

# Documented Code For glossaries v4.37

Nicola L.C. Talbot

Dickimaw Books

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

2018-04-07

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 v4.37: L<sup>A</sup>T<sub>E</sub>X2e 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

<b>1</b>	<b>Main Package Code</b>	<b>4</b>
1.1	Package Definition	4
1.2	Package Options	5
1.3	Predefined Text	33
1.4	Xindy	43
1.5	Loops and conditionals	52
1.6	Defining new glossaries	58
1.7	Defining new entries	62
1.8	Resetting and unsetting entry flags	88
1.9	Keeping Track of How Many Times an Entry Has Been Unset	92
1.10	Loading files containing glossary entries	96
1.11	Using glossary entries in the text	97
1.12	Adding an entry to the glossary without generating text	156
1.13	Creating associated files	158
1.14	Writing information to associated files	177
1.15	Glossary Entry Cross-References	186
1.16	Displaying the glossary	188
1.17	Acronyms	217
1.18	Predefined acronym styles	221
1.19	Predefined Glossary Styles	253
1.20	Debugging Commands	254
1.21	Compatibility with version 2.07 and below	259
<b>2</b>	<b>Prefix Support (glossaries-prefix Code)</b>	<b>261</b>
<b>3</b>	<b>Glossary Styles</b>	<b>268</b>
3.1	Glossary hyper-navigation definitions (glossary-hypernav package)	268
3.2	In-line Style (glossary-inline.sty)	270
3.3	List Style (glossary-list.sty)	273
3.4	Glossary Styles using longtable (the glossary-long package)	276
3.5	Glossary Styles using longtable and booktabs (the glossary-longbooktabs) package	282
3.6	Glossary Styles using longtable (the glossary-longragged package)	287
3.7	Glossary Styles using multicol (glossary-mcols.sty)	292
3.8	Glossary Styles using supertabular environment (glossary-super package)	298
3.9	Glossary Styles using supertabular environment (glossary-superragged package)	305
3.10	Tree Styles (glossary-tree.sty)	311

<b>4 Backwards Compatibility</b>	<b>321</b>
4.1 glossaries-compatible-207 . . . . .	321
4.2 glossaries-compatible-307 . . . . .	327
<b>5 Accessibility Support (glossaries-accsupp Code)</b>	<b>341</b>
5.1 Defining Replacement Text . . . . .	342
5.2 Accessing Replacement Text . . . . .	345
5.3 Displaying the Glossary . . . . .	361
5.4 Acronyms . . . . .	362
5.5 Debugging Commands . . . . .	377
<b>6 Multi-Lingual Support</b>	<b>379</b>
6.1 Polyglossia Captions . . . . .	379
<b>Glossary</b>	<b>381</b>
<b>Change History</b>	<b>382</b>
<b>Index</b>	<b>405</b>

# 1 Main Package Code

## 1.1 Package Definition

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

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries}[2018/04/07 v4.37 (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}[\val\nr]{true,false,showtargets}[true]{%
34   \ifcase\nr\relax
35     \@gls@debugtrue
36     \renewcommand*\GlossariesWarning[1]{%
37       \PackageWarning{glossaries}{##1}%
38     }%
39     \renewcommand*\GlossariesWarningNoLine[1]{%
40       \PackageWarningNoLine{glossaries}{##1}%
41     }%
42     \let\@glsshowtarget\@gobble
43     \PackageInfo{glossaries}{debug mode ON (nowarn option disabled)}%
44   \or
45     \@gls@debugfalse
46     \let\@glsshowtarget\@gobble
47     \PackageInfo{glossaries}{debug mode OFF}%
48   \or
49     \@gls@debugtrue
50     \renewcommand*\GlossariesWarning[1]{%
51       \PackageWarning{glossaries}{##1}%
```

```

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

```

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

```

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

```

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

```

71 \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`

```

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

```

`seenoinindex`

```

88 \define@choicekey{glossaries.sty}{seenoinindex}[\val\nr]{error,warn,ignore}{%
89   \ifcase\nr

```

```

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

```

**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.

```
106 \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`.

```
107 \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`.

```

108 \ifcsundef{chapter}%
109 {\newcommand*{\@glossarysec}{section}}%
110 {\newcommand*{\@glossarysec}{chapter}}

```

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

```

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

```

Determine whether or not to use numbered sections.

**glossarysecstar**

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

```

**lossaryseclabel**

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

```

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

```
116 \newcommand*{\glsautoprefix}{}

```

numberedsection

```

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

```

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

```

136 \ifpackageloaded{classicthesis}
137 {\newcommand*{\@glossary@default@style}{index}}
138 {\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`.

```

139 \define@key{glossaries.sty}{style}{%
140   \def\@glossary@default@style{#1}%
141 }

```

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

```

142 \newcommand*{\@gls@declareoption}[2]{%
143   \DeclareOptionX{#1}{#2}%
144   \DeclareOption{#1}{#2}%
145 }

```



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

```
146 \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).

```
147 \@gls@declareoption{nonumberlist}{%
148   \renewcommand*{\glossaryentrynumbers}[1]{\gls@save@numberlist{#1}}%
149 }
```

savenumberlist

Provide means to store the number list for entries.

```
150 \define@boolkey{glossaries.sty}[gls]{savenumberlist}[true]{}
151 \glssavenumberlistfalse
```

eautionumberlist

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

eautionumberlist

Automatically activates number list for entries containing the see key.

```
153 \@gls@declareoption{seeautonumberlist}{%
154   \renewcommand*{\@glo@seeautonumberlist}{%
155     \def\@glo@prefix{\glsnextpages}%
156   }%
157 }
```

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.

```
158 \define@boolkey{glossaries.sty}[gls]{esclocations}[true]{}
159 \glseesclocationstrue
```

\@gls@loadlong

```
160 \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.

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

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

```

162 \IfFileExists{supertabular.sty}{%
163   \newcommand*{\@gls@loadsuper}{\RequirePackage{glossary-super}}}%
164   \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.

```

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

```

`\@gls@loadlist`

```

166 \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`.

```

167 \@gls@declareoption{nolist}{%
168   \renewcommand*{\@gls@loadlist}{%
169     \ifdefstring{\@glossary@default@style}{list}%
170     {\let\@glossary@default@style\relax}%
171     }%
172   }%
173 }

```

`\@gls@loadtree`

```

174 \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.

```

175 \@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).

```

176 \@gls@declareoption{nostyles}{%
177   \renewcommand*{\@gls@loadlong}{}%
178   \renewcommand*{\@gls@loadsuper}{}%
179   \renewcommand*{\@gls@loadlist}{}%
180   \renewcommand*{\@gls@loadtree}{}%
181   \let\@glossary@default@style\relax
182 }

```

`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.)

```

183 \newcommand*{\glspostdescription}{%
184   \ifglsnopostdot\else.\spacefactor\sfcode'\. \fi
185 }

```

**nopostdot** Boolean option to suppress post description dot

```

186 \define@boolkey{glossaries.sty}[gls]{nopostdot}[true]{}
187 \glsnopostdotfalse

```

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

```

188 \define@boolkey{glossaries.sty}[gls]{nogroupskip}[true]{}
189 \glsnogroupskipfalse

```

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

```

190 \define@boolkey{glossaries.sty}[gls]{ucmark}[true]{}
191 \@ifclassloaded{memoir}
192 {%
193   \glsucmarktrue
194 }%
195 {%
196   \glsucmarkfalse
197 }

```

**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`.

```

198 \define@boolkey{glossaries.sty}[gls]{entrycounter}[true]{}
199 \glsentrycounterfalse

```

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

```

200 \define@key{glossaries.sty}{counterwithin}{%
201   \renewcommand*{\@gls@counterwithin}{#1}%
202   \glsentrycountertrue
203 }

```

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

```

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

```

**subentrycounter** Define a counter that can be used in the standard glossary styles to number each level 1 entry. If true, this will define a counter called `glossarysubentry`.

```

205 \define@boolkey{glossaries.sty}[gls]{subentrycounter}[true]{}
206 \glssubentrycounterfalse

```

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

```

207 \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`.

```

208 \define@choicekey{glossaries.sty}{sort}{standard,def,use,none}{%
209   \renewcommand*{\@gls@default@sorttype}{#1}%
210   \csname @gls@setupsort@#1\endcsname
211 }

```

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.

```
212 \newcommand*{\glsprestandardsort}[3]{%
213   \glsdosanitizesort
214 }
```

glscheck@sortallowed

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

glssetupsort@standard

Set up the macros for default sorting.

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

Store entry information when it's defined.

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

No count register required for standard sort.

```
218   \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.)

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

```
220     \ifx\@glo@sort\@glsdefaultsort
```

```
221       \let\@glo@sort\@glo@name
```

```
222     \fi
```

```
223     \let\glsdosanitizesort\@gls@sanitizesort
```

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

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

```
226   }%
```

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

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

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

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

```
229 }
```

Set standard sort as the default:

```
230 \@gls@setupsort@standard
```

glsnumberfmt

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

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

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

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

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

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

```

236 \ifnum#1<10 0\fi
237 \number#1%
238 }

s@setupsort@def Set up the macros for order of definition sorting.
239 \newcommand*{\@gls@setupsort@def}{%
  Store entry information when it's defined.
240 \def\do@glo@storeentry{\@glo@storeentry}%
  Defined count register associated with the glossary.
241 \def\@gls@defsortcount##1{%
242 \expandafter\global
243 \expandafter\newcount\csname glossary@##1@sortcount\endcsname
244 }%
  Increment count register associated with the glossary and use as the sort key.
245 \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.
246 \ifcsundef{glossary@##1@sortcount}%
247 {\@gls@defsortcount{##1}}%
248 {}%
249 \expandafter\global\expandafter
250 \advance\csname glossary@##1@sortcount\endcsname by 1\relax
251 \expandafter\protected@xdef\csname glo@##2@sort\endcsname{%
252 \expandafter\glssortnumberfmt
253 {\csname glossary@##1@sortcount\endcsname}}%
254 }%
  Don't need to do anything when the entry is used.
255 \def\@gls@setsort##1{%
  This sort option is allowed with \makeglossaries and \makenoidxglossaries.
256 \let\@glo@check@sortallowed\@gobble
257 }

s@setupsort@use Set up the macros for order of use sorting.
258 \newcommand*{\@gls@setupsort@use}{%
  Don't store entry information when it's defined.
259 \let\do@glo@storeentry\@gobble
  Defined count register associated with the glossary.
260 \def\@gls@defsortcount##1{%
261 \expandafter\global
262 \expandafter\newcount\csname glossary@##1@sortcount\endcsname
263 }%
  Initialise the sort key to empty.
264 \def\@gls@defsort##1##2{%
265 \expandafter\gdef\csname glo@##2@sort\endcsname{%
266 }%

```

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

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

Get the parent, if one exists

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

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

```
269 \ifx\@glo@parent\@empty
```

```
270 \else
```

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

```
272 \fi
```

Set index information for this entry

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

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

```
275 \ifx\@gls@tmp\@empty
```

```
276 \expandafter\global\expandafter
```

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

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

```
279 \expandafter\glssortnumberfmt
```

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

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

```
282 \fi
```

```
283 }%
```

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

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

```
285 }
```

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

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

Don't store entry index information.

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

No count register required for standard sort.

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

Don't modify sort value.

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

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

```
291 }%
```

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

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

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

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

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

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

```
296 }
```

`\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.)

```
297 \newcommand*{\glsdefmain}{%
298   \if@gls@docloaded
299     \newglossary[glg2]{main}{gls2}{glo2}{\glossaryname}%
300   \else
301     \newglossary{main}{gls}{glo}{\glossaryname}%
302   \fi
```

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

```
303 \newcommand*{\gls@tr@set@main@toctitle}{%
304   \translatelet{\glossarytoctitle}{Glossary}%
305 }%
306 }
```

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`

```
307 \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`

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

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

```
309 \@gls@declareoption{nomain}{%
310   \let\glsdefaulttype\relax
311   \renewcommand*{\glsdefmain}{}%
312 }
```

`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.

```
313 \define@boolkey{glossaries.sty}[gls]{acronym}[true]{%
314   \ifglsacronym
315     \renewcommand{\@gls@do@acronymsdef}{%
316       \DeclareAcronymList{acronym}%
317       \newglossary[alg]{acronym}{acr}{acn}{\acronymname}%
318       \renewcommand*{\acronymtype}{acronym}%
319     }
```

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

```
319 \newcommand*{\gls@tr@set@acronym@toctitle}{%
320 \translatelet{\glossarytoctitle}{Acronyms}%
321 }%
322 }%
323 \else
324 \let\@gls@do@acronymsdef\relax
325 \fi
326 }
```

`\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.

```
327 \AtBeginDocument{%
328 \ifglsacronym
329 \ifbool{glscompatible-3.07}%
330 {}%
331 {%
332 \providecommand*{\printacronyms}[1][1]{%
333 \printglossary[type=\acronymtype,#1]}%
334 }%
335 \fi
336 }
```

`@do@acronymsdef` Set default value

```
337 \newcommand*{\@gls@do@acronymsdef}{}%
```

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

```
338 \@gls@declareoption{acronyms}{%
339 \glsacronymtrue
340 \renewcommand*{\@gls@do@acronymsdef}{%
341 \DeclareAcronymList{acronym}%
342 \newglossary[alg]{acronym}{acr}{acn}{\acronymname}%
343 \renewcommand*{\acronymtype}{acronym}%
344 }
```

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

```
344 \newcommand*{\gls@tr@set@acronym@toctitle}{%
345 \translatelet{\glossarytoctitle}{Acronyms}%
346 }%
347 }%
348 }
```

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

```
349 \newcommand*{\@glsacronymlists}{}%
```

`dtoacronymlists`

```
350 \newcommand*{\@addtoacronymlists}[1]{%
351 \ifx\@glsacronymlists\empty
```



```

352     \protected@xdef\@glsacronymlists{#1}%
353   \else
354     \protected@xdef\@glsacronymlists{\@glsacronymlists,#1}%
355   \fi
356 }

```

`\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.)

```

357 \newcommand*{\DeclareAcronymList}[1]{%
358   \glsIfListOfAcronyms{#1}{\@addtoacronymlists{#1}}%
359 }

```

`\IfListOfAcronyms`

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

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

```

360 \newcommand{\glsIfListOfAcronyms}[1]{%
361   \edef\@do@gls@islistofacronyms{%
362     \noexpand\@gls@islistofacronyms{#1}{\@glsacronymlists}}%
363   \@do@gls@islistofacronyms
364 }

```

Internal command requires label and list to be expanded:

```

365 \newcommand{\@gls@islistofacronyms}[4]{%
366   \def\gls@islistofacronyms##1,#1,##2\end@gls@islistofacronyms{%
367     \def\@before{##1}\def\@after{##2}}%
368   \gls@islistofacronyms,#2,#1,\@nil\end@gls@islistofacronyms
369   \ifx\@after\@nnil

```

Not found

```

370     #4%
371   \else

```

Found

```

372     #3%
373   \fi
374 }

```

`\glsisacronymlist` Convenient boolean.

```
375 \newif\if@glsisacronymlist
```

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

```

376 \newcommand*{\glsc@ckisacronymlist}[1]{%
377   \glsIfListOfAcronyms{#1}%
378   {\@glsisacronymlisttrue}{\@glsisacronymlistfalse}%
379 }

```

**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.)

```
380 \newcommand*{\SetAcronymLists}[1]{%
381   \renewcommand*{\@glsacronymlists}{#1}%
382 }
```

**acronymlists**

```
383 \define@key{glossaries.sty}{acronymlists}{%
384   \DeclareAcronymList{#1}%
385 }
```

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**

```
386 \newcommand{\glscounter}{page}
```

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

```
387 \define@key{glossaries.sty}{counter}{%
388   \renewcommand*{\glscounter}{#1}%
389 }
```

**gls@nohyperlist**

```
390 \newcommand*{\@gls@nohyperlist}{}%
```

**lareNoHyperList**

```
391 \newcommand*{\GlsDeclareNoHyperList}[1]{%
392   \ifdefempty\@gls@nohyperlist
393   {%
394     \renewcommand*{\@gls@nohyperlist}{#1}%
395   }%
396   {%
397     \appto\@gls@nohyperlist{,#1}%
398   }%
399 }
```

**nohypertypes**

```
400 \define@key{glossaries.sty}{nohypertypes}{%
401   \GlsDeclareNoHyperList{#1}%
402 }
```

**ossariesWarning** Prints a warning message.

```
403 \newcommand*{\GlossariesWarning}[1]{%
404   \PackageWarning{glossaries}{#1}%
405 }
```

esWarningNoLine Prints a warning message without the line number.

```

406 \newcommand*\GlossariesWarningNoLine}[1]{%
407   \PackageWarningNoLine{glossaries}{#1}%
408 }

```

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

```

409 \newcommand{\glosortentrieswarning}{%
410   \typeout{Using TeX to sort glossary entries---this may
411   take a while}%
412 }

```

nowarn Define package option to suppress warnings

```

413 \@gls@declareoption{nowarn}{%
414   \if@gls@debug
415     \GlossariesWarning{Warnings can't be suppressed in debug mode}%
416   \else
417     \renewcommand*\GlossariesWarning[1]{}%
418     \renewcommand*\GlossariesWarningNoLine[1]{}%
419     \renewcommand*\glosortentrieswarning{}%
420     \renewcommand*\@gls@missinglang@warn[2]{}%
421   \fi
422 }

```

issinglang@warn Missing language warning.

```

423 \newcommand*\@gls@missinglang@warn[2]{%
424   \PackageWarningNoLine{glossaries}%
425   {No language module detected for '#1'.\MessageBreak
426   Language modules need to be installed separately.\MessageBreak
427   Please check on CTAN for a bundle called\MessageBreak
428   'glossaries-#2' or similar}%
429 }

```

nolangwarn Suppress warning if language support not found.

```

430 \@gls@declareoption{nolangwarn}{%
431   \renewcommand*\@gls@missinglang@warn[2]{}%
432 }

```

nonglossdefined Issue a warning if overriding \printglossary

```

433 \newcommand*\@gls@warnonglossdefined}{%
434   \GlossariesWarning{Overriding \string\printglossary}%
435 }

```

theglossdefined Issue a warning if overriding theglossary

```

436 \newcommand*\@gls@warnontheglossdefined}{%
437   \GlossariesWarning{Overriding 'theglossary' environment}%
438 }

```

noredefwarn Suppress warning on redefinition of \printglossary

```
439 \@gls@declareoption{noredefwarn}{%  
440   \renewcommand*{\@gls@warnonglossdefined}{}%  
441   \renewcommand*{\@gls@warnontheglossdefined}{}%  
442 }
```

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

```
443 \newcommand*{\@gls@sanitizedesc}{%  
444 }
```

lssetexpandfield

```
\glssetexpandfield{<field>}
```

Sets field to always expand.

```
445 \newcommand*{\glssetexpandfield}[1]{%  
446   \csdef{gls@assign@#1@field}##1##2{%  
447     \@gls@expand@field{##1}{#1}{##2}%  
448   }%  
449 }
```

setnoexpandfield

```
\glssetnoexpandfield{<field>}
```

Sets field to never expand.

```
450 \newcommand*{\glssetnoexpandfield}[1]{%  
451   \csdef{gls@assign@#1@field}##1##2{%  
452     \@gls@noexpand@field{##1}{#1}{##2}%  
453   }%  
454 }
```

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

```
455 \glssetexpandfield{type}
```

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

```
456 \glssetnoexpandfield{desc}
```

escplural@field

```
457 \glssetnoexpandfield{descplural}
```

ls@sanitizename

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

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

```
459 \glssetnoexpandfield{name}
```

@sanitizesymbol

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

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

```
461 \glssetnoexpandfield{symbol}
```

bolplural@field

```
462 \glssetnoexpandfield{symbolplural}
```

Sanitizing stuff:

ls@sanitizesort

```
463 \newcommand*{\@gls@sanitizesort}{%  
464   \ifglssanitizesort  
465     \@gls@sanitizesort  
466   \else  
467     \@gls@nosanitizesort  
468   \fi  
469 }
```

ls@sanitizesort

```
470 \newcommand*{\@gls@sanitizesort{%  
471   \@onelevel@sanitize\@glo@sort  
472 }
```

@nosanitizesort

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

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

```
474 \newcommand*{\@gls@noidx@sanitizesort{%  
475   \ifdefvoid\@glo@sort  
476   }{%  
477   {%  
478     \expandafter\@gls@noidx@sanitizesort\@glo@sort\gls@end@sanitizesort  
479   }%  
480 }  
481 \def\@gls@noidx@sanitizesort#1#2\gls@end@sanitizesort{%  
482   \def\@glo@sort{#1#2}%  
483   \@onelevel@sanitize\@glo@sort  
484 }
```

@nosanitizesort

```
485 \newcommand*{\@gls@noidx@nosanitizesort}{%  
486   \ifdefvoid\@glo@sort  
487   }{%  
488   {%  
489     \expandafter\@gls@noidx@no@sanitizesort\@glo@sort\gls@end@sanitizesort  
490   }%
```

```

491 }
492 \def\@gls@noidx@no@sanitizesort#1#2\gls@end@sanitizesort{%
493   \bgroup
494     \glsnoidxstripaccents
495     \protected@xdef\@glo@sort{#1#2}%
496   \egroup
497   \let\@glo@sort\@glo@sort
498 }

```

`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`.

```

499 \newcommand*\glsnoidxstripaccents{%
500   \let\IeC\@firstofone
501   \let\'\@firstofone
502   \let\'\@firstofone
503   \let\~\@firstofone
504   \let\"@firstofone
505   \let\u\@firstofone
506   \let\t\@firstofone
507   \let\d\@firstofone
508   \let\r\@firstofone
509   \let\=\@firstofone
510   \let\.\@firstofone
511   \let\~\@firstofone
512   \let\v\@firstofone
513   \let\H\@firstofone
514   \let\c\@firstofone
515   \let\b\@firstofone

516   \let\a\@secondoftwo
517   \def\AE{AE}%
518   \def\ae{ae}%
519   \def\OE{OE}%
520   \def\oe{oe}%
521   \def\AA{AA}%
522   \def\aa{aa}%
523   \def\L{L}%
524   \def\l{l}%
525   \def\O{O}%
526   \def\o{o}%
527   \def\SS{SS}%
528   \def\ss{ss}%
529   \def\th{th}%

530   \def\TH{TH}%
531   \def\dh{dh}%
532   \def\DH{DH}%
533 }

```

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.

```

534 \define@boolkey[glS]{sanitize}{description}[true]{%
535   \GlossariesWarning{sanitize={description} package option deprecated}%
536   \ifglS@sanitize@description
537     \glSsetnoexpandfield{desc}%
538     \glSsetnoexpandfield{descplural}%
539   \else
540     \glSsetexpandfield{desc}%
541     \glSsetexpandfield{descplural}%
542   \fi
543 }

544 \define@boolkey[glS]{sanitize}{name}[true]{%
545   \GlossariesWarning{sanitize={name} package option deprecated}%
546   \ifglS@sanitize@name
547     \glSsetnoexpandfield{name}%
548   \else
549     \glSsetexpandfield{name}%
550   \fi
551 }

552 \define@boolkey[glS]{sanitize}{symbol}[true]{%
553   \GlossariesWarning{sanitize={symbol} package option deprecated}%
554   \ifglS@sanitize@symbol
555     \glSsetnoexpandfield{symbol}%
556     \glSsetnoexpandfield{symbolplural}%
557   \else
558     \glSsetexpandfield{symbol}%
559     \glSsetexpandfield{symbolplural}%
560   \fi
561 }

```

sanitizesort

```

562 \define@boolkey{glossaries.sty}[glS]{sanitizesort}[true]{%
563   \ifglssanitizesort
564     \glSsetnoexpandfield{sortvalue}%
565     \renewcommand*{\@glS@noidx@setsanitizesort}{%
566       \glssanitizesorttrue
567       \glSsetnoexpandfield{sortvalue}%
568     }%
569   \else
570     \glSsetexpandfield{sortvalue}%
571     \renewcommand*{\@glS@noidx@setsanitizesort}{%
572       \glssanitizesortfalse
573       \glSsetexpandfield{sortvalue}%
574     }%
575   \fi
576 }

```

Default setting:

```
577 \glssanitizesorttrue
578 \glsssetnoexpandfield{sortvalue}%
```

setsanitizesort Default behaviour for \makenoidxglossaries is sanitizesort=false.

```
579 \newcommand*{\@gls@noidx@setsanitizesort}{%
580   \glssanitizesortfalse
581   \glsssetexpandfield{sortvalue}%
582 }

583 \define@choicekey[gls]{sanitize}{sort}{true,false}[true]{%
584   \setbool{glssanitizesort}{#1}%
585   \ifglssanitizesort
586     \glsssetnoexpandfield{sortvalue}%
587   \else
588     \glsssetexpandfield{sortvalue}%
589   \fi
590   \GlossariesWarning{sanitize={sort} package option
591     deprecated. Use sanitizesort instead}%
592 }
```

sanitize

```
593 \define@key{glossaries.sty}{sanitize}[description=true,symbol=true,name=true]{%
594   \ifthenelse{\equal{#1}{none}}{}%
595   {%
596     \GlossariesWarning{sanitize package option deprecated}%
597     \glsssetexpandfield{name}%
598     \glsssetexpandfield{symbol}%
599     \glsssetexpandfield{symbolplural}%
600     \glsssetexpandfield{desc}%
601     \glsssetexpandfield{descplural}%
602   }%
603   {%
604     \setkeys[gls]{sanitize}{#1}%
605   }%
606 }
```

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

```
607 \newif\ifglstranslate
```

otranslatorhook \@gls@notranslatorhook has been removed.

s@usetranslator

```
608 \newcommand*{\@gls@usetranslator{%
polyglossia tricks \@ifpackageloaded into thinking that babel has been loaded, so check for
polyglossia as well.
609   \@ifpackageloaded{polyglossia}%
```



```

610 {%
611   \let\glsifusetranslator\@secondoftwo
612 }%
613 {%
614   \@ifpackageloaded{babel}%
615   {%
616     \IfFileExists{translator.sty}%
617     {%
618       \RequirePackage{translator}%
619       \let\glsifusetranslator\@firstoftwo
620     }%
621   }%
622 }%
623 {}%
624 }%
625 }

```

**dtranslator**dict Checks if given translator dictionary has been loaded.

```

626 \newcommand{\glsifusedtranslator}{3}{%
627   \glsifusetranslator
628   {\ifcsdef{ver@glossaries-dictionary-#1.dict}{#2}{#3}}%
629   {#3}%
630 }

```

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

```

631 \@gls@declareoption{notranslate}{%
632   \glstranslatefalse
633   \let\@gls@usetranslator\relax
634   \let\glsifusetranslator\@secondoftwo
635 }

```

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

```

636 \define@choicekey{glossaries.sty}{translate}[\val\nr]%
637 {true,false,babel}[true]%
638 {%
639   \ifcase\nr\relax
640     \glstranslatetrue
641     \renewcommand*\@gls@usetranslator{%
642       \@ifpackageloaded{polyglossia}%
643       {%
644         \let\glsifusetranslator\@secondoftwo
645       }%
646     }%
647     \@ifpackageloaded{babel}%
648     {%
649       \IfFileExists{translator.sty}%
650       {%
651         \RequirePackage{translator}%
652         \let\glsifusetranslator\@firstoftwo

```

```

653         }%
654     {}%
655     }%
656     {}%
657     }%
658     }%
659 \or
660     \glstranslatefalse
661     \let\@gls@usetranslator\relax
662     \let\glsifusetranslator\@secondoftwo
663 \or
664     \glstranslatetrue
665     \let\@gls@usetranslator\relax
666     \let\glsifusetranslator\@secondoftwo
667 \fi
668 }

```

Set the default value:

```

669 \glstranslatefalse
670 \let\glsifusetranslator\@secondoftwo
671 \@ifpackageloaded{translator}%
672 {%
673     \glstranslatetrue
674     \let\glsifusetranslator\@firstoftwo
675 }%
676 {%
677     \@for\gls@thissty:=tracklang,babel,ngerman,polyglossia\do
678     {
679         \@ifpackageloaded{\gls@thissty}%
680         {%
681             \glstranslatetrue
682             \@endfortrue
683         }%
684         {}%
685     }
686 }

```

**indexonlyfirst** Set whether to only index on first use.

```

687 \define@boolkey{glossaries.sty}[gls]{indexonlyfirst}[true]{}
688 \glsindexonlyfirstfalse

```

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

```

689 \define@boolkey{glossaries.sty}[gls]{hyperfirst}[true]{}
690 \glshyperfirsttrue

```

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

```

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

```

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

```

692 \define@boolkey{glossaries.sty}[glsacr]{footnote}[true]{%
693   \ifbool{glsacrdescription}%
694   {}%
695   {%
696     \renewcommand*{\@gls@sanitizedesc}{}%
697   }%
698   \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
699 }

```

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

```

700 \define@boolkey{glossaries.sty}[glsacr]{description}[true]{%
701   \renewcommand*{\@gls@sanitizesymbol}{}%
702   \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
703 }

```

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

```

704 \define@boolkey{glossaries.sty}[glsacr]{smallcaps}[true]{%
705   \renewcommand*{\@gls@sanitizesymbol}{}%
706   \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
707 }

```

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

```

708 \define@boolkey{glossaries.sty}[glsacr]{smaller}[true]{%
709   \renewcommand*{\@gls@sanitizesymbol}{}%
710   \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
711 }

```

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

```

712 \define@boolkey{glossaries.sty}[glsacr]{dua}[true]{%
713   \renewcommand*{\@gls@sanitizesymbol}{}%
714   \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
715 }

```

**shortcuts** Define acronym shortcuts.

```

716 \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.

```

717 \newcommand*{\glsorder}{word}

```

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

```

718 \newcommand*{\@glsorder}[1]{}

```

**order**

```

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

```

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

```
721 \newif\ifglxindy
```

The default is `makeindex`:

```
722 \glxindyfalse
```

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

```
723 \@gls@declareoption{makeindex}{\glxindyfalse}
```

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

```
724 \define@boolkey[gls]{xindy}{glsnumbers}[true]{}
725 \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.)

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

Define key to set the language

```
727 \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.

```
728 \ifcsundef{inputencodingname}{%
729   \def\gls@codepage{}}{%
730   \def\gls@codepage{\inputencodingname}
731 }
```

Define a key to set the code page.

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

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

```
733 \define@key{glossaries.sty}{xindy}[]{%
734   \glxindytrue
735   \setkeys[gls]{xindy}{#1}%
736 }
```

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

```
737 \@gls@declareoption{xindygloss}{%
738   \glxindytrue
739 }
```

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

```
740 \@gls@declareoption{xindynoglsnumbers}{%
741   \glxindytrue
742   \gls@xindy@glsnumbersfalse
743 }
```

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

```

744 \define@boolkey{glossaries.sty}[gls]{automake}[true]{%
745   \ifglssautomake
746     \renewcommand*{\@gls@doautomake}{%
747       \PackageError{glossaries}{You must use
748       \string\makeglossaries\space with automake=true}
749       {%
750         Either remove the automake=true setting or
751         add \string\makeglossaries\space to your document preamble.%
752       }%
753     }%
754   \else
755     \renewcommand*{\@gls@doautomake}{}%
756   \fi
757 }
758 \glssautomakefalse

```

**@gls@doautomake**

```

759 \newcommand*{\@gls@doautomake}{}
760 \AtEndDocument{\@gls@doautomake}

```

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

```

761 \define@boolkey{glossaries.sty}[gls]{savewrites}[true]{%
762   \ifglssavewrites
763     \renewcommand*{\glswritefiles}{\@glswritefiles}%
764   \else
765     \let\glswritefiles\@empty
766   \fi
767 }

```

Set default:

```

768 \glssavewritesfalse
769 \let\glswritefiles\@empty

```

**compatible-3.07**

```

770 \define@boolkey{glossaries.sty}[gls]{compatible-3.07}[true]{}
771 \boolfalse{glscpatible-3.07}

```

**compatible-2.07**

```

772 \define@boolkey{glossaries.sty}[gls]{compatible-2.07}[true]{%
  Also set 3.07 compatibility if this option is set.
773   \ifbool{glscpatible-2.07}{%
774     {%
775       \booltrue{glscpatible-3.07}%
776     }%
777   }%
778 }
779 \boolfalse{glscpatible-2.07}

```

symbols Create a “symbols” glossary type

```
780 \@gls@declareoption{symbols}{%  
781 \let\@gls@do@symbolsdef\@gls@symbolsdef  
782 }
```

Default is not to define the symbols glossary:

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

@gls@symbolsdef

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

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

```
787 \newcommand*{\gls@tr@set@symbols@toctitle}{%  
788 \translatelet{\glossarytoctitle}{Symbols (glossaries)}%  
789 }%  
790 }%
```

numbers Create a “symbols” glossary type

```
791 \@gls@declareoption{numbers}{%  
792 \let\@gls@do@numbersdef\@gls@numbersdef  
793 }
```

Default is not to define the numbers glossary:

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

@gls@numbersdef

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

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

```
798 \newcommand*{\gls@tr@set@numbers@toctitle}{%  
799 \translatelet{\glossarytoctitle}{Numbers (glossaries)}%  
800 }%  
801 }%
```

index Create an “index” glossary type

```
802 \@gls@declareoption{index}{%  
803 \let\@gls@do@indexdef\@gls@indexdef  
804 }
```

Default is not to define index glossary:

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

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

```
806 \newcommand*{\@gls@indexdef}{%  
807 \newglossary[ilg]{index}{ind}{idx}{\indexname}%  
808 \newcommand*{\printindex}[1][]{\printglossary[type=index,##1]}%
```

```

809 \newcommand*{\newterm}[2] [] {%
810   \newglossaryentry{##2}%
811   {type={index},name={##2},description={\nopostdesc},##1}}
812 }%

```

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

```

813 \@for\CurrentOption := \@declaredoptions\do{%
814   \ifx\CurrentOption \@empty
815   \else
816     \@expandtwoargs
817     \in@ {,\CurrentOption ,}{,\@classoptionslist,\@curroptions,}%
818     \ifin@
819     \@use@option
820     \expandafter \let\csname ds@\CurrentOption\endcsname \@empty
821   \fi
822 \fi
823 }

```

Now process options passed to the package:

```
824 \ProcessOptionsX
```

Load backward compatibility stuff:

```
825 \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:

```

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

```

Now define `\setupglossaries`:

```

829 \newcommand*{\setupglossaries}[1] {%
830   \renewcommand*{\@gls@setacrstyle}{}%
831   \ifglsacrshortcuts
832     \def\@gls@setupshortcuts{\glsacrshortcutstrue}%
833   \else
834     \def\@gls@setupshortcuts{%
835       \ifglsacrshortcuts
836         \DefineAcronymSynonyms
837       \fi
838     }%
839   \fi
840   \glsacrshortcutsfalse
841   \let\@gls@do@numbersdef\relax
842   \let\@gls@do@symbolssdef\relax
843   \let\@gls@do@indexdef\relax
844   \let\@gls@do@acronymsdef\relax
845   \setkeys{glossaries.sty}{#1}%
846   \@gls@setacrstyle

```

```

847 \@gls@setupshortcuts
848 \@gls@do@acronymsdef
849 \@gls@do@numbersdef
850 \@gls@do@symbolssdef
851 \@gls@do@indexdef
852 }

```

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

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

```

853 \ifthenelse{\equal{\glscounter}{section}}{%
854 {%
855   \ifcsundef{chapter}{}%
856   {%
857     \let\@gls@old@chapter\@chapter
858     \def\@chapter[#1]#2{\@gls@old@chapter[#1]{#2}%
859     \ifcsundef{hyperdef}{}{\hyperdef{section}{\thesection}{}}}%
860   }%
861 }%
862 {}

```

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

```

863 \newcommand*{\@gls@onlypremakeg}{}

```

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

```

864 \newcommand*{\@onlypremakeg}[1]{%
865   \ifx\@gls@onlypremakeg\@empty
866     \def\@gls@onlypremakeg{#1}%
867   \else
868     \expandafter\toks@\expandafter{\@gls@onlypremakeg}%
869     \edef\@gls@onlypremakeg{\the\toks@,\noexpand#1}%
870   \fi
871 }

```

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

```

872 \newcommand*{\@disable@onlypremakeg}{%
873 \@for\@thiscs:=\@gls@onlypremakeg\do{%
874   \expandafter\@disable@premakecs\@thiscs%
875 }}

```

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



```

876 \newcommand*{\@disable@premakecs}[1]{%
877   \def#1{\PackageError{glossaries}{\string#1\space may only be
878     used before \string\makeglossaries}{You can't use
879     \string#1\space after \string\makeglossaries}}}%
880 }

```

## 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

```
881 \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

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

\glstocctitle Sets the TOC title for the given glossary.

```

883 \newcommand*{\glstocctitle}[1]{%
884   \def\glossarytocctitle{\csname @glotype@#1@title\endcsname}}

```

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

\entryname

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

\descriptionname

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

\symbolname

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

\pagelistname

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

Labels for makeindex's symbol and number groups:

\symbolsgroupname

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

\numbersgroupname

```
890 \providecommand*{\glsnnumbersgroupname}{Numbers}
```

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

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

`acrpluralsuffix`  
893 `\newcommand*{\glsupacrpluralsuffix}{\glstextup{\glsacrpluralsuffix}}`

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

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

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

`eGlossariesLang`  
896 `\newcommand*{\RequireGlossariesLang}[1]{%`  
897 `\@ifundefined{ver@glossaries-#1.ldf}{\input{glossaries-#1.ldf}}{}`  
898 `}`

`sGlossariesLang`  
899 `\newcommand*{\ProvidesGlossariesLang}[1]{%`  
900 `\ProvidesFile{glossaries-#1.ldf}%`  
901 `}`

`ssarytocaptions` Does nothing if translator hasn't been loaded.  
902 `\newcommand*{\addglossarytocaptions}[1]{}`

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

903 `\ifglstranslate`

Load tracklang  
904 `\RequirePackage{tracklang}`

Load translator if required.  
905 `\@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.)  
906 `\@ifpackageloaded{translator}`  
907 `{%`

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.

```

908   \ifboolexpr
909   {
910     test {\ifdefstring{\trans@languages}{English}}
911     and not
912     test {\ifdefstring{\bbl@loaded}{english}}
913   }
914   {%
915     \let\glsifusetranslator\@secondoftwo
916   }%
917   {%
918     \usedictionary{glossaries-dictionary}%
919     \renewcommand*{\addglossarytocaptions}[1]{%
920       \ifcsundef{captions#1}{}%
921       {%
922         \expandafter\let\expandafter\@gls@tmp\csname captions#1\endcsname
923         \expandafter\toks@\expandafter{\@gls@tmp
924         \renewcommand*{\glossaryname}{\translate{Glossary}}}%
925       }%
926       \expandafter\edef\csname captions#1\endcsname{\the\toks@}%
927     }%
928   }%
929 }%
930 }%
931 {}%
```

Check for tracked languages

```

932 \AnyTrackedLanguages
933 {%
934   \ForEachTrackedDialect{\this@dialect}{%
935     \IfTrackedLanguageFileExists{\this@dialect}%
936     {glossaries-}% prefix
937     {.ldf}%
938     {%
939       \RequireGlossariesLang{\CurrentTrackedTag}%
940     }%
941     {%
942       \@gls@missinglang@warn\this@dialect\CurrentTrackedLanguage
943     }%
944   }%
945 }%
946 {}%
```

if using translator use translator interface.

```

947 \glsifusetranslator
948 {%
949   \renewcommand*{\glssettoctitle}[1]{%
```

```

950     \ifcsdef{gls@tr@set@#1@toctitle}%
951     {%
952         \csuse{gls@tr@set@#1@toctitle}%
953     }%
954     {%
955         \def\glossarytoctitle{\csname @glotype@#1@title\endcsname}%
956     }%
957 }%
958 \renewcommand*{\glossaryname}{\translate{Glossary}}%
959 \renewcommand*{\acronymname}{\translate{Acronyms}}%
960 \renewcommand*{\entryname}{\translate{Notation (glossaries)}}%
961 \renewcommand*{\descriptionname}{%
962     \translate{Description (glossaries)}}%
963 \renewcommand*{\symbolname}{\translate{Symbol (glossaries)}}%
964 \renewcommand*{\pagelistname}{%
965     \translate{Page List (glossaries)}}%
966 \renewcommand*{\glssymbolsgroupname}{%
967     \translate{Symbols (glossaries)}}%
968 \renewcommand*{\glsnumbersgroupname}{%
969     \translate{Numbers (glossaries)}}%
970 }{}%
971 \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.

```
972 \DeclareRobustCommand*{\nopostdesc}{}

```

`\@nopostdesc` Suppress next description terminator.

```

973 \newcommand*{\@nopostdesc}{%
974     \let\org@glspostdescription\glspostdescription
975     \def\glspostdescription{%
976         \let\glspostdescription\org@glspostdescription}%
977 }

```

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

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

```

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

```
979 \newcommand{\glspar}{\par}

```

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

```

980 \newcommand{\setStyleFile}[1]{%
981     \renewcommand*{\gls@istfilebase}{#1}%
    Just in case \istfilename has been modified.
982     \ifglxsindy
983         \def\istfilename{\gls@istfilebase.xdy}
984     \else
985         \def\istfilename{\gls@istfilebase.ist}

```

```
986 \fi
987 }
```

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

```
988 \@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
```

```
989 \ifglxindy
990 \def\istfilename{\gls@istfilebase.xdy}
991 \else
992 \def\istfilename{\gls@istfilebase.ist}
993 \fi
```

```
gls@istfilebase
```

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

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

```
\@istfilename
```

```
995 \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
```

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

```
lsSetCompositor Sets the compositor.
```

```
997 \newcommand*{\glsSetCompositor}[1]{%
998 \renewcommand*{\glscompositor}{#1}}
```

Only use before `\makeglossaries`

```
999 \@onlypremakeg\glsSetCompositor
```

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

**Alphacompositor** This is only used by xindy. It specifies the compositor to use when location numbers are in the form  $\langle letter \rangle \langle compositor \rangle \langle number \rangle$ . 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.

```
1000 \newcommand*{\@glsAlphacompositor}{\glscompositor}
```

**AlphaCompositor** Sets the alpha compositor.

```
1001 \ifglsxindy
1002   \newcommand*\glsSetAlphaCompositor[1]{%
1003     \renewcommand*\@glsAlphacompositor{#1}}
1004 \else
1005   \newcommand*\glsSetAlphaCompositor[1]{%
1006     \glsnxindywarning\glsSetAlphaCompositor}
1007 \fi
```

Can only be used before `\makeglossaries`

```
1008 \@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.

```
1009 \newcommand*{\gls@suffixF}{}
```

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

```
1010 \newcommand*{\glsSetSuffixF}[1]{%
1011   \renewcommand*{\gls@suffixF}{#1}}
```

Only has an effect when used before `\makeglossaries`

```
1012 \@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.

```
1013 \newcommand*{\gls@suffixFF}{}
```

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

```
1014 \newcommand*{\glsSetSuffixFF}[1]{%
1015   \renewcommand*{\gls@suffixFF}{#1}%
1016 }
```

**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”.

```
1017 \ifcsundef{hyperlink}%
1018 {%
1019   \newcommand*\glsnumberformat[1]{#1}%
1020 }%
1021 {%
```

```

1022 \newcommand*{\glsnumberformat}[1]{\glshypernumber{#1}}%
1023 }

```

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
1024 \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
1025 \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
1026 \newcommand*{\glossarypreamble}{%
1027   \csuse{@glossarypreamble@currentglossary}%
1028 }

```

```

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

```

Code provided by Michael Pock.

```

1029 \newcommand{\setglossarypreamble}[2][\glsdefaultttype]{%
1030   \ifglossaryexists{#1}{%
1031     \csgdef{@glossarypreamble@#1}{#2}%
1032   }{%
1033     \GlossariesWarning{%
1034       Glossary ‘#1’ is not defined%
1035     }%
1036   }%
1037 }

```

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`

```
1038 \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`.

```
1039 \newcommand*{\glossarysection}[2][\@gls@title]{%
1040   \def\@gls@title{#2}%
1041   \ifcsundef{phantomsection}%
1042   {%
1043     \@glossarysection{#1}{#2}%
1044   }%
1045   {%
1046     \p@glossarysection{#1}{#2}%
1047   }%

1048   \glsglossarymark{\glossarytoctitle}%
1049 }
```

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

```
1050 \ifcsundef{glossarymark}%
1051 {%
1052   \newcommand{\glsglossarymark}[1]{\glossarymark{#1}}
1053 }%
1054 {%
1055   \@ifclassloaded{memoir}
1056   {%
1057     \newcommand{\glsglossarymark}[1]{%
1058       \ifglsucmark
1059         \markboth{\memUHead{#1}}{\memUHead{#1}}%
1060       \else
1061         \markboth{#1}{#1}%
1062       \fi
1063     }
1064   }%
1065   {%
1066     \newcommand{\glsglossarymark}[1]{%
1067       \ifglsucmark
1068         \@mkboth{\mfirstucMakeUppercase{#1}}{\mfirstucMakeUppercase{#1}}%
1069       \else
1070         \@mkboth{#1}{#1}%
1071       \fi
1072     }
1073   }%
1074 }
```



```

1072   }
1073 }
1074 }

```

`\glossarymark` Provided for backward compatibility:

```

1075 \providecommand{\glossarymark}[1]{%
1076   \ifglsucmark
1077     \mkboth{\mfirstucMakeUppercase{#1}}{\mfirstucMakeUppercase{#1}}%
1078   \else
1079     \mkboth{#1}{#1}%
1080   \fi
1081 }

```

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`

```

1082 \newcommand*{\setglossarysection}[1]{%
1083 \setkeys{glossaries.sty}{section=#1}}

```

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

`glossarysection`

```

1084 \newcommand*{\@glossarysection}[2]{%
1085   \ifdefempty\@glossarysecstar
1086   {%
1087     \csname\@glossarysec\endcsname[#1]{#2}%
1088   }%
1089   {%
1090     \csname\@glossarysec\endcsname*{#2}%
1091     \@gls@toc{#1}{\@glossarysec}%
1092   }%

```

Do automatic labelling if required

```

1093   \@glossaryseclabel
1094 }

```

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`

```

1095 \newcommand*{\@pglossarysection}[2]{%
1096   \glsclearpage
1097   \phantomsection
1098   \ifdefempty\@glossarysecstar
1099   {%

```

```

1100     \csname\@glossarysec\endcsname{#2}%
1101 }%
1102 {%
1103     \@gls@toc{#1}{\@glossarysec}%
1104     \csname\@glossarysec\endcsname*{#2}%
1105 }%

```

Do automatic labelling if required

```

1106 \@glossaryseclabel
1107 }

```

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

```

1108 \newcommand*{\gls@doclearpage}{%
1109     \ifthenelse{\equal{\@glossarysec}{chapter}}{%
1110     {%
1111         \ifcsundef{cleardoublepage}%
1112         {%
1113             \clearpage
1114         }%
1115     {%
1116         \ifcsdef{if@openright}%
1117         {%
1118             \if@openright
1119                 \cleardoublepage
1120             \else
1121                 \clearpage
1122             \fi
1123         }%
1124     {%
1125         \cleardoublepage
1126     }%
1127 }%
1128 }%
1129 {}%
1130 }

```

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

```

1131 \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`

```

1132 \newcommand*{\@gls@toc}[2]{%
1133     \ifglstoc

```

```

1134     \ifglslnumberline
1135         \addcontentsline{toc}{#2}{\protect\numberline{#1}}%
1136     \else
1137         \addcontentsline{toc}{#2}{#1}%
1138     \fi
1139 \fi
1140 }

```

## 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

```

1141 \newcommand*{\glsnoxywarning}[1]{%
1142   \GlossariesWarning{Not in xindy mode --- ignoring \string#1}%
1143 }

```

**makeindexwarning** Reverse for commands that may only be used with makeindex.

```

1144 \newcommand*{\glsnomakeindexwarning}[1]{%
1145   \GlossariesWarning{Not in makeindex mode --- ignoring \string#1}%
1146 }

```

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

```

1147 \ifglslxindy
1148   \edef\@xdyattributes{\string"default\string"}%
1149 \fi

```

**dyattributelist** Comma-separated list of attributes.

```

1150 \ifglslxindy
1151   \edef\@xdyattributelist{}%
1152 \fi

```

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

```

1153 \ifglslxindy
1154   \def\@xdylocref{}
1155 \fi

```

**\@gls@ifinlist**

```

1156 \newcommand*{\@gls@ifinlist}[4]{%
1157   \def\@do@ifinlist##1,##2\end@ifinlist{%
1158     \def\@gls@listsuffix{##2}%
1159     \ifx\@gls@listsuffix\@empty
1160       #4%
1161     \else
1162       #3%

```

```

1163     \fi
1164 }%
1165 \do@ifinlist,#2,#1,\end@do@ifinlist
1166 }

```

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

```

1167 \ifglxindy
1168   \newcommand*{\@xdycounters}{\glscounter}
1169   \newcommand*\GlsAddXdyCounters[1]{%
1170     \@for\@gls@ctr:=#1\do{%
1171       \edef\@do@addcounter{%
1172         \noexpand\@gls@ifinlist{\@gls@ctr}{\@xdycounters}{}%
1173         {%
1174           \noexpand\edef\noexpand\@xdycounters{\@xdycounters,%
1175             \noexpand\@gls@ctr}%
1176         }%
1177       }%
1178       \@do@addcounter
1179     }
1180   }

```

Only has an effect before `\writeist`:

```

1181   \@onlypremakeg\GlsAddXdyCounters
1182 \else
1183   \newcommand*\GlsAddXdyCounters[1]{%
1184     \glsnnoxindywarning\GlsAddXdyAttribute
1185   }
1186 \fi

```

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

```

1187 \newcommand*\@disabled@glssaddxdycounters{%
1188   \PackageError{glossaries}{\string\GlsAddXdyCounters\space
1189     can't be used after \string\GlsAddXdyAttribute}{Move all
1190     occurrences of \string\GlsAddXdyCounters\space before the first
1191     instance of \string\GlsAddXdyAttribute}%
1192 }

```

**AddXdyAttribute** Adds an attribute.

```

1193 \ifglxindy
1194   \newcommand*\@glssaddxdyattribute[2]{%
1195     \edef\@xdyattributes{\@xdyattributes ^^J \string"#1\string" ^^J
1196       \string"#2#1\string"}%

```

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

Add to xindy attribute list

Add to xindy markup location.

```

1197 \expandafter\toks@\expandafter{\@xdylocref}%
1198 \edef\@xdylocref{\the\toks@ ^^J%
1199 (markup-locref
1200 :open \string"glstildechar n%
1201 \expandafter\string\csname glsX#2X#1\endcsname
1202 \string" ^^J
1203 :close \string"\string" ^^J
1204 :attr \string"#2#1\string")}%

```

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

```

1205 \expandafter\gdef\csname glsX#2X#1\endcsname##1##2{%
1206 \setentrycounter[##1]{#2}\csname #1\endcsname{##2}%
1207 }%
1208 }

```

High-level command:

```

1209 \newcommand*\GlsAddXdyAttribute[1]{%

```

Add to comma-separated attribute list

```

1210 \ifx\@xdyattributelist\@empty
1211 \edef\@xdyattributelist{#1}%
1212 \else
1213 \edef\@xdyattributelist{\@xdyattributelist,#1}%
1214 \fi

```

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

```

1215 \@for\@this@counter:=\@xdycounters\do{%
1216 \protected@edef\gls@do@addxdyattribute{%
1217 \noexpand\@glsaddxdyattribute{#1}{\@this@counter}%
1218 }
1219 \gls@do@addxdyattribute
1220 }%

```

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

```

1221 \let\GlsAddXdyCounters\@disabled@glsaddxdycounters
1222 }

```

Only has an effect before `\writeist`:

```

1223 \@onlypremakeg\GlsAddXdyAttribute
1224 \else
1225 \newcommand*\GlsAddXdyAttribute[1]{%
1226 \glsnoxindywarning\GlsAddXdyAttribute}
1227 \fi

```

`\definedattributes` Add known attributes for all defined counters

```

1228 \ifglsxindy
1229 \newcommand*\@gls@addpredefinedattributes{%
1230 \GlsAddXdyAttribute{glsnumberformat}
1231 \GlsAddXdyAttribute{textrm}
1232 \GlsAddXdyAttribute{textsf}

```

```

1233 \GlsAddXdyAttribute{texttt}
1234 \GlsAddXdyAttribute{textbf}
1235 \GlsAddXdyAttribute{textmd}
1236 \GlsAddXdyAttribute{textit}
1237 \GlsAddXdyAttribute{textup}
1238 \GlsAddXdyAttribute{textsl}
1239 \GlsAddXdyAttribute{textsc}
1240 \GlsAddXdyAttribute{emph}
1241 \GlsAddXdyAttribute{glshypernumber}
1242 \GlsAddXdyAttribute{hyperrrm}
1243 \GlsAddXdyAttribute{hypersf}
1244 \GlsAddXdyAttribute{hypertt}
1245 \GlsAddXdyAttribute{hyperbf}
1246 \GlsAddXdyAttribute{hypermd}
1247 \GlsAddXdyAttribute{hyperit}
1248 \GlsAddXdyAttribute{hyperup}
1249 \GlsAddXdyAttribute{hypersl}
1250 \GlsAddXdyAttribute{hypersc}
1251 \GlsAddXdyAttribute{hyperemph}

1252 \GlsAddXdyAttribute{glsglignore}
1253 }
1254 \else
1255 \let\@gls@addpredefinedattributes\relax
1256 \fi

```

**dyuseralphabets** List of additional alphabets

```
1257 \def\@xdyuseralphabets{}
```

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

```

1258 \ifglsxindy
1259 \newcommand*\GlsAddXdyAlphabet[2]{%
1260 \edef\@xdyuseralphabets{%
1261 \@xdyuseralphabets ^^J
1262 (define-alphabet "#1" (#2))}}
1263 \else
1264 \newcommand*\GlsAddXdyAlphabet[2]{%
1265 \glsnnoxindywarning\GlsAddXdyAlphabet}
1266 \fi

```

This code is only required for xindy:

```
1267 \ifglsxindy
```

**dy@locationlist** List of predefined location names.

```

1268 \newcommand*\@gls@xdy@locationlist{%
1269 roman-page-numbers,%
1270 Roman-page-numbers,%
1271 arabic-page-numbers,%

```

```

1272     alpha-page-numbers,%
1273     Alpha-page-numbers,%
1274     Appendix-page-numbers,%
1275     arabic-section-numbers%
1276 }

```

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

```

1277 \protected@edef\@gls@roman{\@roman{0}\string"
1278     \string"roman-numbers-lowercase\string" :sep \string"}}%
1279 \@onelevel@sanitize\@gls@roman
1280 \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
1281     :sep \string"}%
1282 \@onelevel@sanitize\@tmp
1283 \ifx\@tmp\@gls@roman
1284     \expandafter
1285         \edef\csname @gls@xdy@Lclass@roman-page-numbers\endcsname{%
1286             \string"roman-numbers-lowercase\string"%
1287         }%
1288 \else
1289     \expandafter
1290         \edef\csname @gls@xdy@Lclass@roman-page-numbers\endcsname{
1291             :sep \string"\@gls@roman\string"%
1292         }%
1293 \fi

```

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

```

1294 \expandafter\def\csname @gls@xdy@Lclass@Roman-page-numbers\endcsname{%
1295     \string"roman-numbers-uppercase\string"%
1296 }%

```

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

```

1297 \expandafter\def\csname @gls@xdy@Lclass@arabic-page-numbers\endcsname{%
1298     \string"arabic-numbers\string"%
1299 }%

```

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

```

1300 \expandafter\def\csname @gls@xdy@Lclass@alpha-page-numbers\endcsname{%
1301     \string"alpha\string"%
1302 }%

```

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

```

1303 \expandafter\def\csname @gls@xdy@Lclass@Alpha-page-numbers\endcsname{%
1304     \string"ALPHA\string"%
1305 }%

```

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

```
1306 \expandafter\def\csname @gls@xdy@Lclass@Appendix-page-numbers\endcsname{%
1307   \string"ALPHA\string"
1308   :sep \string"\@glsAlphacompositor\string"
1309   \string"arabic-numbers\string"%
1310 }
```

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

```
1311 \expandafter\def\csname @gls@xdy@Lclass@arabic-section-numbers\endcsname{%
1312   \string"arabic-numbers\string"
1313   :sep \string"\glscompositor\string"
1314   \string"arabic-numbers\string"%
1315 }%
```

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

```
1316 \def\@xdyuserlocationdefs{}
```

erlocationnames List of additional user location names

```
1317 \def\@xdyuserlocationnames{}
```

End of xindy-only block:

```
1318 \fi
```

xdycrossrefhook Hook used after writing cross-reference class information.

```
1319 \ifglsxindy
1320 \newcommand\@xdycrossrefhook{}
1321 \fi
```

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

```
1322 \ifglsxindy
1323   \newcommand*\GlsAddXdyLocation[3][[]]{%
1324     \def\@gls@tmp{#1}%
1325     \ifx\@gls@tmp\@empty
1326       \edef\@xdyuserlocationdefs{%
1327         \@xdyuserlocationdefs ^^J%
1328         (define-location-class \string"#2\string"^^J\space\space
1329         \space(:sep \string"{}\glsopenbrace\string" #3
1330         :sep \string"\glsclosebrace\string"))
1331       }%
1332     \else
1333       \edef\@xdyuserlocationdefs{%
1334         \@xdyuserlocationdefs ^^J%
1335         (define-location-class \string"#2\string"^^J\space\space
1336         \space(:sep "\glsopenbrace"
1337         #1
```



```

1338             :sep "\glsclosebrace\glsopenbrace" #3
1339             :sep "\glsclosebrace"))
1340     }%
1341 \fi

1342 \edef\@xdyuserlocationnames{%
1343     \@xdyuserlocationnames^^J\space\space\space
1344     \string"#2\string"}%
1345 }

```

Only has an effect before \writeist:

```

1346 \@onlypremakeg\GlsAddXdyLocation
1347 \else
1348 \newcommand*\GlsAddXdyLocation[2]{%
1349     \glsnoxindywarning\GlsAddXdyLocation}
1350 \fi

```

ationclassorder Define location class order

```

1351 \ifglxsindy
1352 \def\@xdylocationclassorder{^^J\space\space\space
1353     \string"roman-page-numbers\string"^^J\space\space\space
1354     \string"arabic-page-numbers\string"^^J\space\space\space
1355     \string"arabic-section-numbers\string"^^J\space\space\space
1356     \string"alpha-page-numbers\string"^^J\space\space\space
1357     \string"Roman-page-numbers\string"^^J\space\space\space
1358     \string"Alpha-page-numbers\string"^^J\space\space\space
1359     \string"Appendix-page-numbers\string"
1360     \@xdyuserlocationnames^^J\space\space\space
1361     \string"see\string"
1362 }
1363 \fi

```

Change the location order.

ationClassOrder

```

1364 \ifglxsindy
1365 \newcommand*\GlsSetXdyLocationClassOrder[1]{%
1366     \def\@xdylocationclassorder{#1}}
1367 \else
1368 \newcommand*\GlsSetXdyLocationClassOrder[1]{%
1369     \glsnoxindywarning\GlsSetXdyLocationClassOrder}
1370 \fi

```

\@xdysortrules Define sort rules

```

1371 \ifglxsindy
1372 \def\@xdysortrules{}
1373 \fi

```

\GlsAddSortRule Add a sort rule

```

1374 \ifglxindy
1375   \newcommand*\GlsAddSortRule[2]{%
1376     \expandafter\toks@\expandafter{\@xdysortrules}%
1377     \protected@edef\@xdysortrules{\the\toks@ ^^J
1378       (sort-rule \string"#1\string" \string"#2\string")}%
1379   }
1380 \else
1381   \newcommand*\GlsAddSortRule[2]{%
1382     \glsnxindywarning\GlsAddSortRule}
1383 \fi

```

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

```

1384 \ifglxindy
1385   \def\@xdyrequiredstyles{tex}
1386 \fi

```

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

```

1387 \ifglxindy
1388   \newcommand*\GlsAddXdyStyle[1]{%
1389     \edef\@xdyrequiredstyles{\@xdyrequiredstyles,#1}}%
1390 \else
1391   \newcommand*\GlsAddXdyStyle[1]{%
1392     \glsnxindywarning\GlsAddXdyStyle}
1393 \fi

```

**GlsSetXdyStyles** Reset the list of required styles

```

1394 \ifglxindy
1395   \newcommand*\GlsSetXdyStyles[1]{%
1396     \edef\@xdyrequiredstyles{#1}}
1397 \else
1398   \newcommand*\GlsSetXdyStyles[1]{%
1399     \glsnxindywarning\GlsSetXdyStyles}
1400 \fi

```

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

```

1401 \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.

