renewcommand*{\subglossentry}[3]{%
8372   \glssubentryitem{##2}%
```

```

8373   \glstarget{##2}{\strut}\space
8374   \glossentrydesc{##2}\glspostdescription\space ##3.}%
      Add vertical space between groups:
8375   \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
8376 }

```

listgroup The listgroup style is like the list style, but the glossary groups have headings.

```

8377 \newglossarystyle{listgroup}{%
      Base it on the list style:
8378   \setglossarystyle{list}%
      Each group has a heading:
8379   \renewcommand*{\glsgroupheading}[1]{%
8380     \item[\glslistgroupheaderfmt{\glsgetgrouptitle{##1}}]}

```

listhypergroup The listhypergroup style is like the listgroup style, but has a set of links to the groups at the start of the glossary.

```

8381 \newglossarystyle{listhypergroup}{%
      Base it on the list style:
8382   \setglossarystyle{list}%
      Add navigation links at the start of the environment.
8383   \renewcommand*{\glossaryheader}{%
8384     \glslistnavigationitem{\glsnavigation}}%
      Each group has a heading with a hypertext:
8385   \renewcommand*{\glsgroupheading}[1]{%
8386     \item[\glslistgroupheaderfmt
8387           {\glsnavhypertext{##1}{\glsgetgrouptitle{##1}}]}

```

altlist The altlist glossary style is like the list style, but places the description on a new line. Sub-entries follow in separate paragraphs without the sub-entry name. This style does not use the entry's symbol.

```

8388 \newglossarystyle{altlist}{%
      Base it on the list style:
8389   \setglossarystyle{list}%
      Main (level 0) entries start a new item in the list with a line break after the entry name:
8390   \renewcommand*{\glossentry}[2]{%
8391     \item[\glsentryitem{##1}%
8392           \glstarget{##1}{\glossentryname{##1}}]}

```

Version 3.04 changed `\newline` to the following paragraph break stuff (thanks to Daniel Gebhardt for supplying the fix) to prevent a page break occurring at this point.

```

8393     \mbox{} \par \nobreak \@afterheading
8394     \glossentrydesc{##1}\glspostdescription\space ##2}%

```

Sub-entries start a new paragraph:

```
8395 \renewcommand{\subglossentry}[3]{%
8396   \par
8397   \glssubentryitem{##2}%
8398   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space ##3}%
8399 }
```

`altlistgroup` The `altlistgroup` glossary style is like the `altlist` style, but the glossary groups have headings.

```
8400 \newglossarystyle{altlistgroup}{%
      Base it on the altlist style:
8401   \setglossarystyle{altlist}%
      Each group has a heading:
8402   \renewcommand*\glsgroupheading[1]{%
8403     \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.

```
8404 \newglossarystyle{altlisthypergroup}{%
      Base it on the altlist style:
8405   \setglossarystyle{altlist}%
      Add navigation links at the start of the environment.
8406   \renewcommand*\glossaryheader{%
8407     \glslistnavigationitem{\glslnavigation}}%
      Each group has a heading with a hypertarget:
8408   \renewcommand*\glsgroupheading[1]{%
8409     \item[\glslistgroupheaderfmt
8410       {\glslnavhypertarget{##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.

```
8411 \newglossarystyle{listdotted}{%
      Base it on the list style:
8412   \setglossarystyle{list}%
      Each main (level 0) entry starts a new item:
8413   \renewcommand*\glossentry[2]{%
8414     \item[]\makebox[\glslistdottedwidth][l]{%
8415       \glssentryitem{##1}%
8416       \glstarget{##1}{\glossentryname{##1}}%
8417       \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##1}}%
```

Sub entries have the same format as main entries:

```
8418 \renewcommand*{\subglossentry}[3]{%
8419   \item[\makebox[\glslistdottedwidth][l]{%
8420     \glssubentryitem{##2}}%
8421   \glstarget{##2}{\glossentryname{##2}}%
8422   \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##2}}%
8423 }
```

listdottedwidth

```
8424 \newlength\glslistdottedwidth
8425 \setlength{\glslistdottedwidth}{.5\hsize}
```

sublistdotted This style is similar to the `glostylelistdotted` style, except that the main entries just have the name displayed.

```
8426 \newglossarystyle{sublistdotted}{%
```

Base it on the `listdotted` style:

```
8427 \setglossarystyle{listdotted}%
```

Main (level 0) entries just display the name:

```
8428 \renewcommand*{\glossentry}[2]{%
8429   \item[\glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}}}%
8430 }
```

3.4 Glossary Styles using `longtable` (the `glossary-long` package)

The glossary styles defined in the package used the `longtable` environment in the glossary.

```
8431 \ProvidesPackage{glossary-long}[2019/01/06 v4.42 (NLCT)]
```

Requires the package:

```
8432 \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`.)

```
8433 \@ifundefined{glsdescwidth}{%
8434   \newlength\glsdescwidth
8435   \setlength{\glsdescwidth}{0.6\hsize}
8436 }{}
```

`\glspagelistwidth` This is a length that governs the width of the page list column.

```
8437 \@ifundefined{glspagelistwidth}{%
8438   \newlength\glspagelistwidth
8439   \setlength{\glspagelistwidth}{0.1\hsize}
8440 }{}
```

`long` The `long` glossary style command which uses the `longtable` environment:

```
8441 \newglossarystyle{long}{%
```

Use `longtable` with two columns:

```
8442 \renewenvironment{theglossary}{%
8443     {\begin{longtable}{lp{\glsdescwidth}}}%
8444     {\end{longtable}}%
```

Do nothing at the start of the environment:

```
8445 \renewcommand*{\glossaryheader}{}%
```

No heading between groups:

```
8446 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries displayed in a row:

```
8447 \renewcommand{\glossentry}[2]{%
8448     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8449     \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline
8450 }%
```

Sub entries displayed on the following row without the name:

```
8451 \renewcommand{\subglossentry}[3]{%
8452     &
8453     \glssubentryitem{##2}%
8454     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
8455     ##3\tabularnewline
8456 }%
```

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>)

```
8457 \ifglsnogroupskip
8458     \renewcommand*{\glsgroupskip}{}%
8459 \else
8460     \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
8461 \fi
8462 }
```

`longborder` The `longborder` style is like the above, but with horizontal and vertical lines:

```
8463 \newglossarystyle{longborder}{%
```

Base it on the `glostylelong` style:

```
8464 \setglossarystyle{long}%
```

Use `longtable` with two columns with vertical lines between each column:

```
8465 \renewenvironment{theglossary}{%
8466     \begin{longtable}{|lp{\glsdescwidth}|}{\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
8467 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8468 }
```

`longheader` The `longheader` style is like the `long` style but with a header:

```
8469 \newglossarystyle{longheader}{%
```


Base it on the `glostylelong` style:

```
8470 \setglossarystyle{long}%
```

Set the table's header:

```
8471 \renewcommand*{\glossaryheader}{%
8472   \bfseries \entryname & \bfseries \descriptionname\tabularnewline\endhead}%
8473 }
```

`longheaderborder` The `longheaderborder` style is like the `long` style but with a header and border:

```
8474 \newglossarystyle{longheaderborder}{%
```

Base it on the `glostylelongborder` style:

```
8475 \setglossarystyle{longborder}%
```

Set the table's header and add horizontal line to table's foot:

```
8476 \renewcommand*{\glossaryheader}{%
8477   \hline\bfseries \entryname & \bfseries
8478   \descriptionname\tabularnewline\hline
8479   \endhead
8480   \hline\endfoot}%
8481 }
```

`long3col` The `long3col` style is like `long` but with 3 columns

```
8482 \newglossarystyle{long3col}{%
```

Use a `longtable` with 3 columns:

```
8483 \renewenvironment{theglossary}%
8484   {\begin{longtable}[lp{\glwidth}p{\glwidth}]}%
8485   {\end{longtable}}%
```

No table header:

```
8486 \renewcommand*{\glossaryheader}{}%
```

No headings between groups:

```
8487 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
8488 \renewcommand{\glossentry}[2]{%
8489   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8490   \glossentrydesc{##1} & ##2\tabularnewline
8491   }%
```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```
8492 \renewcommand{\subglossentry}[3]{%
8493   &
8494   \glssubentryitem{##2}%
8495   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8496   ##3\tabularnewline
8497   }%
```

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>)

```
8498 \ifglsnogroupskip
8499 \renewcommand*\glsgroupskip}{}%
8500 \else
8501 \renewcommand*\glsgroupskip}{ & & \tabularnewline}%
8502 \fi
8503 }
```

`long3colborder` The `long3colborder` style is like the `long3col` style but with a border:

```
8504 \newglossarystyle{long3colborder}{%
  Base it on the glostylelong3col style:
8505 \setglossarystyle{long3col}%
  Use a longtable with 3 columns with vertical lines around them:
8506 \renewenvironment{theglossary}{%
8507   {\begin{longtable}{|l|p{\glsdescwidth}|p{\glspagelistwidth}|}}%
8508   {\end{longtable}}%
  Place horizontal lines at the head and foot of the table:
8509 \renewcommand*\glossaryheader{\hline\endhead\hline\endfoot}%
8510 }
```

`long3colheader` The `long3colheader` style is like `long3col` but with a header row:

```
8511 \newglossarystyle{long3colheader}{%
  Base it on the glostylelong3col style:
8512 \setglossarystyle{long3col}%
  Set the table's header:
8513 \renewcommand*\glossaryheader}{%
8514   \bfseries\entryname&\bfseries\descriptionname&
8515   \bfseries\pagelistname\tabularnewline\endhead}%
8516 }
```

`colheaderborder` The `long3colheaderborder` style is like the above but with a border

```
8517 \newglossarystyle{long3colheaderborder}{%
  Base it on the glostylelong3colborder style:
8518 \setglossarystyle{long3colborder}%
  Set the table's header and add horizontal line at table's foot:
8519 \renewcommand*\glossaryheader}{%
8520   \hline
8521   \bfseries\entryname&\bfseries\descriptionname&
8522   \bfseries\pagelistname\tabularnewline\hline\endhead
8523   \hline\endfoot}%
8524 }
```

`long4col` The `long4col` style has four columns where the third column contains the value of the associated symbol key.

```
8525 \newglossarystyle{long4col}{%
```

Use a `longtable` with 4 columns:

```
8526 \renewenvironment{theglossary}{%
```

```
8527   {\begin{longtable}{l111}}%
```

```
8528   {\end{longtable}}%
```

No table header:

```
8529 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8530 \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):

```
8531 \renewcommand{\glossentry}[2]{%
```

```
8532   \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
```

```
8533   \glossentrydesc{##1} &
```

```
8534   \glossentrysymbol{##1} &
```

```
8535   ##2\tabularnewline
```

```
8536 }%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
8537 \renewcommand{\subglossentry}[3]{%
```

```
8538   &
```

```
8539   \glssubentryitem{##2}%
```

```
8540   \glstarget{##2}{\strut}\glossentrydesc{##2} &
```

```
8541   \glossentrysymbol{##2} & ##3\tabularnewline
```

```
8542 }%
```

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>)

```
8543 \ifglsnogroupskip
```

```
8544   \renewcommand*{\glsgroupskip}{}%
```

```
8545 \else
```

```
8546   \renewcommand*{\glsgroupskip}{ & & & \tabularnewline}%
```

```
8547 \fi
```

```
8548 }
```

`long4colheader` The `long4colheader` style is like `long4col` but with a header row.

```
8549 \newglossarystyle{long4colheader}{%
```

Base it on the `glostylelong4col` style:

```
8550 \setglossarystyle{long4col}%
```

Table has a header:

```
8551 \renewcommand*{\glossaryheader}{%
```

```
8552   \bfseries\entryname&\bfseries\descriptionname&
```

```
8553   \bfseries \symbolname&
```

```

8554     \bfseries\pagelistname\tabularnewline\endhead}%
8555 }

```

`long4colborder` The `long4colborder` style is like `long4col` but with a border.

```

8556 \newglossarystyle{long4colborder}{%

```

Base it on the `glostylelong4col` style:

```

8557 \setglossarystyle{long4col}%

```

Use a `longtable` with 4 columns surrounded by vertical lines:

```

8558 \renewenvironment{theglossary}%
8559     {\begin{longtable}{|l|l|l|l|}}%
8560     {\end{longtable}}%

```

Add horizontal lines to the head and foot of the table:

```

8561 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8562 }

```

`colheaderborder` The `long4colheaderborder` style is like the above but with a border.

```

8563 \newglossarystyle{long4colheaderborder}{%

```

Base it on the `glostylelong4col` style:

```

8564 \setglossarystyle{long4col}%

```

Use a `longtable` with 4 columns surrounded by vertical lines:

```

8565 \renewenvironment{theglossary}%
8566     {\begin{longtable}{|l|l|l|l|}}%
8567     {\end{longtable}}%

```

Add table header and horizontal line at the table's foot:

```

8568 \renewcommand*{\glossaryheader}{%
8569     \hline\bfseries\entryname&\bfseries\descriptionname&
8570     \bfseries \symbolname&
8571     \bfseries\pagelistname\tabularnewline\hline\endhead
8572     \hline\endfoot}%
8573 }

```

`altlong4col` The `altlong4col` style is like the `long4col` style but can have multiline descriptions and page lists.

```

8574 \newglossarystyle{altlong4col}{%

```

Base it on the `glostylelong4col` style:

```

8575 \setglossarystyle{long4col}%

```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```

8576 \renewenvironment{theglossary}%
8577     {\begin{longtable}{lp{\glsdescwidth}lp{\glspagelistwidth}}}%
8578     {\end{longtable}}%
8579 }

```

`altlong4colheader` The `altlong4colheader` style is like `altlong4col` but with a header row.

```
8580 \newglossarystyle{altlong4colheader}{%
      Base it on the glostylelong4colheader style:
8581   \setglossarystyle{long4colheader}%
      Use a longtable with 4 columns where the second and last columns may have multiple lines
      in each row:
8582   \renewenvironment{theglossary}%
8583     {\begin{longtable}{lp{\glsdescwidth}lp{\glspagelistwidth}}}%
8584     {\end{longtable}}%
8585 }
```

`altlong4colborder` The `altlong4colborder` style is like `altlong4col` but with a border.

```
8586 \newglossarystyle{altlong4colborder}{%
      Base it on the glostylelong4colborder style:
8587   \setglossarystyle{long4colborder}%
      Use a longtable with 4 columns where the second and last columns may have multiple lines
      in each row:
8588   \renewenvironment{theglossary}%
8589     {\begin{longtable}{|lp{\glsdescwidth}|lp{\glspagelistwidth}|}}%
8590     {\end{longtable}}%
8591 }
```

`altlong4colheaderborder` The `altlong4colheaderborder` style is like the above but with a header as well as a border.

```
8592 \newglossarystyle{altlong4colheaderborder}{%
      Base it on the glostylelong4colheaderborder style:
8593   \setglossarystyle{long4colheaderborder}%
      Use a longtable with 4 columns where the second and last columns may have multiple lines
      in each row:
8594   \renewenvironment{theglossary}%
8595     {\begin{longtable}{|lp{\glsdescwidth}|lp{\glspagelistwidth}|}}%
8596     {\end{longtable}}%
8597 }
```

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>

```
8598 \ProvidesPackage{glossary-longbooktabs}[2019/01/06 v4.42 (NLCT)]
```

Requires `booktabs` package:

```
8599 \RequirePackage{booktabs}
```

and the base packages for long styles:

```
8600 \RequirePackage{glossary-long}
8601 \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.

```
8602 \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.

```
8603 \glspatchLToutput
```

As with the longheader style, use the long style as a base.

```
8604 \setglossarystyle{long}{%
```

Add a header with rules.

```
8605 \renewcommand*{\glossaryheader}{%
8606 \toprule \bfseries \entryname & \bfseries
8607 \descriptionname\tabularnewline\midrule\endhead
8608 \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.

```
8609 \ifglsgnogroupskip
8610 \renewcommand*{\glsgroupskip}{}%
8611 \else
8612 \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
8613 \fi
8614 }
```

long3col-booktabs The long3col-booktabs style is similar to the long3colheader style but uses the booktabs rules and patches longtable to check for group skip occurring at a page break.

```
8615 \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.

```
8616 \glspatchLToutput
```

Use the long3col style as a base.

```
8617 \setglossarystyle{long3col}{%
```

Add a header with rules.

```
8618 \renewcommand*{\glossaryheader}{%
8619 \toprule \bfseries \entryname &
8620 \bfseries \descriptionname &
8621 \bfseries \pagelistname
8622 \tabularnewline\midrule\endhead
8623 \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.

```
8624 \ifglsgroupskip
8625   \renewcommand*{\glsgroupskip}{}%
8626 \else
8627   \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
8628 \fi
8629 }
```

`long4col-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.

```
8630 \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.

```
8631   \glspatchLToutput
```

Use the `long4col` style as a base.

```
8632   \setglossarystyle{long4col}{%
```

Add a header with rules.

```
8633   \renewcommand*{\glossaryheader}{%
8634     \toprule \bfseries \entryname &
8635     \bfseries \descriptionname &
8636     \bfseries \symbolname &
8637     \bfseries \pagelistname
8638     \tabularnewline\midrule\endhead
8639     \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.

```
8640   \ifglsgroupskip
8641     \renewcommand*{\glsgroupskip}{}%
8642   \else
8643     \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
8644   \fi
8645 }
```

`altlong4col-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.

```
8646 \newglossarystyle{altlong4col-booktabs}{%
```

The patch `\glspatchLToutput` is already applied in `long4col-booktabs` and so doesn't need to be here.

```
8647   \glspatchLToutput
```

Use the `long4col-booktabs` style as a base.

```
8648   \setglossarystyle{long4col-booktabs}{%
```

Change the column specifications:

```
8649 \renewenvironment{theglossary}%  
8650   {\begin{longtable}{lp{\glsdescwidth}lp{\glspagelistwidth}}}%  
8651   {\end{longtable}}%  
8652 }
```

Ragged styles.

ragged-booktabs The longragged-booktabs style is similar to the longragged style but uses the booktabs rules and patches longtable to check for group skip occurring at a page break.

```
8653 \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.

```
8654 \glspatchLToutput
```

Use the long-booktabs style as a base.

```
8655 \setglossarystyle{long-booktabs}%
```

Adjust the column specification.

```
8656 \renewenvironment{theglossary}%  
8657   {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}}}%  
8658   {\end{longtable}}%  
8659 }
```

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.

```
8660 \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.

```
8661 \glspatchLToutput
```

Use the long3col-booktabs style as a base.

```
8662 \setglossarystyle{long3col-booktabs}%
```

Adjust the column specification.

```
8663 \renewenvironment{theglossary}%  
8664   {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}%  
8665     >{\raggedright}p{\glspagelistwidth}}}%  
8666   {\end{longtable}}%  
8667 }
```

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.

```
8668 \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.

```
8669 \glspatchLToutput
```


Use the `altlong4col-booktabs` style as a base.

```
8670 \setglossarystyle{altlong4col-booktabs}%
```

Adjust the column specification.

```
8671 \renewenvironment{theglossary}%  
8672   {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}l%  
8673     >{\raggedright}p{\glspagelistwidth}}}%  
8674   {\end{longtable}}%  
8675 }
```

`sLTpenaltycheck`

```
8676 \newcommand*{\glsLTpenaltycheck}{%  
8677   \ifnum\outputpenalty=-50\vskip-\normalbaselineskip\relax\fi  
8678 }
```

`enaltygroupskip`

```
8679 \newcommand{\glspenaltygroupskip}{%  
8680   \noalign{\penalty-50\vskip\normalbaselineskip}}
```

`restoreLToutput` Provide a way of restoring `\LT@output` for the user.

```
8681 \let\@gls@org@LT@output\LT@output  
8682 \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`

```
8683 \newcommand*{\glspatchLToutput}{%  
8684   \renewcommand*{\LT@output}{%  
8685     \ifnum\outputpenalty <-\@Mi  
8686       \ifnum\outputpenalty > -\LT@end@pen  
8687         \LT@err{floats and marginpars not allowed in a longtable}\@ehc  
8688       \else  
8689         \setbox\z@\vbox{\unvbox\@cclv}%  
8690         \ifdim \ht\LT@lastfoot>\ht\LT@foot  
8691           \dimen@\pagegoal  
8692           \advance\dimen@-\ht\LT@lastfoot  
8693           \ifdim\dimen@<\ht\z@  
8694             \setbox\@cclv\vbox{\unvbox\z@\copy\LT@foot\vss}%  
8695             \@makecol  
8696             \@outputpage  
8697             \setbox\z@\vbox{\box\LT@head\glsLTpenaltycheck}%  
8698           \fi  
8699         \fi  
8700         \global\@colroom\@colht  
8701         \global\vsiz@\@colht  
8702         {\unvbox\z@\box\ifvoid\LT@lastfoot\LT@foot\else\LT@lastfoot\fi}%  
8703       \fi  
8704     \else
```

```

8705     \setbox\@cclv\vbox{\unvbox\@cclv\copy\LT@foot\vss}%
8706     \@makecol
8707     \@outputpage
8708     \global\ysize\@colroom
8709     \copy\LT@head
8710     \glsLTpenaltycheck
8711     \nobreak
8712     \fi
8713 }%
8714 }

```

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.

```
8715 \ProvidesPackage{glossary-longragged}[2019/01/06 v4.42 (NLCT)]
```

Requires the package:

```
8716 \RequirePackage{array}
```

Requires the package:

```
8717 \RequirePackage{longtable}
```

`\glsdescwidth` This is a length that governs the width of the description column. This may have already been defined.

```

8718 \@ifundefined{glsdescwidth}{%
8719   \newlength\glsdescwidth
8720   \setlength{\glsdescwidth}{0.6\hsize}
8721 }{}

```

`glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```

8722 \@ifundefined{glspagelistwidth}{%
8723   \newlength\glspagelistwidth
8724   \setlength{\glspagelistwidth}{0.1\hsize}
8725 }{}

```

`longragged` The longragged glossary style is like the long but uses ragged right formatting for the description column.

```
8726 \newglossarystyle{longragged}{%
```

Use longtable with two columns:

```

8727   \renewenvironment{theglossary}%
8728     {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}}}%
8729     {\end{longtable}}%

```

Do nothing at the start of the environment:

```
8730   \renewcommand*{\glossaryheader}{}%
```

No heading between groups:

```
8731 \renewcommand*\glsgroupheading}[1]{}
```

Main (level 0) entries displayed in a row:

```
8732 \renewcommand{\glossentry}[2]{%
8733   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8734   \glossentrydesc{##1}\glspostdescription\space ##2%
8735   \tabularnewline
8736 }%
```

Sub entries displayed on the following row without the name:

```
8737 \renewcommand{\subglossentry}[3]{%
8738   &
8739   \glssubentryitem{##2}%
8740   \glstarget{##2}{\strut}\glossentrydesc{##2}%
8741   \glspostdescription\space ##3%
8742   \tabularnewline
8743 }%
```

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>)

```
8744 \ifglsnogroupskip
8745   \renewcommand*\glsgroupskip}{%
8746   \else
8747     \renewcommand*\glsgroupskip}{ & \tabularnewline}%
8748   \fi
8749 }
```

`longraggedborder` The `longraggedborder` style is like the above, but with horizontal and vertical lines:

```
8750 \newglossarystyle{longraggedborder}{%
```

Base it on the `glostylelongragged` style:

```
8751 \setglossarystyle{longragged}%
```

Use `longtable` with two columns with vertical lines between each column:

```
8752 \renewenvironment{theglossary}{%
8753   \begin{longtable}{|l|>{\raggedright}p{\glsdescwidth}|}%
8754   {\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
8755 \renewcommand*\glossaryheader}{\hline\endhead\hline\endfoot}%
8756 }
```

`longraggedheader` The `longraggedheader` style is like the `longragged` style but with a header:

```
8757 \newglossarystyle{longraggedheader}{%
```

Base it on the `glostylelongragged` style:

```
8758 \setglossarystyle{longragged}%
```

Set the table's header:

```
8759 \renewcommand*\glossaryheader}{%
8760   \bfseries \entryname & \bfseries \descriptionname
```

```
8761 \tabularnewline\endhead}%
8762 }
```

gedheaderborder The longraggedheaderborder style is like the longragged style but with a header and border:

```
8763 \newglossarystyle{longraggedheaderborder}{%
```

Base it on the glostylelongraggedborder style:

```
8764 \setglossarystyle{longraggedborder}{%
```

Set the table's header and add horizontal line to table's foot:

```
8765 \renewcommand*\glossaryheader}{%
8766 \hline\bfseries \entryname & \bfseries \descriptionname
8767 \tabularnewline\hline
8768 \endhead
8769 \hline\endfoot}%
8770 }
```

longragged3col The longragged3col style is like longragged but with 3 columns

```
8771 \newglossarystyle{longragged3col}{%
```

Use a longtable with 3 columns:

```
8772 \renewenvironment{theglossary}%
8773 {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}%
8774 >{\raggedright}p{\glspagelistwidth}}}%
8775 {\end{longtable}}%
```

No table header:

```
8776 \renewcommand*\glossaryheader}{}%
```

No headings between groups:

```
8777 \renewcommand*\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
8778 \renewcommand{\glossentry}[2]{%
8779 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8780 \glossentrydesc{##1} & ##2\tabularnewline
8781 }%
```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```
8782 \renewcommand{\subglossentry}[3]{%
8783 &
8784 \glssubentryitem{##2}%
8785 \glstarget{##2}{\strut}\glossentrydesc{##2} &
8786 ##3\tabularnewline
8787 }%
```

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>)

```
8788 \ifglsnogroupskip
8789 \renewcommand*\glsgroupskip}{}%
```

```

8790 \else
8791 \renewcommand*{\glsgroupskip}{ & & \tabularnewline}%
8792 \fi
8793 }

```

`ragged3colborder` The `longragged3colborder` style is like the `longragged3col` style but with a border:

```

8794 \newglossarystyle{longragged3colborder}{%
    Base it on the glostylelongragged3col style:
8795 \setglossarystyle{longragged3col}%
    Use a longtable with 3 columns with vertical lines around them:
8796 \renewenvironment{theglossary}%
8797 {\begin{longtable}{|l|>{\raggedright}p{\glsdescwidth}|%
8798 >{\raggedright}p{\glspagelistwidth}|}%
8799 {\end{longtable}}%
    Place horizontal lines at the head and foot of the table:
8800 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8801 }

```

`ragged3colheader` The `longragged3colheader` style is like `longragged3col` but with a header row:

```

8802 \newglossarystyle{longragged3colheader}{%
    Base it on the glostylelongragged3col style:
8803 \setglossarystyle{longragged3col}%
    Set the table's header:
8804 \renewcommand*{\glossaryheader}{%
8805 \bfseries\entryname&\bfseries\descriptionname&
8806 \bfseries\pagelistname\tabularnewline\endhead}%
8807 }

```

`colheaderborder` The `longragged3colheaderborder` style is like the above but with a border

```

8808 \newglossarystyle{longragged3colheaderborder}{%
    Base it on the glostylelongragged3colborder style:
8809 \setglossarystyle{longragged3colborder}%
    Set the table's header and add horizontal line at table's foot:
8810 \renewcommand*{\glossaryheader}{%
8811 \hline
8812 \bfseries\entryname&\bfseries\descriptionname&
8813 \bfseries\pagelistname\tabularnewline\hline\endhead
8814 \hline\endfoot}%
8815 }

```

`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.

```

8816 \newglossarystyle{altlongragged4col}{%

```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8817 \renewenvironment{theglossary}%
8818   {\begin{longtable}{1>{\raggedright}p{\glstdescwidth}1%
8819     >{\raggedright}p{\glspagelistwidth}}}%
8820   {\end{longtable}}%
```

No table header:

```
8821 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8822 \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):

```
8823 \renewcommand{\glossentry}[2]{%
8824   \glstarget{##1}{\glstarget{##1}{\glossentryname{##1}} &
8825   \glossentrydesc{##1} & \glossentrysymbol{##1} &
8826   ##2\tabularnewline
8827   }%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
8828 \renewcommand{\subglossentry}[3]{%
8829   &
8830   \glssubentryitem{##2}%
8831   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8832   \glossentrysymbol{##2} & ##3\tabularnewline
8833   }%
```

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>)

```
8834 \ifglsgroupskip
8835   \renewcommand*{\glsgroupskip}{}%
8836 \else
8837   \renewcommand*{\glsgroupskip}{ & & \tabularnewline}%
8838 \fi
8839 }
```

`ragged4colheader` The `altlongragged4colheader` style is like `altlongragged4col` but with a header row.

```
8840 \newglossarystyle{altlongragged4colheader}{%
```

Base it on the `glostylealtlongragged4col` style:

```
8841 \setglossarystyle{altlongragged4col}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8842 \renewenvironment{theglossary}%
8843   {\begin{longtable}{1>{\raggedright}p{\glstdescwidth}1%
8844     >{\raggedright}p{\glspagelistwidth}}}%
8845   {\end{longtable}}%
```

Table has a header:

```
8846 \renewcommand*{\glossaryheader}{%
8847 \bfseries\entryname&\bfseries\descriptionname&
8848 \bfseries \symbolname&
8849 \bfseries\pagelistname\tabularnewline\endhead}%
8850 }
```

`ragged4colborder` The `altlongragged4colborder` style is like `altlongragged4col` but with a border.

```
8851 \newglossarystyle{altlongragged4colborder}{%
```

Base it on the `glostylealtlongragged4col` style:

```
8852 \setglossarystyle{altlongragged4col}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8853 \renewenvironment{theglossary}%
8854 {\begin{longtable}{|l|>{\raggedright}p{\glstdescwidth}|l|}%
8855 >{\raggedright}p{\glspagelistwidth}|}}%
8856 {\end{longtable}}%
```

Add horizontal lines to the head and foot of the table:

```
8857 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8858 }
```

`colheaderborder` The `altlongragged4colheaderborder` style is like the above but with a header as well as a border.

```
8859 \newglossarystyle{altlongragged4colheaderborder}{%
```

Base it on the `glostylealtlongragged4col` style:

```
8860 \setglossarystyle{altlongragged4col}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8861 \renewenvironment{theglossary}%
8862 {\begin{longtable}{|l|>{\raggedright}p{\glstdescwidth}|l|}%
8863 >{\raggedright}p{\glspagelistwidth}|}}%
8864 {\end{longtable}}%
```

Add table header and horizontal line at the table's foot:

```
8865 \renewcommand*{\glossaryheader}{%
8866 \hline\bfseries\entryname&\bfseries\descriptionname&
8867 \bfseries \symbolname&
8868 \bfseries\pagelistname\hline\endhead
8869 \hline\endfoot}%
8870 }
```

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.

```
8871 \ProvidesPackage{glossary-mcols}[2019/01/06 v4.42 (NLCT)]
```

Required packages:

```
8872 \RequirePackage{multicol}
8873 \RequirePackage{glossary-tree}
```

`\indexspace` The are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```
8874 \providecommand{\indexspace}{%
8875   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
8876 }
```

`\glsmcols` Define macro in which to store the number of columns. (Defaults to 2.)

```
8877 \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.)

```
8878 \newglossarystyle{mcolindex}{%
8879   \setglossarystyle{index}%
8880   \renewenvironment{theglossary}%
8881     {%
8882       \begin{multicols}{\glsmcols}
8883       \setlength{\parindent}{0pt}%
8884       \setlength{\parskip}{0pt plus 0.3pt}%
8885       \let\item\glstreeitem
8886       \let\subitem\glstreesubitem
8887       \let\subsubitem\glstreesubsubitem
8888     }%
8889     {\end{multicols}}%
8890 }
```

`mcolindexgroup` As `mcolindex` but has headings:

```
8891 \newglossarystyle{mcolindexgroup}{%
8892   \setglossarystyle{mcolindex}%
8893   \renewcommand*{\glsgroupheading}[1]{%
8894     \item\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}\indexspace}%
8895 }
```

`indexhypergroup` The `mcolindexhypergroup` style is like the `mcolindexgroup` style but has hyper navigation.

```
8896 \newglossarystyle{mcolindexhypergroup}{%
```

Base it on the `glostylemcolindex` style:

```
8897   \setglossarystyle{mcolindex}%
```

Put navigation links to the groups at the start of the glossary:

```
8898   \renewcommand*{\glossaryheader}{%
8899     \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.

```
8900 \renewcommand*{\glsgroupheading}[1]{%
8901   \item\glstreegroupheaderfmt
8902     {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
8903   \indexspace}%
8904 }
```

`colindexspannav` Similar to `mcolindexhypergroup`, but puts the navigation line in the optional argument of `multicols`.

```
8905 \newglossarystyle{mcolindexspannav}{%
8906   \setglossarystyle{index}%
8907   \renewenvironment{theglossary}%
8908     {%
8909       \begin{multicols}{\glsmcols}[\noindent\glstreenavigationfmt{\glsnavigation}]
8910       \setlength{\parindent}{0pt}%
8911       \setlength{\parskip}{0pt plus 0.3pt}%
8912       \let\item\glstreeitem}%
8913     {\end{multicols}}%
```

Add a heading for each group (with a target). The group's title is in bold followed by a vertical gap.

```
8914 \renewcommand*{\glsgroupheading}[1]{%
8915   \item\glstreegroupheaderfmt
8916     {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
8917   \indexspace}%
8918 }
```

`mcoltree` Multi-column index style. Same as the tree, but puts the glossary in multiple columns.

```
8919 \newglossarystyle{mcoltree}{%
8920   \setglossarystyle{tree}%
8921   \renewenvironment{theglossary}%
8922     {%
8923       \begin{multicols}{\glsmcols}
8924       \setlength{\parindent}{0pt}%
8925       \setlength{\parskip}{0pt plus 0.3pt}%
8926     }%
8927     {\end{multicols}}%
8928 }
```

`mcoltreegroup` Like the `mcoltree` style but the glossary groups have headings.

```
8929 \newglossarystyle{mcoltreegroup}{%
   Base it on the glostylemcoltree style:
8930   \setglossarystyle{mcoltree}%
```

Each group has a heading (in bold) followed by a vertical gap):

```
8931 \renewcommand{\glsgroupheading}[1]{\par
8932   \noindent\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}\par\indexspace}%
8933 }
```

`treehypergroup` The `mcoltreehypergroup` style is like the `treegroup` style, but has a set of links to the groups at the start of the glossary.

```
8934 \newglossarystyle{mcoltreehypergroup}{%
```

Base it on the `glostylemcoltree` style:

```
8935 \setglossarystyle{mcoltree}{%
```

Put navigation links to the groups at the start of the `theglossary` environment:

```
8936 \renewcommand*{\glossaryheader}{%
```

```
8937   \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
8938 \renewcommand*{\glsgroupheading}[1]{%
```

```
8939   \par\noindent
```

```
8940   \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
```

```
8941   \indexspace}%
```

```
8942 }
```

`mcoltreespannav` Similar to the `mcoltreehypergroup` style but the navigation line is put in the optional argument of the `multicols` environment.

```
8943 \newglossarystyle{mcoltreespannav}{%
```

```
8944   \setglossarystyle{tree}%
```

```
8945   \renewenvironment{theglossary}{%
```

```
8946     {%
```

```
8947       \begin{multicols}{\glsncols}\noindent\glstreenavigationfmt{\glsnavigation}]
```

```
8948       \setlength{\parindent}{0pt}%
```

```
8949       \setlength{\parskip}{0pt plus 0.3pt}%
```

```
8950     }%
```

```
8951   {\end{multicols}}}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
8952 \renewcommand*{\glsgroupheading}[1]{%
```

```
8953   \par\noindent
```

```
8954   \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
```

```
8955   \indexspace}%
```

```
8956 }
```

`mcoltreename` Multi-column index style. Same as the `treename`, but puts the glossary in multiple columns.

```
8957 \newglossarystyle{mcoltreename}{%
```

```
8958   \setglossarystyle{treename}%
```

```
8959   \renewenvironment{theglossary}{%
```

```
8960     {%
```

```

8961     \begin{multicols}{\glsmcols}
8962     \setlength{\parindent}{0pt}%
8963     \setlength{\parskip}{0pt plus 0.3pt}%
8964 }%
8965 {\end{multicols}}%
8966 }

```

`treenamegroup` Like the `mcoltreename` style but the glossary groups have headings.

```

8967 \newglossarystyle{mcoltreenamegroup}{%
    Base it on the glostylemcoltreename style:
8968 \setglossarystyle{mcoltreename}%
    Give each group a heading:
8969 \renewcommand{\glsgroupheading}[1]{\par
8970 \noindent\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}\par\indexspace}%
8971 }

```

`namehypergroup` The `mcoltreenamehypergroup` style is like the `mcoltreenamegroup` style, but has a set of links to the groups at the start of the glossary.

```

8972 \newglossarystyle{mcoltreenamehypergroup}{%
    Base it on the glostylemcoltreename style:
8973 \setglossarystyle{mcoltreename}%
    Put navigation links to the groups at the start of the theglossary environment:
8974 \renewcommand*{\glossaryheader}{%
8975 \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
    Each group has a heading (in bold with a target) followed by a vertical gap):
8976 \renewcommand*{\glsgroupheading}[1]{%
8977 \par\noindent
8978 \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8979 \indexspace}%
8980 }

```

`treenamepannav` Similar to the `mcoltreenamehypergroup` style but the navigation line is put in the optional argument of the `multicols` environment.

```

8981 \newglossarystyle{mcoltreenamepannav}{%
8982 \setglossarystyle{treename}%
8983 \renewenvironment{theglossary}%
8984 {%
8985 \begin{multicols}{\glsmcols}[\noindent\glstreenavigationfmt{\glsnavigation}]
8986 \setlength{\parindent}{0pt}%
8987 \setlength{\parskip}{0pt plus 0.3pt}%
8988 }%
8989 {\end{multicols}}%
    Each group has a heading (in bold with a target) followed by a vertical gap):
8990 \renewcommand*{\glsgroupheading}[1]{%
8991 \par\noindent

```

```

8992     \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8993     \indexspace}%
8994 }

```

`mcolalttree` Multi-column index style. Same as the `alttree`, but puts the glossary in multiple columns.

```

8995 \newglossarystyle{mcolalttree}{%
8996   \setglossarystyle{alttree}%
8997   \renewenvironment{theglossary}%
8998   {%
8999     \begin{multicols}{\glscols}
9000     \def\@gls@prevlevel{-1}%
9001     \mbox{}\par
9002   }%
9003   {\par\end{multicols}}%
9004 }

```

`colalttreegroup` Like the `mcolalttree` style but the glossary groups have headings.

```

9005 \newglossarystyle{colalttreegroup}{%
    Base it on the glostylemcolalttree style:
9006   \setglossarystyle{mcolalttree}%
    Give each group a heading.
9007   \renewcommand{\glsgroupheading}[1]{\par
9008     \def\@gls@prevlevel{-1}%
9009     \hangindent0pt\relax
9010     \parindent0pt\relax
9011     \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}\par\indexspace}%
9012 }

```

`treehypergroup` The `mcolalttreehypergroup` style is like the `mcolalttreegroup` style, but has a set of links to the groups at the start of the glossary.

```

9013 \newglossarystyle{mcolalttreehypergroup}{%
    Base it on the glostylemcolalttree style:
9014   \setglossarystyle{mcolalttree}%
    Put the navigation links in the header
9015   \renewcommand*{\glossaryheader}{%
9016     \par
9017     \def\@gls@prevlevel{-1}%
9018     \hangindent0pt\relax
9019     \parindent0pt\relax
9020     \glstreenavigationfmt{\glsnavigation}\par\indexspace}%
    Put a hypertext at the start of each group
9021   \renewcommand*{\glsgroupheading}[1]{%
9022     \par
9023     \def\@gls@prevlevel{-1}%
9024     \hangindent0pt\relax

```

```

9025 \parindent0pt\relax
9026 \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
9027 \indexspace}%
9028 }

```

`laltreespannav` Similar to the `mcolalttreehypergroup` style but the navigation line is put in the optional argument of the `multicols` environment.

```

9029 \newglossarystyle{mcolalttreespannav}{%
9030 \setglossarystyle{alttree}%
9031 \renewenvironment{theglossary}%
9032 {%
9033 \begin{multicols}{\glsncols}[\noindent\glstreenavigationfmt{\glsnavigation}]
9034 \def\@gls@prevlevel{-1}%
9035 \mbox{}\par
9036 }%
9037 {\par\end{multicols}}%

```

Put a `hypertarget` at the start of each group

```

9038 \renewcommand*{\glsgroupheading}[1]{%
9039 \par
9040 \def\@gls@prevlevel{-1}%
9041 \hangindent0pt\relax
9042 \parindent0pt\relax
9043 \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
9044 \indexspace}%
9045 }

```

3.8 Glossary Styles using supertabular environment (glossary-super package)

The glossary styles defined in the package use the `supertabular` environment.

```

9046 \ProvidesPackage{glossary-super}[2019/01/06 v4.42 (NLCT)]

```

Requires the package:

```

9047 \RequirePackage{supertabular}

```

`\glsdescwidth` This is a length that governs the width of the description column. This may already have been defined if `has` has been loaded.

```

9048 \@ifundefined{glsdescwidth}{%
9049 \newlength{glsdescwidth}
9050 \setlength{glsdescwidth}{0.6\hsize}
9051 }{}

```

`lspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined if `has` has been loaded.

```

9052 \@ifundefined{glspagelistwidth}{%
9053 \newlength{glspagelistwidth}
9054 \setlength{glspagelistwidth}{0.1\hsize}

```

9055 }{}

super The super glossary style uses the supertabular environment (it uses lengths defined in the package.)