```

1402 \def\@xdylanguage#1#2{}

```

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

```

1403 \ifglxindy
1404   \newcommand*\GlsSetXdyLanguage[2][\glsdefaultttype]{%
1405     \ifglossaryexists{#1}{%
1406       \expandafter\def\csname @xdy@#1@language\endcsname{#2}%
1407     }{%
1408       \PackageError{glossaries}{Can't set language type for
1409         glossary type '#1' --- no such glossary}{%
1410         You have specified a glossary type that doesn't exist}}
1411 \else
1412   \newcommand*\GlsSetXdyLanguage[2][]{%
1413     \glsnoxywarning\GlsSetXdyLanguage}
1414 \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.

```

1415 \def\@gls@codepage#1#2{}

```

`sSetXdyCodePage` Define command to set the code page.

```

1416 \ifglxindy
1417   \newcommand*\GlsSetXdyCodePage[1]{%
1418     \renewcommand*\@gls@codepage{#1}%
1419   }

```

Suggested by egreg:

```

1420 \AtBeginDocument{%
1421   \ifx\gls@codepage\@empty
1422     \@ifpackageloaded{fontspec}{\def\gls@codepage{utf8}}{}%
1423   \fi
1424 }
1425 \else
1426   \newcommand*\GlsSetXdyCodePage[1]{%
1427     \glsnoxywarning\GlsSetXdyCodePage}
1428 \fi

```

`xdylettergroups` Store letter group definitions.

```

1429 \ifglxindy
1430   \ifglx@xindy@glsnumbers
1431     \def\@xdylettergroups{(define-letter-group
1432       \string"glxnumbers\string"^^J\space\space\space
1433       :prefixes (\string"0\string" \string"1\string"
1434       \string"2\string" \string"3\string" \string"4\string"
1435       \string"5\string" \string"6\string" \string"7\string"
1436       \string"8\string" \string"9\string")^^J\space\space\space
1437       \@xdynumbergrouporder)}
1438   \else
1439     \def\@xdylettergroups{}
1440   \fi
1441 \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.

```
1442 \newcommand*\GlsAddLetterGroup[2]{%
1443   \expandafter\toks@\expandafter{\@xdylettergroups}%
1444   \protected@edef\@xdylettergroups{\the\toks@^^J%
1445     (define-letter-group \string"#1\string"^^J\space\space\space#2)}%
1446   }%
```

## 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.

```
1447 \newcommand*\forallglossaries[3][\@glo@types]{%
1448   \@for#2:=#1\do{\ifx#2\@empty\else#3\fi}%
1449 }
```

`\forallacronyms`

```
1450 \newcommand*\forallacronyms[2]{%
1451   \@for#1:=\@glsacronymlists\do{\ifx#1\@empty\else#2\fi}%
1452 }
```

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

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

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

```
1453 \newcommand*\forglentries[3][\glsdefaulttype]{%
1454   \edef\@glo@list{\csname glolist@#1\endcsname}%
1455   \@for#2:=\@glo@list\do
1456   {%
1457     \ifdefempty{#2}{\fi}%
1458   }%
1459 }
```

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

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

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

```
1460 \newcommand*\forallglentries[3][\@glo@types]{%
```

```

1461 \expandafter\foralllglossaries\expandafter[#1]{\@@this@glo@}%
1462 {%
1463   \forallgsentries[\@@this@glo@]{#2}{#3}%
1464 }%
1465 }

```

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

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

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

```

1466 \newcommand{\ifglossaryexists}[3]{%
1467   \ifcsundef{@glo@#1@out}{#3}{#2}%
1468 }

```

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`

```
1469 \newcommand*{\glsdetoklabel}[1]{#1}
```

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

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

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

```

1470 \newcommand{\ifglsentryexists}[3]{%
1471   \ifcsundef{glo@\glsdetoklabel{#1}@name}{#3}{#2}%
1472 }

```

`\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>*.

```

1473 \newcommand*{\ifglsused}[3]{%
1474   \ifbool{glo@\glsdetoklabel{#1}@flag}{#2}{#3}%
1475 }

```

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 exists, otherwise do *<code>*.

```
1476 \newcommand{\glsdoifexists}[2]{%
1477   \ifglentryexists{#1}{#2}{%
1478     \PackageError{glossaries}{Glossary entry ‘\glsdetoklabel{#1}’
1479     has not been defined}{You need to define a glossary entry before you
1480     can use it.}}%
1481 }
```

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

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

```
1482 \newcommand{\glsdoifnoexists}[2]{%
1483   \ifglentryexists{#1}{#2}{%
1484     \PackageError{glossaries}{Glossary entry ‘\glsdetoklabel{#1}’ has already
1485     been defined}{}}{#2}%
1486 }
```

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

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

```
1487 \newcommand{\glsdoifexistsorwarn}[2]{%
1488   \ifglentryexists{#1}{#2}{%
1489     \GlossariesWarning{Glossary entry ‘\glsdetoklabel{#1}’
1490     has not been defined}%
1491   }%
1492 }
```

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

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

```
1493 \newcommand{\glsdoifexistsordo}[3]{%
1494   \ifglentryexists{#1}{#2}{%
1495     \PackageError{glossaries}{Glossary entry ‘\glsdetoklabel{#1}’
1496     has not been defined}{You need to define a glossary entry before you
1497     can use it.}}%
1498   #3%
1499 }%
1500 }
```

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

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

```

1501 \newcommand{\doifglossarynoexistsordo}[3]{%
1502   \ifglossaryexists{#1}%
1503   {%
1504     \PackageError{glossaries}{Glossary type ‘#1’ already exists}{}%
1505     #3%
1506   }%
1507   {#2}%
1508 }
```

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

```

1509 \newcommand{\ifglshaschildren}[3]{%
1510   \glstoifexists{#1}%
1511   {%
1512     \def\do@glshaschildren{#3}%
1513     \edef\@gls@thislabel{\glstoiflabel{#1}}%
1514     \expandafter\for@glstentries\expandafter
1515       [\csname glo@\@gls@thislabel @type\endcsname]
1516     {\glo@label}%
1517     {%
1518       \letcs\glo@parent{glo@\glo@label @parent}%
1519       \ifdefequal\@gls@thislabel\glo@parent
1520       {%
1521         \def\do@glshaschildren{#2}%
1522         \@endfortrue
1523       }%
1524     }%
1525   }%
1526   \do@glshaschildren
1527 }%
1528 }
```

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

```

1529 \newcommand{\ifglshasparent}[3]{%
1530   \glstoifexists{#1}%
1531   {%
1532     \ifcempty{glo@\glstoiflabel{#1}@parent}{#3}{#2}%
1533   }%
1534 }
```

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

```

1535 \newcommand*\ifglshasdesc[3]{%
```

```

1536 \ifcseempty{glo@\glsdetoklabel{#1}@desc}%
1537 {#3}%
1538 {#2}%
1539 }

```

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

```

1540 \newcommand*{\ifglsdesccsuppressed}[3]{%
1541 \ifcsequal{glo@\glsdetoklabel{#1}@desc}{@no@post@desc}%
1542 {#2}%
1543 {#3}%
1544 }

```

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

```

1545 \newcommand*{\ifglshassymbol}[3]{%
1546 \letcs{\@glo@symbol}{glo@\glsdetoklabel{#1}@symbol}%
1547 \ifdefempty\@glo@symbol
1548 {#3}%
1549 {%
1550 \ifdefequal\@glo@symbol\@gls@default@value
1551 {#3}%
1552 {#2}%
1553 }%
1554 }

```

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

```

1555 \newcommand*{\ifglshaslong}[3]{%
1556 \letcs{\@glo@long}{glo@\glsdetoklabel{#1}@long}%
1557 \ifdefempty\@glo@long
1558 {#3}%
1559 {%
1560 \ifdefequal\@glo@long\@gls@default@value
1561 {#3}%
1562 {#2}%
1563 }%
1564 }

```

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

```

1565 \newcommand*{\ifglshasshort}[3]{%
1566 \letcs{\@glo@short}{glo@\glsdetoklabel{#1}@short}%
1567 \ifdefempty\@glo@short
1568 {#3}%
1569 {%
1570 \ifdefequal\@glo@short\@gls@default@value
1571 {#3}%
1572 {#2}%
1573 }%
1574 }

```



<code>\ifglshasfield</code>	<code>\ifglshasfield{&lt;field&gt;}{&lt;label&gt;}{&lt;true part&gt;}{&lt;false part&gt;}</code>
-----------------------------	--

```

1575 \newcommand*{\ifglshasfield}[4]{%
1576   \glstoifexists{#2}%
1577   {%
1578     \letcs{\@glo@thisvalue}{glo\glstetoklabel{#2}@#1}%

```

First check supplied field label is defined.

```

1579   \ifdef\@glo@thisvalue
1580   {%

```

Is defined, so now check if empty.

```

1581     \ifdefempty\@glo@thisvalue
1582     {%

```

Is empty, so doesn't have field set.

```

1583         #4%
1584     }%
1585   {%

```

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

```

1586     \ifdefequal\@glo@thisvalue\@gls@default@value
1587     {%

```

Value is set to the default value.

```

1588         #4%
1589     }%
1590   {%

```

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

```

1591     \let\glscurrentfieldvalue\@glo@thisvalue
1592     #3%
1593   }%
1594 }%
1595 }%
1596 {%

```

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

```

1597   \@gls@fetchfield{\@gls@thisfield}{#1}%

```

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

```

1598   \ifdef\@gls@thisfield
1599   {%

```

Is defined, so now check if empty.

```

1600     \letcs{\@glo@thisvalue}{glo\glstetoklabel{#2}@\@gls@thisfield}%
1601     \ifdefempty\@glo@thisvalue
1602     {%

```

Is empty so field hasn't been set.

```

1603         #4%

```

```

1604         }%
1605         {%

    Isn't empty so check if it's been set to \@gls@default@value.
1606         \ifdefequal\@glo@thisvalue\@gls@default@value
1607         {%

    Value is set to the default value.
1608         #4%
1609         }%
1610         {%

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

1611         \let\glscurrentfieldvalue\@glo@thisvalue
1612         #3%
1613         }%
1614         }%
1615     }%
1616     {%

    Not defined.
1617     \GlossariesWarning{Unknown entry field '#1'}%
1618     #4%
1619     }%
1620 }%
1621 }%
1622 }

```

urrentfieldvalue

```

1623 \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

```

1624 \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.

```

1625 \newcommand*\@gls@provide@newglossary{%
1626   \protected@write\@auxout{}{\string\providecommand\string\@newglossary[4]{} }%

```

Only need to do this once.

```

1627   \let\@gls@provide@newglossary\relax
1628 }

```

```

\defglentryfmt Allow different glossaries to have different display styles.
1629 \newcommand*\defglentryfmt}[2][\gldefaulttype]{%
1630   \csgdef{gls@#1@entryfmt}{#2}%
1631 }

\gls@doentryfmt
1632 \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.
1633 \newcommand*\@gls@forbidtexext}[1]{%
1634   \ifboolexpr{test {\ifdefstring{#1}{tex}}
1635     or test {\ifdefstring{#1}{TEX}}}
1636   {%
1637     \def#1{nottex}%
1638     \PackageError{glossaries}%
1639       {Forbidden '.tex' extension replaced with '.nottex'}%
1640       {I'm sorry, I can't allow you to do something so reckless.\MessageBreak
1641         Don't use '.tex' as an extension for a temporary file.}%
1642   }%
1643   {%
1644   }%
1645 }

\gls@gobbleopt Discard optional argument.
1646 \newcommand*\gls@gobbleopt{\new@ifnextchar[\@gls@gobbleopt]}
1647 \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`

```
1648 \newcommand*\newglossary{\@ifstar\s@newglossary\ns@newglossary}
```

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

```

1649 \newcommand*\s@newglossary}[2]{%
1650   \ns@newglossary[#1-glg]{#1}{#1-gls}{#1-glo}{#2}%
1651 }

```

`\ns@newglossary` Define the unstarred version.

```
1652 \newcommand*{\ns@newglossary}[5][glg]{%
1653   \doifglossarynoexistsordo{#2}%
1654   {%
```

Check if default has been set

```
1655   \ifundef\glsdefaultttype
1656   {%
1657     \gdef\glsdefaultttype{#2}%
1658   }{}%
```

Add this to the list of glossary types:

```
1659   \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.

```
1660   \expandafter\gdef\csname glolist@#2\endcsname{,}%
```

Store the file extensions:

```
1661   \expandafter\edef\csname @glotype@#2@log\endcsname{#1}%
1662   \expandafter\edef\csname @glotype@#2@in\endcsname{#3}%
1663   \expandafter\edef\csname @glotype@#2@out\endcsname{#4}%
1664   \expandafter\@gls@forbidtexext\csname @glotype@#2@log\endcsname
1665   \expandafter\@gls@forbidtexext\csname @glotype@#2@in\endcsname
1666   \expandafter\@gls@forbidtexext\csname @glotype@#2@out\endcsname
```

Store the title:

```
1667   \expandafter\def\csname @glotype@#2@title\endcsname{#5}%
```

```
1668   \@gls@provide@newglossary
```

```
1669   \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.

```
1670   \ifcsundef{gls@#2@entryfmt}%
1671   {%
1672     \defglsentryfmt[#2]{\glsentryfmt}%
1673   }%
1674   {}%
```

Define sort counter if required:

```
1675   \@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.)

```
1676   \@ifnextchar[{\@gls@setcounter{#2}}%
1677   {\@gls@setcounter{#2}[\glscounter]}%
1678   }%
1679   {%
1680   \gls@gobbleopt
```

```

1681 }%
1682 }

```

`\altnewglossary`

```

1683 \newcommand*{\altnewglossary}[3]{%
1684   \newglossary[#2-glg]{#1}{#2-gls}{#2-glo}{#3}%
1685 }

```

Only define new glossaries in the preamble:

```

1686 \@onlypreamble{\newglossary}

```

Only define new glossaries before `\makeglossaries`

```

1687 \@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  $\text{\LaTeX}$ , `\@newglossary` simply ignores its arguments.

`\@newglossary`

```

1688 \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`

```

1689 \def\@gls@setcounter#1[#2]{%
1690   \expandafter\def\csname @glotype@#1@counter\endcsname{#2}%

```

Add counter to xindy list, if not already added:

```

1691   \ifglsxindy
1692     \GlsAddXdyCounters{#2}%
1693   \fi
1694 }

```

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

`@gls@getcounter`

```

1695 \newcommand*{\@gls@getcounter}[1]{%
1696   \csname @glotype@#1@counter\endcsname
1697 }

```

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

```

1698 \glsdefmain

```

Define the “acronym” glossaries if required.

```

1699 \@gls@do@acronymsdef

```

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

```

1700 \@gls@do@symbolsdef
1701 \@gls@do@numbersdef
1702 \@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.

```

1703 \newcommand*\newignoredglossary}[1]{%
1704   \ifdefempty\@ignored@glossaries
1705   {%
1706     \edef\@ignored@glossaries{#1}%
1707   }%
1708   {%
1709     \eappto\@ignored@glossaries{, #1}%
1710   }%
1711   \csgdef{glolist@#1}{,}%
1712   \ifcsundef{gls@#1@entryfmt}%
1713   {%
1714     \defglentryfmt[#1]{\glentryfmt}%
1715   }%
1716   {}%
1717   \ifdefempty\@gls@nohyperlist
1718   {%
1719     \renewcommand*\@gls@nohyperlist}{#1}%
1720   }%
1721   {%
1722     \eappto\@gls@nohyperlist{, #1}%
1723   }%
1724 }
```

`ignored@glossaries` List of ignored glossaries.

```

1725 \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.

```

1726 \newcommand*\ifignoredglossary}[3]{%
1727   \edef\@gls@igtype{#1}%
1728   \expandafter\DTLifinlist\expandafter
1729   {\@gls@igtype}{\@ignored@glossaries}{#2}{#3}%
1730 }
```

## 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.

```
1731 \define@key{glossentry}{name}{%  
1732 \def\@glo@name{#1}%  
1733 }
```

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

```
1734 \define@key{glossentry}{description}{%  
1735 \def\@glo@desc{#1}%  
1736 }
```

**descriptionplural**

```
1737 \define@key{glossentry}{descriptionplural}{%  
1738 \def\@glo@descplural{#1}%  
1739 }
```

**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`.

```
1740 \define@key{glossentry}{sort}{%  
1741 \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.

```
1742 \define@key{glossentry}{text}{%  
1743 \def\@glo@text{#1}%  
1744 }
```

**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.

```
1745 \define@key{glossentry}{plural}{%  
1746 \def\@glo@plural{#1}%  
1747 }
```

**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.

```
1748 \define@key{glossentry}{first}{%  
1749 \def\@glo@first{#1}%  
1750 }
```

**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.

```

1751 \define@key{glossentry}{firstplural}{%
1752 \def\@glo@firstplural{#1}%
1753 }

```

s@default@value

```

1754 \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).

```

1755 \define@key{glossentry}{symbol}{%
1756 \def\@glo@symbol{#1}%
1757 }

```

symbolplural

```

1758 \define@key{glossentry}{symbolplural}{%
1759 \def\@glo@symbolplural{#1}%
1760 }

```

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

```

1761 \define@key{glossentry}{type}{%
1762 \def\@glo@type{#1}}

```

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

```

1763 \define@key{glossentry}{counter}{%
1764   \ifcsundef{c@#1}%
1765   {%
1766     \PackageError{glossaries}%
1767     {There is no counter called ‘#1’}%
1768     {%
1769       The counter key should have the name of a valid counter
1770       as its value%
1771     }%
1772   }%
1773   {%
1774     \def\@glo@counter{#1}%
1775   }%
1776 }

```

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

```

1777 \define@key{glossentry}{see}{%
1778   \gls@set@xr@key{see}{\@glo@see}{#1}%
1779 }

```



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

Assign a cross-reference key.

```
1780 \newcommand*{\gls@set@xr@key}[3]{%
1781   \renewcommand*{\gls@xr@key}{#1}%
1782   \gls@checkseeallowed
1783   \def#2{#3}%
1784   \@glo@seeautonumberlist
1785 }
```

`\gls@xr@key`

```
1786 \newcommand*{\gls@xr@key}{see}
```

`checkseeallowed`

```
1787 \newcommand*{\gls@checkseeallowed}{%
1788   \@gls@see@noindex
1789 }
```

`ed@preambleonly`

```
1790 \newcommand*{\gls@checkseeallowed@preambleonly}{%
1791   \GlossariesWarning{glossaries}%
1792   {'\gls@xr@key' key doesn't have any effect when used in the document
1793   environment. Move the definition to the preamble
1794   after \string\makeglossaries\space
1795   or \string\makenoidxglossaries}%
1796 }
```

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

```
1797 \define@key{glossentry}{parent}{%
1798 \def\@glo@parent{#1}}
```

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

```
1799 \define@choicekey{glossentry}{nonumberlist}[\val\nr]{true,false}[true]{%
1800   \ifcase\nr\relax
1801     \def\@glo@prefix{\glsnonextpages}%
1802     \@gls@savenonumberlist{true}%
1803   \else
1804     \def\@glo@prefix{\glsnextpages}%
1805     \@gls@savenonumberlist{false}%
1806   \fi
1807 }
```

`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.

```
1808 \newcommand*{\@gls@savenonumberlist}[1]{}
```

nitnonnumberlist

```
1809 \newcommand*{\@gls@initnonnumberlist}{}%
```

nitnonnumberlist

```
1810 \newcommand*{\@gls@storenonnumberlist}[1]{}
```

savenonnumberlist Allow the nonnumberlist value to be saved.

```
1811 \newcommand*{\@gls@enablesavenonnumberlist}{%
1812   \renewcommand*{\@gls@initnonnumberlist}{%
1813     \undef\@glo@nonnumberlist
1814   }%
1815   \renewcommand*{\@gls@savenonnumberlist}[1]{%
1816     \def\@glo@nonnumberlist{##1}%
1817   }%
1818   \renewcommand*{\@gls@storenonnumberlist}[1]{%
1819     \ifdef\@glo@nonnumberlist
1820     {%
1821       \cslet{glo@glstetoklabel{##1}@nonnumberlist}{\@glo@nonnumberlist}%
1822     }%
1823     {}%
1824   }%
1825   \appto\@gls@keymap{,{nonnumberlist}{nonnumberlist}}%
1826 }
```

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

user1

```
1827 \define@key{glossentry}{user1}{%
1828   \def\@glo@useri{#1}%
1829 }
```

user2

```
1830 \define@key{glossentry}{user2}{%
1831   \def\@glo@userii{#1}%
1832 }
```

user3

```
1833 \define@key{glossentry}{user3}{%
1834   \def\@glo@useriii{#1}%
1835 }
```

user4

```
1836 \define@key{glossentry}{user4}{%
1837   \def\@glo@useriv{#1}%
1838 }
```

user5

```
1839 \define@key{glossentry}{user5}{%
1840   \def\@glo@userv{#1}%
1841 }
```

user6

```
1842 \define@key{glossentry}{user6}{%  
1843   \def\@glo@user6{#1}%  
1844 }
```

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.

```
1845 \define@key{glossentry}{short}{%  
1846   \def\@glo@short{#1}%  
1847 }
```

shortplural This key is provided for use by \newacronym.

```
1848 \define@key{glossentry}{shortplural}{%  
1849   \def\@glo@shortpl{#1}%  
1850 }
```

long This key is provided for use by \newacronym.

```
1851 \define@key{glossentry}{long}{%  
1852   \def\@glo@long{#1}%  
1853 }
```

longplural This key is provided for use by \newacronym.

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

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

```
1857 \newcommand*\@glsnname{%  
1858   \PackageError{glossaries}{name key required in  
1859   \string\newglossaryentry\space for entry '\@glo@label'}{You  
1860   haven't specified the entry name}}
```

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

```
1861 \newcommand*\@glsnodelsc{%  
1862   \PackageError{glossaries}  
1863   {%  
1864     description key required in \string\newglossaryentry\space  
1865     for entry '\@glo@label'%  
1866   }%  
1867   {%  
1868     You haven't specified the entry description%  
1869   }%  
1870 }
```

lsdefaultplural Now obsolete. Don't use.

```
1871 \newcommand*\@glsdefaultplural{}
```

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

```
1872 \newcommand*{\@gls@missingnumberlist}[1]{%
1873   ??%
1874   \ifglssavenumberlist
1875     \GlossariesWarning{Missing number list for entry ‘#1’.
1876       Maybe makeglossaries + rerun required}%
1877   \else
1878     \PackageError{glossaries}%
1879     {Package option ‘savenumberlist=true’ required}%
1880     {%
1881       You must use the ‘savenumberlist’ package option
1882       to reference location lists.%
1883     }%
1884   \fi
1885 }
```

`\@gls@defaultsort` Define command to set default sort.

```
1886 \newcommand*{\@gls@defaultsort}{\@glo@name}
```

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

```
1887 \newcount\gls@level
```

`\@noexpand@field`

```
1888 \newcommand{\@@gls@noexpand@field}[3]{%
1889   \expandafter\global\expandafter
1890     \let\csname glo@#1@#2\endcsname#3%
1891 }
```

`\noexpand@fields`

```
1892 \newcommand{\@gls@noexpand@fields}[4]{%
1893   \ifcsdef{gls@assign@#3@field}
1894     {%
1895       \ifdefequal{#4}{\@gls@default@value}%
1896       {%
1897         \edef\@gls@value{\expandonce{#1}}%
1898         \csuse{gls@assign@#3@field}{#2}{\@gls@value}%
1899       }%
1900     }%
1901     \csuse{gls@assign@#3@field}{#2}{#4}%
1902   }%
1903 }%
1904 {%
1905   \ifdefequal{#4}{\@gls@default@value}%
1906   {%
1907     \edef\@gls@value{\expandonce{#1}}%
1908     \@@gls@noexpand@field{#2}{#3}{\@gls@value}%
1909   }%
1910   {%
```

```

1911      \@@gls@noexpand@field{#2}{#3}{#4}%
1912    }%
1913  }%
1914 }

```

ls@expand@field

```

1915 \newcommand{\@@gls@expand@field}[3]{%
1916   \expandafter
1917   \protected@xdef\csname glo@#1@#2\endcsname{#3}%
1918 }

```

s@expand@fields

```

1919 \newcommand{\@gls@expand@fields}[4]{%
1920   \ifcsdef{gls@assign@#3@field}
1921   {%
1922     \ifdefequal{#4}{\@gls@default@value}%
1923     {%
1924       \edef\@gls@value{\expandonce{#1}}%
1925       \csuse{gls@assign@#3@field}{#2}{\@gls@value}%
1926     }%
1927     {%
1928       \expandafter\@gls@startswithexpandonce#4\relax\relax@gls@endcheck
1929       {%
1930         \@@gls@expand@field{#2}{#3}{#4}%
1931       }%
1932       {%
1933         \csuse{gls@assign@#3@field}{#2}{#4}%
1934       }%
1935     }%
1936   }%
1937   {%
1938     \ifdefequal{#4}{\@gls@default@value}%
1939     {%
1940       \@@gls@expand@field{#2}{#3}{#1}%
1941     }%
1942     {%
1943       \@@gls@expand@field{#2}{#3}{#4}%
1944     }%
1945   }%
1946 }

```

swithexpandonce

```

1947 \def\@gls@expandonce{\expandonce}
1948 \def\@gls@startswithexpandonce#1#2@gls@endcheck#3#4{%
1949   \def\@gls@tmp{#1}%
1950   \ifdefequal{\@gls@expandonce}{\@gls@tmp}{#3}{#4}%
1951 }

```

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.

```
1952 \let\gls@assign@field\@gls@expand@fields
```

gls@expand@fields

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

```
1953 \newcommand*{\gls@expand@fields}{%
1954   \let\gls@assign@field\@gls@expand@fields
1955 }
```

snoexpand@fields

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

```
1956 \newcommand*{\gls@snoexpand@fields}{%
1957   \let\gls@assign@field\@gls@noexpand@fields
1958 }
```

newglossaryentry

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

```
1959 \newrobustcmd{\newglossaryentry}[2]{%
```

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

```
1960   \glsdoifnoexists{#1}%
1961   {%
1962     \gls@defglossaryentry{#1}{#2}%
1963   }%
1964 }
```

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.

```
1965 \newcommand*{\gls@defdocnewglossaryentry}{%
1966   \let\gls@checkseeallowed\gls@checkseeallowed@preambleonly
1967   \let\newglossaryentry\new@glossaryentry
1968 }
```

deglossaryentry

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

```
1969 \newrobustcmd{\provideglossaryentry}[2]{%
1970   \ifglsentryexists{#1}%
1971   }{%
1972   {%
1973     \gls@defglossaryentry{#1}{#2}%
1974   }%
1975 }
1976 \@onlypreamble{\provideglossaryentry}
```

`w@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  $\TeX$  run. This means that any glossaries at the start of the document can access the entry information.

```

1977 \newrobustcmd{\new@glossaryentry}[2]{%
1978   \ifundef\@gls@deffile
1979   {%
1980     \global\newwrite\@gls@deffile
1981     \immediate\openout\@gls@deffile=\jobname.glsdefs
1982   }%
1983   {}%
1984   \ifglentryexists{#1}{}%
1985   {%
1986     \gls@defglossaryentry{#1}{#2}%
1987   }%
1988   \@gls@writedef{#1}%
1989 }
```

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`.

```

1990 \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.

```

1991 \AtEndDocument{\ifdef\@gls@deffile{\closeout\@gls@deffile}{}}
```

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

```

1992 \newcommand*{\gls@begindocdefs}{%
1993   \@gls@enablesavenonumberlist
1994   \edef\@gls@restoreat{\noexpand\catcode'\noexpand\@=\number\catcode'\@relax}%
1995   \makeatletter
1996   \InputIfFileExists{\jobname.glsdefs}{\relax}{}%
1997   \@gls@restoreat
1998   \undef\@gls@restoreat
1999   \gls@defdocnewglossaryentry
2000 }
```

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

```

2001 \newcommand*{\@gls@writedef}[1]{%
2002   \immediate\write\@gls@deffile
2003   {%
2004     \string\ifglentryexists{#1}{}\glspercentchar^^J%
2005     \expandafter\@gobble\string{\glspercentchar^^J%
2006       \string\gls@defglossaryentry{\glsdetoklabel{#1}}\glspercentchar^^J%
2007       \expandafter\@gobble\string{\glspercentchar%
2008     }%
2009   }
```

Write key value information:

```

2009 \@for\@gls@map:=\@gls@keymap\do
2010 {%
2011   \letcs\glo@value{glo\@glsdetoklabel{#1}\@expandafter\@secondoftwo\@gls@map}%
2012   \ifdef\glo@value
2013   {%
2014     \@onelevel@sanitize\glo@value
2015     \immediate\write\@gls@deffile
2016     {%
2017       \expandafter\@firstoftwo\@gls@map
2018       =\expandafter\@gobble\string\{\glo@value\expandafter\@gobble\string\},%
2019       \glspercentchar
2020     }%
2021   }%
2022 }%
2023 }%

```

Provide hook:

```

2024 \gls.writedefhook
2025 \immediate\write\@gls@deffile
2026 {%
2027   \glspercentchar^^J%
2028   \expandafter\@gobble\string\}\glspercentchar^^J%
2029   \expandafter\@gobble\string\}\glspercentchar%
2030 }%
2031 }

```

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

```

2032 \newcommand*{\@gls@keymap}{%
2033   {name}{name},%
2034   {sort}{sortvalue},% unescaped sort value
2035   {type}{type},%
2036   {first}{first},%
2037   {firstplural}{firstpl},%
2038   {text}{text},%
2039   {plural}{plural},%
2040   {description}{desc},%
2041   {descriptionplural}{descplural},%
2042   {symbol}{symbol},%
2043   {symbolplural}{symbolplural},%
2044   {user1}{useri},%
2045   {user2}{userii},%
2046   {user3}{useriii},%
2047   {user4}{useriv},%
2048   {user5}{userv},%
2049   {user6}{uservi},%
2050   {long}{long},%
2051   {longplural}{longpl},%
2052   {short}{short},%
2053   {shortplural}{shortpl},%

```



```

2054 {counter}{counter},%
2055 {parent}{parent}%
2056 }

```

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

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

```

2057 \newcommand*{\@gls@fetchfield}[2]{%
    Ensure user field name is fully expanded
2058 \edef\@gls@thisval{#2}%
    Iterate through known mappings until we find the one for this field.
2059 \for\@gls@map:=\@gls@keymap\do{%
2060 \edef\@this@key{\expandafter\@firstoftwo\@gls@map}%
2061 \ifdefequal{\@this@key}{\@gls@thisval}%
2062 {%
    Found it.
2063 \edef#1{\expandafter\@secondoftwo\@gls@map}%
    Break out of loop.
2064 \@endfortrue
2065 }%
2066 {}%
2067 }%
2068 }

```

`\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.

```

2069 \newcommand*{\glsaddstoragekey}{\@ifstar\@sglsaddstoragekey\@glsaddstoragekey}
    Starred version switches on expansion for this key.
2070 \newcommand*{\@sglsaddstoragekey}[1]{%
2071 \key@ifundefined{glossentry}{#1}%
2072 {%
2073 \expandafter\newcommand\expandafter*\expandafter
2074 {\csname gls@assign@#1@field\endcsname}[2]{%
2075 \@gls@expand@field{##1}{#1}{##2}%
2076 }%
2077 }%
2078 {}%
2079 \@glsaddstoragekey{#1}%
2080 }

```

Unstarred version doesn't override default expansion.

```

2081 \newcommand*{\@glsaddstoragekey}[3]{%

```

Check the specified key doesn't already exist.

```
2082 \key@ifundefined{glossentry}{#1}%  
2083 {%
```

Set up the key.

```
2084 \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%  
2085 \appto\@gls@keymap{,{#1}{#1}}%
```

Set the default value.

```
2086 \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
```

Assignment code.

```
2087 \appto\@newglossaryentryposthook{%  
2088 \letcs{\@glo@tmp}{@glo@#1}%  
2089 \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%  
2090 }%
```

Define the no-link commands.

```
2091 \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%  
2092 }%  
2093 {%  
2094 \PackageError{glossaries}{Key ‘#1’ already exists}{}%  
2095 }%  
2096 }
```

```
\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.

```
2097 \newcommand*{\glsaddkey}{\@ifstar\@sglsaddkey\@glsaddkey}
```

Starred version switches on expansion for this key.

```
2098 \newcommand*{\@sglsaddkey}[1]{%  
2099 \key@ifundefined{glossentry}{#1}%  
2100 {%  
2101 \expandafter\newcommand\expandafter*\expandafter  
2102 {\csname gls@assign@#1@field\endcsname}[2]{%  
2103 \@gls@expand@field{##1}{#1}{##2}%  
2104 }%  
2105 }%  
2106 {}%  
2107 \@glsaddkey{#1}%  
2108 }
```

Unstarred version doesn't override default expansion.

```
2109 \newcommand*{\@glsaddkey}[7]{%
```

Check the specified key doesn't already exist.

```
2110 \key@ifundefined{glossentry}{#1}%  
2111 {%
```

Set up the key.

```
2112 \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
2113 \appto\@gls@keymap{, {#1}{#1}}%
```

Set the default value.

```
2114 \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
```

Assignment code.

```
2115 \appto\@newglossaryentryposthook{%
2116 \letcs{\@glo@tmp}{@glo@#1}%
2117 \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
2118 }%
```

Define the no-link commands.

```
2119 \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
2120 \newcommand*{#4}[1]{\@Gls@entry@field{##1}{#1}}%
```

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

```
2121 \ifcsdef{@gls@user@#1@}%
2122 {%
2123 \PackageError{glossaries}%
2124 {Can't define '\string#5' as helper command
2125 '\expandafter\string\csname @gls@user@#1@\endcsname' already exists}%
2126 }%
2127 }%
2128 {%
2129 \expandafter\newcommand\expandafter*\expandafter
2130 {\csname @gls@user@#1\endcsname}[2][1]{%
2131 \new@ifnextchar[%
2132 {\csuse{@gls@user@#1@}{##1}{##2}}%
2133 {\csuse{@gls@user@#1@}{##1}{##2}[1]}%
2134 \csdef{@gls@user@#1@}##1##2[##3]{%
2135 \@gls@field@link{##1}{##2}{#3{##2}##3}%
2136 }%
2137 \newrobustcmd*{#5}{%
2138 \expandafter\@gls@hyp@opt\csname @gls@user@#1\endcsname}%
2139 }%
```

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

```
2140 \ifcsdef{@Gls@user@#1@}%
2141 {%
2142 \PackageError{glossaries}%
2143 {Can't define '\string#6' as helper command
2144 '\expandafter\string\csname @Gls@user@#1@\endcsname' already exists}%
2145 }%
2146 }%
2147 {%
2148 \expandafter\newcommand\expandafter*\expandafter
2149 {\csname @Gls@user@#1\endcsname}[2][1]{%
```

```

2150         \new@ifnextchar[%
2151         {\csuse{@Gls@user@#1@}{##1}{##2}}}%
2152         {\csuse{@Gls@user@#1@}{##1}{##2} []}}}%
2153     \csdef{@Gls@user@#1@}##1##2[##3]{%
2154         \@gls@field@link{##1}{##2}{#4{##2}##3}%
2155     }%
2156     \newrobustcmd*{#6}{%
2157         \expandafter\@gls@hyp@opt\csname @Gls@user@#1\endcsname}%
2158     }%

```

Finally the all caps version:

```

2159     \ifcsdef{@GLS@user@#1@}%
2160     {%
2161         \PackageError{glossaries}%
2162         {Can't define '\string#7' as helper command
2163         '\expandafter\string\csname @GLS@user@#1@\endcsname' already exists}%
2164         {}%
2165     }%
2166     {%
2167         \expandafter\newcommand\expandafter*\expandafter
2168         {\csname @GLS@user@#1\endcsname}[2] []{%
2169             \new@ifnextchar[%
2170             {\csuse{@GLS@user@#1@}{##1}{##2}}}%
2171             {\csuse{@GLS@user@#1@}{##1}{##2} []}}}%
2172         \csdef{@GLS@user@#1@}##1##2[##3]{%
2173             \@gls@field@link{##1}{##2}{\mfirstucMakeUppercase{#3{##2}##3}}}%
2174         }%
2175         \newrobustcmd*{#7}{%
2176             \expandafter\@gls@hyp@opt\csname @GLS@user@#1\endcsname}%
2177         }%
2178     }%
2179     {%
2180         \PackageError{glossaries}{Key '#1' already exists}{}%
2181     }%
2182 }

```

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

```

2183 \newcommand{\glsfieldxdef}[3]{%
2184     \glsdoifexists{#1}%
2185     {%
2186         \edef\@glo@label{\glsdetoklabel{#1}}%
2187         \ifcsdef{glo@\@glo@label @#2}%
2188         {%
2189             \expandafter\xdef\csname glo@\@glo@label @#2\endcsname{#3}%
2190         }%
2191         {%

```

```

2192     \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2193 }%
2194 }%
2195 }

```

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

```

2196 \newcommand{\glsfielddedf}[3]{%
2197   \glsdoifexists{#1}%
2198   {%
2199     \edef\@glo@label{\glsdetoklabel{#1}}%
2200     \ifcsdef{glo@\@glo@label @#2}%
2201     {%
2202       \expandafter\edef\csname glo@\@glo@label @#2\endcsname{#3}%
2203     }%
2204     {%
2205       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2206     }%
2207   }%
2208 }

```

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

```

2209 \newcommand{\glsfielddgdef}[3]{%
2210   \glsdoifexists{#1}%
2211   {%
2212     \edef\@glo@label{\glsdetoklabel{#1}}%
2213     \ifcsdef{glo@\@glo@label @#2}%
2214     {%
2215       \expandafter\gdef\csname glo@\@glo@label @#2\endcsname{#3}%
2216     }%
2217     {%
2218       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2219     }%
2220   }%
2221 }

```

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

```

2222 \newcommand{\glsfieldddef}[3]{%
2223   \glsdoifexists{#1}%
2224   {%
2225     \edef\@glo@label{\glsdetoklabel{#1}}%

```

```

2226 \ifcsdef{glo@\@glo@label @#2}%
2227 {%
2228   \expandafter\def\csname glo@\@glo@label @#2\endcsname{#3}%
2229 }%
2230 {%
2231   \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2232 }%
2233 }%
2234 }

```

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

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

```

2235 \newcommand{\glsfieldfetch}[3]{%
2236   \glsdoifexists{#1}%
2237   {%
2238     \edef\@glo@label{\glsdetoklabel{#1}}%
2239     \ifcsdef{glo@\@glo@label @#2}%
2240     {%
2241       \letcs#3{glo@\@glo@label @#2}%
2242     }%
2243   }%
2244   \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2245 }%
2246 }%
2247 }

```

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

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

```

2248 \newcommand{\ifglsfieldeq}[5]{%
2249   \glsdoifexists{#1}%
2250   {%
2251     \edef\@glo@label{\glsdetoklabel{#1}}%
2252     \ifcsdef{glo@\@glo@label @#2}%
2253     {%
2254       \ifcsstring{glo@\@glo@label @#2}{#3}{#4}{#5}%
2255     }%
2256   }%
2257   \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2258 }%
2259 }%
2260 }

```

`\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.

```

2261 \newcommand{\ifglsfielddefeq}[5]{%
2262   \glsdoifexists{#1}%
2263   {%
2264     \edef\@glo@label{\glsdetoklabel{#1}}%
2265     \ifcsdef{glo@\@glo@label @#2}%
2266     {%
2267       \expandafter\ifdefstrequal
2268       \csname glo@\@glo@label @#2\endcsname{#3}{#4}{#5}%
2269     }%
2270     {%
2271       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2272     }%
2273   }%
2274 }

```

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

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

```

2275 \newcommand{\ifglsfieldcseq}[5]{%
2276   \glsdoifexists{#1}%
2277   {%
2278     \edef\@glo@label{\glsdetoklabel{#1}}%
2279     \ifcsdef{glo@\@glo@label @#2}%
2280     {%
2281       \ifcsstrequal{glo@\@glo@label @#2}{#3}{#4}{#5}%
2282     }%
2283     {%
2284       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2285     }%
2286   }%
2287 }

```

`glswritedefhook`

```

2288 \newcommand*{\glswritedefhook}{}

```

`gls@assign@desc`

```

2289 \newcommand*{\gls@assign@desc}[1]{%
2290   \gls@assign@field{#1}{desc}{\@glo@desc}%
2291   \gls@assign@field{\@glo@desc}{#1}{descplural}{\@glo@descplural}%
2292 }

```

`ewglossaryentry`

```

2293 \newcommand{\longnewglossaryentry}[3]{%
2294   \glsdoifnoexists{#1}%
2295   {%
2296     \bgroup

```

```

2297 \let\@org@newglossaryentryprehook\@newglossaryentryprehook
2298 \long\def\@newglossaryentryprehook{%
2299 \long\def\@glo@desc{#3\leavevmode\unskip\nopostdesc}%
2300 \@org@newglossaryentryprehook
2301 }%
2302 \renewcommand*\@gls@assign@desc}[1]{%
2303 \global\cslet{glo@\glsdetoklabel{#1}@desc}{\@glo@desc}%
2304 \global\cslet{glo@\glsdetoklabel{#1}@descplural}{\@glo@desc}%
2305 }
2306 \gls@defglossaryentry{#1}{#2}%
2307 \egroup
2308 }
2309 }

```

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

```
2310 \@onlypreamble{\longnewglossaryentry}
```

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

```

2311 \newcommand{\longprovideglossaryentry}[3]{%
2312 \ifglstryexists{#1}{}%
2313 {\longnewglossaryentry{#1}{#2}{#3}}%
2314 }
2315 \@onlypreamble{\longprovideglossaryentry}

```

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

Defines a new entry without checking if it already exists.

```
2316 \newcommand{\gls@defglossaryentry}[2]{%
```

Prevent any further use of `\GlsSetQuote`:

```
2317 \let\GlsSetQuote\gls@nosetquote
```

Store label

```
2318 \edef\@glo@label{\glsdetoklabel{#1}}%
```

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

```
2319 \let\glslabel\@glo@label
```

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

```
2320 \let\@glo@name\@gls@name
```

```
2321 \let\@glo@desc\@gls@desc
```

```
2322 \let\@glo@descplural\@gls@default@value
```

```
2323 \let\@glo@type\@gls@default@value
```

```
2324 \let\@glo@symbol\@gls@default@value
```

```
2325 \let\@glo@symbolplural\@gls@default@value
```

```
2326 \let\@glo@text\@gls@default@value
```



```
2327 \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.)

```
2328 \let\@glo@first\@gls@default@value
```

```
2329 \let\@glo@firstplural\@gls@default@value
```

Set the default sort:

```
2330 \let\@glo@sort\@gls@default@value
```

Set the default counter:

```
2331 \let\@glo@counter\@gls@default@value
```

```
2332 \def\@glo@see{}%
```

```
2333 \def\@glo@parent{}%
```

```
2334 \def\@glo@prefix{}%
```

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

```
2335 \@gls@initnonnumberlist
```

```
2336 \def\@glo@useri{}%
```

```
2337 \def\@glo@userii{}%
```

```
2338 \def\@glo@useriii{}%
```

```
2339 \def\@glo@useriv{}%
```

```
2340 \def\@glo@userv{}%
```

```
2341 \def\@glo@uservi{}%
```

```
2342 \def\@glo@short{}%
```

```
2343 \def\@glo@shortpl{}%
```

```
2344 \def\@glo@long{}%
```

```
2345 \def\@glo@longpl{}%
```

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

```
2346 \@newglossaryentryprehook
```

Extract key-val information from third parameter:

```
2347 \setkeys{glossentry}{#2}%
```

Check there is a default glossary.

```
2348 \ifundef\glsdefaultttype
```

```
2349 {%
```

```
2350 \PackageError{glossaries}%
```

```
2351 {No default glossary type (have you used 'nomain' by mistake?)}%
```

```
2352 {If you use package option 'nomain' you must define
```

```
2353 a new glossary before you can define entries}%
```

```
2354 }%
```

```
2355 {}%
```

Assign type. This must be fully expandable

```
2356 \gls@assign@field{\glsdefaulttype}{\@glo@label}{type}{\@glo@type}%
2357 \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.

```
2358 \ifcsundef{glolist@\@glo@type}%
2359 {%
2360     \PackageError{glossaries}%
2361     {Glossary type ‘\@glo@type’ has not been defined}%
2362     {You need to define a new glossary type, before making entries
2363      in it}%
2364 }%
2365 {%
```

Check if it's an ignored glossary

```
2366 \ifignoredglossary\@glo@type
2367 {%
```

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

```
2368     \ifx\@glo@desc\glsnodesc
2369         \let\@glo@desc\@empty
2370     \fi
2371 }%
2372 {%
2373 }%
2374 \protected@edef\@glolist@{\csname glolist@\@glo@type\endcsname}%
2375 \expandafter\xdef\csname glolist@\@glo@type\endcsname{%
2376     \@glolist@{\@glo@label},}%
2377 }%
```

Initialise level to 0.

```
2378 \gls@level=0\relax
```

Has this entry been assigned a parent?

```
2379 \ifx\@glo@parent\@empty
```

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

```
2380 \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%
2381 \else
```

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

```
2382 \ifdefequal\@glo@label\@glo@parent%
2383 {%
2384     \PackageError{glossaries}{Entry ‘\@glo@label’ can’t be its own parent}{}%
2385     \def\@glo@parent{}%
2386     \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%
2387 }%
2388 {%
```

Check the parent exists:

```
2389 \ifglsentryexists{\@glo@parent}%
2390 {%
```

Parent exists. Set `\glo@{label}@parent`.

```

2391      \expandafter\xdef\csname glo@\@glo@label @parent\endcsname{%
2392          \@glo@parent}%

Determine level.
2393      \gls@level=\csname glo@\@glo@parent @level\endcsname\relax
2394      \advance\gls@level by 1\relax

If name hasn't been specified, use same as the parent name
2395      \ifx\@glo@name\@gls@name
2396          \expandafter\let\expandafter\@glo@name
2397          \csname glo@\@glo@parent @name\endcsname

If name and plural haven't been specified, use same as the parent
2398      \ifx\@glo@plural\@gls@default@value
2399          \expandafter\let\expandafter\@glo@plural
2400          \csname glo@\@glo@parent @plural\endcsname
2401      \fi
2402      \fi
2403  }%
2404  {%

Parent doesn't exist, so issue an error message and change this entry to have no parent
2405      \PackageError{glossaries}%
2406      {%
2407          Invalid parent '\@glo@parent'
2408          for entry '\@glo@label' - parent doesn't exist%
2409      }%
2410      {%
2411          Parent entries must be defined before their children%
2412      }%
2413      \def\@glo@parent{%
2414          \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{%
2415              }%
2416          }%
2417      \fi

Set the level for this entry
2418      \expandafter\xdef\csname glo@\@glo@label @level\endcsname{\number\gls@level}%

Define commands associated with this entry:
2419      \gls@assign@field{\@glo@name}{\@glo@label}{sortvalue}{\@glo@sort}%
2420      \letcs\@glo@sort{glo@\@glo@label @sortvalue}%
2421      \gls@assign@field{\@glo@name}{\@glo@label}{text}{\@glo@text}%
2422      \expandafter\gls@assign@field\expandafter
2423          {\csname glo@\@glo@label @text\endcsname\glspluralsuffix}%
2424          {\@glo@label}{plural}{\@glo@plural}%
2425      \expandafter\gls@assign@field\expandafter
2426          {\csname glo@\@glo@label @text\endcsname}%
2427          {\@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.

```

2428 \ifx\@glo@first\@gls@default@value
2429   \expandafter\gls@assign@field\expandafter
2430     {\csname glo@\@glo@label @plural\endcsname}%
2431     {\@glo@label}{firstpl}{\@glo@firstplural}%
2432 \else
2433   \expandafter\gls@assign@field\expandafter
2434     {\csname glo@\@glo@label @first\endcsname\glspluralsuffix}%
2435     {\@glo@label}{firstpl}{\@glo@firstplural}%
2436 \fi

2437 \ifcsundef{@glo@type@\@glo@type @counter}%
2438 {%
2439   \def\@glo@defaultcounter{\glscounter}%
2440 }%
2441 {%
2442   \letcs\@glo@defaultcounter{@glo@type@\@glo@type @counter}%
2443 }%
2444 \gls@assign@field{\@glo@defaultcounter}{\@glo@label}{counter}{\@glo@counter}%
2445 \gls@assign@field{}{\@glo@label}{useri}{\@glo@useri}%
2446 \gls@assign@field{}{\@glo@label}{userii}{\@glo@userii}%
2447 \gls@assign@field{}{\@glo@label}{useriii}{\@glo@useriii}%
2448 \gls@assign@field{}{\@glo@label}{useriv}{\@glo@useriv}%
2449 \gls@assign@field{}{\@glo@label}{userv}{\@glo@userv}%
2450 \gls@assign@field{}{\@glo@label}{uservi}{\@glo@uservi}%
2451 \gls@assign@field{}{\@glo@label}{short}{\@glo@short}%
2452 \gls@assign@field{}{\@glo@label}{shortpl}{\@glo@shortpl}%
2453 \gls@assign@field{}{\@glo@label}{long}{\@glo@long}%
2454 \gls@assign@field{}{\@glo@label}{longpl}{\@glo@longpl}%
2455 \ifx\@glo@name\@gls@name
2456   \@gls@name
2457   \let\@glo@name\@gls@default@value
2458 \fi
2459 \gls@assign@field{}{\@glo@label}{name}{\@glo@name}%

```

Set default numberlist if not defined:

```

2460 \ifcsundef{glo@\@glo@label @numberlist}%
2461 {%
2462   \csxdef{glo@\@glo@label @numberlist}{%
2463     \noexpand\@gls@missingnumberlist{\@glo@label}}%
2464 }%
2465 {}%

```

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

```

2466 \@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.

```

2467 \def\@glo@@desc{\@glo@first}%

```

```

2468 \ifx\@glo@desc\@glo@@desc
2469 \let\@glo@desc\@glo@first
2470 \fi
2471 \ifx\@glo@desc\@glsnodesc
2472 \@glsnodesc
2473 \let\@glo@desc\@gls@default@value
2474 \fi
2475 \gls@assign@desc{\@glo@label}%

```

Set the sort key for this entry:

```

2476 \@gls@defsort{\@glo@type}{\@glo@label}%

2477 \def\@glo@@symbol{\@glo@text}%
2478 \ifx\@glo@symbol\@glo@@symbol
2479 \let\@glo@symbol\@glo@text
2480 \fi
2481 \gls@assign@field{\relax}{\@glo@label}{symbol}{\@glo@symbol}%
2482 \expandafter
2483 \gls@assign@field\expandafter
2484 {\csname glo@\@glo@label @symbol\endcsname}
2485 {\@glo@label}{symbolplural}{\@glo@symbolplural}%

```

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

```

2486 \expandafter\xdef\csname glo@\@glo@label @flagfalse\endcsname{%
2487 \noexpand\global
2488 \noexpand\let\expandafter\noexpand
2489 \csname ifglo@\@glo@label @flag\endcsname\noexpand\iffalse
2490 }%
2491 \expandafter\xdef\csname glo@\@glo@label @flagtrue\endcsname{%
2492 \noexpand\global
2493 \noexpand\let\expandafter\noexpand
2494 \csname ifglo@\@glo@label @flag\endcsname\noexpand\iftrue
2495 }%
2496 \csname glo@\@glo@label @flagfalse\endcsname

```

Sort out any cross-referencing if required.

```

2497 \@glo@autosee

```

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

```

2498 \ifignoredglossary\@glo@type
2499 {%
2500 \csdef{glo@\@glo@label @index}{}%
2501 }
2502 {%
2503 \do@glo@storeentry{\@glo@label}%
2504 }%

```

Define entry counters if enabled:

```

2505 \@newglossaryentry@defcounters

```

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

```
2506 \newglossaryentryposthook
2507 }
```

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

```
2508 \newcommand*{\@glo@autosee}{%
2509   \ifdefined\@glo@see{%
2510     {%
2511       \protected@edef\@do@glsee{%
2512         \noexpand\@gls@fixbraces\noexpand\@glo@list\@glo@see\noexpand\@nil
2513         \noexpand\expandafter\noexpand\@glsee\noexpand\@glo@list{\@glo@label}}%
2514       \do@glsee
2515     }%
2516     \@glo@autoseehook
2517 }%
```

`\glo@autoseehook`

```
2518 \newcommand*{\@glo@autoseehook}{}
```

`\@newglossaryentryprehook` Allow extra information to be added to glossary entries:

```
2519 \newcommand*{\@newglossaryentryprehook}{}
```

`\@newglossaryentryposthook` Allow extra information to be added to glossary entries:

```
2520 \newcommand*{\@newglossaryentryposthook}{}
```

`\@newglossaryentry@defcounters`

```
2521 \newcommand*{\@newglossaryentry@defcounters}{}
```

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

```
2522 \newcommand*{\glsmoveentry}[2]{%
2523   \edef\@glo@thislabel{\glsdetoklabel{#1}}%
2524   \edef\glo@type{\csname glo@\@glo@thislabel @type\endcsname}%
2525   \def\glo@list{,%
2526   \forglsentries[\glo@type]{\glo@label}%
2527   {%
2528     \ifdefequal\@glo@thislabel\glo@label
2529       {\eappto\glo@list{\glo@label,}}%
2530     }%
2531     \cslet{glolist@\glo@type}{\glo@list}%
2532     \csdef{glo@\@glo@thislabel @type}{#2}%
2533 }
```

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

```
2534 \ifglxindy
2535   \newcommand*{\@glossaryentryfield}{\string\glossentry}
```

```

2536 \else
2537   \newcommand*{\@glossaryentryfield}{\string\glossentry}
2538 \fi

```

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

```

2539 \ifglxindy
2540   \newcommand*{\@glossarysubentryfield}{%
2541     \string\subglossentry}
2542 \else
2543   \newcommand*{\@glossarysubentryfield}{%
2544     \string\subglossentry}
2545 \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.)

```

2546 \newcommand{\@glo@storeentry}[1]{%

```

Escape makeindex/xindy special characters in the label:

```

2547   \edef\@glo@esclabel{#1}%
2548   \@gls@checkmkidxchars\@glo@esclabel

```

Get the sort string and escape any special characters

```

2549   \protected@edef\@glo@sort{\csname glo@#1@sort\endcsname}%
2550   \@gls@checkmkidxchars\@glo@sort

```

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

```

2551   \@gls@checkmkidxchars\@glo@prefix

```

Get the parent, if one exists

```

2552   \edef\@glo@parent{\csname glo@#1@parent\endcsname}%

```

Write the information to the glossary file.

```

2553   \ifglxindy

```

Store using xindy syntax.

```

2554     \ifx\@glo@parent\@empty

```

Entry doesn't have a parent

```

2555       \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2556         (\string"\@glo@sort\string" %
2557         \string"\@glo@prefix\@glossaryentryfield{\@glo@esclabel}\string") %
2558       }%
2559     \else

```

Entry has a parent

```

2560     \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2561         \csname glo@\@glo@parent @index\endcsname
2562         (\string"\@glo@sort\string" %
2563         \string"\@glo@prefix\@glossarysubentryfield
2564         {\csname glo@#1@level\endcsname}{\@glo@esclabel}\string") %
2565     }%
2566     \fi
2567     \else

Store using makeindex syntax.
2568     \ifx\@glo@parent\@empty

Sanitize \@glo@prefix
2569     \@onelevel@sanitize\@glo@prefix

Entry doesn't have a parent
2570     \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2571         \@glo@sort\@gls@actualchar\@glo@prefix
2572         \@glossaryentryfield{\@glo@esclabel}%
2573     }%
2574     \else

Entry has a parent
2575     \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2576         \csname glo@\@glo@parent @index\endcsname\@gls@levelchar
2577         \@glo@sort\@gls@actualchar\@glo@prefix
2578         \@glossarysubentryfield
2579         {\csname glo@#1@level\endcsname}{\@glo@esclabel}%
2580     }%
2581     \fi
2582     \fi
2583 }
```

## 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`

```

2584 \AtBeginDocument{%
2585     \@ifpackageloaded{amsmath}%
2586     {\let\gls@ifnotmeasuring\@gls@ifnotmeasuring}%
2587     }%
2588 }
2589 \newcommand*{\@gls@ifnotmeasuring}[1]{%
2590     \ifmeasuring@
```



```

2591 \else
2592   #1%
2593 \fi
2594 }
2595 \newcommand*\gls@ifnotmeasuring[1]{#1}

```

`\glspatchtabularx` 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.

```

2596 \def\@gls@patchtabularx#1\hbox#2#3!!{%
2597   \def\TX@trial##1{#1\hbox{\let\glsunset\@gobble#2}#3}%
2598 }
2599 \newcommand*\glspatchtabularx{%
2600   \ifdef\TX@trial
2601   {%
2602     \expandafter\@gls@patchtabularx\TX@trial{##1}!!%
2603     \let\glspatchtabularx\relax
2604   }%
2605   {%
2606 }

```

`\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.

```

2607 \newcommand*\glsreset}[1]{%
2608   \gls@ifnotmeasuring
2609   {%
2610     \glsdoifexists{#1}%
2611     {%
2612       \@glsreset{#1}%
2613     }%
2614   }%
2615 }

```

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

```

2616 \newcommand*\glslocalreset}[1]{%
2617   \gls@ifnotmeasuring
2618   {%
2619     \glsdoifexists{#1}%
2620     {%
2621       \@glslocalreset{#1}%
2622     }%
2623   }%
2624 }

```

`\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.

```

2625 \newcommand*\glsunset}[1]{%
2626   \gls@ifnotmeasuring
2627   {%

```

```

2628     \glsdoifexists{#1}%
2629     {%
2630         \@glsunset{#1}%
2631     }%
2632 }%
2633 }

```

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

```

2634 \newcommand*{\glslocalunset}[1]{%
2635     \gls@ifnotmeasuring
2636     {%
2637         \glsdoifexists{#1}%
2638         {%
2639             \@glslocalunset{#1}%
2640         }%
2641     }%
2642 }

```

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

```

2643 \newcommand*{\@glslocalunset}{\@@glslocalunset}

```

`@@glslocalunset` Local unset without checks.

```

2644 \newcommand*{\@@glslocalunset}[1]{%
2645     \expandafter\let\csname ifglo@glsdetoklabel{#1}@flag\endcsname\iftrue
2646 }

```

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

```

2647 \newcommand*{\@glsunset}{\@@glsunset}

```

`\@@glsunset` Global unset without checks.

```

2648 \newcommand*{\@@glsunset}[1]{%
2649     \expandafter\global\csname glo@glsdetoklabel{#1}@flagtrue\endcsname
2650 }

```

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

```

2651 \newcommand*{\@glslocalreset}{\@@glslocalreset}

```

`@@glslocalreset` Local reset without checks.

```

2652 \newcommand*{\@@glslocalreset}[1]{%
2653     \expandafter\let\csname ifglo@glsdetoklabel{#1}@flag\endcsname\iffalse
2654 }

```

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

```

2655 \newcommand*{\@glsreset}{\@@glsreset}

```

`\@@glsreset` Global reset without checks.

```
2656 \newcommand*{\@@glsreset}[1]{%
2657   \expandafter\global\csname glo@glsdetoklabel{#1}@flagfalse\endcsname
2658 }
```

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

`\glsresetall`

```
2659 \newcommand*{\glsresetall}[1][\@glo@types]{%
2660   \forallglsentries[#1]{\@glsentry}%
2661   {%
2662     \glsreset{\@glsentry}%
2663   }%
2664 }
```

As above, but with only a local effect:

`lslocalresetall`

```
2665 \newcommand*{\glslocalresetall}[1][\@glo@types]{%
2666   \forallglsentries[#1]{\@glsentry}%
2667   {%
2668     \glslocalreset{\@glsentry}%
2669   }%
2670 }
```

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

`\glsunsetall`

```
2671 \newcommand*{\glsunsetall}[1][\@glo@types]{%
2672   \forallglsentries[#1]{\@glsentry}%
2673   {%
2674     \glsunset{\@glsentry}%
2675   }%
2676 }
```

As above, but with only a local effect:

`lslocalunsetall`

```
2677 \newcommand*{\glslocalunsetall}[1][\@glo@types]{%
2678   \forallglsentries[#1]{\@glsentry}%
2679   {%
2680     \glslocalunset{\@glsentry}%
2681   }%
2682 }
```

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

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

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

```
2683 \newcommand*{\@newglossaryentry@defcounters}{%
2684   \csdef{glo@\@glo@label @currcount}{0}%
2685   \csdef{glo@\@glo@label @prevcount}{0}%
2686 }
```

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

```
2687 \newcommand*{\glsenableentrycount}{%
```

Enable new entry fields.

```
2688 \let\@newglossaryentry@defcounters\@newglossaryentry@defcounters
```

Disable `\newglossaryentry` in the document environment.

```
2689 \renewcommand*{\gls@defdocnewglossaryentry}{%
2690   \renewcommand*{\newglossaryentry}[2]{%
2691     \PackageError{glossaries}{\string\newglossaryentry\space
2692     may only be used in the preamble when entry counting has
2693     been activated}{If you use \string\glsenableentrycount\space
2694     you must place all entry definitions in the preamble not in
2695     the document environment}%
2696   }%
2697 }
```

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

```
2698 \newcommand*{\glsentrycurrcount}[1]{%
2699   \ifcsundef{glo@\glsdetoklabel{##1}@currcount}%
2700   {0}{\@gls@entry@field{##1}{currcount}}%
2701 }%
2702 \newcommand*{\glsentryprevcount}[1]{%
2703   \ifcsundef{glo@\glsdetoklabel{##1}@prevcount}%
2704   {0}{\@gls@entry@field{##1}{prevcount}}%
2705 }
```

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

```
2706 \renewcommand*{\@glsunset}[1]{%
2707   \@glsunset{##1}%
2708   \@gls@increment@currcount{##1}%
2709 }
```

```

2710 \renewcommand*\@glslocalunset}[1]{%
2711   \@glslocalunset{##1}%
2712   \@gls@local@increment@currcount{##1}%
2713 }%
2714 \renewcommand*\@glsreset}[1]{%
2715   \@glsreset{##1}%
2716   \csgdef{glo@glsdetoklabel{##1}@currcount}{0}%
2717 }%
2718 \renewcommand*\@glslocalreset}[1]{%
2719   \@glslocalreset{##1}%
2720   \csdef{glo@glsdetoklabel{##1}@currcount}{0}%
2721 }%

```

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.)

```

2722 \def\@cgl's@##1##2[##3]{%
2723   \ifnum\glsentryprevcount{##2}=1\relax
2724     \cgl'sformat{##2}{##3}%
2725     \glsunset{##2}%
2726   \else
2727     \@gls@{##1}{##2}[##3]%
2728   \fi
2729 }%

```

Similarly for the analogous commands. No case change plural:

```

2730 \def\@cgl'spl@##1##2[##3]{%
2731   \ifnum\glsentryprevcount{##2}=1\relax
2732     \cgl'splformat{##2}{##3}%
2733     \glsunset{##2}%
2734   \else
2735     \@cgl'spl@{##1}{##2}[##3]%
2736   \fi
2737 }%

```

First letter uppercase singular:

```

2738 \def\@cGls@##1##2[##3]{%
2739   \ifnum\glsentryprevcount{##2}=1\relax
2740     \cGlsformat{##2}{##3}%
2741     \glsunset{##2}%
2742   \else
2743     \@Gls@{##1}{##2}[##3]%
2744   \fi
2745 }%

```

First letter uppercase plural:

```

2746 \def\@cGlspl@##1##2[##3]{%
2747   \ifnum\glsentryprevcount{##2}=1\relax
2748     \cGlsplformat{##2}{##3}%
2749     \glsunset{##2}%
2750   \else
2751     \@Glspl@{##1}{##2}[##3]%

```

```
2752 \fi
2753 }%
```

Write information to aux file at the end of the document

```
2754 \AtEndDocument{\@gls@write@entrycounts}%
```

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

```
2755 \renewcommand*{\@gls@entry@count}[2]{%
2756   \csxdef{glo@glstoklabel{##1}@prevcount}{##2}%
2757 }%
```

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

```
2758 \let\glsenableentrycount\relax
2759 }
2760 \@onlypreamble\glsenableentrycount
```

`ement@currcount`

```
2761 \newcommand*{\@gls@increment@currcount}[1]{%
2762   \csxdef{glo@glstoklabel{##1}@currcount}{%
2763     \number\numexpr\glsentrycurrcount{##1}+1}%
2764 }
```

`ement@currcount`

```
2765 \newcommand*{\@gls@local@increment@currcount}[1]{%
2766   \csxdef{glo@glstoklabel{##1}@currcount}{%
2767     \number\numexpr\glsentrycurrcount{##1}+1}%
2768 }
```

`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.)

```
2769 \newcommand*{\@gls@write@entrycounts}{%
2770   \immediate\write\@auxout
2771     {\string\providecommand*{\string\@gls@entry@count}[2]{}}%
2772   \forallglsentries{\@glsentry}{%
2773     \ifglsused{\@glsentry}%
2774     {\immediate\write\@auxout
2775       {\string\@gls@entry@count{\@glsentry}{\glsentrycurrcount{\@glsentry}}}%
2776     }%
2777   }%
2778 }
```

`gls@entry@count`

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

```
2779 \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.)

```
2780 \newrobustcmd*{\cgl}{\@gls@hyp@opt\@cgl}
```

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

```
2781 \newcommand*{\@cgl}{2} [] {%
2782   \new@ifnextchar[{\@cgl@{#1}{#2}}{\@cgl@{#1}{#2} []}%
2783 }
```

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

```
2784 \def\@cgl@#1#2[#3]{%
2785   \GlossariesWarning{\string\cgl\space is defaulting to
2786     \string\gl\space since you haven't enabled entry counting}%
2787   \@gl@{#1}{#2}[#3]%
2788 }
```

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

```
2789 \newcommand*{\cglformat}{2} {%
2790   \ifglshaslong{#1}{\glentrylong{#1}}{\glentryfirst{#1}}#2%
2791 }
```

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

```
2792 \newrobustcmd*{\cGl}{\@gl@hyp@opt\@cGl}
```

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

```
2793 \newcommand*{\@cGl}{2} [] {%
2794   \new@ifnextchar[{\@cGl@{#1}{#2}}{\@cGl@{#1}{#2} []}%
2795 }
```

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

```
2796 \def\@cGl@#1#2[#3]{%
2797   \GlossariesWarning{\string\cGl\space is defaulting to
2798     \string\Gl\space since you haven't enabled entry counting}%
2799   \@Gl@{#1}{#2}[#3]%
2800 }
```

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

```
2801 \newcommand*{\cGlsformat}{2} {%
2802   \ifglshaslong{#1}{\Glentrylong{#1}}{\Glentryfirst{#1}}#2%
2803 }
```

`\cgl`sp1 Define command that works like `\gl`sp1 but behaves differently if the entry count function is enabled. (If not enabled, it behaves the same as `\gl`sp1 but issues a warning.)

```
2804 \newrobustcmd*{\cglsp1}{\@gl@hyp@opt\@cglsp1}
```

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

```
2805 \newcommand*{\@cglsp1}[2] [] {%
2806   \new@ifnextchar[{\@cglsp1@{#1}{#2}}{\@cglsp1@{#1}{#2} []}%
2807 }
```

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

```
2808 \def\@cglsp1@#1#2[#3] {%
2809   \GlossariesWarning{\string\cglsp1\space is defaulting to
2810     \string\glsp1\space since you haven't enabled entry counting}%
2811   \@glsp1@{#1}{#2}[#3]%
2812 }
```

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

```
2813 \newcommand*{\cglsp1format}[2] {%
2814   \ifglshaslong{#1}{\gl Sentrylongpl{#1}}{\gl Sentryfirstplural{#1}}#2%
2815 }
```

`\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.)

```
2816 \newrobustcmd*{\cGlsp1}{\@glshyp@opt\@cGlsp1}
```

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

```
2817 \newcommand*{\@cGlsp1}[2] [] {%
2818   \new@ifnextchar[{\@cGlsp1@{#1}{#2}}{\@cGlsp1@{#1}{#2} []}%
2819 }
```

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

```
2820 \def\@cGlsp1@#1#2[#3] {%
2821   \GlossariesWarning{\string\cGlsp1\space is defaulting to
2822     \string\Glsp1\space since you haven't enabled entry counting}%
2823   \@Glsp1@{#1}{#2}[#3]%
2824 }
```

`\cGlsp1format` Format used by `\cGlsp1` if entry only used once on previous run. The first argument is the label, the second argument is the insert text.

```
2825 \newcommand*{\cGlsp1format}[2] {%
2826   \ifglshaslong{#1}{\Gl Sentrylongpl{#1}}{\Gl Sentryfirstplural{#1}}#2%
2827 }
```

## 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.<sup>1</sup>

---

<sup>1</sup>and any other valid  $\LaTeX$  code that can be used in the preamble.



`\loadglsentries[⟨type⟩]{⟨filename⟩}`

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

`\loadglsentries`

```
2828 \newcommand*{\loadglsentries}[2][\@gls@default]{%
2829   \let\@gls@default\glsdefaulttype
2830   \def\glsdefaulttype{#1}\input{#2}%
2831   \let\glsdefaulttype\@gls@default
2832 }
```

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

```
2833 \@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`

```
2834 \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

```
2835 \newcommand*{\glsentryfmt}{%
2836   \@gls@default@entryfmt\glsdisplayfirst\glsdisplay
2837 }
```

Format that provides backwards compatibility:

```
2838 \newcommand*{\@gls@default@entryfmt}[2]{%
2839   \ifdefempty\glscustomtext
2840   {%
2841     \glsifplural
2842     {%
```

Plural form

```
2843     \glscapscase
2844     {%
```

Don't adjust case

```
2845      \ifglused\glslabel
2846      {%
```

Subsequent use

```
2847      #2{\glentryplural{\glslabel}}%
2848      {\glentrydescplural{\glslabel}}%
2849      {\glentrysymbolplural{\glslabel}}{\glinsert}%
2850      }%
2851      {%
```

First use

```
2852      #1{\glentryfirstplural{\glslabel}}%
2853      {\glentrydescplural{\glslabel}}%
2854      {\glentrysymbolplural{\glslabel}}{\glinsert}%
2855      }%
2856      }%
2857      {%
```

Make first letter upper case

```
2858      \ifglused\glslabel
2859      {%
```

Subsequent use. (Expansion was used in version 3.07 and below in case the name wasn't the first thing to be displayed, but now the user can sort out the upper casing in `\defglentryfmt`, which avoids the issues caused by fragile commands.)

```
2860      \ifbool{glcompatible-3.07}%
2861      {%
2862      \protected@edef\@glo@etext{%
2863      #2{\glentryplural{\glslabel}}%
2864      {\glentrydescplural{\glslabel}}%
2865      {\glentrysymbolplural{\glslabel}}{\glinsert}}%
2866      \xmakefirstuc\@glo@etext
2867      }%
2868      {%
2869      #2{\Glsentryplural{\glslabel}}%
2870      {\glentrydescplural{\glslabel}}%
2871      {\glentrysymbolplural{\glslabel}}{\glinsert}%
2872      }%
2873      }%
2874      {%
```

First use

```
2875      \ifbool{glcompatible-3.07}%
2876      {%
2877      \protected@edef\@glo@etext{%
2878      #1{\glentryfirstplural{\glslabel}}%
2879      {\glentrydescplural{\glslabel}}%
2880      {\glentrysymbolplural{\glslabel}}{\glinsert}}%
2881      \xmakefirstuc\@glo@etext
2882      }%
```

```

2883         {%
2884         #1{\Glsentryfirstplural{\glslabel}}}%
2885         {\glsentrydescplural{\glslabel}}}%
2886         {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2887     }%
2888 }%
2889 }%
2890 {%

```

Make all upper case

```

2891     \ifglsused\glslabel
2892     {%

```

Subsequent use

```

2893         \mfirstucMakeUppercase{#2{\glsentryplural{\glslabel}}}%
2894         {\glsentrydescplural{\glslabel}}}%
2895         {\glsentrysymbolplural{\glslabel}}{\glsinsert}}}%
2896     }%
2897     {%

```

First use

```

2898         \mfirstucMakeUppercase{#1{\glsentryfirstplural{\glslabel}}}%
2899         {\glsentrydescplural{\glslabel}}}%
2900         {\glsentrysymbolplural{\glslabel}}{\glsinsert}}}%
2901     }%
2902 }%
2903 }%
2904 {%

```

Singular form

```

2905     \glscapscase
2906     {%

```

Don't adjust case

```

2907     \ifglsused\glslabel
2908     {%

```

Subsequent use

```

2909         #2{\glsentrytext{\glslabel}}}%
2910         {\glsentrydesc{\glslabel}}}%
2911         {\glsentrysymbol{\glslabel}}{\glsinsert}%
2912     }%
2913     {%

```

First use

```

2914         #1{\glsentryfirst{\glslabel}}}%
2915         {\glsentrydesc{\glslabel}}}%
2916         {\glsentrysymbol{\glslabel}}{\glsinsert}%
2917     }%
2918 }%
2919 {%

```

### Make first letter upper case

```
2920 \ifglused\glslabel
2921 {%
```

#### Subsequent use

```
2922 \ifbool{glscompatible-3.07}%
2923 {%
2924 \protected@edef\@glo@etext{%
2925 #2{\glsentrytext{\glslabel}}%
2926 {\glsentrydesc{\glslabel}}%
2927 {\glsentrysymbol{\glslabel}}{\glsinsert}}%
2928 \xmakefirstuc\@glo@etext
2929 }%
2930 {%
2931 #2{\Glsentrytext{\glslabel}}%
2932 {\glsentrydesc{\glslabel}}%
2933 {\glsentrysymbol{\glslabel}}{\glsinsert}}%
2934 }%
2935 }%
2936 {%
```

#### First use

```
2937 \ifbool{glscompatible-3.07}%
2938 {%
2939 \protected@edef\@glo@etext{%
2940 #1{\glsentryfirst{\glslabel}}%
2941 {\glsentrydesc{\glslabel}}%
2942 {\glsentrysymbol{\glslabel}}{\glsinsert}}%
2943 \xmakefirstuc\@glo@etext
2944 }%
2945 {%
2946 #1{\Glsentryfirst{\glslabel}}%
2947 {\glsentrydesc{\glslabel}}%
2948 {\glsentrysymbol{\glslabel}}{\glsinsert}}%
2949 }%
2950 }%
2951 }%
2952 {%
```

### Make all upper case

```
2953 \ifglused\glslabel
2954 {%
```

#### Subsequent use

```
2955 \mfirstucMakeUppercase{#2{\glsentrytext{\glslabel}}%
2956 {\glsentrydesc{\glslabel}}%
2957 {\glsentrysymbol{\glslabel}}{\glsinsert}}%
2958 }%
2959 {%
```

#### First use

```

2960      \mfirstucMakeUppercase{#1{\glsentryfirst{\glslabel}}}%
2961      {\glsentrydesc{\glslabel}}}%
2962      {\glsentrysymbol{\glslabel}}{\glsinsert}}}%
2963    }%
2964  }%
2965 }%
2966 }%
2967 {%

```

Custom text provided in \glsdisp

```

2968   \ifglsused{\glslabel}%
2969   {%

```

Subsequent use

```

2970     #2{\glscustomtext}%
2971     {\glsentrydesc{\glslabel}}}%
2972     {\glsentrysymbol{\glslabel}}{}%
2973   }%
2974   {%

```

First use

```

2975     #1{\glscustomtext}%
2976     {\glsentrydesc{\glslabel}}}%
2977     {\glsentrysymbol{\glslabel}}{}%
2978   }%
2979 }%
2980 }

```

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

```

2981 \newcommand*{\glsentryfmt}{%
2982   \ifdefempty\glscustomtext
2983   {%
2984     \glsifplural
2985     {%

```

Plural form

```

2986     \glscapscase
2987     {%

```

Don't adjust case

```

2988     \ifglsused\glslabel
2989     {%

```

Subsequent use

```

2990     \glsentryplural{\glslabel}\glsinsert
2991   }%
2992   {%

```

First use

```

2993     \glsentryfirstplural{\glslabel}\glsinsert
2994   }%

```

2995 }%  
2996 {%

#### Make first letter upper case

2997 \ifglused\glslabel  
2998 {%

#### Subsequent use.

2999 \Glseentryplural{\glslabel}\glsinsert  
3000 }%  
3001 {%

#### First use

3002 \Glseentryfirstplural{\glslabel}\glsinsert  
3003 }%  
3004 }%  
3005 {%

#### Make all upper case

3006 \ifglused\glslabel  
3007 {%

#### Subsequent use

3008 \mfirstucMakeUppercase  
3009 {\glseentryplural{\glslabel}\glsinsert}%  
3010 }%  
3011 {%

#### First use

3012 \mfirstucMakeUppercase  
3013 {\glseentryfirstplural{\glslabel}\glsinsert}%  
3014 }%  
3015 }%  
3016 }%  
3017 {%

#### Singular form

3018 \glscapscase  
3019 {%

#### Don't adjust case

3020 \ifglused\glslabel  
3021 {%

#### Subsequent use

3022 \glseentrytext{\glslabel}\glsinsert  
3023 }%  
3024 {%

#### First use

3025 \glseentryfirst{\glslabel}\glsinsert  
3026 }%  
3027 }%  
3028 {%

Make first letter upper case

```
3029      \ifglused\glslabel
3030      {%
```

Subsequent use

```
3031      \Glsentrytext{\glslabel}\glsinsert
3032      }%
3033      {%
```

First use

```
3034      \Glsentryfirst{\glslabel}\glsinsert
3035      }%
3036      }%
3037      {%
```

Make all upper case

```
3038      \ifglused\glslabel
3039      {%
```

Subsequent use

```
3040      \mfirstucMakeUppercase{\glsentrytext{\glslabel}\glsinsert}%
3041      }%
3042      {%
```

First use

```
3043      \mfirstucMakeUppercase{\glsentryfirst{\glslabel}\glsinsert}%
3044      }%
3045      }%
3046      }%
3047      }%
3048      {%
```

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

```
3049      \glscustomtext\glsinsert
3050      }%
3051 }
```

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

```
3052 \newcommand*{\glsgenacfmt}{%
3053   \ifdefempty\glscustomtext
3054   {%
3055     \ifglused\glslabel
3056     {%
```

Subsequent use:

```
3057     \glsifplural
3058     {%
```

Subsequent plural form:

```
3059     \glscapscase
3060     {%
```

Subsequent plural form, don't adjust case:

```
3061      \acronymfont{\glsentryshortpl{\glslabel}}\glsinsert
3062      }%
3063      {%
```

Subsequent plural form, make first letter upper case:

```
3064      \acronymfont{\Glsentryshortpl{\glslabel}}\glsinsert
3065      }%
3066      {%
```

Subsequent plural form, all caps:

```
3067      \mfirstucMakeUppercase
3068      {\acronymfont{\glsentryshortpl{\glslabel}}\glsinsert}%
3069      }%
3070      }%
3071      {%
```

Subsequent singular form

```
3072      \glscapscase
3073      {%
```

Subsequent singular form, don't adjust case:

```
3074      \acronymfont{\glsentryshort{\glslabel}}\glsinsert
3075      }%
3076      {%
```

Subsequent singular form, make first letter upper case:

```
3077      \acronymfont{\Glsentryshort{\glslabel}}\glsinsert
3078      }%
3079      {%
```

Subsequent singular form, all caps:

```
3080      \mfirstucMakeUppercase
3081      {\acronymfont{\glsentryshort{\glslabel}}\glsinsert}%
3082      }%
3083      }%
3084      }%
3085      {%
```

First use:

```
3086      \glsifplural
3087      {%
```

First use plural form:

```
3088      \glscapscase
3089      {%
```

First use plural form, don't adjust case:

```
3090      \genplacrformat{\glslabel}{\glsinsert}%
3091      }%
3092      {%
```



First use plural form, make first letter upper case:

```
3093      \Genplacrfullformat{\glslabel}{\glsinsert}%
3094      }%
3095      {%
```

First use plural form, all caps:

```
3096      \mfirstucMakeUppercase
3097      {\genplacrfullformat{\glslabel}{\glsinsert}}%
3098      }%
3099      }%
3100      {%
```

First use singular form

```
3101      \glscapscase
3102      {%
```

First use singular form, don't adjust case:

```
3103      \genacrfullformat{\glslabel}{\glsinsert}%
3104      }%
3105      {%
```

First use singular form, make first letter upper case:

```
3106      \Genacrfullformat{\glslabel}{\glsinsert}%
3107      }%
3108      {%
```

First use singular form, all caps:

```
3109      \mfirstucMakeUppercase
3110      {\genacrfullformat{\glslabel}{\glsinsert}}%
3111      }%
3112      }%
3113      }%
3114      }%
3115      {%
```

User supplied text.

```
3116      \glscustomtext
3117      }%
3118 }
```

genacrfullformat

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

The full format used by \gls`genacfmt` (singular).

```
3119 \newcommand*{\genacrfullformat}[2]{%
3120   \glsentrylong{#1}#2\space
3121   (\protect\firstacronymfont{\glsentryshort{#1}})%
3122 }
```

Genacrfullformat

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

As above but makes the first letter upper case.

```
3123 \newcommand*{\Genacrfullformat}[2]{%
3124   \protected@edef\gls@text{\genacrfullformat{#1}{#2}}%
3125   \xmakefirstuc\gls@text
3126 }
```

`\genplacrfullformat` `\genplacrfullformat{\label}{\insert}`

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

```
3127 \newcommand*{\genplacrfullformat}[2]{%
3128   \glsentrylongpl{#1}#2\space
3129   (\protect\firstacronymfont{\glsentryshortpl{#1}})%
3130 }
```

`\Genplacrfullformat` `\Genplacrfullformat{\label}{\insert}`

As above but makes the first letter upper case.

```
3131 \newcommand*{\Genplacrfullformat}[2]{%
3132   \protected@edef\gls@text{\genplacrfullformat{#1}{#2}}%
3133   \xmakefirstuc\gls@text
3134 }
```

`\glsdisplayfirst` Deprecated. Kept for backward compatibility.

```
3135 \newcommand*{\glsdisplayfirst}[4]{#1#4}
```

`\glsdisplay` Deprecated. Kept for backward compatibility.

```
3136 \newcommand*{\glsdisplay}[4]{#1#4}
```

`\defglsdisplay` Deprecated. Kept for backward compatibility.

```
3137 \newcommand*{\defglsdisplay}[2][\glsdefaulttype]{%
3138   \GlossariesWarning{\string\defglsdisplay\space is now obsolete.^^J
3139   Use \string\defglsentryfmt\space instead}%
3140   \expandafter\def\csname gls@#1@display\endcsname##1##2##3##4{#2}%
3141   \edef\@gls@doentrydef{%
3142     \noexpand\defglsentryfmt[#1]{%
3143       \noexpand\ifcsdef\gls@#1@displayfirst}%
3144     {%
3145       \noexpand\@@gls@default@entryfmt
3146       {\noexpand\csuse\gls@#1@displayfirst}}%
3147     {\noexpand\csuse\gls@#1@display}}%
3148   }%
3149   {%
3150     \noexpand\@@gls@default@entryfmt
3151     {\noexpand\glsdisplayfirst}%
3152     {\noexpand\csuse\gls@#1@display}}%
3153   }%
```

```

3154 }%
3155 }%
3156 \@gls@doentrydef
3157 }

```

`glsdisplayfirst` Deprecated. Kept for backward compatibility.

```

3158 \newcommand*{\defglsdisplayfirst}[2][\glsdefaulttype]{%
3159   \GlossariesWarning{\string\defglsdisplayfirst\space is now obsolete.^^J
3160   Use \string\defglsentryfmt\space instead}%
3161   \expandafter\def\csname gls@#1@displayfirst\endcsname##1##2##3##4{#2}%
3162   \edef\@gls@doentrydef{%
3163     \noexpand\defglsentryfmt[#1]{%
3164       \noexpand\ifcsdef{gls@#1@display}%
3165       {%
3166         \noexpand\@gls@default@entryfmt
3167         {\noexpand\csuse{gls@#1@displayfirst}}}%
3168         {\noexpand\csuse{gls@#1@display}}}%
3169     }%
3170     {%
3171       \noexpand\@gls@default@entryfmt
3172       {\noexpand\csuse{gls@#1@displayfirst}}}%
3173       {\noexpand\glsdisplay}%
3174     }%
3175   }%
3176 }%
3177 \@gls@doentrydef
3178 }

```

## 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.

```

3179 \define@key{glslink}{counter}{%
3180   \ifcsundef{c@#1}%
3181   {%
3182     \PackageError{glossaries}%
3183     {There is no counter called '#1'}%
3184     {%

```

```

3185      The counter key should have the name of a valid counter
3186      as its value%
3187    }%
3188  }%
3189  {%
3190    \def\@gls@counter{#1}%
3191  }%
3192 }

```

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.

```

3193 \define@key{glslink}{format}{%
3194   \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.

```

3195 \define@boolkey{glslink}{hyper}[true]{}

```

Initialise hyper key.

```

3196 \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.

```

3197 \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.

```

3198 \newcommand*{\glslinkvar}[3]{#1}

```

`\glsifhyper` Now deprecated.

```

3199 \newcommand*{\glsifhyper}[2]{%
3200   \glslinkvar{#1}{#2}{#1}%
3201   \GlossariesWarning{\string\glsifhyper\space is deprecated. Did
3202     you mean \string\glsifhyperon\space or \string\glslinkvar?}%
3203 }

```

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

```

3204 \newcommand*{\@gls@hyp@opt}[1]{%

```

```

3205 \let\glslinkvar\@firstofthree
3206 \let\@gls@hyp@opt@cs#1\relax
3207 \@ifstar{\s@gls@hyp@opt}%
3208 {\@ifnextchar+{\@firstoftwo{\p@gls@hyp@opt}}{\#1}}%
3209 }

```

\s@gls@hyp@opt Starred version

```

3210 \newcommand*{\s@gls@hyp@opt}[1] [] {%
3211 \let\glslinkvar\@secondofthree
3212 \@gls@hyp@opt@cs[hyper=false,#1]}

```

\p@gls@hyp@opt Plus version

```

3213 \newcommand*{\p@gls@hyp@opt}[1] [] {%
3214 \let\glslinkvar\@thirdofthree
3215 \@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

```

3216 \newrobustcmd*{\glslink}{%
3217 \@gls@hyp@opt\@gls@link
3218 }

```

\@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.

```

3219 \newcommand*{\@gls@link}[3] [] {%
3220 \glsdoifexistsordo{#2}%
3221 {%
3222 \let\do@gls@link@checkfirsthyper\relax
3223 \@gls@link[#1]{#2}{#3}%
3224 }{%

```

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

```

3225 \glstextformat{#3}%
3226 }%

```

```

3227 \glspostlinkhook
3228 }

glspostlinkhook
3229 \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.
3230 \newcommand*{\@gls@link@checkfirsthyper}{%
3231 \ifglsused{\glslabel}%
3232 {%
3233 }%
3234 {%
3235 \gls@checkisacronymlist\glstype
3236 \ifglshyperfirst
3237 \if@glsisacronymlist
3238 \ifglsacrfootnote
3239 \KV@glslink@hyperfalse
3240 \fi
3241 \fi
3242 \else
3243 \KV@glslink@hyperfalse
3244 \fi
3245 }%

Allow user to hook into this
3246 \glslinkcheckfirsthyperhook
3247 }

kfirsthyperhook Allow used to hook into the \@gls@link@checkfirsthyper macro
3248 \newcommand*{\glslinkcheckfirsthyperhook}{}

linkpostsetkeys
3249 \newcommand*{\glslinkpostsetkeys}{}

\glsifhyperon Check the value of the hyper key:
3250 \newcommand{\glsifhyperon}[2]{\ifKV@glslink@hyper#1\else#2\fi}

ablehyperinlist Disable hyperlink if in the “nohyper” list.
3251 \newcommand*{\do@glsdisablehyperinlist}{%
3252 \expandafter\DTLifinlist\expandafter{\glstype}{\@gls@nohyperlist}%
3253 {\KV@glslink@hyperfalse}}%
3254 }

lt@glslink@opts Hook to set default options for \@glslink.
3255 \newcommand*{\@gls@setdefault@glslink@opts}{}

```

\@gls@link

```
3256 \def\@gls@link[#1]#2#3{%
```

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

```
3257 \leavevmode
```

```
3258 \edef\glslabel{\glsdetoklabel{#2}}%
```

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

```
3259 \def\@gls@link@opts{#1}%
```

```
3260 \let\@gls@link@label\glslabel
```

```
3261 \def\@glsnumberformat{glsnumberformat}%
```

```
3262 \edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%
```

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

```
3263 \edef\glstype{\csname glo@\glslabel @type\endcsname}%
```

Save original setting

```
3264 \let\org@ifKV@glslink@hyper\ifKV@glslink@hyper
```

Set defaults:

```
3265 \@gls@setdefault@glslink@opts
```

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

```
3266 \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.

```
3267 \do@gls@link@checkfirsthyper
```

```
3268 \setkeys{glslink}{#1}%
```

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

```
3269 \glslinkpostsetkeys
```

Store the entry’s counter in \theglsentrycounter

```
3270 \@gls@saveentrycounter
```

Define sort key if necessary:

```
3271 \@gls@setsort{\glslabel}%
```

(De-tok’ing done by \@do@wrglossary)

```
3272 \@do@wrglossary{#2}%
```

```
3273 \ifKV@glslink@hyper
```

```
3274 \glslink{\glslinkprefix\glslabel}{\glstextformat{#3}}%
```

```
3275 \else
```

```
3276 \glsdonohyperlink{\glslinkprefix\glslabel}{\glstextformat{#3}}%
```

```
3277 \fi
```

Restore original setting

```
3278 \let\ifKV@glslink@hyper\org@ifKV@glslink@hyper
```

```
3279 }
```

```

\glolinkprefix
3280 \newcommand*{\glolinkprefix}{glo:}

glentrycounter  Set default value of entry counter
3281 \def\glentrycounter{\glscounter}%

saveentrycounter  Need to check if using equation counter in align environment:
3282 \newcommand*{\@gls@saveentrycounter}{%
3283   \def\@gls@Hcounter{}}%

  Are we using equation counter?
3284   \ifthenelse{\equal{\@gls@counter}{equation}}{%
3285     {

      If we're in align environment, \xatlevel@ will be defined. (Can't test for \@currentvir as
      may be inside an inner environment.)
3286       \ifcsundef{xatlevel@}%
3287       {%
3288         \edef\theglentrycounter{\expandafter\noexpand
3289           \csname the\@gls@counter\endcsname}%
3290       }%
3291       {%
3292         \ifx\xatlevel@\@empty
3293           \edef\theglentrycounter{\expandafter\noexpand
3294             \csname the\@gls@counter\endcsname}%
3295         \else
3296           \savecounters@
3297           \advance\c@equation by 1\relax
3298           \edef\theglentrycounter{\csname the\@gls@counter\endcsname}%

      Check if hyperref version of this counter
3299           \ifcsundef{theH\@gls@counter}%
3300           {%
3301             \def\@gls@Hcounter{\theglentrycounter}%
3302           }%
3303           {%
3304             \def\@gls@Hcounter{\csname theH\@gls@counter\endcsname}%
3305           }%
3306           \protected@edef\theHglentrycounter{\@gls@Hcounter}%
3307           \restorecounters@
3308         \fi
3309       }%
3310     }%
3311     {%

      Not using equation counter so no special measures:
3312       \edef\theglentrycounter{\expandafter\noexpand
3313         \csname the\@gls@counter\endcsname}%
3314     }%

```



Check if hyperref version of this counter

```

3315 \ifx\@gls@Hcounter\@empty
3316 \ifcsundef{theH\@gls@counter}%
3317 {%
3318 \def\theHglentrycounter{\theglentrycounter}%
3319 }%
3320 {%
3321 \protected@edef\theHglentrycounter{\expandafter\noexpand
3322 \csname theH\@gls@counter\endcsname}%
3323 }%
3324 \fi
3325 }

```

`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.

```

3326 \def\@set@glo@numformat#1#2#3#4{%
3327 \expandafter\@glo@check@mkidxrangechar#3\@nil
3328 \protected@edef#1{%
3329 \@glo@prefix setentrycounter[#4]{#2}%
3330 \expandafter\string\csname\@glo@suffix\endcsname
3331 }%
3332 \@gls@checkmkidxchars#1%
3333 }

```

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.

```

3334 \def\@glo@check@mkidxrangechar#1#2\@nil{%
3335 \if#1(\relax
3336 \def\@glo@prefix{(%}
3337 \if\relax#2\relax
3338 \def\@glo@suffix{glsnumberformat}%
3339 \else
3340 \def\@glo@suffix{#2}%
3341 \fi
3342 \else
3343 \if#1)\relax
3344 \def\@glo@prefix{)}%
3345 \if\relax#2\relax
3346 \def\@glo@suffix{glsnumberformat}%
3347 \else
3348 \def\@glo@suffix{#2}%
3349 \fi
3350 \else
3351 \def\@glo@prefix{}\def\@glo@suffix{#1#2}%

```

```

3352 \fi
3353 \fi}

```

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

```

3354 \newcommand*{\@gls@escbsdq}[1]{%
3355   \def\@gls@checkedmkidx{}%
3356   \let\gls@xdystring=#1\relax
3357   \@onelevel@sanitize\gls@xdystring
3358   \edef\do@gls@xdycheckbackslash{%
3359     \noexpand\@gls@xdycheckbackslash\gls@xdystring\noexpand\@nil
3360     \@backslashchar\@backslashchar\noexpand\null}%
3361   \do@gls@xdycheckbackslash
3362   \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdystring}%
3363   \def\@gls@checkedmkidx{}%
3364   \expandafter\@gls@xdycheckquote\gls@xdystring\@nil""\null
3365   \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdystring}%

```

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

```

3366 \for\@gls@tmp:=\gls@protected@pagefmts\do
3367 {%
3368   \edef\@gls@sanitized@tmp{\expandafter\@gobble\string\\ \expandonce\@gls@tmp}%
3369   \@onelevel@sanitize\@gls@sanitized@tmp
3370   \edef\gls@dosubst{%
3371     \noexpand\DTLsubstituteall\noexpand\gls@xdystring
3372     {\@gls@sanitized@tmp}{\expandonce\@gls@tmp}%
3373   }%
3374   \gls@dosubst
3375 }%

```

Assign to required control sequence

```

3376 \let#1=\gls@xdystring
3377 }

```

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

`checkmkidxchars`

```

3378 \newcommand{\@gls@checkmkidxchars}[1]{%
3379   \ifglxsindy
3380     \@gls@escbsdq{#1}%
3381   \else
3382     \def\@gls@checkedmkidx{}%
3383     \expandafter\@gls@checkquote#1\@nil""\null
3384     \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3385     \def\@gls@checkedmkidx{}%
3386     \expandafter\@gls@checkescquote#1\@nil""\null
3387     \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3388     \def\@gls@checkedmkidx{}%
3389     \expandafter\@gls@checkescactual#1\@nil"??\null
3390     \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%

```

```

3391 \def\@gls@checkedmkidx{}%
3392 \expandafter\@gls@checkactual#1\@nil??\null
3393 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3394 \def\@gls@checkedmkidx{}%
3395 \expandafter\@gls@checkbar#1\@nil||\null
3396 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3397 \def\@gls@checkedmkidx{}%
3398 \expandafter\@gls@checkescbar#1\@nil|||\null
3399 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3400 \def\@gls@checkedmkidx{}%
3401 \expandafter\@gls@checklevel#1\@nil!!\null
3402 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3403 \fi
3404 }

```

Update the control sequence and strip trailing \@nil:

s@updatechecked

```

3405 \def\@gls@updatechecked#1\@nil#2{\def#2{#1}}

```

\@gls@tmpb Define temporary token

```

3406 \newtoks\@gls@tmpb

```

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

```

3407 \def\@gls@checkquote#1"#2"#3\null{%
3408 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3409 \toks@={#1}%
3410 \ifx\null#2\null
3411 \ifx\null#3\null
3412 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3413 \def\@gls@checkquote{\relax}%
3414 \else
3415 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3416 \@gls@quotechar\@gls@quotechar\@gls@quotechar\@gls@quotechar}%
3417 \def\@gls@checkquote{\@gls@checkquote#3\null}%
3418 \fi
3419 \else
3420 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3421 \@gls@quotechar\@gls@quotechar}%
3422 \ifx\null#3\null
3423 \def\@gls@checkquote{\@gls@checkquote#2""\null}%
3424 \else
3425 \def\@gls@checkquote{\@gls@checkquote#2"#3\null}%
3426 \fi
3427 \fi
3428 \@gls@checkquote
3429 }

```

s@checkescquote Do the same for \:

```

3430 \def\@gls@checkescquote#1\"#2\"#3\null{%
3431   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3432   \toks@={#1}%
3433   \ifx\null#2\null
3434     \ifx\null#3\null
3435       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3436       \def\@gls@checkescquote{\relax}%
3437     \else
3438       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3439         \@gls@quotechar\string\"@gls@quotechar
3440         \@gls@quotechar\string\"@gls@quotechar}%
3441       \def\@gls@checkescquote{\@gls@checkescquote#3\null}%
3442     \fi
3443   \else
3444     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3445       \@gls@quotechar\string\"@gls@quotechar}%
3446     \ifx\null#3\null
3447       \def\@gls@checkescquote{\@gls@checkescquote#2\"\" \null}%
3448     \else
3449       \def\@gls@checkescquote{\@gls@checkescquote#2\"#3\null}%
3450     \fi
3451   \fi
3452 \@@gls@checkescquote
3453 }

```

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

```

3454 \def\@gls@checkescactual#1\?#2\?#3\null{%
3455   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3456   \toks@={#1}%
3457   \ifx\null#2\null
3458     \ifx\null#3\null
3459       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3460       \def\@gls@checkescactual{\relax}%
3461     \else
3462       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3463         \@gls@quotechar\string\"@gls@actualchar
3464         \@gls@quotechar\string\"@gls@actualchar}%
3465       \def\@gls@checkescactual{\@gls@checkescactual#3\null}%
3466     \fi
3467   \else
3468     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3469       \@gls@quotechar\string\"@gls@actualchar}%
3470     \ifx\null#3\null
3471       \def\@gls@checkescactual{\@gls@checkescactual#2\?\?\null}%
3472     \else
3473       \def\@gls@checkescactual{\@gls@checkescactual#2\?#3\null}%
3474     \fi
3475   \fi
3476 \@@gls@checkescactual

```

3477 }

gls@checkeschar Similarly for \|:

```
3478 \def\@gls@checkeschar#1\|#2\|#3\null{%
3479   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3480   \toks@={#1}%
3481   \ifx\null#2\null
3482   \ifx\null#3\null
3483     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3484     \def\@gls@checkeschar{\relax}%
3485   \else
3486     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3487       \@gls@quotechar\string"\@gls@encapchar
3488       \@gls@quotechar\string"\@gls@encapchar}%
3489     \def\@gls@checkeschar{\@gls@checkeschar#3\null}%
3490   \fi
3491 \else
3492   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3493     \@gls@quotechar\string"\@gls@encapchar}%
3494   \ifx\null#3\null
3495     \def\@gls@checkeschar{\@gls@checkeschar#2\|\|\null}%
3496   \else
3497     \def\@gls@checkeschar{\@gls@checkeschar#2\|#3\null}%
3498   \fi
3499 \fi
3500 \@gls@checkeschar
3501 }
```

s@checkesclevel Similarly for \!:

```
3502 \def\@gls@checkesclevel#1\!#2\!#3\null{%
3503   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3504   \toks@={#1}%
3505   \ifx\null#2\null
3506   \ifx\null#3\null
3507     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3508     \def\@gls@checkesclevel{\relax}%
3509   \else
3510     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3511       \@gls@quotechar\string"\@gls@levelchar
3512       \@gls@quotechar\string"\@gls@levelchar}%
3513     \def\@gls@checkesclevel{\@gls@checkesclevel#3\null}%
3514   \fi
3515 \else
3516   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3517     \@gls@quotechar\string"\@gls@levelchar}%
3518   \ifx\null#3\null
3519     \def\@gls@checkesclevel{\@gls@checkesclevel#2\!\!\null}%
3520   \else
3521     \def\@gls@checkesclevel{\@gls@checkesclevel#2\!#3\null}%

```

```

3522 \fi
3523 \fi
3524 \@gls@checkesclevel
3525 }

```

\gls@checkbar and for |:

```

3526 \def\@gls@checkbar#1|#2|#3\null{%
3527 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3528 \toks@={#1}%
3529 \ifx\null#2\null
3530 \ifx\null#3\null
3531 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3532 \def\@gls@checkbar{\relax}%
3533 \else
3534 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3535 \@gls@quotechar\@gls@encapchar\@gls@quotechar\@gls@encapchar}%
3536 \def\@gls@checkbar{\@gls@checkbar#3\null}%
3537 \fi
3538 \else
3539 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3540 \@gls@quotechar\@gls@encapchar}%
3541 \ifx\null#3\null
3542 \def\@gls@checkbar{\@gls@checkbar#2||\null}%
3543 \else
3544 \def\@gls@checkbar{\@gls@checkbar#2|#3\null}%
3545 \fi
3546 \fi
3547 \@gls@checkbar
3548 }

```

@gls@checklevel and for !:

```

3549 \def\@gls@checklevel#1!#2!#3\null{%
3550 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3551 \toks@={#1}%
3552 \ifx\null#2\null
3553 \ifx\null#3\null
3554 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3555 \def\@gls@checklevel{\relax}%
3556 \else
3557 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3558 \@gls@quotechar\@gls@levelchar\@gls@quotechar\@gls@levelchar}%
3559 \def\@gls@checklevel{\@gls@checklevel#3\null}%
3560 \fi
3561 \else
3562 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3563 \@gls@quotechar\@gls@levelchar}%
3564 \ifx\null#3\null
3565 \def\@gls@checklevel{\@gls@checklevel#2!!\null}%
3566 \else

```

```

3567     \def\@gls@checklevel{\@gls@checklevel#2!#3\null}%
3568   \fi
3569 \fi
3570 \@gls@checklevel
3571 }

```

gls@checkactual and for ?:

```

3572 \def\@gls@checkactual#1?#2?#3\null{%
3573   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3574   \toks@={#1}%
3575   \ifx\null#2\null
3576     \ifx\null#3\null
3577       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3578       \def\@gls@checkactual{\relax}%
3579     \else
3580       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3581         \@gls@quotechar\@gls@actualchar\@gls@quotechar\@gls@actualchar}%
3582       \def\@gls@checkactual{\@gls@checkactual#3\null}%
3583     \fi
3584   \else
3585     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3586       \@gls@quotechar\@gls@actualchar}%
3587     \ifx\null#3\null
3588       \def\@gls@checkactual{\@gls@checkactual#2??\null}%
3589     \else
3590       \def\@gls@checkactual{\@gls@checkactual#2?#3\null}%
3591     \fi
3592   \fi
3593   \@gls@checkactual
3594 }

```

s@xdycheckquote As before but for use with xindy

```

3595 \def\@gls@xdycheckquote#1"#2"#3\null{%
3596   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3597   \toks@={#1}%
3598   \ifx\null#2\null
3599     \ifx\null#3\null
3600       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3601       \def\@gls@xdycheckquote{\relax}%
3602     \else
3603       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3604         \string"\string"}%
3605       \def\@gls@xdycheckquote{\@gls@xdycheckquote#3\null}%
3606     \fi
3607   \else
3608     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3609       \string"}%
3610     \ifx\null#3\null
3611       \def\@gls@xdycheckquote{\@gls@xdycheckquote#2""\null}%

```

```

3612 \else
3613 \def\@gls@xdycheckquote{\@gls@xdycheckquote#2"#3\null}%
3614 \fi
3615 \fi
3616 \@gls@xdycheckquote
3617 }

```

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

```

3618 \edef\def@gls@xdycheckbackslash{%
3619 \noexpand\def\noexpand\@gls@xdycheckbackslash##1\@backslashchar
3620 ##2\@backslashchar##3\noexpand\null{%
3621 \noexpand\@gls@tmpb=\noexpand\expandafter
3622 {\noexpand\@gls@checkedmkidx}%
3623 \noexpand\toks@={##1}%
3624 \noexpand\ifx\noexpand\null##2\noexpand\null
3625 \noexpand\ifx\noexpand\null##3\noexpand\null
3626 \noexpand\edef\noexpand\@gls@checkedmkidx{%
3627 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
3628 \noexpand\def\noexpand\@gls@xdycheckbackslash{\relax}%
3629 \noexpand\else
3630 \noexpand\edef\noexpand\@gls@checkedmkidx{%
3631 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
3632 \@backslashchar\@backslashchar\@backslashchar\@backslashchar}%
3633 \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3634 \noexpand\@gls@xdycheckbackslash##3\noexpand\null}%
3635 \noexpand\fi
3636 \noexpand\else
3637 \noexpand\edef\noexpand\@gls@checkedmkidx{%
3638 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
3639 \@backslashchar\@backslashchar}%
3640 \noexpand\ifx\noexpand\null##3\noexpand\null
3641 \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3642 \noexpand\@gls@xdycheckbackslash##2\@backslashchar
3643 \@backslashchar\noexpand\null}%
3644 \noexpand\else
3645 \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3646 \noexpand\@gls@xdycheckbackslash##2\@backslashchar
3647 ##3\noexpand\null}%
3648 \noexpand\fi
3649 \noexpand\fi
3650 \noexpand\@gls@xdycheckbackslash
3651 }%
3652 }

```

Now go ahead and define \@gls@xdycheckbackslash

```

3653 \def@gls@xdycheckbackslash

```

lsdohypertarget

```

3654 \newlength\gls@tmplen
3655 \newcommand*{\glsdohypertarget}[2]{%

```



```

3656 \@glsshowtarget{#1}%
3657 \settoheight{\gls@tmplen}{#2}%
3658 \raisebox{\gls@tmplen}{\hypertarget{#1}{}}#2%
3659 }

```

`\glsdohyperlink`

```

3660 \newcommand*\glsdohyperlink[2]{%
3661 \@glsshowtarget{#1}%
3662 \hyperlink{#1}{#2}%
3663 }

```

`\glsdonohyperlink`

```

3664 \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`.