9056 \newglossarystyle{super}{%

Put the glossary in a supertabular environment with two columns and no head or tail:

```
9057 \renewenvironment{theglossary}%
9058   {\tablehead{ }\tabletail{ }}%
9059   \begin{supertabular}[lp{\glsdescwidth}]{ }%
9060   {\end{supertabular}}%
```

Do nothing at the start of the table:

```
9061 \renewcommand*{\glossaryheader}{ }%
```

No group headings:

```
9062 \renewcommand*{\glsgroupheading}[1]{ }%
```

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```
9063 \renewcommand{\glossentry}[2]{%
9064   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
9065   \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline
9066   }%
```

Sub entries put in a row (no name, description and page list in second column):

```
9067 \renewcommand{\subglossentry}[3]{%
9068   &
9069   \glssubentryitem{##2}%
9070   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
9071   ##3\tabularnewline
9072   }%
```

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>)

```
9073 \ifglsnogroupskip
9074   \renewcommand*{\glsgroupskip}{ }%
9075 \else
9076   \renewcommand*{\glsgroupskip}{& \tabularnewline}%
9077 \fi
9078 }
```

superborder The superborder style is like the above, but with horizontal and vertical lines:

```
9079 \newglossarystyle{superborder}{%
```

Base it on the `glostylesuper` style:

```
9080 \setglossarystyle{super}%
```

Put the glossary in a supertabular environment with two columns and a horizontal line in the head and tail:

```
9081 \renewenvironment{theglossary}%
9082   {\tablehead{\hline}\tabletail{\hline}}%
```

```

9083     \begin{supertabular}{|l|p{\glsdescwidth}|}%
9084     {\end{supertabular}}%
9085 }

```

superheader The superheader style is like the super style, but with a header:

```

9086 \newglossarystyle{superheader}{%
    Base it on the glostylesuper style:
9087  \setglossarystyle{super}%
    Put the glossary in a supertabular environment with two columns, a header and no tail:
9088 \renewenvironment{theglossary}%
9089  {\tablehead{\bfseries \entryname &
9090   \bfseries\descriptionname\tabularnewline}%
9091   \tabletail{}}%
9092   \begin{supertabular}{lp{\glsdescwidth}}%
9093   {\end{supertabular}}%
9094 }

```

superheaderborder The superheaderborder style is like the super style but with a header and border:

```

9095 \newglossarystyle{superheaderborder}{%
    Base it on the glostylesuper style:
9096  \setglossarystyle{super}%
    Put the glossary in a supertabular environment with two columns, a header and horizontal
    lines above and below the table:
9097  \renewenvironment{theglossary}%
9098   {\tablehead{\hline\bfseries \entryname &
9099    \bfseries \descriptionname\tabularnewline\hline}%
9100   \tabletail{\hline}
9101   \begin{supertabular}{|l|p{\glsdescwidth}|}%
9102   {\end{supertabular}}%
9103 }

```

super3col The super3col style is like the super style, but with 3 columns:

```

9104 \newglossarystyle{super3col}{%
    Put the glossary in a supertabular environment with three columns and no head or tail:
9105  \renewenvironment{theglossary}%
9106   {\tablehead{}\tabletail{}}%
9107   \begin{supertabular}{lp{\glsdescwidth}p{\glspagelistwidth}}%
9108   {\end{supertabular}}%
    Do nothing at the start of the table:
9109  \renewcommand*{\glossaryheader}{}%
    No group headings:
9110  \renewcommand*{\glsgroupheading}[1]{}%

```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```

9111 \renewcommand{\glossentry}[2]{%
9112   \glentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
9113   \glossentrydesc{##1} & ##2\tabularnewline
9114 }%

```

Sub entries on a row (no name, description in second column, page list in last column):

```

9115 \renewcommand{\subglossentry}[3]{%
9116   &
9117   \glssubentryitem{##2}%
9118   \glstarget{##2}{\strut}\glossentrydesc{##2} &
9119   ##3\tabularnewline
9120 }%

```

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>)

```

9121 \ifglsnogroupskip
9122   \renewcommand*{\glsgroupskip}{}%
9123 \else
9124   \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
9125 \fi
9126 }

```

`super3colborder` The `super3colborder` style is like the `super3col` style, but with a border:

```
9127 \newglossarystyle{super3colborder}{%
```

Base it on the `glostylesuper3col` style:

```
9128 \setglossarystyle{super3col}{%
```

Put the glossary in a `supertabular` environment with three columns and a horizontal line in the head and tail:

```

9129 \renewenvironment{theglossary}%
9130   {\tablehead{\hline}\tabletail{\hline}%
9131   \begin{supertabular}{|l|p{\glsdescwidth}|p{\glspagelistwidth}|}%
9132   {\end{supertabular}}%
9133 }

```

`super3colheader` The `super3colheader` style is like the `super3col` style but with a header row:

```
9134 \newglossarystyle{super3colheader}{%
```

Base it on the `glostylesuper3col` style:

```
9135 \setglossarystyle{super3col}{%
```

Put the glossary in a `supertabular` environment with three columns, a header and no tail:

```

9136 \renewenvironment{theglossary}%
9137   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
9138   \bfseries\pagelistname\tabularnewline}\tabletail{}}%
9139   \begin{supertabular}{lp{\glsdescwidth}p{\glspagelistwidth}}}%
9140   {\end{supertabular}}%
9141 }

```


colheaderborder The super3colheaderborder style is like the super3col style but with a header and border:

```
9142 \newglossarystyle{super3colheaderborder}{%
```

Base it on the glostylesuper3colborder style:

```
9143 \setglossarystyle{super3colborder}{%
```

Put the glossary in a supertabular environment with three columns, a header with horizontal lines and a horizontal line in the tail:

```
9144 \renewenvironment{theglossary}{%
9145   {\tablehead{\hline
9146     \bfseries\entryname&\bfseries\descriptionname&
9147     \bfseries\pagelistname\tabularnewline\hline}%
9148   \tabletail{\hline}%
9149   \begin{supertabular}{|l|p{\glstdescwidth}|p{\glspagelistwidth}|}%
9150   {\end{supertabular}}%
9151 }
```

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.

```
9152 \newglossarystyle{super4col}{%
```

Put the glossary in a supertabular environment with four columns and no head or tail:

```
9153 \renewenvironment{theglossary}{%
9154   {\tablehead{}\tabletail{}%
9155   \begin{supertabular}{|l|l|l|l|}%
9156   \end{supertabular}}%
```

Do nothing at the start of the table:

```
9157 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9158 \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:

```
9159 \renewcommand{\glossentry}[2]{%
9160   \glstarget{##1}\glstarget{##1}{\glossentryname{##1}} &
9161   \glossentrydesc{##1} &
9162   \glossentrysymbol{##1} & ##2\tabularnewline
9163 }%
```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```
9164 \renewcommand{\subglossentry}[3]{%
9165   &
9166   \glssubentryitem{##2}%
9167   \glstarget{##2}{\strut}\glossentrydesc{##2} &
9168   \glossentrysymbol{##2} & ##3\tabularnewline
9169 }%
```

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>)

```
9170 \ifglsgroupskip
9171   \renewcommand*\glsgroupskip}{}%
9172 \else
9173   \renewcommand*\glsgroupskip}{& & & \tabularnewline}%
9174 \fi
9175 }
```

`super4colheader` The `super4colheader` style is like the `super4col` but with a header row.

```
9176 \newglossarystyle{super4colheader}{%
  Base it on the glostylesuper4col style:
9177   \setglossarystyle{super4col}%
  Put the glossary in a supertabular environment with four columns, a header and no tail:
9178   \renewenvironment{theglossary}%
9179     {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
9180               \bfseries\symbolname &
9181               \bfseries\pagelistname\tabularnewline}%
9182     \tabletail{}}%
9183     \begin{supertabular}{1111}}%
9184     {\end{supertabular}}%
9185 }
```

`super4colborder` The `super4colborder` style is like the `super4col` but with a border.

```
9186 \newglossarystyle{super4colborder}{%
  Base it on the glostylesuper4col style:
9187   \setglossarystyle{super4col}%
  Put the glossary in a supertabular environment with four columns and a horizontal line in the
  head and tail:
9188   \renewenvironment{theglossary}%
9189     {\tablehead{\hline}\tabletail{\hline}%
9190     \begin{supertabular}{11111111}}%
9191     {\end{supertabular}}%
9192 }
```

`colheaderborder` The `super4colheaderborder` style is like the `super4col` but with a header and border.

```
9193 \newglossarystyle{super4colheaderborder}{%
  Base it on the glostylesuper4col style:
9194   \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:
9195   \renewenvironment{theglossary}%
9196     {\tablehead{\hline\bfseries\entryname&\bfseries\descriptionname&
9197               \bfseries\symbolname &
```

```

9198     \bfseries\pagelistname\tabularnewline\hline}%
9199     \tabletail{\hline}%
9200     \begin{supertabular}{|l|l|l|l|}%
9201     {\end{supertabular}}%
9202 }

```

`altsuper4col` The `altsuper4col` glossary style is like `super4col` but has provision for multiline descriptions.

```

9203 \newglossarystyle{altsuper4col}{%
    Base it on the glostylesuper4col style:
9204 \setglossarystyle{super4col}%
    Put the glossary in a supertabular environment with four columns and no head or tail:
9205 \renewenvironment{theglossary}%
9206     {\tablehead{}\tabletail{}%
9207     \begin{supertabular}{lp{\glsdescwidth}lp{\glspagelistwidth}}}%
9208     {\end{supertabular}}%
9209 }

```

`super4colheader` The `altsuper4colheader` style is like the `altsuper4col` but with a header row.

```

9210 \newglossarystyle{altsuper4colheader}{%
    Base it on the glostylesuper4colheader style:
9211 \setglossarystyle{super4colheader}%
    Put the glossary in a supertabular environment with four columns, a header and no tail:
9212 \renewenvironment{theglossary}%
9213     {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
9214     \bfseries\symbolname &
9215     \bfseries\pagelistname\tabularnewline}\tabletail{}}%
9216     \begin{supertabular}{lp{\glsdescwidth}lp{\glspagelistwidth}}}%
9217     {\end{supertabular}}%
9218 }

```

`super4colborder` The `altsuper4colborder` style is like the `altsuper4col` but with a border.

```

9219 \newglossarystyle{altsuper4colborder}{%
    Base it on the glostylesuper4colborder style:
9220 \setglossarystyle{super4colborder}%
    Put the glossary in a supertabular environment with four columns and a horizontal line in the
    head and tail:
9221 \renewenvironment{theglossary}%
9222     {\tablehead{\hline}\tabletail{\hline}%
9223     \begin{supertabular}%
9224     {l|lp{\glsdescwidth}|l|lp{\glspagelistwidth}|}%
9225     {\end{supertabular}}%
9226 }

```

`colheaderborder` The `altsuper4colheaderborder` style is like the `altsuper4col` but with a header and border.

```

9227 \newglossarystyle{altsuper4colheaderborder}{%

```

Base it on the `glostylesuper4colheaderborder` style:

```
9228 \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:

```
9229 \renewenvironment{theglossary}%
9230   {\tablehead{\hline
9231     \bfseries\entryname &
9232     \bfseries\descriptionname &
9233     \bfseries\symbolname &
9234     \bfseries\pagelistname\tabularnewline\hline}%
9235   \tabletail{\hline}%
9236   \begin{supertabular}%
9237     {ll|p{\glsdescwidth}|l|p{\glspagelistwidth}|}%
9238   {\end{supertabular}}%
9239 }
```

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.

```
9240 \ProvidesPackage{glossary-superragged}[2019/01/06 v4.42 (NLCT)]
```

Requires the package:

```
9241 \RequirePackage{array}
```

Requires the package:

```
9242 \RequirePackage{supertabular}
```

`\glsdescwidth` This is a length that governs the width of the description column. This may already have been defined.

```
9243 \@ifundefined{glsdescwidth}{%
9244   \newlength\glsdescwidth
9245   \setlength{\glsdescwidth}{0.6\hsize}
9246 }{}
```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```
9247 \@ifundefined{glspagelistwidth}{%
9248   \newlength\glspagelistwidth
9249   \setlength{\glspagelistwidth}{0.1\hsize}
9250 }{}
```

`superragged` The `superragged` glossary style uses the `supertabular` environment.

```
9251 \newglossarystyle{superragged}{%
```

Put the glossary in a supertabular environment with two columns and no head or tail:

```
9252 \renewenvironment{theglossary}%
9253   {\tablehead{}\tabletail{}}%
9254   \begin{supertabular}{1>{\raggedright}p{\glsdescwidth}}%
9255   {\end{supertabular}}%
```

Do nothing at the start of the table:

```
9256 \renewcommand*{\glossaryheader}{}
```

No group headings:

```
9257 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```
9258 \renewcommand{\glossentry}[2]{%
9259   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
9260   \glossentrydesc{##1}\glspostdescription\space ##2%
9261   \tabularnewline
9262 }%
```

Sub entries put in a row (no name, description and page list in second column):

```
9263 \renewcommand{\subglossentry}[3]{%
9264   &
9265   \glsesubentryitem{##2}%
9266   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
9267   ##3%
9268   \tabularnewline
9269 }%
```

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>)

```
9270 \ifglsnogroupskip
9271   \renewcommand*{\glsgroupskip}{}%
9272 \else
9273   \renewcommand*{\glsgroupskip}{& \tabularnewline}%
9274 \fi
9275 }
```

`superraggedborder` The `superraggedborder` style is like the above, but with horizontal and vertical lines:

```
9276 \newglossarystyle{superraggedborder}{%
```

Base it on the `glostylesuperragged` style:

```
9277 \setglossarystyle{superragged}%
```

Put the glossary in a supertabular environment with two columns and a horizontal line in the head and tail:

```
9278 \renewenvironment{theglossary}%
9279   {\tablehead{\hline}\tabletail{\hline}%
9280   \begin{supertabular}{|1|>{\raggedright}p{\glsdescwidth}|}%
9281   {\end{supertabular}}%
9282 }
```

superraggedheader The superraggedheader style is like the super style, but with a header:

```
9283 \newglossarystyle{superraggedheader}{%
```

Base it on the glostylesuperragged style:

```
9284 \setglossarystyle{superragged}{%
```

Put the glossary in a supertabular environment with two columns, a header and no tail:

```
9285 \renewenvironment{theglossary}{%
```

```
9286 {\tablehead{\bfseries \entryname & \bfseries \descriptionname
```

```
9287 \tabularnewline}}%
```

```
9288 \tabletail{}}%
```

```
9289 \begin{supertabular}{1>{\raggedright}p{\glsdescwidth}}%
```

```
9290 {\end{supertabular}}%
```

```
9291 }
```

superraggedheaderborder The superraggedheaderborder style is like the superragged style but with a header and border:

```
9292 \newglossarystyle{superraggedheaderborder}{%
```

Base it on the glostylesuper style:

```
9293 \setglossarystyle{superragged}{%
```

Put the glossary in a supertabular environment with two columns, a header and horizontal lines above and below the table:

```
9294 \renewenvironment{theglossary}{%
```

```
9295 {\tablehead{\hline\bfseries \entryname &
```

```
9296 \bfseries \descriptionname\tabularnewline\hline}}%
```

```
9297 \tabletail{\hline}
```

```
9298 \begin{supertabular}{1|1>{\raggedright}p{\glsdescwidth}|}}%
```

```
9299 {\end{supertabular}}%
```

```
9300 }
```

superragged3col The superragged3col style is like the superragged style, but with 3 columns:

```
9301 \newglossarystyle{superragged3col}{%
```

Put the glossary in a supertabular environment with three columns and no head or tail:

```
9302 \renewenvironment{theglossary}{%
```

```
9303 {\tablehead{}\tabletail{}}%
```

```
9304 \begin{supertabular}{1>{\raggedright}p{\glsdescwidth}%
```

```
9305 >{\raggedright}p{\glspagelistwidth}}%
```

```
9306 {\end{supertabular}}%
```

Do nothing at the start of the table:

```
9307 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9308 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
9309 \renewcommand{\glossentry}[2]{%
```

```
9310 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
```

```
9311 \glossentrydesc{##1} &
```

```

9312     ##2\tabularnewline
9313 }%

```

Sub entries on a row (no name, description in second column, page list in last column):

```

9314 \renewcommand{\subglossentry}[3]{%
9315     &
9316     \glssubentryitem{##2}%
9317     \glstarget{##2}{\strut}\glossentrydesc{##2} &
9318     ##3\tabularnewline
9319 }%

```

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>)

```

9320 \ifglsnogroupskip
9321 \renewcommand*{\glsgroupskip}{}%
9322 \else
9323 \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
9324 \fi
9325 }

```

`ragged3colborder` The `superragged3colborder` style is like the `superragged3col` style, but with a border:

```

9326 \newglossarystyle{superragged3colborder}{%

```

Base it on the `glostylesuperragged3col` style:

```

9327 \setglossarystyle{superragged3col}%

```

Put the glossary in a `supertabular` environment with three columns and a horizontal line in the head and tail:

```

9328 \renewenvironment{theglossary}%
9329     {\tablehead{\hline}\tabletail{\hline}%
9330     \begin{supertabular}{|l|>{\raggedright}p{\glsdescwidth}|%
9331     >{\raggedright}p{\glspagelistwidth}|}%
9332     {\end{supertabular}}%
9333 }

```

`ragged3colheader` The `superragged3colheader` style is like the `superragged3col` style but with a header row:

```

9334 \newglossarystyle{superragged3colheader}{%

```

Base it on the `glostylesuperragged3col` style:

```

9335 \setglossarystyle{superragged3col}%

```

Put the glossary in a `supertabular` environment with three columns, a header and no tail:

```

9336 \renewenvironment{theglossary}%
9337     {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
9338     \bfseries\pagelistname\tabularnewline}\tabletail{}}%
9339     \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}%
9340     >{\raggedright}p{\glspagelistwidth}}%
9341     {\end{supertabular}}%
9342 }

```

colheaderborder The superragged3colheaderborder style is like the superragged3col style but with a header and border:

```
9343 \newglossarystyle{superragged3colheaderborder}{%
```

Base it on the glostylesuperragged3colborder style:

```
9344 \setglossarystyle{superragged3colborder}{%
```

Put the glossary in a supertabular environment with three columns, a header with horizontal lines and a horizontal line in the tail:

```
9345 \renewenvironment{theglossary}{%
```

```
9346   {\tablehead{\hline
```

```
9347     \bfseries\entryname&\bfseries\descriptionname&
```

```
9348     \bfseries\pagelistname\tabularnewline\hline}%
```

```
9349   \tabletail{\hline}%
```

```
9350   \begin{supertabular}{|l|>{\raggedright}p{\glsdescwidth}|%
```

```
9351     >{\raggedright}p{\glspagelistwidth}|}}%
```

```
9352   {\end{supertabular}}%
```

```
9353 }
```

superragged4col The altsuperragged4col glossary style is like altsuper4col style in the package but uses ragged right formatting in the description and page list columns.

```
9354 \newglossarystyle{altsuperragged4col}{%
```

Put the glossary in a supertabular environment with four columns and no head or tail:

```
9355 \renewenvironment{theglossary}{%
```

```
9356   {\tablehead{}\tabletail{}}%
```

```
9357   \begin{supertabular}{|l>{\raggedright}p{\glsdescwidth}l%
```

```
9358     >{\raggedright}p{\glspagelistwidth}|}}%
```

```
9359   {\end{supertabular}}%
```

Do nothing at the start of the table:

```
9360 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9361 \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:

```
9362 \renewcommand{\glossentry}[2]{%
```

```
9363   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
```

```
9364   \glossentrydesc{##1} &
```

```
9365   \glossentrysymbol{##1} & ##2\tabularnewline
```

```
9366   }%
```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```
9367 \renewcommand{\subglossentry}[3]{%
```

```
9368   &
```

```
9369   \glssubentryitem{##2}%
```

```
9370   \glstarget{##2}{\strut}\glossentrydesc{##2} &
```

```
9371   \glossentrysymbol{##2} & ##3\tabularnewline
```

```
9372   }%
```


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>)

```
9373 \ifglsnogroupskip
9374 \renewcommand*\glsgroupskip}{}%
9375 \else
9376 \renewcommand*\glsgroupskip}{& & \tabularnewline}%
9377 \fi
9378 }
```

`ragged4colheader` The `altsuperragged4colheader` style is like the `altsuperragged4col` style but with a header row.

```
9379 \newglossarystyle{altsuperragged4colheader}{%
```

Base it on the `glostylealtsuperragged4col` style:

```
9380 \setglossarystyle{altsuperragged4col}{%
```

Put the glossary in a `supertabular` environment with four columns, a header and no tail:

```
9381 \renewenvironment{theglossary}{%
9382 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
9383 \bfseries\symbolname &
9384 \bfseries\pagelistname\tabularnewline}\tabletail{}}%
9385 \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}l%
9386 >{\raggedright}p{\glspagelistwidth}}}%
9387 {\end{supertabular}}}%
9388 }
```

`ragged4colborder` The `altsuperragged4colborder` style is like the `altsuperragged4col` style but with a border.

```
9389 \newglossarystyle{altsuperragged4colborder}{%
```

Base it on the `glostylealtsuperragged4col` style:

```
9390 \setglossarystyle{altsuper4col}{%
```

Put the glossary in a `supertabular` environment with four columns and a horizontal line in the head and tail:

```
9391 \renewenvironment{theglossary}{%
9392 {\tablehead{\hline}\tabletail{\hline}%
9393 \begin{supertabular}%
9394 {ll|>{\raggedright}p{\glsdescwidth}ll|}%
9395 >{\raggedright}p{\glspagelistwidth}ll}}%
9396 {\end{supertabular}}}%
9397 }
```

`colheaderborder` The `altsuperragged4colheaderborder` style is like the `altsuperragged4col` style but with a header and border.

```
9398 \newglossarystyle{altsuperragged4colheaderborder}{%
```

Base it on the `glostylealtsuperragged4col` style:

```
9399 \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:

```

9400 \renewenvironment{theglossary}%
9401   {\tablehead{\hline
9402     \bfseries\entryname &
9403     \bfseries\descriptionname &
9404     \bfseries\symbolname &
9405     \bfseries\pagelistname\tabularnewline\hline}%
9406   \tabletail{\hline}%
9407   \begin{supertabular}%
9408     {||>\raggedright}p{\glstdescwidth}|||%
9409     >\raggedright}p{\glspagelistwidth}||}%
9410   {\end{supertabular}}%
9411 }

```

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.

```

9412 \ProvidesPackage{glossary-tree}[2019/01/06 v4.42 (NLCT)]

```

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```

9413 \providecommand{\indexspace}{%
9414   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
9415 }

```

`\glstreenamefmt` Format used to display the name in the tree styles. (This may be counteracted by `\glstnamefont`.) This command was previously also used to format the group headings.

```

9416 \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.

```

9417 \newcommand*{\glstreegroupheaderfmt}[1]{\glstreenamefmt{#1}}

```

`\glstreenavigationfmt` Format used to display the navigation header in the tree styles.

```

9418 \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.

```

9419 \ifdef\@idxitem
9420 {\newcommand{\glstreeitem}{\@idxitem}}
9421 {\newcommand{\glstreeitem}{\par\hangindent40\p@}}

```

`\glstreesubitem` Level 1 item used in index style.

```
9422 \ifdef\subitem
9423 {\let\glstreesubitem\subitem}
9424 {\newcommand\glstreesubitem{\glstreeitem\hspace*{20\p@}}}
```

`streesubsubitem` Level 1 item used in index style.

```
9425 \ifdef\subsubitem
9426 {\let\glstreesubsubitem\subsubitem}
9427 {\newcommand\glstreesubsubitem{\glstreeitem\hspace*{30\p@}}}
```

`\glstreepredesc` Allow the user to adjust the space before the description (except for the `alttree` style).

```
9428 \newcommand{\glstreepredesc}{\space}
```

`reechildpredesc` Allow the user to adjust the space before the description for sub-entries (except for the `treenoname` and `alttree` style).

```
9429 \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.

```
9430 \newglossarystyle{index}{%
```

Set the paragraph indentation and skip and define `\item` to be the same as that used by `theindex`:

```
9431 \renewenvironment{theglossary}%
9432   {\setlength{\parindent}{0pt}}%
9433   \setlength{\parskip}{0pt plus 0.3pt}}%
9434   \let\item\glstreeitem
9435   \let\subitem\glstreesubitem
9436   \let\subsubitem\glstreesubsubitem
9437   }%
```

```
9438   {\par}}%
```

Do nothing at the start of the environment:

```
9439 \renewcommand*{\glossaryheader}{}
```

No group headers:

```
9440 \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.

```
9441 \renewcommand*{\glossentry}[2]{%
9442   \item\glstreeitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
9443   \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
9444   \glstreepredesc \glossentrydesc{##1}\glspostdescription\space ##2%
9445 }%
```

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.

```

9446 \renewcommand{\subglossentry}[3]{%
9447   \ifcase##1\relax
9448     % level 0
9449     \item
9450   \or
9451     % level 1
9452     \subitem
9453     \glssubentryitem{##2}%
9454   \else
9455     % all other levels
9456     \subsubitem
9457   \fi
9458   \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
9459   \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
9460   \glstreechildpredesc\glossentrydesc{##2}\glspostdescription\space ##3%
9461 }%
```

Vertical gap between groups is the same as that used by indices:

```

9462 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}}
```

`indexgroup` The `indexgroup` style is like the `index` style but has headings.

```

9463 \newglossarystyle{indexgroup}{%
```

Base it on the `glostyleindex` style:

```

9464 \setglossarystyle{index}%
```

Add a heading for each group. This puts the group's title in bold followed by a vertical gap.

```

9465 \renewcommand*{\glsgroupheading}[1]{%
9466   \item\glstreegroupheaderfmt{\glsggetgrouptitle{##1}}%
9467   \indexspace
9468 }%
9469 }
```

`indexhypergroup` The `indexhypergroup` style is like the `indexgroup` style but has hyper navigation.

```

9470 \newglossarystyle{indexhypergroup}{%
```

Base it on the `glostyleindex` style:

```

9471 \setglossarystyle{index}%
```

Put navigation links to the groups at the start of the glossary:

```

9472 \renewcommand*{\glossaryheader}{%
9473   \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.

```

9474 \renewcommand*{\glsgroupheading}[1]{%
9475   \item\glstreegroupheaderfmt
```

```

9476     {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
9477     \indexspace}%
9478 }

```

tree The tree glossary style is similar in style to the index style, but can have arbitrary levels.

```
9479 \newglossarystyle{tree}{%
```

Set the paragraph indentation and skip:

```

9480 \renewenvironment{theglossary}%
9481   {\setlength{\parindent}{0pt}%
9482    \setlength{\parskip}{0pt plus 0.3pt}}%
9483   {}%

```

Do nothing at the start of the theglossary environment:

```
9484 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9485 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries: name in bold, followed by symbol in brackets (if it exists), the description and the page list:

```

9486 \renewcommand{\glossentry}[2]{%
9487   \hangindent0pt\relax
9488   \parindent0pt\relax
9489   \glsentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
9490   \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
9491   \glstreepredesc\glossentrydesc{##1}\glspostdescription\space##2\par
9492 }%

```

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.

```

9493 \renewcommand{\subglossentry}[3]{%
9494   \hangindent##1\glstreeindent\relax
9495   \parindent##1\glstreeindent\relax
9496   \ifnum##1=1\relax
9497     \glssubentryitem{##2}%
9498     \fi
9499     \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
9500     \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
9501     \glstreechildpredesc\glossentrydesc{##2}\glspostdescription\space ##3\par
9502 }%

```

Vertical gap between groups is the same as that used by indices:

```
9503 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}}
```

treegroup Like the tree style but the glossary groups have headings.

```
9504 \newglossarystyle{treegroup}{%
```

Base it on the glostyletree style:

```
9505 \setglossarystyle{tree}%
```

Each group has a heading (in bold) followed by a vertical gap):

```
9506 \renewcommand{\glsgroupheading}[1]{\par
9507 \noindent\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}\par
9508 \indexspace}%
9509 }
```

`treehypergroup` The `treehypergroup` style is like the `treegroup` style, but has a set of links to the groups at the start of the glossary.

```
9510 \newglossarystyle{treehypergroup}{%
```

Base it on the `glostyletree` style:

```
9511 \setglossarystyle{tree}%
```

Put navigation links to the groups at the start of the `theglossary` environment:

```
9512 \renewcommand*{\glossaryheader}{%
```

```
9513 \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
9514 \renewcommand*{\glsgroupheading}[1]{%
```

```
9515 \par\noindent
```

```
9516 \glstreegroupheaderfmt
```

```
9517 {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
```

```
9518 \indexspace}%
```

```
9519 }
```

`\glstreeindent` Length governing left indent for each level of the tree style.

```
9520 \newlength\glstreeindent
```

```
9521 \setlength{\glstreeindent}{10pt}
```

`treenoname` The `treenoname` glossary style is like the `tree` style, but doesn't print the name or symbol for sub-levels.

```
9522 \newglossarystyle{treenoname}{%
```

Set the paragraph indentation and skip:

```
9523 \renewenvironment{theglossary}{%
```

```
9524 {\setlength{\parindent}{0pt}%
```

```
9525 \setlength{\parskip}{0pt plus 0.3pt}}%
```

```
9526 {}%
```

No header:

```
9527 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9528 \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.

```
9529 \renewcommand{\glossentry}[2]{%
```

```
9530 \hangindent0pt\relax
```

```
9531 \parindent0pt\relax
```

```
9532 \glsentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
```

```

9533   \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
9534   \glstreepredesc\glossentrydesc{##1}\glspostdescription\space##2\par
9535 }%

```

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.

```

9536 \renewcommand{\subglossentry}[3]{%
9537   \hangindent##1\glstreeindent\relax
9538   \parindent##1\glstreeindent\relax
9539   \ifnum##1=1\relax
9540     \glssubentryitem{##2}%
9541   \fi
9542   \glstarget{##2}{\strut}%
9543   \glossentrydesc{##2}\glspostdescription\space##3\par
9544 }%

```

Vertical gap between groups is the same as that used by indices:

```

9545 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
9546 }

```

`treenonamegroup` Like the `treenoname` style but the glossary groups have headings.

```

9547 \newglossarystyle{treenonamegroup}{%
  Base it on the glostyletreenoname style:
9548 \setglossarystyle{treenoname}%
  Give each group a heading:
9549 \renewcommand{\glsgroupheading}[1]{\par
9550   \noindent\glstreegroupheaderfmt
9551   {\glsgrouptitle{##1}}\par\indexspace}%
9552 }

```

`onamehypergroup` The `treenonamehypergroup` style is like the `treenonamegroup` style, but has a set of links to the groups at the start of the glossary.

```

9553 \newglossarystyle{treenonamehypergroup}{%
  Base it on the glostyletreenoname style:
9554 \setglossarystyle{treenoname}%
  Put navigation links to the groups at the start of the theglossary environment:
9555 \renewcommand*{\glossaryheader}{%
9556   \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
  Each group has a heading (in bold with a target) followed by a vertical gap):
9557 \renewcommand*{\glsgroupheading}[1]{%
9558   \par\noindent
9559   \glstreegroupheaderfmt
9560   {\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}\par
9561   \indexspace}%
9562 }

```

`esttoplevelname` Find the widest name over all parentless entries in the given glossary or glossaries.

```
9563 \newrobustcmd*{\glsfindwidesttoplevelname}[1][\@glo@types]{%
9564   \dimen@=0pt\relax
9565   \gls@tmplen=0pt\relax
9566   \forallglossaries[#1]{\@gls@type}%
9567   {%
9568     \forallglsentries[\@gls@type]{\@glo@label}%
9569     {%
9570       \ifglsahasparent{\@glo@label}%
9571       }%
9572       {%
9573         \settowidth{\dimen@}%
9574         {\glsstreenamefmt{\glsentryname{\@glo@label}}}%
9575         \ifdim\dimen@>\gls@tmplen
9576           \gls@tmplen=\dimen@
9577           \letcs{\@glswidestname}{glo\@glsdetoklabel{\@glo@label}@name}%
9578           \fi
9579         }%
9580       }%
9581     }%
9582 }
```

`\glssetwidest` `\glssetwidest [⟨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.

```
9583 \newcommand*{\glssetwidest}[2][0]{%
9584   \expandafter\def\csname @glswidestname\romannumeral#1\endcsname{%
9585     #2}%
9586 }
```

`\@glswidestname` Initialise `\@glswidestname`.

```
9587 \newcommand*{\@glswidestname}{}
```

`\glsstreenamebox` Used by the alttree style to create the box for the name and associated information.

```
9588 \newcommand*{\glsstreenamebox}[2]{%
9589   \makebox[#1][l]{#2}%
9590 }
```

`alttree` The alttree glossary style is similar in style to the tree style, but the indentation is obtained from the width of `\@glswidestname` which is set using `\glssetwidest`.

```
9591 \newglossarystyle{alttree}{%
```

Redefine theglossary environment.

```
9592 \renewenvironment{theglossary}%
9593   {\def\@gls@prevlevel{-1}%
9594    \mbox{}\par}%
9595   {\par}%
```

Set the header and group headers to nothing.

```
9596 \renewcommand*{\glossaryheader}{}%
9597 \renewcommand*{\glsgroupheading}[1]{}%
```


Redefine the way that the level 0 entries are displayed.

```
9598 \renewcommand{\glossentry}[2]{%
9599   \ifnum\@gls@prevlevel=0\relax
9600   \else
```

Find out how big the indentation should be by measuring the widest entry.

```
9601     \settowidth{\glstreeindent}{\glstreenamefmt{\@glswidestname\space}}%
9602   \fi
```

Set the hangindent and paragraph indent.

```
9603   \hangindent\glstreeindent
9604   \parindent\glstreeindent
```

Put the name to the left of the paragraph block.

```
9605   \makebox[0pt][r]{\glstreenamebox{\glstreeindent}{%
9606     \glsentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}}}%
```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
9607   \ifglshassymbol{##1}{(\glossentrysymbol{##1})\space}{}%
```

Do the description followed by the description terminator and location list.

```
9608   \glossentrydesc{##1}\glspostdescription \space ##2\par
```

Set the previous level to 0.

```
9609   \def\@gls@prevlevel{0}%
9610 }%
```

Redefine the way sub-entries are displayed.

```
9611 \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.

```
9612   \ifnum##1=1\relax
9613     \glssubentryitem{##2}%
9614   \fi
```

If the level hasn't changed, keep the same settings, otherwise adjust `\glstreeindent` accordingly.

```
9615   \ifnum\@gls@prevlevel=##1\relax
9616   \else
```

Compute the widest entry for this level, or for level 0 if not defined for this level. Store in `\gls@tmplen`

```
9617     \@ifundefined{@glswidestname\romannumeral##1}{%
9618       \settowidth{\gls@tmplen}{\glstreenamefmt{\@glswidestname\space}}{%
9619       \settowidth{\gls@tmplen}{\glstreenamefmt{%
9620         \csname @glswidestname\romannumeral##1\endcsname\space}}}%
```

Determine if going up or down a level

```
9621   \ifnum\@gls@prevlevel<##1\relax
```

Depth has increased, so add the width of the widest entry to `\glstreeindent`.

```
9622     \setlength\glstreeindent\gls@tmplen
9623     \addtolength\glstreeindent\parindent
9624     \parindent\glstreeindent
9625     \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`.

```
9626     \@ifundefined{@glswidestname\romannumeral\@gls@prevlevel}{%
9627     \settowidth{\glstreeindent}{\glstreenamfmt{%
9628     \@glswidestname\space}}}{%
9629     \settowidth{\glstreeindent}{\glstreenamfmt{%
9630     \csname @glswidestname\romannumeral\@gls@prevlevel
9631     \endcsname\space}}}{%
```

Subtract this length from the previous level's paragraph indent and set to `\glstreeindent`.

```
9632     \addtolength\parindent{-\glstreeindent}%
9633     \setlength\glstreeindent\parindent
9634     \fi
9635     \fi
```

Set the hanging indentation.

```
9636     \hangindent\glstreeindent
```

Put the name to the left of the paragraph block

```
9637     \makebox[0pt][r]{\glstreenamebox{\gls@tmplen}{%
9638     \glstreenamfmt{\glstarget{##2}{\glossentryname{##2}}}}}%
```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
9639     \ifglshassymbol{##2}{(\glossentrysymbol{##2})\space}{}%
```

Do the description followed by the description terminator and location list.

```
9640     \glossentrydesc{##2}\glspostdescription\space ##3\par
```

Set the previous level macro to the current level.

```
9641     \def\@gls@prevlevel{##1}%
9642     }%
```

Vertical gap between groups is the same as that used by indices:

```
9643     \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
9644 }
```

`almtreegroup` Like the `almtree` style but the glossary groups have headings.

```
9645 \newglossarystyle{almtreegroup}{%
```

Base it on the `glostylealmtree` style:

```
9646     \setglossarystyle{almtree}%
```

Give each group a heading.

```
9647     \renewcommand{\glsgroupheading}[1]{\par
9648     \def\@gls@prevlevel{-1}%
9649     \hangindent0pt\relax
```

```

9650     \parindent0pt\relax
9651     \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}%
9652     \par\indexspace}%
9653 }

```

alttreehypergroup The alttreehypergroup style is like the alttreegroup style, but has a set of links to the groups at the start of the glossary.

```

9654 \newglossarystyle{alttreehypergroup}{%
    Base it on the glostylealttree style:
9655   \setglossarystyle{alttree}%
    Put the navigation links in the header
9656   \renewcommand*{\glossaryheader}{%
9657     \par
9658     \def\@gls@prevlevel{-1}%
9659     \hangindent0pt\relax
9660     \parindent0pt\relax
9661     \glstreenavigationfmt{\glsnavigation}\par\indexspace}%
    Put a hypertarget at the start of each group
9662   \renewcommand*{\glsgroupheading}[1]{%
9663     \par
9664     \def\@gls@prevlevel{-1}%
9665     \hangindent0pt\relax
9666     \parindent0pt\relax
9667     \glstreegroupheaderfmt
9668     {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
9669     \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.

```
9670 \NeedsTeXFormat{LaTeX2e}
9671 \ProvidesPackage{glossaries-compatible-207}[2019/01/06 v4.42 (NLCT)]
```

AddXdyAttribute Adds an attribute in old format.

```
9672 \ifglxsindy
9673 \renewcommand*\GlsAddXdyAttribute[1]{%
9674 \edef\xdyattributes{\xdyattributes ^^J \string"#1\string"}%
9675 \expandafter\toks@\expandafter{\xdylocref}%
9676 \edef\xdylocref{\the\toks@ ^^J%
9677 (markup-locref
9678 :open \string"\string~n\string\setentrycounter
9679 {\noexpand\glscounter}%
9680 \expandafter\string\csname#1\endcsname
9681 \expandafter@gobble\string\{\string" ^^J
9682 :close \string"\expandafter@gobble\string}\string" ^^J
9683 :attr \string"#1\string")}}
```

Only has an effect before `\writeist`:

```
9684 \fi
```

sAddXdyCounters

```
9685 \renewcommand*\GlsAddXdyCounters[1]{%
9686 \GlossariesWarning{\string\GlsAddXdyCounters\space not available
9687 in compatibility mode.}%
9688 }
```

Add predefined attributes

```
9689 \GlsAddXdyAttribute{glsnumberformat}
9690 \GlsAddXdyAttribute{textrm}
9691 \GlsAddXdyAttribute{textsf}
9692 \GlsAddXdyAttribute{texttt}
9693 \GlsAddXdyAttribute{textbf}
9694 \GlsAddXdyAttribute{textmd}
9695 \GlsAddXdyAttribute{textit}
9696 \GlsAddXdyAttribute{textup}
9697 \GlsAddXdyAttribute{textsl}
```

```

9698 \GlsAddXdyAttribute{textsc}
9699 \GlsAddXdyAttribute{emph}
9700 \GlsAddXdyAttribute{glshypernumber}
9701 \GlsAddXdyAttribute{hyperrm}
9702 \GlsAddXdyAttribute{hypersf}
9703 \GlsAddXdyAttribute{hypertt}
9704 \GlsAddXdyAttribute{hyperbf}
9705 \GlsAddXdyAttribute{hypermd}
9706 \GlsAddXdyAttribute{hyperit}
9707 \GlsAddXdyAttribute{hyperup}
9708 \GlsAddXdyAttribute{hypersl}
9709 \GlsAddXdyAttribute{hypersc}
9710 \GlsAddXdyAttribute{hyperemph}

```

sAddXdyLocation Restore v2.07 definition:

```

9711 \ifglxindy
9712 \renewcommand*{\GlsAddXdyLocation}[2]{%
9713 \edef\@xdyuserlocationdefs{%
9714 \@xdyuserlocationdefs ^^J%
9715 (define-location-class \string"#1\string"^^J\space\space
9716 \space(#2))
9717 }%
9718 \edef\@xdyuserlocationnames{%
9719 \@xdyuserlocationnames^^J\space\space\space
9720 \string"#1\string"}%
9721 }
9722 \fi

```

\@do@wrglossary

```

9723 \renewcommand{\@do@wrglossary}[1]{%
  Determine whether to use xindy or makeindex syntax
9724 \ifglxindy
  Need to determine if the formatting information starts with a ( or ) indicating a range.
9725 \expandafter\@glo@check@mkidxrangechar\@glsnumberformat\@nil
9726 \def\@glo@range{}%
9727 \expandafter\if\@glo@prefix(\relax
9728 \def\@glo@range{:open-range}%
9729 \else
9730 \expandafter\if\@glo@prefix)\relax
9731 \def\@glo@range{:close-range}%
9732 \fi
9733 \fi

  Get the location and escape any special characters
9734 \protected@edef\@glslocref{\theglsentrycounter}%
9735 \@gls@checkmkidxchars\@glslocref

  Write to the glossary file using xindy syntax.
9736 \glossary[\csname glo@#1@type\endcsname]{%

```

```

9737 (indexentry :tkey (\csname glo@#1@index\endcsname)
9738   :locref \string"\@glslocref\string" %
9739   :attr \string"\@glo@suffix\string" \@glo@range
9740 )
9741 }%
9742 \else

```

Convert the format information into the format required for makeindex

```

9743 \@set@glo@numformat\@glo@numfmt\@gls@counter\@glsnumberformat

```

Write to the glossary file using makeindex syntax.

```

9744 \glossary[\csname glo@#1@type\endcsname]{%
9745 \string\glossaryentry{\csname glo@#1@index\endcsname
9746   \@gls@encapchar\@glo@numfmt}{\theglsentrycounter}}%
9747 \fi
9748 }

```

t@glo@numformat Only had 3 arguments in v2.07

```

9749 \def\@set@glo@numformat#1#2#3{%
9750   \expandafter\@glo@check@mkidxrangechar#3\@nil
9751   \protected@edef#1{%
9752     \@glo@prefix setentrycounter[] {#2}%
9753     \expandafter\string\csname\@glo@suffix\endcsname
9754   }%
9755   \@gls@checkmkidxchars#1%
9756 }

```

\writeist Redefine \writeist back to the way it was in v2.07, but change \istfile to \glswrite.

```

9757 \ifglxindy
9758   \def\writeist{%
9759     \openout\glswrite=\istfilename
9760     \write\glswrite{;; xindy style file created by the glossaries
9761       package in compatible-2.07 mode}%
9762     \write\glswrite{;; for document '\jobname' on
9763       \the\year-\the\month-\the\day}%
9764     \write\glswrite{^^J; required styles^^J}
9765     \@for\@xdystyle:=\@xdyrequiredstyles\do{%
9766       \ifx\@xdystyle\@empty
9767       \else
9768         \protected@write\glswrite{{(require
9769           \string"\@xdystyle.xdy\string")}}%
9770       \fi
9771     }%
9772     \write\glswrite{^^J%
9773       ; list of allowed attributes (number formats)^^J}%
9774     \write\glswrite{(define-attributes ((\@xdyattributes)))}%
9775     \write\glswrite{^^J; user defined alphabets^^J}%
9776     \write\glswrite{\@xdyuseralphabets}%
9777     \write\glswrite{^^J; location class definitions^^J}%
9778     \protected@edef\@gls@roman{\@roman{0}\string"

```

```

9779     \string"roman-numbers-lowercase\string" :sep \string"}}%
9780 \@onelevel@sanitize\@gls@roman
9781 \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
9782     :sep \string"}%
9783 \@onelevel@sanitize\@tmp
9784 \ifx\@tmp\@gls@roman
9785     \write\glswrite{(define-location-class
9786         \string"roman-page-numbers\string"^^J\space\space\space
9787         (\string"roman-numbers-lowercase\string")
9788         :min-range-length \@glsminrange)}%
9789 \else
9790     \write\glswrite{(define-location-class
9791         \string"roman-page-numbers\string"^^J\space\space\space
9792         (:sep "\@gls@roman")
9793         :min-range-length \@glsminrange)}%
9794 \fi
9795 \write\glswrite{(define-location-class
9796     \string"Roman-page-numbers\string"^^J\space\space\space
9797     (\string"roman-numbers-uppercase\string")
9798     :min-range-length \@glsminrange)}%
9799 \write\glswrite{(define-location-class
9800     \string"arabic-page-numbers\string"^^J\space\space\space
9801     (\string"arabic-numbers\string")
9802     :min-range-length \@glsminrange)}%
9803 \write\glswrite{(define-location-class
9804     \string"alpha-page-numbers\string"^^J\space\space\space
9805     (\string"alpha\string")
9806     :min-range-length \@glsminrange)}%
9807 \write\glswrite{(define-location-class
9808     \string"Alpha-page-numbers\string"^^J\space\space\space
9809     (\string"ALPHA\string")
9810     :min-range-length \@glsminrange)}%
9811 \write\glswrite{(define-location-class
9812     \string"Appendix-page-numbers\string"^^J\space\space\space
9813     (\string"ALPHA\string"
9814     :sep \string"\@glsAlphacompositor\string"
9815     \string"arabic-numbers\string")
9816     :min-range-length \@glsminrange)}%
9817 \write\glswrite{(define-location-class
9818     \string"arabic-section-numbers\string"^^J\space\space\space
9819     (\string"arabic-numbers\string"
9820     :sep \string"\glscompositor\string"
9821     \string"arabic-numbers\string")
9822     :min-range-length \@glsminrange)}%
9823 \write\glswrite{^^J; user defined location classes}%
9824 \write\glswrite{\@xdyuserlocationdefs}%
9825 \write\glswrite{^^J; define cross-reference class^^J}%
9826 \write\glswrite{(define-crossref-class \string"see\string"
9827     :unverified )}%

```

```

9828 \write\glswrite{(markup-crossref-list
9829   :class \string"see\string"^^J\space\space\space
9830   :open \string"\string\glsseeformat\string"
9831   :close \string"{}\string")}%
9832 \write\glswrite{^^J; define the order of the location classes}%
9833 \write\glswrite{(define-location-class-order
9834   (\@xdylocationclassorder))}%
9835 \write\glswrite{^^J; define the glossary markup^^J}%
9836 \write\glswrite{(markup-index^^J\space\space\space
9837   :open \string"\string
9838   \glossarysection[\string\glossarytoctitle]{\string
9839   \glossarytitle}\string\glossarypreamble\string~n\string\begin
9840   {theglossary}\string\glossaryheader\string~n\string" ^^J\space
9841   \space\space:close \string"\expandafter\@gobble
9842   \string%\string~n\string
9843   \end{theglossary}\string\glossarypostamble
9844   \string~n\string" ^^J\space\space\space
9845   :tree)}}%
9846 \write\glswrite{(markup-letter-group-list
9847   :sep \string"\string\glsgroupskip\string~n\string")}%
9848 \write\glswrite{(markup-indexentry
9849   :open \string"\string\relax \string\glsresetentrylist
9850   \string~n\string")}%
9851 \write\glswrite{(markup-locclass-list :open
9852   \string"\glsopenbrace\string\glossaryentrynumbers
9853   \glsopenbrace\string\relax\space \string"^^J\space\space\space
9854   :sep \string", \string"
9855   :close \string"\glsclosebrace\glsclosebrace\string")}%
9856 \write\glswrite{(markup-locref-list
9857   :sep \string"\string\delimN\space\string")}%
9858 \write\glswrite{(markup-range
9859   :sep \string"\string\delimR\space\string")}%
9860 \@onelevel@sanitize\gls@suffixF
9861 \@onelevel@sanitize\gls@suffixFF
9862 \ifx\gls@suffixF\@empty
9863 \else
9864   \write\glswrite{(markup-range
9865     :close "\gls@suffixF" :length 1 :ignore-end)}%
9866 \fi
9867 \ifx\gls@suffixFF\@empty
9868 \else
9869   \write\glswrite{(markup-range
9870     :close "\gls@suffixFF" :length 2 :ignore-end)}%
9871 \fi
9872 \write\glswrite{^^J; define format to use for locations^^J}%
9873 \write\glswrite{\@xdylocref}%
9874 \write\glswrite{^^J; define letter group list format^^J}%
9875 \write\glswrite{(markup-letter-group-list
9876   :sep \string"\string\glsgroupskip\string~n\string")}%