```

3665 \ifcsundef{hyperlink}%
3666 {%
3667 \let\@glslink\glsdonohyperlink
3668 }%
3669 {%
3670 \let\@glslink\glsdohyperlink
3671 }

```

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

```

3672 \ifcsundef{hypertarget}%
3673 {%
3674 \let\@glstarget\@secondoftwo
3675 }%
3676 {%
3677 \let\@glstarget\glsdohypertarget
3678 }

```

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

`\glsdisablehyper`

```

3679 \newcommand{\glsdisablehyper}{%
3680 \KV@glslink@hyperfalse
3681 \let\@glslink\glsdonohyperlink
3682 \let\@glstarget\@secondoftwo
3683 }

```

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

`\glsenablehyper`

```

3684 \newcommand{\glsenablehyper}{%
3685 \KV@glslink@hypertrue

```

```

3686 \let\@glslink\glsdohyperlink
3687 \let\@glstarget\glsdohypertarget
3688 }

```

Provide some convenience commands if not already defined:

```

3689 \providecommand{\@firstofthree}[3]{#1}
3690 \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`

```

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

```

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

`\@gls`

```

3692 \newcommand*{\@gls}[2][ ]{%
3693   \new@ifnextchar[{\@gls@{#1}{#2}}{\@gls@{#1}{#2}[ ]}%
3694 }

```

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

```

3695 \def\@gls@#1#2[#3]{%
3696   \glsdoifexists{#2}%
3697   {%
3698     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
3699     \let\glsifplural\@secondoftwo
3700     \let\gls caps case\@firstofthree
3701     \let\gls custom text\@empty
3702     \def\glsinsert{#3}%

```

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

```

3703   \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.

```

3704   \@gls@link[#1]{#2}{\@gls@text}%

```

Indicate that this entry has now been used

```
3705 \ifKV@glslink@local
3706 \glslocalunset{#2}%
3707 \else
3708 \glsunset{#2}%
3709 \fi
3710 }%

3711 \glspostlinkhook
3712 }
```

`\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`

```
3713 \newrobustcmd*{\Gls}{\@Gls@hyp@opt\@Gls}
```

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

```
3714 \newcommand*{\@Gls}[2][\@Gls@hyp@opt\@Gls]{%
3715 \new@ifnextchar[\@Gls@{#1}{#2}]{\@Gls@{#1}{#2}[1]}%
3716 }
```

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

```
3717 \def\@Gls@#1#2[#3]{%
3718 \glsdoifexists{#2}%
3719 {%
3720 \let\do@glslink@checkfirsthyper\@glslink@checkfirsthyper

3721 \let\glsifplural\@secondoftwo
3722 \let\glsifscapscase\@secondofthree
3723 \let\glsifcustomtext\@empty
3724 \def\glsinsert{#3}%
```

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

```
3725 \def\@glo@text{\csname gls@\gls@type @entryfmt\endcsname}%
```

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

```
3726 \@glslink[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3727 \ifKV@glslink@local
3728 \glslocalunset{#2}%
3729 \else
3730 \glsunset{#2}%
3731 \fi
3732 }%
```

```

3733 \glspostlinkhook
3734 }

```

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

\GLS

```

3735 \newrobustcmd*{\GLS}{\@gls@hyp@opt\@GLS}

```

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

```

3736 \newcommand*{\@GLS}[2] [] {%
3737   \new@ifnextchar[{\@GLS@{#1}{#2}}{\@GLS@{#1}{#2} []}%
3738 }

```

\@GLS@ Read in the final optional argument:

```

3739 \def\@GLS@#1#2[#3] {%
3740   \glsdoifexists{#2}%
3741   {%
3742     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3743     \let\glsifplural\@secondoftwo
3744     \let\glsapscase\@thirdofthree
3745     \let\glscustomtext\@empty
3746     \def\glsinsert{#3}%

```

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

```

3747   \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.

```

3748   \@gls@link[#1]{#2}{\@glo@text}%

```

Indicate that this entry has now been used

```

3749   \ifKV@glslink@local
3750     \glslocalunset{#2}%
3751   \else
3752     \glsunset{#2}%
3753   \fi
3754 }%

```

```

3755 \glspostlinkhook
3756 }

```

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

\glspl

```

3757 \newrobustcmd*{\glspl}{\@gls@hyp@opt\@glspl}

```

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

```

3758 \newcommand*{\@glspl}[2] [] {%
3759   \new@ifnextchar[{\@glspl@{#1}{#2}}{\@glspl@{#1}{#2} []}%
3760 }

```

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

```
3761 \def\@glsp1@#1#2[#3]{%
3762   \glstoifexists{#2}%
3763   {%
3764     \let\do@gl@link@checkfirsthyper\@gl@link@checkfirsthyper

3765     \let\gl@ifplural\@firstoftwo
3766     \let\gl@scapscase\@firstofthree
3767     \let\glscustomtext\@empty
3768     \def\glinsert{#3}%
```

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

```
3769   \def\@glo@text{\csname gls@\glstyle @entryfmt\endcsname}%
```

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

```
3770   \@gl@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3771   \ifKV@gl@link@local
3772     \gl@localunset{#2}%
3773   \else
3774     \gl@unset{#2}%
3775   \fi
3776 }%

3777 \glspostlinkhook
3778 }
```

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

`\Glspl`

```
3779 \newrobustcmd*{\Glspl}{\@gl@hyp@opt\@Glspl}
```

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

```
3780 \newcommand*{\@Glspl}[2][ ]{%
3781   \new@ifnextchar[\@Glspl@{#1}{#2}}{\@Glspl@{#1}{#2}[ ]}%
3782 }
```

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

```
3783 \def\@Glspl@#1#2[#3]{%
3784   \glstoifexists{#2}%
3785   {%
3786     \let\do@gl@link@checkfirsthyper\@gl@link@checkfirsthyper

3787     \let\gl@ifplural\@firstoftwo
3788     \let\gl@scapscase\@secondofthree
3789     \let\glscustomtext\@empty
3790     \def\glinsert{#3}%
```

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

```
3791 \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.

```
3792 \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3793 \ifKV@glslink@local
```

```
3794 \glsllocalunset{#2}%
```

```
3795 \else
```

```
3796 \glsunset{#2}%
```

```
3797 \fi
```

```
3798 }%
```

```
3799 \glspostlinkhook
```

```
3800 }
```

\GLSp1 behaves like \glsp1 except that all the link text is converted to uppercase.

\GLSp1

```
3801 \newrobustcmd*{\GLSp1}{\@gls@hyp@opt\@GLSp1}
```

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

```
3802 \newcommand*{\@GLSp1}[2] [] {%
```

```
3803 \new@ifnextchar[{\@GLSp1@{#1}{#2}}{\@GLSp1@{#1}{#2} []}%
```

```
3804 }
```

\@GLSp1 Read in the final optional argument:

```
3805 \def\@GLSp1@#1#2[#3] {%
```

```
3806 \glstdoifexists{#2}%
```

```
3807 {%
```

```
3808 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
```

```
3809 \let\glsifplural\@firstoftwo
```

```
3810 \let\glscapscase\@thirdofthree
```

```
3811 \let\glscustomtext\@empty
```

```
3812 \def\glsinsert{#3}%
```

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

```
3813 \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.

```
3814 \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```

3815 \ifKV@glslink@local
3816 \glslocalunset{#2}%
3817 \else
3818 \glsunset{#2}%
3819 \fi
3820 }%

3821 \glspostlinkhook
3822 }

```

`\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:

```

3823 \newrobustcmd*{\glsdisp}{\@gls@hyp@opt\@glsdisp}

```

Defined the un-starred form.

`\@glsdisp`

```

3824 \newcommand*{\@glsdisp}[3][ ]{%
3825 \glsdoifexists{#2}{%

3826 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3827 \let\glsifplural\@secondoftwo
3828 \let\glscapscase\@firstofthree
3829 \def\glscustomtext{#3}%
3830 \def\glsinsert{}%

```

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

```

3831 \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.

```

3832 \@gls@link[#1]{#2}{\@glo@text}%

```

Indicate that this entry has now been used

```

3833 \ifKV@glslink@local
3834 \glslocalunset{#2}%
3835 \else
3836 \glsunset{#2}%
3837 \fi
3838 }%

3839 \glspostlinkhook
3840 }

```

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

```
3841 \newcommand*{\@gl@link@nocheckfirsthyper}{}
```

@gl@field@link

```
3842 \newcommand{\@gl@field@link}[3]{%
3843   \glstoifexists{#2}%
3844   {%
3845     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
3846     \@gl@link[#1]{#2}{#3}%
3847   }%

3848   \glspostlinkhook
3849 }
```

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

\glstext

```
3850 \newrobustcmd*{\glstext}{\@gl@hyp@opt\@glstext}
```

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

```
3851 \newcommand*{\@glstext}[2][ ]{%
3852   \new@ifnextchar[\@glstext@{#1}{#2}}{\@glstext@{#1}{#2}[ ]}}
```

Read in the final optional argument:

```
3853 \def\@glstext@#1#2[#3]{%
3854   \@gl@field@link{#1}{#2}{\glstentrytext{#2}{#3}}%
3855 }
```

\GLStext behaves like \glstext except the text is converted to uppercase.

\GLStext

```
3856 \newrobustcmd*{\GLStext}{\@gl@hyp@opt\@GLStext}
```

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

```
3857 \newcommand*{\@GLStext}[2][ ]{%
3858   \new@ifnextchar[\@GLStext@{#1}{#2}}{\@GLStext@{#1}{#2}[ ]}}
```

Read in the final optional argument:

```
3859 \def\@GLStext@#1#2[#3]{%
3860   \@gl@field@link{#1}{#2}{\mfirstucMakeUppercase{\glstentrytext{#2}{#3}}}%
3861 }
```

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

\Glstext

```
3862 \newrobustcmd*{\Glstext}{\@gl@hyp@opt\@Glstext}
```



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

```
3863 \newcommand*{\@Glstext}[2] [] {%
3864   \new@ifnextchar[{\@Glstext@{#1}{#2}}{\@Glstext@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3865 \def\@Glstext@#1#2[#3] {%
3866   \@gls@field@link{#1}{#2}{\glstrytext{#2}#3}%
3867 }
```

`\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`

```
3868 \newrobustcmd*{\glsfirst}{\@gls@hyp@opt\@glsfirst}
```

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

```
3869 \newcommand*{\@glsfirst}[2] [] {%
3870   \new@ifnextchar[{\@glsfirst@{#1}{#2}}{\@glsfirst@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3871 \def\@glsfirst@#1#2[#3] {%
3872   \@gls@field@link{#1}{#2}{\glstryfirst{#2}#3}%
3873 }
```

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

`\Glsfirst`

```
3874 \newrobustcmd*{\Glsfirst}{\@gls@hyp@opt\@Glsfirst}
```

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

```
3875 \newcommand*{\@Glsfirst}[2] [] {%
3876   \new@ifnextchar[{\@Glsfirst@{#1}{#2}}{\@Glsfirst@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3877 \def\@Glsfirst@#1#2[#3] {%
3878   \@gls@field@link{#1}{#2}{\glstryfirst{#2}#3}%
3879 }
```

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

`\GLSfirst`

```
3880 \newrobustcmd*{\GLSfirst}{\@gls@hyp@opt\@GLSfirst}
```

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

```
3881 \newcommand*{\@GLSfirst}[2] [] {%
3882   \new@ifnextchar[{\@GLSfirst@{#1}{#2}}{\@GLSfirst@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3883 \def\@GLSfirst@#1#2[#3] {%
3884   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glstryfirst{#2}#3}}%
3885 }
```

`\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`

```
3886 \newrobustcmd*{\glsplural}{\@gls@hyp@opt\@glsplural}
```

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

```
3887 \newcommand*{\@glsplural}[2] [] {%
```

```
3888   \new@ifnextchar[{\@glsplural@{#1}{#2}}{\@glsplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3889 \def\@glsplural@#1#2[#3]{%
```

```
3890   \@gls@field@link{#1}{#2}{\glsentryplural{#2}#3}%
```

```
3891 }
```

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

`\Glsplural`

```
3892 \newrobustcmd*{\Glsplural}{\@gls@hyp@opt\@Glsplural}
```

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

```
3893 \newcommand*{\@Glsplural}[2] [] {%
```

```
3894   \new@ifnextchar[{\@Glsplural@{#1}{#2}}{\@Glsplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3895 \def\@Glsplural@#1#2[#3]{%
```

```
3896   \@gls@field@link{#1}{#2}{\Glsentryplural{#2}#3}%
```

```
3897 }
```

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

`\GLSplural`

```
3898 \newrobustcmd*{\GLSplural}{\@gls@hyp@opt\@GLSplural}
```

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

```
3899 \newcommand*{\@GLSplural}[2] [] {%
```

```
3900   \new@ifnextchar[{\@GLSplural@{#1}{#2}}{\@GLSplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3901 \def\@GLSplural@#1#2[#3]{%
```

```
3902   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryplural{#2}#3}}%
```

```
3903 }
```

`\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`

```
3904 \newrobustcmd*{\glsfirstplural}{\@gls@hyp@opt\@glsfirstplural}
```

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

```
3905 \newcommand*{\@glsfirstplural}[2] [] {%
```

```
3906   \new@ifnextchar[{\@glsfirstplural@{#1}{#2}}{\@glsfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3907 \def\@glsfirstplural@#1#2[#3]{%
```

```
3908   \@gls@field@link{#1}{#2}{\glsentryfirstplural{#2}#3}%
```

```
3909 }
```

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

`\Glsfirstplural`

```
3910 \newrobustcmd*{\Glsfirstplural}{\@gls@hyp@opt\@Glsfirstplural}
```

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

```
3911 \newcommand*{\@Glsfirstplural}[2] [] {%
```

```
3912   \new@ifnextchar[{\@Glsfirstplural@{#1}{#2}}{\@Glsfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3913 \def\@Glsfirstplural@#1#2[#3] {%
```

```
3914   \@gls@field@link{#1}{#2}{\glstryfirstplural{#2}#3}%
```

```
3915 }
```

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

`\GLSfirstplural`

```
3916 \newrobustcmd*{\GLSfirstplural}{\@gls@hyp@opt\@GLSfirstplural}
```

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

```
3917 \newcommand*{\@GLSfirstplural}[2] [] {%
```

```
3918   \new@ifnextchar[{\@GLSfirstplural@{#1}{#2}}{\@GLSfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3919 \def\@GLSfirstplural@#1#2[#3] {%
```

```
3920   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glstryfirstplural{#2}#3}%
```

```
3921 }
```

`\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`

```
3922 \newrobustcmd*{\glsname}{\@gls@hyp@opt\@glsname}
```

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

```
3923 \newcommand*{\@glsname}[2] [] {%
```

```
3924   \new@ifnextchar[{\@glsname@{#1}{#2}}{\@glsname@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3925 \def\@glsname@#1#2[#3] {%
```

```
3926   \@gls@field@link{#1}{#2}{\glstryname{#2}#3}%
```

```
3927 }
```

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

`\Glsname`

```
3928 \newrobustcmd*{\Glsname}{\@gls@hyp@opt\@Glsname}
```

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

```
3929 \newcommand*{\@Glsname}[2] [] {%
```

```
3930   \new@ifnextchar[{\@Glsname@{#1}{#2}}{\@Glsname@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3931 \def\@Glsname#1#2[#3]{%
3932   \@gls@field@link{#1}{#2}{\Glsentryname{#2}#3}%
3933 }
```

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

\GLSname

```
3934 \newrobustcmd*{\GLSname}{\@gls@hyp@opt\@GLSname}
```

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

```
3935 \newcommand*{\@GLSname}[2][\%]
3936   \new@ifnextchar[{\@GLSname@{#1}{#2}}{\@GLSname@{#1}{#2}[]}]
```

Read in the final optional argument:

```
3937 \def\@GLSname#1#2[#3]{%
3938   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentryname{#2}#3}}%
3939 }
```

\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

```
3940 \newrobustcmd*{\glsdesc}{\@gls@hyp@opt\@glsdesc}
```

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

```
3941 \newcommand*{\@glsdesc}[2][\%]
3942   \new@ifnextchar[{\@glsdesc@{#1}{#2}}{\@glsdesc@{#1}{#2}[]}]
```

Read in the final optional argument:

```
3943 \def\@glsdesc#1#2[#3]{%
3944   \@gls@field@link{#1}{#2}{\Glsentrydesc{#2}#3}%
3945 }
```

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

\Glsdesc

```
3946 \newrobustcmd*{\Glsdesc}{\@gls@hyp@opt\@Glsdesc}
```

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

```
3947 \newcommand*{\@Glsdesc}[2][\%]
3948   \new@ifnextchar[{\@Glsdesc@{#1}{#2}}{\@Glsdesc@{#1}{#2}[]}]
```

Read in the final optional argument:

```
3949 \def\@Glsdesc#1#2[#3]{%
3950   \@gls@field@link{#1}{#2}{\Glsentrydesc{#2}#3}%
3951 }
```

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

\GLSdesc

```
3952 \newrobustcmd*{\GLSdesc}{\@gls@hyp@opt\@GLSdesc}
```

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

```
3953 \newcommand*{\@GLSdesc}[2] [] {%  
3954   \new@ifnextchar [{\@GLSdesc@{#1}{#2}}{\@GLSdesc@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3955 \def\@GLSdesc@#1#2[#3] {%  
3956   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrydesc{#2}#3}}%  
3957 }
```

`\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`

```
3958 \newrobustcmd*{\glsdescplural}{\@gls@hyp@opt\@glsdescplural}
```

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

```
3959 \newcommand*{\@glsdescplural}[2] [] {%  
3960   \new@ifnextchar [{\@glsdescplural@{#1}{#2}}{\@glsdescplural@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3961 \def\@glsdescplural@#1#2[#3] {%  
3962   \@gls@field@link{#1}{#2}{\glsentrydescplural{#2}#3}}%  
3963 }
```

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

`\Glsdescplural`

```
3964 \newrobustcmd*{\Glsdescplural}{\@gls@hyp@opt\@Glsdescplural}
```

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

```
3965 \newcommand*{\@Glsdescplural}[2] [] {%  
3966   \new@ifnextchar [{\@Glsdescplural@{#1}{#2}}{\@Glsdescplural@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3967 \def\@Glsdescplural@#1#2[#3] {%  
3968   \@gls@field@link{#1}{#2}{\Glsentrydescplural{#2}#3}}%  
3969 }
```

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

`\GLSdescplural`

```
3970 \newrobustcmd*{\GLSdescplural}{\@gls@hyp@opt\@GLSdescplural}
```

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

```
3971 \newcommand*{\@GLSdescplural}[2] [] {%  
3972   \new@ifnextchar [{\@GLSdescplural@{#1}{#2}}{\@GLSdescplural@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3973 \def\@GLSdescplural@#1#2[#3] {%  
3974   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrydescplural{#2}#3}}%  
3975 }
```

`\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`

```
3976 \newrobustcmd*{\glssymbol}{\@gls@hyp@opt\@glssymbol}
```

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

```
3977 \newcommand*{\@glssymbol}[2] [] {%
```

```
3978   \new@ifnextchar[{\@glssymbol@{#1}{#2}}{\@glssymbol@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3979 \def\@glssymbol@#1#2[#3] {%
```

```
3980   \@gls@field@link{#1}{#2}{\glsentrysymbol{#2}#3}%
```

```
3981 }
```

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

`\Glssymbol`

```
3982 \newrobustcmd*{\Glssymbol}{\@gls@hyp@opt\@Glssymbol}
```

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

```
3983 \newcommand*{\@Glssymbol}[2] [] {%
```

```
3984   \new@ifnextchar[{\@Glssymbol@{#1}{#2}}{\@Glssymbol@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3985 \def\@Glssymbol@#1#2[#3] {%
```

```
3986   \@gls@field@link{#1}{#2}{\glsentrysymbol{#2}#3}%
```

```
3987 }
```

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

`\GLSsymbol`

```
3988 \newrobustcmd*{\GLSsymbol}{\@gls@hyp@opt\@GLSsymbol}
```

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

```
3989 \newcommand*{\@GLSsymbol}[2] [] {%
```

```
3990   \new@ifnextchar[{\@GLSsymbol@{#1}{#2}}{\@GLSsymbol@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3991 \def\@GLSsymbol@#1#2[#3] {%
```

```
3992   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrysymbol{#2}#3}}%
```

```
3993 }
```

`\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`

```
3994 \newrobustcmd*{\glsymbolplural}{\@gls@hyp@opt\@glsymbolplural}
```

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

```
3995 \newcommand*{\@glsymbolplural}[2] [] {%
```

```
3996   \new@ifnextchar[{\@glsymbolplural@{#1}{#2}}{\@glsymbolplural@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3997 \def\@glssymbolplural@#1#2[#3]{%
3998   \@gls@field@link{#1}{#2}{\glstrysymbolplural{#2}#3}%
3999 }
```

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

`Glssymbolplural`

```
4000 \newrobustcmd*{\Glssymbolplural}{\@gls@hyp@opt\@Glssymbolplural}
```

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

```
4001 \newcommand*{\@Glssymbolplural}[2] [] {%
4002   \new@ifnextchar[{\@Glssymbolplural@{#1}{#2}}{\@Glssymbolplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4003 \def\@Glssymbolplural@#1#2[#3]{%
4004   \@gls@field@link{#1}{#2}{\glstrysymbolplural{#2}#3}%
4005 }
```

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

`GLSsymbolplural`

```
4006 \newrobustcmd*{\GLSsymbolplural}{\@gls@hyp@opt\@GLSsymbolplural}
```

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

```
4007 \newcommand*{\@GLSsymbolplural}[2] [] {%
4008   \new@ifnextchar[{\@GLSsymbolplural@{#1}{#2}}{\@GLSsymbolplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4009 \def\@GLSsymbolplural@#1#2[#3]{%
4010   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glstrysymbolplural{#2}#3}}%
4011 }
```

`\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`

```
4012 \newrobustcmd*{\glsuseri}{\@gls@hyp@opt\@glsuseri}
```

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

```
4013 \newcommand*{\@glsuseri}[2] [] {%
4014   \new@ifnextchar[{\@glsuseri@{#1}{#2}}{\@glsuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4015 \def\@glsuseri@#1#2[#3]{%
4016   \@gls@field@link{#1}{#2}{\glstryuseri{#2}#3}%
4017 }
```

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

\Glsuseri

```
4018 \newrobustcmd*{\Glsuseri}{\@gls@hyp@opt\@Glsuseri}
```

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

```
4019 \newcommand*{\@Glsuseri}[2] [] {%
```

```
4020   \new@ifnextchar[{\@Glsuseri@{#1}{#2}}{\@Glsuseri@{#1}{#2} []}]
```

Read in the final optional argument:

```
4021 \def\@Glsuseri@#1#2[#3] {%
```

```
4022   \@gls@field@link{#1}{#2}{\Glsentryuseri{#2}#3}%
```

```
4023 }
```

\Glsuseri behaves like \glsuseri except that the link text is converted to uppercase.

\GLSuseri

```
4024 \newrobustcmd*{\GLSuseri}{\@gls@hyp@opt\@GLSuseri}
```

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

```
4025 \newcommand*{\@GLSuseri}[2] [] {%
```

```
4026   \new@ifnextchar[{\@GLSuseri@{#1}{#2}}{\@GLSuseri@{#1}{#2} []}]
```

Read in the final optional argument:

```
4027 \def\@GLSuseri@#1#2[#3] {%
```

```
4028   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentryuseri{#2}#3}}%
```

```
4029 }
```

\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

```
4030 \newrobustcmd*{\glsuserii}{\@gls@hyp@opt\@glsuserii}
```

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

```
4031 \newcommand*{\@glsuserii}[2] [] {%
```

```
4032   \new@ifnextchar[{\@glsuserii@{#1}{#2}}{\@glsuserii@{#1}{#2} []}]
```

Read in the final optional argument:

```
4033 \def\@glsuserii@#1#2[#3] {%
```

```
4034   \@gls@field@link{#1}{#2}{\Glsentryuserii{#2}#3}%
```

```
4035 }
```

\Glsuserii behaves like \glsuserii except that the first letter is converted to uppercase.

\Glsuserii

```
4036 \newrobustcmd*{\Glsuserii}{\@gls@hyp@opt\@Glsuserii}
```

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

```
4037 \newcommand*{\@Glsuserii}[2] [] {%
```

```
4038   \new@ifnextchar[{\@Glsuserii@{#1}{#2}}{\@Glsuserii@{#1}{#2} []}]
```

Read in the final optional argument:

```
4039 \def\@Glsuserii@#1#2[#3] {%
```

```
4040   \@gls@field@link{#1}{#2}{\Glsentryuserii{#2}#3}%
```

```
4041 }
```



\GLSuserii behaves like \glsuserii except that the link text is converted to uppercase.

\GLSuserii

```
4042 \newrobustcmd*{\GLSuserii}{\@gls@hyp@opt\@GLSuserii}
```

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

```
4043 \newcommand*{\@GLSuserii}[2] [] {%
```

```
4044   \new@ifnextchar[{\@GLSuserii@{#1}{#2}}{\@GLSuserii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4045 \def\@GLSuserii@#1#2[#3] {%
```

```
4046   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuserii{#2}#3}}%
```

```
4047 }
```

\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

```
4048 \newrobustcmd*{\glsuseriii}{\@gls@hyp@opt\@glsuseriii}
```

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

```
4049 \newcommand*{\@glsuseriii}[2] [] {%
```

```
4050   \new@ifnextchar[{\@glsuseriii@{#1}{#2}}{\@glsuseriii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4051 \def\@glsuseriii@#1#2[#3] {%
```

```
4052   \@gls@field@link{#1}{#2}{\glsentryuseriii{#2}#3}}%
```

```
4053 }
```

\Glsuseriii behaves like \glsuseriii except that the first letter is converted to uppercase.

\Glsuseriii

```
4054 \newrobustcmd*{\Glsuseriii}{\@gls@hyp@opt\@Glsuseriii}
```

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

```
4055 \newcommand*{\@Glsuseriii}[2] [] {%
```

```
4056   \new@ifnextchar[{\@Glsuseriii@{#1}{#2}}{\@Glsuseriii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4057 \def\@Glsuseriii@#1#2[#3] {%
```

```
4058   \@gls@field@link{#1}{#2}{\Glsentryuseriii{#2}#3}}%
```

```
4059 }
```

\GLSuseriii behaves like \glsuseriii except that the link text is converted to uppercase.

\GLSuseriii

```
4060 \newrobustcmd*{\GLSuseriii}{\@gls@hyp@opt\@GLSuseriii}
```

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

```
4061 \newcommand*{\@GLSuseriii}[2] [] {%
```

```
4062   \new@ifnextchar[{\@GLSuseriii@{#1}{#2}}{\@GLSuseriii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4063 \def\@GLSuseriii@#1#2[#3]{%
4064   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseriii{#2}#3}}%
4065 }
```

\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

```
4066 \newrobustcmd*{\glsuseriv}{\@gls@hyp@opt\@glsuseriv}
```

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

```
4067 \newcommand*{\@glsuseriv}[2][\%]
4068   \new@ifnextchar[\@glsuseriv@{#1}{#2}]{\@glsuseriv@{#1}{#2}[]}}
```

Read in the final optional argument:

```
4069 \def\@glsuseriv@#1#2[#3]{%
4070   \@gls@field@link{#1}{#2}{\glsentryuseriv{#2}#3}%
4071 }
```

\Glsuseriv behaves like \glsuseriv except that the first letter is converted to uppercase.

\Glsuseriv

```
4072 \newrobustcmd*{\Glsuseriv}{\@gls@hyp@opt\@Glsuseriv}
```

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

```
4073 \newcommand*{\@Glsuseriv}[2][\%]
4074   \new@ifnextchar[\@Glsuseriv@{#1}{#2}]{\@Glsuseriv@{#1}{#2}[]}}
```

Read in the final optional argument:

```
4075 \def\@Glsuseriv@#1#2[#3]{%
4076   \@gls@field@link{#1}{#2}{\Glsentryuseriv{#2}#3}%
4077 }
```

\GLSuseriv behaves like \glsuseriv except that the link text is converted to uppercase.

\GLSuseriv

```
4078 \newrobustcmd*{\GLSuseriv}{\@gls@hyp@opt\@GLSuseriv}
```

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

```
4079 \newcommand*{\@GLSuseriv}[2][\%]
4080   \new@ifnextchar[\@GLSuseriv@{#1}{#2}]{\@GLSuseriv@{#1}{#2}[]}}
```

Read in the final optional argument:

```
4081 \def\@GLSuseriv@#1#2[#3]{%
4082   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseriv{#2}#3}}%
4083 }
```

\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

```
4084 \newrobustcmd*{\glsuserv}{\@gls@hyp@opt\@glsuserv}
```

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

```
4085 \newcommand*{\@glsuserv}[2] [] {%
4086   \new@ifnextchar [{\@glsuserv@{#1}{#2}}]{\@glsuserv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4087 \def\@glsuserv@#1#2[#3] {%
4088   \@gls@field@link{#1}{#2}{\glsentryuserv{#2}#3}%
4089 }
```

\Glsuserv behaves like \glsuserv except that the first letter is converted to uppercase.

\Glsuserv

```
4090 \newrobustcmd*{\Glsuserv}{\@gls@hyp@opt\@Glsuserv}
```

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

```
4091 \newcommand*{\@GLsuserv}[2] [] {%
4092   \new@ifnextchar [{\@GLsuserv@{#1}{#2}}]{\@GLsuserv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4093 \def\@GLsuserv@#1#2[#3] {%
4094   \@gls@field@link{#1}{#2}{\Glsentryuserv{#2}#3}%
4095 }
```

\GLSuserv behaves like \glsuserv except that the link text is converted to uppercase.

\GLSuserv

```
4096 \newrobustcmd*{\GLSuserv}{\@gls@hyp@opt\@GLSuserv}
```

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

```
4097 \newcommand*{\@GLSuserv}[2] [] {%
4098   \new@ifnextchar [{\@GLSuserv@{#1}{#2}}]{\@GLSuserv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4099 \def\@GLSuserv@#1#2[#3] {%
4100   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuserv{#2}#3}}%
4101 }
```

\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

```
4102 \newrobustcmd*{\glsuservi}{\@gls@hyp@opt\@glsuservi}
```

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

```
4103 \newcommand*{\@glsuservi}[2] [] {%
4104   \new@ifnextchar [{\@glsuservi@{#1}{#2}}]{\@glsuservi@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4105 \def\@glsuservi@#1#2[#3] {%
4106   \@gls@field@link{#1}{#2}{\glsentryuservi{#2}#3}%
4107 }
```

\Glsuservi behaves like \glsuservi except that the first letter is converted to uppercase.

\Glsuservi

```
4108 \newrobustcmd*{\Glsuservi}{\@gls@hyp@opt\@Glsuservi}
```

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

```
4109 \newcommand*{\@Glsuservi}[2][\@Glsuservi@{#1}{#2}]{\@Glsuservi@{#1}{#2}[]}}
```

```
4110 \new@ifnextchar[\@Glsuservi@{#1}{#2}]{\@Glsuservi@{#1}{#2}[]}}
```

Read in the final optional argument:

```
4111 \def\@Glsuservi@#1#2[#3]{%
```

```
4112 \@gls@field@link{#1}{#2}{\Glsentryuservi{#2}#3}%
```

```
4113 }
```

\GLSuservi behaves like \glsuservi except that the link text is converted to uppercase.

\GLSuservi

```
4114 \newrobustcmd*{\GLSuservi}{\@gls@hyp@opt\@GLSuservi}
```

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

```
4115 \newcommand*{\@GLSuservi}[2][\@GLSuservi@{#1}{#2}]{\@GLSuservi@{#1}{#2}[]}}
```

```
4116 \new@ifnextchar[\@GLSuservi@{#1}{#2}]{\@GLSuservi@{#1}{#2}[]}}
```

Read in the final optional argument:

```
4117 \def\@GLSuservi@#1#2[#3]{%
```

```
4118 \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentryuservi{#2}#3}}%
```

```
4119 }
```

Now deal with acronym related keys. First the short form:

\acrshort

```
4120 \newrobustcmd*{\acrshort}{\@gls@hyp@opt\@ns@acrshort}
```

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

```
4121 \newcommand*{\@ns@acrshort}[2][\@ns@acrshort@{#1}{#2}]{\@ns@acrshort@{#1}{#2}[]}}
```

```
4122 \new@ifnextchar[\@ns@acrshort@{#1}{#2}]{\@ns@acrshort@{#1}{#2}[]}}
```

```
4123 }
```

Read in the final optional argument:

```
4124 \def\@ns@acrshort#1#2[#3]{%
```

```
4125 \glsdoifexists{#2}%
```

```
4126 {%
```

```
4127 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
```

```
4128 \let\glsifplural\@secondoftwo
```

```
4129 \let\glsapscase\@firstofthree
```

```
4130 \let\glsinsert\@empty
```

```
4131 \def\glscustomtext{%
```

```
4132 \acronymfont{\Glsentryshort{#2}#3}%
```

```
4133 }%
```

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

```
4134 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
```

```
4135 }%
```

```

4136 \glspostlinkhook
4137 }

```

\Acrshort

```

4138 \newrobustcmd*{\Acrshort}{\@gls@hyp@opt\ns@Acrshort}

```

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

```

4139 \newcommand*{\ns@Acrshort}[2][\%
4140 \new@ifnextchar[\@Acrshort{#1}{#2}]{\@Acrshort{#1}{#2}[]}%
4141 }

```

Read in the final optional argument:

```

4142 \def\@Acrshort#1#2[#3]{%
4143 \glsdoifexists{#2}%
4144 {%
4145 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
4146 \def\glslabel{#2}%
4147 \let\glsifplural\@secondoftwo
4148 \let\glsapscase\@secondofthree
4149 \let\glsinsert\@empty
4150 \def\glscustomtext{%
4151 \acronymfont{\Glsentryshort{#2}}#3%
4152 }%

```

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

```

4153 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4154 }%
4155 \glspostlinkhook
4156 }

```

\ACRshort

```

4157 \newrobustcmd*{\ACRshort}{\@gls@hyp@opt\ns@ACRshort}

```

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

```

4158 \newcommand*{\ns@ACRshort}[2][\%
4159 \new@ifnextchar[\@ACRshort{#1}{#2}]{\@ACRshort{#1}{#2}[]}%
4160 }

```

Read in the final optional argument:

```

4161 \def\@ACRshort#1#2[#3]{%
4162 \glsdoifexists{#2}%
4163 {%
4164 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

```

```

4165 \def\glslabel{#2}%
4166 \let\glsifplural\@secondoftwo
4167 \let\glsapscase\@thirdofthree
4168 \let\glsinsert\@empty
4169 \def\glscustomtext{%
4170 \mfirstucMakeUppercase{\acronymfont{\glsentryshort{#2}}#3}%
4171 }%

```

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

```

4172 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4173 }%

4174 \glspostlinkhook
4175 }

```

Short plural:

\acrshortpl

```

4176 \newrobustcmd*{\acrshortpl}{\@gls@hyp@opt\@ns@acrshortpl}

```

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

```

4177 \newcommand*{\ns@acrshortpl}[2] [] {%
4178 \new@ifnextchar[{\@acrshortpl{#1}{#2}}{\@acrshortpl{#1}{#2} []}%
4179 }

```

Read in the final optional argument:

```

4180 \def\@acrshortpl#1#2[#3]{%
4181 \glsdoifexists{#2}%
4182 {%

4183 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

4184 \def\glslabel{#2}%
4185 \let\glsifplural\@firstoftwo
4186 \let\glsapscase\@firstofthree
4187 \let\glsinsert\@empty
4188 \def\glscustomtext{%
4189 \acronymfont{\glsentryshortpl{#2}}#3%
4190 }%

```

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

```

4191 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4192 }%

4193 \glspostlinkhook
4194 }

```

\Acrshortpl

```

4195 \newrobustcmd*{\Acrshortpl}{\@gls@hyp@opt\@ns@Acrshortpl}

```

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

```
4196 \newcommand*{\ns@Acrshortpl}[2][\%
4197   \new@ifnextchar[\@Acrshortpl{#1}{#2}]{\@Acrshortpl{#1}{#2}[]}%
4198 }
```

Read in the final optional argument:

```
4199 \def\@Acrshortpl#1#2[#3]{%
4200   \glsdoifexists{#2}%
4201   {%
4202     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4203     \def\glslabel{#2}%
4204     \let\glsifplural\@firstoftwo
4205     \let\glscapscase\@secondofthree
4206     \let\glsinsert\@empty
4207     \def\glscustomtext{%
4208       \acronymfont{\Glsentryshortpl{#2}}#3%
4209     }%
```

Call \@gl@link Note that \@gl@link sets \glstype.

```
4210   \@gl@link[#1]{#2}{\csname gls\@glstype @entryfmt\endcsname}%
4211   }%
4212   \glspostlinkhook
4213 }
```

\ACRshortpl

```
4214 \newrobustcmd*{\ACRshortpl}{\@gl@hyp@opt\ns@ACRshortpl}
```

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

```
4215 \newcommand*{\ns@ACRshortpl}[2][\%
4216   \new@ifnextchar[\@ACRshortpl{#1}{#2}]{\@ACRshortpl{#1}{#2}[]}%
4217 }
```

Read in the final optional argument:

```
4218 \def\@ACRshortpl#1#2[#3]{%
4219   \glsdoifexists{#2}%
4220   {%
4221     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4222     \def\glslabel{#2}%
4223     \let\glsifplural\@firstoftwo
4224     \let\glscapscase\@thirdofthree
4225     \let\glsinsert\@empty
4226     \def\glscustomtext{%
4227       \mfirstucMakeUppercase{\acronymfont{\Glsentryshortpl{#2}}#3}%
4228     }%
```

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

```
4229   \@gls@link[#1]{#2}{\csname gls@\glsstyle @entryfmt\endcsname}%  
4230 }%  
  
4231 \glspostlinkhook  
4232 }
```

\acrlong

```
4233 \newrobustcmd*{\acrlong}{\@gls@hyp@opt\@ns@acrlong}
```

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

```
4234 \newcommand*{\ns@acrlong}[2][\@gls@hyp@opt\@ns@acrlong]{%  
4235   \new@ifnextchar[\@acrlong{#1}{#2}}{\@acrlong{#1}{#2}[]}%  
4236 }
```

Read in the final optional argument:

```
4237 \def\@acrlong#1#2[#3]{%  
4238   \glsdoifexists{#2}%  
4239   {%  
  
4240     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper  
  
4241     \def\glslabel{#2}%  
4242     \let\glsifplural\@secondoftwo  
4243     \let\glsapscase\@firstofthree  
4244     \let\glsinsert\@empty
```

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

```
4245   \def\glscustomtext{%  
4246     \glsentrylong{#2}#3%  
4247   }%
```

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

```
4248   \@gls@link[#1]{#2}{\csname gls@\glsstyle @entryfmt\endcsname}%  
4249 }%  
  
4250 \glspostlinkhook  
4251 }
```

\Acrlong

```
4252 \newrobustcmd*{\Acrlong}{\@gls@hyp@opt\@ns@Acrlong}
```

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

```
4253 \newcommand*{\ns@Acrlong}[2][\@gls@hyp@opt\@ns@Acrlong]{%  
4254   \new@ifnextchar[\@Acrlong{#1}{#2}}{\@Acrlong{#1}{#2}[]}%  
4255 }
```

Read in the final optional argument:

```
4256 \def\@Acrlong#1#2[#3]{%  
4257   \glsdoifexists{#2}%  
4258   {%
```



```

4259 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4260 \def\glslabel{#2}%
4261 \let\gl@ifplural\@secondoftwo
4262 \let\gl@scapscase\@secondofthree
4263 \let\gl@insert\@empty

```

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

```

4264 \def\glscustomtext{%
4265 \gl@entrylong{#2}#3%
4266 }%

```

Call \@gl@link. Note that \@gl@link sets \glstyle.

```

4267 \@gl@link[#1]{#2}{\csname gls@\glstyle @entryfmt\endcsname}%
4268 }%

4269 \gl@postlinkhook
4270 }

```

\ACRlong

```

4271 \newrobustcmd*{\ACRlong}{\@gl@hyp@opt\ns@ACRlong}

```

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

```

4272 \newcommand*{\ns@ACRlong}[2][{}]{%
4273 \new@ifnextchar[{\@ACRlong{#1}{#2}}{\@ACRlong{#1}{#2}[]}%
4274 }

```

Read in the final optional argument:

```

4275 \def\@ACRlong#1#2[#3]{%
4276 \gl@doifexists{#2}%
4277 {%
4278 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper

4279 \def\glslabel{#2}%
4280 \let\gl@ifplural\@secondoftwo
4281 \let\gl@scapscase\@thirdofthree
4282 \let\gl@insert\@empty

```

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

```

4283 \def\glscustomtext{%
4284 \mfirstucMakeUppercase{\gl@entrylong{#2}#3}%
4285 }%

```

Call \@gl@link. Note that \@gl@link sets \glstyle.

```

4286 \@gl@link[#1]{#2}{\csname gls@\glstyle @entryfmt\endcsname}%
4287 }%

4288 \gl@postlinkhook
4289 }

```

Short plural:

`\acrlongpl`

```
4290 \newrobustcmd*{\acrlongpl}{\@gls@hyp@opt\@ns@acrlongpl}
```

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

```
4291 \newcommand*{\ns@acrlongpl}[2][\%  
4292   \new@ifnextchar[{\@acrlongpl{#1}{#2}}{\@acrlongpl{#1}{#2}[]}]%  
4293 }
```

Read in the final optional argument:

```
4294 \def\@acrlongpl#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\glscapscase\@firstofthree  
4301     \let\glsinsert\@empty
```

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

```
4302   \def\glscustomtext{%  
4303     \glsentrylongpl{#2}#3%  
4304   }%
```

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

```
4305   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%  
4306   }%  
  
4307   \glspostlinkhook  
4308 }
```

`\Acrlongpl`

```
4309 \newrobustcmd*{\Acrlongpl}{\@gls@hyp@opt\@ns@Acrlongpl}
```

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

```
4310 \newcommand*{\ns@Acrlongpl}[2][\%  
4311   \new@ifnextchar[{\@Acrlongpl{#1}{#2}}{\@Acrlongpl{#1}{#2}[]}]%  
4312 }
```

Read in the final optional argument:

```
4313 \def\@Acrlongpl#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\glsupcase\@secondofthree
4320 \let\glsinsert\@empty

```

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

```

4321 \def\glscustomtext{%
4322 \Glsentrylongpl{#2}#3%
4323 }%

```

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

```

4324 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4325 }%

```

```

4326 \glspostlinkhook
4327 }

```

\ACRlongpl

```

4328 \newrobustcmd*{\ACRlongpl}{\@gls@hyp@opt\@ns@ACRlongpl}

```

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

```

4329 \newcommand*{\ns@ACRlongpl}[2][{}]{%
4330 \new@ifnextchar[\@ACRlongpl{#1}{#2}}{\@ACRlongpl{#1}{#2}[]}%
4331 }

```

Read in the final optional argument:

```

4332 \def\@ACRlongpl#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\glsupcase\@thirdofthree
4339 \let\glsinsert\@empty

```

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

```

4340 \def\glscustomtext{%
4341 \mfirstucMakeUppercase{\Glsentrylongpl{#2}#3}%
4342 }%

```

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

```

4343 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4344 }%

```

```

4345 \glspostlinkhook
4346 }

```

## Displaying entry details without adding information to the glossary

These commands merely display entry information without adding entries in the associated file or having hyperlinks.

`gls@entry@field` Generic version.

```
\@gls@entry@field{\<label>}{\<field>}
```

```
4347 \newcommand*{\@gls@entry@field}[2]{%
4348   \csname glo@glsdetoklabel{#1}@#2\endcsname
4349 }
```

`glsletentryfield`

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

```
4350 \newcommand*{\glsletentryfield}[3]{%
4351   \letcs{#1}{glo@glsdetoklabel{#2}@#3}%
4352 }
```

`Gls@entry@field` Generic first letter uppercase version.

```
\@Gls@entry@field{\<label>}{\<field>}
```

```
4353 \newcommand*{\@Gls@entry@field}[2]{%
4354   \glsdoifexistsordo{#1}%
4355   {%
4356     \letcs\@glo@text{glo@glsdetoklabel{#1}@#2}%
4357     \ifdef\@glo@text
4358     {%
4359       \xmakefirstuc{\@glo@text}%
4360     }%
4361     {%
4362       ??\PackageError{glossaries}{The field ‘#2’ doesn’t exist for glossary
4363       entry ‘\glsdetoklabel{#1}’}{Check you have correctly spelt the entry
4364       label and the field name}%
4365     }%
4366   }%
4367   {%
4368     ??%
4369   }%
4370 }
```

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
4371 \newcommand*{\glsentryname}[1]{\@gls@entry@field{#1}{name}}

\Glsentryname
4372 \newrobustcmd*{\Glsentryname}[1]{%
4373   \@Gls@entryname{#1}%
4374 }

\@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:
4375 \newcommand*{\@Gls@entryname}[1]{%
4376   \@Gls@entry@field{#1}{name}%
4377 }

ls@acentryname Now the behaviour when \setacronymstyle is used:
4378 \newcommand*{\@Gls@acentryname}[1]{%
4379   \ifglshaslong{#1}%
4380   {%
4381     \letcs\@glo@text{glo@\glsdetoklabel{#1}@name}%
4382     \expandafter\@gls@getbody\@glo@text{}\@nil
4383     \expandafter\ifx\@gls@body\glsentrylong\relax
4384       \expandafter\Glsentrylong\@gls@rest
4385     \else
4386       \expandafter\ifx\@gls@body\glsentryshort\relax
4387         \expandafter\Glsentryshort\@gls@rest
4388       \else
4389         \expandafter\ifx\@gls@body\acronymfont\relax
          Temporarily make \glsentryshort behave like \Glsentryshort. (This is on the assump-
          tion that the argument of \acronymfont is \glsentryshort{\langle label \rangle}, as that's the behaviour
          of the predefined acronym styles.) This is scoped to localise the effect of the assignment.
4390         {%
4391           \let\glsentryshort\Glsentryshort
4392           \@glo@text
4393         }%
4394       \else
4395         \xmakefirstuc{\@glo@text}%
4396       \fi
4397     \fi
4398   \fi
4399 }%
4400 {%

Not an acronym
4401   \@Gls@entry@field{#1}{name}%
4402 }%
4403 }

```

Get the entry description (as specified by the description when the entry was defined). The argument is the label associated with the entry. Note that unless you used `description=false` in the `sanitize` package option you may get unexpected results if the description key contained any commands.

`\glentrydesc`

```
4404 \newcommand*{\glentrydesc}[1]{\@gls@entry@field{#1}{desc}}
```

`\Glsentrydesc`

```
4405 \newrobustcmd*{\Glsentrydesc}[1]{%
4406   \@Gls@entry@field{#1}{desc}%
4407 }
```

Plural form:

`entrydescplural`

```
4408 \newcommand*{\glentrydescplural}[1]{%
4409   \@gls@entry@field{#1}{descplural}%
4410 }
```

`entrydescplural`

```
4411 \newrobustcmd*{\Glsentrydescplural}[1]{%
4412   \@Gls@entry@field{#1}{descplural}%
4413 }
```

Get the entry text, as specified by the text key when the entry was defined. The argument is the label associated with the entry:

`\glentrytext`

```
4414 \newcommand*{\glentrytext}[1]{\@gls@entry@field{#1}{text}}
```

`\Glsentrytext`

```
4415 \newrobustcmd*{\Glsentrytext}[1]{%
4416   \@Gls@entry@field{#1}{text}%
4417 }
```

Get the plural form:

`\glentryplural`

```
4418 \newcommand*{\glentryplural}[1]{%
4419   \@gls@entry@field{#1}{plural}%
4420 }
```

`\Glsentryplural`

```
4421 \newrobustcmd*{\Glsentryplural}[1]{%
4422   \@Gls@entry@field{#1}{plural}%
4423 }
```

Get the symbol associated with this entry. The argument is the label associated with the entry.

`\glsentrysymbol`

```
4424 \newcommand*{\glsentrysymbol}[1]{%
4425   \@gls@entry@field{#1}{symbol}%
4426 }
```

`\Glsentrysymbol`

```
4427 \newrobustcmd*{\Glsentrysymbol}[1]{%
4428   \@Gls@entry@field{#1}{symbol}%
4429 }
```

Plural form:

`trysymbolplural`

```
4430 \newcommand*{\glsentrysymbolplural}[1]{%
4431   \@gls@entry@field{#1}{symbolplural}%
4432 }
```

`trysymbolplural`

```
4433 \newrobustcmd*{\Glsentrysymbolplural}[1]{%
4434   \@Gls@entry@field{#1}{symbolplural}%
4435 }
```

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`

```
4436 \newcommand*{\glsentryfirst}[1]{%
4437   \@gls@entry@field{#1}{first}%
4438 }
```

`\Glsentryfirst`

```
4439 \newrobustcmd*{\Glsentryfirst}[1]{%
4440   \@Gls@entry@field{#1}{first}%
4441 }
```

Get the plural form (as specified by the firstplural key when the entry was defined).

`ntryfirstplural`

```
4442 \newcommand*{\glsentryfirstplural}[1]{%
4443   \@gls@entry@field{#1}{firstpl}%
4444 }
```

`ntryfirstplural`

```
4445 \newrobustcmd*{\Glsentryfirstplural}[1]{%
4446   \@Gls@entry@field{#1}{firstpl}%
4447 }
```

sentrytitlecase

```
4448 \newrobustcmd*{\@glsentrytitlecase}[2]{%
4449   \glsfieldfetch{#1}{#2}{\@gls@value}%
4450   \xcapitalisewords{\@gls@value}%
4451 }
4452 \ifdef\texorpdfstring
4453 {
4454   \newcommand*{\glsentrytitlecase}[2]{%
4455     \texorpdfstring
4456       {\@glsentrytitlecase{#1}{#2}}%
4457     {\@gls@entry@field{#1}{#2}}%
4458   }
4459 }
4460 {
4461   \newcommand*{\glsentrytitlecase}[2]{\@glsentrytitlecase{#1}{#2}}
4462 }
```

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

\glsentrytype

```
4463 \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

```
4464 \newcommand*{\glsentrysort}[1]{%
4465   \@gls@entry@field{#1}{sort}%
4466 }
```

\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.

```
4467 \newcommand*{\glsentryuseri}[1]{%
4468   \@gls@entry@field{#1}{useri}%
4469 }
```

\Glsentryuseri

```
4470 \newrobustcmd*{\Glsentryuseri}[1]{%
4471   \@Gls@entry@field{#1}{useri}%
4472 }
```

\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.