```



```

9877 \write\glswrite{^^J; letter group headings^^J}%
9878 \write\glswrite{(markup-letter-group
9879 :open-head \string"\string\glsgroupheading
9880 \glsopenbrace\string"^^J\space\space\space
9881 :close-head \string"\glsclosebrace\string")}%
9882 \write\glswrite{^^J; additional letter groups^^J}%
9883 \write\glswrite{\@xdylettergroups}%
9884 \write\glswrite{^^J; additional sort rules^^J}
9885 \write\glswrite{\@xdysortrules}%
9886 \noist}
9887 \else
9888 \edef\@gls@actualchar{\string?}
9889 \edef\@gls@encapchar{\string|}
9890 \edef\@gls@levelchar{\string!}
9891 \edef\@gls@quotechar{\string"}
9892 \def\writeist{\relax
9893 \openout\glswrite=\istfilename
9894 \write\glswrite{\expandafter\@gobble\string\% makeindex style file
9895 created by the glossaries package}
9896 \write\glswrite{\expandafter\@gobble\string\% for document
9897 'jobname' on \the\year-\the\month-\the\day}
9898 \write\glswrite{actual '@gls@actualchar'}
9899 \write\glswrite{encap '@gls@encapchar'}
9900 \write\glswrite{level '@gls@levelchar'}
9901 \write\glswrite{quote '@gls@quotechar'}
9902 \write\glswrite{keyword \string"\string\glossaryentry\string"}
9903 \write\glswrite{preamble \string"\string\glossarysection[\string
9904 \glossarytoctitle]{\string\glossarytitle}\string
9905 \glossarypreamble\string\n\string\begin{theglossary}\string
9906 \glossaryheader\string\n\string"}
9907 \write\glswrite{postamble \string"\string%\string\n\string
9908 \end{theglossary}\string\glossarypostamble\string\n
9909 \string"}
9910 \write\glswrite{group_skip \string"\string\glsgroupskip\string\n
9911 \string"}
9912 \write\glswrite{item_0 \string"\string%\string\n\string"}
9913 \write\glswrite{item_1 \string"\string%\string\n\string"}
9914 \write\glswrite{item_2 \string"\string%\string\n\string"}
9915 \write\glswrite{item_01 \string"\string%\string\n\string"}
9916 \write\glswrite{item_x1
9917 \string"\string\relax \string\glsresetentrylist\string\n
9918 \string"}
9919 \write\glswrite{item_12 \string"\string%\string\n\string"}
9920 \write\glswrite{item_x2
9921 \string"\string\relax \string\glsresetentrylist\string\n
9922 \string"}
9923 \write\glswrite{delim_0 \string"\string{\string
9924 \glossaryentrynumbers\string{\string\relax \string"}
9925 \write\glswrite{delim_1 \string"\string{\string

```

```

9926     \glossaryentrynumbers\string\{\string\relax \string"}
9927 \write\glswrite{delim_2 \string"\string\{\string
9928     \glossaryentrynumbers\string\{\string\relax \string"}
9929 \write\glswrite{delim_t \string"\string}\string}\string"}
9930 \write\glswrite{delim_n \string"\string\delimN \string"}
9931 \write\glswrite{delim_r \string"\string\delimR \string"}
9932 \write\glswrite{headings_flag 1}
9933 \write\glswrite{heading_prefix
9934     \string"\string\glsgroupheading\string\{\string"}
9935 \write\glswrite{heading_suffix
9936     \string"\string}\string\relax
9937     \string\glsresetentrylist \string"}
9938 \write\glswrite{symhead_positive \string"glssymbols\string"}
9939 \write\glswrite{numhead_positive \string"glnumbers\string"}
9940 \write\glswrite{page_compositor \string"glscpositor\string"}
9941 \@gls@escbsdq\gls@suffixF
9942 \@gls@escbsdq\gls@suffixFF
9943 \ifx\gls@suffixF\@empty
9944 \else
9945     \write\glswrite{suffix_2p \string"\gls@suffixF\string"}
9946 \fi
9947 \ifx\gls@suffixFF\@empty
9948 \else
9949     \write\glswrite{suffix_3p \string"\gls@suffixFF\string"}
9950 \fi
9951 \noist
9952 }
9953 \fi

```

\noist

```
9954 \renewcommand*{\noist}{\let\writeist\relax}
```

4.2 glossaries-compatible-307

```

9955 \NeedsTeXFormat{LaTeX2e}
9956 \ProvidesPackage{glossaries-compatible-307}[2019/01/06 v4.42 (NLCT)]

```

Compatibility macros for predefined glossary styles:

`atglossarystyle` Defines a compatibility glossary style.

```

9957 \newcommand{\compatglossarystyle}[2]{%
9958   \ifcsundef{@glscompstyle@#1}%
9959   {%
9960     \csdef{@glscompstyle@#1}{#2}%
9961   }%
9962   {%
9963     \PackageError{glossaries}{Glossary compatibility style ‘#1’ is already defined}{%
9964     }%
9965 }

```

Backward compatible inline style.

```
9966 \compatglossarystyle{inline}{%
9967   \renewcommand{\glossaryentryfield}[5]{%
9968     \glsinlinedopostchild
9969     \gls@inlinesep
9970     \def\glo@desc{##3}%
9971     \def\@no@post@desc{\nopostdesc}%
9972     \glsentryitem{##1}\glsinlinenameformat{##1}{##2}%
9973     \ifx\glo@desc\@no@post@desc
9974       \glsinlineemptydescformat{##4}{##5}%
9975     \else
9976       \ifstrempy{##3}%
9977         {\glsinlineemptydescformat{##4}{##5}}%
9978         {\glsinlinedescformat{##3}{##4}{##5}}%
9979     \fi
9980     \ifglshaschildren{##1}%
9981     {%
9982       \glsresetsubentrycounter
9983       \glsinlineparentchildseparator
9984       \def\gls@inlinesubsep{}%
9985       \def\gls@inlinepostchild{\glsinlinepostchild}%
9986     }%
9987   }%
9988   \def\gls@inlinesep{\glsinlineseparator}%
9989 }%
```

Sub-entries display description:

```
9990 \renewcommand{\glossarysubentryfield}[6]{%
9991   \gls@inlinesubsep%
9992   \glsinlinesubnameformat{##2}{##3}%
9993   \glssubentryitem{##2}\glsinlinesubdescformat{##4}{##5}{##6}%
9994   \def\gls@inlinesubsep{\glsinlinesubseparator}%
9995 }%
9996 }
```

Backward compatible list style.

```
9997 \compatglossarystyle{list}{%
9998   \renewcommand*{\glossaryentryfield}[5]{%
9999     \item[\glsentryitem{##1}\glstarget{##1}{##2}]
10000     ##3\glspostdescription\space ##5}%

```

Sub-entries continue on the same line:

```
10001 \renewcommand*{\glossarysubentryfield}[6]{%
10002   \glssubentryitem{##2}%
10003   \glstarget{##2}{\strut}##4\glspostdescription\space ##6.}%
10004 }
```

Backward compatible listgroup style.

```
10005 \compatglossarystyle{listgroup}{%
10006   \csuse{@glscompstyle@list}%
10007 }%
```

Backward compatible listhypergroup style.

```
10008 \compatglossarystyle{listhypergroup}{%
10009 \csuse{@glscompstyle@list}%
10010 }%
```

Backward compatible altlist style.

```
10011 \compatglossarystyle{altlist}{%
10012 \renewcommand*{\glossaryentryfield}[5]{%
10013 \item[\glsentryitem{##1}\glstarget{##1}{##2}]%
10014 \mbox{}\par\nobreak\@afterheading
10015 ##3\glspostdescription\space ##5}%
10016 \renewcommand{\glossarysubentryfield}[6]{%
10017 \par
10018 \glssubentryitem{##2}%
10019 \glstarget{##2}{\strut}##4\glspostdescription\space ##6}%
10020 }%
```

Backward compatible altlistgroup style.

```
10021 \compatglossarystyle{altlistgroup}{%
10022 \csuse{@glscompstyle@altlist}%
10023 }%
```

Backward compatible altlisthypergroup style.

```
10024 \compatglossarystyle{altlisthypergroup}{%
10025 \csuse{@glscompstyle@altlist}%
10026 }%
```

Backward compatible listdotted style.

```
10027 \compatglossarystyle{listdotted}{%
10028 \renewcommand*{\glossaryentryfield}[5]{%
10029 \item[]\makebox[\glslistdottedwidth][l]{%
10030 \glsentryitem{##1}\glstarget{##1}{##2}%
10031 \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##3}%
10032 \renewcommand*{\glossarysubentryfield}[6]{%
10033 \item[]\makebox[\glslistdottedwidth][l]{%
10034 \glssubentryitem{##2}%
10035 \glstarget{##2}{##3}%
10036 \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##4}%
10037 }%
```

Backward compatible sublistdotted style.

```
10038 \compatglossarystyle{sublistdotted}{%
10039 \csuse{@glscompstyle@listdotted}%
10040 \renewcommand*{\glossaryentryfield}[5]{%
10041 \item[\glsentryitem{##1}\glstarget{##1}{##2}]}%
10042 }%
```

Backward compatible long style.

```
10043 \compatglossarystyle{long}{%
10044 \renewcommand*{\glossaryentryfield}[5]{%
10045 \glsentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\}%
10046 \renewcommand*{\glossarysubentryfield}[6]{%

```

```

10047      &
10048      \glssubentryitem{##2}%
10049      \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
10050 }%

```

Backward compatible longborder style.

```

10051 \compatglossarystyle{longborder}{%
10052 \csuse{@glscompstyle@long}%
10053 }%

```

Backward compatible longheader style.

```

10054 \compatglossarystyle{longheader}{%
10055 \csuse{@glscompstyle@long}%
10056 }%

```

Backward compatible longheaderborder style.

```

10057 \compatglossarystyle{longheaderborder}{%
10058 \csuse{@glscompstyle@long}%
10059 }%

```

Backward compatible long3col style.

```

10060 \compatglossarystyle{long3col}{%
10061 \renewcommand*{\glossaryentryfield}[5]{%
10062 \glstarget{##1}\glstarget{##1}{##2} & ##3 & ##5\\}%
10063 \renewcommand*{\glossarysubentryfield}[6]{%
10064 &
10065 \glssubentryitem{##2}%
10066 \glstarget{##2}{\strut}##4 & ##6\\}%
10067 }%

```

Backward compatible long3colborder style.

```

10068 \compatglossarystyle{long3colborder}{%
10069 \csuse{@glscompstyle@long3col}%
10070 }%

```

Backward compatible long3colheader style.

```

10071 \compatglossarystyle{long3colheader}{%
10072 \csuse{@glscompstyle@long3col}%
10073 }%

```

Backward compatible long3colheaderborder style.

```

10074 \compatglossarystyle{long3colheaderborder}{%
10075 \csuse{@glscompstyle@long3col}%
10076 }%

```

Backward compatible long4col style.

```

10077 \compatglossarystyle{long4col}{%
10078 \renewcommand*{\glossaryentryfield}[5]{%
10079 \glstarget{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%
10080 \renewcommand*{\glossarysubentryfield}[6]{%
10081 &
10082 \glssubentryitem{##2}%

```

```

10083 \glstarget{##2}{\strut}##4 & ##5 & ##6\}%
10084 }%

Backward compatible long4colheader style.
10085 \compatglossarystyle{long4colheader}{%
10086 \csuse{@glscompstyle@long4col}%
10087 }%

Backward compatible long4colborder style.
10088 \compatglossarystyle{long4colborder}{%
10089 \csuse{@glscompstyle@long4col}%
10090 }%

Backward compatible long4colheaderborder style.
10091 \compatglossarystyle{long4colheaderborder}{%
10092 \csuse{@glscompstyle@long4col}%
10093 }%

Backward compatible altlong4col style.
10094 \compatglossarystyle{altlong4col}{%
10095 \csuse{@glscompstyle@long4col}%
10096 }%

Backward compatible altlong4colheader style.
10097 \compatglossarystyle{altlong4colheader}{%
10098 \csuse{@glscompstyle@long4col}%
10099 }%

Backward compatible altlong4colborder style.
10100 \compatglossarystyle{altlong4colborder}{%
10101 \csuse{@glscompstyle@long4col}%
10102 }%

Backward compatible altlong4colheaderborder style.
10103 \compatglossarystyle{altlong4colheaderborder}{%
10104 \csuse{@glscompstyle@long4col}%
10105 }%

Backward compatible long style.
10106 \compatglossarystyle{longragged}{%
10107 \renewcommand*{\glossaryentryfield}[5]{%
10108 \glstarget{##1}{\strut}##4 & ##3\glspostdescription\space ##5%
10109 \tabularnewline}%
10110 \renewcommand*{\glossarysubentryfield}[6]{%
10111 &
10112 \glssubentryitem{##2}%
10113 \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
10114 \tabularnewline}%
10115 }%

Backward compatible longraggedborder style.
10116 \compatglossarystyle{longraggedborder}{%
10117 \csuse{@glscompstyle@longragged}%
10118 }%

```

Backward compatible longraggedheader style.

```
10119 \compatglossarystyle{longraggedheader}{%
10120 \csuse{@glscompstyle@longragged}%
10121 }%
```

Backward compatible longraggedheaderborder style.

```
10122 \compatglossarystyle{longraggedheaderborder}{%
10123 \csuse{@glscompstyle@longragged}%
10124 }%
```

Backward compatible longragged3col style.

```
10125 \compatglossarystyle{longragged3col}{%
10126 \renewcommand*{\glossaryentryfield}[5]{%
10127 \glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%
10128 \renewcommand*{\glossarysubentryfield}[6]{%
10129 &
10130 \glssubentryitem{##2}%
10131 \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%
10132 }%
```

Backward compatible longragged3colborder style.

```
10133 \compatglossarystyle{longragged3colborder}{%
10134 \csuse{@glscompstyle@longragged3col}%
10135 }%
```

Backward compatible longragged3colheader style.

```
10136 \compatglossarystyle{longragged3colheader}{%
10137 \csuse{@glscompstyle@longragged3col}%
10138 }%
```

Backward compatible longragged3colheaderborder style.

```
10139 \compatglossarystyle{longragged3colheaderborder}{%
10140 \csuse{@glscompstyle@longragged3col}%
10141 }%
```

Backward compatible altlongragged4col style.

```
10142 \compatglossarystyle{altlongragged4col}{%
10143 \renewcommand*{\glossaryentryfield}[5]{%
10144 \glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%
10145 \renewcommand*{\glossarysubentryfield}[6]{%
10146 &
10147 \glssubentryitem{##2}%
10148 \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%
10149 }%
```

Backward compatible altlongragged4colheader style.

```
10150 \compatglossarystyle{altlongragged4colheader}{%
10151 \csuse{@glscompstyle@altlong4col}%
10152 }%
```

Backward compatible altlongragged4colborder style.

```
10153 \compatglossarystyle{altlongragged4colborder}{%
```

```
10154 \csuse{@glscompstyle@altlong4col}%
10155 }%
```

Backward compatible altlongragged4colheaderborder style.

```
10156 \compatglossarystyle{altlongragged4colheaderborder}{%
10157 \csuse{@glscompstyle@altlong4col}%
10158 }%
```

Backward compatible index style.

```
10159 \compatglossarystyle{index}{%
10160 \renewcommand*\glossaryentryfield}[5]{%
10161 \item\glsentryitem{##1}\textbf{\glstarget{##1}{##2}}%
10162 \ifx\relax##4\relax
10163 \else
10164 \space{##4}%
10165 \fi
10166 \space ##3\glspostdescription \space ##5}%
10167 \renewcommand*\glossarysubentryfield}[6]{%
10168 \ifcase##1\relax
10169 % level 0
10170 \item
10171 \or
10172 % level 1
10173 \subitem
10174 \glssubentryitem{##2}%
10175 \else
10176 % all other levels
10177 \subsubitem
10178 \fi
10179 \textbf{\glstarget{##2}{##3}}%
10180 \ifx\relax##5\relax
10181 \else
10182 \space{##5}%
10183 \fi
10184 \space##4\glspostdescription\space ##6}%
10185 }%
```

Backward compatible indexgroup style.

```
10186 \compatglossarystyle{indexgroup}{%
10187 \csuse{@glscompstyle@index}%
10188 }%
```

Backward compatible indexhypergroup style.

```
10189 \compatglossarystyle{indexhypergroup}{%
10190 \csuse{@glscompstyle@index}%
10191 }%
```

Backward compatible tree style.

```
10192 \compatglossarystyle{tree}{%
10193 \renewcommand*\glossaryentryfield}[5]{%
10194 \hangindent0pt\relax
```



```

10195 \parindent0pt\relax
10196 \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
10197 \ifx\relax##4\relax
10198 \else
10199 \space{##4}%
10200 \fi
10201 \space ##3\glspostdescription \space ##5\par}%
10202 \renewcommand{\glossarysubentryfield}[6]{%
10203 \hangindent##1\glstreeindent\relax
10204 \parindent##1\glstreeindent\relax
10205 \ifnum##1=1\relax
10206 \glssubentryitem{##2}%
10207 \fi
10208 \textbf{\glstarget{##2}{##3}}%
10209 \ifx\relax##5\relax
10210 \else
10211 \space{##5}%
10212 \fi
10213 \space##4\glspostdescription\space ##6\par}%
10214 }%

```

Backward compatible treegroup style.

```

10215 \compatglossarystyle{treegroup}{%
10216 \csuse{@glscompstyle@tree}%
10217 }%

```

Backward compatible treehypergroup style.

```

10218 \compatglossarystyle{treehypergroup}{%
10219 \csuse{@glscompstyle@tree}%
10220 }%

```

Backward compatible treenoname style.

```

10221 \compatglossarystyle{treenoname}{%
10222 \renewcommand{\glossaryentryfield}[5]{%
10223 \hangindent0pt\relax
10224 \parindent0pt\relax
10225 \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
10226 \ifx\relax##4\relax
10227 \else
10228 \space{##4}%
10229 \fi
10230 \space ##3\glspostdescription \space ##5\par}%
10231 \renewcommand{\glossarysubentryfield}[6]{%
10232 \hangindent##1\glstreeindent\relax
10233 \parindent##1\glstreeindent\relax
10234 \ifnum##1=1\relax
10235 \glssubentryitem{##2}%
10236 \fi
10237 \glstarget{##2}{\strut}%
10238 ##4\glspostdescription\space ##6\par}%
10239 }%

```

Backward compatible treenonamegroup style.

```
10240 \compatglossarystyle{treenonamegroup}{%
10241 \csuse{@glscompstyle@treenoname}%
10242 }%
```

Backward compatible treenonamehypergroup style.

```
10243 \compatglossarystyle{treenonamehypergroup}{%
10244 \csuse{@glscompstyle@treenoname}%
10245 }%
```

Backward compatible altree style.

```
10246 \compatglossarystyle{almtree}{%
10247 \renewcommand{\glossaryentryfield}[5]{%
10248 \ifnum\@gls@prevlevel=0\relax
10249 \else
10250 \settowidth{\glstreeindent}{\textbf{\@glswidestname\space}}%
10251 \hangindent\glstreeindent
10252 \parindent\glstreeindent
10253 \fi
10254 \makebox[0pt][r]{\makebox[\glstreeindent][l]{%
10255 \glsentryitem{##1}\textbf{\glstarget{##1}{##2}}}}%
10256 \ifx\relax##4\relax
10257 \else
10258 (##4)\space
10259 \fi
10260 ##3\glspostdescription \space ##5\par
10261 \def\@gls@prevlevel{0}%
10262 }%
10263 \renewcommand{\glossarysubentryfield}[6]{%
10264 \ifnum##1=1\relax
10265 \glsentryitem{##2}%
10266 \fi
10267 \ifnum\@gls@prevlevel=##1\relax
10268 \else
10269 \@ifundefined{@glswidestname\romannumeral##1}{%
10270 \settowidth{\gls@tmplen}{\textbf{\@glswidestname\space}}{%
10271 \settowidth{\gls@tmplen}{\textbf{%
10272 \csname @glswidestname\romannumeral##1\endcsname\space}}}%
10273 \ifnum\@gls@prevlevel<##1\relax
10274 \setlength\glstreeindent\gls@tmplen
10275 \addtolength\glstreeindent\parindent
10276 \parindent\glstreeindent
10277 \else
10278 \@ifundefined{@glswidestname\romannumeral\@gls@prevlevel}{%
10279 \settowidth{\glstreeindent}{\textbf{%
10280 \@glswidestname\space}}{%
10281 \settowidth{\glstreeindent}{\textbf{%
10282 \csname @glswidestname\romannumeral\@gls@prevlevel
10283 \endcsname\space}}}%
10284 \addtolength\parindent{-\glstreeindent}}%
```

```

10285     \setlength\glstreeindent\parindent
10286     \fi
10287     \fi
10288     \hangindent\glstreeindent
10289     \makebox[0pt][r]{\makebox[\gls@tmplen][l]{%
10290       \textbf{\glstarget{##2}{##3}}}}%
10291     \ifx##5\relax\relax
10292     \else
10293       (##5)\space
10294     \fi
10295     ##4\glspostdescription\space ##6\par
10296     \def\@gls@prevlevel{##1}%
10297   }%
10298 }%

```

Backward compatible alttreegroup style.

```

10299 \compatglossarystyle{alttreegroup}{%
10300 \csuse{@glscompstyle@almtree}%
10301 }%

```

Backward compatible alttreehypergroup style.

```

10302 \compatglossarystyle{alttreehypergroup}{%
10303 \csuse{@glscompstyle@almtree}%
10304 }%

```

Backward compatible mcolindex style.

```

10305 \compatglossarystyle{mcolindex}{%
10306 \csuse{@glscompstyle@index}%
10307 }%

```

Backward compatible mcolindexgroup style.

```

10308 \compatglossarystyle{mcolindexgroup}{%
10309 \csuse{@glscompstyle@index}%
10310 }%

```

Backward compatible mcolindexhypergroup style.

```

10311 \compatglossarystyle{mcolindexhypergroup}{%
10312 \csuse{@glscompstyle@index}%
10313 }%

```

Backward compatible mcoltree style.

```

10314 \compatglossarystyle{mcoltree}{%
10315 \csuse{@glscompstyle@tree}%
10316 }%

```

Backward compatible mcoltreegroup style.

```

10317 \compatglossarystyle{mcolindextreegroup}{%
10318 \csuse{@glscompstyle@tree}%
10319 }%

```

Backward compatible mcoltreehypergroup style.

```

10320 \compatglossarystyle{mcolindextreehypergroup}{%

```

```

10321 \csuse{@glscompstyle@tree}%
10322 }%

    Backward compatible mcoltreenoname style.
10323 \compatglossarystyle{mcoltreenoname}{%
10324 \csuse{@glscompstyle@tree}%
10325 }%

    Backward compatible mcoltreenonamegroup style.
10326 \compatglossarystyle{mcoltreenonamegroup}{%
10327 \csuse{@glscompstyle@tree}%
10328 }%

    Backward compatible mcoltreenonamehypergroup style.
10329 \compatglossarystyle{mcoltreenonamehypergroup}{%
10330 \csuse{@glscompstyle@tree}%
10331 }%

    Backward compatible mcolalmtree style.
10332 \compatglossarystyle{mcolalmtree}{%
10333 \csuse{@glscompstyle@almtree}%
10334 }%

    Backward compatible mcolalmtreegroup style.
10335 \compatglossarystyle{mcolalmtreegroup}{%
10336 \csuse{@glscompstyle@almtree}%
10337 }%

    Backward compatible mcolalmtreehypergroup style.
10338 \compatglossarystyle{mcolalmtreehypergroup}{%
10339 \csuse{@glscompstyle@almtree}%
10340 }%

    Backward compatible superragged style.
10341 \compatglossarystyle{superragged}{%
10342 \renewcommand*{\glossaryentryfield}[5]{%
10343 \glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
10344 \tabularnewline}%
10345 \renewcommand*{\glossarysubentryfield}[6]{%
10346 &
10347 \glssubentryitem{##2}%
10348 \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
10349 \tabularnewline}%
10350 }%

    Backward compatible superraggedborder style.
10351 \compatglossarystyle{superraggedborder}{%
10352 \csuse{@glscompstyle@superragged}%
10353 }%

    Backward compatible superraggedheader style.
10354 \compatglossarystyle{superraggedheader}{%
10355 \csuse{@glscompstyle@superragged}%
10356 }%

```

Backward compatible superraggedheaderborder style.

```
10357 \compatglossarystyle{superraggedheaderborder}{%
10358 \csuse{@glscompstyle@superragged}%
10359 }%
```

Backward compatible superragged3col style.

```
10360 \compatglossarystyle{superragged3col}{%
10361 \renewcommand*{\glossaryentryfield}[5]{%
10362 \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%
10363 \renewcommand*{\glossarysubentryfield}[6]{%
10364 &
10365 \glssubentryitem{##2}%
10366 \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%
10367 }%
```

Backward compatible superragged3colborder style.

```
10368 \compatglossarystyle{superragged3colborder}{%
10369 \csuse{@glscompstyle@superragged3col}%
10370 }%
```

Backward compatible superragged3colheader style.

```
10371 \compatglossarystyle{superragged3colheader}{%
10372 \csuse{@glscompstyle@superragged3col}%
10373 }%
```

Backward compatible superragged3colheaderborder style.

```
10374 \compatglossarystyle{superragged3colheaderborder}{%
10375 \csuse{@glscompstyle@superragged3col}%
10376 }%
```

Backward compatible altsuperragged4col style.

```
10377 \compatglossarystyle{altsuperragged4col}{%
10378 \renewcommand*{\glossaryentryfield}[5]{%
10379 \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%
10380 \renewcommand*{\glossarysubentryfield}[6]{%
10381 &
10382 \glssubentryitem{##2}%
10383 \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%
10384 }%
```

Backward compatible altsuperragged4colheader style.

```
10385 \compatglossarystyle{altsuperragged4colheader}{%
10386 \csuse{@glscompstyle@altsuperragged4col}%
10387 }%
```

Backward compatible altsuperragged4colborder style.

```
10388 \compatglossarystyle{altsuperragged4colborder}{%
10389 \csuse{@glscompstyle@altsuperragged4col}%
10390 }%
```

Backward compatible altsuperragged4colheaderborder style.

```
10391 \compatglossarystyle{altsuperragged4colheaderborder}{%
```

```
10392 \csuse{@glscompstyle@altsuperragged4col}%
10393 }%
```

Backward compatible super style.

```
10394 \compatglossarystyle{super}{%
10395   \renewcommand*{\glossaryentryfield}[5]{%
10396     \glstarget{##1}{##2} & ##3\glspostdescription\space ##5\}%
10397   \renewcommand*{\glossarysubentryfield}[6]{%
10398     &
10399     \glssubentryitem{##2}%
10400     \glstarget{##2}{\strut}##4\glspostdescription\space ##6\}%
10401 }%
```

Backward compatible superborder style.

```
10402 \compatglossarystyle{superborder}{%
10403   \csuse{@glscompstyle@super}%
10404 }%
```

Backward compatible superheader style.

```
10405 \compatglossarystyle{superheader}{%
10406   \csuse{@glscompstyle@super}%
10407 }%
```

Backward compatible superheaderborder style.

```
10408 \compatglossarystyle{superheaderborder}{%
10409   \csuse{@glscompstyle@super}%
10410 }%
```

Backward compatible super3col style.

```
10411 \compatglossarystyle{super3col}{%
10412   \renewcommand*{\glossaryentryfield}[5]{%
10413     \glstarget{##1}{##2} & ##3 & ##5\}%
10414   \renewcommand*{\glossarysubentryfield}[6]{%
10415     &
10416     \glssubentryitem{##2}%
10417     \glstarget{##2}{\strut}##4 & ##6\}%
10418 }%
```

Backward compatible super3colborder style.

```
10419 \compatglossarystyle{super3colborder}{%
10420   \csuse{@glscompstyle@super3col}%
10421 }%
```

Backward compatible super3colheader style.

```
10422 \compatglossarystyle{super3colheader}{%
10423   \csuse{@glscompstyle@super3col}%
10424 }%
```

Backward compatible super3colheaderborder style.

```
10425 \compatglossarystyle{super3colheaderborder}{%
10426   \csuse{@glscompstyle@super3col}%
10427 }%
```

Backward compatible super4col style.

```
10428 \compatglossarystyle{super4col}{%
10429   \renewcommand*\glossaryentryfield}[5]{%
10430     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\}%
10431   \renewcommand*\glossarysubentryfield}[6]{%
10432     &
10433     \glssubentryitem{##2}%
10434     \glstarget{##2}{\strut}##4 & ##5 & ##6\}%
10435 }%
```

Backward compatible super4colheader style.

```
10436 \compatglossarystyle{super4colheader}{%
10437   \csuse{@glscompstyle@super4col}%
10438 }%
```

Backward compatible super4colborder style.

```
10439 \compatglossarystyle{super4colborder}{%
10440   \csuse{@glscompstyle@super4col}%
10441 }%
```

Backward compatible super4colheaderborder style.

```
10442 \compatglossarystyle{super4colheaderborder}{%
10443   \csuse{@glscompstyle@super4col}%
10444 }%
```

Backward compatible altsuper4col style.

```
10445 \compatglossarystyle{altsuper4col}{%
10446   \csuse{@glscompstyle@super4col}%
10447 }%
```

Backward compatible altsuper4colheader style.

```
10448 \compatglossarystyle{altsuper4colheader}{%
10449   \csuse{@glscompstyle@super4col}%
10450 }%
```

Backward compatible altsuper4colborder style.

```
10451 \compatglossarystyle{altsuper4colborder}{%
10452   \csuse{@glscompstyle@super4col}%
10453 }%
```

Backward compatible altsuper4colheaderborder style.

```
10454 \compatglossarystyle{altsuper4colheaderborder}{%
10455   \csuse{@glscompstyle@super4col}%
10456 }%
```

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.

```
10457 \NeedsTeXFormat{LaTeX2e}
```

Package version number now in line with main glossaries package number.

```
10458 \ProvidesPackage{glossaries-accsupp}[2019/01/06 v4.42 (NLCT)
```

```
10459 Experimental glossaries accessibility]
```

Pass all options to glossaries:

```
10460 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
10461 \ProcessOptions
```

This package should be loaded before glossaries-extra, so complain if that has already been loaded.

```
10462 \@ifpackageloaded{glossaries-extra}
```

```
10463 {%
```

If the accsupp option was used, `\@glsxtr@doaccsupp` will have been set, otherwise it will be empty.

```
10464 \ifx\@glsxtr@doaccsupp\empty
```

```
10465 \GlossariesWarning{The ‘glossaries-accsupp’
```

```
10466 package has been loaded\MessageBreak
```

```
10467 after the ‘glossaries-extra’ package. This\MessageBreak
```

```
10468 can cause a failure to integrate both packages. \MessageBreak
```

```
10469 Either use the ‘accsupp’ option when you load\MessageBreak
```

```
10470 ‘glossaries-extra’ or load ‘glossaries-accsupp’\MessageBreak
```

```
10471 before loading ‘glossaries-extra’}%
```

```
10472 \fi
```

```
10473 }
```

```
10474 {}
```

`tibleglossentry` Override style compatibility macros:

```
10475 \def\compatibleglossentry#1#2{%
```

```
10476 \toks@{#2}%
```

```
10477 \protected@edef\do@glossentry{%
```

```
10478 \noexpand\accsuppglossaryentryfield{#1}%
```

```
10479 {\noexpand\glsnamefont
```

```
10480 {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@name\endcsname}}%
```



```

10481     {\expandafter\expandonce\csname glo@glstetoklabel{#1}@desc\endcsname}%
10482     {\expandafter\expandonce\csname glo@glstetoklabel{#1}@symbol\endcsname}%
10483     {\the\toks@}%
10484     }%
10485     \@do@glossentry
10486 }

```

lesubglossentry

```

10487 \def\compatiblesubglossentry#1#2#3{%
10488   \toks@{#3}%
10489   \protected@edef\do@subglossentry{%
10490     \noexpand\accsuppglossarysubentryfield{\number#1}%
10491     {#2}%
10492     {\noexpand\glsnamefont
10493       {\expandafter\expandonce\csname glo@glstetoklabel{#2}@name\endcsname}}%
10494     {\expandafter\expandonce\csname glo@glstetoklabel{#2}@desc\endcsname}%
10495     {\expandafter\expandonce\csname glo@glstetoklabel{#2}@symbol\endcsname}%
10496     {\the\toks@}%
10497   }%
10498   \@do@subglossentry
10499 }

```

Required packages:

```

10500 \RequirePackage{glossaries}
10501 \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:

```

10502 \define@key{glossentry}{access}{%
10503   \def\@glo@access{#1}%
10504 }

```

textaccess The replacement text corresponding to the text key:

```

10505 \define@key{glossentry}{textaccess}{%
10506   \def\@glo@textaccess{#1}%
10507 }

```

firstaccess The replacement text corresponding to the first key:

```

10508 \define@key{glossentry}{firstaccess}{%
10509   \def\@glo@firstaccess{#1}%
10510 }

```

pluralaccess The replacement text corresponding to the plural key:

```
10511 \define@key{glossentry}{pluralaccess}{%
10512 \def\@glo@pluralaccess{#1}%
10513 }
```

firstpluralaccess The replacement text corresponding to the firstplural key:

```
10514 \define@key{glossentry}{firstpluralaccess}{%
10515 \def\@glo@firstpluralaccess{#1}%
10516 }
```

symbolaccess The replacement text corresponding to the symbol key:

```
10517 \define@key{glossentry}{symbolaccess}{%
10518 \def\@glo@symbolaccess{#1}%
10519 }
```

symbolpluralaccess The replacement text corresponding to the symbolplural key:

```
10520 \define@key{glossentry}{symbolpluralaccess}{%
10521 \def\@glo@symbolpluralaccess{#1}%
10522 }
```

descriptionaccess The replacement text corresponding to the description key:

```
10523 \define@key{glossentry}{descriptionaccess}{%
10524 \def\@glo@descaccess{#1}%
10525 }
```

descriptionpluralaccess The replacement text corresponding to the descriptionplural key:

```
10526 \define@key{glossentry}{descriptionpluralaccess}{%
10527 \def\@glo@descpluralaccess{#1}%
10528 }
```

shortaccess The replacement text corresponding to the short key:

```
10529 \define@key{glossentry}{shortaccess}{%
10530 \def\@glo@shortaccess{#1}%
10531 }
```

shortpluralaccess The replacement text corresponding to the shortplural key:

```
10532 \define@key{glossentry}{shortpluralaccess}{%
10533 \def\@glo@shortpluralaccess{#1}%
10534 }
```

longaccess The replacement text corresponding to the long key:

```
10535 \define@key{glossentry}{longaccess}{%
10536 \def\@glo@longaccess{#1}%
10537 }
```

longpluralaccess The replacement text corresponding to the longplural key:

```
10538 \define@key{glossentry}{longpluralaccess}{%
10539 \def\@glo@longpluralaccess{#1}%
10540 }
```

There are no equivalent keys for the user1...user6 keys. The replacement text would have to be explicitly put in the value, e.g., user1={\glsaccsupp{inches}{in}}.

Append these new keys to \@gls@keymap:

```

10541 \appto\@gls@keymap{,%
10542   {access}{access},%
10543   {textaccess}{textaccess},%
10544   {firstaccess}{firstaccess},%
10545   {pluralaccess}{pluralaccess},%
10546   {firstpluralaccess}{firstpluralaccess},%
10547   {symbolaccess}{symbolaccess},%
10548   {symbolpluralaccess}{symbolpluralaccess},%
10549   {descaccess}{descaccess},%
10550   {descpluralaccess}{descpluralaccess},%
10551   {shortaccess}{shortaccess},%
10552   {shortpluralaccess}{shortpluralaccess},%
10553   {longaccess}{longaccess},%
10554   {longpluralaccess}{longpluralaccess}%
10555 }

```

\@gls@noaccess Indicates that no replacement text has been provided.

```

10556 \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):

```

10557 \let\@gls@oldnewglossaryentryprehook\@newglossaryentryprehook
10558 \renewcommand*{\@newglossaryentryprehook}{%
10559   \@gls@oldnewglossaryentryprehook
10560   \def\@glo@access{\@glo@symbol}%

```

Initialise the other keys:

```

10561   \def\@glo@textaccess{\@glo@access}%
10562   \def\@glo@firstaccess{\@glo@access}%
10563   \def\@glo@pluralaccess{\@glo@textaccess}%
10564   \def\@glo@firstpluralaccess{\@glo@pluralaccess}%
10565   \def\@glo@symbolaccess{\relax}%
10566   \def\@glo@symbolpluralaccess{\@glo@symbolaccess}%
10567   \def\@glo@descaccess{\relax}%
10568   \def\@glo@descpluralaccess{\@glo@descaccess}%
10569   \def\@glo@shortaccess{\relax}%
10570   \def\@glo@shortpluralaccess{\@glo@shortaccess}%
10571   \def\@glo@longaccess{\relax}%
10572   \def\@glo@longpluralaccess{\@glo@longaccess}%
10573 }

```

Add to the end hook:

```

10574 \let\@gls@oldnewglossaryentryposthook\@newglossaryentryposthook
10575 \renewcommand*{\@newglossaryentryposthook}{%
10576   \@gls@oldnewglossaryentryposthook

```

Store the access information:

```
10577 \expandafter
10578   \protected@xdef\csname glo@\@glo@label @access\endcsname{%
10579     \@glo@access}%
10580 \expandafter
10581   \protected@xdef\csname glo@\@glo@label @textaccess\endcsname{%
10582     \@glo@textaccess}%
10583 \expandafter
10584   \protected@xdef\csname glo@\@glo@label @firstaccess\endcsname{%
10585     \@glo@firstaccess}%
10586 \expandafter
10587   \protected@xdef\csname glo@\@glo@label @pluralaccess\endcsname{%
10588     \@glo@pluralaccess}%
10589 \expandafter
10590   \protected@xdef\csname glo@\@glo@label @firstpluralaccess\endcsname{%
10591     \@glo@firstpluralaccess}%
10592 \expandafter
10593   \protected@xdef\csname glo@\@glo@label @symbolaccess\endcsname{%
10594     \@glo@symbolaccess}%
10595 \expandafter
10596   \protected@xdef\csname glo@\@glo@label @symbolpluralaccess\endcsname{%
10597     \@glo@symbolpluralaccess}%
10598 \expandafter
10599   \protected@xdef\csname glo@\@glo@label @descaccess\endcsname{%
10600     \@glo@descaccess}%
10601 \expandafter
10602   \protected@xdef\csname glo@\@glo@label @descpluralaccess\endcsname{%
10603     \@glo@descpluralaccess}%
10604 \expandafter
10605   \protected@xdef\csname glo@\@glo@label @shortaccess\endcsname{%
10606     \@glo@shortaccess}%
10607 \expandafter
10608   \protected@xdef\csname glo@\@glo@label @shortpluralaccess\endcsname{%
10609     \@glo@shortpluralaccess}%
10610 \expandafter
10611   \protected@xdef\csname glo@\@glo@label @longaccess\endcsname{%
10612     \@glo@longaccess}%
10613 \expandafter
10614   \protected@xdef\csname glo@\@glo@label @longpluralaccess\endcsname{%
10615     \@glo@longpluralaccess}%
10616 }
```

5.2 Accessing Replacement Text

`\glsentryaccess` Get the value of the access key for the entry with the given label:

```
10617 \newcommand*{\glsentryaccess}[1]{%
10618   \@gls@entry@field{#1}{access}%
10619 }
```

entrytextaccess Get the value of the textaccess key for the entry with the given label:

```
10620 \newcommand*{\glsentrytextaccess}[1]{%
10621 \@gls@entry@field{#1}{textaccess}%
10622 }
```

entryfirstaccess Get the value of the firstaccess key for the entry with the given label:

```
10623 \newcommand*{\glsentryfirstaccess}[1]{%
10624 \@gls@entry@field{#1}{firstaccess}%
10625 }
```

entrypluralaccess Get the value of the pluralaccess key for the entry with the given label:

```
10626 \newcommand*{\glsentrypluralaccess}[1]{%
10627 \@gls@entry@field{#1}{pluralaccess}%
10628 }
```

entryfirstpluralaccess Get the value of the firstpluralaccess key for the entry with the given label:

```
10629 \newcommand*{\glsentryfirstpluralaccess}[1]{%
10630 \csname glo@#1@firstpluralaccess\endcsname
10631 }
```

entrysymbolaccess Get the value of the symbolaccess key for the entry with the given label:

```
10632 \newcommand*{\glsentrysymbolaccess}[1]{%
10633 \@gls@entry@field{#1}{symbolaccess}%
10634 }
```

entrysymbolpluralaccess Get the value of the symbolpluralaccess key for the entry with the given label:

```
10635 \newcommand*{\glsentrysymbolpluralaccess}[1]{%
10636 \@gls@entry@field{#1}{symbolpluralaccess}%
10637 }
```

entrydescaccess Get the value of the descriptionaccess key for the entry with the given label:

```
10638 \newcommand*{\glsentrydescaccess}[1]{%
10639 \@gls@entry@field{#1}{descaccess}%
10640 }
```

entrydescpluralaccess Get the value of the descriptionpluralaccess key for the entry with the given label:

```
10641 \newcommand*{\glsentrydescpluralaccess}[1]{%
10642 \@gls@entry@field{#1}{descaccess}%
10643 }
```

entryshortaccess Get the value of the shortaccess key for the entry with the given label:

```
10644 \newcommand*{\glsentryshortaccess}[1]{%
10645 \@gls@entry@field{#1}{shortaccess}%
10646 }
```

entryshortpluralaccess Get the value of the shortpluralaccess key for the entry with the given label:

```
10647 \newcommand*{\glsentryshortpluralaccess}[1]{%
10648 \@gls@entry@field{#1}{shortpluralaccess}%
10649 }
```

`entrylongaccess` Get the value of the longaccess key for the entry with the given label:

```
10650 \newcommand*{\glsentrylongaccess}[1]{%
10651   \@gls@entry@field{#1}{longaccess}%
10652 }
```

`ongpluralaccess` Get the value of the longpluralaccess key for the entry with the given label:

```
10653 \newcommand*{\glsentrylongpluralaccess}[1]{%
10654   \@gls@entry@field{#1}{longpluralaccess}%
10655 }
```

```
\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.)

```
10656 \newcommand*{\glsaccsupp}[2]{%
10657   \BeginAccSupp{ActualText={#1}}#2\EndAccSupp{}%
10658 }
```

`\xglsaccsupp` Fully expands replacement text before calling `\glsaccsupp`

```
10659 \newcommand*{\xglsaccsupp}[2]{%
10660   \protected@edef\@gls@replacementtext{#1}%
10661   \expandafter\glsaccsupp\expandafter{\@gls@replacementtext}{#2}%
10662 }
```

`@access@display`

```
10663 \newcommand*{\@gls@access@display}[2]{%
10664   \protected@edef\@glo@access{#2}%
10665   \ifx\@glo@access\@gls@noaccess
10666     #1%
10667   \else
10668     \xglsaccsupp{\@glo@access}{#1}%
10669   \fi
10670 }
```

`meaccessdisplay` Displays the first argument with the accessibility text for the entry with the label given by the second argument (if set).

```
10671 \DeclareRobustCommand*{\glsnameaccessdisplay}[2]{%
10672   \@gls@access@display{#1}{\glsentryaccess{#2}}%
10673 }
```

`xtaccessdisplay` As above but for the textaccess replacement text.

```
10674 \DeclareRobustCommand*{\glsstextaccessdisplay}[2]{%
10675   \@gls@access@display{#1}{\glsentrytextaccess{#2}}%
10676 }
```

alaccessdisplay As above but for the pluralaccess replacement text.

```
10677 \DeclareRobustCommand*\glspluralaccessdisplay}[2]{%
10678 \@gls@access@display{#1}{\glsentrypluralaccess{#2}}%
10679 }
```

staccessdisplay As above but for the firstaccess replacement text.

```
10680 \DeclareRobustCommand*\glsfirstaccessdisplay}[2]{%
10681 \@gls@access@display{#1}{\glsentryfirstaccess{#2}}%
10682 }
```

alaccessdisplay As above but for the firstpluralaccess replacement text.

```
10683 \DeclareRobustCommand*\glsfirstpluralaccessdisplay}[2]{%
10684 \@gls@access@display{#1}{\glsentryfirstpluralaccess{#2}}%
10685 }
```

olaccessdisplay As above but for the symbolaccess replacement text.

```
10686 \DeclareRobustCommand*\glsymbolaccessdisplay}[2]{%
10687 \@gls@access@display{#1}{\glsentrysymbolaccess{#2}}%
10688 }
```

alaccessdisplay As above but for the symbolpluralaccess replacement text.

```
10689 \DeclareRobustCommand*\glsymbolpluralaccessdisplay}[2]{%
10690 \@gls@access@display{#1}{\glsentrysymbolpluralaccess{#2}}%
10691 }
```

onaccessdisplay As above but for the descriptionaccess replacement text.

```
10692 \DeclareRobustCommand*\glsdescriptionaccessdisplay}[2]{%
10693 \@gls@access@display{#1}{\glsentrydescaccess{#2}}%
10694 }
```

alaccessdisplay As above but for the descriptionpluralaccess replacement text.

```
10695 \DeclareRobustCommand*\glsdescriptionpluralaccessdisplay}[2]{%
10696 \@gls@access@display{#1}{\glsentrydescpluralaccess{#2}}%
10697 }
```

rtaccessdisplay As above but for the shortaccess replacement text.

```
10698 \DeclareRobustCommand*\glsshortaccessdisplay}[2]{%
10699 \@gls@access@display{#1}{\glsentryshortaccess{#2}}%
10700 }
```

alaccessdisplay As above but for the shortpluralaccess replacement text.

```
10701 \DeclareRobustCommand*\glsshortpluralaccessdisplay}[2]{%
10702 \@gls@access@display{#1}{\glsentryshortpluralaccess{#2}}%
10703 }
```

ngaccessdisplay As above but for the longaccess replacement text.

```
10704 \DeclareRobustCommand*\glslongaccessdisplay}[2]{%
10705 \@gls@access@display{#1}{\glsentrylongaccess{#2}}%
10706 }
```

alaccessdisplay As above but for the longpluralaccess replacement text.

```
10707 \DeclareRobustCommand*\glslongpluralaccessdisplay}[2]{%
10708 \@gls@access@display{#1}{\glsentrylongpluralaccess{#2}}%
10709 }
```

lsaccessdisplay Gets the replacement text corresponding to the named key given by the first argument and calls the appropriate command defined above.