```
4473 \newcommand*{\glsentryuserii}[1]{%
4474   \@gls@entry@field{#1}{userii}%
4475 }
```



```

\Glsentryuserii
4476 \newrobustcmd*{\Glsentryuserii}[1]{%
4477   \@Gls@entry@field{#1}{userii}%
4478 }

\glentryuseriii  Get the third user key (as specified by the user3 when the entry was defined). The argument
                  is the label associated with the entry.
4479 \newcommand*{\glentryuseriii}[1]{%
4480   \@Gls@entry@field{#1}{useriii}%
4481 }

\Glsentryuseriii
4482 \newrobustcmd*{\Glsentryuseriii}[1]{%
4483   \@Gls@entry@field{#1}{useriii}%
4484 }

\glentryuseriv   Get the fourth user key (as specified by the user4 when the entry was defined). The argument
                  is the label associated with the entry.
4485 \newcommand*{\glentryuseriv}[1]{%
4486   \@Gls@entry@field{#1}{useriv}%
4487 }

\Glsentryuseriv
4488 \newrobustcmd*{\Glsentryuseriv}[1]{%
4489   \@Gls@entry@field{#1}{useriv}%
4490 }

\glentryuserv    Get the fifth user key (as specified by the user5 when the entry was defined). The argument is
                  the label associated with the entry.
4491 \newcommand*{\glentryuserv}[1]{%
4492   \@Gls@entry@field{#1}{userv}%
4493 }

\Glsentryuserv
4494 \newrobustcmd*{\Glsentryuserv}[1]{%
4495   \@Gls@entry@field{#1}{userv}%
4496 }

\glentryuservi   Get the sixth user key (as specified by the user6 when the entry was defined). The argument
                  is the label associated with the entry.
4497 \newcommand*{\glentryuservi}[1]{%
4498   \@Gls@entry@field{#1}{uservi}%
4499 }

\Glsentryuservi
4500 \newrobustcmd*{\Glsentryuservi}[1]{%
4501   \@Gls@entry@field{#1}{uservi}%
4502 }

```

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

```
4503 \newcommand*{\glsentryshort}[1]{\@gls@entry@field{#1}{short}}
```

`\Glsentryshort`

```
4504 \newrobustcmd*{\Glsentryshort}[1]{%
4505   \@Gls@entry@field{#1}{short}%
4506 }
```

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

```
4507 \newcommand*{\glsentryshortpl}[1]{\@gls@entry@field{#1}{shortpl}}
```

`\Glsentryshortpl`

```
4508 \newrobustcmd*{\Glsentryshortpl}[1]{%
4509   \@Gls@entry@field{#1}{shortpl}%
4510 }
```

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

```
4511 \newcommand*{\glsentrylong}[1]{\@gls@entry@field{#1}{long}}
```

`\Glsentrylong`

```
4512 \newrobustcmd*{\Glsentrylong}[1]{%
4513   \@Gls@entry@field{#1}{long}%
4514 }
```

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

```
4515 \newcommand*{\glsentrylongpl}[1]{\@gls@entry@field{#1}{longpl}}
```

`\Glsentrylongpl`

```
4516 \newrobustcmd*{\Glsentrylongpl}[1]{%
4517   \@Gls@entry@field{#1}{longpl}%
4518 }
```

Short cut macros to access full form:

`\glsentryfull`

```
4519 \newcommand*{\glsentryfull}[1]{%
4520   \acrfullformat{\glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%
4521 }
```

`\Glsentryfull`

```
4522 \newrobustcmd*{\Glsentryfull}[1]{%
4523   \acrfullformat{\Glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%
4524 }
```

`\glsentryfullpl`

```
4525 \newcommand*{\glsentryfullpl}[1]{%
4526   \acrfullformat{\glsentrylongpl{#1}}{\acronymfont{\glsentryshortpl{#1}}}%
4527 }
```

`\Glsentryfullpl`

```
4528 \newrobustcmd*{\Glsentryfullpl}[1]{%
4529   \acrfullformat{\Glsentrylongpl{#1}}{\acronymfont{\glsentryshortpl{#1}}}%
4530 }
```

`entrynumberlist` Displays the number list as is.

```
4531 \newcommand*{\glsentrynumberlist}[1]{%
4532   \glsdoifexists{#1}%
4533   {%
4534     \@gls@entry@field{#1}{numberlist}%
4535   }%
4536 }
```

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

```
4537 \@ifpackageloaded{hyperref} {%
4538   \newcommand*{\glsdisplaynumberlist}[1]{%
4539     \GlossariesWarning
4540     {%
4541       \string\glsdisplaynumberlist\space
4542       doesn't work with hyperref.^^JUsing
4543       \string\glsentrynumberlist\space instead%
4544     }%
4545     \glsentrynumberlist{#1}%
4546   }%
4547 }%
4548 {%
4549   \newcommand*{\glsdisplaynumberlist}[1]{%
4550     \glsdoifexists{#1}%
4551     {%
4552       \bgroup
4553
4554       \edef\@glo@label{\glsdetoklabel{#1}}%
4555       \let\@org@glsglsnumberformat\glsglsnumberformat
4556       \def\glsglsnumberformat##1{##1}%
4557       \protected@edef\the@numberlist{%
4558         \csname glo@\@glo@label @numberlist\endcsname}%
4559       \def\@gls@numlist@sep{}%
4560       \def\@gls@numlist@nextsep{}%
4561       \def\@gls@numlist@lastsep{}%
4562       \def\@gls@thislist{}%
4563       \def\@gls@donext@def{}%
4564       \renewcommand\do[1]{%
4565         \protected@edef\@gls@thislist{%
4566           \@gls@thislist
```

```

4566         \noexpand\@gls@numlist@sep
4567         ##1%
4568     }%
4569     \let\@gls@numlist@sep\@gls@numlist@nextsep
4570     \def\@gls@numlist@nextsep{\glsnumlistsep}%
4571     \@gls@donext@def
4572     \def\@gls@donext@def{%
4573         \def\@gls@numlist@lastsep{\glsnumlistlastsep}%
4574     }%
4575 }%
4576 \expandafter \glsnumlistparser \expandafter{\the@numberlist}%
4577 \let\@gls@numlist@sep\@gls@numlist@lastsep
4578 \@gls@thislist
4579 \egroup
4580 }%
4581 }
4582 }

```

`\glsnumlistsep`

```

4583 \newcommand*{\glsnumlistsep}{, }

```

`\glsnumlistlastsep`

```

4584 \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.

```

4585 \newcommand*{\gls hyperlink}[2][\glsentrytext{\@glo@label}]{%
4586   \def\@glo@label{#2}%
4587   \@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`:

```

4588 \define@key{gloss add}{counter}{\def\@gls@counter{#1}}
4589 \define@key{gloss add}{format}{\def\@gls number format{#1}}

```

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

```

4590 \define@key{gloss add}{types}{\def\@glo@type{#1}}

```

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

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

`\glsadd`

```
4591 \newrobustcmd*{\glsadd}[2] [] {%
```

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

```
4592 \@gls@adjustmode
```

```
4593 \glsdoifexists{#2}%
```

```
4594 {%
```

```
4595 \def\@glsnumberformat{glsnumberformat}%
```

```
4596 \edef\@gls@counter{\csname glo@\glsdetoklabel{#2}@counter\endcsname}%
```

```
4597 \setkeys{glossadd}{#1}%
```

Store the entry's counter in `\theglsentrycounter`

```
4598 \@gls@saveentrycounter
```

Define sort key if necessary:

```
4599 \@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.

```
4600 \@do@wrglossary{#2}%
```

```
4601 }%
```

```
4602 }
```

`@gls@adjustmode`

```
4603 \newcommand*{\@gls@adjustmode}{}%
```

```
4604 \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`

```
4605 \newrobustcmd*{\glsaddall}[1] [] {%
```

```
4606 \edef\@glo@type{\@glo@types}%
```

```
4607 \setkeys{glossadd}{#1}%
```

```
4608 \forallglsentries[\@glo@type]{\@glo@entry}{%
```

```
4609 \glsadd[#1]{\@glo@entry}%
```

```
4610 }%
```

```
4611 }
```

`\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.

```
4612 \newrobustcmd*{\glsaddallunused}[1] [\@glo@types] {%
```

```

4613 \forallglsentries[#1]{\@glo@entry}%
4614 {%
4615     \ifglsused{\@glo@entry}{\glsadd[format=glsignore]{\@glo@entry}}%
4616 }%
4617 }

```

`\glsignore`

```

4618 \newcommand*{\glsignore}[1]{}

```

## 1.13 Creating associated files

The `\writeist` command creates the associated customized `.ist` makeindex style file. While defining this command, some characters have their catcodes temporarily changed to ensure they get written to the `.ist` file correctly. The makeindex actual character (usually `@`) is redefined to be a `?`, to allow internal commands to be written to the glossary file output file.

The special characters are stored in `\@gls@actualchar`, `\@gls@encapchar`, `\@gls@levelchar` and `\@gls@quotechar` to make them easier to use later, but don't change these values, because the characters are encoded in the command definitions that are used to escape the special characters (which means that the user no longer needs to worry about makeindex special characters).

The symbols and numbers label for group headings are hardwired into the `.ist` file as `glsymbols` and `glsnumbers`, the group titles can be translated (so that `\glsymbolsgroupname` replaces `glsymbols` and `\glsnumbersgroupname` replaces `glsnumbers`) using the command `\glsgetgrouptitle` which is defined in `.` This is done to prevent any problem characters in `\glsymbolsgroupname` and `\glsnumbersgroupname` from breaking hyperlinks.

`\glsopenbrace` Define `\glsopenbrace` to make it easier to write an opening brace to a file.

```

4619 \edef\glsopenbrace{\expandafter\@gobble\string\{ }

```

`\glsclosebrace` Define `\glsclosebrace` to make it easier to write an opening brace to a file.

```

4620 \edef\glsclosebrace{\expandafter\@gobble\string\} }

```

`\glsbackslash` Define `\glsbackslash` to make it easier to write a backslash to a file.

```

4621 \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.

```

4622 \edef\glsquote#1{\string"#1\string"}

```

`\glsperscentchar` Define `\glsperscentchar` to make it easier to write a percent character to a file.

```

4623 \edef\glsperscentchar{\expandafter\@gobble\string\% }

```

`\glstildechar` Define `\glstildechar` to make it easier to write a tilde character to a file.

```

4624 \edef\glstildechar{\string~ }

```

`\@glsfirstletter` Define the first letter to come after the digits 0,...,9. Only required for xindy.

```
4625 \ifglsxindy
4626   \newcommand*{\@glsfirstletter}{A}
4627 \fi
```

`\@letterAfterDigits` Sets the first letter to come after the digits 0,...,9. The starred version sanitizes.

```
4628 \newcommand*{\GlsSetXdyFirstLetterAfterDigits}{%
4629   \@ifstar\s@GlsSetXdyFirstLetterAfterDigits\@GlsSetXdyFirstLetterAfterDigits}
4630 \ifglsxindy
4631   \newcommand*{\@GlsSetXdyFirstLetterAfterDigits}[1]{%
4632     \renewcommand*{\@glsfirstletter}{#1}}
4633   \newcommand*{\s@GlsSetXdyFirstLetterAfterDigits}[1]{%
4634     \renewcommand*{\@glsfirstletter}{#1}%
4635     \@onelevel@sanitize\@glsfirstletter
4636   }
4637 \else
4638   \newcommand*{\@GlsSetXdyFirstLetterAfterDigits}[1]{%
4639     \glsnxindywarning\GlsSetXdyFirstLetterAfterDigits}
4640   \newcommand*{\s@GlsSetXdyFirstLetterAfterDigits}{%
4641     \@GlsSetXdyFirstLetterAfterDigits
4642   }
4643 \fi
```

`\@numbergrouporder` Specifies the order of the number group.

```
4644 \ifglsxindy
4645   \newcommand*{\@xdynumbergrouporder}{:before \string"\@glsfirstletter\string"}
4646 \fi
```

`\@numberGroupOrder` Sets the relative location of the number group. The starred version sanitizes.

```
4647 \newcommand*{\GlsSetXdyNumberGroupOrder}[1]{%
4648   \@ifstar\s@GlsSetXdyNumberGroupOrder\@GlsSetXdyNumberGroupOrder
4649 }
4650 \ifglsxindy
4651   \newcommand*{\@GlsSetXdyNumberGroupOrder}[1]{%
4652     \renewcommand*{\@xdynumbergrouporder}{#1}%
4653   }
4654   \newcommand*{\s@GlsSetXdyNumberGroupOrder}[1]{%
4655     \renewcommand*{\@xdynumbergrouporder}{#1}%
4656     \@onelevel@sanitize\@xdynumbergrouporder
4657   }
4658 \else
4659   \newcommand*{\@GlsSetXdyNumberGroupOrder}[1]{%
4660     \glsnxindywarning\GlsSetXdyNumberGroupOrder}
4661   \newcommand*{\s@GlsSetXdyNumberGroupOrder}{%
4662     \@GlsSetXdyNumberGroupOrder}
4663 \fi
```

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

```
4664 \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.

```

4665 \ifglxindy
4666   \newcommand*\GlsSetXdyMinRangeLength[1]{%
4667     \renewcommand*\@glxminrange{#1}}
4668 \else
4669   \newcommand*\GlsSetXdyMinRangeLength[1]{%
4670     \glsnnoxindywarning\GlsSetXdyMinRangeLength}
4671 \fi

```

\writeist

```

4672 \ifglxindy
    Code to use if xindy is required.
4673   \def\writeist{%
    Define write register if not already defined
4674     \ifundef{\glswrite}{\newwrite\glswrite}{}%
    Update attributes list
4675     \@glx@addpredefinedattributes
    Open the file.
4676     \openout\glswrite=\istfilename
    Write header comment at the start of the file
4677     \write\glswrite{;; xindy style file created by the glossaries
4678       package}%
4679     \write\glswrite{;; for document '\jobname' on
4680       \the\year-\the\month-\the\day}%
    Specify the required styles
4681     \write\glswrite{^^J; required styles^^J}
4682     \@for\@xdystyle:=\@xdyrequiredstyles\do{%
4683       \ifx\@xdystyle\@empty
4684         \else
4685           \protected@write\glswrite{{(require
4686             \string"\@xdystyle.xdy\string")}}%
4687       \fi
4688     }%
    List the allowed attributes (possible values used by the format key)
4689     \write\glswrite{^^J%
4690       ; list of allowed attributes (number formats)^^J}%
4691     \write\glswrite{(define-attributes ((\@xdyattributes)))}%
    Define any additional alphabets
4692     \write\glswrite{^^J; user defined alphabets^^J}%
4693     \write\glswrite{\@xdyuseralphabets}%
    Define location classes.
4694     \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.

```
4695 \@for\@gls@classI:=\@gls@xdy@locationlist\do{%
```

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

```
4696 \protected@write\glswrite{}\{(define-location-class
4697 \string"\@gls@classI\string"^^J\space\space\space
4698 (
4699 :sep "{ }{"
4700 \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
4701 :sep "}"
4702 )
4703 ^^J\space\space\space
4704 :min-range-length \@glsminrange^^J%
4705 )
4706 }%
```

Nested iteration over all classes:

```
4707 {%
4708 \@for\@gls@classII:=\@gls@xdy@locationlist\do{%
4709 \protected@write\glswrite{}\{(define-location-class
4710 \string"\@gls@classII-\@gls@classI\string"
4711 ^^J\space\space\space
4712 (
4713 :sep "{ }{"
4714 \csname @gls@xdy@Lclass@\@gls@classII\endcsname\space
4715 :sep "{ }{"
4716 \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
4717 :sep "}"
4718 )
4719 ^^J\space\space\space
4720 :min-range-length \@glsminrange^^J%
4721 )
4722 }%
4723 }%
4724 }%
4725 }%
```

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

```
4726 \write\glswrite{^^J; user defined location classes}%
4727 \write\glswrite{\@xdyuserlocationdefs}%
```

Cross-reference class. (The unverified option is used as the cross-references are supplied using the list of labels along with the optional argument for `\glsseeformat` which xindy won't recognise.)

```
4728 \write\glswrite{^^J; define cross-reference class^^J}%
4729 \write\glswrite{(define-crossref-class \string"see\string"
4730 :unverified )}%
```

Define how cross-references should be displayed. This adds an empty set of braces after the cross-referencing information allowing for the final argument of `\glsseeformat` which

gets ignored. (When using `makeindex` this final argument contains the location information which is not required.)

```
4731 \write\glswrite{(markup-crossref-list
4732 :class \string"see\string"^^J\space\space\space
4733 :open \string"\string\glssseeformat\string"
4734 :close \string"{}\string")}%
```

Provide hook to write extra material here (used by `glossaries-extra` to define a `seealso` class).

```
4735 \@xdycrossrefhook
```

List the order to sort the classes.

```
4736 \write\glswrite{^^J; define the order of the location classes}%
4737 \write\glswrite{(define-location-class-order
4738 (\@xdylocationclassorder))}%
```

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

```
4739 \write\glswrite{^^J; define the glossary markup^^J}%

4740 \write\glswrite{(markup-index^^J\space\space\space
4741 :open \string"\string
4742 \glossarysection[\string\glossarytoctitle]{\string
4743 \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`)

```
4744 \@for\@this@ctr:=\@xdycounters\do{%
4745   {%
4746     \@for\@this@attr:=\@xdyattributelist\do{%
4747       \protected@write\glswrite{}{\string\providecommand*%
4748         \expandafter\string
4749         \csname glsX\@this@ctr X\@this@attr\endcsname[2]%
4750         {%
4751           \string\setentrycounter
4752           [\expandafter\@gobble\string\#1]{\@this@ctr}%
4753           \expandafter\string
4754           \csname\@this@attr\endcsname
4755           {\expandafter\@gobble\string\#2}%
4756         }%
4757       }%
4758     }%
4759   }%
4760 }%
```

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

```
4761 \write\glswrite{%
4762 \string\begin
4763 {theglossary}\string\glossaryheader\glstildechar n\string" ^^J\space
4764 \space\space:close \string"\glpercentchar\glstildechar n\string
4765 \end{theglossary}\string\glossarypostamble
4766 \glstildechar n\string" ^^J\space\space\space
4767 :tree}}%
```

Specify what to put between letter groups

```
4768 \write\glswrite{(markup-letter-group-list
4769 :sep \string"\string\glsgroupskip\glstildechar n\string")}%
```

Specify what to put between entries

```
4770 \write\glswrite{(markup-indexentry
4771 :open \string"\string\relax \string\glresetentrylist
4772 \glstildechar n\string")}%
```

Specify how to format entries

```
4773 \write\glswrite{(markup-locclass-list :open
4774 \string"\glsoopenbrace\string\glossaryentrynumbers
4775 \glsoopenbrace\string\relax\space \string"^^J\space\space\space
4776 :sep \string", \string"
4777 :close \string"\glsclosebrace\glsclosebrace\string")}%
```

Specify how to separate location numbers

```
4778 \write\glswrite{(markup-locref-list
4779 :sep \string"\string\delimN\space\string")}%
```

Specify how to indicate location ranges

```
4780 \write\glswrite{(markup-range
4781 :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.

```
4782 \@onelevel@sanitize\gls@suffiF
4783 \@onelevel@sanitize\gls@suffiFF
4784 \ifx\gls@suffiF\@empty
4785 \else
4786 \write\glswrite{(markup-range
4787 :close "\gls@suffiF" :length 1 :ignore-end)}%
4788 \fi
4789 \ifx\gls@suffiFF\@empty
4790 \else
4791 \write\glswrite{(markup-range
4792 :close "\gls@suffiFF" :length 2 :ignore-end)}%
4793 \fi
```

Specify how to format locations.

```
4794 \write\glswrite{^^J; define format to use for locations^^J}%
4795 \write\glswrite{\@xdylocref}%
```

Specify how to separate letter groups.

```
4796 \write\glswrite{^^J; define letter group list format^^J}%
4797 \write\glswrite{(markup-letter-group-list
4798 :sep \string"\string\glsgroupskip\glstildechar n\string")}%
```

Define letter group headings.

```
4799 \write\glswrite{^^J; letter group headings^^J}%
4800 \write\glswrite{(markup-letter-group
```

```

4801      :open-head \string"\string\glsgroupheading
4802      \glsopenbrace\string"^^J\space\space\space
4803      :close-head \string"\glsclosebrace\string"))%

Define additional letter groups.
4804      \write\glswrite{^^J; additional letter groups^^J}%
4805      \write\glswrite{\@xdylettergroups}%

Define additional sort rules
4806      \write\glswrite{^^J; additional sort rules^^J}
4807      \write\glswrite{\@xdysortrules}%

Hook for any additional information:
4808      \@gls@writeisthook

Close the style file
4809      \closeout\glswrite

Suppress any further calls.
4810      \let\writeist\relax
4811  }
4812 \else

Code to use if makeindex is required.
4813 \edef\@gls@actualchar{\string?}
4814 \edef\@gls@encapchar{\string|}
4815 \edef\@gls@levelchar{\string!}
4816 \edef\@gls@quotechar{\string"}%
4817 \let\GlsSetQuote\gls@nosetquote
4818 \def\writeist{\relax
4819 \ifundef{\glswrite}{\newwrite\glswrite}{}\relax
4820 \openout\glswrite=\istfilename
4821 \write\glswrite{\glspercentchar\space makeindex style file
4822 created by the glossaries package}
4823 \write\glswrite{\glspercentchar\space for document
4824 '\jobname' on \the\year-\the\month-\the\day}
4825 \write\glswrite{actual '@gls@actualchar'}
4826 \write\glswrite{encap '@gls@encapchar'}
4827 \write\glswrite{level '@gls@levelchar'}
4828 \write\glswrite{quote '@gls@quotechar'}
4829 \write\glswrite{keyword \string"\string\glossaryentry\string"}
4830 \write\glswrite{preamble \string"\string\glossarysection[\string
4831 \glossarytoctitle]{\string\glossarytitle}\string
4832 \glossarypreamble\string\n\string\begin{theglossary}\string
4833 \glossaryheader\string\n\string"}
4834 \write\glswrite{postamble \string"\string%\string\n\string
4835 \end{theglossary}\string\glossarypostamble\string\n
4836 \string"}
4837 \write\glswrite{group_skip \string"\string\glsgroupskip\string\n
4838 \string"}
4839 \write\glswrite{item_0 \string"\string%\string\n\string"}
4840 \write\glswrite{item_1 \string"\string%\string\n\string"}

```

```

4841 \write\glswrite{item_2 \string\string%\string\n\string}
4842 \write\glswrite{item_01 \string\string%\string\n\string}
4843 \write\glswrite{item_x1
4844 \string\string\relax \string\glresetentrylist\string\n
4845 \string}
4846 \write\glswrite{item_12 \string\string%\string\n\string}
4847 \write\glswrite{item_x2
4848 \string\string\relax \string\glresetentrylist\string\n
4849 \string}

4850 \write\glswrite{delim_0 \string\string\{\string
4851 \glossaryentrynumbers\string\{\string\relax \string}
4852 \write\glswrite{delim_1 \string\string\{\string
4853 \glossaryentrynumbers\string\{\string\relax \string}
4854 \write\glswrite{delim_2 \string\string\{\string
4855 \glossaryentrynumbers\string\{\string\relax \string}
4856 \write\glswrite{delim_t \string\string\}\string\}\string}
4857 \write\glswrite{delim_n \string\string\delimN \string}
4858 \write\glswrite{delim_r \string\string\delimR \string}
4859 \write\glswrite{headings_flag 1}
4860 \write\glswrite{heading_prefix
4861 \string\string\glsgroupheading\string\{\string}
4862 \write\glswrite{heading_suffix
4863 \string\string\}\string\relax
4864 \string\glresetentrylist \string}
4865 \write\glswrite{symhead_positive \string\glssymbols\string}
4866 \write\glswrite{numhead_positive \string\glnumbers\string}
4867 \write\glswrite{page_compositor \string\glscpositor\string}
4868 \@gls@escbsdq\gls@suffixF
4869 \@gls@escbsdq\gls@suffixFF
4870 \ifx\gls@suffixF\@empty
4871 \else
4872 \write\glswrite{suffix_2p \string\gls@suffixF\string}
4873 \fi
4874 \ifx\gls@suffixFF\@empty
4875 \else
4876 \write\glswrite{suffix_3p \string\gls@suffixFF\string}
4877 \fi

```

Hook for any additional information:

```
4878 \@gls@writeisthook
```

Close the file and disable \writeist.

```

4879 \closeout\glswrite
4880 \let\writeist\relax
4881 }
4882 \fi

```

**SetWriteIstHook** Allow user to append information to the style file.

```

4883 \newcommand*\GlsSetWriteIstHook}[1]{\renewcommand*\@gls@writeisthook}{#1}}
4884 \@onlypremake\GlsSetWriteIstHook

```

ls@writeisthook

```
4885 \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.

```
4886 \ifglxindy
4887 \newcommand*{\GlsSetQuote}[1]{\glsnomakeindexwarning\GlsSetQuote}
4888 \newcommand*{\gls@nosetquote}[1]{\glsnomakeindexwarning\GlsSetQuote}
4889 \else
4890 \newcommand*{\GlsSetQuote}[1]{\edef\@gls@quotechar{\string#1}}%
```

If German is in use, set the extra makeindex option so makeglossaries can pick it up.

```
4891 \@ifpackageloaded{tracklang}%
4892 {%
4893 \IfTrackedLanguage{german}%
4894 {%
4895 \def\@gls@extramakeindexopts{-g}%
4896 }%
4897 }%
4898 }%
4899 {}%
```

Need to redefine `\@gls@checkquote`

```
4900 \edef\@gls@docheckquotedef{%
4901 \noexpand\def\noexpand\@gls@checkquote####1#1####2#1####3\noexpand\null{%
4902 \noexpand\@gls@tmpb=\noexpand\expandafter{\noexpand\@gls@checkedmkidx}%
4903 \noexpand\toks@={####1}%
4904 \noexpand\ifx\noexpand\null####2\noexpand\null
4905 \noexpand\ifx\noexpand\null####3\noexpand\null
4906 \noexpand\edef\noexpand\@gls@checkedmkidx{%
4907 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
4908 \noexpand\def\noexpand\@gls@checkquote{\noexpand\relax}%
4909 \noexpand\else
4910 \noexpand\edef\noexpand\@gls@checkedmkidx{%
4911 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
4912 \noexpand\@gls@quotechar\noexpand\@gls@quotechar
4913 \noexpand\@gls@quotechar\noexpand\@gls@quotechar}%
4914 \noexpand\def\noexpand\@gls@checkquote{%
4915 \noexpand\@gls@checkquote####3\noexpand\null}%
4916 \noexpand\fi
4917 \noexpand\else
4918 \noexpand\edef\noexpand\@gls@checkedmkidx{%
4919 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
4920 \noexpand\@gls@quotechar\noexpand\@gls@quotechar}%
4921 \noexpand\ifx\noexpand\null####3\noexpand\null
4922 \noexpand\def\noexpand\@gls@checkquote{%
4923 \noexpand\@gls@checkquote####2#1#1\noexpand\null}%
4924 \noexpand\else
4925 \noexpand\def\noexpand\@gls@checkquote{%
4926 \noexpand\@gls@checkquote####2#1#1####3\noexpand\null}%
```

```

4927         \noexpand\fi
4928     \noexpand\fi
4929     \noexpand\@@gls@checkquote
4930 }%
4931 }%
4932 \@gls@docheckquotedef
4933 \edef\@gls@docheckquotedef{%
4934     \noexpand\renewcommand{\noexpand\@gls@checkmkidxchars}[1]{%
4935         \noexpand\def\noexpand\@gls@checkedmkidx{%
4936             \noexpand\expandafter\noexpand\@gls@checkquote####1\noexpand\@nil
4937             #1#1\noexpand\null
4938             \noexpand\expandafter\noexpand\@gls@updatechecked
4939             \noexpand\@gls@checkedmkidx{####1}%
4940             \noexpand\def\noexpand\@gls@checkedmkidx{%
4941                 \noexpand\expandafter\noexpand\@gls@checkescquote####1\noexpand\@nil
4942                 \expandonce{\csname#1\endcsname}\expandonce{\csname#1\endcsname}%
4943                 \noexpand\null
4944                 \noexpand\expandafter\noexpand\@gls@updatechecked
4945                 \noexpand\@gls@checkedmkidx{####1}%
4946                 \noexpand\def\noexpand\@gls@checkedmkidx{%
4947                     \noexpand\expandafter\noexpand\@gls@checkescactual####1\noexpand\@nil
4948                     \noexpand\?\noexpand\?\noexpand\null
4949                     \noexpand\expandafter\noexpand\@gls@updatechecked
4950                     \noexpand\@gls@checkedmkidx{####1}%
4951                     \noexpand\def\noexpand\@gls@checkedmkidx{%
4952                         \noexpand\expandafter\noexpand\@gls@checkactual####1\noexpand\@nil
4953                         \noexpand?\noexpand?\noexpand\null
4954                         \noexpand\expandafter\noexpand\@gls@updatechecked
4955                         \noexpand\@gls@checkedmkidx{####1}%
4956                         \noexpand\def\noexpand\@gls@checkedmkidx{%
4957                             \noexpand\expandafter\noexpand\@gls@checkbar####1\noexpand\@nil
4958                             \noexpand|\noexpand|\noexpand\null
4959                             \noexpand\expandafter\noexpand\@gls@updatechecked
4960                             \noexpand\@gls@checkedmkidx{####1}%
4961                             \noexpand\def\noexpand\@gls@checkedmkidx{%
4962                                 \noexpand\expandafter\noexpand\@gls@checkescbar####1\noexpand\@nil
4963                                 \noexpand\\|noexpand\\|noexpand\null
4964                                 \noexpand\expandafter\noexpand\@gls@updatechecked
4965                                 \noexpand\@gls@checkedmkidx{####1}%
4966                                 \noexpand\def\noexpand\@gls@checkedmkidx{%
4967                                     \noexpand\expandafter\noexpand\@gls@checklevel####1\noexpand\@nil
4968                                     \noexpand!\noexpand!\noexpand\null
4969                                     \noexpand\expandafter\noexpand\@gls@updatechecked
4970                                     \noexpand\@gls@checkedmkidx{####1}%
4971                                 }%
4972                             }%
4973                         \@gls@docheckquotedef
4974                     \edef\@gls@docheckquotedef{%
4975                         \noexpand\def\noexpand\@gls@checkescquote####1%

```

```

4976 \expandonce{\csname#1\endcsname}####2\expandonce{\csname#1\endcsname}%
4977 ####3\noexpand\null{%
4978 \noexpand\@gls@tmpb=\noexpand\expandafter{\noexpand\@gls@checkedmkidx}%
4979 \noexpand\toks@={####1}%
4980 \noexpand\ifx\noexpand\null####2\noexpand\null
4981 \noexpand\ifx\noexpand\null####3\noexpand\null
4982 \noexpand\edef\noexpand\@gls@checkedmkidx{%
4983 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
4984 \noexpand\def\noexpand\@gls@checkescquote{\noexpand\relax}%
4985 \noexpand\else
4986 \noexpand\edef\noexpand\@gls@checkedmkidx{%
4987 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
4988 \noexpand\@gls@quotechar\noexpand\string\expandonce{%
4989 \csname#1\endcsname}\noexpand\@gls@quotechar
4990 \noexpand\@gls@quotechar\noexpand\string\expandonce{%
4991 \csname#1\endcsname}\noexpand\@gls@quotechar}%
4992 \noexpand\def\noexpand\@gls@checkescquote{%
4993 \noexpand\@gls@checkescquote####3\noexpand\null}%
4994 \noexpand\fi
4995 \noexpand\else
4996 \noexpand\edef\noexpand\@gls@checkedmkidx{%
4997 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
4998 \noexpand\@gls@quotechar\noexpand\string
4999 \expandonce{\csname#1\endcsname}\noexpand\@gls@quotechar}%
5000 \noexpand\ifx\noexpand\null####3\noexpand\null
5001 \noexpand\def\noexpand\@gls@checkescquote{%
5002 \noexpand\@gls@checkescquote####2\expandonce{\csname#1\endcsname}%
5003 \expandonce{\csname#1\endcsname}\noexpand\null}%
5004 \noexpand\else
5005 \noexpand\def\noexpand\@gls@checkescquote{%
5006 \noexpand\@gls@checkescquote####2\expandonce{\csname#1\endcsname}%
5007 ####3\noexpand\null}%
5008 \noexpand\fi
5009 \noexpand\fi
5010 \noexpand\@gls@checkescquote
5011 }%
5012 }%
5013 \@gls@docheckquotedef
5014 }
5015 \newcommand*{\gls@nosetquote}[1]{\PackageError{glossaries}%
5016 {\string\GlsSetQuote\space not permitted here}%
5017 {Move \string\GlsSetQuote\space earlier in the preamble, as
5018 soon as possible after glossaries.sty has been loaded}}
5019 \fi

```

ramakeindexopts

```
5020 \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`

```
5021 \newcommand{\noist}{%
```

Update attributes list

```
5022 \@gls@addpredefinedattributes
```

```
5023 \let\writeist\relax
```

```
5024 }
```

`\@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`

```
5025 \newcommand*{\@makeglossary}[1]{%
```

```
5026 \ifglossaryexists{#1}%
```

```
5027 {%
```

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

```
5028 \ifglssavewrites
```

```
5029 \expandafter\newtoks\csname glo@#1filetok\endcsname
```

```
5030 \else
```

```
5031 \expandafter\newwrite\csname glo@#1file\endcsname
```

```
5032 \expandafter\@glsopenfile\csname glo@#1file\endcsname{#1}%
```

```
5033 \fi
```

```
5034 \@gls@renewglossary
```

```
5035 \writeist
```

```
5036 }%
```

```
5037 {%
```

```
5038 \PackageError{glossaries}%
```

```
5039 {Glossary type ‘#1’ not defined}%
```

```
5040 {New glossaries must be defined before using \string\makeglossary}%
```

```
5041 }%
```

```
5042 }
```

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

```
5043 \newcommand*{\@glsopenfile}[2]{%
```

```
5044 \immediate\openout#1=\jobname.\csname @glotype@#2out\endcsname
```

```
5045 \PackageInfo{glossaries}{Writing glossary file
```

```
5046 \jobname.\csname @glotype@#2out\endcsname}%
```

```
5047 }
```

`\@closegls`

```

5048 \newcommand*{\@closegls}[1]{%
5049   \closeout\csname glo@#1@file\endcsname
5050 }

```

\@gls@automake

```

5051 \ifglxindy
5052 \newcommand*{\@gls@automake}[1]{%
5053   \ifglossaryexists{#1}
5054   {%
5055     \@closegls{#1}%
5056     \ifdefstring{\glsorder}{letter}%
5057     {\def\@gls@order{-M ord/letorder }}%
5058     {\let\@gls@order\@empty}%
5059     \ifcsundef{\xdy@#1@language}%
5060     {\let\@gls@langmod\@xdy@main@language}%
5061     {\letcs\@gls@langmod{\xdy@#1@language}}%
5062     \edef\@gls@dothiswrite{\noexpand\write18{xindy
5063       -I xindy
5064       \@gls@order
5065       -L \@gls@langmod\space
5066       -M \gls@istfilebase\space
5067       -C \gls@codepage\space
5068       -t \jobname.\csuse{\gls@glotype@#1@log}
5069       -o \jobname.\csuse{\gls@glotype@#1@in}
5070       \jobname.\csuse{\gls@glotype@#1@out}}}%
5071     }%
5072     \@gls@dothiswrite
5073   }%
5074   {%
5075     \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
5076   }%
5077 }
5078 \else
5079 \newcommand*{\@gls@automake}[1]{%
5080   \ifglossaryexists{#1}
5081   {%
5082     \@closegls{#1}%
5083     \ifdefstring{\glsorder}{letter}%
5084     {\def\@gls@order{-l }}%
5085     {\let\@gls@order\@empty}%
5086     \edef\@gls@dothiswrite{\noexpand\write18{makeindex \@gls@order
5087       -s \istfilename\space
5088       -t \jobname.\csuse{\gls@glotype@#1@log}
5089       -o \jobname.\csuse{\gls@glotype@#1@in}
5090       \jobname.\csuse{\gls@glotype@#1@out}}}%
5091     }%
5092     \@gls@dothiswrite
5093   }%
5094   {%

```

```

5095     \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
5096   }%
5097 }
5098 \fi

```

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

```

5099 \newcommand*{\@warn@nomakeglossaries}{}

```

Only use this if warning if `\printglossary` has been used without `\makeglossaries`

```

5100 \newcommand*{\@warn@nomakeglossaries}{\@warn@nomakeglossaries}

```

`\makeglossaries` will use `\@makeglossary` for each glossary type that has been defined. New glossaries need to be defined before using `\makeglossary`, so have `\makeglossaries` redefine `\newglossary` to prevent it being used afterwards.

`\makeglossaries`

```

5101 \newcommand*{\makeglossaries}{%

```

Define the write used for style file also used for all other output files if `savewrites=true`.

```

5102   \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.

```

5103   \protected@write\@auxout{}{\string\providecommand\string\@glsorder[1]{} }

```

```

5104   \protected@write\@auxout{}{\string\providecommand\string\@istfilename[1]{} }

```

If `\@gls@extramakeindexopts` has been defined, write it:

```

5105   \ifundef\@gls@extramakeindexopts

```

```

5106   {}%

```

```

5107   {%

```

```

5108     \protected@write\@auxout{}{\string\providecommand

```

```

5109       \string\@gls@extramakeindexopts[1]{} }

```

```

5110     \protected@write\@auxout{}{\string\@gls@extramakeindexopts

```

```

5111       {\@gls@extramakeindexopts}}%

```

```

5112   }%

```

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

```

5113   \protected@write\@auxout{}{\string\@istfilename{\istfilename}}%

```

```

5114   \protected@write\@auxout{}{\string\@glsorder{\glsorder}}

```

Iterate through each glossary type and activate it.

```

5115   \@for\@glo@type:=\@glo@types\do{%

```

```

5116     \ifthenelse{\equal{\@glo@type}{}}{}{}{%

```

```

5117       \@makeglossary{\@glo@type}}%

```

```

5118   }%

```

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

```

5119   \renewcommand*\newglossary[4][]{%

```

```

5120     \PackageError{glossaries}{New glossaries

```

```

5121     must be created before \string\makeglossaries}{You need

```

```

5122     to move \string\makeglossaries\space after all your

```

```

5123     \string\newglossary\space commands}}%

```

Any subsequence instances of this command should have no effect

```
5124 \let\makeglossary\relax
5125 \let\makeglossary\relax
5126 \let\makeglossaries\relax
```

Disable all commands that have no effect after \makeglossaries

```
5127 \@disable@onlypremakeg
```

Allow see key:

```
5128 \let\gls@checkseeallowed\relax
```

Suppress warning about no \makeglossaries

```
5129 \let\warn@nomakeglossaries\relax
```

Activate warning about missing \printglossary

```
5130 \def\warn@noprintglossary{%
5131   \ifdefstring{\@glo@types}{,}%
5132   {%
5133     \GlossariesWarningNoLine{No glossaries have been defined}%
5134   }%
5135   {%
5136     \GlossariesWarningNoLine{No \string\printglossary\space
5137       or \string\printglossaries\space
5138       found. ^^J(Remove \string\makeglossaries\space if you
5139       don't want any glossaries.) ^^JThis document will not
5140       have a glossary}%
5141   }%
5142 }%
```

Declare list parser for \glsdisplaynumberlist

```
5143 \ifglssavenumberlist
5144   \edef\@gls@dodolistparser{\noexpand\DeclareListParser
5145     {\noexpand\glsnumlistparser}{\delimN}}%
5146   \@gls@dodolistparser
5147 \fi
```

Prevent user from also using \makenoidxglossaries

```
5148 \let\makenoidxglossaries\@no@makeglossaries
```

Prohibit sort key in printgloss family:

```
5149 \renewcommand*{\@printgloss@setsort}{%
5150   \let\@glo@assign@sortkey\@glo@no@assign@sortkey
5151 }%
```

Check the automake setting:

```
5152 \ifglsautomake
5153   \renewcommand*{\@gls@doautomake}{%
5154     \@for\@gls@type:=\@glo@types\do{%
5155       \ifdefempty{\@gls@type}{}%
5156       {\@gls@automake{\@gls@type}}%
5157     }%
5158   }%
5159 \fi
```

Check the sort setting:

```
5160 \@glo@check@sortallowed\makeglossaries
5161 }
```

Must occur in the preamble:

```
5162 \@onlypreamble{\makeglossaries}
```

`\glswrite` The definition of `\glswrite` has now been moved to `\makeglossaries` so that it's only defined if needed.

The `\makeglossary` command is redefined to be identical to `\makeglossaries`. (This is done to reinforce the message that you must either use `\@makeglossary` for all the glossaries or for none of them.)

`\makeglossary`

```
5163 \let\makeglossary\makeglossaries
```

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

```
5164 \AtEndDocument{%
5165   \warn@nomakeglossaries
5166   \warn@noprintglossary
5167 }
```

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

```
5168 \newcommand*{\makenoidxglossaries}{%
```

Redefine empty glossary warning:

```
5169   \renewcommand{\@gls@noref@warn}[1]{%
5170     \GlossariesWarning{Empty glossary for
5171       \string\printnoidxglossary[type={##1}].
5172     Rerun may be required (or you may have forgotten to use
5173     commands like \string\gls)}}%
5174   }%
```

Don't escape makeindex/xindy characters

```
5175 \let\@gls@checkmkidxchars\@gobble
```

Write glossary information to aux instead of glossary files

```
5176 \let\@do@wrglossary\gls@noidxglossary
```

Switch on group headings that use the character code:

```
5177 \let\@gls@getgrouptitle\@gls@noidx@getgrouptitle
```

Allow see key:

```
5178 \let\gls@checkseeallowed\relax
```

Redefine cross-referencing macro:

```
5179 \renewcommand{\@do@seeglossary}[2]{%
5180   \edef\@gls@label{\glsdetoklabel{##1}}%
5181   \protected@write\@auxout{}{%
```

```

5182     \string\@gls@reference
5183     {\csname glo@\@gls@label @type\endcsname}%
5184     {\@gls@label}%
5185     {%
5186     \string\glsseeformat##2}%
5187     }%
5188 }%
5189 }%

```

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.

```

5190 \AtBeginDocument
5191 {%
5192 \write\@auxout{\string\providecommand\string\@gls@reference[3]{}}%
5193 }%

```

Change warning about no glossaries

```

5194 \def\warn@noprintglossary{%
5195 \GlossariesWarningNoLine{No \string\printnoidxglossary\space
5196 or \string\printnoidxglossaries ^^J
5197 found. (Remove \string\makenoidxglossaries\space if you
5198 don't want any glossaries.)^^JThis document will not have a glossary}%
5199 }%

```

Suppress warning about no \makeglossaries

```

5200 \let\warn@nomakeglossaries\relax

```

Prevent user from also using \makeglossaries

```

5201 \let\makeglossaries\@no@makeglossaries

```

Allow sort key in printgloss family:

```

5202 \renewcommand*{\@printgloss@setsort}{%
5203 \let\@glo@assign@sortkey\@glo@assign@sortkey

```

Initialise default sort order:

```

5204 \def\@glo@sorttype{\@glo@default@sorttype}%
5205 }%

```

All entries must be defined in the preamble:

```

5206 \renewcommand*\new@glossaryentry[2]{%
5207 \PackageError{glossaries}{Glossary entries must be
5208 defined in the preamble^^Jwhen you use
5209 \string\makenoidxglossaries}%
5210 {Either move your definitions to the preamble or use
5211 \string\makeglossaries}%
5212 }%

```

Redefine \glsentrynumberlist

```

5213 \renewcommand*\glsentrynumberlist[1]{%
5214 \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
5215 \ifdef\@gls@loclist
5216 {%

```

```

5217     \glsnoidxloclist{\@gls@loclist}%
5218 }%
5219 {%
5220     ??\glsdoifexists{##1}%
5221     {%
5222         \GlossariesWarning{Missing location list for ‘##1’. Either
5223             a rerun is required or you haven’t referenced the entry}%
5224     }%
5225 }%
5226 }%

```

Redefine \glsdisplaynumberlist

```

5227 \renewcommand*{\glsdisplaynumberlist}[1]{%
5228     \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
5229     \ifdef\@gls@loclist
5230     {%
5231         \def\@gls@noidxloclist@sep{%
5232             \def\@gls@noidxloclist@sep{%
5233                 \def\@gls@noidxloclist@sep{%
5234                     \glsnumlistsep
5235                 }%
5236                 \def\@gls@noidxloclist@finalsep{\glsnumlistlastsep}%
5237             }%
5238         }%
5239         \def\@gls@noidxloclist@finalsep{}%
5240         \def\@gls@noidxloclist@prev{}%
5241         \forlistloop{\glsnoidxdisplayloclisthandler}{\@gls@loclist}%
5242         \@gls@noidxloclist@finalsep
5243         \@gls@noidxloclist@prev
5244     }%
5245     {%
5246         ??\glsdoifexists{##1}%
5247         {%
5248             \GlossariesWarning{Missing location list for ‘##1’. Either
5249                 a rerun is required or you haven’t referenced the entry}%
5250         }%
5251     }%
5252 }%

```

Provide a generic way of iterating through the number list:

```

5253 \renewcommand*{\glsnumberlistloop}[3]{%
5254     \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
5255     \let\@gls@org@glsnoidxdisplayloc\glsnoidxdisplayloc
5256     \let\@gls@org@glsseeformat\glsseeformat
5257     \let\glsnoidxdisplayloc##2\relax
5258     \let\glsseeformat##3\relax
5259     \ifdef\@gls@loclist
5260     {%
5261         \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
5262     }%

```

```

5263   {%
5264     ??\glsdoifexists{##1}%
5265     {%
5266       \GlossariesWarning{Missing location list for ‘##1’. Either
5267         a rerun is required or you haven’t referenced the entry}%
5268     }%
5269   }%
5270   \let\glsnoidxdisplayloc\@gls@org@glsnoidxdisplayloc
5271   \let\glsseeformat\@gls@org@glsseeformat
5272 }%

```

Modify sanitize sort function

```

5273 \let\@gls@sanitizesort\@gls@noidx@sanitizesort
5274 \let\@gls@nosanitizesort\@gls@noidx@nosanitizesort
5275 \@gls@noidx@setsanitizesort

```

Check sort option allowed.

```

5276 \@gls@check@sortallowed\makenoidxglossaries
5277 }

```

Preamble-only command:

```

5278 \@onlypreamble{\makenoidxglossaries}

```

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

```

5279 \newcommand*{\glsnumberlistloop}[2]{%
5280   \PackageError{glossaries}{\string\glsnumberlistloop\space
5281     only works with \string\makenoidxglossaries}{}%
5282 }

```

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

```

5283 \newcommand*{\glsnoidxnumberlistloophandler}[1]{%
5284   #1%
5285 }

```

`\makeglossaries` Can’t use both `\makeglossaries` and `\makenoidxglossaries`

```

5286 \newcommand*{\@no@makeglossaries}{%
5287   \PackageError{glossaries}{You can’t use both
5288     \string\makeglossaries\space and \string\makenoidxglossaries}%
5289   {Either use one or other (or none) of those commands but not both
5290     together.}%
5291 }

```

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

```

5292 \newcommand{\@gls@noref@warn}[1]{%
5293   \GlossariesWarning{\string\makenoidxglossaries\space
5294     is required to make \string\printnoidxglossary[type={#1}] work}%
5295 }

```



s@noidxglossary Write the glossary information to the aux file:

```
5296 \newcommand*{\gls@noidxglossary}{%
5297   \protected@write\@auxout{}{%
5298     \string\gls@reference
5299     {\csname glo@\@gls@label @type\endcsname}%
5300     {\@gls@label}%
5301     {\string\glsnoidxdisplayloc
5302      {\@glo@counterprefix}%
5303      {\@gls@counter}%
5304      {\@glsnumberformat}%
5305      {\@glslocref}%
5306     }%
5307   }%
5308 }
```

## 1.14 Writing information to associated files

\istfile Deprecated.

```
5309 \def\istfile{\glswrite}
```

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

```
5310 \AtEndDocument{%
5311   \glswritefiles
5312 }
```

\@glswritefiles Only write the files if savewrites=true

```
5313 \newcommand*{\@glswritefiles}{%
```

Iterate through all the glossaries

```
5314   \forallglossaries{\@glo@type}{%
```

Check for empty glossaries (patch provided by Patrick Häcker)

```
5315     \ifcsundef{glo@\@glo@type @filetok}%
5316     {%
5317       \def\gls@tmp{}%
5318     }%
5319     {%
5320       \edef\gls@tmp{\expandafter\the
5321         \csname glo@\@glo@type @filetok\endcsname}%
5322     }%
5323     \ifx\gls@tmp\@empty
5324       \ifx\@glo@type\glsdefaulttype
5325         \GlossariesWarningNoLine{Glossary ‘\@glo@type’ has no
5326           entries.^^JRemember to use package option ‘nomain’ if
5327 you
5328           don’t want to^^Juse the main glossary}%
5329       \else
5330         \GlossariesWarningNoLine{Glossary ‘\@glo@type’ has no
```

```

5331         entries}%
5332     \fi
5333 \else
5334     \@glsopenfile{\glswrite}{\@glo@type}%
5335     \immediate\write\glswrite{%
5336         \expandafter\the
5337         \csname glo@\@glo@type @filetok\endcsname}%
5338     \immediate\closeout\glswrite
5339 \fi
5340 }%
5341 }

```

As from v4.10, the `\glossary` command is used by the `glossaries` package. Since the user isn't expected to use this command (as `glossaries` takes care of the particular format required for `makeindex/xindy`) there's no need for a user level command. Using a custom internal command prevents any conflict with other packages (and with the `\mark` mechanism).

In v4.10, the redefinition of `\glossary` was removed since it wasn't intended as a user level command, however it seems there are packages that have hacked the internal macros used by `glossaries` and no longer work with this redefinition removed, so it's been restored in v4.11 but is not used at all by `glossaries`. (This may be removed or moved to a compatibility mode in future.)

`\glossary`

```

5342 \if@gls@docloaded
5343 \else
5344   \renewcommand*\glossary[1][main]{\gls@glossary{#1}}
5345 \fi

```

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

`\gls@glossary`

```

5346 \newcommand*\gls@glossary[1]{%
5347   \@gls@glossary{#1}%
5348 }

```

`\@gls@glossary` `\@gls@glossary{<type>}{<indexing info>}`

(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 `\glpercentchar` would be needed with the sort sanitization switched off.) This means that it's safe to simply ignore the second argument.

```

5349 \newcommand*{\@gls@glossary}[2]{%
5350   \if@gls@debug
5351     \PackageInfo{glossaries}{wrglossary(#1)(#2)}%
5352   \fi
5353 }
```

This is a convenience command to set `\@gls@glossary`. It's used by `\@makeglossary` and then redefined to do nothing, as it only needs to be done once.

`s@renewglossary`

```

5354 \newcommand{\@gls@renewglossary}{%
5355   \gdef\@gls@glossary##1{\@bsphack\beginingroup\gls@wrglossary{##1}}%
5356   \let\@gls@renewglossary\@empty
5357 }
```

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`

```

5358 \newcommand*{\gls@wrglossary}[2]{%
5359   \ifglssavewrites
5360     \protected@edef\@gls@tmp{\the\csname glo@#1@filetok\endcsname#2}%
5361     \expandafter\global\expandafter\csname glo@#1@filetok\endcsname
5362       \expandafter{\@gls@tmp^^J}%
5363   \else
5364     \ifcsdef{glo@#1@file}%
5365       {%
5366         \expandafter\protected@write\csname glo@#1@file\endcsname{%
5367           \gls@disablepagerefexpansion}{#2}%
5368       }%
5369       {%
5370         \ifignoredglossary{#1}{}%
5371         {%
5372           \GlossariesWarning{No file defined for glossary ‘#1’}%
5373         }%
5374       }%
5375   \fi
5376   \endgroup\@esphack
5377 }
```

```

\do@wrglossary
5378 \newcommand*{\do@wrglossary}[1]{%
5379   \glswriteentry{#1}{\do@wrglossary{#1}}%
5380 }

\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.
5381 \newcommand*{\glswriteentry}[2]{%
5382   \ifglindexonlyfirst
5383     \ifglused{#1}{#2}%
5384   \else
5385     #2%
5386   \fi
5387 }

protected@pagefmts  List of page formats to be protected against expansion.
5388 \newcommand{\gls@protected@pagefmts}{%
5389   \gls@numberpage,\gls@alphpage,\gls@Alphpage,\gls@romanpage,\gls@Romanpage,\gls@arabicpage%
5390 }

pagerefexpansion
5391 \newcommand*{\gls@disablepagerefexpansion}{%
5392   \@for\@gls@this:=\gls@protected@pagefmts\do
5393   {%
5394     \expandafter\let\@gls@this\relax
5395   }%
5396 }

\gls@alphpage
5397 \newcommand*{\gls@alphpage}{\@alph\c@page}

\gls@Alphpage
5398 \newcommand*{\gls@Alphpage}{\@Alph\c@page}

\gls@numberpage
5399 \newcommand*{\gls@numberpage}{\number\c@page}

\gls@arabicpage
5400 \newcommand*{\gls@arabicpage}{\@arabic\c@page}

\gls@romanpage
5401 \newcommand*{\gls@romanpage}{\romannumeral\c@page}

\gls@Romanpage
5402 \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).

```
5403 \newcommand*{\glsaddprotectedpagefmt}[1]{%
5404   \eappto\gls@protected@pagefmts{,\expandonce{\csname gls#1page\endcsname}}%
5405   \csedef{gls#1page}{\expandonce{\csname#1\endcsname}\noexpand\c@page}%
5406   \eappto\@wrglossarynumberhook{%
5407     \noexpand\let\expandonce{\csname org@gls#1\endcsname}%
5408     \expandonce{\csname#1\endcsname}%
5409     \noexpand\def\expandonce{\csname#1\endcsname}{%
5410       \noexpand\@wrglossary@pageformat
5411       \expandonce{\csname gls#1page\endcsname}%
5412       \expandonce{\csname org@gls#1\endcsname}%
5413     }%
5414   }%
5415 }
```

ssarynumberhook Hook used by `\@do@wrglossary`

```
5416 \newcommand*{\@wrglossarynumberhook{}
```

sary@pageformat

```
5417 \newcommand{\@wrglossary@pageformat}[3]{%
5418   \ifx#3\c@page #1\else #2#3\fi
5419 }
```

@do@wrglossary Write the glossary entry in the appropriate format.

```
5420 \newcommand*{\@do@wrglossary}[1]{%
5421   \ifglsclocations
5422     \@do@esc@wrglossary{#1}%
5423   \else
5424     \@do@noesc@wrglossary{#1}%
5425   \fi
5426 }
```

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.

```
5427 \newcommand*{\@do@noesc@wrglossary}[1]{%
```

Don't fully expand yet.

```
5428 \expandafter\def\expandafter\@glsloc\ref\expandafter{\theglsentrycounter}%
5429 \expandafter\def\expandafter\@glsHloc\ref\expandafter{\theHglentrycounter}%
```

Find the prefix if `\@glsHloc\ref` and `\@glsloc\ref` aren't the same.

```
5430 \ifx\@glsHloc\ref\@glsloc\ref
5431   \def\@glo@counterprefix{}%
5432 \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 `\theglentrycounter` and `\theHglentrycounter` at this point, even if it hasn't been updated yet. The page number is not expected to occur in the prefix.)

```

5433 \protected@edef\@do@gl@getcounterprefix{\noexpand\@gl@getcounterprefix
5434 {\@gl@loc@ref}{\@gl@Hloc@ref}}%
5435 }%
5436 \do@gl@getcounterprefix
5437 \fi

```

De-tok label if required

```

5438 \edef\@gl@label{\gl@detoklabel{#1}}%

```

Write the information to file:

```

5439 \@@do@esc@wrglossary
5440 }

```

`\owprimitivemods` Conditional to determine whether or not `\@@do@esc@wrglossary` should be allowed to temporarily redefine `\the` and `\number`.

```

5441 \newif\ifgl@swrallowprimitivemods
5442 \gl@swrallowprimitivemodstrue

```

`\@esc@wrglossary` Write the glossary entry in the appropriate format. (Need to set `\@gl@numberformat` and `\@gl@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\csname 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).

```
5443 \newcommand*{\@@do@esc@wrglossary}[1]{% please read documented code!
```

```
5444 \begingroup
```

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

```
5445 \let\gls@orgthe\the
5446 \let\gls@orgnumber\number
5447 \let\gls@orgarabic\@arabic
5448 \let\gls@orgromannumeral\romannumeral
5449 \let\gls@orgalph\@alph
5450 \let\gls@orgAlph\@Alph
5451 \let\gls@orgRoman\@Roman
```

Redefine:

```
5452 \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.

```
5453 \def\gls@the##1{%
5454 \ifx##1\c@page \gls@numberpage\else\gls@orgthe##1\fi}%
5455 \def\the{\expandafter\gls@the}%
5456 \def\gls@number##1{%
5457 \ifx##1\c@page \gls@numberpage\else\gls@orgnumber##1\fi}%

```

```

5458     \def\number{\expandafter\gls@number}%
5459     \fi
5460     \def\@arabic##1{%
5461         \ifx##1\c@page \gls@arabicpage\else\gls@orgarabic##1\fi}%
5462     \def\romannumeral##1{%
5463         \ifx##1\c@page \gls@romanpage\else\gls@orgromannumeral##1\fi}%
5464     \def\@Roman##1{%
5465         \ifx##1\c@page \gls@Romanpage\else\gls@orgRoman##1\fi}%
5466     \def\@alph##1{%
5467         \ifx##1\c@page \gls@alphpage\else\gls@orgalph##1\fi}%
5468     \def\@Alph##1{%
5469         \ifx##1\c@page \gls@Alphpage\else\gls@orgAlph##1\fi}%

```

Add hook to allow for other number formats:

```
5470 \@wrglossarynumberhook
```

Prevent expansion:

```
5471 \gls@disablepagerefexpansion
```

Now store location in \@glslocref:

```

5472 \protected@xdef\@glslocref{\theHglentrycounter}%
5473 \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.

```
5474 \@gls@checkmkidxchars\@glslocref
```

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

```

5475 \expandafter\ifx\theHglentrycounter\theHglentrycounter\relax
5476     \def\@glo@counterprefix{}%
5477 \else
5478     \protected@edef\@glsHlocref{\theHglentrycounter}%
5479     \@gls@checkmkidxchars\@glsHlocref
5480     \edef\@do@gls@getcounterprefix{\noexpand\@gls@getcounterprefix
5481         {\@glslocref}{\@glsHlocref}%
5482     }%
5483     \@do@gls@getcounterprefix
5484 \fi

```

De-tok label if required

```
5485 \edef\@gls@label{\glsdetoklabel{#1}}%
```

Write the information to file:

```

5486 \@do@wrglossary
5487 }

```

@do@wrglossary

```
5488 \newcommand*{\@do@wrglossary}{%
```

Determine whether to use xindy or makeindex syntax

```
5489 \ifglxindy
```



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

```

5490 \expandafter\@glo@check@mkidxrangechar\@glsnumberformat\@nil
5491 \def\@glo@range{}%
5492 \expandafter\if\@glo@prefix\relax
5493 \def\@glo@range{:open-range}%
5494 \else
5495 \expandafter\if\@glo@prefix\relax
5496 \def\@glo@range{:close-range}%
5497 \fi
5498 \fi

```

Write to the glossary file using xindy syntax.

```

5499 \gls@glossary{\csname glo@\@gls@label @type\endcsname}{%
5500 (indexentry :tkey (\csname glo@\@gls@label @index\endcsname)

5501 :locref \string"\@glo@counterprefix}\@glslocref}\string" %
5502 :attr \string"\@gls@counter\@glo@suffix\string"
5503 \@glo@range
5504 )
5505 }%
5506 \else

```

Convert the format information into the format required for makeindex

```

5507 \@set@glo@numformat{\@glo@numfmt}{\@gls@counter}{\@glsnumberformat}%
5508 {\@glo@counterprefix}%

```

Write to the glossary file using makeindex syntax.

```

5509 \gls@glossary{\csname glo@\@gls@label @type\endcsname}{%
5510 \string\glossaryentry{\csname glo@\@gls@label @index\endcsname
5511 \@gls@encapchar\@glo@numfmt}{\@glslocref}}%
5512 \fi
5513 }

```

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.)

```

5514 \newcommand*\@gls@getcounterprefix[2]{%
5515 \edef\@gls@thisloc{#1}\edef\@gls@thisHloc{#2}%
5516 \ifx\@gls@thisloc\@gls@thisHloc
5517 \def\@glo@counterprefix{}%
5518 \else
5519 \def\@gls@get@counterprefix##1.#1##2\end@getprefix{%
5520 \def\@glo@tmp{##2}%
5521 \ifx\@glo@tmp\@empty
5522 \def\@glo@counterprefix{}%
5523 \else
5524 \def\@glo@counterprefix{##1}%
5525 \fi

```

```

5526 }%
5527 \@gls@get@counterprefix#2.#1\end@getprefix
Warn if no prefix can be formed.
5528 \ifx\@glo@counterprefix\@empty
5529 \GlossariesWarning{Hyper target ‘#2’ can’t be formed by
5530 prefixing^^Jlocation ‘#1’. You need to modify the
5531 definition of \string\theH\@gls@counter^^Jotherwise you
5532 will get the warning: “name{\@gls@counter.#1}’ has been^^J
5533 referenced but does not exist”}%
5534 \fi
5535 \fi
5536 }

```

## 1.15 Glossary Entry Cross-References

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

```

5537 \newcommand{\@do@seeglossary}[2]{%
5538 \def\@gls@xref{#2}%
5539 \@onelevel@sanitize\@gls@xref
5540 \@gls@checkmkidxchars\@gls@xref
5541 \ifglxsindy
5542 \gls@glossary{\csname glo@#1@type\endcsname}{%
5543 (indexentry
5544 :key (\csname glo@#1@index\endcsname)
5545 :xref (\string"\@gls@xref\string")
5546 :attr \string"see\string"
5547 )
5548 }%
5549 \else
5550 \gls@glossary{\csname glo@#1@type\endcsname}{%
5551 \string\glossaryentry{\csname glo@#1@index\endcsname
5552 \@gls@encapchar glsseeformat\@gls@xref}{Z}}%
5553 \fi
5554 }

```

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

```

5555 \def\@gls@fixbraces#1#2#3\@nil{%
5556 \ifx#2[\relax
5557 \@gls@fixbraces#1#2#3\@end@fixbraces
5558 \else
5559 \def#1{{#2#3}}%
5560 \fi
5561 }

```

`@@gls@fixbraces`

```

5562 \def\@gls@fixbraces#1[#2]#3\@end@fixbraces{%
5563   \def#1{[#2]{#3}}}%
5564 }

```

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

```

5565 \DeclareRobustCommand*\glssee}[3][\seename]{%
5566   \@do@seeglossary{#2}{#1}{#3}}
5567 \newcommand*\@glssee}[3][\seename]{%
5568   \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.

```

5569 \DeclareRobustCommand*\glsseeformat}[3][\seename]{%
5570   \emph{#1} \glsseelist{#2}}

```

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

```

5571 \DeclareRobustCommand*\glsseelist}[1]{%

```

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

```

5572   \let\@gls@dolast\relax

```

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

```

5573   \let\@gls@donext\relax

```

Iterate through the labels

```

5574   \@for\@gls@thislabel:=#1\do{%

```

Check if on last iteration of loop

```

5575     \ifx\@xfor@nextelement\@nnil
5576       \@gls@dolast
5577     \else
5578       \@gls@donext
5579     \fi

```

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

```

5580     \expandafter\glsseeitem\expandafter{\@gls@thislabel}%

```

Update separators

```

5581     \let\@gls@dolast\glsseelastsep
5582     \let\@gls@donext\glsseesep
5583   }%
5584 }

```

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

```

5585 \newcommand*\glsseelastsep{\space\andname\space}

```

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

```

5586 \newcommand*\glsseesep{, }

```

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

```
5587 \DeclareRobustCommand*{\glsseeitem}[1]{\glshyperlink[\glsseeitemformat{#1}]{#1}}
```

`\glsseeitemformat` As from v3.0, default is to use `\glstentrytext` instead of `\glstentryname`. (To avoid problems with the name key being sanitized, although this is no longer a problem now.)

```
5588 \newcommand*{\glsseeitemformat}[1]{\glstentrytext{#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.

```
5589 \newcommand*{\gls@save@numberlist}[1]{%
5590   \ifglssavenumberlist
5591     \toks@{#1}%
5592     \edef\@do@writeaux@info{%
5593       \noexpand\csgdef{glo@\glscurrententrylabel @numberlist}{\the\toks@}%
5594     }%
5595     \@onelevel@sanitize\@do@writeaux@info
5596     \protected@write\@auxout{}\@do@writeaux@info}%
5597   \fi
5598 }
```

`\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.)

```
5599 \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.

```
5600 \ifcsundef{printglossary}{}%
5601 {%
```

If `\printglossary` is already defined, issue a warning and undefine it.

```
5602   \@gls@warnonglossdefined
5603   \undef\printglossary
5604 }
```

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

```
5605 \newcommand*{\printglossary}[1][type=\glsdefaultttype]{%
5606   \@printglossary{#1}{\@print@glossary}%
5607 }
```

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`

```
5608 \newcommand*{\printglossaries}{%
5609   \forallglossaries{\@glo@type}{\printglossary[type=\@glo@type]}%
5610 }
```

`\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.

```
5611 \newcommand*{\printnoidxglossary}[1][type=\glsdefaulttype]{%
5612   \@printglossary{#1}{\@printnoidxglossary}%
5613 }
```

`\printnoidxglossaries` Analogous to `\printglossaries`

```
5614 \newcommand*{\printnoidxglossaries}{%
5615   \forallglossaries{\@glo@type}{\printnoidxglossary[type=\@glo@type]}%
5616 }
```

`\printgloss@setsort` Initialise to do nothing.

```
5617 \newcommand*{\@printgloss@setsort}{{}}
```

`\preglossaryhook`

```
5618 \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.

```
5619 \newcommand{\@printglossary}[2]{%
```

Set up defaults.

```
5620   \def\@glo@type{\glsdefaulttype}%
5621   \def\glossarytitle{\csname @glo@type\@glo@type @title\endcsname}%

5622   \def\glossarytoctitle{\glossarytitle}%
5623   \let\org@glossarytitle\glossarytitle

5624   \def\@glossarystyle{%
5625     \ifx\@glossary@default@style\relax
5626       \GlossariesWarning{No default glossary style provided \MessageBreak
5627         for the glossary '\@glo@type'. \MessageBreak
5628         Using deprecated fallback. \MessageBreak
5629         To fix this set the style with \MessageBreak
```

```

5630      \string\setglossarystyle\space or use the \MessageBreak
5631      style key=value option}%
5632  \fi
5633 }%
5634 \def\gls@dotocitle{\gls@settocitle{\@glo@type}}%

Store current value of \glossaryentrynumbers. (This may be changed via the optional ar-
gument)
5635 \let\org@glossaryentrynumbers\glossaryentrynumbers

Localise the effects of the optional argument
5636 \bgroup

Activate or deactivate sort key:
5637 \@printgloss@setsort

Determine settings specified in the optional argument.
5638 \setkeys{printgloss}{#1}%

Does the glossary exist?
5639 \ifglossaryexists{\@glo@type}%
5640 {%

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)
5641 \ifx\glossarytitle\org@glossarytitle
5642 \else
5643 \expandafter\let\csname @glo@type\@glo@type @title\endcsname
5644 \glossarytitle
5645 \fi

Allow a high-level user command to indicate the current glossary
5646 \let\currentglossary\@glo@type

Enable individual number lists to be suppressed.
5647 \let\org@glossaryentrynumbers\glossaryentrynumbers
5648 \let\glsnonextpages\@glsnonextpages

Enable individual number list to be activated:
5649 \let\glsnextpages\@glsnextpages

Enable suppression of description terminators.
5650 \let\nopostdesc\@nopostdesc

Set up the entry for the TOC
5651 \gls@dotocitle

Set the glossary style
5652 \@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):
5653 \let\gls@org@glossaryentryfield\glossentry
5654 \let\gls@org@glossarysubentryfield\subglossentry

```

```

5655 \renewcommand{\glossentry}[1]{%
5656 \xdef\glscurrententrylabel{\glstetoklabel{##1}}%
5657 \gls@org@glossaryentryfield{##1}%
5658 }%
5659 \renewcommand{\subglossentry}[2]{%
5660 \xdef\glscurrententrylabel{\glstetoklabel{##2}}%
5661 \gls@org@glossarysubentryfield{##1}{##2}%
5662 }%

5663 \@gls@preglossaryhook

```

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

```

5664 #2%
5665 }%
5666 {\GlossariesWarning{Glossary '@glo@type' doesn't exist}}%

End the current scope
5667 \egroup

Reset \glossaryentrynumbers
5668 \global\let\glossaryentrynumbers\@org@glossaryentrynumbers

Suppress warning about no \printglossary
5669 \global\let\warn@noprintglossary\relax
5670 }

```

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

```

5671 \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.)

5672 \makeatletter

Input the glossary file, if it exists.
5673 \@input@{\jobname.\csname @glo@type\@glo@type @in\endcsname}%

If the glossary file doesn't exist, do \null. (This ensures that the page is shipped out and all
write commands are done.) This might produce an empty page, but at this point the docu-
ment isn't complete, so it shouldn't matter.

5674 \IfFileExists{\jobname.\csname @glo@type\@glo@type @in\endcsname}%
5675 {}%
5676 {\null}%

If xindy is being used, need to write the language dependent information to the .aux file for
makeglossaries.
5677 \ifglxindy
5678 \ifcsundef{@xdy\@glo@type @language}%
5679 {%
5680 \edef\@do@auxoutstuff{%
5681 \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.

```

5682      \noexpand\immediate\noexpand\write\@auxout{%
5683      \string\providecommand\string\@xdylanguage[2]{}}%
5684      \noexpand\immediate\noexpand\write\@auxout{%
5685      \string\@xdylanguage{\@glo@type}{\@xdy@main@language}}%
5686      }%
5687    }%
5688  }%
5689  {%
5690    \edef\@do@auxoutstuff{%
5691      \noexpand\AtEndDocument{%
5692        \noexpand\immediate\noexpand\write\@auxout{%
5693        \string\providecommand\string\@xdylanguage[2]{}}%
5694        \noexpand\immediate\noexpand\write\@auxout{%
5695        \string\@xdylanguage{\@glo@type}{\csname @xdy@\@glo@type
5696        @language\endcsname}}%
5697      }%
5698    }%
5699  }%
5700  \@do@auxoutstuff
5701  \edef\@do@auxoutstuff{%
5702  \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.

```

5703      \noexpand\immediate\noexpand\write\@auxout{%
5704      \string\providecommand\string\@gls@codepage[2]{}}%
5705      \noexpand\immediate\noexpand\write\@auxout{%
5706      \string\@gls@codepage{\@glo@type}{\@gls@codepage}}%
5707      }%
5708    }%
5709  \@do@auxoutstuff
5710 \fi

```

Activate warning if \makeglossaries hasn't been used.

```

5711 \renewcommand*{\@warn@nomakeglossaries}{%
5712   \GlossariesWarningNoLine{\string\makeglossaries\space
5713   hasn't been used,^^Jthe glossaries will not be updated}%
5714 }%
5715 }

```

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

```
5716 \newcommand*{\@glo@sortentries}[2]{%
5717   \glosortentrieswarning
5718   \def\@glo@sortinglist{}%
5719   \def\@glo@sortinghandler{#1}%
5720   \edef\@glo@type{#2}%
5721   \forlistcsloop{\@glo@do@sortentries}{\@glsref@#2}%
5722   \csdef{\@glsref@#2}{}%
5723   \@for\@this@label:=\@glo@sortinglist\do{%
```

Has this entry already been added?

```
5724     \xifinlistcs{\@this@label}{\@glsref@#2}%
5725     {}%
5726     {%
5727       \listcsxadd{\@glsref@#2}{\@this@label}%
5728     }%
5729     \ifcsdef{\@glo@sortingchildren@\@this@label}%
5730     {%
5731       \@glo@addchildren{#2}{\@this@label}%
5732     }%
5733     {}%
5734   }%
5735 }
```

@glo@addchildren

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

```
5736 \newcommand*{\@glo@addchildren}[2]{%
```

Scope to allow nesting.

```
5737   \bgroup
5738   \letcs{\@glo@childlist}{\@glo@sortingchildren@#2}%
5739   \@for\@this@childlabel:=\@glo@childlist\do
5740   {%
```

Check this label hasn't already been added.

```
5741     \xifinlistcs{\@this@childlabel}{\@glsref@#1}%
5742     {}%
5743     {%
5744       \listcsxadd{\@glsref@#1}{\@this@childlabel}%
5745     }%
```

Does this child have children?

```
5746     \ifcsdef{\@glo@sortingchildren@\@this@childlabel}%
5747     {%
5748       \@glo@addchildren{#1}{\@this@childlabel}%
5749     }%
5750     {%
5751     }%
5752   }%
```

```

5753 \egroup
5754 }

```

@do@sortentries

```

5755 \newcommand*{\@glo@do@sortentries}[1]{%
5756   \ifglshasparent{#1}%
5757   {%
      This entry has a parent, so add it to the child list
5758     \edef\@glo@parent{\csuse{glo@glstdetoklabel{#1}@parent}}%
5759     \ifcsundef{glo@sortingchildren@\@glo@parent}%
5760     {%
5761       \csdef{glo@sortingchildren@\@glo@parent}{}%
5762     }%
5763   }%
5764   \expandafter\@glo@sortedinsert
5765   \csname @glo@sortingchildren@\@glo@parent\endcsname{#1}%

```

Has the parent been added?

```

5766   \xifinlistcs{\@glo@parent}{\@glstoklabel{\@glo@parent}}{%
5767   {%

```

Yes, it has so do nothing.

```

5768   }%
5769   {%

```

No, it hasn't so add it now.

```

5770     \expandafter\@glo@do@sortentries\expandafter{\@glo@parent}%
5771   }%
5772 }%
5773 {%
5774   \@glo@sortedinsert{\@glo@sortinglist}{#1}%
5775 }%
5776 }

```

glo@sortedinsert

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

Insert into list.

```

5777 \newcommand*{\@glo@sortedinsert}[2]{%
5778   \dtl@insertinto{#2}{#1}{\@glo@sortinghandler}%
5779 }%

```

The sort handlers need to be in the form required by datatool's \dtl@sortlist macro. These must set the count register \dtl@sortresult to either -1 (#1 less than #2), 0 (#1 = #2) or +1 (#1 greater than #2).

orthandler@word

```

5780 \newcommand*{\@glo@orthandler@word}[2]{%
5781   \letcs@gls@sort@A{glo@glstdetoklabel{#1}@sort}%

```

```

5782 \letcs\@gls@sort@B{glo\glsdetoklabel{#2}@sort}%
5783 \edef\glo@do@compare{%
5784   \noexpand\dtlwordindexcompare{\noexpand\dtl@sortresult}%
5785   {\expandonce\@gls@sort@B}%
5786   {\expandonce\@gls@sort@A}%
5787 }%
5788 \glo@do@compare
5789 }

```

thandler@letter

```

5790 \newcommand*{\@glo@sorthandler@letter}[2]{%
5791   \letcs\@gls@sort@A{glo\glsdetoklabel{#1}@sort}%
5792   \letcs\@gls@sort@B{glo\glsdetoklabel{#2}@sort}%
5793   \edef\glo@do@compare{%
5794     \noexpand\dtlletterindexcompare{\noexpand\dtl@sortresult}%
5795     {\expandonce\@gls@sort@B}%
5796     {\expandonce\@gls@sort@A}%
5797   }%
5798   \glo@do@compare
5799 }

```

orthandler@case Case-sensitive sort.

```

5800 \newcommand*{\@glo@sorthandler@case}[2]{%
5801   \letcs\@gls@sort@A{glo\glsdetoklabel{#1}@sort}%
5802   \letcs\@gls@sort@B{glo\glsdetoklabel{#2}@sort}%
5803   \edef\glo@do@compare{%
5804     \noexpand\dtlcompare{\noexpand\dtl@sortresult}%
5805     {\expandonce\@gls@sort@B}%
5806     {\expandonce\@gls@sort@A}%
5807   }%
5808   \glo@do@compare
5809 }

```

thandler@nocase Case-insensitive sort.

```

5810 \newcommand*{\@glo@sorthandler@nocase}[2]{%
5811   \letcs\@gls@sort@A{glo\glsdetoklabel{#1}@sort}%
5812   \letcs\@gls@sort@B{glo\glsdetoklabel{#2}@sort}%
5813   \edef\glo@do@compare{%
5814     \noexpand\dtlicompare{\noexpand\dtl@sortresult}%
5815     {\expandonce\@gls@sort@B}%
5816     {\expandonce\@gls@sort@A}%
5817   }%
5818   \glo@do@compare
5819 }

```

@sortmacro@word Sort macro for ‘word’

```

5820 \newcommand*{\@glo@sortmacro@word}[1]{%
5821   \ifdefstring{\@glo@default@sorttype}{standard}%
5822   {%

```

```

5823 \glo@sortentries{\glo@sorthandler@word}{#1}%
5824 }%
5825 {%
5826 \PackageError{glossaries}{Conflicting sort options:^^J
5827 \string\usepackage[sort=\glo@default@sorttype]{glossaries}^^J
5828 \string\printnoidxglossary[sort=word]}{}%
5829 }%
5830 }

```

ortmacro@letter Sort macro for ‘letter’

```

5831 \newcommand*{\glo@sortmacro@letter}[1]{%
5832 \ifdefstring{\glo@default@sorttype}{standard}%
5833 {%
5834 \glo@sortentries{\glo@sorthandler@letter}{#1}%
5835 }%
5836 {%
5837 \PackageError{glossaries}{Conflicting sort options:^^J
5838 \string\usepackage[sort=\glo@default@sorttype]{glossaries}^^J
5839 \string\printnoidxglossary[sort=letter]}{}%
5840 }%
5841 }

```

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

```

5842 \newcommand*{\glo@sortmacro@standard}[1]{%
5843 \ifdefstring{\glo@default@sorttype}{standard}%
5844 {%
5845 \ifcsdef{\glo@sorthandler@glorder}%
5846 {%
5847 \glo@sortentries{\csuse{\glo@sorthandler@glorder}}{#1}%
5848 }%
5849 {%
5850 \PackageError{glossaries}{Unknown sort handler ‘\glorder’}{}%
5851 }%
5852 }%
5853 {%
5854 \PackageError{glossaries}{Conflicting sort options:^^J
5855 \string\usepackage[sort=\glo@default@sorttype]{glossaries}^^J
5856 \string\printnoidxglossary[sort=standard]}{}%
5857 }%
5858 }

```

@sortmacro@case Sort macro for ‘case’

```

5859 \newcommand*{\glo@sortmacro@case}[1]{%
5860 \ifdefstring{\glo@default@sorttype}{standard}%
5861 {%
5862 \glo@sortentries{\glo@sorthandler@case}{#1}%
5863 }%
5864 {%
5865 \PackageError{glossaries}{Conflicting sort options:^^J

```

```

5866     \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5867     \string\printnoidxglossary[sort=case]}{}%
5868 }%
5869 }

```

ortmacro@nocase Sort macro for ‘nocase’

```

5870 \newcommand*{\@glo@sortmacro@nocase}[1]{%
5871   \ifdefstring{\@glo@default@sorttype}{standard}%
5872   {%
5873     \@glo@sortentries{\@glo@sorthandler@nocase}{#1}%
5874   }%
5875   {%
5876     \PackageError{glossaries}{Conflicting sort options:^^J
5877       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5878       \string\printnoidxglossary[sort=nocase]}{}%
5879   }%
5880 }

```

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

```

5881 \newcommand*{\@glo@sortmacro@def}[1]{%
5882   \def\@glo@sortinglist{%
5883     \for@gl@sentries[#1]{\@gl@s@thislabel}%
5884     {%
5885       \xifinlistcs{\@gl@s@thislabel}{\@gl@sref@#1}%
5886     }%
5887     \listcsadd{\@glo@sortinglist}{\@gl@s@thislabel}%
5888   }%
5889   {%

```

Hasn't been referenced.

```

5890   }%
5891 }%
5892 \cslet{\@gl@sref@#1}{\@glo@sortinglist}%
5893 }

```

ortmacro@def@do This won't include parent entries that haven't been referenced.

```

5894 \newcommand*{\@glo@sortmacro@def@do}[1]{%
5895   \ifinlistcs{#1}{\@gl@sref@\@glo@type}%
5896   {}%
5897   {%
5898     \listcsadd{\@gl@sref@\@glo@type}{#1}%
5899   }%
5900   \ifcsdef{\@glo@sortingchildren@#1}%
5901   {%
5902     \@glo@addchildren{\@glo@type}{#1}%
5903   }%
5904   {}%
5905 }

```

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

```
5906 \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.

```
5907 \newcommand*{\@print@noidx@glossary}{%

```

```
5908   \ifcsdef{@glsref@ \@glo@type}%

```

```
5909   {%

```

Sort the entries:

```
5910   \ifcsdef{@glo@sortmacro@ \@glo@sorttype}%

```

```
5911   {%

```

```
5912     \csuse{@glo@sortmacro@ \@glo@sorttype}{\@glo@type}%

```

```
5913   }%

```

```
5914   {%

```

```
5915     \PackageError{glossaries}{Unknown sort handler ‘\@glo@sorttype’}{}%

```

```
5916   }%

```

Do the glossary heading and preamble

```
5917   \glossarysection[\glossarytoctitle]{\glossarytitle}%

```

```
5918   \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.

```
5919   \def\@gls@currentlettergroup{%

```

```
5920     \begin{theglossary}%

```

```
5921     \glossaryheader

```

```
5922     \glsresetentrylist

```

Iterate through the entries.

```
5923   \forlistcsloop{\@gls@noidx@do}{@glsref@ \@glo@type}%

```

Finally end the glossary and do the postamble:

```
5924   \end{theglossary}%

```

```
5925   \glossarypostamble

```

```
5926   }%

```

```
5927   {%

```

```
5928     \@gls@noref@warn{\@glo@type}%

```

```
5929   }%

```

```
5930 }

```

\glo@grabfirst

```
5931 \def\glo@grabfirst#1#2\@nil{%

```

```
5932   \def\@gls@firsttok{#1}%

```

```

5933 \ifdefempty\@gls@firsttok
5934 {%
5935   \def\@glo@thislettergrp{0}%
5936 }%
5937 {%

  Sanitize it:
5938   \@onelevel@sanitize\@gls@firsttok

  Fetch the first letter:
5939   \expandafter\@glo@grabfirst\@gls@firsttok{}\{}\@nil
5940 }%
5941 }

```

\@glo@grabfirst

```

5942 \def\@glo@grabfirst#1#2\@nil{%
5943   \ifdefempty\@glo@thislettergrp
5944   {%
5945     \def\@glo@thislettergrp{glssymbols}%
5946   }%
5947   {%
5948     \count@=\uccode'#1\relax
5949     \ifnum\count@=0\relax
5950       \def\@glo@thislettergrp{glssymbols}%
5951     \else
5952       \ifdefstring\@glo@sorttype{case}%
5953       {%
5954         \count@=#1\relax
5955       }%
5956       {%
5957       }%
5958       \edef\@glo@thislettergrp{\the\count@}%
5959     \fi
5960   }%
5961 }

```

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

```

5962 \newcommand{\@gls@noidx@do}[1]{%

  Get this entry's location list
5963   \global\letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%

  Does this entry have a parent?
5964   \ifglshasparent{#1}%
5965   {%

    Has a parent.
5966     \gls@level=\csuse{glo@\glsdetoklabel{#1}@level}\relax
5967     \ifdefvoid{\@gls@loclist}
5968     {%

```

```

5969     \subglossentry{\gls@level}{#1}{}%
5970 }%
5971 {%
5972     \subglossentry{\gls@level}{#1}%
5973     {%
5974         \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
5975     }%
5976 }%
5977 }%
5978 {%

```

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

```

5979     \letcs{\@gls@sort}{\glo@\glsdetoklabel{#1}@sort}%

```

Fetch the first letter:

```

5980     \expandafter\glo@grabfirst\@gls@sort{}{}\@nil
5981     \ifdefequal{\@glo@thislettergrp}{\@gls@currentlettergroup}%
5982     {}%
5983     {%

```

Do the group header:

```

5984     \ifdefempty{\@gls@currentlettergroup}{}%
5985     {%

```

The group skip may start a new scope, so make a global assignment.

```

5986         \global\let\@glo@thislettergrp\@glo@thislettergrp
5987         \glsgroupskip
5988     }%
5989     \glsgroupheading{\@glo@thislettergrp}%
5990 }%

5991     \global\let\@gls@currentlettergroup\@glo@thislettergrp

```

Do this entry:

```

5992     \ifdefvoid{\@gls@loclist}
5993     {%
5994         \glossentry{#1}{}%
5995     }%
5996     {%
5997         \glossentry{#1}%
5998     }%
5999         \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
6000     }%
6001 }%
6002 }%
6003 }

```

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

Display location list.



```

6004 \newcommand*{\glsnoidxloclist}[1]{%
6005   \def\@gls@noidxloclist@sep{}%
6006   \def\@gls@noidxloclist@prev{}%
6007   \forlistloop{\glsnoidxloclisthandler}{#1}%
6008 }

```

`xloclisthandler` Handler for location list iterator.

```

6009 \newcommand*{\glsnoidxloclisthandler}[1]{%
6010   \ifdefstring{\@gls@noidxloclist@prev}{#1}%
6011   {%
        Same as previous location so skip.
6012   }%
6013   {%
6014     \@gls@noidxloclist@sep
6015     #1%
6016     \def\@gls@noidxloclist@sep{\delimN}%
6017     \def\@gls@noidxloclist@prev{#1}%
6018   }%
6019 }

```

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

```

6020 \newcommand*{\glsnoidxdisplayloclisthandler}[1]{%
6021   \ifdefstring{\@gls@noidxloclist@prev}{#1}%
6022   {%
        Same as previous location so skip.
6023   }%
6024   {%
6025     \@gls@noidxloclist@sep
6026     \@gls@noidxloclist@prev
6027     \def\@gls@noidxloclist@prev{#1}%
6028   }%
6029 }

```

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

Display a location in the location list.

```

6030 \newcommand*\glsnoidxdisplayloc[4]{%
6031   \setentrycounter[#1]{#2}%
6032   \csuse{#3}{#4}%
6033 }

```

`\@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.

```

6034 \newcommand*{\@gls@reference}[3]{%

```

Add to label list

```
6035 \glsdoifexistsorwarn{#2}%  
6036 {%  
6037   \ifcsundef{@glsref@#1}{\csgdef{@glsref@#1}{}}{}%  
6038   \ifinlistcs{#2}{@glsref@#1}%  
6039   {}%  
6040   {\listcsgadd{@glsref@#1}{#2}}%
```

Add to location list

```
6041   \ifcsundef{glo@glstdetoklabel{#2}@loclist}%  
6042   {\csgdef{glo@glstdetoklabel{#2}@loclist}{}}%  
6043   {}%  
6044   \listcsgadd{glo@glstdetoklabel{#2}@loclist}{#3}%  
6045 }%  
6046 }
```

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

```
6047 \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`.

```
6048 \define@key{printgloss}{title}{%  
6049   \def@glossarytitle{#1}%  
6050   \let@gls@dotoc@title\relax  
6051 }
```

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

```
6052 \define@key{printgloss}{toctitle}{%  
6053   \def@glossarytoctitle{#1}%  
6054   \let@gls@dotoc@title\relax  
6055 }
```

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

```
6056 \define@key{printgloss}{style}{%  
6057   \ifcsundef{@glsstyle@#1}%  
6058   {%  
6059     \PackageError{glossaries}%  
6060     {Glossary style ‘#1’ undefined}{}%  
6061   }%  
6062   {%  
6063     \def@glossarystyle{\setglossentrycompatibility  
6064       \csname @glsstyle@#1\endcsname}%  
6065   }%  
6066 }
```

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

```
6067 \define@choicekey{printgloss}{numberedsection}[\val\nr]{%  
6068   false,nolabel,autolabel,nameref}[nolabel]{%  
6069   \ifcase\nr\relax  
6070   \renewcommand*{\@@glossarysecstar}{*}%
```

```

6071 \renewcommand*{\@@glossaryseclabel}{}%
6072 \or
6073 \renewcommand*{\@@glossarysecstar}{}%
6074 \renewcommand*{\@@glossaryseclabel}{}%
6075 \or
6076 \renewcommand*{\@@glossarysecstar}{}%
6077 \renewcommand*{\@@glossaryseclabel}{\label{\glsautoprefix\@glo@type}}%
6078 \or
6079 \renewcommand*{\@@glossarysecstar}{*}%
6080 \renewcommand*{\@@glossaryseclabel}{%
6081 \protected@edef\@currentlabelname{\glossarytoctitle}%
6082 \label{\glsautoprefix\@glo@type}}%
6083 \fi
6084 }

```

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

```

6085 \define@choicekey{printgloss}{nogroupskip}{true,false}[true]{%
6086 \csuse{glsnogroupskip#1}%
6087 }

```

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

```

6088 \define@choicekey{printgloss}{nopostdot}{true,false}[true]{%
6089 \csuse{glsnopostdot#1}%
6090 }

```

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

```

6091 \define@choicekey{printgloss}{entrycounter}{true,false}[true]{%
6092 \csuse{glsentrycounter#1}%
6093 \ifglsentrycounter
6094 \ifx\@gls@counterwithin\@empty
6095 \newcounter{glossaryentry}%
6096 \else
6097 \newcounter{glossaryentry}[\@gls@counterwithin]%
6098 \fi
6099 \def\theHglossaryentry{\currentglossary.\theglossaryentry}%
6100 \renewcommand*{\glsresetentrycounter}{%
6101 \setcounter{glossaryentry}{0}%
6102 }%
6103 \renewcommand*{\glsstepentry}[1]{%
6104 \refstepcounter{glossaryentry}%
6105 \label{glsentry-\glsdetoklabel{##1}}%
6106 }%
6107 \renewcommand*{\glsentrycounterlabel}{\theglossaryentry.\space}%
6108 \renewcommand*{\glsentryitem}[1]{%
6109 \glsstepentry{##1}\glsentrycounterlabel
6110 }%
6111 \else
6112 \renewcommand*{\glsresetentrycounter}{}%
6113 \renewcommand*{\glsstepentry}[1]{}%
6114 \renewcommand*{\glsentrycounterlabel}{}%

```

```

6115 \renewcommand*{\glsubentryitem}[1]{\glresetsubentrycounter}
6116 \fi
6117 }

```

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.

```

6118 \define@choicekey{printgloss}{subentrycounter}{true,false}[true]{%
6119 \csuse{glssubentrycounter#1}%
6120 \ifglssubentrycounter
6121 \ifundef\c@glossarysubentry
6122 {%
6123 \ifglsubentrycounter
6124 \newcounter{glossarysubentry}[glossaryentry]%
6125 \else
6126 \newcounter{glossarysubentry}
6127 \fi
6128 }}%
6129 \renewcommand*{\glstepsentry}[1]{%
6130 \edef\currentglssubentry{\glstoklabel{##1}}%
6131 \refstepcounter{glossarysubentry}%
6132 \label{glsubentry-\currentglssubentry}%
6133 }%
6134 \renewcommand*{\glresetsubentrycounter}{%
6135 \setcounter{glossarysubentry}{0}%
6136 }%
6137 \renewcommand*{\glssubentryitem}[1]{%
6138 \glstepsentry{##1}\glssubentrycounterlabel
6139 }%
6140 \renewcommand*{\glssubentrycounterlabel}{\theglossarysubentry}\space}%
6141 \def\theHglossarysubentry{\currentglssubentry.\theglossarysubentry}
6142 \else
6143 \renewcommand*{\glssubentryitem}[1]{}%
6144 \renewcommand*{\glstepsentry}[1]{}%
6145 \renewcommand*{\glresetsubentrycounter}{}%
6146 \renewcommand*{\glssubentrycounterlabel}{}%
6147 \fi
6148 }

```

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

```

6149 \define@boolkey{printgloss}[gls]{nonnumberlist}[true]{%
6150 \ifglsnonnumberlist
6151 \def\glossaryentrynumbers##1{%
6152 \else
6153 \def\glossaryentrynumbers##1{##1}%
6154 \fi}

```

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

```

6155 \define@key{printgloss}{sort}{\@glo@assign@sortkey{#1}}

```

@assign@sortkey Issue error if used with \printglossary

```
6156 \newcommand*{\@glo@no@assign@sortkey}[1]{%
6157   \PackageError{glossaries}{'sort' key not permitted with
6158   \string\printglossary}%
6159   {The 'sort' key may only be used with \string\printnoidxglossary}%
6160 }
```

@assign@sortkey For use with \printnoidxglossary

```
6161 \newcommand*{\@glo@assign@sortkey}[1]{%
6162   \def\@glo@sorttype{#1}%
6163 }
```

\glsnnextpages 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 \glsnnextpages is place in the entry's description and 3 column tabular style glossary is used.) \org@glossaryentrynumbers needs to be set at the start of each glossary, in the event that \glossaryentrynumber is re-defined.

```
6164 \newcommand*{\@glsnnextpages}{%
6165   \gdef\glossaryentrynumbers##1{%
6166     \glsresetentrylist
6167   }%
6168 }
```

\@glsnnextpages 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 \@glsnnextpages is place in the entry's description and 3 column tabular style glossary is used.) \org@glossaryentrynumbers needs to be set at the start of each glossary, in the event that \glossaryentrynumber is re-defined.

```
6169 \newcommand*{\@glsnnextpages}{%
6170   \gdef\glossaryentrynumbers##1{%
6171     ##1\glsresetentrylist}}

```

glsresetentrylist Resets \glossaryentrynumbers

```
6172 \newcommand*{\glsresetentrylist}{%
6173   \global\let\glossaryentrynumbers\org@glossaryentrynumbers}

```

\glsnnextpages Outside of \printglossary this does nothing.

```
6174 \newcommand*{\glsnnextpages}{}

```

\glsnextpages Outside of \printglossary this does nothing.

```
6175 \newcommand*{\glsnextpages}{}

```

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

```
6176 \ifglsentrycounter
6177   \ifx\@gls@counterwithin\@empty
6178     \newcounter{glossaryentry}

```

```

6179 \else
6180   \newcounter{glossaryentry}[\@gls@counterwithin]
6181 \fi
6182 \def\theHglossaryentry{\currentglossary.\theglossaryentry}
6183 \fi

```

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

```

6184 \ifglssubentrycounter
6185   \ifglsentrycounter
6186     \newcounter{glossarysubentry}[glossaryentry]
6187   \else
6188     \newcounter{glossarysubentry}
6189   \fi
6190   \def\theHglossarysubentry{\currentglssubentry.\theglossarysubentry}
6191 \fi

```

**subentrycounter** Resets the glossarysubentry counter.

```

6192 \ifglssubentrycounter
6193   \newcommand*{\glsresetsubentrycounter}{%
6194     \setcounter{glossarysubentry}{0}%
6195   }
6196 \else
6197   \newcommand*{\glsresetsubentrycounter}{}
6198 \fi

```

**subentrycounter** Resets the glossaryentry counter.

```

6199 \ifglsentrycounter
6200   \newcommand*{\glsresetentrycounter}{%
6201     \setcounter{glossaryentry}{0}%
6202   }
6203 \else
6204   \newcommand*{\glsresetentrycounter}{}
6205 \fi

```

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

```

6206 \ifglsentrycounter
6207   \newcommand*{\glsstepentry}[1]{%
6208     \refstepcounter{glossaryentry}%
6209     \label{glsentry-\glsdetoklabel{#1}}%
6210   }
6211 \else
6212   \newcommand*{\glsstepentry}[1]{}
6213 \fi

```

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

```

6214 \ifglssubentrycounter
6215   \newcommand*{\glssubentry}[1]{%
6216     \edef\currentglssubentry{\glsetoklabel{#1}}%
6217     \refstepcounter{glossarysubentry}%
6218     \label{glentry-\currentglssubentry}%
6219   }
6220 \else
6221   \newcommand*{\glssubentry}[1]{}
6222 \fi

```

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

```

6223 \ifglentrycounter
6224   \newcommand*{\glrefentry}[1]{\ref{glentry-\glsetoklabel{#1}}}
6225 \else
6226   \ifglssubentrycounter
6227     \newcommand*{\glrefentry}[1]{\ref{glentry-\glsetoklabel{#1}}}
6228   \else
6229     \newcommand*{\glrefentry}[1]{\gls{#1}}
6230   \fi
6231 \fi

```

`entrycounterlabel` Defines how to display the glossaryentry counter.

```

6232 \ifglentrycounter
6233   \newcommand*{\glentrycounterlabel}{\theglossaryentry.\space}
6234 \else
6235   \newcommand*{\glentrycounterlabel}{}
6236 \fi

```

`entrycounterlabel` Defines how to display the glossarysubentry counter.

```

6237 \ifglssubentrycounter
6238   \newcommand*{\glssubentrycounterlabel}{\theglossarysubentry)\space}
6239 \else
6240   \newcommand*{\glssubentrycounterlabel}{}
6241 \fi

```

`\glentryitem` Step and display glossaryentry counter, if appropriate.

```

6242 \ifglentrycounter
6243   \newcommand*{\glentryitem}[1]{%
6244     \glstepentry{#1}\glentrycounterlabel
6245   }
6246 \else
6247   \newcommand*{\glentryitem}[1]{\glresetsubentrycounter}
6248 \fi

```

`glssubentryitem` Step and display glossarysubentry counter, if appropriate.

```

6249 \ifglssubentrycounter
6250   \newcommand*{\glssubentryitem}[1]{%
6251     \glssubentry{#1}\glssubentrycounterlabel
6252   }

```

```

6253 \else
6254   \newcommand*{\glssubentryitem}[1]{}
6255 \fi

```

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

```

6256 \ifcsundef{theglossary}%
6257 {%
6258   \newenvironment{theglossary}{}{}}%
6259}%
6260 {%
6261   \@gls@warnontheglossdefined
6262   \renewenvironment{theglossary}{}{}}%
6263 }

```

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**

```

6264 \newcommand*{\glossaryheader}{}

```

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

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

```

6265 \newcommand*{\glstarget}[2]{\@glstarget{\glolinkprefix#1}{#2}}

```

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

**compatibleglossentry**

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

```

6266 \providecommand*{\compatibleglossentry}[2]{%
6267   \toks@{#2}%
6268   \protected@edef\@do@glossentry{\noexpand\glossaryentryfield{#1}%
6269     {\noexpand\glsnamefont
6270       {\expandafter\expandonce\csname glo@#1@name\endcsname}}}%
6271     {\expandafter\expandonce\csname glo@#1@desc\endcsname}%
6272     {\expandafter\expandonce\csname glo@#1@symbol\endcsname}%
6273     {\the\toks@}}%
6274   }%
6275   \@do@glossentry
6276 }

```



\glossentryname

```
6277 \newcommand*{\glossentryname}[1]{%
6278   \glsdoifexistsorwarn{#1}%
6279   {%
6280     \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
6281     \expandafter\glsnamefont\expandafter{\glo@name}%
6282   }%
6283 }
```

\Glossentryname

```
6284 \newcommand*{\Glossentryname}[1]{%
6285   \glsdoifexistsorwarn{#1}%
6286   {%
6287     \glsnamefont{\Glsentryname{#1}}%
6288   }%
6289 }
```

\glossentrydesc

```
6290 \newcommand*{\glossentrydesc}[1]{%
6291   \glsdoifexistsorwarn{#1}%
6292   {%
6293     \glsentrydesc{#1}%
6294   }%
6295 }
```

\Glossentrydesc

```
6296 \newcommand*{\Glossentrydesc}[1]{%
6297   \glsdoifexistsorwarn{#1}%
6298   {%
6299     \Glsentrydesc{#1}%
6300   }%
6301 }
```

\glossentrysymbol

```
6302 \newcommand*{\glossentrysymbol}[1]{%
6303   \glsdoifexistsorwarn{#1}%
6304   {%
6305     \glsentrysymbol{#1}%
6306   }%
6307 }
```

\Glossentrysymbol

```
6308 \newcommand*{\Glossentrysymbol}[1]{%
6309   \glsdoifexistsorwarn{#1}%
6310   {%
6311     \Glsentrysymbol{#1}%
6312   }%
6313 }
```

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

```

6314 \providecommand*{\compatiblesubglossentry}[3]{%
6315   \toks@{#3}%
6316   \protected@edef\@do@subglossentry{\noexpand\glossarysubentryfield{\number#1}%
6317     {#2}%
6318     {\noexpand\glsnamefont
6319       {\expandafter\expandonce\csname glo@#2@name\endcsname}}}%
6320   {\expandafter\expandonce\csname glo@#2@desc\endcsname}%
6321   {\expandafter\expandonce\csname glo@#2@symbol\endcsname}%
6322   {\the\toks@}%
6323   }%
6324   \@do@subglossentry
6325 }

```

rycompatibility

```

6326 \newcommand*{\setglossentrycompatibility}{%
6327   \let\glossentry\compatibleglossentry
6328   \let\subglossentry\compatiblesubglossentry
6329 }
6330 \setglossentrycompatibility

```

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

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

```

6331 \newcommand{\glossaryentryfield}[5]{%
6332   \GlossariesWarning
6333   {Deprecated use of \string\glossaryentryfield.^^J
6334     I recommend you change to \string\glossentry.^^J
6335     If you've just upgraded, try removing your gls auxiliary
6336     files^^J and recompile}%
6337   \noindent\textbf{\glstarget{#1}{#2}} #4 #3. #5\par}

```

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

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

```

6338 \newcommand*{\glossarysubentryfield}[6]{%
6339   \GlossariesWarning
6340   {Deprecated use of \string\glossarysubentryfield.^^J

```

```

6341 I recommend you change to \string\subglossentry.^~J
6342 If you've just upgraded, try removing your gls auxiliary
6343 files^~J and recompile}%
6344 \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`

```
6345 \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`

```
6346 \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`

```

6347 \newcommand*{\glsgetgrouptitle}[1]{%
6348   \@gls@getgrouptitle{#1}{\@gls@grptitle}%
6349   \@gls@grptitle
6350 }

```

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

```
6351 \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.

6352 \dtl@ifsingle{#1}%
6353 {%
6354   \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
6355 }%
6356 {%
6357   \ifboolexpr{test{\ifstrequal{#1}{glssymbols}}
6358               or test{\ifstrequal{#1}{glsnumbers}}}%
6359   {%
6360     \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
6361   }%
6362   {%
6363     \def#2{#1}%
6364   }%
6365 }%
6366 }
```

`x@getgrouptitle` Version for the no-indexing app option:

```
6367 \newcommand*{\@gls@noidx@getgrouptitle}[2]{%
6368   \DTLifint{#1}%
6369   {\edef#2{\char#1\relax}}%
6370   {%
6371     \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
6372   }%
6373 }
```

`\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`

```
6374 \newcommand*{\glsgetgrouplabel}[1]{%
6375 \ifthenelse{\equal{#1}{\glssymbolsgroupname}}{\glssymbols}{%
6376 \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`

```
6377 \newcommand*{\setentrycounter}[2] [] {%
```

```

6378 \def\@glo@counterprefix{#1}%
6379 \ifx\@glo@counterprefix\empty
6380   \def\@glo@counterprefix{.}%
6381 \else
6382   \def\@glo@counterprefix{.#1.}%
6383 \fi
6384 \def\glsentrycounter{#2}%
6385 }

```

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

`\setglossarystyle`

```

6386 \newcommand*{\setglossarystyle}[1]{%
6387   \ifcsundef{@glsstyle@#1}%
6388   {%
6389     \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
6390   }%
6391   {%
6392     \csname @glsstyle@#1\endcsname
6393   }%

```

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

```

6394 \ifx\@glossary@default@style\relax
6395   \protected@edef\@glossary@default@style{#1}%
6396 \fi
6397 }

```

`\glossarystyle`

```

6398 \newcommand*{\glossarystyle}[1]{%
6399   \ifcsundef{@glsstyle@#1}%
6400   {%
6401     \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
6402   }%
6403   {%
6404     \GlossariesWarning
6405     {Deprecated command \string\glossarystyle.^~J
6406      I recommend you switch to \string\setglossarystyle\space unless
6407      you want to maintain backward compatibility}%
6408     \setglossentrycompatibility
6409     \csname @glsstyle@#1\endcsname

6410     \ifcsdef{@glscompstyle@#1}%
6411     {\setglossentrycompatibility\csuse{@glscompstyle@#1}}%
6412     {}%
6413   }%

```

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

```

6414 \ifx\@glossary@default@style\relax
6415   \protected@edef\@glossary@default@style{#1}%

```

```

6416 \fi
6417 }

```

`\newglossarystyle` New glossary styles can be defined using:

```
\newglossarystyle{<name>}{<definition>}
```

The *<definition>* argument should redefine `\theglossary`, `\glossaryheader`, `\glsgroupheading`, `\glossaryentryfield` and `\glsgroupskip` (see [section 1.19](#) for the definitions of predefined styles). Glossary styles should not redefine `\glossarypreamble` and `\glossarypostamble`, as the user should be able to switch between styles without affecting the pre- and postambles.

```

6418 \newcommand{\newglossarystyle}[2]{%
6419   \ifcsundef{@glsstyle@#1}%
6420   {%
6421     \expandafter\def\csname @glsstyle@#1\endcsname{#2}%
6422   }%
6423   {%
6424     \PackageError{glossaries}{Glossary style ‘#1’ is already defined}{}%
6425   }%
6426 }

```

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

```

6427 \newcommand{\renewglossarystyle}[2]{%
6428   \ifcsundef{@glsstyle@#1}%
6429   {%
6430     \PackageError{glossaries}{Glossary style ‘#1’ isn’t already defined}{}%
6431   }%
6432   {%
6433     \csdef{@glsstyle@#1}{#2}%
6434   }%
6435 }

```

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`

```
6436 \newcommand*{\glsnamefont}[1]{#1}
```

Each glossary entry has an associated number list (usually page numbers) that indicate where in the document the entry has been used. The format for these number lists can be changed using the format key in commands like `\glslink`. The default format is given by `\glsnumber`. This takes a single argument which may be a single number, a number range or a number list. The number ranges are delimited with `\delimR`, the number lists are delimited with `\delimN`.

If the document doesn't have hyperlinks, the numbers can be displayed just as they are, but if the document supports hyperlinks, the numbers should link to the relevant location. This means extracting the individual numbers from the list or ranges. The package does this with the `\hyperpage` command, but this is encoded for comma and dash delimiters and only for the page counter, but this code needs to be more general. So I have adapted the code used in the package.

`\glshypernumber`

```
6437 \ifcsundef{hyperlink}%
6438 {%
6439   \def\glshypernumber#1{#1}%
6440 }%
6441 {%
6442   \def\glshypernumber#1{\@glshypernumber#1\nohyperpage{}}\@nil}
6443 }
```

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

```
6444 \def\@glshypernumber#1\@nohyperpage#2#3\@nil{%
6445   \ifx\#1\%
6446   \else
6447     \@delimR#1\delimR\delimR\%
6448   \fi
6449   \ifx\#2\%
6450   \else
6451     #2%
6452   \fi
6453   \ifx\#3\%
6454   \else
6455     \@glshypernumber#3\@nil
6456   \fi
6457 }
```

`\@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`

```
6458 \def\@delimR#1\delimR #2\delimR #3\{%
6459 \ifx\#2\%
6460   \@delimN{#1}%
6461 \else
6462   \@gls@numberlink{#1}\delimR\@gls@numberlink{#2}%
6463 \fi}
```

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

`\@delimN`

```
6464 \def\@delimN#1{\@delimN#1\delimN \delimN\%
6465 \def\@delimN#1\delimN #2\delimN#3\{%
6466 \ifx\#3\%
```

```

6467 \@gls@numberlink{#1}%
6468 \else
6469 \@gls@numberlink{#1}\delimN\@gls@numberlink{#2}%
6470 \fi
6471 }

```

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

```

6472 \def\@gls@numberlink#1{%
6473 \begingroup
6474 \toks@={}%
6475 \@gls@removespaces#1 \@nil
6476 \endgroup}

6477 \def\@gls@removespaces#1 #2\@nil{%
6478 \toks@=\expandafter{\the\toks@#1}%
6479 \ifx\#2\%
6480 \edef\x{\the\toks@}%
6481 \ifx\x\empty
6482 \else

6483 \hyperlink{\glstrycounter\@glo@counterprefix\the\toks@}%
6484 {\the\toks@}%
6485 \fi
6486 \else
6487 \@gls@ReturnAfterFi{%
6488 \@gls@removespaces#2\@nil
6489 }%
6490 \fi
6491 }
6492 \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
6493 \newcommand*\hyperrm[1]{\textrm{\glshypernumber{#1}}}

\hypersf
6494 \newcommand*\hypersf[1]{\textsf{\glshypernumber{#1}}}

\hypertt
6495 \newcommand*\hypertt[1]{\texttt{\glshypernumber{#1}}}

\hyperbf
6496 \newcommand*\hyperbf[1]{\textbf{\glshypernumber{#1}}}

\hypermd
6497 \newcommand*\hypermd[1]{\textmd{\glshypernumber{#1}}}

```



`\hyperit`

```
6498 \newcommand*{\hyperit}[1]{\textit{\glshypernumber{#1}}}
```

`\hypersl`

```
6499 \newcommand*{\hypersl}[1]{\textsl{\glshypernumber{#1}}}
```

`\hyperup`

```
6500 \newcommand*{\hyperup}[1]{\textup{\glshypernumber{#1}}}
```

`\hypersc`

```
6501 \newcommand*{\hypersc}[1]{\textsc{\glshypernumber{#1}}}
```

`\hyperemph`

```
6502 \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.

```
6503 \newcommand{\oldacronym}[4][\gls@label]{%
6504   \def\gls@label{#2}%
6505   \newacronym[#4]{#1}{#2}{#3}%
6506   \ifcsundef{xspace}%
6507   {%
6508     \expandafter\edef\csname#1\endcsname{%
6509       \noexpand\@ifstar{\noexpand\Gls{#1}}{\noexpand\gls{#1}}}%
6510   }%
6511   }%
6512   {%
6513     \expandafter\edef\csname#1\endcsname{%
6514       \noexpand\@ifstar{\noexpand\Gls{#1}\noexpand\xspace}{%
6515         \noexpand\gls{#1}\noexpand\xspace}%

```

```

6516     }%
6517   }%
6518 }

```

```
\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`

```
6519 \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.

```
6520 \newcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}
```

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

`\glstextup`

```
6521 \newrobustcmd*{\glstextup}[1]{\ifdef\textulc{\textulc{#1}}{\textup{#1}}}
```

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

`\glsshortkey`

```
6522 \newcommand*{\glsshortkey}{short}
```

`\glsshortpluralkey`

```
6523 \newcommand*{\glsshortpluralkey}{shortplural}
```

`\glslongkey`

```
6524 \newcommand*{\glslongkey}{long}
```

`\glslongpluralkey`

```
6525 \newcommand*{\glslongpluralkey}{longplural}
```

`\acrfull` Full form of the acronym.