```
10710 \DeclareRobustCommand*\glsaccessdisplay}[3]{%
10711 \ifundefined{gls#1accessdisplay}%
10712 {%
10713 \PackageError{glossaries-accsupp}{No accessibility support
10714 for key ‘#1’}{%
10715 }%
10716 {%
10717 \csname gls#1accessdisplay\endcsname{#2}{#3}%
10718 }%
10719 }
```

default@entryfmt Redefine the default entry format to use accessibility information

```
10720 \renewcommand*\@gls@default@entryfmt}[2]{%
10721 \ifdefempty\glscustomtext
10722 {%
10723 \glsifplural
10724 {%
```

Plural form

```
10725 \glscapscase
10726 {%
```

Don't adjust case

```
10727 \ifglsused\glslabel
10728 {%
```

Subsequent use

```
10729 #2{\glspluralaccessdisplay
10730 {\glsentryplural{\glslabel}}{\glslabel}}%
10731 {\glsdescriptionpluralaccessdisplay
10732 {\glsentrydescplural{\glslabel}}{\glslabel}}%
10733 {\glsymbolpluralaccessdisplay
10734 {\glsentrysymbolplural{\glslabel}}{\glslabel}}
10735 {\glsinsert}}%
10736 }%
10737 {%
```

First use

```
10738 #1{\glsfirstpluralaccessdisplay
10739 {\glsentryfirstplural{\glslabel}}{\glslabel}}%
10740 {\glsdescriptionpluralaccessdisplay
10741 {\glsentrydescplural{\glslabel}}{\glslabel}}%
10742 {\glsymbolpluralaccessdisplay
```



```

10743         {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10744     {\glsinsert}%
10745     }%
10746     }%
10747     {%

```

Make first letter upper case

```

10748     \ifglsused\glslabel
10749     {%

```

Subsequent use.

```

10750     #2{\glspluralaccessdisplay
10751         {\Glsentryplural{\glslabel}}{\glslabel}}%
10752     {\glsdescriptionpluralaccessdisplay
10753         {\glsentrydescplural{\glslabel}}{\glslabel}}%
10754     {\glsymbolpluralaccessdisplay
10755         {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10756     {\glsinsert}%
10757     }%
10758     {%

```

First use

```

10759     #1{\glsfirstpluralaccessdisplay
10760         {\Glsentryfirstplural{\glslabel}}{\glslabel}}%
10761     {\glsdescriptionpluralaccessdisplay
10762         {\glsentrydescplural{\glslabel}}{\glslabel}}%
10763     {\glsymbolpluralaccessdisplay
10764         {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10765     {\glsinsert}%
10766     }%
10767     }%
10768     {%

```

Make all upper case

```

10769     \ifglsused\glslabel
10770     {%

```

Subsequent use

```

10771     \MakeUppercase{%
10772     #2{\glspluralaccessdisplay
10773         {\glsentryplural{\glslabel}}{\glslabel}}%
10774     {\glsdescriptionpluralaccessdisplay
10775         {\glsentrydescplural{\glslabel}}{\glslabel}}%
10776     {\glsymbolpluralaccessdisplay
10777         {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10778     {\glsinsert}}%
10779     }%
10780     {%

```

First use

```

10781     \MakeUppercase{%
10782     #1{\glsfirstpluralaccessdisplay

```

```

10783         {\glsentryfirstplural{\glslabel}}{\glslabel}}%
10784     {\glsdescriptionpluralaccessdisplay
10785         {\glsentrydescplural{\glslabel}}{\glslabel}}%
10786     {\glsymbolpluralaccessdisplay
10787         {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10788     {\glsinsert}}%
10789     }%
10790 }%
10791 }%
10792 {%

```

Singular form

```

10793     \glscapscase
10794     {%

```

Don't adjust case

```

10795     \ifglsused\glslabel
10796     {%

```

Subsequent use

```

10797     #2{\glstextaccessdisplay
10798         {\glsentrytext{\glslabel}}{\glslabel}}%
10799     {\glsdescriptionaccessdisplay
10800         {\glsentrydesc{\glslabel}}{\glslabel}}%
10801     {\glsymbolaccessdisplay
10802         {\glsentrysymbol{\glslabel}}{\glslabel}}%
10803     {\glsinsert}}%
10804     }%
10805     {%

```

First use

```

10806     #1{\glsfirstaccessdisplay
10807         {\glsentryfirst{\glslabel}}{\glslabel}}%
10808     {\glsdescriptionaccessdisplay
10809         {\glsentrydesc{\glslabel}}{\glslabel}}%
10810     {\glsymbolaccessdisplay
10811         {\glsentrysymbol{\glslabel}}{\glslabel}}%
10812     {\glsinsert}}%
10813     }%
10814     }%
10815     {%

```

Make first letter upper case

```

10816     \ifglsused\glslabel
10817     {%

```

Subsequent use

```

10818     #2{\glstextaccessdisplay
10819         {\Glsentrytext{\glslabel}}{\glslabel}}%
10820     {\glsdescriptionaccessdisplay
10821         {\Glsentrydesc{\glslabel}}{\glslabel}}%
10822     {\glssymbolaccessdisplay

```

```

10823         {\glsentrysymbol{\glslabel}}{\glslabel}}%
10824     {\glsinsert}}%
10825     }%
10826     {%

```

First use

```

10827     #1{\glsfirstaccessdisplay
10828         {\Glsentryfirst{\glslabel}}{\glslabel}}%
10829     {\glsdescriptionaccessdisplay
10830         {\glsentrydesc{\glslabel}}{\glslabel}}%
10831     {\glsymbolaccessdisplay
10832         {\glsentrysymbol{\glslabel}}{\glslabel}}%
10833     {\glsinsert}}%
10834     }%
10835     }%
10836     {%

```

Make all upper case

```

10837     \ifglsused\glslabel
10838     {%

```

Subsequent use

```

10839     \MakeUppercase{%
10840     #2{\glsfirstaccessdisplay
10841         {\glsentrytext{\glslabel}}{\glslabel}}%
10842     {\glsdescriptionaccessdisplay
10843         {\glsentrydesc{\glslabel}}{\glslabel}}%
10844     {\glsymbolaccessdisplay
10845         {\glsentrysymbol{\glslabel}}{\glslabel}}%
10846     {\glsinsert}}%
10847     }%
10848     {%

```

First use

```

10849     \MakeUppercase{%
10850     #1{\glsfirstaccessdisplay
10851         {\glsentryfirst{\glslabel}}{\glslabel}}%
10852     {\glsdescriptionaccessdisplay
10853         {\glsentrydesc{\glslabel}}{\glslabel}}%
10854     {\glsymbolaccessdisplay
10855         {\glsentrysymbol{\glslabel}}{\glslabel}}%
10856     {\glsinsert}}%
10857     }%
10858     }%
10859     }%
10860     }%
10861     {%

```

Custom text provided in \glsdisp

```

10862     \ifglsused{\glslabel}%
10863     {%

```

Subsequent use

```
10864     #2{\glscustomtext}%
10865     {\glsdescriptionaccessdisplay
10866       {\glsentrydesc{\glslabel}}{\glslabel}}%
10867     {\glsymbolaccessdisplay
10868       {\glsentrysymbol{\glslabel}}{\glslabel}}%
10869     {\glsinsert}%
10870   }%
10871   {%
```

First use

```
10872     #1{\glscustomtext}%
10873     {\glsdescriptionaccessdisplay
10874       {\glsentrydesc{\glslabel}}{\glslabel}}%
10875     {\glsymbolaccessdisplay
10876       {\glsentrysymbol{\glslabel}}{\glslabel}}%
10877     {\glsinsert}%
10878   }%
10879 }%
10880 }
```

`\glsentryfmt` Redefine to use accessibility information.

```
10881 \renewcommand*{\glsentryfmt}{%
10882   \ifdefempty\glscustomtext
10883   {%
10884     \glsifplural
10885     {%
```

Plural form

```
10886     \glscapscase
10887     {%
```

Don't adjust case

```
10888     \ifglsused\glslabel
10889     {%
```

Subsequent use

```
10890     \glspluralaccessdisplay
10891     {\glsentryplural{\glslabel}}{\glslabel}%
10892     \glsinsert
10893   }%
10894   {%
```

First use

```
10895     \glsfirstpluralaccessdisplay
10896     {\glsentryfirstplural{\glslabel}}{\glslabel}%
10897     \glsinsert
10898   }%
10899 }%
10900 }
```

Make first letter upper case

10901 `\ifglsused\glslabel`
10902 `{%`

Subsequent use.

10903 `\glspluralaccessdisplay`
10904 `{\Glsentryplural{\glslabel}}{\glslabel}%`
10905 `\glsinsert`
10906 `}%`
10907 `{%`

First use

10908 `\glsfirstpluralaccessdisplay`
10909 `{\Glsentryfirstplural{\glslabel}}{\glslabel}%`
10910 `\glsinsert`
10911 `}%`
10912 `}%`
10913 `{%`

Make all upper case

10914 `\ifglsused\glslabel`
10915 `{%`

Subsequent use

10916 `\glspluralaccessdisplay`
10917 `{\mfirstucMakeUppercase{\glsentryplural{\glslabel}}}%`
10918 `{\glslabel}%`
10919 `\mfirstucMakeUppercase{\glsinsert}%`
10920 `}%`
10921 `{%`

First use

10922 `\glsfirstpluralacessdisplay`
10923 `{\mfirstucMakeUppercase{\glsentryfirstplural{\glslabel}}}%`
10924 `{\glslabel}%`
10925 `\mfirstucMakeUppercase{\glsinsert}%`
10926 `}%`
10927 `}%`
10928 `}%`
10929 `{%`

Singular form

10930 `\glscaps case`
10931 `{%`

Don't adjust case

10932 `\ifglsused\glslabel`
10933 `{%`

Subsequent use

10934 `\glstextaccessdisplay{\glsentrytext{\glslabel}}{\glslabel}%`
10935 `\glsinsert`

10936 }%
10937 {%

First use

10938 \glsfirstaccessdisplay{\glsentryfirst{\glslabel}}{\glslabel}%
10939 \glsinsert
10940 }%
10941 }%
10942 {%

Make first letter upper case

10943 \ifglsused\glslabel
10944 {%

Subsequent use

10945 \glstextaccessdisplay{\Glsentrytext{\glslabel}}{\glslabel}%
10946 \glsinsert
10947 }%
10948 {%

First use

10949 \glsfirstaccessdisplay{\Glsentryfirst{\glslabel}}{\glslabel}%
10950 \glsinsert
10951 }%
10952 }%
10953 {%

Make all upper case

10954 \ifglsused\glslabel
10955 {%

Subsequent use

10956 \glstextaccessdisplay
10957 {\mfirstucMakeUppercase{\glsentrytext{\glslabel}}}{\glslabel}%
10958 \mfirstucMakeUppercase{\glsinsert}%
10959 }%
10960 {%

First use

10961 \glsfirstaccessdisplay
10962 {\mfirstucMakeUppercase{\glsentryfirst{\glslabel}}}{\glslabel}%
10963 \mfirstucMakeUppercase{\glsinsert}%
10964 }%
10965 }%
10966 }%
10967 }%
10968 {%

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.

10969 \glscustomtext\glsinsert
10970 }%
10971 }

`\glsgenacfmt` Redefine to include accessibility information.

```
10972 \renewcommand*{\glsgenacfmt}{%
10973   \ifdefempty\glscustomtext
10974   {%
10975     \ifglused\glslabel
10976     {%
```

Subsequent use:

```
10977     \glsifplural
10978     {%
```

Subsequent plural form:

```
10979     \glscapscase
10980     {%
```

Subsequent plural form, don't adjust case:

```
10981     \acronymfont
10982     {\glsshortpluralaccessdisplay
10983       {\glentryshortpl{\glslabel}}{\glslabel}}%
10984     \glsinsert
10985     }%
10986     {%
```

Subsequent plural form, make first letter upper case:

```
10987     \acronymfont
10988     {\glsshortpluralaccessdisplay
10989       {\Glsentryshortpl{\glslabel}}{\glslabel}}%
10990     \glsinsert
10991     }%
10992     {%
```

Subsequent plural form, all caps:

```
10993     \mfirstucMakeUppercase
10994     {\acronymfont
10995       {\glsshortpluralaccessdisplay
10996         {\glentryshortpl{\glslabel}}{\glslabel}}%
10997       \glsinsert}%
10998     }%
10999     }%
11000     {%
```

Subsequent singular form

```
11001     \glscapscase
11002     {%
```

Subsequent singular form, don't adjust case:

```
11003     \acronymfont
11004     {\glsshortaccessdisplay{\glentryshort{\glslabel}}{\glslabel}}%
11005     \glsinsert
11006     }%
11007     {%
```

Subsequent singular form, make first letter upper case:

```
11008      \acronymfont
11009      {\glsshortaccessdisplay{\Glsentryshort{\glslabel}}{\glslabel}}%
11010      \glsinsert
11011      }%
11012      {%
```

Subsequent singular form, all caps:

```
11013      \mfirstucMakeUppercase
11014      {\acronymfont{%
11015      \glsshortaccessdisplay{\glsentryshort{\glslabel}}{\glslabel}}%
11016      \glsinsert}%
11017      }%
11018      }%
11019      }%
11020      {%
```

First use:

```
11021      \glsifplural
11022      {%
```

First use plural form:

```
11023      \glscapscase
11024      {%
```

First use plural form, don't adjust case:

```
11025      \genplacrfullformat{\glslabel}{\glsinsert}%
11026      }%
11027      {%
```

First use plural form, make first letter upper case:

```
11028      \Genplacrfullformat{\glslabel}{\glsinsert}%
11029      }%
11030      {%
```

First use plural form, all caps:

```
11031      \mfirstucMakeUppercase
11032      {\genplacrfullformat{\glslabel}{\glsinsert}}%
11033      }%
11034      }%
11035      {%
```

First use singular form

```
11036      \glscapscase
11037      {%
```

First use singular form, don't adjust case:

```
11038      \genacrfullformat{\glslabel}{\glsinsert}%
11039      }%
11040      {%
```


First use singular form, make first letter upper case:

```
11041      \Genacrfullformat{\glslabel}{\glsinsert}%
11042      }%
11043      {%
```

First use singular form, all caps:

```
11044      \mfirstucMakeUppercase
11045      {\genacrfullformat{\glslabel}{\glsinsert}}%
11046      }%
11047      }%
11048      }%
11049      }%
11050      {%
```

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.

```
11051      \glscustomtext
11052      }%
11053 }
```

enacrfullformat Redefine to include accessibility information.

```
11054 \renewcommand*{\genacrfullformat}[2]{%
11055   \glslongaccessdisplay{\glsentrylong{#1}}{#1}#2\space
11056   (\glsshortaccessdisplay{\protect\firstacronymfont{\glsentryshort{#1}}}{#1})%
11057 }
```

enacrfullformat Redefine to include accessibility information.

```
11058 \renewcommand*{\Genacrfullformat}[2]{%
11059   \glslongaccessdisplay{\Glsentrylong{#1}}{#1}#2\space
11060   (\glsshortaccessdisplay{\protect\firstacronymfont{\Glsentryshort{#1}}}{#1})%
11061 }
```

placrfullformat Redefine to include accessibility information.

```
11062 \renewcommand*{\genplacrfullformat}[2]{%
11063   \glslongpluralaccessdisplay{\glsentrylongpl{#1}}{#1}#2\space
11064   (\glsshortpluralaccessdisplay
11065     {\protect\firstacronymfont{\glsentryshortpl{#1}}}{#1})%
11066 }
```

placrfullformat Redefine to include accessibility information.

```
11067 \renewcommand*{\Genplacrfullformat}[2]{%
11068   \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}#2\space
11069   (\glsshortpluralaccessdisplay
11070     {\protect\firstacronymfont{\glsentryshortpl{#1}}}{#1})%
11071 }
```

\@acrshort

```
11072 \def\@acrshort#1#2[#3]{%
11073   \glsdoifexists{#2}%
```

```

11074 {%
11075   \let\do@gls@link@checkfirsthyper\relax

11076   \let\glsifplural\@secondoftwo
11077   \let\glsapscase\@firstofthree
11078   \let\glsinsert\@empty
11079   \def\glscustomtext{%
11080     \acronymfont{\glsshortaccessdisplay{\glsentryshort{#2}}{#2}}#3%
11081   }%

   Call \@gls@link
11082   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
11083 }%

11084 \glspostlinkhook
11085 }

```

\@Acrshort

```

11086 \def\@Acrshort#1#2[#3]{%
11087   \glsdoifexists{#2}%
11088   {%
11089     \let\do@gls@link@checkfirsthyper\relax

11090     \let\glsifplural\@secondoftwo
11091     \let\glsapscase\@secondofthree
11092     \let\glsinsert\@empty
11093     \def\glscustomtext{%
11094       \acronymfont{\glsshortaccessdisplay{\Glsentryshort{#2}}{#2}}#3%
11095     }%

     Call \@gls@link
11096     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
11097   }%

11098   \glspostlinkhook
11099 }

```

\@ACRshort

```

11100 \def\@ACRshort#1#2[#3]{%
11101   \glsdoifexists{#2}%
11102   {%
11103     \let\do@gls@link@checkfirsthyper\relax

11104     \let\glsifplural\@secondoftwo
11105     \let\glsapscase\@thirdofthree
11106     \let\glsinsert\@empty
11107     \def\glscustomtext{%
11108       \acronymfont{\glsshortaccessdisplay
11109         {\MakeUppercase{\glsentryshort{#2}}}{#2}}#3%
11110     }%

```

```

    Call \@gls@link
11111 \gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
11112 }%

11113 \glspostlinkhook
11114 }

```

\@acrlong

```

11115 \def\@acrlong#1#2[#3]{%
11116 \glsdoifexists{#2}%
11117 {%
11118 \let\do@gls@link@checkfirsthyper\relax

11119 \let\glsifplural\@secondoftwo
11120 \let\gls caps case\@firstofthree
11121 \let\glsinsert\@empty
11122 \def\gls custom text{%
11123 \acronymfont{\gls long access display{\glsentrylong{#2}}{#2}}#3%
11124 }%

```

Call \@gls@link

```

11125 \gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
11126 }%

11127 \glspostlinkhook
11128 }

```

\@Acrlong

```

11129 \def\@Acrlong#1#2[#3]{%
11130 \glsdoifexists{#2}%
11131 {%
11132 \let\do@gls@link@checkfirsthyper\relax

11133 \let\glsifplural\@secondoftwo
11134 \let\gls caps case\@firstofthree
11135 \let\glsinsert\@empty
11136 \def\gls custom text{%
11137 \acronymfont{\Gls long access display{\Glsentrylong{#2}}{#2}}#3%
11138 }%

```

Call \@gls@link

```

11139 \gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
11140 }%

11141 \glspostlinkhook
11142 }

```

\@ACRlong

```

11143 \def\@ACRlong#1#2[#3]{%
11144 \glsdoifexists{#2}%
11145 {%
11146 \let\do@gls@link@checkfirsthyper\relax

```

```

11147 \let\glsifplural\@secondoftwo
11148 \let\glsifscaps\@firstofthree
11149 \let\glsinsert\@empty
11150 \def\glscustomtext{%
11151   \acronymfont{\glslongaccessdisplay{%
11152     \MakeUppercase{\glsentrylong{#2}}{#2}#3}%
11153   }%

Call \@gls@link
11154 \@gls@link[#1]{#2}{\csname gls@\gls@glstype @entryfmt\endcsname}%
11155 }%

11156 \glspostlinkhook
11157 }

```

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.

```

11158 \renewcommand*\glossentryname}[1]{%
11159   \glsdoifexists{#1}%
11160   {%
11161     \glsnamefont{\glsnameaccessdisplay{\glsentryname{#1}}{#1}}%
11162   }%
11163 }

11164 \renewcommand*\glossentrydesc}[1]{%
11165   \glsdoifexists{#1}%
11166   {%
11167     \glsnamefont{\glsnameaccessdisplay{\Glsentryname{#1}}{#1}}%
11168   }%
11169 }

11170 \renewcommand*\glossentrydesc}[1]{%
11171   \glsdoifexists{#1}%
11172   {%
11173     \glsdescriptionaccessdisplay{\glsentrydesc{#1}}{#1}%
11174   }%
11175 }

11176 \renewcommand*\Glossentrydesc}[1]{%
11177   \glsdoifexists{#1}%
11178   {%
11179     \glsdescriptionaccessdisplay{\Glsentrydesc{#1}}{#1}%
11180   }%
11181 }

```

```

11182 \renewcommand*{\glossentrysymbol}[1]{%
11183   \glsdoifexists{#1}%
11184   {%
11185     \glssymbolaccessdisplay{\glsentrysymbol{#1}}{#1}%
11186   }%
11187 }

11188 \renewcommand*{\Glossentrysymbol}[1]{%
11189   \glsdoifexists{#1}%
11190   {%
11191     \glssymbolaccessdisplay{\Glsentrysymbol{#1}}{#1}%
11192   }%
11193 }

```

ssaryentryfield

```

11194 \newcommand*{\accsuppglossaryentryfield}[5]{%
11195   \glossaryentryfield{#1}%
11196   {\glsnameaccessdisplay{#2}{#1}}%
11197   {\glsdescriptionaccessdisplay{#3}{#1}}%
11198   {\glssymbolaccessdisplay{#4}{#1}}{#5}%
11199 }

```

rysubentryfield

```

11200 \newcommand*{\accsuppglossarysubentryfield}[6]{%
11201   \glossarysubentryfield{#1}{#2}%
11202   {\glsnameaccessdisplay{#3}{#2}}%
11203   {\glsdescriptionaccessdisplay{#4}{#2}}%
11204   {\glssymbolaccessdisplay{#5}{#2}}{#6}%
11205 }

```

5.4 Acronyms

Redefine acronym styles provided by glossaries:

long-short *<long>* (*<short>*) acronym style.

```

11206 \renewacronymstyle{long-short}%
11207 {%

```

Check for long form in case this is a mixed glossary.

```

11208   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
11209 }%
11210 {%
11211   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
11212   \renewcommand*{\genacrfullformat}[2]{%
11213     \glslongaccessdisplay{\glsentrylong{##1}}{##1}##2\space
11214     (\glsshortaccessdisplay
11215       {\protect\firstacronymfont{\glsentryshort{##1}}}{##1})%
11216   }%
11217   \renewcommand*{\Genacrfullformat}[2]{%

```

```

11218 \glslongaccessdisplay{\Glsentrylong{##1}}{##1}##2\space
11219 (\glsshortaccessdisplay
11220   {\protect\firstacronymfont{\glsentryshort{##1}}}{##1})%
11221 }%
11222 \renewcommand*{\genplacrformat}[2]{%
11223   \glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}##2\space
11224   (\glsshortpluralaccessdisplay
11225     {\protect\firstacronymfont{\glsentryshortpl{##1}}}{##1})%
11226   }%
11227 \renewcommand*{\Genplacrformat}[2]{%
11228   \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}##2\space
11229   (\glsshortpluralaccessdisplay
11230     {\protect\firstacronymfont{\glsentryshortpl{##1}}}{##1})%
11231   }%
11232 \renewcommand*{\acronymentry}[1]{%
11233   \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}
11234 \renewcommand*{\acronymsort}[2]{##1}%
11235 \renewcommand*{\acronymfont}[1]{##1}%
11236 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
11237 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11238 }

```

short-long (*short*) (*long*) acronym style.

```

11239 \renewacronymstyle{short-long}%
11240 {%

```

Check for long form in case this is a mixed glossary.

```

11241 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
11242 }%
11243 {%
11244 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
11245 \renewcommand*{\genacrformat}[2]{%
11246   \glsshortaccessdisplay
11247     {\protect\firstacronymfont{\glsentryshort{##1}}}{##1}##2\space
11248   (\glslongaccessdisplay{\glsentrylong{##1}}{##1})%
11249   }%
11250 \renewcommand*{\Genacrformat}[2]{%
11251   \glsshortaccessdisplay
11252     {\protect\firstacronymfont{\Glsentryshort{##1}}}{##1}##2\space
11253   (\glslongaccessdisplay{\Glsentrylong{##1}}{##1})%
11254   }%
11255 \renewcommand*{\genplacrformat}[2]{%
11256   \glsshortpluralaccessdisplay
11257     {\protect\firstacronymfont{\glsentryshortpl{##1}}}{##1}##2\space
11258   (\glslongpluralaccessdisplay
11259     {\glsentrylongpl{##1}}{##1})%
11260   }%
11261 \renewcommand*{\Genplacrformat}[2]{%
11262   \glsshortpluralaccessdisplay
11263     {\protect\firstacronymfont{\Glsentryshortpl{##1}}}{##1}##2\space

```

```

11264 (\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1})%
11265 }%
11266 \renewcommand*{\acronymentry}[1]{%
11267   \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}%
11268 \renewcommand*{\acronymsort}[2]{##1}%
11269 \renewcommand*{\acronymfont}[1]{##1}%
11270 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
11271 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11272 }

```

long-short-desc *long* (*short*) acronym style that has an accompanying description (which the user needs to supply).

```

11273 \renewacronymstyle{long-short-desc}%
11274 {%
11275   \GlsUseAcrEntryDispStyle{long-short}%
11276 }%
11277 {%
11278   \GlsUseAcrStyleDefs{long-short}%
11279   \renewcommand*{\GenericAcronymFields}{}%
11280   \renewcommand*{\acronymsort}[2]{##2}%
11281   \renewcommand*{\acronymentry}[1]{%
11282     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11283     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11284 }

```

g-sc-short-desc *long* (\textsc{short}) acronym style that has an accompanying description (which the user needs to supply).

```

11285 \renewacronymstyle{long-sc-short-desc}%
11286 {%
11287   \GlsUseAcrEntryDispStyle{long-sc-short}%
11288 }%
11289 {%
11290   \GlsUseAcrStyleDefs{long-sc-short}%
11291   \renewcommand*{\GenericAcronymFields}{}%
11292   \renewcommand*{\acronymsort}[2]{##2}%
11293   \renewcommand*{\acronymentry}[1]{%
11294     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11295     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11296 }

```

g-sm-short-desc *long* (\textsmaller{short}) acronym style that has an accompanying description (which the user needs to supply).

```

11297 \renewacronymstyle{long-sm-short-desc}%
11298 {%
11299   \GlsUseAcrEntryDispStyle{long-sm-short}%
11300 }%
11301 {%
11302   \GlsUseAcrStyleDefs{long-sm-short}%
11303   \renewcommand*{\GenericAcronymFields}{}%

```

```

11304 \renewcommand*\acronymsort}[2]{##2}%
11305 \renewcommand*\acronymentry}[1]{%
11306   \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11307   (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11308 }

```

short-long-desc *(short)* (*{<long>}*) acronym style that has an accompanying description (which the user needs to supply).

```

11309 \renewacronymstyle{short-long-desc}%
11310 {%
11311   \GlsUseAcrEntryDispStyle{short-long}%
11312 }%
11313 {%
11314   \GlsUseAcrStyleDefs{short-long}%
11315   \renewcommand*\GenericAcronymFields{}%
11316   \renewcommand*\acronymsort}[2]{##2}%
11317   \renewcommand*\acronymentry}[1]{%
11318     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11319     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11320 }

```

short-long-desc *(long)* (*\textsc{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

11321 \renewacronymstyle{sc-short-long-desc}%
11322 {%
11323   \GlsUseAcrEntryDispStyle{sc-short-long}%
11324 }%
11325 {%
11326   \GlsUseAcrStyleDefs{sc-short-long}%
11327   \renewcommand*\GenericAcronymFields{}%
11328   \renewcommand*\acronymsort}[2]{##2}%
11329   \renewcommand*\acronymentry}[1]{%
11330     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11331     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11332 }

```

short-long-desc *(long)* (*\textsmaller{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

11333 \renewacronymstyle{sm-short-long-desc}%
11334 {%
11335   \GlsUseAcrEntryDispStyle{sm-short-long}%
11336 }%
11337 {%
11338   \GlsUseAcrStyleDefs{sm-short-long}%
11339   \renewcommand*\GenericAcronymFields{}%
11340   \renewcommand*\acronymsort}[2]{##2}%
11341   \renewcommand*\acronymentry}[1]{%
11342     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11343     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%

```


11344 }

dua *<long>* only acronym style.

11345 \renewacronymstyle{dua}%
11346 {%

Check for long form in case this is a mixed glossary.

11347 \ifdefempty\glscustomtext
11348 {%
11349 \ifglshaslong{\glslabel}%
11350 {%
11351 \glsifplural
11352 {%

Plural form:

11353 \glscapscase
11354 {%

Plural form, don't adjust case:

11355 \glslongpluralaccessdisplay{\glsentrylongpl{\glslabel}}{\glslabel}%
11356 \glsinsert
11357 }%
11358 {%

Plural form, make first letter upper case:

11359 \glslongpluralaccessdisplay{\Glsentrylongpl{\glslabel}}{\glslabel}%
11360 \glsinsert
11361 }%
11362 {%

Plural form, all caps:

11363 \glslongpluralaccessdisplay
11364 {\mfirstucMakeUppercase{\glsentrylongpl{\glslabel}}}{\glslabel}%
11365 \mfirstucMakeUppercase{\glsinsert}%
11366 }%
11367 }%
11368 {%

Singular form

11369 \glscapscase
11370 {%

Singular form, don't adjust case:

11371 \glslongaccessdisplay{\glsentrylong{\glslabel}}{\glslabel}\glsinsert
11372 }%
11373 {%

Subsequent singular form, make first letter upper case:

11374 \glslongaccessdisplay{\Glsentrylong{\glslabel}}{\glslabel}\glsinsert
11375 }%
11376 {%

Subsequent singular form, all caps:

```

11377     \glslongaccessdisplay
11378     {\mfirstucMakeUppercase
11379       {\glsentrylong{\glslabel}\glsinsert}}{\glslabel}%
11380     \mfirstucMakeUppercase{\glsinsert}%
11381     }%
11382   }%
11383 }%
11384 {%
```

Not an acronym:

```

11385     \glsgenentryfmt
11386   }%
11387 }%
11388 {\glscustomtext\glsinsert}%
11389}%
11390{%
```

```

11391 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
11392 \renewcommand*\acrfullfmt}[3]{%
11393   \glslink[##1]{##2}{%
11394     \glslongaccessdisplay{\glsentrylong{##2}}{##2}##3\space
11395     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2})}}%
11396 \renewcommand*\Acrfullfmt}[3]{%
11397   \glslink[##1]{##2}{%
11398     \glslongaccessdisplay{\Glsentrylong{##2}}{##2}##3\space
11399     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2})}}%
11400 \renewcommand*\ACRfullfmt}[3]{%
11401   \glslink[##1]{##2}{%
11402     \glslongaccessdisplay
11403     {\mfirstucMakeUppercase{\glsentrylong{##2}}{##2}##3\space
11404     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2})}}}%
11405 \renewcommand*\acrfullplfmt}[3]{%
11406   \glslink[##1]{##2}{%
11407     \glslongpluralaccessdisplay
11408     {\glsentrylongpl{##2}}{##2}##3\space
11409     (\glsshortpluralaccessdisplay
11410     {\acronymfont{\glsentryshortpl{##2}}}{##2})}}%
11411 \renewcommand*\Acrfullplfmt}[3]{%
11412   \glslink[##1]{##2}{%
11413     \glslongpluralaccessdisplay
11414     {\Glsentrylongpl{##2}}{##2}##3\space
11415     (\glsshortpluralaccessdisplay
11416     {\acronymfont{\glsentryshortpl{##2}}}{##2})}}%
11417 \renewcommand*\ACRfullplfmt}[3]{%
11418   \glslink[##1]{##2}{%
11419     \glslongpluralaccessdisplay
11420     {\mfirstucMakeUppercase{\glsentrylongpl{##2}}{##2}##3\space
11421     (\glsshortpluralaccessdisplay
11422     {\acronymfont{\glsentryshortpl{##2}}}{##2})}}}%
11423 \renewcommand*\glsentryfull}[1]{%
```

```

11424 \glslongaccessdisplay{\glsentrylong{##1}}\space
11425 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})%
11426 }%
11427 \renewcommand*{\Glsentryfull}[1]{%
11428 \glslongaccessdisplay{\Glsentrylong{##1}}{##1}\space
11429 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})%
11430 }%
11431 \renewcommand*{\glsentryfullpl}[1]{%
11432 \glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}\space
11433 (\glsshortpluralaccessdisplay{\acronymfont{\glsentryshortpl{##1}}}{##1})%
11434 }%
11435 \renewcommand*{\Glsentryfullpl}[1]{%
11436 \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}\space
11437 (\glsshortpluralaccessdisplay{\acronymfont{\glsentryshortpl{##1}}}{##1})%
11438 }%
11439 \renewcommand*{\acronymentry}[1]{%
11440 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}%
11441 \renewcommand*{\acronymsort}[2]{##1}%
11442 \renewcommand*{\acronymfont}[1]{##1}%
11443 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11444 }

```

dua-desc *<long>* only acronym style with user-supplied description.

```

11445 \renewacronymstyle{dua-desc}%
11446 {%
11447 \GlsUseAcrEntryDispStyle{dua}%
11448 }%
11449 {%
11450 \GlsUseAcrStyleDefs{dua}%
11451 \renewcommand*{\GenericAcronymFields}{}%
11452 \renewcommand*{\acronymentry}[1]{%
11453 \glslongaccessdisplay{\acronymfont{\glsentrylong{##1}}}{##1}}%
11454 \renewcommand*{\acronymsort}[2]{##2}%
11455 }%

```

footnote *<short>*\footnote{*<long>*} acronym style.

```

11456 \renewacronymstyle{footnote}%
11457 {%
11458 \ifglshaslong{\glslabel}{\glsngenacfmt}{\glsngenentryfmt}%
11459 }%
11460 {%
11461 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%

```

Need to ensure hyperlinks are switched off on first use:

```

11462 \glshyperfirstfalse
11463 \renewcommand*{\genacrfullformat}[2]{%
11464 \glsshortaccessdisplay
11465 {\protect\firstacronymfont{\glsentryshort{##1}}}{##1}##2%

```

```

11466 \protect\footnote{\glslongaccessdisplay{\glsentrylong{##1}}{##1}}%
11467 }%
11468 \renewcommand*{\Genacrfullformat}[2]{%
11469 \glsshortaccessdisplay
11470   {\firstacronymfont{\Glsentryshort{##1}}{##1}##2%
11471 \protect\footnote{\glslongaccessdisplay{\glsentrylong{##1}}{##1}}%
11472 }%
11473 \renewcommand*{\genplacrfullformat}[2]{%
11474 \glsshortpluralaccessdisplay
11475   {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}##2%
11476 \protect\footnote{\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}}%
11477 }%
11478 \renewcommand*{\Genplacrfullformat}[2]{%
11479 \glsshortpluralaccessdisplay
11480   {\protect\firstacronymfont{\Glsentryshortpl{##1}}{##1}##2%
11481 \protect\footnote{\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}}%
11482 }%
11483 \renewcommand*{\acronymentry}[1]{%
11484 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}{##1}}%
11485 \renewcommand*{\acronymsort}[2]{##1}%
11486 \renewcommand*{\acronymfont}[1]{##1}%
11487 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%

```

Don't use footnotes for \acrfull:

```

11488 \renewcommand*{\acrfullfmt}[3]{%
11489 \glslink[##1]{##2}{%
11490 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}{##2}##3\space
11491 (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}%
11492 \renewcommand*{\Acrfullfmt}[3]{%
11493 \glslink[##1]{##2}{%
11494 \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##2}}{##2}##3\space
11495 (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}%
11496 \renewcommand*{\ACRfullfmt}[3]{%
11497 \glslink[##1]{##2}{%
11498 \glsshortaccessdisplay
11499   {\mfirstucMakeUppercase
11500   {\acronymfont{\glsentryshort{##2}}{##2}##3\space
11501   (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}}%
11502 \renewcommand*{\acrfullplfmt}[3]{%
11503 \glslink[##1]{##2}{%
11504 \glsshortpluralaccessdisplay
11505   {\acronymfont{\glsentryshortpl{##2}}{##2}##3\space
11506   (\glslongpluralaccessdisplay{\glsentrylongpl{##2}}{##2})}}}%
11507 \renewcommand*{\Acrfullplfmt}[3]{%
11508 \glslink[##1]{##2}{%
11509 \glsshortpluralaccessdisplay
11510   {\acronymfont{\Glsentryshortpl{##2}}{##2}##3\space
11511   (\glslongpluralaccessdisplay{\glsentrylongpl{##2}}{##2})}}}%
11512 \renewcommand*{\ACRfullplfmt}[3]{%
11513 \glslink[##1]{##2}{%

```

```

11514 \glsshortpluralaccessdisplay
11515     {\mfirstucMakeUppercase
11516      {\acronymfont{\glentryshortpl{##2}}{##2}##3\space
11517      (\glslongpluralaccessdisplay{\glentrylongpl{##2}}{##2})}}}%

```

Similarly for \glentryfull etc:

```

11518 \renewcommand*{\glentryfull}[1]{%
11519     \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}{##1}\space
11520     (\glslongaccessdisplay{\glentrylong{##1}}{##1})}%
11521 \renewcommand*{\Glsentryfull}[1]{%
11522     \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##1}}{##1}\space
11523     (\glslongaccessdisplay{\glentrylong{##1}}{##1})}%
11524 \renewcommand*{\glentryfullpl}[1]{%
11525     \glsshortpluralaccessdisplay
11526     {\acronymfont{\glentryshortpl{##1}}{##1}\space
11527     (\glslongpluralaccessdisplay{\glentrylongpl{##1}}{##1})}%
11528 \renewcommand*{\Glsentryfullpl}[1]{%
11529     \glsshortpluralaccessdisplay
11530     {\acronymfont{\Glsentryshortpl{##1}}{##1}\space
11531     (\glslongpluralaccessdisplay{\glentrylongpl{##1}}{##1})}%
11532 }

```

footnote-sc \textsc{<short>}\footnote{<long>} acronym style.

```

11533 \renewacronymstyle{footnote-sc}%
11534 {%
11535     \GlsUseAcrEntryDispStyle{footnote}%
11536 }%
11537 {%
11538     \GlsUseAcrStyleDefs{footnote}%
11539     \renewcommand{\acronymentry}[1]{%
11540         \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}{##1}}
11541         \renewcommand{\acronymfont}[1]{\textsc{##1}}%
11542         \renewcommand*{\acrpluralsuffix}{\glstextup{\glspluralsuffix}}%
11543 }%

```

footnote-sm \textsmaller{<short>}\footnote{<long>} acronym style.

```

11544 \renewacronymstyle{footnote-sm}%
11545 {%
11546     \GlsUseAcrEntryDispStyle{footnote}%
11547 }%
11548 {%
11549     \GlsUseAcrStyleDefs{footnote}%
11550     \renewcommand{\acronymentry}[1]{%
11551         \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}{##1}}
11552         \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
11553         \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11554 }%

```

footnote-desc <short>\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

11555 \renewacronymstyle{footnote-desc}%
11556 {%
11557   \GlsUseAcrEntryDispStyle{footnote}%
11558 }%
11559 {%
11560   \GlsUseAcrStyleDefs{footnote}%
11561   \renewcommand*{\GenericAcronymFields}{}%
11562   \renewcommand*{\acronymsort}[2]{##2}%
11563   \renewcommand*{\acronymentry}[1]{%
11564     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11565     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11566 }

```

ootnote-sc-desc \textsc{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

11567 \renewacronymstyle{footnote-sc-desc}%
11568 {%
11569   \GlsUseAcrEntryDispStyle{footnote-sc}%
11570 }%
11571 {%
11572   \GlsUseAcrStyleDefs{footnote-sc}%
11573   \renewcommand*{\GenericAcronymFields}{}%
11574   \renewcommand*{\acronymsort}[2]{##2}%
11575   \renewcommand*{\acronymentry}[1]{%
11576     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11577     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11578 }

```

ootnote-sm-desc \textsmaller{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

11579 \renewacronymstyle{footnote-sm-desc}%
11580 {%
11581   \GlsUseAcrEntryDispStyle{footnote-sm}%
11582 }%
11583 {%
11584   \GlsUseAcrStyleDefs{footnote-sm}%
11585   \renewcommand*{\GenericAcronymFields}{}%
11586   \renewcommand*{\acronymsort}[2]{##2}%
11587   \renewcommand*{\acronymentry}[1]{%
11588     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11589     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11590 }

```

Use \newacronymhook to modify the key list to set the access text to the long version by default.

```

11591 \renewcommand*{\newacronymhook}{%
11592   \edef\@gls@keylist{shortaccess=\the\gls\longtok,%
11593     \the\glskeylisttok}%
11594   \expandafter\glskeylisttok\expandafter{\@gls@keylist}%

```

11595 }

ltNewAcronymDef Modify default style to use access text:

```
11596 \renewcommand*{\DefaultNewAcronymDef}{%
11597   \edef\@do@newglossaryentry{%
11598     \noexpand\newglossaryentry{\the\glslabeltok}%
11599     {%
11600       type=\acronymtype,%
11601       name={\the\glsshorttok},%
11602       description={\the\glslongtok},%
11603       descriptionaccess=\relax,
11604       text={\the\glsshorttok},%
11605       access={\noexpand\@glo@textaccess},%
11606       sort={\the\glsshorttok},%
11607       short={\the\glsshorttok},%
11608       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11609       shortaccess={\the\glslongtok},%
11610       long={\the\glslongtok},%
11611       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11612       descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11613       first={\noexpand\glslongaccessdisplay
11614         {\the\glslongtok}{\the\glslabeltok}\space
11615         {\noexpand\glsshortaccessdisplay
11616           {\the\glsshorttok}{\the\glslabeltok}}},%
11617       plural={\the\glsshorttok\acrpluralsuffix},%
11618       firstplural={\noexpand\glslongpluralaccessdisplay
11619         {\noexpand\@glo@longpl}{\the\glslabeltok}\space
11620         {\noexpand\glsshortpluralaccessdisplay
11621           {\noexpand\@glo@shortpl}{\the\glslabeltok}}},%
11622       firstaccess=\relax,
11623       firstpluralaccess=\relax,
11624       textaccess={\noexpand\@glo@shortaccess},%
11625       \the\glskeylisttok
11626     }%
11627   }%
11628   \let\@org@gls@assign@firstpl\gls@assign@firstpl
11629   \let\@org@gls@assign@plural\gls@assign@plural
11630   \let\@org@gls@assign@descplural\gls@assign@descplural
11631   \def\gls@assign@firstpl##1##2{%
11632     \@gls@expand@field{##1}{firstpl}{##2}%
11633   }%
11634   \def\gls@assign@plural##1##2{%
11635     \@gls@expand@field{##1}{plural}{##2}%
11636   }%
11637   \def\gls@assign@descplural##1##2{%
11638     \@gls@expand@field{##1}{descplural}{##2}%
11639   }%
11640   \@do@newglossaryentry
11641   \let\gls@assign@firstpl\@org@gls@assign@firstpl
```

```

11642 \let\gls@assign@plural\@org@gls@assign@plural
11643 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11644 }

```

teNewAcronymDef

```

11645 \renewcommand*{\DescriptionFootnoteNewAcronymDef}{%
11646 \edef\@do@newglossaryentry{%
11647 \noexpand\newglossaryentry{\the\glslabeltok}%
11648 {%
11649 type=\acronymtype,%
11650 name={\noexpand\acronymfont{\the\glsshorttok}},%
11651 sort={\the\glsshorttok},%
11652 text={\the\glsshorttok},%
11653 short={\the\glsshorttok},%
11654 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11655 shortaccess={\the\glslongtok},%
11656 long={\the\glslongtok},%
11657 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11658 access={\noexpand\@glo@textaccess},%
11659 plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11660 symbol={\the\glslongtok},%
11661 symbolplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11662 firstpluralaccess=\relax,
11663 textaccess={\noexpand\@glo@shortaccess},%
11664 \the\glskeylisttok
11665 }%
11666 }%
11667 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11668 \let\@org@gls@assign@plural\gls@assign@plural
11669 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
11670 \def\gls@assign@firstpl##1##2{%
11671 \@@gls@expand@field{##1}{firstpl}{##2}%
11672 }%
11673 \def\gls@assign@plural##1##2{%
11674 \@@gls@expand@field{##1}{plural}{##2}%
11675 }%
11676 \def\gls@assign@symbolplural##1##2{%
11677 \@@gls@expand@field{##1}{symbolplural}{##2}%
11678 }%
11679 \@do@newglossaryentry
11680 \let\gls@assign@plural\@org@gls@assign@plural
11681 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11682 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11683 }

```

onNewAcronymDef

```

11684 \renewcommand*{\DescriptionNewAcronymDef}{%
11685 \edef\@do@newglossaryentry{%
11686 \noexpand\newglossaryentry{\the\glslabeltok}%

```



```

11687  {%
11688     type=\acronymtype,%
11689     name={\noexpand
11690         \acronymformat{\the\glssshorttok}{\the\glslongtok}},%
11691     access={\noexpand\@glo@textaccess},%
11692     sort={\the\glssshorttok},%
11693     short={\the\glssshorttok},%
11694     shortplural={\the\glssshorttok\noexpand\acrpluralsuffix},%
11695     shortaccess={\the\glslongtok},%
11696     long={\the\glslongtok},%
11697     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11698     first={\the\glslongtok},%
11699     firstaccess=\relax,
11700     firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11701     text={\the\glssshorttok},%
11702     textaccess={\the\glslongtok},%
11703     plural={\the\glssshorttok\noexpand\acrpluralsuffix},%
11704     symbol={\noexpand\@glo@text},%
11705     symbolaccess={\noexpand\@glo@textaccess},%
11706     symbolplural={\noexpand\@glo@plural},%
11707     firstpluralaccess=\relax,
11708     textaccess={\noexpand\@glo@shortaccess},%
11709     \the\glskeylisttok}%
11710 }%
11711 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11712 \let\@org@gls@assign@plural\gls@assign@plural
11713 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
11714 \def\gls@assign@firstpl##1##2{%
11715     \@gls@expand@field{##1}{firstpl}{##2}%
11716 }%
11717 \def\gls@assign@plural##1##2{%
11718     \@gls@expand@field{##1}{plural}{##2}%
11719 }%
11720 \def\gls@assign@symbolplural##1##2{%
11721     \@gls@expand@field{##1}{symbolplural}{##2}%
11722 }%
11723 \@do@newglossaryentry
11724 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11725 \let\gls@assign@plural\@org@gls@assign@plural
11726 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11727 }

```

teNewAcronymDef

```

11728 \renewcommand*{\FootnoteNewAcronymDef}{%
11729     \edef\@do@newglossaryentry{%
11730         \noexpand\newglossaryentry{\the\glslabeltok}%
11731         {%
11732             type=\acronymtype,%
11733             name={\noexpand\acronymfont{\the\glssshorttok}},%

```

```

11734     sort={\the\glsshorttok},%
11735     text={\the\glsshorttok},%
11736     textaccess={\the\glslongtok},%
11737     access={\noexpand\@glo@textaccess},%
11738     plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11739     short={\the\glsshorttok},%
11740     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11741     long={\the\glslongtok},%
11742     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11743     description={\the\glslongtok},%
11744     descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11745     \the\glskeylisttok
11746   }%
11747 }%
11748 \let\@org@gls@assign@plural\gls@assign@plural
11749 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11750 \let\@org@gls@assign@descplural\gls@assign@descplural
11751 \def\gls@assign@firstpl##1##2{%
11752   \@@gls@expand@field{##1}{firstpl}{##2}%
11753 }%
11754 \def\gls@assign@plural##1##2{%
11755   \@@gls@expand@field{##1}{plural}{##2}%
11756 }%
11757 \def\gls@assign@descplural##1##2{%
11758   \@@gls@expand@field{##1}{descplural}{##2}%
11759 }%
11760 \@do@newglossaryentry
11761 \let\gls@assign@plural\@org@gls@assign@plural
11762 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11763 \let\gls@assign@descplural\@org@gls@assign@descplural
11764 }

```

11NewAcronymDef

```

11765 \renewcommand*{\SmallNewAcronymDef}{%
11766   \edef\@do@newglossaryentry{%
11767     \noexpand\newglossaryentry{\the\glslabeltok}%
11768     {%
11769       type=\acronymtype,%
11770       name={\noexpand\acronymfont{\the\glsshorttok}},%
11771       access={\noexpand\@glo@symbolaccess},%
11772       sort={\the\glsshorttok},%
11773       short={\the\glsshorttok},%
11774       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11775       shortaccess={\the\glslongtok},%
11776       long={\the\glslongtok},%
11777       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11778       text={\noexpand\@glo@short},%
11779       textaccess={\noexpand\@glo@shortaccess},%
11780       plural={\noexpand\@glo@shortpl},%

```

```

11781     first={\the\glslongtok},%
11782     firstaccess=\relax,
11783     firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11784     description={\noexpand\@glo@first},%
11785     descriptionplural={\noexpand\@glo@firstplural},%
11786     symbol={\the\glsshorttok},%
11787     symbolaccess={\the\glslongtok},%
11788     symbolplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11789     \the\glskeylisttok
11790   }%
11791 }%
11792 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11793 \let\@org@gls@assign@plural\gls@assign@plural
11794 \let\@org@gls@assign@descplural\gls@assign@descplural
11795 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
11796 \def\gls@assign@firstpl##1##2{%
11797   \@@gls@expand@field{##1}{firstpl}{##2}%
11798 }%
11799 \def\gls@assign@plural##1##2{%
11800   \@@gls@expand@field{##1}{plural}{##2}%
11801 }%
11802 \def\gls@assign@descplural##1##2{%
11803   \@@gls@expand@field{##1}{descplural}{##2}%
11804 }%
11805 \def\gls@assign@symbolplural##1##2{%
11806   \@@gls@expand@field{##1}{symbolplural}{##2}%
11807 }%
11808 \@do@newglossaryentry
11809 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11810 \let\gls@assign@plural\@org@gls@assign@plural
11811 \let\gls@assign@descplural\@org@gls@assign@descplural
11812 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11813 }

```

The following are kept for compatibility with versions before 3.0:

sshortaccesskey

```
11814 \newcommand*{\glsshortaccesskey}{\glsshortkey access}%
```

pluralaccesskey

```
11815 \newcommand*{\glsshortpluralaccesskey}{\glsshortpluralkey access}%
```

lslongaccesskey

```
11816 \newcommand*{\glslongaccesskey}{\glslongkey access}%
```

pluralaccesskey

```
11817 \newcommand*{\glslongpluralaccesskey}{\glslongpluralkey access}%
```

5.5 Debugging Commands

owglonameaccess

```
11818 \newcommand*{\showglonameaccess}[1]{%
11819   \expandafter\show\csname glo@\glsdetoklabel{#1}@access\endcsname
11820 }
```

owglotextaccess

```
11821 \newcommand*{\showglotextaccess}[1]{%
11822   \expandafter\show\csname glo@\glsdetoklabel{#1}@textaccess\endcsname
11823 }
```

glopluralaccess

```
11824 \newcommand*{\showglopluralaccess}[1]{%
11825   \expandafter\show\csname glo@\glsdetoklabel{#1}@pluralaccess\endcsname
11826 }
```

wglofirstaccess

```
11827 \newcommand*{\showglofirstaccess}[1]{%
11828   \expandafter\show\csname glo@\glsdetoklabel{#1}@firstaccess\endcsname
11829 }
```

rstpluralaccess

```
11830 \newcommand*{\showglofirstpluralaccess}[1]{%
11831   \expandafter\show\csname glo@\glsdetoklabel{#1}@firstpluralaccess\endcsname
11832 }
```

glosymbolaccess

```
11833 \newcommand*{\showglosymbolaccess}[1]{%
11834   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolaccess\endcsname
11835 }
```

bolpluralaccess

```
11836 \newcommand*{\showglosymbolpluralaccess}[1]{%
11837   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolpluralaccess\endcsname
11838 }
```

owglodescaccess

```
11839 \newcommand*{\showglodescaccess}[1]{%
11840   \expandafter\show\csname glo@\glsdetoklabel{#1}@descaccess\endcsname
11841 }
```

escpluralaccess

```
11842 \newcommand*{\showglodescpluralaccess}[1]{%
11843   \expandafter\show\csname glo@\glsdetoklabel{#1}@descpluralaccess\endcsname
11844 }
```

wgloshortaccess

```
11845 \newcommand*{\showgloshortaccess}[1]{%  
11846   \expandafter\show\csname glo@glstdetoklabel{#1}@shortaccess\endcsname  
11847 }
```

ortpluralaccess

```
11848 \newcommand*{\showgloshortpluralaccess}[1]{%  
11849   \expandafter\show\csname glo@glstdetoklabel{#1}@shortpluralaccess\endcsname  
11850 }
```

owglolongaccess

```
11851 \newcommand*{\showglolongaccess}[1]{%  
11852   \expandafter\show\csname glo@glstdetoklabel{#1}@longaccess\endcsname  
11853 }
```

ongpluralaccess

```
11854 \newcommand*{\showglolongpluralaccess}[1]{%  
11855   \expandafter\show\csname glo@glstdetoklabel{#1}@longpluralaccess\endcsname  
11856 }
```

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.

```
11857 \NeedsTeXFormat{LaTeX2e}
11858 \ProvidesPackage{glossaries-babel}[2019/01/06 v4.42 (NLCT)]
```

Load tracklang to obtain language settings.

```
11859 \RequirePackage{tracklang}
11860 \let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
11861 \AnyTrackedLanguages
11862 {%
11863   \ForEachTrackedDialect{\this@dialect}{%
11864     \IfTrackedLanguageFileExists{\this@dialect}%
11865       {glossaries-}% prefix
11866       {.ldf}%
11867       {%
11868         \RequireGlossariesLang{\CurrentTrackedTag}%
11869       }%
11870     }%
11871     \PackageWarningNoLine{glossaries}%
11872     {No language module detected for ‘\this@dialect’.\MessageBreak
11873     Language modules need to be installed separately.\MessageBreak
11874     Please check on CTAN for a bundle called\MessageBreak
11875     ‘glossaries-\CurrentTrackedLanguage’ or similar}%
11876   }%
11877 }%
11878 }%
11879 {}%
```

6.1 Polyglossia Captions

Language support has now been split off into independent language modules.

```
11880 \NeedsTeXFormat{LaTeX2e}
11881 \ProvidesPackage{glossaries-polyglossia}[2019/01/06 v4.42 (NLCT)]
```

Load tracklang to obtain language settings.

```
11882 \RequirePackage{tracklang}
11883 \let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
11884 \AnyTrackedLanguages
```

```

11885 {%
11886   \ForEachTrackedDialect{\this@dialect}{%
11887     \IfTrackedLanguageFileExists{\this@dialect}%
11888     {glossaries-}% prefix
11889     {.ldf}%
11890     {%
11891       \RequireGlossariesLang{\CurrentTrackedTag}%
11892     }%
11893     {%
11894       \PackageWarningNoLine{glossaries}%
11895       {No language module detected for ‘\this@dialect’.\MessageBreak
11896       Language modules need to be installed separately.\MessageBreak
11897       Please check on CTAN for a bundle called\MessageBreak
11898       ‘glossaries-\CurrentTrackedLanguage’ or similar}%
11899     }%
11900   }%
11901 }%
11902 {}%

```

Glossary

`makeindex` An indexing application. [9](#), [13](#), [29](#), [30](#), [183](#)

`xindy` An flexible indexing application with multilingual support written in Perl. [9](#), [13](#), [29](#), [30](#), [183](#)

Change History

1.01 (2007-05-17)	numberline: numberline option added .. 7
General: Added range facility in format key	117
\writeist: Added spaces after \delimN and \delimR in ist file	164
1.04 (2007-08-03)	
General: Added \glstextformat	101
1.05 (2007-08-10)	
\glossarysection: added \@mkboth to \glossarysection	43
\gls@defglossaryentry: Changed the default value of the sort key to just the value of the name key	85
1.07 (2007-09-13)	
\@gls@link: fixed bug caused by \theglsentrycounter setting the page number too soon	114
\glsadd: fixed bug caused by \theglsentrycounter setting the page number too soon	161
1.08 (2007-10-13)	
General: Added babel support	38
listgroup: changed listgroup style to use \glsgetgrouptitle	277
altlistgroup: changed altlistgroup style to use \glsgetgrouptitle	278
1.1 (2008-02-22)	
\@glossarysection: numbered sections and auto label added	45
\@gls@tmpb: changed \toksdef to \newtoks	119
\@gls@toc: numberline added	46
\@p@glossarysection: numbered sections and auto label added	45
General: amsgen now loaded (\new@ifnextchar needed)	4
translate: translate option added	26
\setglossarysection: new	44
numberedsection: numberedsection package option added	8
1.12 (2008-03-08)	
\@GLSpl: now uses \glsentrydescplural and \glsentrysymbolplural instead of \glsentrydesc and \glsentrysymbol	130
\@Glspl@: now uses \glsentrydescplural and \glsentrysymbolplural instead of \glsentrydesc and \glsentrysymbol	130
\@glspl@: now uses \glsentrydescplural and \glsentrysymbolplural instead of \glsentrydesc and \glsentrysymbol	129
General: added check for \hypertarget separate to \hyperlink (memoir defines \hyperlink but not \hypertarget)	125
descriptionplural: new	67
\gls@defglossaryentry: Changed default first plural to be first key with s appended (was text key with s appended)	84
descriptionplural support added	84
symbolplural support added	84
\Glsentrydescplural: New	154
\glsentrydescplural: New	154
\Glsentrysymbolplural: New	155
\glsentrysymbolplural: New	155
\SetDescriptionFootnoteAcronymStyle: Added \protect before \footnote and \glslink	244
\SetFootnoteAcronymStyle: Added \protect before \footnote and \glslink	250
symbolplural: new	68

1.13 (2008-05-10)	
General: fixed bug that ignored 3rd parameter	132–139
\ACRfullpl: new	225
\Acrfullpl: new	224
\acrfullpl: new	224
\acrpluralsuffix: New	222
\gls@defglossaryentry: Changed default first value	84
Changed default firstplural value	84
Removed restriction on only using \newglossaryentry in the preamble	89
\newacronym: Removed restriction on only using \newacronym in the preamble	222
1.14 (2008-06-17)	
\@gls@hypergroup: new	272
General: added nonumberlist key to \printglossary	209
added numberedsection key to \printglossary	208
\firstacronymfont: new	225
\glsautoprefix: new	8
\glsnavhyperlink: changed \edef to \protected@edef	271
\glsnavhypertarget: added write to aux file	271
\glsnavigation: changed to only use labels for groups that are present	273
1.15 (2008-08-15)	
\@gls@link: added \glslabel	114
\gls@defglossaryentry: check for \@glo@first in description	88
check for \@glo@text in symbol	89
\gls@hypergrouprerun: new	272
\glsnavhypertarget: added check if rerun required	271
\glssettoctitle: new	37
\printglossary: changed the way the TOC title is set	193
1.16 (2008-08-27)	
\@GLS@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	128
\@GLSpl: Test glossary type is \acronymtype in addition to checking if footnote option has been used	130
\@GLs@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	127
\@GLspl@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	130
\@gls@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	126
\@glsdisp: Test glossary type is \acronymtype in addition to checking if footnote option has been used	131
\@glspl@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	129
\@glstarget: raised the hypertarget so the target text doesn't scroll off the top of the page	125
\gls@defglossaryentry: Changed def to let	84
1.17 (2008-12-26)	
\@do@esc@wrglossary: new	187
\do@seeglossary: new	191
\@glo@storeentry: new	91
\@gls@glossary: changed definition to use \index instead of \@index	184
\@glsdefaultplural: new	71
\@glsdefaultsort: new	72
\@gls@hypernumber: new	219
\@glsnoname: new	71
\@glsnonextpages: new	209
General: added xindy support	29
parent: new	69
see: new	68
\gls@defglossaryentry: added nonumberlist key	85
added parent key	85
added see key	85
Stored main part of entry format when entry is defined	89
\gls@suffixF: new	41
\gls@suffixFF: new	42
\gls@wrglossary: modified to allow for xindy support	184

<code>\glshyperlink</code> : new	160	<code>\SetDescriptionFootnoteAcronymStyle</code> : changed <code>\acronymfont</code> to use <code>\textsmaller</code> instead of <code>\smaller</code>	244
<code>\glshypernumber</code> : modified to allow material to be attached to location	219	<code>\SetFootnoteAcronymStyle</code> : changed <code>\acronymfont</code> to use <code>\textsmaller</code> instead of <code>\smaller</code>	250
<code>\glsnavhyperlink</code> : replaced <code>\hyperlink</code> to <code>\@glslink</code>	271	<code>\SetSmallAcronymStyle</code> : changed <code>\acronymfont</code> to use <code>\textsmaller</code> instead of <code>\smaller</code>	253
<code>\glsnavhypertarget</code> : replaced <code>\hypertarget</code> to <code>\@glstarget</code>	271	2.01 (2009 May 30)	
<code>\glssee</code> : new	192	<code>\@gls@link</code> : moved <code>\@do@wrglossary</code> before term is displayed to prevent unwanted whatsit	115
<code>\glsseeformat</code> : new	192	<code>\forallglossaries</code> : replaced <code>\ifthenelse</code> with <code>\ifx</code>	55
<code>\glsSetSuffixF</code> : new	42	<code>\forallglsentries</code> : replaced <code>\ifthenelse</code> with <code>\ifx</code>	56
<code>\glsSetSuffixFF</code> : new	42	<code>\glsdefmain</code> : new	16
<code>\ifglsxindy</code> : new	29	<code>\glsdescwidth</code> : changed <code>\linewidth</code> to <code>\hsize</code>	279, 301
<code>\istfilename</code> : added xindy support	40	<code>\glslistdottedwidth</code> : changed <code>\linewidth</code> to <code>\hsize</code>	279
<code>\newglossarystyle</code> : made <code>\newglossarystyle long</code>	218	<code>\glspagerlistwidth</code> : changed <code>\linewidth</code> to <code>\hsize</code>	279, 301
<code>\nopostdesc</code> : new	40	<code>\nomain</code> : added <code>\nomain</code> package option	16
<code>\nonumberlist</code> : new	69	<code>\writeist</code> : removed <code>item_02</code> - no such makeindex key	168
<code>\printglossary</code> : added check to determine if <code>\printglossary</code> is already defined	193	2.02 (2007-07-13)	
added print language to aux file	193	<code>\@printglossary</code> : suppressed warning globally rather than locally	196
order: order package option added	29	2.02 (2009-07-13)	
<code>\writeist</code> : added xindy support	164	<code>\glossarysection</code> : changed <code>\@mkboth</code> to <code>\glossarymark</code>	43
1.18 (2009-01-14)		<code>\gls@glossarymark</code> : New	44
<code>\@gls@loadlist</code> : new	10	2.03 (2009-09-23)	
<code>\@gls@loadlong</code> : new	9	<code>\@GLS@</code> : Added check for hyperfirst	128
<code>\@gls@loadsuper</code> : new	10	<code>\@GLSpl</code> : Added check for hyperfirst	130
<code>\@gls@loadtree</code> : new	10	<code>\@Gls@</code> : Added check for hyperfirst	127
<code>\gls@defglossaryentry</code> : Changed default value of sort to <code>\@glsdefaultsort</code>	85	<code>\@Glspl@</code> : Added check for hyperfirst	130
moved sort sanitization to <code>\newglossaryentry</code>	89	<code>\@gls@</code> : Added check for hyperfirst	126
<code>\glstarget</code> : new	212	<code>\@gls@link</code> : new	113
<code>\oldacronym</code> : new	221	<code>\@gls@link</code> : added <code>\leavevmode</code>	114
<code>\nolist</code> : new	10	Moved entry existence check to avoid duplicate code	114
<code>\nolong</code> : new	10	<code>\@glsdisp</code> : Added check for hyperfirst	131
<code>\sort</code> : moved sanitization to <code>\newglossaryentry</code>	67	<code>\@glspl@</code> : Added check for hyperfirst	129
<code>\nostyles</code> : new	10	<code>\gls@glossarymark</code> : Added check to see if it's already defined	44
<code>\nosuper</code> : new	10	<code>\hyperfirst</code> : new	27
<code>\notree</code> : new	10		
1.19 (2009-03-02)			
<code>\gls@clearpage</code> : new	46		
<code>\glsdisp</code> : new	131		
<code>\SetDescriptionAcronymStyle</code> : changed <code>\acronymfont</code> to use <code>\textsmaller</code> instead of <code>\smaller</code>	248		

2.04 (2009-11-10)	
\@GLS@: Changed test to check if glossary type has been identified as a list of acronyms	128
\@GLSpl@: Changed test to check if glossary type has been identified as a list of acronyms	130
\@GLs@: Changed test to check if glossary type has been identified as a list of acronyms	127
\@GLspl@: Changed test to check if glossary type has been identified as a list of acronyms	130
\@glossaryentryfield: new	90
\@glossarysubentryfield: new	90
\@gls@: Changed test to check if glossary type has been identified as a list of acronyms	126
\@glsacronymlists: new	18
\@glsdisp: Changed test to check if glossary type has been identified as a list of acronyms	131
\@glspl@: Changed test to check if glossary type has been identified as a list of acronyms	129
\@newglossaryentryposthook: new	90
\@newglossaryentryprehook: new	90
acronymlists: new	19
\DeclareAcronymList: new	18
\DefineAcronymSynonyms: new	238
\gls@defglossaryentry: added user1-6 keys	85
\glsadd: fixed bug that ignored counter	161
\Glsentryuseri: new	156
\glsentryuseri: new	156
\Glsentryuserii: new	157
\glsentryuserii: new	156
\Glsentryuseriii: new	157
\glsentryuseriii: new	157
\Glsentryuseriv: new	157
\glsentryuseriv: new	157
\Glsentryuserv: new	157
\glsentryuserv: new	157
\Glsentryuservi: new	157
\glsentryuservi: new	157
\ns@newglossary: added check to determine if \gls@<type>@display and \gls@<type>@displayfirst have been defined.	64
\SetAcronymLists: new	19
\SetDefaultAcronymDisplayStyle: new	240
\SetDefaultAcronymStyle: new	241
\SetDescriptionAcronymDisplayStyle: new	246
\SetDescriptionDUAAcronymDisplayStyle: new	244
\SetDescriptionFootnoteAcronymDisplayStyle: new	242
\SetDUADisplayStyle: new	253
\SetFootnoteAcronymDisplayStyle: new	248
\SetSmallAcronymDisplayStyle: new	251
2.05 (2010-02-06)	
\@glsdisp: Added closing brace. Patch provided by Sergiu Dotenco	131
Removed spurious brace. Patch provided by Sergiu Dotenco	131
\writeist: Added \string before opening and closing braces. Patch provided by Segiu Dotenco	169
2.06 (2010-06-14)	
\altnewglossary: new	64
\CustomAcronymFields: new	256
\CustomNewAcronymDef: new	256
\SetCustomDisplayStyle: new	256
\SetCustomStyle: new	256
2.07 (2010-07-10)	
General: glsadd format key stored in \@glsnumberformat (was mistakenly stored in \@glo@format)	160
3.0 (2010-07-12)	
\@makeglossary: Added check for savewrites	173
\gls@wrglossary: modified to take into account savewrites	184
3.0 (2010/03/31)	
\@set@glo@numformat: added 4th argument	117
3.0 (2011-04-02)	
\@do@esc@wrglossary: added check for hyper location prefix	189
modified to use new format	187
\@glossarysec: replaced \@ifundefined with \ifcsundef	7
\@do@seeglossary: Sanitize and escape cross-referencing information	191
\@gls@counterwithin: new	12

<code>\@gls@ifinlist</code> : new	47	<code>\glsadd</code> : added	
<code>\@gls@link</code> : added		<code>\@gls@saveentrycounter</code>	161
<code>\@gls@saveentrycounter</code>	115	<code>\GlsAddXdyCounters</code> : new	47
added <code>\@gls@setsort</code>	115	<code>\glseentrycounterlabel</code> : new	211
<code>\@gls@saveentrycounter</code> : new	116	<code>\glseentryitem</code> : new	211
<code>\@gls@setupsort@def</code> : new	14	<code>\Glsentrylong</code> : new	158
<code>\@gls@setupsort@standard</code> : new	13	<code>\glseentrylong</code> : new	158
<code>\@gls@setupsort@use</code> : new	14	<code>\Glsentrylongpl</code> : new	158
<code>\@gls@xdy@locationlist</code> : new	50	<code>\glseentrylongpl</code> : new	158
<code>\@glslink</code> : replaced <code>\@ifundefined</code>		<code>\Glsentryshort</code> : new	158
with <code>\ifcsundef</code>	125	<code>\glseentryshort</code> : new	158
<code>\@glsnextpages</code> : new	209	<code>\Glsentryshortpl</code> : new	158
<code>\@print@glossary</code> : replaced		<code>\glseentryshortpl</code> : new	158
<code>\@ifundefined</code> with <code>\ifcsundef</code> ..	197	<code>\glsgetgrouptitle</code> : replaced	
<code>\@printglossary</code> : added		<code>\@ifundefined</code> with <code>\ifcsundef</code> ..	215
<code>\currentglossary</code>	195	<code>\gls glossarymark</code> : replaced	
added <code>\glsnextpages</code>	195	<code>\@ifundefined</code> with <code>\ifcsundef</code> ..	44
make toctitle default to title	195	<code>\gls hyperlink</code> : changed default from	
<code>\@xdy@attributelist</code> : new	47	<code>\glseentryname</code> to <code>\glseentrytext</code> ..	160
General: added prefix to hyperlink	220	<code>\gls hypernumber</code> : replaced	
etoolbox now loaded	4	<code>\@ifundefined</code> with <code>\ifcsundef</code> ..	219
replaced <code>\@ifundefined</code> with		<code>\gls numberformat</code> : replaced	
<code>\ifcsundef</code>	35, 38, 111, 207	<code>\@ifundefined</code> with <code>\ifcsundef</code> ..	42
<code>\acrfootnote</code> : new	242	<code>\glsrefentry</code> : new	211
<code>\ACRfull</code> : added starred version	224	<code>\glsresetsubentrycounter</code> : new ...	210
<code>\Acrfull</code> : added starred version	223	<code>\glsseeitem</code> : hyperlink uses	
<code>\acrfull</code> : added starred version	222	<code>\glsseeitemformat</code> instead of	
<code>\ACRfullpl</code> : added starred version ...	225	<code>\glseentryname</code>	193
<code>\Acrfullpl</code> : added starred version ...	224	<code>\glsseeitemformat</code> : new	193
<code>\acrfullpl</code> : added starred version ...	224	<code>\gls sortnumberfmt</code> : new	14
<code>\acrlinkfootnote</code> : new	242	<code>\glsstepentry</code> : new	210
<code>\acrno linkfootnote</code> : new	242	<code>\glsstepsubentry</code> : new	210
<code>savewrites</code> : new	31	<code>\gls subentrycounterlabel</code> : new ...	211
<code>see</code> : added <code>\@glo@seeautonumberlist</code> ..	68	<code>\gls subentryitem</code> : new	211
<code>seeautonumberlist</code> : new	9	<code>theglossary</code> : replaced <code>\@ifundefined</code>	
<code>\glossarysection</code> : replaced		with <code>\ifcsundef</code>	211
<code>\@ifundefined</code> with <code>\ifcsundef</code> ..	43	<code>short</code> : new	70
<code>\glossarystyle</code> : replaced		<code>shortplural</code> : new	71
<code>\@ifundefined</code> with <code>\ifcsundef</code> ..	217	<code>\if glossaryexists</code> : replaced	
<code>\gls@codepage</code> : replaced		<code>\@ifundefined</code> with <code>\ifcsundef</code> ..	56
<code>\@ifundefined</code> with <code>\ifcsundef</code> ..	29	<code>\if glseentryexists</code> : replaced	
<code>\gls@def glossaryentry</code> : added		<code>\@ifundefined</code> with <code>\ifcsundef</code> ..	57
<code>\@gls@defsort</code>	89	<code>\istfile</code> : deprecated	182
added short and long keys	85	<code>glossaryentry</code> : new	11
replaced <code>\@ifundefined</code> with		<code>glossarysubentry</code> : new	12
<code>\ifcsundef</code>	85	<code>\newglossaryentry</code> : replaced	
<code>\gls@doclearpage</code> : replaced		<code>\DeclareRobustCommand</code> with	
<code>\@ifundefined</code> with <code>\ifcsundef</code> ..	45	<code>\newrobustcmd</code>	74

<code>\newglossarystyle</code> : replaced	
<code>\@ifundefined</code> with <code>\ifcsundef</code> .	218
<code>\ns@newglossary</code> : added	
<code>\@gls@defsortcount</code>	64
replaced <code>\@ifundefined</code> with	
<code>\ifcsundef</code>	64
<code>entrycounter</code> : new	12
<code>\oldacronym</code> : replaced <code>\@ifundefined</code>	
with <code>\ifcsundef</code>	221
compatible-2.07: compatible-2.07	
option added	31
<code>long</code> : new	71
<code>longplural</code> : new	71
<code>nonumberlist</code> : now boolean	69
<code>sort</code> : new	13
<code>counter</code> : replaced <code>\@ifundefined</code> with	
<code>\ifcsundef</code>	68
<code>counterwithin</code> : new	12
<code>\printglossary</code> : replaced	
<code>\@ifundefined</code> with <code>\ifcsundef</code> .	193
<code>\SetDescriptionFootnoteAcronymDisplayStyle</code>	
expanded options link options	242
<code>\setentrycounter</code> : added optional	
argument	216
<code>\showacronymlists</code> : new	262
<code>\showglocounter</code> : new	259
<code>\showglodesc</code> : new	260
<code>\showglodescplural</code> : new	260
<code>\showglofirst</code> : new	258
<code>\showglofirstpl</code> : new	258
<code>\showgloflag</code> : new	261
<code>\showgloindex</code> : new	261
<code>\showglolevel</code> : new	258
<code>\showglongame</code> : new	260
<code>\showgloparent</code> : new	258
<code>\showgloplural</code> : new	258
<code>\showglosort</code> : new	260
<code>\showglossaries</code> : new	262
<code>\showglossarycounter</code> : new	263
<code>\showglossaryentries</code> : new	263
<code>\showglossaryin</code> : new	262
<code>\showglossaryout</code> : new	262
<code>\showglossarytitle</code> : new	263
<code>\showglosymbol</code> : new	261
<code>\showglosymbolplural</code> : new	261
<code>\showglotext</code> : new	258
<code>\showglotype</code> : new	259
<code>\showglouserii</code> : new	259
<code>\showglouseriii</code> : new	259
<code>\showglouseriv</code> : new	259
<code>\showglouserv</code> : new	260
<code>\showglouservi</code> : new	260
<code>subentrycounter</code> : new	12
<code>\writeist</code> : added xindy-only macro	
definitions to glossary open tag	166
modified to support new format	164
3.01 (2011-04-12)	
<code>\@glswritefiles</code> : added check for	
empty glossaries	183
General: made robust	127
<code>\ACRfull</code> : made robust	224
<code>\Acrfull</code> : made robust	223
<code>\acrfull</code> : made robust	222
<code>\acrfullformat</code> : removed	
<code>\acronymfont</code> as it should already be	
set in the second argument.	223
<code>\ACRfullpl</code> : made robust	225
<code>\Acrfullpl</code> : made robust	224
<code>\acrfullpl</code> : made robust	224
<code>\ACRlong</code> : made robust	149
<code>\Acrlong</code> : made robust	148
<code>\acrlong</code> : made robust	148
<code>\ACRlongpl</code> : made robust	151
<code>\Acrlongpl</code> : made robust	150
<code>\acrlongpl</code> : made robust	150
<code>\ACRshort</code> : made robust	145
<code>\Acrshort</code> : made robust	145
<code>\acrshort</code> : made robust	144
<code>\ACRshortpl</code> : made robust	147
<code>\Acrshortpl</code> : made robust	146
<code>\acrshortpl</code> : made robust	146
<code>\Gls</code> : made robust	127
<code>\glsadd</code> : made robust	161
<code>\glsaddall</code> : made robust	161
<code>\GLSdesc</code> : made robust	136
<code>\Glsdesc</code> : made robust	136
<code>\glsdesc</code> : made robust	136
<code>\GLSdescplural</code> : made robust	137
<code>\Glsdescplural</code> : made robust	137
<code>\glsdescplural</code> : made robust	137
<code>\glsfirst</code> : made robust	133
<code>\GLSfirstplural</code> : made robust	135
<code>\Glsfirstplural</code> : made robust	135
<code>\glsfirstplural</code> : made robust	134
<code>\glslink</code> : made robust	113
<code>\GLSname</code> : made robust	136
<code>\Glsname</code> : made robust	135

<code>\glsname</code> : made robust	135	<code>\@printglossary</code> : add a way to fetch	
<code>\GLSpl</code> : made robust	130	current entry label	196
<code>\Glspl</code> : made robust	129	<code>savenumberlist</code> : new	9
<code>\glspl</code> : made robust	128	<code>ucmark</code> : new	11
<code>\GLSplural</code> : made robust	134	<code>\gls@defglossaryentry</code> : added	
<code>\GLSsymbol</code> : made robust	138	numberlist element	88
<code>\Glsymbol</code> : made robust	138	<code>\gls@save@numberlist</code> : new	193
<code>\glssymbol</code> : made robust	138	<code>\gls@wrglossary</code> : added check for	
<code>\GLSsymbolplural</code> : made robust	139	glossary file defined	185
<code>\Glsymbolplural</code> : made robust	139	<code>\glsdisplaynumberlist</code> : new	159
<code>\glssymbolplural</code> : made robust	138	<code>\glsentrycounter</code> : set default value ..	115
<code>\Glstext</code> : made robust	132	<code>\Glsentryfull</code> : fixed bug (replaced	
<code>\glstext</code> : made robust	132	<code>\glsentryshortpl</code> with	
<code>\GLSuseri</code> : made robust	140	<code>\glsentryshort</code>)	158
<code>\Glsuseri</code> : made robust	140	<code>\glsentryfullpl</code> : fixed bug (replaced	
<code>\glsuseri</code> : made robust	139	<code>\glsentryshort</code> with	
<code>\GLSuserii</code> : made robust	141	<code>\glsentryshortpl</code>)	159
<code>\Glsuserii</code> : made robust	140	<code>\glsentrynumberlist</code> : new	159
<code>\glsuserii</code> : made robust	140	<code>\glsmoveentry</code> : new	90
<code>\GLSuseriii</code> : made robust	141	<code>\glsresetsubentrycounter</code> : new ...	210
<code>\Glsuseriii</code> : made robust	141	<code>\ifglshaschildren</code> : new	58
<code>\glsuseriii</code> : made robust	141	<code>\ifglshasparent</code> : new	59
<code>\GLSuseriv</code> : made robust	142	<code>\makeglossaries</code> : added list parser ..	178
<code>\Glsuseriv</code> : made robust	142	<code>indexonlyfirst</code> : new	27
<code>\glsuseriv</code> : made robust	142	<code>\renewglossarystyle</code> : new	218
<code>\GLSuseriv</code> : made robust	143	<code>\showglossaryentries</code> : fixed misspelt	
<code>\Glsuseriv</code> : made robust	143	command	263
<code>\glsuseriv</code> : made robust	142	<code>\SmallNewAcronymDef</code> : fixed broken	
<code>\GLSuservi</code> : made robust	144	short and long plural	251
<code>\Glsuservi</code> : made robust	144	3.03 (2012/09/21)	
<code>\glsuservi</code> : made robust	143	<code>\@gls@sanitizesort</code> : new	22
		<code>\@gls@setupsort@standard</code> : used	
3.02 (2012-05-19)		<code>\@gls@sanitizesort</code>	13
<code>\glsnumlistlastsep</code> : new	160	<code>\@printglossary</code> : allow title to override	
<code>\glsnumlistsep</code> : new	160	default toctitle	195
3.02 (2012-05-21)		General: allow title to set toctitle	207
<code>\@do@wrglossary</code> : changed		<code>\glsinlinedescformat</code> : new	275
<code>\@glslocref</code> to		<code>\glsinlineemptydescformat</code> : new ..	275
<code>\theglsentrycounter</code>	190	<code>\glsinlinenameformat</code> : new	275
<code>\@do@wrglossary</code> : changed		<code>\glsinlinepostchild</code> : new	275
<code>\@do@wr@glossary</code> to test for		<code>\glsinlinesubdescformat</code> : new	275
<code>indexonlyfirst</code> option; put old		<code>\glsinlinesubnameformat</code> : new	275
<code>\@do@wr@glossary</code> code into		<code>\glspostinline</code> : replaced “.” with	
<code>\@do@wrglossary</code>	185	<code>\glspostdescription</code>	275
<code>\@gls@missingnumberlist</code> : new	71	list: added check for <code>glsnogroupskip</code> .	277
<code>\@glswritefiles</code> : added check for		<code>altlongragged4col</code> : added check for	
existence of token in case		<code>glsnogroupskip</code>	294
<code>\makeglossaries</code> has been		<code>altsuperragged4col</code> : added check for	
omitted	183	<code>glsnogroupskip</code>	313

alttree: added check for		\gls@disablepagerefexpansion: new	185
glsnogroupskip	322	\gls@numberpage: new	186
index: added check for glsnogroupskip	316	\gls@protected@pagefmts: new	185
nogroupskip: new	11	\gls@romanpage: new	186
long: added check for glsnogroupskip	280	\glsdefmain: added check for doc	
long3col: added check for		package	16
glsnogroupskip	282	\glsorg@endtheglossary: new	5
long4col: added check for		\glsorg@theglossary: new	5
glsnogroupskip	283	\PrintChanges: new	5
longragged: added check for		3.05 (2013-04-21)	
glsnogroupskip	291	@@do@esc@wrglossary: add Roman	
longragged3col: added check for		case. Fixed bugs in the else	
glsnogroupskip	292	statements	189
nopostdot: new	11	\@gls@link: added check for	
tree: added check for glsnogroupskip	317	“nohypertypes”	115
treenoname: added check for		mcolalttree: replaced ‘2’ with	
glsnogroupskip	319	\glscols	300
super: added check for glsnogroupskip	302	mcolindex: replaced ‘2’ with \glscols	296
super3col: added check for		mcolindexspannav: replaced ‘2’ with	
glsnogroupskip	304	\glscols	297
super4col: added check for		mcoltree: replaced ‘2’ with \glscols	297
glsnogroupskip	306	mcoltreenoname: replaced ‘2’ with	
superragged: added check for		\glscols	299
glsnogroupskip	309	mcoltreespannav: replaced ‘2’ with	
superragged3col: added check for		\glscols	298
glsnogroupskip	311	\gls@protected@pagefmts: added	
3.04 (2012-11-11)		Roman to list	185
altlist: replaced \newline with		\gls@Romanpage: new	186
paragraph break	277	\glsgetgrouplabel: fixed bug (typo in	
3.04 (2012-11-18)		\equal)	216
@@do@@wrglossary: changed		\nopostdesc: made robust	40
\theglsentrycounter back to		3.05 (2013/04/21)	
\@glslocref	190	\@gls@nohyperlist: new	19
@@do@esc@wrglossary: modified to		\GlsDeclareNoHyperList: new	19
compensate for possible incorrect		nohypertypes: new	19
page number	188	3.06 (2013/06/17)	
\@gls@escbsdq: unsanitize		\@xdy@main@language: Changed back to	
\gls@numberpage, \gls@alphpage,		using \languagename	29
\gls@Alphpage and		\findrootlanguage: Obsoleted	54
\gls@romanpage	118	3.07 (2013-07-05)	
\@print@glossary: Moved aux write to		\@gls@link: fixed bug that failed to find	
end of document to prevent		entry in list	115
unwanted whatsit occurring here. . .	197	\glossarypreamble: modified to work	
General: Added check for doc package	4	with \setglossarypreamble	43
added datatool-base as a required		\gls@doclearpage: added check for	
package	4	openright	45
added local key	112	\glspostdescription: Added	
\gls@Alphpage: new	185	spacefactor code	11
\gls@alphpage: new	185		

<code>\GlsSetXdyCodePage</code> : Added check for fontspec	54	<code>\glsseelist</code> : made robust	192
<code>\SetDescriptionAcronymDisplayStyle</code> : now using <code>\glsdoparenifnotempty</code>	246	<code>\ifglsdescsuppressed</code> : new	59
<code>\setglossarypreamble</code> : new	43	<code>\ifglsdesc</code> : new	59
3.08a (2013-08-30)		<code>\ifglsdescsymbol</code> : new	59
list: updated list style to use <code>\glossentry</code> and <code>\subglossentry</code>	276	<code>altlongragged4col</code> : updated to use <code>\glossentry</code> and <code>\subglossentry</code>	294
listdotted: updated listdotted style to use <code>\glossentry</code> and <code>\subglossentry</code>	278	<code>alttree</code> : updated to use <code>\glossentry</code> and <code>\subglossentry</code>	321
altlist: updated altlist style to use <code>\glossentry</code> and <code>\subglossentry</code>	277	index: added paragraph break at end of environment	315
inline: updated inline style to use <code>\glossentry</code> and <code>\subglossentry</code>	274	updated to use <code>\glossentry</code> and <code>\subglossentry</code>	315
3.08a (2013-09-28)		long: updated to use <code>\glossentry</code> and <code>\subglossentry</code>	280
<code>\@glo@storeentry</code> : no longer need to check for special characters in any of the fields other than sort	91	<code>longragged</code> : updated to use <code>\glossentry</code> and <code>\subglossentry</code>	291
updated for <code>\glossentry</code>	91	<code>longragged3col</code> : updated to use <code>\glossentry</code> and <code>\subglossentry</code>	292
<code>\@glossaryentryfield</code> : switched to <code>\glossentry</code>	90	tree: updated to use <code>\glossentry</code> and <code>\subglossentry</code>	317
<code>\@glossarysubentryfield</code> : switched to <code>\subglossentry</code>	90	<code>\setglossarystyle</code> : new	217
General: added <code>nogroupskip</code> key to <code>\printglossary</code>	208	<code>\setglossentrycompatibility</code> : new	214
removed definition of <code>\@glossaryentryfield</code>	364	<code>superragged</code> : updated to use <code>\glossentry</code> and <code>\subglossentry</code>	309
removed definition of <code>\@glossarysubentryfield</code>	364	3.09a (2013-10-09)	
<code>\compatibleglossentry</code> : new	212	<code>\@gls@assign@symbolplural@field</code> : new	22
<code>\compatiblesubglossentry</code> : new	213	<code>\@gls@default@value</code> : new	67
<code>\glossaryentryfield</code> : deprecated	214	<code>\Glsentrydesc</code> : made robust	154
<code>\Glossentrydesc</code> : new	213	<code>\Glsentrydescplural</code> : made robust ..	154
<code>\glossentrydesc</code> : new	213	<code>\Glsentryfirst</code> : made robust	155
<code>\Glossentryname</code> : new	213	<code>\Glsentryfirstplural</code> : made robust ..	155
<code>\glossentryname</code> : new	212	<code>\Glsentryfull</code> : made robust	158
<code>\Glossentrysymbol</code> : new	213	<code>\Glsentryfullpl</code> : made robust	159
<code>\glossentrysymbol</code> : new	213	<code>\Glsentrylong</code> : made robust	158
<code>\gls@assign@desc@field</code> : new	21	<code>\Glsentrylongpl</code> : made robust	158
<code>\gls@assign@descplural@field</code> : new	21	<code>\Glsentryname</code> : made robust	153
<code>\gls@assign@field</code> : new	73	<code>\Glsentryplural</code> : made robust	154
<code>\gls@ifnotmeasuring</code> : new	92	<code>\Glsentryshort</code> : made robust	158
<code>\glsaddallunused</code> : new	161	<code>\Glsentryshortpl</code> : made robust	158
<code>\glsexpandfields</code> : new	73	<code>\Glsentrysymbol</code> : made robust	155
<code>\glsnoexpandfields</code> : new	74	<code>\Glsentrysymbolplural</code> : made robust	155
<code>\glssee</code> : made robust	192	<code>\Glsentrytext</code> : made robust	154
<code>\glsseeformat</code> : made robust	192	<code>\Glsentryuseri</code> : made robust	156
<code>\glsseeitem</code> : made robust	193	<code>\Glsentryuserii</code> : made robust	157
		<code>\Glsentryuseriii</code> : made robust	157
		<code>\Glsentryuseriv</code> : made robust	157
		<code>\Glsentryuserv</code> : made robust	157
		<code>\Glsentryuservi</code> : made robust	157