```
6526 \newrobustcmd*{\acrfull}{\@gls@hyp@opt\ns@acrfull}

6527 \newcommand*\ns@acrfull[2] [] {%
6528   \new@ifnextchar[{\@acrfull{#1}{#2}}%
6529     {\@acrfull{#1}{#2} []}%
6530 }
```

`\@acrfull` Low-level macro:

```
6531 \def\@acrfull#1#2[#3] {%
    Make it easier for acronym styles to change this:
6532   \acrfullfmt{#1}{#2}{#3}%
6533 }
```

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.

```
6534 \newcommand*\acrfullfmt[3] {%
6535   \acrlinkfullformat{\@acrlong}{\@acrshort}{#1}{#2}{#3}%
6536 }
```

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

```
6537 \newcommand{\acrlinkfullformat}[5] {%
6538   \acrfullformat{#1{#3}{#4}[#5]}{#2{#3}{#4} []}%
6539 }
```

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

```
6540 \newcommand{\acrfullformat}[2]{#1\glsspace(#2)}
```

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

```
6541 \newrobustcmd{\glsspace}{\space}
```

Default format for full acronym

`\Acrfull`

```
6542 \newrobustcmd*{\Acrfull}{\@gls@hyp@opt\ns@Acrfull}

6543 \newcommand*\ns@Acrfull[2] [] {%
6544   \new@ifnextchar[{\@Acrfull{#1}{#2}}%
6545     {\@Acrfull{#1}{#2} []}%
6546 }
```

Low-level macro:

```
6547 \def\@Acrfull#1#2[#3] {%
```

Make it easier for acronym styles to change this:

```
6548 \Acrfullfmt{#1}{#2}{#3}%  
6549 }
```

\Acrfullfmt First letter upper case full format.

```
6550 \newcommand*{\Acrfullfmt}[3]{%  
6551 \acrlinkfullformat{\@Acrlong}{\@acrshort}{#1}{#2}{#3}%  
6552 }
```

\ACRfull

```
6553 \newrobustcmd*{\ACRfull}{\@gls@hyp@opt\ns@ACRfull}  
  
6554 \newcommand*{\ns@ACRfull}[2][{}]{%  
6555 \new@ifnextchar[{\@ACRfull{#1}{#2}}%  
6556 {\@ACRfull{#1}{#2}[]}%  
6557 }
```

Low-level macro:

```
6558 \def\@ACRfull#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6559 \ACRfullfmt{#1}{#2}{#3}%  
6560 }
```

\ACRfullfmt All upper case full format.

```
6561 \newcommand*{\ACRfullfmt}[3]{%  
6562 \acrlinkfullformat{\@ACRlong}{\@ACRshort}{#1}{#2}{#3}%  
6563 }
```

Plural:

\acrfullpl

```
6564 \newrobustcmd*{\acrfullpl}{\@gls@hyp@opt\ns@acrfullpl}  
  
6565 \newcommand*{\ns@acrfullpl}[2][{}]{%  
6566 \new@ifnextchar[{\@acrfullpl{#1}{#2}}%  
6567 {\@acrfullpl{#1}{#2}[]}%  
6568 }
```

Low-level macro:

```
6569 \def\@acrfullpl#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6570 \acrfullplfmt{#1}{#2}{#3}%  
6571 }
```

\acrfullplfmt No case change plural full format.

```
6572 \newcommand*{\acrfullplfmt}[3]{%  
6573 \acrlinkfullformat{\@acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%  
6574 }
```

`\Acrfullpl`

```
6575 \newrobustcmd*{\Acrfullpl}{\@gls@hyp@opt\@ns@Acrfullpl}
```

```
6576 \newcommand*{\ns@Acrfullpl}[2][\%  
6577   \new@ifnextchar[{\@Acrfullpl{#1}{#2}}%  
6578   {\@Acrfullpl{#1}{#2}[]}]%  
6579 }
```

Low-level macro:

```
6580 \def\@Acrfullpl#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6581   \Acrfullplfmt{#1}{#2}{#3}%  
6582 }
```

`\Acrfullplfmt` First letter upper case plural full format.

```
6583 \newcommand*{\Acrfullplfmt}[3]{%  
6584   \acrlinkfullformat{\@Acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%  
6585 }
```

`\ACRfullpl`

```
6586 \newrobustcmd*{\ACRfullpl}{\@gls@hyp@opt\@ns@ACRfullpl}
```

```
6587 \newcommand*{\ns@ACRfullpl}[2][\%  
6588   \new@ifnextchar[{\@ACRfullpl{#1}{#2}}%  
6589   {\@ACRfullpl{#1}{#2}[]}]%  
6590 }
```

Low-level macro:

```
6591 \def\@ACRfullpl#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6592   \ACRfullplfmt{#1}{#2}{#3}%  
6593 }
```

`\ACRfullplfmt` All upper case plural full format.

```
6594 \newcommand*{\ACRfullplfmt}[3]{%  
6595   \acrlinkfullformat{\@ACRlongpl}{\@ACRshortpl}{#1}{#2}{#3}%  
6596 }
```

## 1.18 Predefined acronym styles

`\acronymfont` This is only used with the additional acronym styles:

```
6597 \newcommand{\acronymfont}[1]{#1}
```

`\firstacronymfont` This is only used with the additional acronym styles:

```
6598 \newcommand{\firstacronymfont}[1]{\acronymfont{#1}}
```

`\acronymformat` The styles that allow an additional description use `\acronymformat{<short>}{<long>}` to determine what information is displayed in the name.

```
6599 \newcommand*{\acronymformat}[2]{\acronymfont{#1}}
```

Define some tokens used by `\newacronym`:

`\glskeylisttok`

```
6600 \newtoks\glskeylisttok
```

`\glslabeltok`

```
6601 \newtoks\glslabeltok
```

`\glsshorttok`

```
6602 \newtoks\glsshorttok
```

`\gslongtok`

```
6603 \newtoks\gslongtok
```

`\newacronymhook` Provide a hook for `\newacronym`:

```
6604 \newcommand*{\newacronymhook}{}
```

`\genericNewAcronym` New improved version of setting the acronym style.

```
6605 \newcommand*{\SetGenericNewAcronym}{%
```

Change the behaviour of `\Glsentryname` to workaround expansion issues that cause a problem for `\makefirstuc`

```
6606 \let\@Gls@entryname\@Gls@acrentryname
```

Change the way acronyms are defined:

```
6607 \renewcommand{\newacronym}[4][\{%
6608   \ifdefempty{\@glsacronymlists}%
6609   {%
6610     \def\@glo@type{\acronymtype}%
6611     \setkeys{glossentry}{##1}%
6612     \DeclareAcronymList{\@glo@type}%
6613   }%
6614 }%
6615 \glskeylisttok{##1}%
6616 \glslabeltok{##2}%
6617 \glsshorttok{##3}%
6618 \gslongtok{##4}%
6619 \newacronymhook
6620 \protected@edef\@do@newglossaryentry{%
6621   \noexpand\newglossaryentry{\the\glslabeltok}%
6622   {%
6623     type=\acronymtype,%
6624     name={\expandonce{\acronymentry{##2}}},%
6625     sort={\acronymsort{\the\glsshorttok}{\the\gslongtok}},%
6626     text={\the\glsshorttok},%
```

```

6627     short={\the\glsshorttok},%
6628     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6629     long={\the\glslongtok},%
6630     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6631     \GenericAcronymFields,%
6632     \the\glskeylisttok
6633 }%
6634 }%
6635 \do@newglossaryentry
6636 }%

```

Make sure that `\acrfull` etc reflects the new style:

```

6637 \renewcommand*{\acrfullfmt}[3]{%
6638   \glslink[##1]{##2}{\genacrfullformat{##2}{##3}}}%
6639 \renewcommand*{\Acrfullfmt}[3]{%
6640   \glslink[##1]{##2}{\Genacrfullformat{##2}{##3}}}%
6641 \renewcommand*{\ACRfullfmt}[3]{%
6642   \glslink[##1]{##2}{%
6643     \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}%
6644 \renewcommand*{\acrfullplfmt}[3]{%
6645   \glslink[##1]{##2}{\genplacrfullformat{##2}{##3}}}%
6646 \renewcommand*{\Acrfullplfmt}[3]{%
6647   \glslink[##1]{##2}{\Genplacrfullformat{##2}{##3}}}%
6648 \renewcommand*{\ACRfullplfmt}[3]{%
6649   \glslink[##1]{##2}{%
6650     \mfirstucMakeUppercase{\genplacrfullformat{##2}{##3}}}%

```

Make sure that `\glsentryfull` etc reflects the new style:

```

6651 \renewcommand*{\glsentryfull}[1]{\genacrfullformat{##1}{}}%
6652 \renewcommand*{\Glsentryfull}[1]{\Genacrfullformat{##1}{}}%
6653 \renewcommand*{\glsentryfullpl}[1]{\genplacrfullformat{##1}{}}%
6654 \renewcommand*{\Glsentryfullpl}[1]{\Genplacrfullformat{##1}{}}%
6655 }

```

`\icAcronymFields` Fields used by `\SetGenericNewAcronym` that can be changed by the acronym style.

```

6656 \newcommand*{\GenericAcronymFields}{description={\the\glslongtok}}

```

`\acronymentry` `\acronymentry{<label>}`

Display style for the name field in the list of acronyms.

```

6657 \newcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}

```

`\acronymsort` `\acronymsort{<short>}{<long>}`

Default sort format for acronyms.

```

6658 \newcommand*{\acronymsort}[2]{#1}

```

`\setacronymstyle`     `\setacronymstyle{<style name>}`

```

6659 \newcommand*{\setacronymstyle}[1]{%
6660   \ifcsundef{@glsacr@dispstyle@#1}%
6661   {%
6662     \PackageError{glossaries}{Undefined acronym style ‘#1’}{}%
6663   }%
6664   {%
6665     \ifdefempty{@glsacronymlists}%
6666     {%
6667       \DeclareAcronymList{\acronymtype}%
6668     }%
6669   }%
6670   \SetGenericNewAcronym
6671   \GlsUseAcrStyleDefs{#1}%
6672   \@for\@gls@type:=\@glsacronymlists\do{%
6673     \defglsentryfmt[\@gls@type]{\GlsUseAcrEntryDispStyle{#1}}%
6674   }%
6675 }%
6676 }

```

`\newacronymstyle`     `\newacronymstyle{<style name>}{<entry format definition>}{<display definitions>}`

Defines a new acronym style called *<style name>*.

```

6677 \newcommand*{\newacronymstyle}[3]{%
6678   \ifcsdef{@glsacr@dispstyle@#1}%
6679   {%
6680     \PackageError{glossaries}{Acronym style ‘#1’ already exists}{}%
6681   }%
6682   {%
6683     \csdef{@glsacr@dispstyle@#1}{#2}%
6684     \csdef{@glsacr@styledefs@#1}{#3}%
6685   }%
6686 }

```

`\renewacronymstyle`     Redefines the given acronym style.

```

6687 \newcommand*{\renewacronymstyle}[3]{%
6688   \ifcsdef{@glsacr@dispstyle@#1}%
6689   {%
6690     \csdef{@glsacr@dispstyle@#1}{#2}%
6691     \csdef{@glsacr@styledefs@#1}{#3}%
6692   }%
6693   {%
6694     \PackageError{glossaries}{Acronym style ‘#1’ doesn’t exist}{}%
6695   }%
6696 }

```



rEntryDispStyle

```
6697 \newcommand*{\GlsUseAcrEntryDispStyle}[1]{\csuse{@glsacr@dispstyle@#1}}
```

UseAcrStyleDefs

```
6698 \newcommand*{\GlsUseAcrStyleDefs}[1]{\csuse{@glsacr@styledefs@#1}}
```

Predefined acronym styles:

long-short    *<long>* (*<short>*) acronym style.

```
6699 \newacronymstyle{long-short}%
```

```
6700 {%
```

Check for long form in case this is a mixed glossary.

```
6701 \ifglshaslong{\glslabel}{\glsngenacfmt}{\glsngenentryfmt}%
```

```
6702 }%
```

```
6703 {%
```

```
6704 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
```

```
6705 \renewcommand*{\genacrfullformat}[2]{%
```

```
6706 \glsentrylong{##1}##2\space
```

```
6707 (\protect\firstacronymfont{\glsentryshort{##1}})%
```

```
6708 }%
```

```
6709 \renewcommand*{\Genacrfullformat}[2]{%
```

```
6710 \Glsentrylong{##1}##2\space
```

```
6711 (\protect\firstacronymfont{\glsentryshort{##1}})%
```

```
6712 }%
```

```
6713 \renewcommand*{\genplacrfullformat}[2]{%
```

```
6714 \glsentrylongpl{##1}##2\space
```

```
6715 (\protect\firstacronymfont{\glsentryshortpl{##1}})%
```

```
6716 }%
```

```
6717 \renewcommand*{\Genplacrfullformat}[2]{%
```

```
6718 \Glsentrylongpl{##1}##2\space
```

```
6719 (\protect\firstacronymfont{\glsentryshortpl{##1}})%
```

```
6720 }%
```

```
6721 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
```

```
6722 \renewcommand*{\acronymsort}[2]{##1}%
```

```
6723 \renewcommand*{\acronymfont}[1]{##1}%
```

```
6724 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
```

```
6725 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
```

```
6726 }
```

long-sp-short    Similar to the previous style but allows the space between the long and short form to be customized.

```
6727 \newacronymstyle{long-sp-short}%
```

```
6728 {%
```

Check for long form in case this is a mixed glossary.

```
6729 \ifglshaslong{\glslabel}{\glsngenacfmt}{\glsngenentryfmt}%
```

```
6730 }%
```

```
6731 {%
```

```
6732 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
```

```

6733 \renewcommand*{\genacrfullformat}[2]{%
6734 \glentrylong{##1}##2\glsacspace{##1}%
6735 (\protect\firstacronymfont{\glentryshort{##1}})%
6736 }%
6737 \renewcommand*{\Genacrfullformat}[2]{%
6738 \Glsentrylong{##1}##2\glsacspace{##1}%
6739 (\protect\firstacronymfont{\glentryshort{##1}})%
6740 }%
6741 \renewcommand*{\genplacrfullformat}[2]{%
6742 \glentrylongpl{##1}##2\glsacspace{##1}%
6743 (\protect\firstacronymfont{\glentryshortpl{##1}})%
6744 }%
6745 \renewcommand*{\Genplacrfullformat}[2]{%
6746 \Glsentrylongpl{##1}##2\glsacspace{##1}%
6747 (\protect\firstacronymfont{\glentryshortpl{##1}})%
6748 }%
6749 \renewcommand*{\acronymentry}[1]{\acronymfont{\glentryshort{##1}}}%
6750 \renewcommand*{\acronymsort}[2]{##1}%
6751 \renewcommand*{\acronymfont}[1]{##1}%
6752 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
6753 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6754 }

```

`\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.

```

6755 \newcommand*{\glsacspace}[1]{%
6756 \settowidth{\dimen@}{(\firstacronymfont{\glentryshort{##1}})}%
6757 \ifdim\dimen@<3em~\else\space\fi
6758 }

```

`short-long` (*short*) (*long*) acronym style.

```

6759 \newacronymstyle{short-long}%
6760 {%
    Check for long form in case this is a mixed glossary.
6761 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6762 }%
6763 {%
6764 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6765 \renewcommand*{\genacrfullformat}[2]{%
6766 \protect\firstacronymfont{\glentryshort{##1}}##2\space
6767 (\glentrylong{##1})%
6768 }%
6769 \renewcommand*{\Genacrfullformat}[2]{%
6770 \protect\firstacronymfont{\Glsentryshort{##1}}##2\space
6771 (\glentrylong{##1})%
6772 }%
6773 \renewcommand*{\genplacrfullformat}[2]{%
6774 \protect\firstacronymfont{\glentryshortpl{##1}}##2\space

```

```

6775 (\glsentrylongpl{##1})%
6776 }%
6777 \renewcommand*{\Genplacrfullformat}[2]{%
6778 \protect\firstacronymfont{\Glsentryshortpl{##1}}##2\space
6779 (\glsentrylongpl{##1})%
6780 }%

6781 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6782 \renewcommand*{\acronymsort}[2]{##1}%
6783 \renewcommand*{\acronymfont}[1]{##1}%
6784 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
6785 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6786 }

```

long-sc-short    *<long>* (\textsc{<short>}) acronym style.

```

6787 \newacronymstyle{long-sc-short}%
6788 {%
6789 \GlsUseAcrEntryDisplayStyle{long-short}%
6790 }%
6791 {%
6792 \GlsUseAcrStyleDefs{long-short}%
6793 \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6794 \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6795 }

```

long-sm-short    *<long>* (\textsmaller{<short>}) acronym style.

```

6796 \newacronymstyle{long-sm-short}%
6797 {%
6798 \GlsUseAcrEntryDisplayStyle{long-short}%
6799 }%
6800 {%
6801 \GlsUseAcrStyleDefs{long-short}%
6802 \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6803 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6804 }

```

sc-short-long    *<short>* (\textsc{<long>}) acronym style.

```

6805 \newacronymstyle{sc-short-long}%
6806 {%
6807 \GlsUseAcrEntryDisplayStyle{short-long}%
6808 }%
6809 {%
6810 \GlsUseAcrStyleDefs{short-long}%
6811 \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6812 \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6813 }

```

sm-short-long    *<short>* (\textsmaller{<long>}) acronym style.

```

6814 \newacronymstyle{sm-short-long}%

```

```

6815 {%
6816   \GlsUseAcrEntryDispStyle{short-long}%
6817 }%
6818 {%
6819   \GlsUseAcrStyleDefs{short-long}%
6820   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6821   \renewcommand*{\acrpluralsuffix}{\glacrpluralsuffix}%
6822 }

```

long-short-desc    *<long>* (*{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

6823 \newacronymstyle{long-short-desc}%
6824 {%
6825   \GlsUseAcrEntryDispStyle{long-short}%
6826 }%
6827 {%
6828   \GlsUseAcrStyleDefs{long-short}%
6829   \renewcommand*{\GenericAcronymFields}{}%
6830   \renewcommand*{\acronymsort}[2]{##2}%
6831   \renewcommand*{\acronymentry}[1]{%
6832     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6833 }

```

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.

```

6834 \newacronymstyle{long-sp-short-desc}%
6835 {%
6836   \GlsUseAcrEntryDispStyle{long-sp-short}%
6837 }%
6838 {%
6839   \GlsUseAcrStyleDefs{long-sp-short}%
6840   \renewcommand*{\GenericAcronymFields}{}%
6841   \renewcommand*{\acronymsort}[2]{##2}%
6842   \renewcommand*{\acronymentry}[1]{%
6843     \glentrylong{##1}\glsacspace{##1}(\acronymfont{\glentryshort{##1}})}%
6844 }

```

g-sc-short-desc    *<long>* (\textsc{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

6845 \newacronymstyle{long-sc-short-desc}%
6846 {%
6847   \GlsUseAcrEntryDispStyle{long-sc-short}%
6848 }%
6849 {%
6850   \GlsUseAcrStyleDefs{long-sc-short}%
6851   \renewcommand*{\GenericAcronymFields}{}%
6852   \renewcommand*{\acronymsort}[2]{##2}%
6853   \renewcommand*{\acronymentry}[1]{%
6854     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%

```

```

6855 }

g-sm-short-desc  <long> (\textsmaller{\<short>}) acronym style that has an accompanying description (which
                  the user needs to supply).
6856 \newacronymstyle{long-sm-short-desc}%
6857 {%
6858   \GlsUseAcrEntryDispStyle{long-sm-short}%
6859 }%
6860 {%
6861   \GlsUseAcrStyleDefs{long-sm-short}%
6862   \renewcommand*{\GenericAcronymFields}{}%
6863   \renewcommand*{\acronymsort}[2]{##2}%
6864   \renewcommand*{\acronymentry}[1]{%
6865     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6866 }

short-long-desc  <short> ({<long>}) acronym style that has an accompanying description (which the user needs
                  to supply).
6867 \newacronymstyle{short-long-desc}%
6868 {%
6869   \GlsUseAcrEntryDispStyle{short-long}%
6870 }%
6871 {%
6872   \GlsUseAcrStyleDefs{short-long}%
6873   \renewcommand*{\GenericAcronymFields}{}%
6874   \renewcommand*{\acronymsort}[2]{##2}%
6875   \renewcommand*{\acronymentry}[1]{%
6876     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6877 }

short-long-desc  <long> (\textsc{\<short>}) acronym style that has an accompanying description (which the
                  user needs to supply).
6878 \newacronymstyle{sc-short-long-desc}%
6879 {%
6880   \GlsUseAcrEntryDispStyle{sc-short-long}%
6881 }%
6882 {%
6883   \GlsUseAcrStyleDefs{sc-short-long}%
6884   \renewcommand*{\GenericAcronymFields}{}%
6885   \renewcommand*{\acronymsort}[2]{##2}%
6886   \renewcommand*{\acronymentry}[1]{%
6887     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6888 }

short-long-desc  <long> (\textsmaller{\<short>}) acronym style that has an accompanying description (which
                  the user needs to supply).
6889 \newacronymstyle{sm-short-long-desc}%
6890 {%

```

```

6891 \GlsUseAcrEntryDispStyle{sm-short-long}%
6892 }%
6893 {%
6894 \GlsUseAcrStyleDefs{sm-short-long}%
6895 \renewcommand*{\GenericAcronymFields}{}%
6896 \renewcommand*{\acronymsort}[2]{##2}%
6897 \renewcommand*{\acronymentry}[1]{%
6898     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6899 }

```

dua *<long>* only acronym style.

```

6900 \newacronymstyle{dua}%
6901 {%

```

Check for long form in case this is a mixed glossary.

```

6902 \ifdefempty\glscustomtext
6903 {%
6904     \ifglshaslong{\glslabel}%
6905     {%
6906         \glssifplural
6907         {%

```

Plural form:

```

6908         \glscapscase
6909         {%

```

Plural form, don't adjust case:

```

6910         \glentrylongpl{\glslabel}\glinsert
6911         }%
6912         {%

```

Plural form, make first letter upper case:

```

6913         \Glsentrylongpl{\glslabel}\glinsert
6914         }%
6915         {%

```

Plural form, all caps:

```

6916         \mfirstucMakeUppercase
6917         {\glentrylongpl{\glslabel}\glinsert}%
6918         }%
6919         }%
6920         {%

```

Singular form

```

6921         \glscapscase
6922         {%

```

Singular form, don't adjust case:

```

6923         \glentrylong{\glslabel}\glinsert
6924         }%
6925         {%

```

Subsequent singular form, make first letter upper case:

```
6926      \Glsentrylong{\glslabel}\glsinsert
6927      }%
6928      {%
```

Subsequent singular form, all caps:

```
6929      \mfirstucMakeUppercase
6930      {\glsentrylong{\glslabel}\glsinsert}%
6931      }%
6932      }%
6933      }%
6934      {%
```

Not an acronym:

```
6935      \glsgenentryfmt
6936      }%
6937      }%
6938      {\glscustomtext\glsinsert}%
6939      }%
6940      {%
6941      \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%

6942      \renewcommand*{\acrfullfmt}[3]{%
6943      \glslink[##1]{##2}{\glsentrylong{##2}##3\space
6944      (\acronymfont{\glsentryshort{##2}})}}%
6945      \renewcommand*{\Acrfullfmt}[3]{%
6946      \glslink[##1]{##2}{\Glsentrylong{##2}##3\space
6947      (\acronymfont{\glsentryshort{##2}})}}%
6948      \renewcommand*{\ACRfullfmt}[3]{%
6949      \glslink[##1]{##2}{%
6950      \mfirstucMakeUppercase{\glsentrylong{##2}##3\space
6951      (\acronymfont{\glsentryshort{##2}})}}}%

6952      \renewcommand*{\acrfullplfmt}[3]{%
6953      \glslink[##1]{##2}{\glsentrylongpl{##2}##3\space
6954      (\acronymfont{\glsentryshortpl{##2}})}}%

6955      \renewcommand*{\Acrfullplfmt}[3]{%
6956      \glslink[##1]{##2}{\Glsentrylongpl{##2}##3\space
6957      (\acronymfont{\glsentryshortpl{##2}})}}%
6958      \renewcommand*{\ACRfullplfmt}[3]{%
6959      \glslink[##1]{##2}{%
6960      \mfirstucMakeUppercase{\glsentrylongpl{##2}##3\space
6961      (\acronymfont{\glsentryshortpl{##2}})}}}%
6962      \renewcommand*{\glsentryfull}[1]{%
6963      \glsentrylong{##1}\space(\acronymfont{\glsentryshort{##1}})%
6964      }%
6965      \renewcommand*{\Glsentryfull}[1]{%
6966      \Glsentrylong{##1}\space(\acronymfont{\glsentryshort{##1}})%
6967      }%
```

```

6968 \renewcommand*{\glsentryfullpl}[1]{%
6969   \glsentrylongpl{##1}\space(\acronymfont{\glsentryshortpl{##1}})%
6970 }%
6971 \renewcommand*{\Glsentryfullpl}[1]{%
6972   \Glsentrylongpl{##1}\space(\acronymfont{\glsentryshortpl{##1}})%
6973 }%
6974 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6975 \renewcommand*{\acronymsort}[2]{##1}%
6976 \renewcommand*{\acronymfont}[1]{##1}%
6977 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6978 }

```

dua-desc *<long>* only acronym style with user-supplied description.

```

6979 \newacronymstyle{dua-desc}%
6980 {%
6981   \GlsUseAcrEntryDisplayStyle{dua}%
6982 }%
6983 {%
6984   \GlsUseAcrStyleDefs{dua}%
6985   \renewcommand*{\GenericAcronymFields}{}%
6986   \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentrylong{##1}}}%
6987   \renewcommand*{\acronymsort}[2]{##2}%
6988 }%

```

footnote *<short>*\footnote{*<long>*} acronym style.

```

6989 \newacronymstyle{footnote}%
6990 {%
6991   Check for long form in case this is a mixed glossary.
6992   \ifglshaslong{\glslabel}{\glsngenacfmt}{\glsngenentryfmt}%
6993 }%
6994 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6995   Need to ensure hyperlinks are switched off on first use:
6996   \glshyperfirstfalse
6997   \renewcommand*{\genacrfullformat}[2]{%
6998     \protect\firstacronymfont{\glsentryshort{##1}}##2%
6999   }%
7000   \renewcommand*{\Genacrfullformat}[2]{%
7001     \firstacronymfont{\Glsentryshort{##1}}##2%
7002     \protect\footnote{\glsentrylong{##1}}%
7003   }%
7004   \renewcommand*{\genplacrfullformat}[2]{%
7005     \protect\firstacronymfont{\glsentryshortpl{##1}}##2%
7006     \protect\footnote{\glsentrylongpl{##1}}%
7007   }%
7008   \renewcommand*{\Genplacrfullformat}[2]{%

```



```

7009 \protect\firstacronymfont{\Glsentryshortpl{##1}}##2%
7010 \protect\footnote{\glsentrylongpl{##1}}%
7011 }%
7012 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
7013 \renewcommand*{\acronymsort}[2]{##1}%
7014 \renewcommand*{\acronymfont}[1]{##1}%
7015 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%

```

Don't use footnotes for \acrfull:

```

7016 \renewcommand*{\acrfullfmt}[3]{%
7017   \glslink[##1]{##2}{\acronymfont{\glsentryshort{##2}}##3\space
7018   (\glsentrylong{##2})}%
7019 \renewcommand*{\Acrfullfmt}[3]{%
7020   \glslink[##1]{##2}{\acronymfont{\Glsentryshort{##2}}##3\space
7021   (\glsentrylong{##2})}%
7022 \renewcommand*{\ACRfullfmt}[3]{%
7023   \glslink[##1]{##2}{%
7024     \mfirstucMakeUppercase{\acronymfont{\glsentryshort{##2}}##3\space
7025     (\glsentrylong{##2})}}}%
7026 \renewcommand*{\acrfullplfmt}[3]{%
7027   \glslink[##1]{##2}{\acronymfont{\glsentryshortpl{##2}}##3\space
7028   (\glsentrylongpl{##2})}%
7029 \renewcommand*{\Acrfullplfmt}[3]{%
7030   \glslink[##1]{##2}{\acronymfont{\Glsentryshortpl{##2}}##3\space
7031   (\glsentrylongpl{##2})}%
7032 \renewcommand*{\ACRfullplfmt}[3]{%
7033   \glslink[##1]{##2}{%
7034     \mfirstucMakeUppercase{\acronymfont{\glsentryshortpl{##2}}##3\space
7035     (\glsentrylongpl{##2})}}}%

```

Similarly for \glsentryfull etc:

```

7036 \renewcommand*{\glsentryfull}[1]{%
7037   \acronymfont{\glsentryshort{##1}}\space(\glsentrylong{##1})}%
7038 \renewcommand*{\Glsentryfull}[1]{%
7039   \acronymfont{\Glsentryshort{##1}}\space(\glsentrylong{##1})}%
7040 \renewcommand*{\glsentryfullpl}[1]{%
7041   \acronymfont{\glsentryshortpl{##1}}\space(\glsentrylongpl{##1})}%
7042 \renewcommand*{\Glsentryfullpl}[1]{%
7043   \acronymfont{\Glsentryshortpl{##1}}\space(\glsentrylongpl{##1})}%
7044 }

```

footnote-sc \textsc{<short>}\footnote{<long>} acronym style.

```

7045 \newacronymstyle{footnote-sc}%
7046 {%
7047   \GlsUseAcrEntryDispStyle{footnote}%
7048 }%
7049 {%
7050   \GlsUseAcrStyleDefs{footnote}%
7051   \renewcommand{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}
7052   \renewcommand{\acronymfont}[1]{\textsc{##1}}%

```

```

7053 \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7054 }%

```

footnote-sm \textsmaller{<short>}\footnote{<long>} acronym style.

```

7055 \newacronymstyle{footnote-sm}%
7056 {%
7057 \GlsUseAcrEntryDisplayStyle{footnote}%
7058 }%
7059 {%
7060 \GlsUseAcrStyleDefs{footnote}%
7061 \renewcommand{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}
7062 \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
7063 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
7064 }%

```

footnote-desc <short>\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

7065 \newacronymstyle{footnote-desc}%
7066 {%
7067 \GlsUseAcrEntryDisplayStyle{footnote}%
7068 }%
7069 {%
7070 \GlsUseAcrStyleDefs{footnote}%
7071 \renewcommand*{\GenericAcronymFields}{}%
7072 \renewcommand*{\acronymsort}[2]{##2}%
7073 \renewcommand*{\acronymentry}[1]{%
7074 \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
7075 }

```

ootnote-sc-desc \textsc{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

7076 \newacronymstyle{footnote-sc-desc}%
7077 {%
7078 \GlsUseAcrEntryDisplayStyle{footnote-sc}%
7079 }%
7080 {%
7081 \GlsUseAcrStyleDefs{footnote-sc}%
7082 \renewcommand*{\GenericAcronymFields}{}%
7083 \renewcommand*{\acronymsort}[2]{##2}%
7084 \renewcommand*{\acronymentry}[1]{%
7085 \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
7086 }

```

ootnote-sm-desc \textsmaller{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

7087 \newacronymstyle{footnote-sm-desc}%
7088 {%
7089 \GlsUseAcrEntryDisplayStyle{footnote-sm}%

```

```

7090 }%
7091 {%
7092   \GlsUseAcrStyleDefs{footnote-sm}%
7093   \renewcommand*{\GenericAcronymFields}{}%
7094   \renewcommand*{\acronymsort}[2]{##2}%
7095   \renewcommand*{\acronymentry}[1]{%
7096     \glstrylong{##1}\space (\acronymfont{\glstryshort{##1}})}%
7097 }

```

## AcronymSynonyms

```

7098 \newcommand*{\DefineAcronymSynonyms}{%

```

Short form

```
\acs
```

```
7099 \let\acs\acrshort
```

First letter uppercase short form

```
\Acs
```

```
7100 \let\Acs\Acrshort
```

Plural short form

```
\acsp
```

```
7101 \let\acsp\acrshortpl
```

First letter uppercase plural short form

```
\Acsp
```

```
7102 \let\Acsp\Acrshortpl
```

Long form

```
\acl
```

```
7103 \let\acl\aclong
```

Plural long form

```
\aclp
```

```
7104 \let\aclp\aclongpl
```

First letter upper case long form

```
\Acl
```

```
7105 \let\Acl\Aclong
```

First letter upper case plural long form

```
\Aclp
```

```
7106 \let\Aclp\Aclongpl
```

Full form

`\acf`

7107 `\let\acf\acrfull`

Plural full form

`\acfp`

7108 `\let\acfp\acrfullpl`

First letter upper case full form

`\Acf`

7109 `\let\Acf\Acrfull`

First letter upper case plural full form

`\Acfp`

7110 `\let\Acfp\Acrfullpl`

Standard form

`\ac`

7111 `\let\ac\gls`

First upper case standard form

`\Ac`

7112 `\let\Ac\Gls`

Standard plural form

`\acp`

7113 `\let\acp\glspl`

Standard first letter upper case plural form

`\Acp`

7114 `\let\Acp\Glspl`

7115 }

Define synonyms if required

7116 `\ifglsacrshortcuts`

7117 `\DefineAcronymSynonyms`

7118 `\fi`

These commands for setting the style are now deprecated but are kept for backward compatibility.

`\AcronymDisplayStyle` Sets the default acronym display style for given glossary.

7119 `\newcommand*{\SetDefaultAcronymDisplayStyle}[1]{%`

7120 `\defglsentryfmt[#1]{\glsentryfmt}%`

7121 }

```

\let\NewAcronymDef Sets up the acronym definition for the default style. The information is provided by the tokens
\glslabeltok, \glsshorttok, \glslongtok and \glskeylisttok.
7122 \newcommand*{\DefaultNewAcronymDef}{%
7123   \edef\@do@newglossaryentry{%
7124     \noexpand\newglossaryentry{\the\glslabeltok}%
7125     {%
7126       type=\acronymtype,%
7127       name={\the\glsshorttok},%
7128       sort={\the\glsshorttok},%
7129       text={\the\glsshorttok},%
7130       first={\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
7131       plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7132       firstplural={\acrfullformat{\noexpand\expandonce\noexpand\@glo@longpl}%
7133                   {\noexpand\expandonce\noexpand\@glo@shortpl}},%
7134       short={\the\glsshorttok},%
7135       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7136       long={\the\glslongtok},%
7137       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7138       description={\the\glslongtok},%
7139       descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%

Remaining options specified by the user:
7140       \the\glskeylisttok
7141     }%
7142   }%
7143   \let\@org@gls@assign@firstpl\gls@assign@firstpl
7144   \let\@org@gls@assign@plural\gls@assign@plural
7145   \let\@org@gls@assign@descplural\gls@assign@descplural
7146   \def\gls@assign@firstpl##1##2{%
7147     \@gls@expand@field{##1}{firstpl}{##2}%
7148   }%
7149   \def\gls@assign@plural##1##2{%
7150     \@gls@expand@field{##1}{plural}{##2}%
7151   }%
7152   \def\gls@assign@descplural##1##2{%
7153     \@gls@expand@field{##1}{descplural}{##2}%
7154   }%
7155   \@do@newglossaryentry
7156   \let\gls@assign@firstpl\@org@gls@assign@firstpl
7157   \let\gls@assign@plural\@org@gls@assign@plural
7158   \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7159 }

\let\SetAcronymStyle Set up the default acronym style:
7160 \newcommand*{\SetDefaultAcronymStyle}{%

Set the display style:
7161   \@for\@gls@type:=\@glsacronymlists\do{%
7162     \SetDefaultAcronymDisplayStyle{\@gls@type}%
7163   }%

```

Set up the definition of `\newacronym`:

```
7164 \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).

```
7165 \ifx\@glsacronymlists\@empty
7166 \def\@glo@type{\acronymtype}%
7167 \setkeys{glossentry}{##1}%
7168 \DeclareAcronymList{\@glo@type}%
7169 \SetDefaultAcronymDisplayStyle{\@glo@type}%
7170 \fi
7171 \glskeylisttok{##1}%
7172 \glslabeltok{##2}%
7173 \glsshorttok{##3}%
7174 \gslongtok{##4}%
7175 \newacronymhook
7176 \DefaultNewAcronymDef
7177 }%
7178 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
7179 }
```

`\acrfootnote` Used by the footnote acronym styles.

```
7180 \newcommand*{\acrfootnote}[3]{\acrlinkfootnote{#1}{#2}{#3}}
```

`\acrlinkfootnote`

```
7181 \newcommand*{\acrlinkfootnote}[3]{%
7182 \footnote{\glslink{#1}{#2}{#3}}%
7183 }
```

`\acrnoflinkfootnote`

```
7184 \newcommand*{\acrnoflinkfootnote}[3]{%
7185 \footnote{#3}%
7186 }
```

`\acronymDisplayStyle` Sets the acronym display style for given glossary for the description and footnote combination.

```
7187 \newcommand*{\SetDescriptionFootnoteAcronymDisplayStyle}[1]{%
7188 \defglsentryfmt{#1}{%

7189 \ifdefempty\glscustomtext
7190 {%
7191 \ifglsused{\glslabel}%
7192 {%
7193 \acronymfont{\glsentryfmt}%
7194 }%
7195 {%
7196 \firstacronymfont{\glsentryfmt}%
7197 \ifglsymbol{\glslabel}%
7198 {%
```

```

7199         \expandafter\protect\expandafter\acrfootnote\expandafter
7200         {\@gls@link@opts}{\@gls@link@label}%
7201         {%
7202         \glsifplural
7203         {\glsentrysymbolplural{\glslabel}}%
7204         {\glsentrysymbol{\glslabel}}%
7205         }%
7206     }%
7207 }%
7208 }%
7209 {\glscustomtext\glsinsert}%
7210 }%
7211 }

```

teNewAcronymDef

```

7212 \newcommand*{\DescriptionFootnoteNewAcronymDef}{%
7213 \edef\@do@newglossaryentry{%
7214 \noexpand\newglossaryentry{\the\glslabeltok}%
7215 {%
7216     type=\acronymtype,%
7217     name={\noexpand\acronymfont{\the\glsshorttok}},%
7218     sort={\the\glsshorttok},%
7219     first={\the\glsshorttok},%
7220     firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7221     text={\the\glsshorttok},%
7222     plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7223     short={\the\glsshorttok},%
7224     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7225     long={\the\glslongtok},%
7226     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7227     symbol={\the\glslongtok},%
7228     symbolplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7229     \the\glskeylisttok
7230 }%
7231 }%
7232 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7233 \let\@org@gls@assign@plural\gls@assign@plural
7234 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7235 \def\gls@assign@firstpl##1##2{%
7236     \@@gls@expand@field{##1}{firstpl}{##2}%
7237 }%
7238 \def\gls@assign@plural##1##2{%
7239     \@@gls@expand@field{##1}{plural}{##2}%
7240 }%
7241 \def\gls@assign@symbolplural##1##2{%
7242     \@@gls@expand@field{##1}{symbolplural}{##2}%
7243 }%
7244 \@do@newglossaryentry
7245 \let\gls@assign@plural\@org@gls@assign@plural

```

```

7246 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7247 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7248 }

```

`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.

```

7249 \newcommand*{\SetDescriptionFootnoteAcronymStyle}{%
7250   \renewcommand{\newacronym}[4][\]{%
7251     \ifx\@glsacronymlists\@empty
7252       \def\@glo@type{\acronymtype}%
7253       \setkeys{glossentry}{##1}%
7254       \DeclareAcronymList{\@glo@type}%
7255       \SetDescriptionFootnoteAcronymDisplayStyle{\@glo@type}%
7256     \fi
7257     \glskeylisttok{##1}%
7258     \glslabeltok{##2}%
7259     \glsshorttok{##3}%
7260     \glslongtok{##4}%
7261     \newacronymhook
7262     \DescriptionFootnoteNewAcronymDef
7263   }%

```

If footnote package option is specified, set the first use to append the long form (stored in symbol) as a footnote.

```

7264 \@for\@gls@type:=\@glsacronymlists\do{%
7265   \SetDescriptionFootnoteAcronymDisplayStyle{\@gls@type}%
7266 }%

```

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.

```

7267 \ifglsacrsmallcaps
7268   \renewcommand*{\acronymfont}[1]{\textsc{##1}}%
7269   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7270 \else
7271   \ifglsacrsmaller
7272     \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
7273   \fi
7274 \fi

```

Check for package option clash

```

7275 \ifglsacrdua
7276   \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’
7277     can’t both be set}{}%
7278 \fi
7279 }%

```

`nymDisplayStyle` Sets the acronym display style for given glossary with description and dua combination.



```

7280 \newcommand*{\SetDescriptionDUAAcronymDisplayStyle}[1]{%
7281   \def\glsentryfmt[#1]{\glsentryfmt}%
7282 }

```

#### UANewAcronymDef

```

7283 \newcommand*{\DescriptionDUANewAcronymDef}{%
7284   \edef\@do@newglossaryentry{%
7285     \noexpand\newglossaryentry{\the\glslabeltok}%
7286     {%
7287       type=\acronymtype,%
7288       name={\the\glslongtok},%
7289       sort={\the\glslongtok},%
7290       text={\the\glslongtok},%
7291       first={\the\glslongtok},%
7292       plural={\noexpand\expandonce\noexpand\@glo@longpl},%
7293       firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7294       short={\the\glsshorttok},%
7295       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7296       long={\the\glslongtok},%
7297       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7298       symbol={\the\glsshorttok},%
7299       symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7300       \the\glskeylisttok
7301     }%
7302   }%
7303   \let\@org@gls@assign@firstpl\gls@assign@firstpl
7304   \let\@org@gls@assign@plural\gls@assign@plural
7305   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7306   \def\gls@assign@firstpl##1##2{%
7307     \@@gls@expand@field{##1}{firstpl}{##2}%
7308   }%
7309   \def\gls@assign@plural##1##2{%
7310     \@@gls@expand@field{##1}{plural}{##2}%
7311   }%
7312   \def\gls@assign@symbolplural##1##2{%
7313     \@@gls@expand@field{##1}{symbolplural}{##2}%
7314   }%
7315   \@do@newglossaryentry
7316   \let\gls@assign@firstpl\@org@gls@assign@firstpl
7317   \let\gls@assign@plural\@org@gls@assign@plural
7318   \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7319 }

```

**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.

```

7320 \newcommand*{\SetDescriptionDUAAcronymStyle}{%
7321   \ifglssacrsmallcaps
7322     \PackageError{glossaries}{Option clash: 'smallcaps' and 'dua'

```

```

7323     can't both be set}{}%
7324 \else
7325     \ifglsacrsmaller
7326         \PackageError{glossaries}{Option clash: 'smaller' and 'dua'
7327             can't both be set}{}%
7328     \fi
7329 \fi
7330 \renewcommand{\newacronym}[4][{}]{%
7331     \ifx\@glsacronymlists\@empty
7332         \def\@glo@type{\acronymtype}%
7333         \setkeys{glossentry}{##1}%
7334         \DeclareAcronymList{\@glo@type}%
7335         \SetDescriptionDUAAcronymDisplayStyle{\@glo@type}%
7336     \fi
7337     \glskeylisttok{##1}%
7338     \glslabeltok{##2}%
7339     \glsshorttok{##3}%
7340     \glslongtok{##4}%
7341     \newacronymhook
7342     \DescriptionDUANewAcronymDef
7343 }%

Set display.
7344 \@for\@gls@type:=\@glsacronymlists\do{%
7345     \SetDescriptionDUAAcronymDisplayStyle{\@gls@type}%
7346 }%
7347 }%

```

`\nymDisplayStyle` Sets the acronym display style for given glossary using the description setting (but not footnote or dua).

```

7348 \newcommand*{\SetDescriptionAcronymDisplayStyle}[1]{%
7349     \defglsentryfmt[##1]{%

7350         \ifdefempty\glscustomtext
7351         {%
7352             \ifglsused{\glslabel}%
7353             {%

Move the inserted text outside of \acronymfont
7354                 \let\gls@org@insert\glsinsert
7355                 \let\glsinsert\@empty
7356                 \acronymfont{\glsgenentryfmt}\gls@org@insert
7357             }%
7358         {%
7359             \glsgenentryfmt
7360             \ifglshassymbol{\glslabel}%
7361             {%
7362                 \glsifplural
7363                 {%
7364                     \def\@glo@symbol{\glsentrysymbolplural{\glslabel}}%

```

```

7365         }%
7366         {%
7367         \def\@glo@symbol{\glsentrysymbol{\glslabel}}}%
7368         }%
7369         \space(\protect\firstacronymfont
7370         {\glscapscase
7371         {\@glo@symbol}
7372         {\@glo@symbol}
7373         {\mfirstucMakeUppercase{\@glo@symbol}}})}%
7374     }%
7375     {}%
7376 }%
7377 }%
7378 {\glscustomtext\glsinsert}%
7379 }%
7380 }

```

onNewAcronymDef

```

7381 \newcommand*{\DescriptionNewAcronymDef}{%
7382 \edef\@do@newglossaryentry{%
7383 \noexpand\newglossaryentry{\the\glslabeltok}%
7384 {%
7385 type=\acronymtype,%
7386 name={\noexpand
7387 \acronymformat{\the\glsshorttok}{\the\glslongtok}},%
7388 sort={\the\glsshorttok},%
7389 first={\the\glslongtok},%
7390 firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7391 text={\the\glsshorttok},%
7392 plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7393 short={\the\glsshorttok},%
7394 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7395 long={\the\glslongtok},%
7396 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7397 symbol={\noexpand\@glo@text},%
7398 symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7399 \the\glskeylisttok}%
7400 }%
7401 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7402 \let\@org@gls@assign@plural\gls@assign@plural
7403 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7404 \def\gls@assign@firstpl##1##2{%
7405 \@@gls@expand@field{##1}{firstpl}{##2}%
7406 }%
7407 \def\gls@assign@plural##1##2{%
7408 \@@gls@expand@field{##1}{plural}{##2}%
7409 }%
7410 \def\gls@assign@symbolplural##1##2{%
7411 \@@gls@expand@field{##1}{symbolplural}{##2}%

```

```

7412 }%
7413 \@do@newglossaryentry
7414 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7415 \let\gls@assign@plural\@org@gls@assign@plural
7416 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7417 }

```

**ionAcronymStyle** Option description is used, but not dua or footnote. Store long form in first key and short form in text and symbol key. The name is stored using `\acrnameformat` to allow the user to override the way the name is displayed in the list of acronyms.

```

7418 \newcommand*{\SetDescriptionAcronymStyle}{%
7419   \renewcommand{\newacronym}[4][]{%
7420     \ifx\@glsacronymlists\@empty
7421       \def\@glo@type{\acronymtype}%
7422       \setkeys{glossentry}{##1}%
7423       \DeclareAcronymList{\@glo@type}%
7424       \SetDescriptionAcronymDisplayStyle{\@glo@type}%
7425     \fi
7426     \glskeylisttok{##1}%
7427     \glslabeltok{##2}%
7428     \glsshorttok{##3}%
7429     \glslongtok{##4}%
7430     \newacronymhook
7431     \DescriptionNewAcronymDef
7432   }%

```

Set display.

```

7433 \@for\@gls@type:=\@glsacronymlists\do{%
7434   \SetDescriptionAcronymDisplayStyle{\@gls@type}%
7435 }%

```

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.

```

7436 \ifglsacrsmallcaps
7437   \renewcommand{\acronymfont}[1]{\textsc{##1}}
7438   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7439 \else
7440   \ifglsacrsmaller
7441     \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
7442   \fi
7443 \fi
7444 }%

```

**nymDisplayStyle** Sets the acronym display style for given glossary with footnote setting (but not description or dua).

```

7445 \newcommand*{\SetFootnoteAcronymDisplayStyle}[1]{%
7446   \defglsentryfmt[#1]{%
7447     \ifdefempty\glscustomtext
7448     {%

```

Move the inserted text outside of \acronymfont

```

7449 \let\gls@org@insert\glsinsert
7450 \let\glsinsert\@empty
7451 \ifglsused{\glslabel}%
7452 {%
7453 \acronymfont{\gls@org@insert\glsinsert}
7454 }%
7455 {%
7456 \firstacronymfont{\gls@org@insert\glsinsert}
7457 \ifgls@long{\glslabel}%
7458 {%
7459 \expandafter\protect\expandafter\acrfootnote\expandafter
7460 {\@gls@link@opts}{\@gls@link@label}%
7461 {%
7462 \glsifplural
7463 {\glsentrylongpl{\glslabel}}%
7464 {\glsentrylong{\glslabel}}%
7465 }%
7466 }%
7467 }%
7468 }%
7469 }%
7470 {\gls@customtext\glsinsert}%
7471 }%
7472 }

```

teNewAcronymDef

```

7473 \newcommand*{\FootnoteNewAcronymDef}{%
7474 \edef\@do@newglossaryentry{%
7475 \noexpand\newglossaryentry{\the\glslabeltok}%
7476 {%
7477 type=\acronymtype,%
7478 name={\noexpand\acronymfont{\the\glsshorttok}},%
7479 sort={\the\glsshorttok},%
7480 text={\the\glsshorttok},%
7481 plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7482 first={\the\glsshorttok},%
7483 firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7484 short={\the\glsshorttok},%
7485 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7486 long={\the\glslongtok},%
7487 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7488 description={\the\glslongtok},%
7489 descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7490 \the\glskeylisttok
7491 }%
7492 }%
7493 \let\@org@gls@assign@plural\gls@assign@plural

```

```

7494 \let\@org@gl@s@assign@firstpl\gl@s@assign@firstpl
7495 \let\@org@gl@s@assign@descplural\gl@s@assign@descplural
7496 \def\gl@s@assign@firstpl##1##2{%
7497   \@@gl@s@expand@field{##1}{firstpl}{##2}%
7498 }%
7499 \def\gl@s@assign@plural##1##2{%
7500   \@@gl@s@expand@field{##1}{plural}{##2}%
7501 }%
7502 \def\gl@s@assign@descplural##1##2{%
7503   \@@gl@s@expand@field{##1}{descplural}{##2}%
7504 }%
7505 \do@newglossaryentry
7506 \let\gl@s@assign@plural\@org@gl@s@assign@plural
7507 \let\gl@s@assign@firstpl\@org@gl@s@assign@firstpl
7508 \let\gl@s@assign@descplural\@org@gl@s@assign@descplural
7509 }

```

**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.

```

7510 \newcommand*\SetFootnoteAcronymStyle{%
7511   \renewcommand{\newacronym}[4][]{%
7512     \ifx\@gl@s@acronymlists\@empty
7513       \def\@glo@type{\acronymtype}%
7514       \setkeys{glossentry}{##1}%
7515       \DeclareAcronymList{\@glo@type}%
7516       \SetFootnoteAcronymDisplayStyle{\@glo@type}%
7517     \fi
7518     \glskeylisttok{##1}%
7519     \glslabeltok{##2}%
7520     \glsshorttok{##3}%
7521     \glslongtok{##4}%
7522     \newacronymhook
7523     \FootnoteNewAcronymDef
7524   }%

```

Set display

```

7525   \@for\@gl@s@type:=\@gl@s@acronymlists\do{%
7526     \SetFootnoteAcronymDisplayStyle{\@gl@s@type}%
7527   }%

```

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.

```

7528   \ifgl@s@crsmallcaps
7529     \renewcommand*\acronymfont[1]{\textsc{##1}}%
7530     \renewcommand*\acrpluralsuffix{\gl@supacrpluralsuffix}%
7531   \else
7532     \ifgl@s@crsmaller
7533       \renewcommand*\acronymfont[1]{\textsmaller{##1}}%
7534     \fi
7535   \fi

```

Check for option clash

```
7536 \ifglsacrdue
7537 \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’
7538 can’t both be set}}{}%
7539 \fi
7540 }%
```

`\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.

```
7541 \DeclareRobustCommand*\glsdoparenifnotempty}[2]{%
7542 \protected@edef\gls@tmp{#1}%
7543 \ifdefempty\gls@tmp
7544 {}%
7545 {%
7546 \ifx\gls@tmp\@gls@default@value
7547 \else
7548 \space (#2{#1})%
7549 \fi
7550 }%
7551 }
```

`\nymDisplayStyle` Sets the acronym display style for given glossary where neither footnote nor description is required, but smallcaps or smaller specified.

```
7552 \newcommand*\SetSmallAcronymDisplayStyle}[1]{%
7553 \defglsentryfmt[#1]{%
7554 \ifdefempty\glscustomtext
7555 {%
```

Move the inserted text outside of `\acronymfont`

```
7556 \let\gls@org@insert\glsinsert
7557 \let\glsinsert\@empty
7558 \ifglsused{\glslabel}%
7559 {%
7560 \acronymfont{\glsentryfmt}\gls@org@insert
7561 }%
7562 {%
7563 \glsentryfmt
7564 \ifglshassymbol{\glslabel}%
7565 {%
7566 \glsifplural
7567 {%
7568 \def\@glo@symbol{\glsentrysymbolplural{\glslabel}}%
7569 }%
7570 {%
7571 \def\@glo@symbol{\glsentrysymbol{\glslabel}}%
7572 }%
7573 \space
7574 (\glscapscase
```

```

7575         {\firstacronymfont{\@glo@symbol}}}%
7576         {\firstacronymfont{\@glo@symbol}}}%
7577         {\firstacronymfont{\mfirstucMakeUppercase{\@glo@symbol}}})}%
7578     }%
7579     {}%
7580 }%
7581 }%
7582 {\glscustomtext\glsinsert}%
7583 }%
7584 }

```

# 11NewAcronymDef

```

7585 \newcommand*{\SmallNewAcronymDef}{%
7586   \edef\@do@newglossaryentry{%
7587     \noexpand\newglossaryentry{\the\glslabeltok}%
7588     {%
7589       type=\acronymtype,%
7590       name={\noexpand\acronymfont{\the\glsshorttok}},%
7591       sort={\the\glsshorttok},%
7592       text={\the\glsshorttok},%
7593       plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7594       first={\the\glslongtok},%
7595       firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7596       short={\the\glsshorttok},%
7597       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7598       long={\the\glslongtok},%
7599       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7600       description={\noexpand\@glo@first},%
7601       descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7602       symbol={\the\glsshorttok},%
7603       symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7604       \the\glskeylisttok
7605     }%
7606   }%
7607   \let\@org@gls@assign@firstpl\gls@assign@firstpl
7608   \let\@org@gls@assign@plural\gls@assign@plural
7609   \let\@org@gls@assign@descplural\gls@assign@descplural
7610   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7611   \def\gls@assign@firstpl##1##2{%
7612     \@gls@expand@field{##1}{firstpl}{##2}%
7613   }%
7614   \def\gls@assign@plural##1##2{%
7615     \@gls@expand@field{##1}{plural}{##2}%
7616   }%

```



```

7617 \def\gls@assign@descplural##1##2{%
7618   \@gls@expand@field{##1}{descplural}{##2}%
7619 }%
7620 \def\gls@assign@symbolplural##1##2{%
7621   \@gls@expand@field{##1}{symbolplural}{##2}%
7622 }%
7623 \do@newglossaryentry
7624 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7625 \let\gls@assign@plural\@org@gls@assign@plural
7626 \let\gls@assign@descplural\@org@gls@assign@descplural
7627 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7628 }

```

`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.