<code>\glstextup: new</code>	222	<code>\@Gls@: add \glsifplural,</code>	
<code>\ifglshassymbol: changed test to check</code>		<code>\glscapscase, \glscustomtext and</code>	
<code>for \@gls@default@symbol</code>	59	<code>\glsinsert</code>	127
3.10a (2013-09-28)		change to using <code>\glentryfmt</code> style	
<code>\gls@assign@type@field: new</code>	21	commands	127
3.10a (2013-10-13)		removed <code>\makefirstuc</code> (now dealt	
<code>\@gls@keymap: new</code>	76	with in <code>\glentryfmt</code>)	127
<code>\@gls@provide@newglossary: new</code> ...	62	<code>\@Glspl@: add \glsifplural,</code>	
<code>\@gls@writedef: new</code>	75	<code>\glscapscase, \glscustomtext and</code>	
<code>\@glsdefaultplural: Obsolete</code>	71	<code>\glsinsert</code>	129
<code>\@glsnodesc: new</code>	71	change to using <code>\glentryfmt</code> style	
<code>\@print@glossary: Added</code>		commands	130
<code>providecommand code to aux file</code> ..	197	removed <code>\makefirstuc</code> (now dealt	
<code>\gls@defglossaryentry: Changed to</code>		with in <code>\glentryfmt</code>)	130
<code>using \@gls@default@value</code>	84, 85	<code>\@acrlong: added \glslabel,</code>	
<code>new</code>	84	<code>\glsifplural, \glscapscase,</code>	
<code>\gls>writedefhook: new</code>	83	<code>\glsinsert and \glscustomtext</code>	363
<code>\makeglossaries: Added</code>		<code>\@acrshort: added \glslabel,</code>	
<code>providecommand code to aux file</code> ..	176	<code>\glsifplural, \glscapscase,</code>	
<code>\new@glossaryentry: new</code>	74	<code>\glsinsert and \glscustomtext</code>	362
<code>\ns@newglossary: added</code>		<code>\@gls@: add \glslabel, \glsifplural,</code>	
<code>\@gls@provide@newglossary</code>	64	<code>\glscapscase, \glscustomtext and</code>	
3.11a (2013-10-15)		<code>\glsinsert</code>	126
<code>\@ACRlong: added \glslabel,</code>		change to using <code>\glentryfmt</code> style	
<code>\glsifplural, \glscapscase,</code>		commands	126
<code>\glsinsert and \glscustomtext</code>	364	<code>\@gls@noexpand@fields: Fixed bug</code>	
<code>\@ACRshort: added \glslabel,</code>		<code>expand replaced with noexpand</code>	72
<code>\glsifplural, \glscapscase,</code>		<code>\@glsdisp: add \glslabel,</code>	
<code>\glsinsert and \glscustomtext</code>	362	<code>\glsifplural, \glscapscase,</code>	
<code>\@Acrlong: added \glslabel,</code>		<code>\glscustomtext and \glsinsert</code>	131
<code>\glsifplural, \glscapscase,</code>		change to using <code>\glentryfmt</code> style	
<code>\glsinsert and \glscustomtext</code>	363	commands	131
<code>\@Acrshort: added \glslabel,</code>		<code>\@glspl@: add \glslabel,</code>	
<code>\glsifplural, \glscapscase,</code>		<code>\glsifplural, \glscapscase,</code>	
<code>\glsinsert and \glscustomtext</code>	362	<code>\glscustomtext and \glsinsert</code>	129
<code>\@GLS@: add \glslabel, \glsifplural,</code>		change to using <code>\glentryfmt</code> style	
<code>\glscapscase, \glscustomtext and</code>		commands	129
<code>\glsinsert</code>	128	General: added <code>\glslabel,</code>	
change to using <code>\glentryfmt</code> style		<code>\glsifplural, \glscapscase,</code>	
commands	128	<code>\glsinsert and</code>	
removed <code>\MakeUppercase</code> (now		<code>\glscustomtext</code>	144–151
moved to <code>\glentryfmt</code>)	128	changed to just use	
<code>\@GLSpl: add \glslabel,</code>		<code>\Glsentrydescplural</code>	137
<code>\glsifplural, \glscapscase,</code>		changed to just use	
<code>\glscustomtext and \glsinsert</code>	130	<code>\glentrydescplural</code>	137
change to using <code>\glentryfmt</code> style		changed to just use <code>\Glsentrydesc</code> .	136
commands	130	changed to just use	
removed <code>\MakeUppercase</code> as now		<code>\glentrydesc</code>	136, 137
dealt with in <code>\glentryfmt</code>	130		

changed to just use		<code>\glsdisplay</code> : obsolete	110
<code>\Glsentryfirstplural</code>	135	<code>\glsdisplayfirst</code> : obsolete	110
changed to just use		<code>\glsgetgroupfmt</code> : new	105
<code>\glsentryfirstplural</code>	134, 135	<code>\glsgetgrouptitle</code> : Added check in	
changed to just use <code>\Glsentryfirst</code>	133	case non-Latin alphabet in use	215
changed to just use <code>\glsentryfirst</code>	133	<code>\gls glossarymark</code> : replaced	
changed to just use <code>\Glsentryname</code>	136	<code>\MakeUppercase</code> with	
changed to just use		<code>\mfirstucMakeUppercase</code>	44
<code>\glsentryname</code>	135, 136	<code>\glsnavigation</code> : switched to using	
changed to just use <code>\Glsentryplural</code>	134	<code>\@gls@getgrouptitle</code>	273
changed to just use <code>\glsentryplural</code>	134	<code>\ifglshasdesc</code> : replaced <code>\ifdefempty</code>	
changed to just use		with <code>\ifcempty</code>	59
<code>\Glsentrysymbolplural</code>	139	<code>\ifglshaslong</code> : new	60
changed to just use		<code>\ifglshasshort</code> : new	60
<code>\glsentrysymbolplural</code>	139	<code>\ifglshassymbol</code> : replaced	
changed to just use <code>\Glsentrysymbol</code>	138	<code>\ifdefempty</code> with <code>\ifcempty</code>	59
changed to just use <code>\glsentrysymbol</code>	138	<code>\ifglused</code> : replaced <code>\ifthenelse</code> with	
Changed to just use <code>\Glsentrytext</code>	133	<code>\ifbool</code>	57
changed to just use <code>\glsentrytext</code>	132	<code>\longnewglossaryentry</code> : new	83
changed to just use		<code>\ns@newglossary</code> : replaced	
<code>\Glsentryuseriii</code>	141	<code>\glsdisplay</code> and	
changed to just use		<code>\glsdisplayfirst</code> with	
<code>\glsentryuseriii</code>	141, 142	<code>\glsentryfmt</code>	64
changed to just use <code>\Glsentryuserii</code>	140	compatible-3.07: <code>cnew</code>	31
changed to just use		<code>\SetCustomDisplayStyle</code> : updated to	
<code>\glsentryuserii</code>	140, 141	use <code>\defglsentryfmt</code>	256
changed to just use <code>\Glsentryuseriv</code>	142	<code>\SetDefaultAcronymDisplayStyle</code> :	
changed to just use <code>\glsentryuseriv</code>	142	changed to use <code>\defglsentryfmt</code>	240
changed to just use <code>\Glsentryuseri</code>	140	<code>\SetDescriptionAcronymDisplayStyle</code> :	
changed to just use		updated to use <code>\defglsentryfmt</code>	246
<code>\glsentryuseri</code>	139, 140	<code>\SetDescriptionDUAAcronymDisplayStyle</code> :	
changed to just use <code>\Glsentryuservi</code>	144	updated to use <code>\defglsentryfmt</code>	244
changed to just use		<code>\SetDescriptionFootnoteAcronymDisplayStyle</code> :	
<code>\glsentryuservi</code>	143, 144	updated to use <code>\defglsentryfmt</code>	242
changed to just use <code>\Glsentryuserv</code>	143	<code>\SetDUADisplayStyle</code> : updated to use	
changed to just use <code>\glsentryuserv</code>	143	<code>\defglsentryfmt</code>	253
Now requires <code>textcase</code>	4	<code>\SetFootnoteAcronymDisplayStyle</code> :	
acronymlists: replaced		updated to use <code>\defglsentryfmt</code>	248
<code>\@addtoacronymlists</code> with		<code>\SetSmallAcronymDisplayStyle</code> :	
<code>\DeclareAcronymList</code>	19	updated to use <code>\defglsentryfmt</code>	251
<code>\defglsdisplay</code> : obsolete	110	<code>\setupglossaries</code> : new	34
<code>\defglsdisplayfirst</code> : obsolete	110	<code>\showglong</code> : new	261
<code>\defglsentryfmt</code> : new	62	<code>\showgshort</code> : new	261
<code>\forglentries</code> : replaced <code>\ifx</code> with		numbers: new	33
<code>\ifdefempty</code>	56	symbols: new	33
<code>\gls@assign@desc</code> : new	83		
<code>\gls@defglossaryentry</code> : Fixed default			
counter if none supplied	88	<code>\gls@defglossaryentry</code> : added	
<code>\gls@doentryfmt</code> : new	62	<code>\glslabel</code>	84
		<code>\glsaddkey</code> : new	78

3.13a (2013-11-05)		
\@gls@assign@symbol@field: changed to use \glssetnoexpandfield	22	
\@gls@assign@symbolplural@field: changed to use \glssetnoexpandfield	22	
\@gls@link: removed \relax	115	
\@gls@notranslatorhook: new	25	
\@gls@setupsort@standard: moved \@gls@santizesort to \glsprestandardsort	13	
ucmark: added check for memoir	11	
see: added \gls@checkseeallowed ...	68	
\glossarysection: changed \glossarymark to \gls glossarymark	44	
\glossarystyle: fixed bug caused by using \ifdef instead of \ifcdef .	217	
\gls@assign@desc@field: changed to use \glssetnoexpandfield	21	
\gls@assign@descplural@field: changed to use \glssetnoexpandfield	21	
\gls@assign@name@field: changed to use \glssetnoexpandfield	22	
\gls@assign@type@field: changed to use \glssetexpandfield	21	
\gls@checkseeallowed: new	69	
\glsaddallunused: set default to \@glo@types	161	
\Glsentryfull: changed to use \acrfullformat	158	
\glsentryfull: changed to use \acrfullformat	158	
\Glsentryfullpl: changed to use \acrfullformat	159	
\glsentryfullpl: changed to use \acrfullformat	159	
\gls glossarymark: renamed \glossarymark to \gls glossarymark to avoid conflict with memoir	44	
\glsprestandardsort: new	13	
\glssetexpandfield: new	21	
\glssetnoexpandfield: new	21	
altsuper4colheader: switched to \tabularnewline	307	
altsuper4colheaderborder: switched to \tabularnewline	308	
	long: switched to \tabularnewline .. 280	
	long3col: switched to \tabularnewline	281
	long3colheader: switched to \tabularnewline	282
	long3colheaderborder: switched to \tabularnewline	282
	long4col: switched to \tabularnewline	283
	long4colheader: switched to \tabularnewline	283
	longheader: switched to \tabularnewline	281
	longheaderborder: switched to \tabularnewline	281
	\SetFootnoteAcronymDisplayStyle: fixed missing argument bug	249
	super: switched to \tabularnewline .	302
	super3col: switched to \tabularnewline	304
	super3colheader: switched to \tabularnewline	304
	super4col: switched to \tabularnewline	305
	super4colheader: switched to \tabularnewline	306
	super4colheaderborder: switched to \tabularnewline	306
	superheader: switched to \tabularnewline	303
	superheaderborder: switched to \tabularnewline	303
3.14a (2013-11-12)		
\@glswritefiles: renamed \glswritefiles to \@glswritefiles and used “savewrites” option to set \glswritefiles	183	
General: new	264	
acronyms: new	17	
\gls@def glossaryentry: added check for existence of default glossary	85	
set the default for firstplural to be the value of plural	87	
xindygloss: new	30	
\longprovideglossaryentry: new ...	84	
compatible-2.07: added check for 2.07 before setting 3.07 compatibility	31	
notranslate: new	26	

\provideglossaryentry:new	74	index:new	34
4.0 (2013-11-14)		\newacronymstyle:new	228
\gls@defglossaryentry:added check		long-sc-short:new	231
for first key	87	long-sc-short-desc:new	232
super: fixed typo in \subglossentry		long-short:new	229
(\glossentrydesc)	302	long-short-desc:new	232
4.01 (2013-11-16)		long-sm-short:new	231
General: fixed non-value options so that		long-sm-short-desc:new	232
they can be passed to document class	8	long-sp-short-desc:new	232
\CustomAcronymFields: inserted		footnote:new	236
missing comma	256	footnote-desc:new	238
4.02 (2013-12-05)		footnote-sc:new	237
\@acrfull: now using \acrfullfmt	223	footnote-sc-desc:new	238
\@gls@indexdef:new	34	footnote-sm:new	237
\@gls@numbersdef:new	33	footnote-sm-desc:new	238
\@gls@symbolsdef:new	33	\setacronymstyle:new	227
General: Removed \acronymfont	148–151	\SetDescriptionAcronymDisplayStyle:	
\ACRfullfmt:new	224	Moved check for empty custom text to	
\Acrfullfmt:new	223	prevent unwanted parenthetical	
\acrfullfmt:new	223	material	246
\ACRfullplfmt:new	225	\SetDescriptionFootnoteAcronymDisplayStyle:	
\Acrfullplfmt:new	225	Moved check for empty custom text to	
\acrfullplfmt:new	224	prevent unwanted parenthetical	
\acronymentry:new	227	material	242
sanitize: fixed bug that caused an error		\SetFootnoteAcronymDisplayStyle:	
here	25	Moved check for empty custom text to	
sc-short-long:new	231	prevent unwanted parenthetical	
sc-short-long-desc:new	233	material	248
\Genacrfullformat:new	109	\SetGenericNewAcronym:new	226
\genacrfullformat:new	109	\SetSmallAcronymDisplayStyle:	
\GenericAcronymFields:new	227	Moved check for empty custom text to	
\Genplacrfullformat:new	110	prevent unwanted parenthetical	
\genplacrfullformat:new	109	material	251
\Glsentryfull: bug fix: added missing		dua:new	234
\acronymfont	158	dua-desc:new	235
\glsentryfull: bug fix: added missing		numberedsection: added nameref	
\acronymfont	158	option	8
\Glsentryfullpl: bug fix: added		4.02 (2013-13-05)	
missing \acronymfont	159	\makeglossaries: made preamble only	178
\glsentryfullpl: bug fix: added		4.03 (2014-01-17)	
missing \acronymfont	159	General: changed default to \@empty	
\glsgenacfmt:new	107	instead of \relax	31
\GlsUseAcrEntryDispStyle:new	228	4.03 (2014-01-20)	
\GlsUseAcrStyleDefs:new	228	\@do@esc@wrglossary: added	
short-long:new	230	\glsdetoklabel	190
short-long-desc:new	233	\@do@noesc@wrglossary: added	
xindynoglsnumbers:new	30	\glsdetoklabel	187
sm-short-long:new	231	\@ACRlong: removed \glslabel	
sm-short-long-desc:new	233	(defined in \@gls@link)	364

<code>\@ACRshort: removed \glslabel</code> (defined in <code>\@gls@link</code>)	362	<code>\Genplacrfullformat: redefined to use</code> accessibility information	361
<code>\@Acrlong: removed \glslabel</code> (defined in <code>\@gls@link</code>)	363	<code>\genplacrfullformat: redefined to use</code> accessibility information	361
<code>\@Acrshort: removed \glslabel</code> (defined in <code>\@gls@link</code>)	362	<code>\glossentryname: added</code> <code>\glsdetoklabel</code>	212
<code>\@GLS@: removed \glslabel</code> (defined in <code>\@gls@link</code>)	128	<code>\gls@defglossaryentry: added</code> <code>\glsdetoklabel</code>	84
<code>\@GLSpl: removed \glslabel</code> (defined in <code>\@gls@link</code>)	130	replaced #1 with <code>\@gls@label</code>	85
<code>\@Gls@: removed \glslabel</code> (defined in <code>\@gls@link</code>)	127	replaced <code>\ifthenelse</code> with <code>\ifdefequal</code>	86
<code>\@Gls@entry@field: new</code>	152	<code>\glsadd: added \glsdetoklabel</code>	161
<code>\@Glspl@: removed \glslabel</code> (defined in <code>\@gls@link</code>)	129	<code>\glsaddkey: switched to using</code> <code>\@gls@field@link</code>	79
<code>\@acrlong: removed \glslabel</code> (defined in <code>\@gls@link</code>)	363	<code>\glsdetoklabel: new</code>	57
<code>\@acrshort: removed \glslabel</code> (defined in <code>\@gls@link</code>)	362	<code>\glsdisplaynumberlist: added</code> <code>\glsdetoklabel</code>	159
<code>\@gls@: removed \glslabel</code> (defined in <code>\@gls@link</code>)	126	<code>\glsdoifexistsorwarn: new</code>	58
<code>\@gls@access@display: new</code>	350	<code>\glsentryaccess: switched to using</code> <code>\@gls@entry@field</code>	348
<code>\@gls@entry@field: new</code>	152	<code>\glsentrydescaccess: switched to</code> using <code>\@gls@entry@field</code>	349
<code>\@gls@fetchfield: new</code>	76	<code>\glsentrydescpluralaccess: switched</code> to using <code>\@gls@entry@field</code>	349
<code>\@gls@field@link: new</code>	132	<code>\glsentryfirstaccess: switched to</code> using <code>\@gls@entry@field</code>	349
<code>\@gls@link: added \glsdetoklabel</code>	114	<code>\glsentryfirstplural: added</code> <code>\glsdetoklabel</code>	155
moved <code>\@gls@link@opts</code> and <code>\@gls@link@label</code> to <code>\@gls@link</code>	115	<code>\glsentrylongaccess: switched to</code> using <code>\@gls@entry@field</code>	350
<code>\@gls@writedef: added</code> <code>\glsdetoklabel</code>	75	<code>\glsentrylongpluralaccess: switched</code> to using <code>\@gls@entry@field</code>	350
<code>\@glsdisp: removed \glslabel</code> (defined in <code>\@gls@link</code>)	131	<code>\glsentrypluralaccess: switched to</code> using <code>\@gls@entry@field</code>	349
<code>\@Glspl@: removed \glslabel</code> (defined in <code>\@gls@link</code>)	129	<code>\glsentryshortaccess: switched to</code> using <code>\@gls@entry@field</code>	349
<code>\@printglossary: added</code> <code>\glsdetoklabel</code>	196	<code>\glsentryshortpluralaccess:</code> switched to using <code>\@gls@entry@field</code>	349
General: removed <code>\glslabel</code> (defined in <code>\@gls@link</code>)	144	<code>\glsentrysymbolaccess: switched to</code> using <code>\@gls@entry@field</code>	349
<code>sc-short-long-desc: redefined to use</code> accessibility information	368	<code>\glsentrysymbolpluralaccess:</code> switched to using <code>\@gls@entry@field</code>	349
<code>\compatibleglossentry: added</code> <code>\glsdetoklabel</code>	344	<code>\glsentrytextaccess: switched to</code> using <code>\@gls@entry@field</code>	349
<code>\compatiblesubglossentry: added</code> <code>\glsdetoklabel</code>	345	<code>\glsngenacfmt: redefined to use</code> accessibility information	359
<code>\Genacrfullformat: redefined to use</code> accessibility information	361		
<code>\genacrfullformat: redefined to use</code> accessibility information	361		

<code>\glsgenentryfmt</code> : redefined to use accessibility information	356	<code>sm-short-long-desc</code> : redefined to use accessibility information	368
<code>\glshyperlink</code> : added <code>\glsdetoklabel</code>	160	<code>long-sc-short-desc</code> : redefined to use accessibility information	367
<code>\glslocalreset</code> : added <code>\glsdetoklabel</code>	93	<code>long-short</code> : redefined to use accessibility information	365
<code>\glslocalunset</code> : added <code>\glsdetoklabel</code>	93	<code>long-short-desc</code> : redefined to use accessibility information	367
<code>\glsmoveentry</code> : added <code>\glsdetoklabel</code>	90	<code>long-sm-short-desc</code> : redefined to use accessibility information	367
replaced <code>\ifthenelse</code> with <code>\ifdefequal</code>	90	<code>footnote</code> : redefined to use accessibility information	371
<code>\glsrefentry</code> : added <code>\glsdetoklabel</code>	211	<code>footnote-desc</code> : redefined to use accessibility information	373
<code>\glsreset</code> : added <code>\glsdetoklabel</code> ...	93	<code>footnote-sc</code> : redefined to use accessibility information	373
<code>\glsseelist</code> : added <code>\expandafter</code> commands	192	<code>footnote-sc-desc</code> : redefined to use accessibility information	374
<code>\glsstepentry</code> : added <code>\glsdetoklabel</code>	210	<code>footnote-sm</code> : redefined to use accessibility information	373
<code>\glsstepsubentry</code> : added <code>\glsdetoklabel</code>	210	<code>footnote-sm-desc</code> : redefined to use accessibility information	374
<code>\glsunset</code> : added <code>\glsdetoklabel</code> ...	93	<code>\renewacronymstyle</code> : new	228
<code>short-long</code> : commented spurious EOL redefined to use accessibility information	230 366	<code>\showglocounter</code> : added <code>\glsdetoklabel</code>	259
<code>short-long-desc</code> : redefined to use accessibility information	368	<code>\showglodesc</code> : added <code>\glsdetoklabel</code>	260
<code>\ifglshdescsuppressed</code> : added <code>\glsdetoklabel</code>	59	<code>\showglodescaccess</code> : added <code>\glsdetoklabel</code>	380
fixed typo	59	<code>\showglodescplural</code> : added <code>\glsdetoklabel</code>	260
<code>\ifglshentryexists</code> : added <code>\glsdetoklabel</code>	57	<code>\showglodescpluralaccess</code> : added <code>\glsdetoklabel</code>	380
<code>\ifglshaschildren</code> : added <code>\glsdetoklabel</code>	58	<code>\showglofirst</code> : added <code>\glsdetoklabel</code>	258
<code>\ifglshasdesc</code> : added <code>\glsdetoklabel</code>	59	<code>\showglofirstaccess</code> : added <code>\glsdetoklabel</code>	380
<code>\ifglshasfield</code> : new	60	<code>\showglofirstpl</code> : added <code>\glsdetoklabel</code>	258
<code>\ifglshaslong</code> : added <code>\glsdetoklabel</code>	60	<code>\showglofirstpluralaccess</code> : added <code>\glsdetoklabel</code>	380
<code>\ifglshasparent</code> : added <code>\glsdetoklabel</code>	59	<code>\showgloflag</code> : added <code>\glsdetoklabel</code>	261
<code>\ifglshasshort</code> : added <code>\glsdetoklabel</code>	60	<code>\showgloindex</code> : added <code>\glsdetoklabel</code>	261
<code>\ifglshassymbol</code> : added <code>\glsdetoklabel</code>	59	<code>\showglolevel</code> : added <code>\glsdetoklabel</code>	258
replaced <code>\ifcsempy</code> with <code>\ifdefempty</code> and replaced <code>\ifx</code> with <code>\ifdefequal</code>	59	<code>\showglolong</code> : added <code>\glsdetoklabel</code>	261
<code>\ifglshused</code> : added <code>\glsdetoklabel</code> ..	57	<code>\showglolongaccess</code> : added <code>\glsdetoklabel</code>	381

\showglolongpluralaccess: added		redefined to use accessibility	
\glsdetoklabel	381	information	371
\showglongname: added \glsdetoklabel	260	4.04 (2014-03-04)	
\showglongnameaccess: added		\@gls@getcounterprefix: added	
\glsdetoklabel	380	warning if no prefix can be formed .	191
\showgloparent: added		4.04 (2014-03-06)	
\glsdetoklabel	258	\@gls@noidx@nosanitizesort: new .	23
\showgloplural: added		\@gls@noidx@sanitizesort: new ...	22
\glsdetoklabel	258	\@gls@nosanitizesort: new	22
\showglopluralaccess: added		\@gls@sanitizesort: new	22
\glsdetoklabel	380	\@glo@addchildren: new	198
\showgloshort: added		\@glo@do@sortentries: new	199
\glsdetoklabel	261	\@glo@grabfirst: new	204
\showgloshortaccess: added		\@glo@sortedinsert: new	199
\glsdetoklabel	381	\@glo@sortentries: new	198
\showgloshortpluralaccess: added		\@glo@sorthandler@case: new	200
\glsdetoklabel	381	\@glo@sorthandler@letter: new ...	200
\showglosort: added \glsdetoklabel	260	\@glo@sorthandler@nocase: new ...	200
\showglosymbol: added		\@glo@sorthandler@word: new	200
\glsdetoklabel	261	\@glo@sortmacro@case: new	201
\showglosymbolaccess: added		\@glo@sortmacro@def: new	202
\glsdetoklabel	380	\@glo@sortmacro@def@do: new	202
\showglosymbolplural: added		\@glo@sortmacro@letter: new	201
\glsdetoklabel	261	\@glo@sortmacro@nocase: new	202
\showglosymbolpluralaccess: added		\@glo@sortmacro@standard: new ...	201
\glsdetoklabel	380	\@glo@sortmacro@use: new	203
\showglotext: added \glsdetoklabel	258	\@glo@sortmacro@word: new	201
\showglotextaccess: added		\@gls@noidx@do: new	204
\glsdetoklabel	380	\@gls@noidx@getgrouptitle: new ..	216
\showglotype: added \glsdetoklabel	259	\@gls@noref@warn: new	182
\showglouser: added		\@gls@reference: new	207
\glsdetoklabel	259	\@gls@warnonglossdefined: new	20
\showglouserii: added		\@gls@warnontheGLOSSdefined: new .	21
\glsdetoklabel	259	\@no@makeglossaries: new	182
\showglouseriii: added		\@print@glossary: new	196
\glsdetoklabel	259	\@print@noidx@glossary: new	203
\showglouseriv: added		\@printgloss@setsort: new	194
\glsdetoklabel	259	\@printglossary: new	194
\showglouserv: added		General: added sort key to printgloss	
\glsdetoklabel	260	group	209
\showglouservi: added		\compatibleglossentry: changed	
\glsdetoklabel	260	\newcommand to \def as is may or	
dua: fixed bug in \acrfullfmt	235	may not be defined	344
fixed bug in \Acrfullplfmt	235	\compatiblesubglossentry: changed	
fixed bug in \acrfullplfmt	235	\newcommand to \def as is may or	
redefined to use accessibility		may not be defined	345
information	369	\defglsdisplayfirst: fixed unwanted	
dua-desc: commented spurious EOL ..	236	space	111
		\glo@grabfirst: new	204

\gls@defglossaryentry: replaced \ifx with \ifdefvoid	89	4.08 (2014-07-30)	\@ACRlong: added	
\glsnoidxdisplayloc: new	206		\do@gls@link@checkfirsthyper	363
\glsnoidxdisplaylocclishandler: new	206		\@ACRshort: added	
			\do@gls@link@checkfirsthyper	362
\glsnoidxloclist: new	206		\@Acrlong: added	
\glsnoidxloclisthandler: new	206		\do@gls@link@checkfirsthyper	363
\glsnoidxstripaccents: new	23		\@Acrshort: added	
alltree: moved hangindent and parindent assignments outside level test	321		\do@gls@link@checkfirsthyper	362
\makeglossaries: Moved definition of \glswrite to \makeglossaries ..	176		\@GLS@: moved \glsifhyper	128
			moved check for first use to	
\makenoidxglossaries: new	178		\@gls@link	128
\printglossary: changed to use new \@printglossary	194		\@GLSpl: moved \glsifhyper	130
			moved check for first use to	
\printnoidxglossaries: new	194		\@gls@link	130
\printnoidxglossary: new	194		\@Gls@: moved \glsifhyper	127
			moved check for first use to	
\showgloclist: new	262		\@gls@link	127
\warn@noprintglossary: Activate warning in \makeglossaries	193		\@Glspl@: moved \glsifhyper	130
			moved check for first use to	
\writeist: checked for definition of \glswrite	164, 168		\@gls@link	130
			\@acrlong: added	
4.06 (2014-03-12)			\do@gls@link@checkfirsthyper	363
\@GLS@: added \glsifhyper	128		\@acrshort: added	
			\do@gls@link@checkfirsthyper	361
\@GLSpl: added \glsifhyper	130		\@closegls: new	173
\@Gls@: added \glsifhyper	127		\@gls@: moved \glsifhyper	126
			moved check for first use to	
\@Glspl@: added \glsifhyper	130		\@gls@link	126
\@gls@: added \glsifhyper	126		\@gls@automake: new	174
\@gls@numbersdef: added hook to set toc title	33		\@gls@doautomake: new	30
			\@gls@field@link: added assignment of	
\@gls@symbolsdef: added hook to set toc title	33		\do@gls@link@checkfirsthyper	132
\@glsdisp: added \glsifhyper	131		\@gls@forbidtexext: new	62
\@Glspl@: added \glsifhyper	129		\@gls@hyp@opt: new	112
General: added \glsifhyper	144–151		\@gls@link: removed redundancy	115
acronym: added hook to set toc title	17		renamed \gls@type to \glstype ...	115
acronyms: added hook to set toc title ...	17		\@gls@link@checkfirsthyper: new .	114
\glsdefmain: added hook to set toc title	16		\@glsdisp: moved \glsifhyper	131
4.07 (2014-04-04)			moved check for first use to	
\@glossarysection: added optional argument when using unstarred version	45		\@gls@link	131
\@gls@noidx@do: added \global in case it's used in a tabular-like style	204		\@Glspl@: moved \glsifhyper	129
			moved check for first use to	
\Acrfullplfmt: fixed no case change bug	225		\@gls@link	129
\glsletentryfield: new	152		\@ignored@glossaries: new	66
			General: added entrycounter option to printgloss family	208

added nopostdot option to printgloss family	208	removed \@sGLstext	132
added subentrycounter option to printgloss family	208	removed \@sGLSuseriii	141
explicitly initialise hyper key	112	removed \@sGlsuseriii	141
moved \glsifhyper	144–151	removed \@sglsuseriii	141
removed \@sACRlongpl	151	removed \@sGLSuserii	141
removed \@sAcrlongpl	150	removed \@sGlsuserii	140
removed \@sacrlongpl	150	removed \@sGLSuseriv	142
removed \@sACRlong	149	removed \@sGlsuseriv	142
removed \@sAcrlong	148	removed \@sglsuseriv	142
removed \@sacrlong	148	removed \@sGLSuseri	140
removed \@sACRshortpl	147	removed \@sGlsuseri	140
removed \@sAcrshortpl	147	removed \@sglsuseri	139
removed \@sacrshortpl	146	removed \@sGLSuservi	144
removed \@sACRshort	145	removed \@sGlsuservi	144
removed \@sAcrshort	145	removed \@sglsuservi	143
removed \@sacrshort	144	removed \@sGLSuserv	143
removed \@sgls@link	113	removed \@sGlsuserv	143
removed \@sGLSdescplural	137	removed \@sglsuserv	143
removed \@sGlsdescplural	137	removed \@sGLS	128
removed \@sglsdescplural	137	removed \@sGls	127
removed \@sGLSdesc	137	removed \@sgls	126
removed \@sGlsdesc	136	removed \@thirdofthree (defined in kernel)	125
removed \@sglsdesc	136	removed sPGLS	269
removed \@sglsdisp	131	removed sPglS	267
removed \@sGLSfirstplural	135	removed spgls	266
removed \@sGlsfirstplural	135	removed sPGLSpl	269
removed \@sglsfirstplural	134	removed sPglSpl	268
removed \@sGLSfirst	133	removed spglspl	267
removed \@sGlsfirst	133	\ACRfull: removed \s@ACRfull	224
removed \@sglsfirst	133	switched to using \@gls@hyp@opt ..	224
removed \@sGLSname	136	\Acrfull: removed \@sAcrfull	223
removed \@sGlsname	135	switched to using \@gls@hyp@opt ..	223
removed \@sglsname	135	\acrfull: removed \@sacrfull	222
removed \@sGLSplural	134	switched to using \@gls@hyp@opt ..	222
removed \@sGlsplural	134	\ACRfullpl: removed \s@ACRfullpl ..	225
removed \@sglsplural	134	switched to using \@gls@hyp@opt ..	225
removed \@sGLSpl	130	\Acrfullpl: removed \s@Acrfullpl ..	224
removed \@sGlspl	129	switched to using \@gls@hyp@opt ..	224
removed \@sglspl	128	\acrfullpl: removed \s@acrfullpl ..	224
removed \@sGLSsymbolplural	139	switched to using \@gls@hyp@opt ..	224
removed \@sGlsymbolplural	139	\ACRlong: switched to using \@gls@hyp@opt	149
removed \@sglsymbolplural	138	\Acrlong: switched to using \@gls@hyp@opt	148
removed \@sGLSsymbol	138	\acrlong: switched to using \@gls@hyp@opt	148
removed \@sGlsymbol	138		
removed \@sglsymbol	138		
removed \@sGLStext	132		
removed \@sGlstext	133		

\ACRlongpl: switched to using \@gls@hyp@opt	151	\glsenablehyper: added \KV@glslink@hypertrue to definition	125
\Acrlongpl: switched to using \@gls@hyp@opt	150	\GLSfirst: switched to using \@gls@hyp@opt	133
\acrlongpl: switched to using \@gls@hyp@opt	150	\Glsfirst: switched to using \@gls@hyp@opt	133
\ACRshort: switched to using \@gls@hyp@opt	145	\glsfirst: switched to using \@gls@hyp@opt	133
\Acrshort: switched to using \@gls@hyp@opt	145	\GLSfirstplural: switched to using \@gls@hyp@opt	135
\acrshort: switched to using \@gls@hyp@opt	144	\Glsfirstplural: switched to using \@gls@hyp@opt	135
\ACRshorttpl: switched to using \@gls@hyp@opt	147	\glsfirstplural: switched to using \@gls@hyp@opt	134
\Acrshorttpl: switched to using \@gls@hyp@opt	146	\glsifhyper: deprecated	112
\acrshorttpl: switched to using \@gls@hyp@opt	146	\glslink: switched to using \@gls@hyp@opt	113
\forallacronyms: new	55	\glslinkcheckfirsthyperhook: new	114
\GLS: switched to using \@gls@hyp@opt	128	\glslinkvar: new	112
\Gls: switched to using \@gls@hyp@opt	127	\GLSname: switched to using \@gls@hyp@opt	136
\gls: switched to using \@gls@hyp@opt	126	\Glsname: switched to using \@gls@hyp@opt	135
\gls@defglossaryentry: added check for ignored glossary	86	\glsname: switched to using \@gls@hyp@opt	135
\gls@istfilebase: new	41	\GLSpl: switched to using \@gls@hyp@opt	130
\glsaddkey: removed \@sGLS@user@⟨key⟩	80	\Glspl: switched to using \@gls@hyp@opt	129
removed \@sGls@user@⟨key⟩	79	\glspl: switched to using \@gls@hyp@opt	128
removed \@sgls@user@⟨key⟩	79	\GLSplural: switched to using \@gls@hyp@opt	134
switched to using \@gls@hyp@opt	79, 80	\Glsplural: switched to using \@gls@hyp@opt	134
\GLSdesc: switched to using \@gls@hyp@opt	136	\glsplural: switched to using \@gls@hyp@opt	134
\Glsdesc: switched to using \@gls@hyp@opt	136	\glsspace: new	223
\glsdesc: switched to using \@gls@hyp@opt	136	\GLSsymbol: switched to using \@gls@hyp@opt	138
\GLSdescplural: switched to using \@gls@hyp@opt	137	\Glsymbol: switched to using \@gls@hyp@opt	138
\Glsdescplural: switched to using \@gls@hyp@opt	137	\glssymbol: switched to using \@gls@hyp@opt	138
\glsdescplural: switched to using \@gls@hyp@opt	137	\GLSsymbolplural: switched to using \@gls@hyp@opt	139
\glsdisablehyper: added \KV@glslink@hyperfalse to definition	125	\Glsymbolplural: switched to using \@gls@hyp@opt	139
\glsdisp: switched to using \@gls@hyp@opt	131		
\glsdohyperlink: new	124		
\glsdohypertarget: new	124		

\glssymbolplural: switched to using \@gls@hyp@opt	138	\newglossary: added starred version ..	63
\GLStext: switched to using \@gls@hyp@opt	132	\newignoredglossary: new	65
\Glstext: switched to using \@gls@hyp@opt	132	\ns@newglossary: added \@glotype@(<name>)@log	64
\glstext: switched to using \@gls@hyp@opt	132	new	63
\glstreenamefmt: new	314	\p@gls@hyp@opt: new	113
\GLSuseri: switched to using \@gls@hyp@opt	140	\PGLS: changed to use \@gls@hyp@opt	269
\Glsuseri: switched to using \@gls@hyp@opt	140	\PglS: changed to use \@gls@hyp@opt	267
\glsuseri: switched to using \@gls@hyp@opt	139	\pglS: changed to use \@gls@hyp@opt	266
\GLSuserii: switched to using \@gls@hyp@opt	141	\PGLSp1: changed to use \@gls@hyp@opt	269
\Glsuserii: switched to using \@gls@hyp@opt	140	\PglSp1: changed to use \@gls@hyp@opt	268
\glsuserii: switched to using \@gls@hyp@opt	140	\pglSp1: changed to use \@gls@hyp@opt	267
\GLSuseriii: switched to using \@gls@hyp@opt	141	\s@gls@hyp@opt: new	112
\Glsuseriii: switched to using \@gls@hyp@opt	141	\s@newglossary: new	63
\glsuseriii: switched to using \@gls@hyp@opt	141	automake: new	30
\GLSuseriv: switched to using \@gls@hyp@opt	142	4.09 (2014-08-12)	
\Glsuseriv: switched to using \@gls@hyp@opt	142	\glsaddkey: fixed bug in user commands	79
\glsuseriv: switched to using \@gls@hyp@opt	142	4.10 (2014-08-27)	
\GLSuserv: switched to using \@gls@hyp@opt	143	\@Gls@acentryname: new	153
\Glsuserv: switched to using \@gls@hyp@opt	143	\@Gls@entryname: new	153
\glsuserv: switched to using \@gls@hyp@opt	143	\@gls@glossary: Renamed \@glossary to \@gls@glossary	184
\GLSuservi: switched to using \@gls@hyp@opt	144	\glspercentchar: new	162
\Glsuservi: switched to using \@gls@hyp@opt	144	\glstildechar: new	162
\glsuservi: switched to using \@gls@hyp@opt	143	alttree: moved space after symbol	321, 322
\ifignoredglossary: new	66	4.11 (2014-09-01)	
altlongragged4col: fixed bug that displayed description instead of symbol	294	\@do@esc@wrglossary: added hook ..	189
		sanitize: none option	25
		\gls@wrglossary: renamed from \@wrglossary to \@gls@wrglossary	184
		\glsaddprotectedpagefmt: new	186
		\glsbackslash: new	162
		4.12 (2014-11-22)	
		\@gls@addpredefinedattributes: Added glsignore attribute	49
		\@gls@adjustmode: new	161
		\@gls@notranslatorhook: removed ...	25
		\@gls@toc: added \protect to \numberline	46
		\@gls@usetranslator: new	26
		\glsacrpluralsuffix: new	37
		\glsadd: added check for vertical mode	161
		\glsaddallunused: replaced @gobble with glsignore	161
		\glsifusedtranslatordict: new	26
		\glsignore: new	162

\glsupacrpluralsuffix: new	37	4.16 (2015-06-18)	
\ProvidesGlossariesLang: new	38	\glsaddstoragekey: new	77
\RequireGlossariesLang: new	38	4.16 (2015-07-08)	
4.13 (2015-02-03)		\@ACRlong: added \glspostlinkhook	364
\indexspace: new	276, 296, 314	\@ACRshort: added \glspostlinkhook	363
4.14 (2015-02-28)		\@Acrlong: added \glspostlinkhook	363
\@glslocalreset: new	94	\@Acrshort: added \glspostlinkhook	362
\@glslocalunset: new	94	\@GLS@: added \glspostlinkhook	128
\@glsreset: new	94	\@GLSpl: added \glspostlinkhook	131
\@glsunset: new	94	\@Gls@: added \glspostlinkhook	127
\@newglossaryentry@defcounters:		\@Glspl@: added \glspostlinkhook	130
new	96	\@acrlong: added \glspostlinkhook	363
\cGls: new	99	\@acrshort: added \glspostlinkhook	362
\cGls@: new	99	\@gls@: added \glspostlinkhook	127
\cGlspl@: new	100	\@gls@link: added	
\cgls: new	98	\glspostlinkhook	113
\cgls@: new	98	\@gls@field@link: added	
\cglspl: new	99, 100	\glspostlinkhook	132
\cglspl@: new	99	\@gls@link: moved definition of	
\gls@entry@count: new	98	\glsifhyperon outside of this	
\gls@increment@currcount: new	98	macro	115
\gls@local@increment@currcount:		\@glsdisp: added \glspostlinkhook	131
new	98	\@glspl@: added \glspostlinkhook	129
\gls@write@entrycounts: new	98	General: added \glspostlinkhook	145–151
\glslocalreset: new	94	\glsacspace: new	230
\glslocalunset: new	94	\glsadd: changed \@do@wrglossary to	
\glsreset: new	94	\@do@wrglossary	161
\glsunset: new	94	\glsfielddef: new	81
\@newglossaryentry@defcounters:		\glsfieldedef: new	81
new	90	\glsfieldfetch: new	82
\cGls: new	99	\glsfieldgdef: new	81
\cgls: new	98	\glsfieldxdef: new	80
\cGlsformat: new	99	\glsifhyperon: moved definition of	
\cglsformat: new	99	\glsifhyperon	114
\cGlspl: new	100	\glslinkpostsetkeys: new	114
\cglspl: new	99	\glspostlinkhook: new	113
\cGlsplformat: new	100	\glswriteentry: new	185
\cglsplformat: new	100	\ifglsfieldcseq: new	83
\gls@defdocnewglossaryentry: new	74	\ifglsfielddefeq: new	82
\glsenableentrycount: new	96	\ifglsfielddeq: new	82
\glslocalreset: switched to		long-sp-short: new	229
\@glslocalreset	93	\showglofield: new	262
\glslocalunset: switched to		4.18 (2015-09-09)	
\@glslocalunset	93	General: split mfirstuc into separate	
\glsreset: switched to \@glsreset	93	bundle	4
\glsunset: switched to \@glsunset	93	4.19 (2015-10-31)	
4.15 (2015-03-16)		\glstreenamebox: new	320
General: bug fix replaced \@glo@type		4.19 (2015-11-22)	
with \gls@type	151	\@gls@link@nocheckfirsthyper: new	132

\@gls@preglossaryhook: new	194	\glsfindwidesttoplevelname: new	320
\@printglossary: added		\glslistgroupheaderfmt: new	276
\@gls@preglossaryhook	196	\glslistnavigationitem: new	276
\do@glsglisablehyperinlist: new	114	\glstreegroupheaderfmt: new	314
\doifglossarynoexistsordo: new	58	\glstreenavigationfmt: new	314
\gls@gobbleopt: new	63	\ifglswrallowprimitivemods: new	187
\glsglsoifexistsordo: new	58	list: fixed missing space before	
4.20 (2015-11-30)		description	276
\@gls@link: added		long: fixed typo in \glossentrydesc	280
\@gls@setdefault@glslink@opts	115	super4col: fixed bug in \glossentry	305
added \glsglsonohyperlink when		4.23 (2016-04-30)	
hyperlink is suppressed	115	\glsglscurrentfieldvalue: new	62
\@gls@setdefault@glslink@opts:		\ifglshasfield: added	
new	114	\glsglscurrentfieldvalue	61
\glsgl@checkseeallowed@preambleonly:		altlongragged4col: check for	
new	69	nogroupskip changed	294
\glsglsonohyperlink: new	125	altsuperragged4col: check for	
4.21 (2016-01-24)		nogroupskip changed	313
\@printglossary: warn if no style has		long: check for nogroupskip changed	280
been set	195	long-booktabs: check for nogroupskip	
General: changed checkfirsthyper		changed	286
assignment	144–151	long3col: check for nogroupskip	
\glossarystyle: set default style if not		changed	282
already set	217	long3col-booktabs: check for	
\glsglLTpenaltycheck: new	289	nogroupskip changed	287
\glsglspatchLToutput: new	289	long4col: check for nogroupskip	
\glsglspenaltygroupskip: new	289	changed	283
altlong4col-booktabs: new	287	long4col-booktabs: check for	
altlongragged4col-booktabs: new	288	nogroupskip changed	287
long-booktabs: new	286	longragged: check for nogroupskip	
long3col-booktabs: new	286	changed	291
long4col-booktabs: new	287	longragged3col: check for nogroupskip	
longragged-booktabs: new	288	changed	292
longragged3col-booktabs: new	288	super: check for nogroupskip changed	302
\setglossarystyle: set default style if		super3col: check for nogroupskip	
not already set	217	changed	304
4.22 (2016-04-19)		super4col: check for nogroupskip	
\@do@esc@wrglossary: added check		changed	306
for \@arabic	188	superragged: check for nogroupskip	
added test to allow temporary primitive		changed	309
modifications and added arabic case	189	superragged3col: check for	
mcolaltrreespannav: new	301	nogroupskip changed	311
mcolindexspannav: new	297	4.24 (2016-05-27)	
mcoltreononamespannav: new	299	\@gls@extramakeindexopts: new	172
mcoltreespannav: new	298	\@gls@glossary: added check for debug	
\glsgl@arabicpage: new	186	mode	184
\glsgl@protected@pagefmts: added		\@gls@see@noindex: new	6
arabic to list	185	debug: new	5
\glsglentrytitlecase: new	156	seenoindex: new	6

<code>\glsnomakeindexwarning</code> : new	46	added <code>\TH</code> , <code>\dh</code> and <code>\DH</code>	24
<code>\GlsSetQuote</code> : new	170		4.31 (2017-08-10)
<code>\GlsSetWriteIstHook</code> : new	169	<code>nolist</code> : added check for “list” style	10
4.25 (2016-06-09)			4.31 (2017-09-10)
<code>\@gls@enablesavenonumberlist</code> : new	69	style: changed <code>\renewcommand</code> to <code>\def</code>	8
<code>\@gls@initnonumberlist</code> : new	69		4.32 (2017-08-24)
<code>\@gls@savenonumberlist</code> : new	69	<code>\@glsnavhypertarget</code> : new	271
4.26 (2016-10-12)		<code>\@glsshowtarget</code> : new	6
<code>\@glossary@default@style</code> : added		<code>\glsshowtarget</code> : new	6
check for classicthesis	8		4.33 (2017-09-20)
<code>mcolindex</code> : replaced <code>\@idxitem</code> with		<code>\@do@esc@wrglossary</code> : added	
<code>\glstreeitem</code>	296	<code>\gls@the</code> and <code>\gls@number</code>	189
<code>mcolindexspannav</code> : replaced <code>\@idxitem</code>		renamed from	
with <code>\glstreeitem</code>	297	<code>\@do@esc@wrglossary</code>	187
<code>\glstreechildpredesc</code> : new	315	<code>\@do@noesc@wrglossary</code> : new	187
<code>\glstreeitem</code> : new	314	<code>\@do@wrglossary</code> : changed to check	
<code>\glstreepredesc</code> : new	315	for eslocations	186
<code>\glstreesubitem</code> : new	315	<code>\@gls@missinglang@warn</code> : new	20
<code>\glstreesubsubitem</code> : new	315	<code>\GlsSetXdyFirstLetterAfterDigits</code> :	
4.28 (2017-01-07)		added starred version	163
<code>\glspatchtabularx</code> : new	92	<code>\GlsSetXdyNumberGroupOrder</code> : new	163
4.29 (2017-01-19)		<code>esclocations</code> : new	9
<code>\@gls@noidx@do</code> : current letter group			4.34 (2017-11-03)
assignment made global	205	<code>mcolalttreespannav</code> : removed spurious	
<code>\@print@noidx@glossary</code> : moved		space	301
definition of		<code>\glsshowtarget</code> : modified to check for	
<code>\@gls@currentlettergroup</code> outside		math mode and inner	6
of theglossary environment	203		4.35 (2017-11-14)
General: added check for		<code>\glsadd</code> : added <code>\@gls@setsort</code> (in case	
<code>\@glsxtr@doaccsupp</code>	344	of <code>sort=use</code>)	161
<code>\glsnavhyperlinkname</code> : new	271		4.36 (2018-03-07)
4.30 (2017-06-11)		<code>\@gls@glossary</code> : removed <code>\index</code>	184
<code>\@glo@autosee</code> : new	89		4.37 (2018-04-07)
<code>\@glo@autoseehook</code> : new	90	<code>\gls@begindocdefs</code> : new	75
<code>\@glo@check@sortallowed</code> : new	13		4.38 (2018-05-10)
<code>\@gls@noidx@do</code> : letter group		<code>\@gls@define@glossaryentrycounter</code> :	
assignment made global	205	added check for existence of	
<code>\@gls@setupsort@def</code> : added check for		<code>glossaryentry counter</code>	11
register	14	new	11
<code>\@gls@setupsort@none</code> : new	15	<code>\@gls@define@glossarysubentrycounter</code> :	
<code>\@xdycrossrefhook</code> : new	52	new	12
<code>\@xdylocationclassorder</code> : bug fix:		prepended <code>\currentglossary</code> . to	
changed <code>\edef</code> to <code>\def</code>	52	<code>\theHglossarysubentry</code> and	
<code>\glosortentrieswarning</code> : new	20	removed spurious eol space	12
<code>\gls@set@xr@key</code> : new	68	<code>\glsaccsupp</code> : added braces around	
<code>\gls@xr@key</code> : new	68	actual text argument	350
<code>\GlsAddXdyLocation</code> : bug fix: changed		<code>\glsentrycounterlabel</code> : bug fix: move	
<code>#1</code> to <code>#2</code>	52	conditional inside command	211
<code>\glsnoidxstripaccents</code> : added <code>\a</code>	23	<code>\GlsEntryCounterLabelPrefix</code> : new	208

<code>\glsentryitem</code> : bug fix: move conditional inside command	211	<code>kernelglossredefs</code> : new	32
<code>\glsrefentry</code> : bug fix: move conditional inside command	211	<code>\glossary</code> : added warning	32
<code>\glsresetsubentrycounter</code> : bug fix: move conditional inside command	.	210	<code>\gls@original@glossary</code> : new	31
<code>\glsstepentry</code> : bug fix: move conditional inside command	210	<code>\gls@original@makeglossary</code> : new	..	31
<code>\glsstepsubentry</code> : bug fix: move conditional inside command	210	<code>\makeglossaries</code> : removed redefinition of <code>\makeglossary</code>	177
<code>\glssubentrycounterlabel</code> : bug fix: move conditional inside command	.	211	<code>\makeglossary</code> : added warning	31
<code>\glssubentryitem</code> : bug fix: move conditional inside command	211	<code>nonumberlist</code> : changed <code>\val</code> and <code>\nr</code> to <code>\gls@nonumberlist@val</code> and <code>\gls@nonumberlist@nr</code>	69
<code>\showglongnameaccess</code> : bug fix: corrected field (was showing text access field)		380	<code>translate</code> : changed <code>\val</code> and <code>\nr</code> to <code>\gls@translate@val</code> and <code>\gls@translate@nr</code>	26
4.40 (2018-06-01)			<code>numberedsection</code> : changed <code>\val</code> and <code>\nr</code> to <code>\gls@numberedsection@val</code> and <code>\gls@numberedsection@nr</code>	8
<code>\listfile</code> : changed <code>\def</code> to <code>\providecommand</code>	182	4.42 (2019-01-06)		
<code>\makenoidxglossaries</code> : false	179	<code>\@gls@@automake@immediate</code> : new	..	176
4.41 (2018-07-23)			<code>\@gls@automake@immediate</code> : new	...	175
<code>\@gls@override@glossary</code> : new	32	<code>\gls@automake@nr</code> : new	30
General: changed <code>\val</code> and <code>\nr</code> to <code>\gls@numberedsection@val</code> and <code>\gls@numberedsection@nr</code>	208	<code>\glsfieldedef</code> : changed from <code>\edef</code> to <code>\protected@csedef</code>	81
debug: changed <code>\val</code> and <code>\nr</code> to <code>\gls@debug@val</code> and <code>\gls@debug@nr</code>	5	<code>\glsfieldxdef</code> : changed from <code>\edef</code> to <code>\protected@csxdef</code>	80
<code>seenoindex</code> : changed <code>\val</code> and <code>\nr</code> to <code>\gls@seenoindex@val</code> and <code>\gls@seenoindex@nr</code>	6	<code>\ifglsautomake</code> : now defined explicitly instead of through boolean key	30
			<code>noglossaryindex</code> : new	34
			<code>automake</code> : switch from boolean to choice		30
			4.42 (??)		
			<code>altnonlong4col-booktabs</code> : removed superfluous <code>\glspatchLToutput</code>	.	287

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	
\!	121
\"	23, 118–121, 123
\#	166
\%	162, 168, 169, 328, 329
\&	37, 160
\'	23
\.	11, 23
\=	23
\?	118, 120, 171
\@	75
\@@delimN	219
\@@do@@wrglossary	179, 187, 190
\@@do@esc@wrglossary	186
\@@do@noesc@wrglossary	186
\@@do@wrglossary	161, 185
\@@glo@assign@sortkey	180
\@@glo@list	56
\@@glo@sort	23
\@@glo@type	194
\@@glossarysec	7, 45
\@@glossaryseclabel	8, 45, 208
\@@glossarysecstar	8, 45, 208
\@@gls@checkactual	123
\@@gls@checkbar	122
\@@gls@checkescactual	120
\@@gls@checkescbar	121
\@@gls@checkesclevel	121
\@@gls@checkescquote	119, 120, 172
\@@gls@checklevel	122
\@@gls@checkquote	119, 170, 171
\@@gls@default@entryfmt	101, 110, 111
\@@gls@expand@field	21, 73, 77, 78, 241, 243, 245, 247, 249, 252, 254, 375–379
\@@gls@extramakeindexopts	170, 177
\@@gls@fixbraces	192
\@@gls@noexpand@field	21, 72
\@@gls@noidx@no@sanitizesort	23
\@@gls@noidx@nosanitizesort	181
\@@gls@nosanitizesort	22, 181
\@@gls@sanitizesort	22, 181
\@@gls@xdycheckbackslash	124
\@@gls@xdycheckquote	123
\@@gls@localreset	94, 96
\@@gls@localunset	94, 96
\@@gls@reset	94, 96
\@@gls@unset	94, 96
\@newglossaryentry@defcounters	96
\@this@glo@	56
\@ACRfull	224
\@ACRfullpl	225
\@ACRlong	149, 224
\@ACRlongpl	151, 225
\@ACRshort	145, 224
\@ACRshortpl	147, 225
\@Acrfull	223
\@Acrfullpl	224, 225
\@Acrlong	148, 223
\@Acrlongpl	150, 225
\@Acrshort	145
\@Acrshortpl	147
\@Alph	185, 188, 189
\@GLS	128
\@GLS@	128, 269
\@GLSdesc	136, 137
\@GLSdesc@	137
\@GLSdescplural	137
\@GLSdescplural@	137
\@GLSfirst	133
\@GLSfirst@	133
\@GLSfirstplural	135
\@GLSfirstplural@	135
\@GLSname	136
\@GLSname@	136

\@GLSpl	130	\@Glsuseri	140
\@GLSpl@	130, 270	\@Glsuseri@	140
\@GLSplural	134	\@Glsuserii	140
\@GLSplural@	134	\@Glsuserii@	140
\@GLSsymbol	138	\@Glsuseriii	141
\@GLSsymbol@	138	\@Glsuseriii@	141
\@GLSsymbolplural	139	\@Glsuseriv	142
\@GLSsymbolplural@	139	\@Glsuseriv@	142
\@GLStext	132	\@Glsuserv	143
\@GLStext@	132	\@Glsuserv@	143
\@GLSuseri	140	\@Glsuservi	144
\@GLSuseri@	140	\@Glsuservi@	144
\@GLSuserii	141	\@Mi	289
\@GLSuserii@	141	\@PGLS	269
\@GLSuseriii	141	\@PGLS@	269
\@GLSuseriii@	141, 142	\@PGLSpl	269
\@GLSuseriv	142	\@PGLSpl@	269
\@GLSuseriv@	142	\@Pgl	267
\@GLSuserv	143	\@Pgl@	267
\@GLSuserv@	143	\@Pglspl	268
\@GLSuservi	144	\@Pglspl@	268
\@GLSuservi@	144	\@Roman	186, 189
\@Gls	127	\@acrfull	222
\@Gls@	97, 99, 127, 268	\@acrfullpl	224
\@Gls@acrentryname	226	\@acrlong	148, 223
\@Gls@entry@field	79, 153–158	\@acrlongpl	150, 224
\@Gls@entryname	153, 226	\@acrshort	144, 223
\@GlsSetXdyFirstLetterAfterDigits	163	\@acrshortpl	146, 224, 225
\@GlsSetXdyNumberGroupOrder	163	\@addtoacronymlists	18
\@Glsdesc	136	\@after	18
\@Glsdesc@	136	\@afterheading	277, 332
\@Glsdescplural	137	\@alph	185, 188, 189
\@Glsdescplural@	137	\@arabic	186, 188, 189
\@Glsfirst	133	\@auxout	62,
\@Glsfirst@	133		64, 98, 176, 177, 179, 182, 193, 197, 272
\@Glsfirstplural	135	\@backslashchar	118, 124
\@Glsfirstplural@	135	\@before	18
\@Glsname	135	\@bsphack	184
\@Glsname@	135, 136	\@cGls	99
\@Glspl	129	\@cGls@	97, 99
\@Glspl@	97, 100, 129, 268, 269	\@cGlspl	100
\@Glsplural	134	\@cGlspl@	97, 100
\@Glsplural@	134	\@cclv	289, 290
\@Glsymbol	138	\@cgl	98
\@Glsymbol@	138	\@cgl@	97, 98
\@Glsymbolplural	139	\@cglpl	99
\@Glsymbolplural@	139	\@cglpl@	97, 99
\@Glstext	132, 133	\@chapter	36
\@Glstext@	133	\@classoptionslist	34

<code>\@closegls</code>	174	<code>\@glo@autoseehook</code>	90
<code>\@colht</code>	289	<code>\@glo@check@mkidxrangechar</code>	117, 190, 325, 326
<code>\@colroom</code>	289, 290	<code>\@glo@check@sortallowed</code>	13–16, 178, 181
<code>\@currentlabelname</code>	8, 208	<code>\@glo@childlist</code>	198
<code>\@curroptions</code>	34	<code>\@glo@counter</code>	68, 85, 88
<code>\@declaredoptions</code>	34	<code>\@glo@counterprefix</code>	182, 187, 189–191, 216, 217, 220
<code>\@delimN</code>	219	<code>\@glo@default@sorttype</code>	13, 180, 201, 202
<code>\@delimR</code>	219	<code>\@glo@defaultcounter</code>	88
<code>\@disable@onlypremakeg</code>	177	<code>\@glo@desc</code>	66, 83, 84, 86, 88
<code>\@disable@premakecs</code>	36	<code>\@glo@descaccess</code>	346–348
<code>\@disabled@gl saddxdycounters</code>	49	<code>\@glo@descplural</code>	67, 83, 84
<code>\@do@addcounter</code>	47	<code>\@glo@descpluralaccess</code>	346–348
<code>\@do@auxoutstuff</code>	197	<code>\@glo@do@sortentries</code>	198
<code>\@do@glossentry</code>	212, 344, 345	<code>\@glo@entry</code>	161, 162
<code>\@do@gl s@getcounterprefix</code>	187, 189	<code>\@glo@entryprefix</code>	264
<code>\@do@gl s@islistofacronyms</code>	18	<code>\@glo@entryprefixfirst</code>	264
<code>\@do@gl ssee</code>	89, 90	<code>\@glo@entryprefixfirstplural</code>	264, 265
<code>\@do@ifinlist</code>	47	<code>\@glo@entryprefixplural</code>	264
<code>\@do@newglossaryentry</code>	226, 240, 241, 243–245, 247, 249, 252, 254, 256, 375–379	<code>\@glo@esclabel</code>	91, 92
<code>\@do@seeglossary</code>	179, 192	<code>\@glo@etext</code>	102–104
<code>\@do@subglossentry</code>	214, 345	<code>\@glo@first</code>	67, 84, 87, 88, 252, 379
<code>\@do@wrglossary</code>	115	<code>\@glo@firstaccess</code>	345, 347, 348
<code>\@do@writeaux@info</code>	193	<code>\@glo@firstplural</code>	67, 84, 87, 88, 379
<code>\@ehc</code>	289	<code>\@glo@firstpluralaccess</code>	346–348
<code>\@empty</code>	11, 15, 18, 31, 34, 36, 47, 48, 52, 55, 56, 86, 91, 92, 116, 126–130, 144–151, 164, 167, 169, 174, 175, 183, 184, 191, 216, 241, 243, 245–248, 250, 251, 253, 254, 256, 326, 328, 330, 362–364	<code>\@glo@grabfirst</code>	204
<code>\@end@fixbraces</code>	192	<code>\@glo@label</code>	71, 78–90, 96, 159, 160, 264, 265, 320, 348
<code>\@endfortrue</code>	27, 59, 77, 272	<code>\@glo@list</code>	90
<code>\@esphack</code>	185	<code>\@glo@long</code>	60, 71, 85, 88
<code>\@expandtwoargs</code>	34	<code>\@glo@longaccess</code>	346–348
<code>\@firstofone</code>	23	<code>\@glo@longpl</code>	71, 85, 88, 241, 243, 245, 247, 249, 252, 254, 375
<code>\@firstofthree</code>	112, 125, 126, 129, 131, 144, 146, 148, 150, 362–364	<code>\@glo@longpluralaccess</code>	346–348
<code>\@firstoftwo</code>	26, 27, 75, 77, 112, 129, 130, 146, 147, 150, 151	<code>\@glo@name</code>	13, 66, 72, 84, 87, 88
<code>\@for</code>	27, 34, 36, 47, 49, 55, 56, 75, 77, 118, 164–166, 176–178, 185, 192, 198, 228, 241, 244, 246, 248, 250, 253, 255, 257, 272, 273, 326	<code>\@glo@no@assign@sortkey</code>	178
<code>\@glo@@desc</code>	88	<code>\@glo@nonumberlist</code>	69, 70
<code>\@glo@@symbol</code>	89	<code>\@glo@numfmt</code>	190, 326
<code>\@glo@access</code>	345, 347, 348, 350	<code>\@glo@parent</code>	15, 69, 85–87, 91, 92, 199
<code>\@glo@addchildren</code>	198, 203	<code>\@glo@plural</code>	67, 84, 87, 377
<code>\@glo@assign@sortkey</code>	178, 180, 209	<code>\@glo@pluralaccess</code>	346–348
<code>\@glo@autosee</code>	89	<code>\@glo@prefix</code>	9, 69, 85, 91, 92, 117, 190, 325, 326
		<code>\@glo@range</code>	190, 325, 326
		<code>\@glo@see</code>	68, 85, 89, 90
		<code>\@glo@seeautonumberlist</code>	9, 68
		<code>\@glo@short</code>	60, 71, 85, 88, 378
		<code>\@glo@shortaccess</code>	346–348, 375–378

<code>\@glo@shortpl</code>	71, 85, 88, 240, 241, 243, 245, 247, 249, 252, 254, 375, 378	<code>\@gls@link</code>	113
<code>\@glo@shortpluralaccess</code>	346–348	<code>\@gls@Hcounter</code>	116
<code>\@glo@sort</code> ...	13, 16, 22, 23, 67, 85, 87, 91, 92	<code>\@gls@ReturnAfterFi</code>	220
<code>\@glo@sortedinsert</code>	199	<code>\@gls@access@display</code>	350–352
<code>\@glo@sortentries</code>	201, 202	<code>\@gls@actualchar</code>	92, 120, 123, 168, 329
<code>\@glo@sorthandler@case</code>	202	<code>\@gls@addpredefinedattributes</code> .	164, 173
<code>\@glo@sorthandler@letter</code>	201	<code>\@gls@adjustmode</code>	161
<code>\@glo@sorthandler@nocase</code>	202	<code>\@gls@automake</code>	178
<code>\@glo@sorthandler@word</code>	201	<code>\@gls@automake@immediate</code>	176
<code>\@glo@sortinghandler</code>	198, 199	<code>\@gls@between</code>	273
<code>\@glo@sortinglist</code>	198, 199, 202	<code>\@gls@body</code>	153
<code>\@glo@sorttype</code>	180, 203, 204, 209	<code>\@gls@checkactual</code>	118, 171
<code>\@glo@storeentry</code>	13–15	<code>\@gls@checkbar</code>	118, 171
<code>\@glo@suffix</code>	117, 190, 326	<code>\@gls@checkedmkidx</code>	117–124, 170–172
<code>\@glo@symbol</code>	59, 68, 84, 89, 246, 251, 347	<code>\@gls@checkescactual</code>	118, 171
<code>\@glo@symbolaccess</code>	346–348, 378	<code>\@gls@checkescbar</code>	119, 171
<code>\@glo@symbolplural</code>	68, 84, 89	<code>\@gls@checkescquote</code>	118, 171, 172
<code>\@glo@symbolpluralaccess</code>	346–348	<code>\@gls@checklevel</code>	119, 171
<code>\@glo@text</code>	67, 84, 87, 89, 126–131, 152, 153, 247, 265, 377	<code>\@gls@checkmkidxchars</code> 91, 117, 171, 179, 189, 191, 325, 326
<code>\@glo@textaccess</code> ...	345, 347, 348, 375–378	<code>\@gls@checkquote</code>	118, 170, 171
<code>\@glo@thislabel</code>	90	<code>\@gls@classI</code>	165
<code>\@glo@thislettergrp</code>	204, 205	<code>\@gls@classII</code>	165
<code>\@glo@thisvalue</code>	60, 61	<code>\@gls@codepage</code>	197
<code>\@glo@tmp</code>	78, 79, 191	<code>\@gls@counter</code> 111, 115–117, 160, 161, 182, 190, 191, 326
<code>\@glo@type</code>	8, 15, 68, 84–86, 88, 89, 160, 161, 177, 183, 194–199, 202–204, 207, 208, 226, 241, 243–245, 247, 248, 250, 253–257, 271, 273	<code>\@gls@counterwithin</code>	11, 12
<code>\@glo@types</code> 55, 56, 63, 95, 161, 176–178, 262, 320	<code>\@gls@ctr</code>	47
<code>\@glo@useri</code>	70, 85, 88	<code>\@gls@currentlettergroup</code>	203, 205
<code>\@glo@userii</code>	70, 85, 88	<code>\@gls@debugfalse</code>	5
<code>\@glo@useriii</code>	70, 85, 88	<code>\@gls@debugtrue</code>	5
<code>\@glo@useriv</code>	70, 85, 88	<code>\@gls@declareoption</code> 9, 10, 16, 17, 20, 21, 26, 29, 30, 33, 34
<code>\@glo@userv</code>	70, 85, 88	<code>\@gls@default</code>	100, 101
<code>\@glo@uservi</code>	70, 85, 88	<code>\@gls@default@value</code> 59–61, 72, 73, 84, 85, 87, 88, 251, 264
<code>\@glodesc</code>	88	<code>\@gls@deffile</code>	74–76
<code>\@glolist@</code>	86	<code>\@gls@define@glossaryentrycounter</code> 12, 35, 208, 210
<code>\@gloname</code>	88	<code>\@gls@define@glossarysubentrycounter</code> 35, 209, 210
<code>\@glossary@default@style</code>	8, 10, 195, 217, 257	<code>\@gls@defsort</code>	13–15, 89
<code>\@glossaryentryfield</code>	91, 92	<code>\@gls@defsortcount</code>	13–15, 64
<code>\@glossarysection</code>	43	<code>\@gls@do@acronymsdef</code>	17, 35, 65
<code>\@glossarystyle</code>	195, 196, 207	<code>\@gls@do@indexdef</code>	34, 35, 65
<code>\@glossarysubentryfield</code>	91, 92	<code>\@gls@do@numbersdef</code>	33, 35, 65
<code>\@gls</code>	126	<code>\@gls@do@symbolsdef</code>	33, 65
<code>\@gls@</code>	97, 99, 126, 267, 268	<code>\@gls@do@symbolssdef</code>	35
<code>\@gls@@automake@immediate</code>	176	<code>\@gls@doautomake</code>	30, 176, 178

<code>\@gls@docheckquotedef</code>	170–172	<code>\@gls@levelchar</code>	92, 121, 122, 168, 329
<code>\@gls@docloadedfalse</code>	4	<code>\@gls@link</code>	113, 126–132, 144–151, 362–364
<code>\@gls@docloadedtrue</code>	4	<code>\@gls@link@checkfirsthyper</code>	126–131
<code>\@gls@dodedeflistparser</code>	178	<code>\@gls@link@label</code>	115, 242, 248
<code>\@gls@doentrycounterdef</code>	35	<code>\@gls@link@nocheckfirsthyper</code>	132, 144–151
<code>\@gls@doentrydef</code>	110, 111	<code>\@gls@link@opts</code>	115, 242, 248
<code>\@gls@dolast</code>	192, 193	<code>\@gls@list</code>	272, 273
<code>\@gls@donext</code>	192, 193	<code>\@gls@listsuffix</code>	47
<code>\@gls@donext@def</code>	159, 160	<code>\@gls@loadlist</code>	10, 257
<code>\@gls@dosubentrycounterdef</code>	35	<code>\@gls@loadlong</code>	10, 257
<code>\@gls@dotothiswrite</code>	174–176	<code>\@gls@loadsuper</code>	10, 257
<code>\@gls@elem</code>	272	<code>\@gls@loadtree</code>	10, 257
<code>\@gls@enablesavenonumberlist</code>	75	<code>\@gls@local@increment@currcount</code>	96
<code>\@gls@encapchar</code>	121, 122, 168, 190, 191, 326, 329	<code>\@gls@loclist</code>	180, 181, 204, 205
<code>\@gls@entry@count</code>	97, 98	<code>\@gls@map</code>	75, 77
<code>\@gls@entry@field</code>	78, 79, 96, 153–159, 348–350	<code>\@gls@missinglang@warn</code>	20, 39
<code>\@gls@escbsdq</code>	118, 169, 330	<code>\@gls@missingnumberlist</code>	88
<code>\@gls@expand@fields</code>	73, 74	<code>\@gls@noaccess</code>	350
<code>\@gls@expandonce</code>	73	<code>\@gls@noexpand@fields</code>	74
<code>\@gls@extramakeindexopts</code>	177	<code>\@gls@nohyperlist</code>	19, 66, 114
<code>\@gls@fetchfield</code>	61	<code>\@gls@noidx@do</code>	203
<code>\@gls@field@link</code>	79, 80, 132–144	<code>\@gls@noidx@getgrouptitle</code>	179
<code>\@gls@firsttok</code>	204	<code>\@gls@noidx@sanitizesort</code>	22, 181
<code>\@gls@fixbraces</code>	90	<code>\@gls@noidx@setsanitizesort</code>	24, 25, 181
<code>\@gls@forbidtexext</code>	64	<code>\@gls@noidx@loclist@finalsep</code>	180, 181
<code>\@gls@get@counterprefix</code>	191	<code>\@gls@noidx@loclist@prev</code>	180, 181, 206
<code>\@gls@getbody</code>	153	<code>\@gls@noidx@loclist@sep</code>	180, 206
<code>\@gls@getcounterprefix</code>	187, 189	<code>\@gls@noref@warn</code>	179, 204
<code>\@gls@getgrouptitle</code>	179, 215, 273	<code>\@gls@numberlink</code>	219, 220
<code>\@gls@glossary</code>	183, 184	<code>\@gls@numbersdef</code>	33
<code>\@gls@gobbleopt</code>	63	<code>\@gls@numlist@lastsep</code>	159, 160
<code>\@gls@grptitle</code>	215, 271, 273	<code>\@gls@numlist@nextsep</code>	159, 160
<code>\@gls@hyp@opt</code>	79, 80, 98–100, 113, 126–151, 222–225, 266–269	<code>\@gls@numlist@sep</code>	159, 160
<code>\@gls@hyp@opt@cs</code>	112, 113	<code>\@gls@old@chapter</code>	36
<code>\@gls@hypergroup</code>	272	<code>\@gls@oldnewglossaryentryposthook</code>	347
<code>\@gls@ifinlist</code>	47	<code>\@gls@oldnewglossaryentryprehook</code>	347
<code>\@gls@ifnotmeasuring</code>	92	<code>\@gls@onlypremakeg</code>	36
<code>\@gls@igtype</code>	66	<code>\@gls@order</code>	174, 175
<code>\@gls@increment@currcount</code>	96	<code>\@gls@org@LT@output</code>	289
<code>\@gls@indexdef</code>	34	<code>\@gls@org@glsnoidxdisplayloc</code>	181
<code>\@gls@initnonumberlist</code>	69, 85	<code>\@gls@org@glssseeformat</code>	181
<code>\@gls@islistofacronyms</code>	18	<code>\@gls@override@glossary</code>	32, 33
<code>\@gls@keylist</code>	374	<code>\@gls@patchtabularx</code>	92, 93
<code>\@gls@keymap</code>	70, 75, 77, 78, 264, 347	<code>\@gls@preglossaryhook</code>	196
<code>\@gls@label</code>	179, 182, 187, 190	<code>\@gls@prevlevel</code>	300, 301, 320–323, 338, 339
<code>\@gls@langmod</code>	174, 175	<code>\@gls@provide@newglossary</code>	64
		<code>\@gls@quotechar</code>	119–123, 168, 170, 172, 329
		<code>\@gls@reference</code>	179, 182
		<code>\@gls@removespaces</code>	220

<code>\@gls@renewglossary</code>	173	<code>\@glsHlocref</code>	187, 189
<code>\@gls@replacementtext</code>	350	<code>\@glsacronymlists</code>	18, 19, 56, 226, 228, 241, 243–248, 250, 253–257, 262
<code>\@gls@rest</code>	153	<code>\@glsaddkey</code>	78
<code>\@gls@restoreat</code>	75	<code>\@glsaddstoragekey</code>	77
<code>\@gls@roman</code>	50, 51, 326, 327	<code>\@glsaddxdyattribute</code>	48, 49
<code>\@gls@sanitized@tmp</code>	118	<code>\@glsdefaultsort</code>	13
<code>\@gls@sanitizedesc</code>	28	<code>\@glsdesc</code>	136
<code>\@gls@sanitizesort</code>	13	<code>\@glsdesc@</code>	136
<code>\@gls@sanitizesymbol</code>	28	<code>\@glsdescplural</code>	137
<code>\@gls@saveentrycounter</code>	115, 161	<code>\@glsdescplural@</code>	137
<code>\@gls@savenonumberlist</code>	69, 70	<code>\@glsdisp</code>	131
<code>\@gls@see@noindex</code>	7, 69	<code>\@glsentry</code>	95, 98
<code>\@gls@setacrstyle</code>	28, 35	<code>\@glsentrytitlecase</code>	156
<code>\@gls@setcounter</code>	64	<code>\@glsfirst</code>	133
<code>\@gls@setdefault@glslink@opts</code>	115	<code>\@glsfirst@</code>	133
<code>\@gls@setsort</code>	13–16, 115, 161	<code>\@glsfirstletter</code>	163
<code>\@gls@setupshortcuts</code>	35	<code>\@glsfirstplural</code>	134
<code>\@gls@sort</code>	205	<code>\@glsfirstplural@</code>	134
<code>\@gls@sort@A</code>	200	<code>\@glshypernumber</code>	219
<code>\@gls@sort@B</code>	200	<code>\@glsisacronymlistfalse</code>	19
<code>\@gls@startswithexpandonce</code>	73	<code>\@glsisacronymlisttrue</code>	19
<code>\@gls@storenonumberlist</code>	69, 70, 88	<code>\@glslink</code>	115, 125, 160, 271
<code>\@gls@symbolsdef</code>	33	<code>\@glslocalreset</code>	93, 96
<code>\@gls@this</code>	185	<code>\@glslocalunset</code>	94, 96
<code>\@gls@thisHloc</code>	191	<code>\@glslocref</code>	182, 187, 189, 190, 325, 326
<code>\@gls@thisfield</code>	61	<code>\@glsminrange</code>	164, 165, 327
<code>\@gls@thislabel</code>	58, 59, 192, 202	<code>\@glsname</code>	135
<code>\@gls@thislist</code>	159, 160	<code>\@glsname@</code>	135
<code>\@gls@thisloc</code>	191	<code>\@glsnavhypertarget</code>	271
<code>\@gls@thisval</code>	77	<code>\@glsnextpages</code>	195
<code>\@gls@title</code>	43	<code>\@glsnodesc</code>	84, 86, 88
<code>\@gls@tmp</code>	15, 39, 52, 73, 118, 184, 272, 273	<code>\@glsnoname</code>	84, 87, 88
<code>\@gls@tmpb</code>	119–124, 170, 172	<code>\@glsnonextpages</code>	195
<code>\@gls@toc</code>	45	<code>\@glsnumberformat</code>	112, 115, 160, 161, 182, 190, 325, 326
<code>\@gls@type</code>	176, 178, 228, 241, 244, 246, 248, 250, 253, 255, 257, 320	<code>\@glsopenfile</code>	173, 183
<code>\@gls@updatechecked</code>	118, 119, 171	<code>\@glsorder</code>	176, 177
<code>\@gls@usetranslator</code>	26, 27, 38	<code>\@glspl</code>	128
<code>\@gls@value</code>	72, 73, 156	<code>\@glspl@</code>	97, 99, 128, 267–269
<code>\@gls@warnonglossdefined</code>	21, 194	<code>\@glsplural</code>	134
<code>\@gls@warnontheGLOSSdefined</code>	21, 212	<code>\@glsplural@</code>	134
<code>\@gls@write@entrycounts</code>	97	<code>\@glsreset</code>	93, 96
<code>\@gls@writedef</code>	75	<code>\@glssee</code>	90, 192
<code>\@gls@writeisthook</code>	168, 169	<code>\@glsshowtarget</code>	5, 6, 124, 125
<code>\@gls@xdy@locationlist</code>	165	<code>\@glsymbol</code>	138
<code>\@gls@xdycheckbackslash</code>	118	<code>\@glsymbol@</code>	138
<code>\@gls@xdycheckquote</code>	118	<code>\@glsymbolplural</code>	138
<code>\@gls@xref</code>	191	<code>\@glsymbolplural@</code>	138, 139
<code>\@gls@Alphacompositor</code>	41, 51, 327		

<code>\@glstarget</code>	125, 212, 272	<code>\@onelevel@sanitize</code>	22, 50, 75, 92, 118, 163, 167, 191, 193, 204, 327, 328
<code>\@glstext</code>	132	<code>\@onlypreamble</code>	64, 74, 84, 98, 101, 178, 181
<code>\@glstext@</code>	132	<code>\@onlypremakeg</code>	40–42, 48, 49, 52, 64, 169
<code>\@glunset</code>	93, 96	<code>\@org@glossaryentrynumbers</code>	195, 196
<code>\@gluseri</code>	139	<code>\@org@gl@assign@descplural</code> 241, 249, 250, 252, 254, 375, 378, 379
<code>\@gluseri@</code>	139	<code>\@org@gl@assign@firstpl</code> 241, 243, 245, 247, 249, 252, 254, 375–379
<code>\@gluserii</code>	140	<code>\@org@gl@assign@plural</code> 241, 243, 245, 247, 249, 252, 254, 375–379
<code>\@gluserii@</code>	140	<code>\@org@gl@assign@symbolplural</code> 241, 243, 245, 247, 252, 254, 376, 377, 379
<code>\@gluseriii</code>	141	<code>\@org@gl@numberformat</code>	159
<code>\@gluseriii@</code>	141	<code>\@org@newglossaryentryprehook</code>	83
<code>\@gluseriv</code>	142	<code>\@outputpage</code>	289, 290
<code>\@gluseriv@</code>	142	<code>\@p@glossarysection</code>	43
<code>\@gluserv</code>	142, 143	<code>\@p@gl</code>	266
<code>\@gluserv@</code>	143	<code>\@p@gl@</code>	266
<code>\@gluservi</code>	143	<code>\@p@gl@spl</code>	267
<code>\@gluservi@</code>	143	<code>\@p@gl@spl@</code>	267
<code>\@glswidestname</code>	320–322, 338	<code>\@plus</code>	276, 296, 314
<code>\@glswritefiles</code>	31	<code>\@print@glossary</code>	194
<code>\@glxtr@doaccsupp</code>	344	<code>\@print@noidx@glossary</code>	194
<code>\@gobble</code>	5, 13–15, 75, 76, 93, 118, 162, 166, 179, 324, 328, 329	<code>\@print@gloss@setsort</code>	178, 180, 195
<code>\@idxitem</code>	314	<code>\@print@glossary</code>	194
<code>\@ifclassloaded</code>	4, 11, 44	<code>\@roman</code>	50, 326
<code>\@ifnextchar</code>	64, 112	<code>\@secondofthree</code> 113, 125, 127, 129, 145, 147, 149, 151, 362
<code>\@ifpackageloaded</code> 4, 8, 26, 27, 38, 55, 92, 159, 170, 344	<code>\@secondoftwo</code>	23, 26, 27, 38, 75, 77, 125– 128, 131, 144–146, 148, 149, 362–364, 382
<code>\@ifstar</code>	63, 77, 78, 112, 163, 221	<code>\@set@glo@numformat</code>	190, 326
<code>\@ifundefined</code>	38, 272, 279, 290, 301, 308, 321, 322, 338, 352	<code>\@sglsaddkey</code>	78
<code>\@ignored@glossaries</code>	65, 66	<code>\@sglsaddstoragekey</code>	77
<code>\@input@</code>	196	<code>\@thirdofthree</code> 113, 128, 130, 146, 147, 149, 151, 362
<code>\@istfilename</code>	176, 177	<code>\@this@attr</code>	166
<code>\@makecol</code>	289, 290	<code>\@this@childlabel</code>	198, 199
<code>\@makeglossary</code>	177	<code>\@this@counter</code>	49
<code>\@minus</code>	276, 296, 314	<code>\@this@ctr</code>	166
<code>\@mkboth</code>	44	<code>\@this@key</code>	77
<code>\@newglossary</code>	62, 64	<code>\@this@label</code>	198
<code>\@newglossaryentry@defcounters</code>	89, 96	<code>\@this@label</code>	198
<code>\@newglossaryentry@posthook</code> 78, 79, 89, 264, 347	<code>\@this@scs</code>	36
<code>\@newglossaryentry@prehook</code> 78, 83, 85, 264, 347	<code>\@tmp</code>	50, 327
<code>\@nil</code>	18, 90, 117–119, 153, 171, 190, 192, 204, 205, 219, 220, 325, 326	<code>\@use@option</code>	34
<code>\@nnil</code>	18, 192	<code>\@warn@nomakeglossaries</code>	176, 197
<code>\@no@makeglossaries</code>	178, 180	<code>\@wrglossary@pageformat</code>	186
<code>\@no@post@desc</code>	331	<code>\@wrglossary@numberhook</code>	186, 189
<code>\@nopostdesc</code>	195	<code>\@xdy@main@language</code>	29, 174, 175, 197

<code>\booltrue</code>	31	<code>\CurrentOption</code>	34, 264, 344
<code>\bottomrule</code>	286, 287	<code>\CurrentTrackedLanguage</code>	39, 382, 383
<code>\box</code>	289	<code>\CurrentTrackedTag</code>	39, 382, 383
C			
<code>\c</code>	23	<code>\CustomAcronymFields</code>	256
<code>\c@equation</code>	116	<code>\CustomNewAcronymDef</code>	257
<code>\c@glossaryentry</code>	11	D	
<code>\c@glossarysubentry</code>	12	<code>\d</code>	23
<code>\c@page</code>	185, 186, 189	datatool package	200
<code>\catcode</code>	75	<code>\day</code>	164, 168, 326, 329
<code>\cGls</code>	99	<code>\DeclareAcronymList</code>	17, 19, 226, 228, 241, 243, 245, 248, 250, 253, 255, 257
<code>\cglS</code>	98	<code>\DeclareListParser</code>	178
<code>\cGlsformat</code>	97	<code>\DeclareOption</code>	9, 264, 344
<code>\cglSformat</code>	97	<code>\DeclareOptionX</code>	9
<code>\cGlspl</code>	100	<code>\DeclareRobustCommand</code>	40, 192, 193, 250, 350–352
<code>\cglSpl</code>	99	<code>\def</code>	8, 9, 11–16, 18, 22–24, 29, 31–33, 35–37, 39, 40, 43, 47, 50–55, 58, 59, 62–71, 73, 81, 83, 85–90, 92, 93, 97–101, 110–112, 114–124, 126– 151, 159–161, 164, 168, 170–172, 174, 175, 177, 179, 180, 183, 186, 187, 189– 192, 194, 195, 198, 202–204, 206, 207, 209, 216–221, 223–226, 241, 243, 245– 247, 249–254, 256, 264, 266–270, 273– 275, 300, 301, 320–323, 325, 326, 329, 331, 338, 339, 344–347, 361–364, 375–379
<code>\cGlsplformat</code>	97	<code>\def@glS@xdycheckbackslash</code>	124
<code>\cglSplformat</code>	97	<code>\DefaultNewAcronymDef</code>	242
<code>\char</code>	216	<code>\defglSentryfmt</code>	64, 66, 110, 111, 228, 240, 242, 244, 246, 248, 251, 253, 256
classicthesis package	8	<code>\define@boolkey</code>	7, 9, 11, 12, 17, 24, 27–29, 31, 112, 209
<code>\cleardoublepage</code>	46	<code>\define@choicekey</code>	5–8, 13, 25, 26, 29, 30, 32, 69, 208, 209
<code>\clearpage</code>	45, 46	<code>\define@key</code>	8, 12, 19, 25, 29, 66–71, 77, 78, 111, 112, 160, 207, 209, 264, 345, 346
<code>\closeout</code>	75, 168, 169, 174, 183	<code>\DefineAcronymSynonyms</code>	35, 240
<code>\compatglossarystyle</code>	331–343	<code>\delimN</code>	167, 178, 206, 219, 328
<code>\compatibleglossentry</code>	214	<code>\delimR</code>	167, 219, 328
<code>\compatiblesubglossentry</code>	214	<code>\DescriptionDUANewAcronymDef</code>	246
<code>\copy</code>	289, 290	<code>\DescriptionFootnoteNewAcronymDef</code>	244
<code>\count@</code>	204	<code>\descriptionname</code>	39, 281–284, 286, 287, 291–293, 295, 303–308, 310–314
<code>\csdef</code>	21, 77–80, 89, 90, 96, 198, 199, 218, 228, 330	<code>\DescriptionNewAcronymDef</code>	248
<code>\csedef</code>	98, 186	<code>\DH</code>	24
<code>\csgdef</code>	43, 62, 65, 96, 97, 193, 207	<code>\dh</code>	24
<code>\cslet</code>	70, 83, 84, 90, 202	<code>\dimen@</code>	230, 289, 320
<code>\csname</code>	13–16, 34, 37, 39, 45, 48, 50, 51, 54, 56, 58, 63–65, 72, 73, 77–81, 83, 86–92, 94, 110, 111, 115–117, 126– 131, 144–152, 159, 161, 165, 166, 171– 174, 179, 182–186, 190, 191, 194–197, 199, 207, 212, 214, 217, 218, 221, 258– 265, 272, 273, 320–322, 324–326, 338, 344, 345, 348, 349, 352, 362–364, 380, 381		
<code>\csshow</code>	262		
<code>\csuse</code>	39, 43, 62, 72, 73, 79, 80, 110, 111, 174–176, 199, 201, 203, 205, 206, 208, 209, 217, 228, 265, 331–343		
<code>\csxdef</code>	88, 98		
<code>\currentglossary</code>	11, 12, 43, 195		
<code>\currentglssubentry</code>	12, 210		

<code>\disable@keys</code>	34	<code>\end@getprefix</code>	191
<code>\do</code>	27, 34, 36, 47, 49, 55, 56, 75, 77, 118, 159, 164–166, 176–178, 185, 192, 198, 228, 241, 244, 246, 248, 250, 253, 255, 257, 272, 273, 326	<code>\end@gl@islistofacronyms</code>	18
<code>\do@gl@storeentry</code>	13–15, 89	<code>\EndAccSupp</code>	350
<code>\do@gl@link@checkfirsthyper</code> 113, 115, 126–132, 144–151, 362, 363	<code>\endcsname</code>	13–16, 34, 37, 39, 45, 48, 50, 51, 54, 56, 58, 63–65, 72, 73, 77–81, 83, 86–92, 94, 110, 111, 115–117, 126– 131, 144–152, 159, 161, 165, 166, 171– 174, 179, 182–186, 190, 191, 194–197, 199, 207, 212, 214, 217, 218, 221, 258– 265, 272, 273, 320–322, 324–326, 338, 344, 345, 348, 349, 352, 362–364, 380, 381
<code>\do@gl@xdycheckbackslash</code>	118	<code>\endfoot</code> .	280–282, 284, 286, 287, 291–293, 295
<code>\do@gl@disablehyperinlist</code>	115	<code>\endgroup</code>	5, 185, 189, 220
<code>\do@gl@shaschildren</code>	58, 59	<code>\endhead</code> .	280–282, 284, 286, 287, 291–293, 295
<code>doc package</code>	4, 5, 16	<code>\endtheglossary</code>	5
<code>\doifglossarynoexistsordo</code>	63	<code>\entryname</code>	39, 281–284, 286, 287, 291–293, 295, 303–308, 310–314
<code>\dtl@ifsingle</code>	216	<code>\equal</code>	25, 35, 45, 116, 177, 216, 272
<code>\dtl@insertinto</code>	199	<code>equation (counter)</code>	116
<code>\dtl@sortresult</code>	200	<code>etoolbox package</code>	4
<code>\dtlcompare</code>	200	<code>\expandafter</code>	13–16, 22, 23, 34, 36, 39, 48, 50, 51, 53–56, 58, 63–66, 72, 73, 75–81, 83, 86, 87, 89–94, 110, 111, 114, 116–124, 153, 160, 162, 166, 170–173, 183–185, 187, 189, 190, 192, 195, 199, 204, 205, 212–214, 218, 220, 221, 242, 248, 258–265, 272, 273, 320, 324–326, 328, 329, 344, 345, 348, 350, 374, 380, 381
<code>\dtlicompare</code>	200	<code>\expandonce</code>	72, 73, 118, 171, 172, 186, 200, 212, 214, 226, 240, 241, 243, 245, 247, 249, 252, 254, 344, 345
<code>\DTLifinlist</code>	66, 114		
<code>\DTLifint</code>	216		
<code>\dtlletterindexcompare</code>	200		
<code>\DTLsubstituteall</code>	118		
<code>\dtlwordindexcompare</code>	200		
<code>\DUANewAcronymDef</code>	255		
E			
<code>\eappto</code>	65, 66, 90, 186		
<code>\edef</code> ..	15, 18, 36, 39, 47, 48, 50–54, 56, 58, 63–66, 72, 73, 75, 77, 80–85, 90, 91, 110, 111, 114–116, 118–124, 159, 161, 162, 168, 170–172, 174, 175, 178, 179, 183, 187, 189–191, 193, 197–200, 204, 210, 216, 220, 221, 240, 243, 244, 247, 249, 252, 254, 271, 324, 325, 327, 329, 374–378		
<code>\egroup</code>	23, 84, 160, 196, 199		
<code>\else</code>	6, 11, 12, 15–18, 20, 22, 24, 25, 30–32, 34–36, 40, 41, 44, 46–50, 52–56, 69, 71, 86, 87, 90–92, 97, 114– 124, 126–131, 153, 163, 164, 167–170, 172–175, 183–187, 189–192, 195, 204, 209, 211, 216, 219, 220, 230, 244, 245, 248, 250, 251, 253, 255, 257, 272, 277, 280, 282, 283, 286, 287, 289, 291, 293, 294, 302, 304, 306, 309, 311, 313, 316, 317, 319, 321, 322, 325–331, 336–339, 350		
<code>\emph</code>	192, 221		
<code>\empty</code>	220, 344		
<code>\end</code>	166, 203, 276, 280–285, 288–314, 328		
<code>\end@doifinlist</code>	47		
		F	
		<code>\fi</code>	5–8, 11–18, 20, 22, 24, 25, 27, 30–36, 40, 41, 44, 46–56, 65, 69, 72, 86–92, 97, 114–117, 119–124, 127–131, 153, 161, 163, 164, 167, 169– 173, 175, 176, 178, 183–187, 189–193, 195, 197, 204, 208–211, 217, 219, 220, 230, 240, 241, 244, 245, 248, 250, 251, 253, 255, 257, 263, 272, 277, 280, 282, 283, 286, 287, 289–291, 293, 294, 302, 304, 306, 309, 311, 313, 316, 317, 319– 322, 324–328, 330, 331, 336–339, 344, 350
		file types	
		<code>.aux</code>	197
		<code>.glo</code>	91
		<code>.ist</code>	162, 172, 173
		<code>.toc</code>	46
		<code>.xdy</code>	41

glo	263	glossary package	1, 32, 221
\firstacronymfont	109, 110, 229–231, 236, 242, 246, 248, 251, 361, 365–367, 371, 372	glossary styles:	
\footnote	236, 242, 372	altlist	277, 278, 332
\FootnoteNewAcronymDef	250	altlistgroup	278, 332
\forallglossaries	56, 183, 194, 320	altlisthypergroup	278, 332
\forallglstentries	95, 98, 161, 162	altlong4col	284, 285, 293, 334
\ForEachTrackedDialect	39, 382, 383	altlong4col-booktabs	287, 289
\forglstentries	56, 58, 90, 202, 320	altlong4colborder	285, 334
\foralllistcsloop	198, 203	altlong4colheader	285, 287, 334
\foralllistloop	181, 206	altlong4colheaderborder	285, 334
G			
garamondx package	222	altlongragged4col	288, 293–295, 335
\gdef	15, 48, 63, 81, 86, 87, 184, 209, 272	altlongragged4col-booktabs	288
\Genacrformat	109, 227, 229, 230, 236, 361, 365, 366, 372	altlongragged4colborder	295, 335
\genacrformat	109, 227, 229, 230, 236, 360, 361, 365, 366, 371	altlongragged4colheader	294, 335
\GenericAcronymFields	226, 229, 230, 232, 233, 235, 236, 238, 365–368, 370, 371, 374	altlongragged4colheaderborder	295, 336
\Genplacrformat	108, 227, 229, 230, 236, 360, 366, 372	altsuper4col	307, 312, 343
\genplacrformat	108, 110, 227, 229, 230, 236, 360, 366, 372	altsuper4colborder	307, 343
\glo@desc	331	altsuper4colheader	307, 343
\glo@do@compare	200, 201	altsuper4colheaderborder	307, 343
\glo@grabfirst	205	altsuperragged4col	312, 313, 341
\glo@label	58, 59, 90	altsuperragged4colborder	313, 341
\glo@list	90	altsuperragged4colheader	313, 341
\glo@name	212, 213	altsuperragged4colheaderborder	313, 341
\glo@parent	59	alttree	300, 315, 320, 322, 338
\glo@type	90	alttreegroup	323, 339
\glo@value	75, 76	alttreehypergroup	323, 339
\global	14–16, 72, 74, 83, 84, 89, 94, 184, 196, 204, 205, 210, 289, 290	index	8, 296, 314–317, 336
\glolinkprefix	115, 160, 212	indexgroup	316, 336
\glosortentrieswarning	20, 198	indexhypergroup	316, 336
glossaries package	32, 34, 54, 164, 257, 264, 276, 324, 344	inline	331
glossaries-accsupp package	90, 344	list	8, 10, 276–278, 331
glossaries-extra package	75, 166, 344	listdotted	278, 279, 332
\GlossariesWarning	5, 7, 20, 21, 24, 25, 31, 32, 43, 46, 58, 61, 69, 71, 98–100, 110, 112, 159, 174–176, 179– 182, 185, 191, 195, 196, 214, 217, 324, 344	listgroup	277, 331
\GlossariesWarningNoLine	5, 6, 20, 177, 179, 183, 197, 272	listhypergroup	277, 332
\glossary	31–33, 325, 326	long	280, 281, 286, 290, 332, 334
		long-booktabs	286, 288
		long3col	281, 282, 286, 333
		long3col-booktabs	286, 288
		long3colborder	282, 333
		long3colheader	282, 286, 333
		long3colheaderborder	282, 333
		long4col	283, 284, 287, 333
		long4col-booktabs	287
		long4colborder	284, 334
		long4colheader	283, 287, 334
		long4colheaderborder	284, 334
		longborder	280, 333

longheader [280](#), [286](#), [333](#)
longheaderborder [281](#), [333](#)
longragged [288](#), [290](#)–[292](#)
longragged-booktabs [288](#)
longragged3col [288](#), [292](#), [293](#), [335](#)
longragged3col-booktabs [288](#)
longragged3colborder [293](#), [335](#)
longragged3colheader [293](#), [335](#)
longragged3colheaderborder . [293](#), [335](#)
longraggedborder [291](#), [334](#)
longraggedheader [291](#), [335](#)
longraggedheaderborder [292](#), [335](#)
mcolalmtree [300](#), [340](#)
mcolalmtreegroup [300](#), [340](#)
mcolalmtreehypergroup ... [300](#), [301](#), [340](#)
mcolindex [296](#), [339](#)
mcolindexgroup [296](#), [339](#)
mcolindexhypergroup [296](#), [297](#), [339](#)
mcoltree [297](#), [339](#)
mcoltreegroup [339](#)
mcoltreehypergroup [298](#), [339](#)
mcoltreenoname [299](#), [340](#)
mcoltreenonamegroup [299](#), [340](#)
mcoltreenonamehypergroup ... [299](#), [340](#)
sublistdotted [332](#)
super [302](#), [303](#), [310](#), [342](#)
super3col [303](#)–[305](#), [342](#)
super3colborder [304](#), [342](#)
super3colheader [304](#), [342](#)
super3colheaderborder [305](#), [342](#)
super4col [305](#)–[307](#), [343](#)
super4colborder [306](#), [343](#)
super4colheader [306](#), [343](#)
super4colheaderborder [306](#), [343](#)
superborder [302](#), [342](#)
superheader [303](#), [342](#)
superheaderborder [303](#), [342](#)
superragged [308](#), [310](#), [340](#)
superragged3col [310](#)–[312](#), [341](#)
superragged3colborder [311](#), [341](#)
superragged3colheader [311](#), [341](#)
superragged3colheaderborder [312](#), [341](#)
superraggedborder [309](#), [340](#)
superraggedheader [310](#), [340](#)
superraggedheaderborder [310](#), [341](#)
tree [297](#), [317](#), [318](#), [320](#), [336](#)
treegroup [298](#), [318](#), [337](#)
treehypergroup [318](#), [337](#)
treenoname [298](#), [315](#), [318](#), [319](#), [337](#)
treenonamegroup [319](#), [338](#)
treenonamehypergroup [319](#), [338](#)
glossary-hypernav package [162](#)
glossary-list package [8](#), [10](#), [276](#)
glossary-long package ... [10](#), [279](#), [293](#), [301](#), [302](#)
glossary-longragged package [290](#)
glossary-mcols package [295](#)
glossary-super package .. [10](#), [279](#), [301](#), [308](#), [312](#)
glossary-superragged package [308](#)
glossary-tree package [10](#), [314](#)
\glossaryentry [190](#), [191](#), [326](#)
glossaryentry (counter) [11](#), [12](#), [210](#), [211](#)
\glossaryentryfield
..... [212](#), [331](#)–[338](#), [340](#)–[343](#), [365](#)
\glossaryentrynumbers
..... [9](#), [167](#), [195](#), [196](#), [205](#), [209](#), [210](#), [328](#)
\glossaryheader
..... [166](#), [203](#), [274](#), [276](#)–[278](#), [280](#)–
[284](#), [286](#), [287](#), [290](#)–[296](#), [298](#)–[300](#), [302](#),
[303](#), [305](#), [309](#), [310](#), [312](#), [315](#)–[320](#), [323](#), [328](#)
\glossarymark [44](#)
\glossaryname [16](#), [39](#)
\glossarypostamble [166](#), [203](#), [328](#)
\glossarypreamble [166](#), [203](#), [328](#)
\glossarysection [166](#), [203](#), [328](#)
glossarysubentry (counter) [12](#), [210](#), [211](#)
\glossarysubentryfield
..... [214](#), [331](#)–[338](#), [340](#)–[343](#), [365](#)
\glossarytitle .. [166](#), [194](#), [195](#), [203](#), [207](#), [328](#)
\glossarytoctitle [8](#), [16](#), [17](#),
[33](#), [37](#), [39](#), [44](#), [166](#), [195](#), [203](#), [207](#), [208](#), [328](#)
\glossentry [90](#), [196](#), [205](#), [214](#),
[274](#), [276](#)–[281](#), [283](#), [291](#), [292](#), [294](#), [302](#),
[304](#), [305](#), [309](#), [310](#), [312](#), [315](#), [317](#), [318](#), [321](#)
\Glossentrydesc [364](#)
\glossentrydesc
. [274](#)–[281](#), [283](#), [291](#), [292](#), [294](#), [302](#), [304](#),
[305](#), [309](#)–[312](#), [315](#)–[317](#), [319](#), [321](#), [322](#), [364](#)
\glossentryname
. [274](#)–[281](#), [283](#), [291](#), [292](#), [294](#), [302](#), [304](#),
[305](#), [309](#), [310](#), [312](#), [315](#)–[318](#), [321](#), [322](#), [364](#)
\Glossentrysymbol [365](#)
\glossentrysymbol [274](#), [275](#), [283](#),
[294](#), [305](#), [312](#), [315](#)–[317](#), [319](#), [321](#), [322](#), [365](#)
\Gls [99](#), [221](#), [240](#)
\gls [32](#), [99](#), [179](#), [211](#), [221](#), [240](#)
\gls@Alphpage [185](#), [189](#)
\gls@alphpage [185](#), [189](#)
\gls@arabicpage [185](#), [189](#)

<code>\gls@assign@desc</code>	83, 88	<code>\gls@org@glossaryentryfield</code>	196
<code>\gls@assign@descplural</code>	241, 249, 250, 252, 254, 375, 378, 379	<code>\gls@org@glossarysubentryfield</code>	196
<code>\gls@assign@field</code>	74, 78, 79, 83, 85, 87–89, 264, 265	<code>\gls@org@insert</code>	246, 248, 251
<code>\gls@assign@firstpl</code>	241, 243, 245, 247, 249, 252, 254, 375–379	<code>\gls@orgAlpha</code>	188, 189
<code>\gls@assign@plural</code>	241, 243, 245, 247, 249, 252, 254, 375–379	<code>\gls@orgalph</code>	188, 189
<code>\gls@assign@symbolplural</code>	241, 243, 245, 247, 252, 254, 376, 377, 379	<code>\gls@orgarabic</code>	188, 189
<code>\gls@automake@nr</code>	30, 176	<code>\gls@orgnumber</code>	188, 189
<code>\gls@automake@val</code>	30	<code>\gls@orgRoman</code>	189
<code>\gls@begindocdefs</code>	75	<code>\gls@orgromannumeral</code>	188, 189
<code>\gls@checkisacronymlist</code>	114	<code>\gls@orgthe</code>	188, 189
<code>\gls@checkseallowed</code>	68, 74, 177, 179	<code>\gls@original@glossary</code>	33
<code>\gls@checkseallowed@preambleonly</code>	74	<code>\gls@original@makeglossary</code>	33
<code>\gls@codepage</code>	54, 55, 174, 175, 197	<code>\gls@protected@pagefmts</code>	118, 185, 186
<code>\gls@debug@nr</code>	5, 32	<code>\gls@Romanpage</code>	185, 189
<code>\gls@debug@val</code>	5, 32	<code>\gls@romanpage</code>	185, 189
<code>\gls@defdocnewglossaryentry</code>	75, 96	<code>\gls@save@numberlist</code>	9
<code>\gls@defglossaryentry</code>	74, 75, 84	<code>\gls@seenoinde@nr</code>	7
<code>\gls@disablepagerefexpansion</code>	185, 189	<code>\gls@seenoinde@val</code>	7
<code>\gls@do@addxdyattribute</code>	49	<code>\gls@set@x@key</code>	68
<code>\gls@doclearpage</code>	46	<code>\gls@suffixF</code>	42, 167, 169, 328, 330
<code>\gls@dosubst</code>	118	<code>\gls@suffixFF</code>	42, 167, 169, 328, 330
<code>\gls@dotoc@title</code>	195, 196, 207	<code>\gls@text</code>	109, 110
<code>\gls@end@sanitizesort</code>	22, 23	<code>\gls@the</code>	189
<code>\gls@endcheck</code>	73	<code>\gls@thissty</code>	27
<code>\gls@glossary</code>	32, 33, 190, 191	<code>\gls@tmp</code>	183, 250, 251
<code>\gls@gobbleopt</code>	64	<code>\gls@tmplen</code>	124, 320–322, 338, 339
<code>\gls@grplabel</code>	271	<code>\gls@tr@set@acronym@toctitle</code>	17
<code>\gls@hypergroup@prerun</code>	272	<code>\gls@tr@set@main@toctitle</code>	16
<code>\gls@ifnotmeasuring</code>	93, 94	<code>\gls@tr@set@numbers@toctitle</code>	33
<code>\gls@inlinepostchild</code>	274, 275, 331	<code>\gls@tr@set@symbols@toctitle</code>	33
<code>\gls@inlinesep</code>	274, 331	<code>\gls@translate@nr</code>	26
<code>\gls@inlinesubsep</code>	274, 275, 331	<code>\gls@translate@val</code>	26
<code>\gls@islistofacronyms</code>	18	<code>\gls@wrglossary</code>	184
<code>\gls@istfilebase</code>	40, 174, 175	<code>\gls@xdystring</code>	117, 118
<code>\gls@label</code>	221	<code>\gls@xindy@glsnumbersfalse</code>	30
<code>\gls@level</code>	86, 87, 205	<code>\gls@xindy@glsnumberstrue</code>	29
<code>\gls@noid@glossary</code>	179	<code>\gls@xr@key</code>	6, 7, 68, 69
<code>\gls@nonumberlist@nr</code>	69	<code>\gls@accsupp</code>	350
<code>\gls@nonumberlist@val</code>	69	<code>\gls@acronymtrue</code>	17
<code>\gls@nosetquote</code>	84, 168, 170, 172	<code>\gls@acrpluralsuffix</code>	37, 222, 231, 235, 236, 238, 242
<code>\gls@number</code>	189	<code>\gls@acrshortcutsfalse</code>	35
<code>\gls@numberedsection@nr</code>	8, 208	<code>\gls@acrshortcutstrue</code>	35
<code>\gls@numberedsection@val</code>	8, 208	<code>\gls@sacspace</code>	229, 230, 232
<code>\gls@numberpage</code>	185, 189	<code>\gls@sadd</code>	32, 161, 162
		<code>\gls@sadd options</code>	
		counter	160
		format	160, 218

<code>\glsaddall options</code>		<code>\glsentrycounterlabel</code>	211
types	160, 161	<code>\GlsEntryCounterLabelPrefix</code> ...	210, 211
<code>\GlsAddXdyAttribute</code>	48, 49, 324, 325	<code>\glsentrycountertrue</code>	12
<code>\GlsAddXdyCounters</code>	48, 49, 65	<code>\glsentrycurrcount</code>	96, 98
<code>\glsautomakefalse</code>	30, 176	<code>\Glsentrydesc</code>	136, 213, 364
<code>\glsautomaketrue</code>	30	<code>\glsentrydesc</code>	
<code>\glsautoprefix</code>	8, 208	103–105, 136, 137, 213, 354–356, 364
<code>\glscapscase</code>	101, 103, 105–	<code>\glsentrydescaccess</code>	351
109, 126–131, 144–151, 234, 246, 251,		<code>\Glsentrydescplural</code>	137
352, 354, 356, 357, 359, 360, 362–364, 369		<code>\glsentrydescplural</code> 101–103, 137, 352–354	
<code>\glsclearpage</code>	45	<code>\glsentrydescpluralaccess</code>	351
<code>\glsfclosebrace</code>	52, 167, 168, 328, 329	<code>\Glsentryfirst</code> ...	99, 104, 107, 133, 355, 358
<code>\glscompositor</code>	41, 51, 169, 327, 330	<code>\glsentryfirst</code>	
<code>\glscounter</code>	19, 35, 47, 64, 88, 115, 324	.	99, 103, 104, 106, 107, 133, 354, 355, 358
<code>\glscurrententrylabel</code>	193, 196	<code>\glsentryfirstaccess</code>	351
<code>\glscurrentfieldvalue</code>	61	<code>\Glsentryfirstplural</code>	
<code>\glscustomtext</code> 101, 105, 107, 109, 126–131,		100, 102, 106, 135, 353, 357
144–151, 234, 235, 242, 246, 248, 249,		<code>\glsentryfirstplural</code>	100, 102,
251, 352, 356, 358, 359, 361–364, 369, 370		103, 105, 106, 134, 135, 352, 354, 356, 357	
<code>\GlsDeclareNoHyperList</code>	19	<code>\glsentryfirstpluralaccess</code>	351
<code>\glsdefaulttype</code>	16,	<code>\glsentryfmt</code>	64, 66
43, 54, 56, 62, 63, 85, 101, 110, 183, 194		<code>\Glsentryfull</code>	227, 235, 237, 371, 373
<code>\glsdefmain</code>	16, 65	<code>\glsentryfull</code>	227, 235, 237, 370, 373
<code>\glsdescriptionaccessdisplay</code>		<code>\Glsentryfullpl</code> ...	227, 235, 237, 371, 373
....	354–356, 364, 365	<code>\glsentryfullpl</code> ...	227, 235, 237, 371, 373
<code>\glsdescriptionpluralaccessdisplay</code>		<code>\glsentryitem</code>	274, 276–281, 283,
....	352–354	291, 292, 294, 302, 304, 305, 309, 310,	
<code>\glsdescwidth</code>	280–	312, 315, 317, 318, 321, 331–338, 340–343	
282, 284, 285, 288–295, 302–305, 307–314		<code>\Glsentrylong</code>	99, 149, 153,
<code>\glsdetoklabel</code> 57–61, 70, 75, 80–84, 90, 94,		158, 229, 234, 235, 361, 363, 366, 369–371	
96–98, 114, 152, 153, 159–161, 179–181,		<code>\glsentrylong</code> ...	99, 109, 148, 149, 153,
187, 190, 196, 199, 200, 204, 205, 207,		158, 229, 230, 232–238, 249, 361, 363–374	
210–212, 258–262, 320, 344, 345, 380, 381		<code>\glsentrylongaccess</code>	351
<code>\glsdisplay</code>	101, 111	<code>\Glsentrylongpl</code>	100, 151,
<code>\glsdisplayfirst</code>	101, 110	159, 229, 230, 234, 235, 361, 366, 369–371	
<code>\glsdisplaynumberlist</code>	180	<code>\glsentrylongpl</code>	
<code>\glsdohyperlink</code>	125	100, 110, 150, 151, 159, 229, 230,
<code>\glsdohypertarget</code>	125	234–237, 249, 256, 361, 366, 367, 369–373	
<code>\glsdoifexists</code>		<code>\glsentrylongpluralaccess</code>	352
..	58–60, 80–83, 93, 94, 126–132, 144–	<code>\Glsentryname</code>	136, 213, 364
151, 159, 161, 180, 181, 266–270, 361–365		<code>\glsentryname</code>	135, 136, 320, 364
<code>\glsdoifexistsordo</code>	113, 152	<code>\glsentrynumberlist</code>	159, 180
<code>\glsdoifexistsorwarn</code>	207, 212, 213	<code>\Glsentryplural</code> ...	102, 105, 134, 353, 357
<code>\glsdoifnoexists</code>	74, 83	<code>\glsentryplural</code>	
<code>\glsdonohyperlink</code>	115, 125	101–103, 105, 106, 134, 352, 353, 356, 357	
<code>\glsdosanitizesort</code>	13	<code>\glsentrypluralaccess</code>	351
<code>\glsentryaccess</code>	350	<code>\Glsentryprefix</code>	268
<code>\glsentrycounter</code>	217, 220	<code>\glsentryprefix</code>	266, 269
<code>\glsentrycounterfalse</code>	12	<code>\Glsentryprefixfirst</code>	268

<code>\glsentryprefixfirst</code>	267, 269	<code>\glsfirstpluralaccessdisplay</code>	357
<code>\Glsentryprefixfirstplural</code>	269	<code>\glsgenacfmt</code>	229, 230, 236, 365, 366, 371
<code>\glsentryprefixfirstplural</code>	267, 270	<code>\glsgenentryfmt</code>	
<code>\Glsentryprefixplural</code>	268	229, 230, 235, 236, 240, 242, 244, 246, 248, 251, 253, 256, 365, 366, 370, 371
<code>\glsentryprefixplural</code>	267, 270	<code>\glsgetgroupitle</code>	
<code>\glsentryprevcount</code>	96, 97	273, 277, 278, 296–301, 316–319, 323
<code>\Glsentryshort</code>	108, 145, 153, 230, 236, 237, 360–362, 366, 372, 373	<code>\gls glossarymark</code>	44
<code>\glsentryshort</code>	108, 109, 144, 146, 153, 158, 227, 229, 230, 232, 233, 235–238, 359–362, 365–368, 370–374	<code>\gls groupheading</code> 168, 205, 274, 276–278, 280, 281, 283, 291, 292, 294, 296–303, 305, 309, 310, 312, 315–320, 322, 323, 329	
<code>\glsentryshortaccess</code>	351	<code>\gls groupskip</code> ...	167, 205, 275, 277, 280, 282, 283, 286, 287, 291–294, 302, 304, 306, 309, 311, 313, 316, 317, 319, 322, 328
<code>\Glsentryshorttpl</code>		<code>\gls hyperfirstfalse</code>	236, 371
<code>\glsentryshorttpl</code>		<code>\gls hyperfirsttrue</code>	28
....	107, 108, 110, 146, 147, 159, 229, 230, 235–237, 256, 359, 361, 366, 370–373	<code>\gls hyperlink</code>	193
<code>\glsentryshortpluralaccess</code>	351	<code>\gls hypernavsep</code>	273
<code>\Glsentrysymbol</code>	138, 213, 365	<code>\gls hypernumber</code>	42, 220, 221
<code>\glsentrysymbol</code>	103– 105, 138, 213, 242, 246, 251, 354–356, 365	<code>\gls ifhyperon</code>	112
<code>\glsentrysymbolaccess</code>	351	<code>\gls ifListOfAcronyms</code>	18, 19
<code>\Glsentrysymbolplural</code>	139	<code>\gls ifplural</code>	101, 105, 107, 108, 126–131, 144–151, 234, 242, 246, 249, 251, 352, 356, 359, 360, 362–364, 369
<code>\glsentrysymbolplural</code>		<code>\gls ifusetranslator</code>	26, 27, 38, 39, 382
....	101–103, 139, 242, 246, 251, 352–354	<code>\gls indexonlyfirstfalse</code>	27
<code>\glsentrysymbolpluralaccess</code>	351	<code>\gls inlinedescformat</code>	274, 331
<code>\Glsentrytext</code>	104, 106, 133, 354, 358	<code>\gls inlinedopostchild</code>	274, 331
<code>\glsentrytext</code>	103, 104, 106, 107, 132, 160, 193, 354, 355, 357, 358	<code>\gls inlineemptydescformat</code>	274, 331
<code>\glsentrytextaccess</code>	350	<code>\gls inlinenameformat</code>	274, 331
<code>\glsentrytype</code>	85	<code>\gls inlineparentchildseparator</code> 274, 331	
<code>\Glsentryuseri</code>	140	<code>\gls inlinepostchild</code>	274, 331
<code>\glsentryuseri</code>	139, 140	<code>\gls inlineseparator</code>	274, 331
<code>\Glsentryuserii</code>	140	<code>\gls inlinesubdescformat</code>	275, 331
<code>\glsentryuserii</code>	140, 141	<code>\gls inlinesubnameformat</code>	274, 331
<code>\Glsentryuseriii</code>	141	<code>\gls inlinesubseparator</code>	275, 331
<code>\glsentryuseriii</code>	141, 142	<code>\gls insert</code>	
<code>\Glsentryuseriv</code>	142	101–109, 126–131, 144–151, 234, 235, 242, 246, 248, 249, 251, 352–364, 369, 370
<code>\glsentryuseriv</code>	142	<code>\gls keylisttok</code>	
<code>\Glsentryuserv</code>	143	226, 241, 243–250, 252–257, 374–379
<code>\glsentryuserv</code>	143	<code>\gls label</code>	84, 101–109, 114, 115, 145–151, 229, 230, 234, 236, 242, 246, 248, 249, 251, 352–361, 365, 366, 369–371
<code>\Glsentryuservi</code>	144	<code>\gls labeltok</code>	226, 240, 242–244, 246–250, 252–257, 375–378
<code>\glsentryuservi</code>	143, 144	<code>\gls link</code>	227, 235–237, 242, 370, 372
<code>\glscsclocationsfalse</code>	179	<code>\gls link options</code>	
<code>\glscsclocationstrue</code>	9	counter	111, 126, 263
<code>\glsfieldfetch</code>	156		
<code>\glsfirstaccessdisplay</code>	354, 355, 358		
<code>\glsfirstpluralaccessdisplay</code>			
....	352, 353, 356, 357		

format	112, 126, 218	<code>\glsorg@endtheglossary</code>	5
hyper	112, 114, 126	<code>\glsorg@PrintChanges</code>	5
local	112	<code>\glsorg@theglossary</code>	5
<code>\glslinkcheckfirsthyperhook</code>	114	<code>\glspagelistwidth</code>	281, 282, 284, 285, 288, 289, 292–295, 303–305, 307, 308, 310–314
<code>\glslinkpostsetkeys</code>	115	<code>\glspatchLTOutput</code>	286–288
<code>\glslinkvar</code>	112, 113	<code>\glspenaltygroupskip</code>	286, 287
<code>\glslistdottedwidth</code>	278, 279, 332	<code>\glspercentchar</code>	75, 76, 166, 168
<code>\glslistgroupheaderfmt</code>	277, 278	<code>\Glspl</code>	100, 240
<code>\glslistnavigationitem</code>	277, 278	<code>\glspl</code>	99, 240
<code>\glslocalreset</code>	95	<code>\glspluralaccessdisplay</code>	352, 353, 356, 357
<code>\glslocalunset</code>	95, 126–131	<code>\glspluralsuffix</code>	...
<code>\glslongaccessdisplay</code>	361, 363–375	...	37, 87, 88, 229–231, 366, 367, 371–373
<code>\glslongkey</code>	379	<code>\glspostdescription</code>	...
<code>\glslongpluralaccessdisplay</code>	40, 275–278, 280, 291, 302, 309, 315– 317, 319, 321, 322, 331–334, 336–340, 342
...	361, 366, 367, 369–373, 375	<code>\glspostinline</code>	274
<code>\glslongpluralkey</code>	379	<code>\glspostlinkhook</code>	...
<code>\glslongtok</code>	113, 127–132, 145–151, 362–364
...	226, 227, 229, 230, 235, 236, 240– 250, 252–257, 365, 366, 370, 371, 374–379	<code>\glsprestandardsort</code>	13
<code>\glsLTpenaltycheck</code>	289, 290	<code>\glsreset</code>	95
<code>\glsncols</code>	296–301	<code>\glsresetentrycounter</code>	210
<code>\glsnameaccessdisplay</code>	364, 365	<code>\glsresetentrylist</code>	167, 203, 209, 328
<code>\glsnamefont</code>	212–214, 344, 345, 364	<code>\glsresetsubentrycounter</code>	211, 274, 331
<code>\glsnavhyperlink</code>	273	<code>\glssanitizeortfalse</code>	25
<code>\glsnavhyperlinkname</code>	271, 272	<code>\glssanitizeorttrue</code>	24, 25
<code>\glsnavhypertarget</code>	...	<code>\glsavenumberlistfalse</code>	9
...	277, 278, 297–301, 317–319, 323	<code>\glsavewritesfalse</code>	31
<code>\glsnavigation</code>	...	<code>\glsseeformat</code>	166, 179, 181, 328
...	277, 278, 296–301, 316, 318, 319, 323	<code>\glsseeitem</code>	192
<code>\glsnextpages</code>	9, 69, 195	<code>\glsseeitemformat</code>	193
<code>\glsnogroupskipfalse</code>	11	<code>\glsseeelastsep</code>	193
<code>\glsnoidxdisplayloc</code>	181, 182	<code>\glsseeelist</code>	192
<code>\glsnoidxdisplayloclisthandler</code>	181	<code>\glsseeesep</code>	193
<code>\glsnoidxloclist</code>	180, 205	<code>\glssetexpandfield</code>	21, 24, 25
<code>\glsnoidxloclisthandler</code>	206	<code>\glssetnoexpandfield</code>	21, 22, 24, 25
<code>\glsnoidxnumberlistloophandler</code>	181	<code>\GlsSetQuote</code>	84, 168
<code>\glsnoidxstripaccents</code>	23	<code>\glssettoctitle</code>	39, 195
<code>\glsnomakeindexwarning</code>	170	<code>\glsshortaccessdisplay</code>	...
<code>\glsnonextpages</code>	69, 195	...	359–362, 365–368, 370–375
<code>\glsnopostdotfalse</code>	11	<code>\glsshortkey</code>	379
<code>\glsnoxindywarning</code>	41, 48–50, 52–55, 163, 164	<code>\glsshortpluralaccessdisplay</code>	...
<code>\glsnumberformat</code>	159	...	359, 361, 366, 370–373, 375
<code>\glsnumberlistloop</code>	181	<code>\glsshortpluralkey</code>	379
<code>\glsnumbersgroupname</code>	33, 40, 216	<code>\glsshorttok</code>	226, 240–250, 252–257, 375–379
<code>\glsnumlistlastsep</code>	160, 180	<code>\glsshowtarget</code>	6
<code>\glsnumlistparser</code>	160, 178	<code>\glsstnumberfmt</code>	14, 15
<code>\glsnumlistsep</code>	160, 180	<code>\glsSPACE</code>	223
<code>\glsopenbrace</code>	52, 167, 168, 328, 329	<code>\glsstentry</code>	211
<code>\glsorder</code>	29, 174, 175, 177, 201		

<code>\ifglossaryexists</code>	.. 43, 54, 58, 173–175, 195	<code>\ifignoredglossary</code> 86, 89, 185
<code>\ifgls@sanitize@description</code> 24	<code>\ifin@</code> 34
<code>\ifgls@sanitize@name</code> 24	<code>\ifinlistcs</code> 202, 207
<code>\ifgls@sanitize@symbol</code> 24	<code>\ifinner</code> 6
<code>\ifgls@xindy@glslnumbers</code> 55	<code>\ifKV@glslink@hyper</code> 114, 115
<code>\ifglsacrdescription</code> 255	<code>\ifKV@glslink@local</code> 126–131
<code>\ifglsacrdua</code> 244, 250, 253, 255	<code>\ifmeasuring@</code> 92
<code>\ifglsacrfootnote</code> 114, 255	<code>\ifmmode</code> 6
<code>\ifglsacrone</code> 17	<code>\ifnum</code> 14, 30, 97, 176, 204, 289, 317, 319, 321, 337, 338
<code>\ifglsacrshortcuts</code> 35, 240	<code>\ifstreempty</code> 331
<code>\ifglsacrsmalldcaps</code>	.. 244, 245, 248, 250, 253	<code>\ifstrequal</code> 216
<code>\ifglsacrsmaller</code> 244, 245, 248, 250	<code>\ifthenelse</code>	25, 35, 45, 116, 177, 216, 255, 272
<code>\ifglsautomake</code> 30, 178	<code>\IfTrackedLanguage</code> 170
<code>\ifglsdescsuppressed</code> 274	<code>\IfTrackedLanguageFileExists</code>	39, 382, 383
<code>\ifglsentrycounter</code> 11, 12, 35, 210, 211	<code>\iftrue</code> 89, 94
<code>\ifglsentryexists</code> 57, 58, 74, 75, 84, 86	<code>\ifundef</code>	.. 11, 12, 63, 74, 85, 164, 168, 176, 177
<code>\ifglsesclocations</code> 186	<code>\ifvmode</code> 161
<code>\ifglsashchildren</code> 274, 331	<code>\ifvoid</code> 289
<code>\ifglsashdesc</code> 274	<code>\ifx</code> 11, 13, 15, 18, 34, 36, 47, 48, 50, 52, 55, 56, 86–89, 91, 92, 116, 119–124, 153, 164, 167, 169, 170, 172, 183, 186, 187, 189, 191, 192, 195, 216, 217, 219, 220, 241, 243, 245, 247, 250, 251, 253, 254, 256, 257, 326–328, 330, 331, 336–339, 344, 350
<code>\ifglsashlong</code> 99, 100, 153, 229, 230, 234, 236, 248, 365, 366, 369, 371	<code>\immediate</code> 74–76, 98, 173, 175, 183, 197
<code>\ifglsashparent</code> 199, 205, 320	<code>\in@</code> 34
<code>\ifglsashprefix</code> 268	<code>\indexname</code> 34
<code>\ifglsashprefixfirst</code> 268	<code>\indexspace</code>	.. 277, 296–301, 316–319, 322, 323
<code>\ifglsashprefixfirstplural</code> 268	<code>\input</code> 38, 101
<code>\ifglsashprefixplural</code> 268	<code>\inputencodingname</code> 29
<code>\ifglsashsymbol</code> 242, 246, 251, 315–317, 319, 321, 322	<code>\InputIfFileExists</code> 75
<code>\ifglsashyperfirst</code> 114	<code>\istfilename</code> 40, 164, 168, 174, 175, 177, 326, 329
<code>\ifglsindexonlyfirst</code> 185	<code>\item</code>	276–279, 296, 297, 315, 316, 331, 332, 336
<code>\ifglsnogroupskip</code> 277, 280, 282, 283, 286, 287, 291, 292, 294, 302, 304, 306, 309, 311, 313, 316, 317, 319, 322	J	
<code>\ifglsnonnumberlist</code> 209	<code>\jobname</code> 41, 74, 75, 164, 168, 173–176, 196, 326, 329
<code>\ifglsnopostdot</code> 11	K	
<code>\ifglsnumberline</code> 46	<code>\key@ifundefined</code> 77, 78
<code>\ifglssanitize@sort</code> 22, 24, 25	<code>\KV@glslink@hyperfalse</code> 112, 114, 125
<code>\ifgls@savenumberlist</code> 71, 178, 193	<code>\KV@glslink@hypertrue</code> 112, 125
<code>\ifgls@savewrites</code> 31, 173, 184	L	
<code>\ifgls@ssubentrycounter</code> 12, 35, 210, 211	<code>\L</code> 23
<code>\ifgls@stoc</code> 46	<code>\l</code> 23
<code>\ifgls@stranslate</code> 38	<code>\label</code> 8, 208, 210
<code>\ifgls@sucmark</code> 44		
<code>\ifgls@sused</code> 98, 101–107, 114, 162, 185, 242, 246, 248, 251, 266–270, 352–359		
<code>\ifgls@swrallowprimitivemods</code> 189		
<code>\ifgls@xindy</code> 40, 41, 47–50, 52–55, 65, 90, 91, 118, 163, 164, 170, 174, 175, 190, 191, 197, 324–326		

entrycounter	208	ngerman package	170
first	67, 87, 126, 133, 155, 247, 253, 345	\noalign	289
firstaccess	349, 351	\nobreak	277, 290, 332
firstplural	67, 134, 155, 346	\noexpand	18, 36, 47, 49, 75, 88–90, 110, 111, 116–118, 124, 160, 170–172, 174, 175, 178, 186, 187, 189, 193, 197, 200, 212, 214, 221, 226, 240, 241, 243–245, 247, 249, 252, 254, 256, 324, 344, 345, 375–379
firstpluralaccess	349, 351	\nohyperpage	219
format	164	\noindent	214, 297–299, 301, 318, 319
long	107, 158, 346	\noist	329, 330
longaccess	350, 351	\nopostdesc	34, 40, 83, 195, 331
longplural	158, 346	\normalbaselineskip	289
longpluralaccess	350, 352	\ns@ACRfull	224
name	66, 67, 71, 74, 84, 135, 152, 193, 345	\ns@Acrfull	223
nonumberlist	69	\ns@acrfull	222
parent	69, 74	\ns@ACRfullpl	225
plural	67, 87, 133, 346	\ns@Acrfullpl	224
pluralaccess	349, 351	\ns@acrfullpl	224
prefix	264	\ns@ACRlong	149
prefixfirst	264	\ns@Acrlong	148
prefixfirstplural	265	\ns@acrlong	148
prefixplural	265	\ns@ACRlongpl	151
see	6, 9, 68, 74, 177, 179	\ns@Acrlongpl	150
short	107, 158, 346	\ns@acrlongpl	150
shortaccess	349, 351	\ns@ACRshort	145
shortplural	158, 346	\ns@Acrshort	145
shortpluralaccess	349, 351	\ns@acrshort	144
sort	67, 156, 184, 215	\ns@ACRshortpl	147
symbol	66, 67, 138, 243–245, 247, 252, 283, 305, 345–347	\ns@Acrshortpl	146, 147
symbolaccess	349, 351	\ns@acrshortpl	146
symbolplural	138, 346	\ns@newglossary	63
symbolpluralaccess	349, 351	\null	118–124, 170–172, 196
text	67, 126, 132, 154, 243, 247, 345	\number	14, 75, 87, 98, 186, 188, 189, 214, 345
textaccess	349, 350	\numberline	46
type	16, 68, 100, 156	\numexpr	98
user1	139, 156, 347		
user2	140, 156		
user3	141, 157		
user4	142, 157		
user5	142, 157		
user6	143, 157, 347		
\newglossarystyle	273, 276–288, 290–313, 315–320, 322, 323		
\newif	4, 5, 18, 25, 29, 30, 187		
\newlength	124, 279, 290, 301, 308, 318		
\newrobustcmd	74, 79, 80, 98–100, 113, 126– 151, 153–159, 161, 222–225, 265–269, 320		
\newterm	34		
\newtoks	119, 173, 225, 226		
\newwrite	74, 164, 168, 173, 176		
\nfss@text	6		
		O	
		\O	23
		\o	23
		\OE	23
		\oe	23
		\openout	74, 164, 168, 173, 326, 329
		\OR	255
		\or	5, 7, 8, 27, 33, 208, 316, 336
		\org@glossaryentrynumbers	195, 210
		\org@glossarytitle	195
		\org@glspostdescription	40
		\org@ifKV@glslink@hyper	115
		\outputpenalty	289

P	
<code>\p@</code>	276, 296, 314, 315
<code>\p@glshyp@opt</code>	112
package options:	
acronym	16, 17, 37, 194, 222
true	17
counter	19
debug	
showtargets	6
description	247, 248
dua	246–248
entrycounter	11, 208, 210
true	12
esclocations	407
false	9
footnote	126–131, 244, 246, 247, 250
hyperfirst	
false	126–131
index	34
indexonlyfirst	391
kernelglossredefs	
nowarn	32
makeindex	166, 263
nogroupskip	280, 282, 283, 286, 287, 291, 292, 294, 302, 304, 306, 309, 311, 313
nolist	257
nolong	257, 279
nomain	16
nonumberlist	9
nosuper	257
notree	257
nowarn	5
numberline	7
sanitize	24, 66, 152, 154
sanitizesort	21
savewrites	31, 388
false	173
true	176, 182, 183
section	7, 44
sort	
def	13, 14
none	13
standard	13
use	13, 14, 407
style	8, 257
subentrycounter	12, 208, 210
toc	7
true	7
translate	26
false	26
translator	25
xindy	29, 30, 166, 263
<code>\PackageError</code>	6, 7, 16, 30, 36, 48, 54, 57, 58, 62, 68, 71, 78–83, 85–87, 96, 111, 152, 172, 173, 177, 180, 182, 201–203, 207, 209, 217, 218, 227, 228, 244, 245, 250, 253, 330, 352
<code>\PackageInfo</code>	5, 6, 173, 184
<code>\PackageWarning</code>	5, 6, 20
<code>\PackageWarningNoLine</code>	5, 6, 20, 382, 383
<code>\pagegoal</code>	289
<code>\pagelistname</code>	39, 282, 284, 286, 287, 293, 295, 304–308, 311–314
<code>\par</code>	40, 214, 276–278, 296, 298–301, 314, 315, 317–323, 332, 337–339
<code>\parindent</code>	296–301, 315, 317–319, 321–323, 337–339
<code>\parskip</code>	296–299, 315, 317, 318
<code>\PassOptionsToPackage</code>	264, 344
<code>\penalty</code>	289
<code>\phantomsection</code>	45
polyglossia package	26, 38
<code>\printglossaries</code>	177
<code>\printglossary</code> ..	17, 20, 33, 34, 177, 194, 209
<code>\printglossary options</code>	
entrycounter	208
nogroupskip	208
nonumberlist	209
nopostdot	208
numberedsection	208
style	207
subentrycounter	208
title	207
toctitle	207
type	16, 193, 207
<code>\printindex</code>	34
<code>\printnoidxglossaries</code>	179
<code>\printnoidxglossary</code>	179, 182, 194, 201, 202, 209
<code>\printnoidxglossary options</code>	
sort	209
<code>\printnumbers</code>	33
<code>\printsymbols</code>	33
<code>\ProcessOptions</code>	264, 344
<code>\ProcessOptionsX</code>	34
<code>\protect</code>	46, 109, 110, 229, 230, 236, 242, 246, 248, 361, 365, 366, 371, 372
<code>\protected@csedef</code>	81

<code>\protected@csxdef</code>	80		
<code>\protected@edef</code>	8, 49, 50, 53, 55, 86, 89, 91, 102–104, 109, 110, 116, 117, 159, 184, 187, 189, 208, 212, 214, 217, 226, 250, 256, 265, 271, 325, 326, 344, 345, 350		
<code>\protected@write</code>	62, 64, 164– 166, 176, 177, 179, 182, 185, 193, 272, 326		
<code>\protected@xdef</code>	13–15, 18, 23, 73, 91, 92, 189, 348		
<code>\providecommand</code>	17, 36, 37, 44, 62, 98, 125, 166, 176, 177, 179, 182, 197, 212, 213, 276, 296, 314		
<code>\ProvidesFile</code>	38		
<code>\ProvidesPackage</code>	4, 264, 271, 273, 276, 279, 285, 290, 295, 301, 308, 314, 324, 330, 344, 382		
R			
<code>\r</code>	23		
<code>\raggedright</code>	288–295, 309–314		
<code>\raisebox</code>	124		
<code>\ref</code>	211		
<code>\refstepcounter</code>	210		
<code>\relax</code>	5, 8, 10, 14–17, 26, 27, 30, 32, 34, 35, 49, 62, 67, 69, 73, 75, 86, 89, 93, 97, 98, 112, 113, 116, 117, 119–124, 153, 167–170, 172, 173, 176, 177, 179, 181, 185, 189, 190, 192, 195, 196, 204, 205, 207, 208, 216, 217, 257, 272, 276, 289, 296, 300, 301, 314, 316–323, 325, 328– 330, 336–339, 347, 362, 363, 375–377, 379		
<code>\renewacronymstyle</code> .	365–369, 371, 373, 374		
<code>\renewcommand</code>	4– 10, 12, 13, 16, 17, 19–21, 24–26, 28, 30, 31, 33, 35, 38–42, 54, 66, 68–70, 83, 96, 97, 159, 161, 163, 164, 169, 171, 176– 181, 196, 197, 208, 226, 227, 229–233, 235–238, 241–245, 247, 248, 250, 253, 254, 256, 274–284, 286, 287, 289–306, 309–313, 315–325, 330–338, 340–343, 347, 352, 356, 359, 361, 364–368, 370–378		
<code>\renewenvironment</code>	212, 273, 276, 280–285, 288–315, 317, 318, 320		
<code>\RequireGlossariesLang</code>	39, 382, 383		
<code>\RequirePackage</code>	4, 9, 10, 26, 27, 34, 38, 257, 263, 264, 279, 285, 286, 290, 296, 301, 308, 345, 382		
<code>\restorecounters@</code>	116		
<code>\romannumeral</code> ...	186, 188, 189, 320–322, 338		
S			
<code>\s@Gls@hyp@opt</code>	112		
<code>\s@Gls@SetXdyFirstLetterAfterDigits</code>	163		
<code>\s@Gls@SetXdyNumberGroupOrder</code>	163		
<code>\s@newglossary</code>	63		
<code>\savecounters@</code>	116		
<code>\seename</code>	192		
<code>\SetAcronymStyle</code>	28		
<code>\setbool</code>	25		
<code>\setbox</code>	289, 290		
<code>\setcounter</code>	210		
<code>\SetCustomDisplayStyle</code>	257		
<code>\SetDefaultAcronymDisplayStyle</code>	241		
<code>\SetDefaultAcronymStyle</code>	255		
<code>\SetDescriptionAcronymDisplayStyle</code>	248		
<code>\SetDescriptionAcronymStyle</code>	255		
<code>\SetDescriptionDUAAcronymDisplayStyle</code>	245, 246		
<code>\SetDescriptionDUAAcronymStyle</code>	255		
<code>\SetDescriptionFootnoteAcronymDisplayStyle</code>	244		
<code>\SetDescriptionFootnoteAcronymStyle</code>	255		
<code>\SetDUADisplayStyle</code>	255		
<code>\SetDUASyle</code>	255		
<code>\setentrycounter</code>	48, 166, 206, 324		
<code>\SetFootnoteAcronymDisplayStyle</code> ...	250		
<code>\SetFootnoteAcronymStyle</code>	255		
<code>\SetGenericNewAcronym</code>	228		
<code>\setglossarystyle</code>	195, 217, 257, 277–289, 291–313, 316–319, 322, 323		
<code>\setglossentrycompatibility</code> ...	207, 217		
<code>\setkeys</code> ..	25, 29, 35, 44, 85, 115, 161, 195, 226, 241, 243, 245, 248, 250, 253, 254, 257		
<code>\setlength</code>	279, 290, 296– 299, 301, 308, 315, 317, 318, 322, 338, 339		
<code>\SetSmallAcronymDisplayStyle</code>	253		
<code>\SetSmallAcronymStyle</code>	255		
<code>\settoheight</code>	124		
<code>\settowidth</code>	230, 320–322, 338		
<code>\sfcode</code>	11		
<code>\show</code>	258–263, 380, 381		
<code>\small</code>	6		
<code>\SmallNewAcronymDef</code>	253		
<code>\space</code>	6, 7, 30–32, 36, 48, 52, 53, 55, 69, 71, 96, 98–100, 109, 110, 112, 159, 165–168, 172, 174, 175, 177, 179, 182, 193, 195, 197, 211, 217, 223, 229, 230, 232, 233, 235–238, 246, 251, 273, 275–278, 280, 291, 302, 309, 315–317,		

\xatlevel@	116	\xmakefirstuc	102, 104, 109, 110, 152, 153, 265
\xcapitalisewords	156	\xspace	221
\xdef	86, 87, 89, 196, 272	xspace package	4, 221
\xglaccsupp	350		
\xifinlistcs	198, 199, 202		
xindy	384	Y	
xindy	9, 13, 29, 30,	\year	164, 168, 326, 329
	40, 41, 46, 50, 52–55, 91, 123, 124, 163–		
	165, 183, 187, 188, 190, 197, 215, 263, 325	Z	
		\z@	289