```

7629 \newcommand*{\SetSmallAcronymStyle}{%
7630   \renewcommand{\newacronym}[4][]{%
7631     \ifx\glsacronymlists\@empty
7632       \def\@glo@type{\acronymtype}%
7633       \setkeys{glossentry}{##1}%
7634       \DeclareAcronymList{\@glo@type}%
7635       \SetSmallAcronymDisplayStyle{\@glo@type}%
7636     \fi
7637     \glskeylisttok{##1}%
7638     \glslabeltok{##2}%
7639     \glsshorttok{##3}%
7640     \glslongtok{##4}%
7641     \newacronymhook
7642     \SmallNewAcronymDef
7643   }%

```

Change the display since first only contains long form.

```

7644 \@for\@gls@type:=\@glsacronymlists\do{%
7645   \SetSmallAcronymDisplayStyle{\@gls@type}%
7646 }%

```

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.

```

7647 \ifglsacrsmallcaps
7648   \renewcommand*{\acronymfont}[1]{\textsc{##1}}
7649   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7650 \else
7651   \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}
7652 \fi

```

check for option clash

```

7653 \ifglsacrdua
7654   \ifglsacrsmallcaps
7655     \PackageError{glossaries}{Option clash: ‘smallcaps’ and ‘dua’
7656       can’t both be set}{}%

```

```

7657 \else
7658 \PackageError{glossaries}{Option clash: ‘smaller’ and ‘dua’
7659 can’t both be set}{}%
7660 \fi
7661 \fi
7662 }%

```

**DUADisplayStyle** Sets the acronym display style for given glossary with dua setting.

```

7663 \newcommand*{\SetDUADisplayStyle}[1]{%
7664 \def\glsentryfmt[#1]{\glsentryfmt}%
7665 }

```

**UANewAcronymDef**

```

7666 \newcommand*{\DUANewAcronymDef}{%
7667 \edef\@do@newglossaryentry{%
7668 \noexpand\newglossaryentry{\the\glslabeltok}%
7669 {%
7670 type=\acronymtype,%
7671 name={\the\glsshorttok},%
7672 text={\the\glslongtok},%
7673 first={\the\glslongtok},%
7674 plural={\noexpand\expandonce\noexpand\@glo@longpl},%
7675 firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7676 short={\the\glsshorttok},%
7677 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7678 long={\the\glslongtok},%
7679 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7680 description={\the\glslongtok},%
7681 descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7682 symbol={\the\glsshorttok},%
7683 symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7684 \the\glskeylisttok
7685 }%
7686 }%
7687 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7688 \let\@org@gls@assign@plural\gls@assign@plural
7689 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7690 \let\@org@gls@assign@descplural\gls@assign@descplural
7691 \def\gls@assign@firstpl##1##2{%
7692 \@@gls@expand@field{##1}{firstpl}{##2}%
7693 }%
7694 \def\gls@assign@plural##1##2{%
7695 \@@gls@expand@field{##1}{plural}{##2}%
7696 }%
7697 \def\gls@assign@symbolplural##1##2{%
7698 \@@gls@expand@field{##1}{symbolplural}{##2}%
7699 }%
7700 \def\gls@assign@descplural##1##2{%
7701 \@@gls@expand@field{##1}{descplural}{##2}%

```

```

7702 }%
7703 \@do@newglossaryentry
7704 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7705 \let\gls@assign@plural\@org@gls@assign@plural
7706 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7707 \let\gls@assign@descplural\@org@gls@assign@descplural
7708 }

```

`\SetDUASStyle` Always expand acronyms.

```

7709 \newcommand*{\SetDUASStyle}{%
7710   \renewcommand{\newacronym}[4][]{%
7711     \ifx\@glsacronymlists\@empty
7712       \def\@glo@type{\acronymtype}%
7713       \setkeys{glossentry}{##1}%
7714       \DeclareAcronymList{\@glo@type}%
7715       \SetDUADisplayStyle{\@glo@type}%
7716     \fi
7717     \glskeylisttok{##1}%
7718     \glslabeltok{##2}%
7719     \glsshorttok{##3}%
7720     \glslongtok{##4}%
7721     \newacronymhook
7722     \DUANewAcronymDef
7723   }%
7724   \@for\@gls@type:=\@glsacronymlists\do{%
7725     \SetDUADisplayStyle{\@gls@type}%
7726   }%
7727 }

```

Set the display

`SetAcronymStyle`

```

7728 \newcommand*{\SetAcronymStyle}{%
7729   \SetDefaultAcronymStyle
7730   \ifglsacrdescription
7731     \ifglsacrfootnote
7732       \SetDescriptionFootnoteAcronymStyle
7733     \else
7734       \ifglsacrdua
7735         \SetDescriptionDUAAcronymStyle
7736       \else
7737         \SetDescriptionAcronymStyle
7738       \fi
7739     \fi
7740   \else
7741     \ifglsacrfootnote
7742       \SetFootnoteAcronymStyle
7743     \else
7744       \ifthenelse{\boolean{glsacrsmalldcaps}}{OR
7745         \boolean{glsacrsmaller}}}%

```

```

7746      {%
7747      \SetSmallAcronymStyle
7748      }%
7749      {%
7750      \ifglsacrdua
7751      \SetDUASStyle
7752      \fi
7753      }%
7754      \fi
7755      \fi
7756 }

```

Set the acronym style according to the package options

```

7757 \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.

`\setacronymstyle` Sets the acronym display style.

```

7758 \newcommand*{\SetCustomDisplayStyle}[1]{%
7759   \defglsentryfmt[#1]{\glsentryfmt}%
7760 }

```

`\setacronymfields`

```

7761 \newcommand*{\CustomAcronymFields}{%
7762   name={\the\glsshorttok},%
7763   description={\the\glslongtok},%
7764   first={\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
7765   firstplural={\acrfullformat
7766     {\noexpand\glsentrylongpl{\the\glslabeltok}}}%
7767     {\noexpand\glsentryshortpl{\the\glslabeltok}}},%
7768   text={\the\glsshorttok},%
7769   plural={\the\glsshorttok\noexpand\acrpluralsuffix}%
7770 }

```

`\setnewacronym`

```

7771 \newcommand*{\CustomNewAcronymDef}{%
7772   \protected@edef\do@newglossaryentry{%
7773     \noexpand\newglossaryentry{\the\glslabeltok}%
7774     {%
7775       type=\acronymtype,%
7776       short={\the\glsshorttok},%
7777       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7778       long={\the\glslongtok},%
7779       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7780       user1={\the\glsshorttok},%

```

```

7781      user2={\the\glsshorttok\noexpand\acrpluralsuffix},%
7782      user3={\the\glslongtok},%
7783      user4={\the\glslongtok\noexpand\acrpluralsuffix},%
7784      \CustomAcronymFields,%
7785      \the\glskeylisttok
7786    }%
7787  }%
7788  \@do@newglossaryentry
7789 }

```

`\SetCustomStyle`

```

7790 \newcommand*{\SetCustomStyle}{%
7791   \renewcommand{\newacronym}[4][]{%
7792     \ifx\@glsacronymlists\@empty
7793       \def\@glo@type{\acronymtype}%
7794       \setkeys{glossentry}{##1}%
7795       \DeclareAcronymList{\@glo@type}%
7796       \SetCustomDisplayStyle{\@glo@type}%
7797     \fi
7798     \glskeylisttok{##1}%
7799     \glslabeltok{##2}%
7800     \glsshorttok{##3}%
7801     \glslongtok{##4}%
7802     \newacronymhook
7803     \CustomNewAcronymDef
7804   }%
7805   \@for\@gls@type:=\@glsacronymlists\do{%
7806     \SetCustomDisplayStyle{\@gls@type}%
7807   }%
7808 }

```

Set the display

## 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.

```
7809 \RequirePackage{glossary-hypernav}
```

The styles that use list-like environments. These are not loaded if the `nolist` option is used:

```
7810 \@gls@loadlist
```

The styles that use the `longtable` environment. These are not loaded if the `nolong` package option is used.

```
7811 \@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.

```
7812 \@gls@loadsuper
```

The tree-like styles. These are not loaded if the notree package option is used.

```
7813 \@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.

```
7814 \ifx\@glossary@default@style\relax
```

```
7815 \else
```

```
7816   \setglossarystyle{\@glossary@default@style}
```

```
7817 \fi
```

## 1.20 Debugging Commands

```
\showgloparent \showgloparent{\<label>}
```

```
7818 \newcommand*{\showgloparent}[1]{%
```

```
7819   \expandafter\show\csname glo@\glsdetoklabel{#1}@parent\endcsname
```

```
7820 }
```

```
\showglolevel \showglolevel{\<label>}
```

```
7821 \newcommand*{\showglolevel}[1]{%
```

```
7822   \expandafter\show\csname glo@\glsdetoklabel{#1}@level\endcsname
```

```
7823 }
```

```
\showglotext \showglotext{\<label>}
```

```
7824 \newcommand*{\showglotext}[1]{%
```

```
7825   \expandafter\show\csname glo@\glsdetoklabel{#1}@text\endcsname
```

```
7826 }
```

```
\showgloplural \showgloplural{\<label>}
```

```
7827 \newcommand*{\showgloplural}[1]{%
```

```
7828   \expandafter\show\csname glo@\glsdetoklabel{#1}@plural\endcsname
```

```
7829 }
```

```
\showglofirst \showglofirst{\<label>}
```

```

7830 \newcommand*{\showglofirst}[1]{%
7831   \expandafter\show\csname glo\glsdetoklabel{#1}@first\endcsname
7832 }

```

\showglofirstpl \showglofirstpl{\label{}}

```

7833 \newcommand*{\showglofirstpl}[1]{%
7834   \expandafter\show\csname glo\glsdetoklabel{#1}@firstpl\endcsname
7835 }

```

\showglotype \showglotype{\label{}}

```

7836 \newcommand*{\showglotype}[1]{%
7837   \expandafter\show\csname glo\glsdetoklabel{#1}@type\endcsname
7838 }

```

\showglocounter \showglocounter{\label{}}

```

7839 \newcommand*{\showglocounter}[1]{%
7840   \expandafter\show\csname glo\glsdetoklabel{#1}@counter\endcsname
7841 }

```

\showglouserii \showglouserii{\label{}}

```

7842 \newcommand*{\showglouserii}[1]{%
7843   \expandafter\show\csname glo\glsdetoklabel{#1}@userii\endcsname
7844 }

```

\showglouseriii \showglouseriii{\label{}}

```

7845 \newcommand*{\showglouseriii}[1]{%
7846   \expandafter\show\csname glo\glsdetoklabel{#1}@useriii\endcsname
7847 }

```

\showglouseriiii \showglouseriiii{\label{}}

```

7848 \newcommand*{\showglouseriii}[1]{%
7849   \expandafter\show\csname glo\glsdetoklabel{#1}@useriii\endcsname
7850 }

```

\showglouseriv \showglouseriv{<label>}

```

7851 \newcommand*{\showglouseriv}[1]{%
7852   \expandafter\show\csname glo\glsdetoklabel{#1}@useriv\endcsname
7853 }

```

\showglouserv \showglouserv{<label>}

```

7854 \newcommand*{\showglouserv}[1]{%
7855   \expandafter\show\csname glo\glsdetoklabel{#1}@userv\endcsname
7856 }

```

\showglouservi \showglouservi{<label>}

```

7857 \newcommand*{\showglouservi}[1]{%
7858   \expandafter\show\csname glo\glsdetoklabel{#1}@uservi\endcsname
7859 }

```

\showgloname \showgloname{<label>}

```

7860 \newcommand*{\showgloname}[1]{%
7861   \expandafter\show\csname glo\glsdetoklabel{#1}@name\endcsname
7862 }

```

\showglodesc \showglodesc{<label>}

```

7863 \newcommand*{\showglodesc}[1]{%
7864   \expandafter\show\csname glo\glsdetoklabel{#1}@desc\endcsname
7865 }

```

howglodescplural \showglodescplural{<label>}



```

7866 \newcommand*{\showglodescplural}[1]{%
7867   \expandafter\show\csname glo@glstdetoklabel{#1}@descplural\endcsname
7868 }

```

\showglosort    \showglosort{<label>}

```

7869 \newcommand*{\showglosort}[1]{%
7870   \expandafter\show\csname glo@glstdetoklabel{#1}@sort\endcsname
7871 }

```

\showglosymbol    \showglosymbol{<label>}

```

7872 \newcommand*{\showglosymbol}[1]{%
7873   \expandafter\show\csname glo@glstdetoklabel{#1}@symbol\endcsname
7874 }

```

wglosymbolplural    \showglosymbolplural{<label>}

```

7875 \newcommand*{\showglosymbolplural}[1]{%
7876   \expandafter\show\csname glo@glstdetoklabel{#1}@symbolplural\endcsname
7877 }

```

\showgloshort    \showgloshort{<label>}

```

7878 \newcommand*{\showgloshort}[1]{%
7879   \expandafter\show\csname glo@glstdetoklabel{#1}@short\endcsname
7880 }

```

\showglolong    \showglolong{<label>}

```

7881 \newcommand*{\showglolong}[1]{%
7882   \expandafter\show\csname glo@glstdetoklabel{#1}@long\endcsname
7883 }

```

\showgloindex    \showgloindex{<label>}

```

7884 \newcommand*{\showgloindex}[1]{%
7885   \expandafter\show\csname glo@\glsdetoklabel{#1}@index\endcsname
7886 }

```

\showgloflag    \showgloflag{<label>}

```

7887 \newcommand*{\showgloflag}[1]{%
7888   \expandafter\show\csname ifglo@\glsdetoklabel{#1}@flag\endcsname
7889 }

```

\showgloloclist    \showgloloclist{<label>}

```

7890 \newcommand*{\showgloloclist}[1]{%
7891   \expandafter\show\csname glo@\glsdetoklabel{#1}@loclist\endcsname
7892 }

```

\showglofield    \showglofield{<label>}{<field>}

```

7893 \newcommand*{\showglofield}[2]{%
7894   \csshow{glo@\glsdetoklabel{#1}@#2}%
7895 }

```

showacronymlists    \showacronymlists

Show list of glossaries that have been flagged as a list of acronyms.

```

7896 \newcommand*{\showacronymlists}{%
7897   \show\@glsacronymlists
7898 }

```

\showglossaries    \showglossaries

Show list of defined glossaries.

```

7899 \newcommand*{\showglossaries}{%
7900   \show\@glo@types
7901 }

```

\showglossaryin    \showglossaryin{<glossary-label>}

Show the ‘in’ extension for the given glossary.

```
7902 \newcommand*{\showglossaryin}[1]{%
7903   \expandafter\show\csname @glotype@#1@in\endcsname
7904 }
```

\showglossaryout    \showglossaryout{<glossary-label>}

Show the ‘out’ extension for the given glossary.

```
7905 \newcommand*{\showglossaryout}[1]{%
7906   \expandafter\show\csname @glotype@#1@out\endcsname
7907 }
```

showglossarytitle    \showglossarytitle{<glossary-label>}

Show the title for the given glossary.

```
7908 \newcommand*{\showglossarytitle}[1]{%
7909   \expandafter\show\csname @glotype@#1@title\endcsname
7910 }
```

wglossarycounter    \showglossarycounter{<glossary-label>}

Show the counter for the given glossary.

```
7911 \newcommand*{\showglossarycounter}[1]{%
7912   \expandafter\show\csname @glotype@#1@counter\endcsname
7913 }
```

wglossaryentries    \showglossaryentries{<glossary-label>}

Show the list of entry labels for the given glossary.

```
7914 \newcommand*{\showglossaryentries}[1]{%
7915   \expandafter\show\csname glolist@#1\endcsname
7916 }
```

## 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.

```

7917\csname ifglcompatible-2.07\endcsname
7918  \RequirePackage{glossaries-compatible-207}
7919\fi

```

## 2 Prefix Support (glossaries-prefix Code)

This package provides a means of adding prefixes to your glossary entries. For example, you may want to use “`\gls{<label>}`” on first use but use “`\an \gls{<label>}`” on subsequent use.

```
7920 \NeedsTeXFormat{LaTeX2e}
```

```
7921 \ProvidesPackage{glossaries-prefix}[2018/04/07 v4.37 (NLCT)]
```

Pass all options to glossaries:

```
7922 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
7923 \ProcessOptions
```

Load glossaries:

```
7924 \RequirePackage{glossaries}
```

Add the new keys:

```
7925 \define@key{glossentry}{prefixfirst}{\def\@glo@entryprefixfirst{#1}}%
```

```
7926 \define@key{glossentry}{prefixfirstplural}{\def\@glo@entryprefixfirstplural{#1}}%
```

```
7927 \define@key{glossentry}{prefix}{\def\@glo@entryprefix{#1}}%
```

```
7928 \define@key{glossentry}{prefixplural}{\def\@glo@entryprefixplural{#1}}%
```

Add them to `\gls@keymap`:

```
7929 \appto\@gls@keymap{,%
```

```
7930   {prefixfirst}{prefixfirst},%
```

```
7931   {prefixfirstplural}{prefixfirstplural},%
```

```
7932   {prefix}{prefix},%
```

```
7933   {prefixplural}{prefixplural}}%
```

```
7934 }
```

Set the default values:

```
7935 \appto\@newglossaryentryprehook{%
```

```
7936   \def\@glo@entryprefix{}}%
```

```
7937   \def\@glo@entryprefixplural{}}%
```

```
7938   \let\@glo@entryprefixfirst\@gls@default@value
```

```
7939   \let\@glo@entryprefixfirstplural\@gls@default@value
```

```
7940 }
```

Set the assignment code:

```
7941 \appto\@newglossaryentryposthook{%
```

```
7942   \gls@assign@field{ }\@glo@label{prefix}{\@glo@entryprefix}}%
```

```
7943   \gls@assign@field{ }\@glo@label{prefixplural}{\@glo@entryprefixplural}}%
```

If `prefixfirst` has not been supplied, make it the same as `prefix`.

```
7944 \expandafter\gls@assign@field\expandafter
```

```
7945   {\csname glo@\@glo@label @prefix\endcsname}{\@glo@label}{prefixfirst}}%
```

```
7946   {\@glo@entryprefixfirst}}%
```

If prefixfirstplural has not been supplied, make it the same as prefixplural.

```

7947 \expandafter\gls@assign@field\expandafter
7948   {\csname glo@\@glo@label @prefixplural\endcsname}{\@glo@label}%
7949   {prefixfirstplural}{\@glo@entryprefixfirstplural}%
7950 }

```

Define commands to access these fields:

entryprefixfirst

```

7951 \newcommand*{\glsentryprefixfirst}[1]{\csuse{glo@#1@prefixfirst}}

```

entryfirstplural

```

7952 \newcommand*{\glsentryprefixfirstplural}[1]{\csuse{glo@#1@prefixfirstplural}}

```

\glsentryprefix

```

7953 \newcommand*{\glsentryprefix}[1]{\csuse{glo@#1@prefix}}

```

entryprefixplural

```

7954 \newcommand*{\glsentryprefixplural}[1]{\csuse{glo@#1@prefixplural}}

```

Now for the initial upper case variants:

entryprefixfirst

```

7955 \newrobustcmd*{\Glsentryprefixfirst}[1]{%
7956   \protected@edef\@glo@text{\csname glo@#1@prefixfirst\endcsname}%
7957   \xmakefirstuc\@glo@text
7958 }

```

entryfirstplural

```

7959 \newrobustcmd*{\Glsentryprefixfirstplural}[1]{%
7960   \protected@edef\@glo@text{\csname glo@#1@prefixfirstplural\endcsname}%
7961   \xmakefirstuc\@glo@text
7962 }

```

\Glsentryprefix

```

7963 \newrobustcmd*{\Glsentryprefix}[1]{%
7964   \protected@edef\@glo@text{\csname glo@#1@prefix\endcsname}%
7965   \xmakefirstuc\@glo@text
7966 }

```

entryprefixplural

```

7967 \newrobustcmd*{\Glsentryprefixplural}[1]{%
7968   \protected@edef\@glo@text{\csname glo@#1@prefixplural\endcsname}%
7969   \xmakefirstuc\@glo@text
7970 }

```

Define commands to determine if the prefix keys have been set:

\ifglshasprefix

```
7971 \newcommand*{\ifglshasprefix}[3]{%
7972   \ifcempty{glo@#1@prefix}%
7973   {#3}%
7974   {#2}%
7975 }
```

hasprefixplural

```
7976 \newcommand*{\ifglshasprefixplural}[3]{%
7977   \ifcempty{glo@#1@prefixplural}%
7978   {#3}%
7979   {#2}%
7980 }
```

shasprefixfirst

```
7981 \newcommand*{\ifglshasprefixfirst}[3]{%
7982   \ifcempty{glo@#1@prefixfirst}%
7983   {#3}%
7984   {#2}%
7985 }
```

efixfirstplural

```
7986 \newcommand*{\ifglshasprefixfirstplural}[3]{%
7987   \ifcempty{glo@#1@prefixfirstplural}%
7988   {#3}%
7989   {#2}%
7990 }
```

Define commands that insert the prefix before commands like \gls:

\pgls

```
7991 \newrobustcmd{\pgls}{\@gls@hyp@opt\@pgls}
```

\@pgls Unstarred version.

```
7992 \newcommand*{\@pgls}[2][ ]{%
7993   \new@ifnextchar[%
7994     {\@pgls@{#1}{#2}}%
7995     {\@pgls@{#1}{#2}[ ]}%
7996 }
```

\@pgls@ Read in the final optional argument:

```
7997 \def\@pgls@#1#2[#3]{%
7998   \glsdoifexists{#2}%
7999   {%
8000     \ifglsused{#2}%
8001     {%
8002       \glstryprefix{#2}%
8003     }%

```

```

8004     {%
8005         \glsentryprefixfirst{#2}%
8006     }%
8007     \@gls@{#1}{#2}[#3]%
8008 }%
8009 }

```

Similarly for the plural version:

```

\pglsp1
8010 \newrobustcmd{\pglsp1}{\@gls@hyp@opt\@pglsp1}

```

\@pglsp1 Unstarred version.

```

8011 \newcommand*{\@pglsp1}[2][ ]{%
8012     \new@ifnextchar[%
8013     {\@pglsp1@{#1}{#2}}%
8014     {\@pglsp1@{#1}{#2}[ ]}%
8015 }

```

\@pglsp1@ Read in the final optional argument:

```

8016 \def\@pglsp1@#1#2[#3]{%
8017     \glsdoifexists{#2}%
8018     {%
8019         \ifglsused{#2}%
8020         {%
8021             \glsentryprefixplural{#2}%
8022         }%
8023         {%
8024             \glsentryprefixfirstplural{#2}%
8025         }%
8026         \@glspl@{#1}{#2}[#3]%
8027     }%
8028 }

```

Now for the first letter upper case versions:

```

\Pgls
8029 \newrobustcmd{\Pgls}{\@gls@hyp@opt\@Pgls}

```

\@Pgls Unstarred version.

```

8030 \newcommand*{\@Pgls}[2][ ]{%
8031     \new@ifnextchar[%
8032     {\@Pgls@{#1}{#2}}%
8033     {\@Pgls@{#1}{#2}[ ]}%
8034 }

```

\@Pgls@ Read in the final optional argument:

```

8035 \def\@Pgls@#1#2[#3]{%

```



```

8036 \glsdoifexists{#2}%
8037 {%
8038   \ifglsused{#2}%
8039   {%
8040     \ifglshasprefix{#2}%
8041     {%
8042       \Glsentryprefix{#2}%
8043       \@gls@{#1}{#2}[#3]%
8044     }%
8045     {\@Gls@{#1}{#2}[#3]}%
8046   }%
8047   {%
8048     \ifglshasprefixfirst{#2}%
8049     {%
8050       \Glsentryprefixfirst{#2}%
8051       \@gls@{#1}{#2}[#3]%
8052     }%
8053     {\@Gls@{#1}{#2}[#3]}%
8054   }%
8055 }%
8056 }

```

Similarly for the plural version:

```

\Pglspl
8057 \newrobustcmd{\Pglspl}{\@gls@hyp@opt\@Pglspl}

```

\@Pglspl Unstarred version.

```

8058 \newcommand*{\@Pglspl}[2] [] {%
8059   \new@ifnextchar[%
8060   {\@Pglspl@{#1}{#2}}%
8061   {\@Pglspl@{#1}{#2} []}%
8062 }

```

\@Pglspl@ Read in the final optional argument:

```

8063 \def\@Pglspl@#1#2[#3] {%
8064   \glsdoifexists{#2}%
8065   {%
8066     \ifglsused{#2}%
8067     {%
8068       \ifglshasprefixplural{#2}%
8069       {%
8070         \Glsentryprefixplural{#2}%
8071         \@glspl@{#1}{#2}[#3]%
8072       }%
8073       {\@Glspl@{#1}{#2}[#3]}%
8074     }%
8075     {%
8076       \ifglshasprefixfirstplural{#2}%

```

```

8077      {%
8078      \Glsentryprefixfirstplural{#2}%
8079      \@glsp1@{#1}{#2}[#3]%
8080      }%
8081      {\@Glspl@{#1}{#2}[#3]}%
8082      }%
8083      }%
8084      }

```

Finally the all upper case versions:

\PGLS

```

8085 \newrobustcmd{\PGLS}{\@gls@hyp@opt\PGLS}

```

\@PGLS Unstarred version.

```

8086 \newcommand*{\@PGLS}[2][{}]{%
8087   \new@ifnextchar[%
8088   {\@PGLS@{#1}{#2}}%
8089   {\@PGLS@{#1}{#2}[]}%
8090   }

```

\@PGLS@ Read in the final optional argument:

```

8091 \def\@PGLS@#1#2[#3]{%
8092   \glsdoifexists{#2}%
8093   {%
8094     \ifglsused{#2}%
8095     {%
8096       \mfirstucMakeUppercase{\glsentryprefix{#2}}%
8097     }%
8098     {%
8099       \mfirstucMakeUppercase{\glsentryprefixfirst{#2}}%
8100     }%
8101     \@GLS@{#1}{#2}[#3]%
8102   }%
8103   }

```

Plural version:

\PGLSp1

```

8104 \newrobustcmd{\PGLSp1}{\@gls@hyp@opt\PGLSp1}

```

\@PGLSp1 Unstarred version.

```

8105 \newcommand*{\@PGLSp1}[2][{}]{%
8106   \new@ifnextchar[%
8107   {\@PGLSp1@{#1}{#2}}%
8108   {\@PGLSp1@{#1}{#2}[]}%
8109   }

```

\@PGLSp1@ Read in the final optional argument:

```
8110 \def\@PGLSp1@#1#2[#3]{%
8111   \glsdoifexists{#2}%
8112   {%
8113     \ifglsused{#2}%
8114     {%
8115       \mfirstucMakeUppercase{\glsentryprefixplural{#2}}%
8116     }%
8117     {%
8118       \mfirstucMakeUppercase{\glsentryprefixfirstplural{#2}}%
8119     }%
8120     \@GLSp1@{#1}{#2}[#3]%
8121   }%
8122 }
```

## 3 Glossary Styles

### 3.1 Glossary hyper-navigation definitions (glossary-hypernav package)

Package Definition:

```
8123 \ProvidesPackage{glossary-hypernav}[2018/04/07 v4.37 (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`

```
8124 \newcommand*{\glsnavhyperlink}[3][\@glo@type]{%
8125   \edef\gls@grplabel{#2}\protected@edef\gls@grptitle{#3}%
8126   \@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.

```
8127 \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.

`snahypertarget`

```
8128 \newcommand*{\glsnavhypertarget}[3][\@glo@type]{%
8129   \@glsnavhypertarget{#1}{#2}{#3}%
8130 }
```

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.

`snahypertarget`

```
8131 \newcommand*{\@glsnavhypertarget}[3]{%}
```

Add this group to the aux file for re-run check.

```
8132 \protected@write\auxout{}\string\@gls@hypergroup{#1}{#2}}%
```

Add the target.

```
8133 \glstarget{\glsnavhyperlinkname{#1}{#2}}{#3}%
```

Check list of known groups to determine if a re-run is required.

```
8134 \expandafter\let
```

```
8135 \expandafter\@gls@list\csname @gls@hypergroup@list@#1\endcsname
```

Iterate through list and terminate loop if this group is found.

```
8136 \@for\@gls@elem:=\@gls@list\do{%
```

```
8137 \ifthenelse{\equal{\@gls@elem}{#2}}{\@endfortrue}{}}%
```

Check if list terminated prematurely.

```
8138 \if@endfor
```

```
8139 \else
```

This group was not included in the list, so issue a warning.

```
8140 \GlossariesWarningNoLine{Navigation panel
```

```
8141 for glossary type ‘#1’~Jmissing group ‘#2’}%
```

```
8142 \gdef\gls@hypergroup@rerun{%
```

```
8143 \GlossariesWarningNoLine{Navigation panel
```

```
8144 has changed. Rerun LaTeX}}%
```

```
8145 \fi
```

```
8146 }
```

`\hypergroup@rerun` Give a warning at the end if re-run required

```
8147 \let\gls@hypergroup@rerun\relax
```

```
8148 \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.

```
8149 \newcommand*{\@gls@hypergroup}[2]{%
```

```
8150 \@ifundefined{\@gls@hypergroup@list@#1}{%
```

```
8151 \expandafter\xdef\csname @gls@hypergroup@list@#1\endcsname{#2}}%
```

```
8152 }{%
```

```
8153 \expandafter\let\expandafter\@gls@tmp
```

```
8154 \csname @gls@hypergroup@list@#1\endcsname
```

```
8155 \expandafter\xdef\csname @gls@hypergroup@list@#1\endcsname{%
```

```
8156 \@gls@tmp,#2}}%
```

```
8157 }%
```

```
8158 }
```

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`

```
8159 \newcommand*{\glsnavigation}{%
8160   \def\@gls@between{}%
8161   \ifcsundef{\@gls@hypergroupplist@\@glo@type}%
8162   {%
8163     \def\@gls@list{}%
8164   }%
8165   {%
8166     \expandafter\let\expandafter\@gls@list
8167       \csname \@gls@hypergroupplist@\@glo@type\endcsname
8168   }%
8169   \@for\@gls@tmp:=\@gls@list\do{%
8170     \@gls@between

8171     \@gls@getgrouptitle{\@gls@tmp}{\@gls@grptitle}%
8172     \glsnavhyperlink{\@gls@tmp}{\@gls@grptitle}%
8173     \let\@gls@between\glshypernavsep
8174   }%
8175 }
```

`\glshypernavsep` Separator for the hyper navigation bar.

```
8176 \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`

```
8177 \newcommand*{\glssymbolnav}{%
8178   \glsnavhyperlink{glssymbols}{\@gls@getgrouptitle{glssymbols}}%
8179   \glshypernavsep
8180   \glsnavhyperlink{glsnumbers}{\@gls@getgrouptitle{glsnumbers}}%
8181   \glshypernavsep
8182 }
```

## 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.

```
8183 \ProvidesPackage{glossary-inline}[2018/04/07 v4.37 (NLCT)]
```

`inline` Define the inline style.

```
8184 \newglossarystyle{inline}{%
    Start of glossary sets up first empty separator between entries. (This is then changed by
    \glossentry)

8185   \renewenvironment{theglossary}%
8186   {%
```

```

8187     \def\gls@inlinesep{}%
8188     \def\gls@inlinesubsep{}%
8189     \def\gls@inlinepostchild{}%
8190 }%
8191 {\glspostinline}%

```

No header:

```

8192 \renewcommand*{\glossaryheader}{}%

```

No group headings (if heading is required, add `\glsinlinedopostchild` to start definition in case heading follows a child entry):

```

8193 \renewcommand*{\glsgroupheading}[1]{}%

```

Just display separator followed by name and description:

```

8194 \renewcommand{\glossentry}[2]{%
8195   \glsinlinedopostchild
8196   \gls@inlinesep
8197   \glsentryitem{##1}%
8198   \glsinlinenameformat{##1}%
8199   \glossentryname{##1}%
8200 }%
8201 \ifglstdescsuppressed{##1}%
8202 {%
8203   \glsinlineemptydescformat
8204   {%
8205     \glossentrysymbol{##1}%
8206   }%
8207   {%
8208     ##2%
8209   }%
8210 }%
8211 {%
8212   \ifglshasdesc{##1}%
8213     {\glsinlinedescformat{\glossentrydesc{##1}}{\glossentrysymbol{##1}}{##2}}%
8214     {\glsinlineemptydescformat{\glossentrysymbol{##1}}{##2}}%
8215   }%
8216   \ifglshaschildren{##1}%
8217   {%
8218     \glsresetsubentrycounter
8219     \glsinlineparentchildseparator
8220     \def\gls@inlinesubsep{}%
8221     \def\gls@inlinepostchild{\glsinlinepostchild}%
8222   }%
8223   {}%
8224   \def\gls@inlinesep{\glsinlineseparator}%
8225 }%

```

Sub-entries display description:

```

8226 \renewcommand{\subglossentry}[3]{%
8227   \gls@inlinesubsep%
8228   \glsinlinesubnameformat{##2}%

```

```

8229      \glossentryname{##2}}}%
8230      \glssubentryitem{##2}%
8231      \glsinlinesubdescformat{\glossentrydesc{##2}}{\glossentrysymbol{##2}}{##3}%
8232      \def\gls@inlinesubsep{\glsinlinesubseparator}%
8233  }%

```

Nothing special between groups:

```

8234  \renewcommand*{\glsgroupskip}{}%
8235 }

```

linedopostchild

```

8236 \newcommand*{\glsinlinedopostchild}{%
8237     \gls@inlinepostchild
8238     \def\gls@inlinepostchild{}}%
8239 }

```

inlineseparator Separator to use between entries.

```

8240 \newcommand*{\glsinlineseparator}{;\space}

```

inlinesubseparator Separator to use between sub-entries.

```

8241 \newcommand*{\glsinlinesubseparator}{,\space}

```

parentchildseparator Separator to use between parent and children.

```

8242 \newcommand*{\glsinlineparentchildseparator}{:\space}

```

inlinepostchild Hook to use between child and next entry

```

8243 \newcommand*{\glsinlinepostchild}{}

```

\glspostinline Terminator for inline glossary.

```

8244 \newcommand*{\glspostinline}{\glspostdescription\space}

```

inlinenameformat Formats the name of the entry (first argument label, second argument name):

```

8245 \newcommand*{\glsinlinenameformat}[2]{\glstarget{#1}{#2}}

```

inlinedescformat Formats the entry's description, symbol and location list:

```

8246 \newcommand*{\glsinlinedescformat}[3]{\space#1}

```

emptydescformat Formats the entry's symbol and location list when the description is empty:

```

8247 \newcommand*{\glsinlineemptydescformat}[2]{}

```

inlinesubnameformat Formats the name of the subentry (first argument label, second argument name):

```

8248 \newcommand*{\glsinlinesubnameformat}[2]{\glstarget{#1}{}}

```

inlinesubdescformat Formats the subentry's description, symbol and location list:

```

8249 \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.

```
8250 \ProvidesPackage{glossary-list}[2018/04/07 v4.37 (NLCT)]
```

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```
8251 \providecommand{\indexspace}{%
8252   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
8253 }
```

`tgrouphaderfmt` Provide a way of adjusting the format of the group headings.

```
8254 \newcommand*{\glslistgrouphaderfmt}[1]{#1}
```

`tnavigationitem` Provide a way of adjusting the format of the navigation header. This puts the navigation line inside the optional argument of `item` to prevent unwanted space occurring at the start, but this can cause a problem if the navigation line is too long. With this command, it makes it easier for the user to customise the style without having to remember to modify `\glossaryheader` after the style has been set.

```
8255 \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.

```
8256 \newglossarystyle{list}{%
```

Use description environment:

```
8257   \renewenvironment{theglossary}%
8258     {\begin{description}}{\end{description}}%
```

No header at the start of the environment:

```
8259   \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8260   \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries start a new item in the list:

```
8261   \renewcommand*{\glossentry}[2]{%
8262     \item[\glsentryitem{##1}%
8263       \glstarget{##1}{\glossentryname{##1}}]
8264     \glossentrydesc{##1}\glspostdescription\space ##2}%
```

Sub-entries continue on the same line:

```
8265   \renewcommand*{\subglossentry}[3]{%
8266     \glssubentryitem{##2}%
```

```

8267 \glstarget{##2}{\strut}\space
8268 \glossentrydesc{##2}\glspostdescription\space ##3.}%

Add vertical space between groups:

8269 \renewcommand*{\glsgroupskip}{\ifglsgroupskip\else\indexspace\fi}%
8270 }

```

**listgroup** The listgroup style is like the list style, but the glossary groups have headings.

```

8271 \newglossarystyle{listgroup}{%
Base it on the list style:
8272 \setglossarystyle{list}%

Each group has a heading:
8273 \renewcommand*{\glsgroupheading}[1]{%
8274 \item[\glslistgroupheaderfmt{\glsgrouptitle{##1}}]}

```

**listhypergroup** The listhypergroup style is like the listgroup style, but has a set of links to the groups at the start of the glossary.

```

8275 \newglossarystyle{listhypergroup}{%
Base it on the list style:
8276 \setglossarystyle{list}%

Add navigation links at the start of the environment.
8277 \renewcommand*{\glossaryheader}{%
8278 \glslistnavigationitem{\glslnavigation}}%

Each group has a heading with a hypertarget:
8279 \renewcommand*{\glsgroupheading}[1]{%
8280 \item[\glslistgroupheaderfmt
8281 {\glslnavhypertarget{##1}{\glsgrouptitle{##1}}]}

```

**altlist** The altlist glossary style is like the list style, but places the description on a new line. Sub-entries follow in separate paragraphs without the sub-entry name. This style does not use the entry's symbol.

```

8282 \newglossarystyle{altlist}{%
Base it on the list style:
8283 \setglossarystyle{list}%

Main (level 0) entries start a new item in the list with a line break after the entry name:
8284 \renewcommand*{\glossentry}[2]{%
8285 \item[\glsentryitem{##1}%
8286 \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.

```

8287 \mbox{}\par\nobreak\@afterheading
8288 \glossentrydesc{##1}\glspostdescription\space ##2}%

```

Sub-entries start a new paragraph:

```
8289 \renewcommand{\subglossentry}[3]{%
8290   \par
8291   \glssubentryitem{##2}%
8292   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space ##3}%
8293 }
```

**altlistgroup** The altlistgroup glossary style is like the altlist style, but the glossary groups have headings.

```
8294 \newglossarystyle{altlistgroup}{%
      Base it on the altlist style:
8295   \setglossarystyle{altlist}%
      Each group has a heading:
8296   \renewcommand*{\glsgroupheading}[1]{%
8297     \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.

```
8298 \newglossarystyle{altlisthypergroup}{%
      Base it on the altlist style:
8299   \setglossarystyle{altlist}%
      Add navigation links at the start of the environment.
8300   \renewcommand*{\glossaryheader}{%
8301     \glslistnavigationitem{\glslnavigation}}%
      Each group has a heading with a hypertarget:
8302   \renewcommand*{\glsgroupheading}[1]{%
8303     \item[\glslistgroupheaderfmt
8304       {\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.

```
8305 \newglossarystyle{listdotted}{%
      Base it on the list style:
8306   \setglossarystyle{list}%
      Each main (level 0) entry starts a new item:
8307   \renewcommand*{\glossentry}[2]{%
8308     \item[]\makebox[\glslistdottedwidth][l]{%
8309       \glssentryitem{##1}%
8310       \glstarget{##1}{\glossentryname{##1}}%
8311       \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##1}}%
```

Sub entries have the same format as main entries:

```

8312 \renewcommand*{\subglossentry}[3]{%
8313   \item[\makebox[\glslistdottedwidth][l]{%
8314     \glssubentryitem{##2}}%
8315   \glstarget{##2}{\glossentryname{##2}}}%
8316   \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut\glossentrydesc{##2}}%
8317 }
```

`listdottedwidth`

```

8318 \newlength\glslistdottedwidth
8319 \setlength{\glslistdottedwidth}{.5\hsize}
```

`sublistdotted` This style is similar to the `glostylelistdotted` style, except that the main entries just have the name displayed.

```
8320 \newglossarystyle{sublistdotted}{%
```

Base it on the `listdotted` style:

```
8321 \setglossarystyle{listdotted}%
```

Main (level 0) entries just display the name:

```

8322 \renewcommand*{\glossentry}[2]{%
8323   \item[\glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}}}%
8324 }
```

### 3.4 Glossary Styles using `longtable` (the `glossary-long` package)

The glossary styles defined in the package used the `longtable` environment in the glossary.

```
8325 \ProvidesPackage{glossary-long}[2018/04/07 v4.37 (NLCT)]
```

Requires the package:

```
8326 \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`.)

```

8327 \@ifundefined{glsdescwidth}{%
8328   \newlength\glsdescwidth
8329   \setlength{\glsdescwidth}{0.6\hsize}
8330 }{}
```

`lspagelistwidth` This is a length that governs the width of the page list column.

```

8331 \@ifundefined{glspagelistwidth}{%
8332   \newlength\glspagelistwidth
8333   \setlength{\glspagelistwidth}{0.1\hsize}
8334 }{}
```

**long** The long glossary style command which uses the longtable environment:

```
8335 \newglossarystyle{long}{%
```

Use longtable with two columns:

```
8336 \renewenvironment{theglossary}{%
8337     {\begin{longtable}\lp{\glstdescwidth}}}%
8338     {\end{longtable}}}%
```

Do nothing at the start of the environment:

```
8339 \renewcommand*{\glossaryheader}{}%
```

No heading between groups:

```
8340 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries displayed in a row:

```
8341 \renewcommand{\glossentry}[2]{%
8342     \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8343     \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline
8344 }%
```

Sub entries displayed on the following row without the name:

```
8345 \renewcommand{\subglossentry}[3]{%
8346     &
8347     \glssubentryitem{##2}%
8348     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
8349     ##3\tabularnewline
8350 }%
```

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>)

```
8351 \ifglsgroupskip
8352     \renewcommand*{\glsgroupskip}{}%
8353 \else
8354     \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
8355 \fi
8356 }
```

**longborder** The longborder style is like the above, but with horizontal and vertical lines:

```
8357 \newglossarystyle{longborder}{%
```

Base it on the glostylelong style:

```
8358 \setglossarystyle{long}%
```

Use longtable with two columns with vertical lines between each column:

```
8359 \renewenvironment{theglossary}{%
8360     \begin{longtable}{|l|p{\glstdescwidth}|}{\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
8361 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8362 }
```

**longheader** The longheader style is like the long style but with a header:

```
8363 \newglossarystyle{longheader}{%
```

Base it on the `glostylelong` style:

```
8364 \setglossarystyle{long}{%
```

Set the table's header:

```
8365 \renewcommand*{\glossaryheader}{%
8366 \bfseries \entryname & \bfseries \descriptionname\tabularnewline\endhead}%
8367 }
```

`longheaderborder` The `longheaderborder` style is like the `long` style but with a header and border:

```
8368 \newglossarystyle{longheaderborder}{%
```

Base it on the `glostylelongborder` style:

```
8369 \setglossarystyle{longborder}{%
```

Set the table's header and add horizontal line to table's foot:

```
8370 \renewcommand*{\glossaryheader}{%
8371 \hline\bfseries \entryname & \bfseries
8372 \descriptionname\tabularnewline\hline
8373 \endhead
8374 \hline\endfoot}%
8375 }
```

`long3col` The `long3col` style is like `long` but with 3 columns

```
8376 \newglossarystyle{long3col}{%
```

Use a `longtable` with 3 columns:

```
8377 \renewenvironment{theglossary}{%
8378 {\begin{longtable}{lp{\glsgdescwidth}p{\glspagelistwidth}}}%
8379 {\end{longtable}}}%
```

No table header:

```
8380 \renewcommand*{\glossaryheader}{}%
```

No headings between groups:

```
8381 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
8382 \renewcommand{\glossentry}[2]{%
8383 \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8384 \glossentrydesc{##1} & ##2\tabularnewline
8385 }%
```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```
8386 \renewcommand{\subglossentry}[3]{%
8387 &
8388 \glssubentryitem{##2}%
8389 \glstarget{##2}{\strut}\glossentrydesc{##2} &
8390 ##3\tabularnewline
8391 }%
```

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>)

```
8392 \ifglsgroupskip
8393 \renewcommand*{\glsgroupskip}{}%
8394 \else
8395 \renewcommand*{\glsgroupskip}{ & & \tabularnewline}%
8396 \fi
8397 }
```

**long3colborder** The long3colborder style is like the long3col style but with a border:

```
8398 \newglossarystyle{long3colborder}{%
    Base it on the glostylelong3col style:
8399 \setglossarystyle{long3col}%
    Use a longtable with 3 columns with vertical lines around them:
8400 \renewenvironment{theglossary}%
8401 {\begin{longtable}{|l|p{\glsdescwidth}|p{\glspagelistwidth}|}%
8402 {\end{longtable}}%
    Place horizontal lines at the head and foot of the table:
8403 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8404 }
```

**long3colheader** The long3colheader style is like long3col but with a header row:

```
8405 \newglossarystyle{long3colheader}{%
    Base it on the glostylelong3col style:
8406 \setglossarystyle{long3col}%
    Set the table's header:
8407 \renewcommand*{\glossaryheader}{%
8408 \bfseries\entryname&\bfseries\descriptionname&
8409 \bfseries\pagelistname\tabularnewline\endhead}%
8410 }
```

**colheaderborder** The long3colheaderborder style is like the above but with a border

```
8411 \newglossarystyle{long3colheaderborder}{%
    Base it on the glostylelong3colborder style:
8412 \setglossarystyle{long3colborder}%
    Set the table's header and add horizontal line at table's foot:
8413 \renewcommand*{\glossaryheader}{%
8414 \hline
8415 \bfseries\entryname&\bfseries\descriptionname&
8416 \bfseries\pagelistname\tabularnewline\hline\endhead
8417 \hline\endfoot}%
8418 }
```

`long4col` The `long4col` style has four columns where the third column contains the value of the associated symbol key.

```
8419 \newglossarystyle{long4col}{%
```

Use a `longtable` with 4 columns:

```
8420 \renewenvironment{theglossary}{%
```

```
8421 {\begin{longtable}{llll}}%
```

```
8422 {\end{longtable}}}%
```

No table header:

```
8423 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8424 \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):

```
8425 \renewcommand{\glossentry}[2]{%
```

```
8426 \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
```

```
8427 \glossentrydesc{##1} &
```

```
8428 \glossentrysymbol{##1} &
```

```
8429 ##2\tabularnewline
```

```
8430 }%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
8431 \renewcommand{\subglossentry}[3]{%
```

```
8432 &
```

```
8433 \glssubentryitem{##2}%
```

```
8434 \glstarget{##2}{\strut}\glossentrydesc{##2} &
```

```
8435 \glossentrysymbol{##2} & ##3\tabularnewline
```

```
8436 }%
```

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>)

```
8437 \ifglsgnogroupskip
```

```
8438 \renewcommand*{\glsgroupskip}{}%
```

```
8439 \else
```

```
8440 \renewcommand*{\glsgroupskip}{ & & & \tabularnewline}%
```

```
8441 \fi
```

```
8442 }
```

`long4colheader` The `long4colheader` style is like `long4col` but with a header row.

```
8443 \newglossarystyle{long4colheader}{%
```

Base it on the `glostylelong4col` style:

```
8444 \setglossarystyle{long4col}%
```

Table has a header:

```
8445 \renewcommand*{\glossaryheader}{%
```

```
8446 \bfseries\entryname&\bfseries\descriptionname&
```

```
8447 \bfseries \symbolname&
```



```

8448     \bfseries\pagelistname\tabularnewline\endhead}%
8449 }

```

**long4colborder** The long4colborder style is like long4col but with a border.

```

8450 \newglossarystyle{long4colborder}{%
    Base it on the glostylelong4col style:
8451     \setglossarystyle{long4col}%
    Use a longtable with 4 columns surrounded by vertical lines:
8452     \renewenvironment{theglossary}%
8453         {\begin{longtable}{|l|l|l|l|}}%
8454         {\end{longtable}}%
    Add horizontal lines to the head and foot of the table:
8455     \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8456 }

```

**colheaderborder** The long4colheaderborder style is like the above but with a border.

```

8457 \newglossarystyle{long4colheaderborder}{%
    Base it on the glostylelong4col style:
8458     \setglossarystyle{long4col}%
    Use a longtable with 4 columns surrounded by vertical lines:
8459     \renewenvironment{theglossary}%
8460         {\begin{longtable}{|l|l|l|l|}}%
8461         {\end{longtable}}%
    Add table header and horizontal line at the table's foot:
8462     \renewcommand*{\glossaryheader}{%
8463         \hline\bfseries\entryname&\bfseries\descriptionname&
8464         \bfseries \symbolname&
8465         \bfseries\pagelistname\tabularnewline\hline\endhead
8466         \hline\endfoot}%
8467 }

```

**altlong4col** The altlong4col style is like the long4col style but can have multiline descriptions and page lists.

```

8468 \newglossarystyle{altlong4col}{%
    Base it on the glostylelong4col style:
8469     \setglossarystyle{long4col}%
    Use a longtable with 4 columns where the second and last columns may have multiple lines
    in each row:
8470     \renewenvironment{theglossary}%
8471         {\begin{longtable}{lp{\glstdescwidth}lp{\glspagelistwidth}}}%
8472         {\end{longtable}}%
8473 }

```

`altlong4colheader` The `altlong4colheader` style is like `altlong4col` but with a header row.

```

8474 \newglossarystyle{altlong4colheader}{%
      Base it on the glostylelong4colheader style:
8475   \setglossarystyle{long4colheader}%
      Use a longtable with 4 columns where the second and last columns may have multiple lines
      in each row:
8476   \renewenvironment{theglossary}%
8477     {\begin{longtable}{\lp{\glsgdescwidth}\lp{\glspagelistwidth}}}%
8478     {\end{longtable}}}%
8479 }
```

`altlong4colborder` The `altlong4colborder` style is like `altlong4col` but with a border.

```

8480 \newglossarystyle{altlong4colborder}{%
      Base it on the glostylelong4colborder style:
8481   \setglossarystyle{long4colborder}%
      Use a longtable with 4 columns where the second and last columns may have multiple lines
      in each row:
8482   \renewenvironment{theglossary}%
8483     {\begin{longtable}{\lllp{\glsgdescwidth}\lllp{\glspagelistwidth}}}%
8484     {\end{longtable}}}%
8485 }
```

`altlong4colheaderborder` The `altlong4colheaderborder` style is like the above but with a header as well as a border.

```

8486 \newglossarystyle{altlong4colheaderborder}{%
      Base it on the glostylelong4colheaderborder style:
8487   \setglossarystyle{long4colheaderborder}%
      Use a longtable with 4 columns where the second and last columns may have multiple lines
      in each row:
8488   \renewenvironment{theglossary}%
8489     {\begin{longtable}{\lllp{\glsgdescwidth}\lllp{\glspagelistwidth}}}%
8490     {\end{longtable}}}%
8491 }
```

### 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>

```

8492 \ProvidesPackage{glossary-longbooktabs}[2018/04/07 v4.37 (NLCT)]
```

Requires `booktabs` package:

```

8493 \RequirePackage{booktabs}
```

and the base packages for long styles:

```
8494 \RequirePackage{glossary-long}
8495 \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.

```
8496 \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.

```
8497 \glspatchLToutput
```

As with the longheader style, use the long style as a base.

```
8498 \setglossarystyle{long}{%
```

Add a header with rules.

```
8499 \renewcommand*{\glossaryheader}{%
8500 \toprule \bfseries \entryname & \bfseries
8501 \descriptionname\tabularnewline\midrule\endhead
8502 \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.

```
8503 \ifglsgnogroupskip
8504 \renewcommand*{\glsgroupskip}{}%
8505 \else
8506 \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
8507 \fi
8508 }
```

**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.

```
8509 \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.

```
8510 \glspatchLToutput
```

Use the long3col style as a base.

```
8511 \setglossarystyle{long3col}{%
```

Add a header with rules.

```
8512 \renewcommand*{\glossaryheader}{%
8513 \toprule \bfseries \entryname &
8514 \bfseries \descriptionname &
8515 \bfseries \pagelistname
8516 \tabularnewline\midrule\endhead
8517 \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.

```

8518 \ifglsnogroupskip
8519 \renewcommand*{\glsgroupskip}{}%
8520 \else
8521 \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
8522 \fi
8523 }

```

ng4col-booktabs The long4col-booktabs style is similar to the long4colheader style but uses the booktabs rules and patches longtable to check for group skip occurring at a page break.

```

8524 \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.

```

8525 \glspatchLToutput

```

Use the long4col style as a base.

```

8526 \setglossarystyle{long4col}%

```

Add a header with rules.

```

8527 \renewcommand*{\glossaryheader}{%
8528 \toprule \bfseries \entryname &
8529 \bfseries \descriptionname &
8530 \bfseries \symbolname &
8531 \bfseries \pagelistname
8532 \tabularnewline\midrule\endhead
8533 \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.

```

8534 \ifglsnogroupskip
8535 \renewcommand*{\glsgroupskip}{}%
8536 \else
8537 \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
8538 \fi
8539 }

```

ng4col-booktabs The altlong4col-booktabs style is similar to the altlong4colheader style but uses the booktabs rules and patches longtable to check for group skip occurring at a page break.

```

8540 \newglossarystyle{altlong4col-booktabs}{%

```

If the style change is scoped, the patch will only have a local effect, which may be useful if it conflicts with other tables in the document.

```

8541 \glspatchLToutput

```

Use the long4col-booktabs style as a base.

```

8542 \setglossarystyle{long4col-booktabs}%

```

Change the column specifications:

```
8543 \renewenvironment{theglossary}%  
8544   {\begin{longtable}{lp{\glstdescwidth}lp{\glspagelistwidth}}}%  
8545   {\end{longtable}}}%  
8546 }
```

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.

```
8547 \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.

```
8548 \glspatchLToutput
```

Use the long-booktabs style as a base.

```
8549 \setglossarystyle{long-booktabs}%
```

Adjust the column specification.

```
8550 \renewenvironment{theglossary}%  
8551   {\begin{longtable}{l>{\raggedright}p{\glstdescwidth}}}%  
8552   {\end{longtable}}}%  
8553 }
```

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.

```
8554 \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.

```
8555 \glspatchLToutput
```

Use the long3col-booktabs style as a base.

```
8556 \setglossarystyle{long3col-booktabs}%
```

Adjust the column specification.

```
8557 \renewenvironment{theglossary}%  
8558   {\begin{longtable}{l>{\raggedright}p{\glstdescwidth}%  
8559     >{\raggedright}p{\glspagelistwidth}}}%  
8560   {\end{longtable}}}%  
8561 }
```

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.

```
8562 \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.

```
8563 \glspatchLToutput
```

Use the altlong4col-booktabs style as a base.

```
8564 \setglossarystyle{altlong4col-booktabs}%
```

Adjust the column specification.

```
8565 \renewenvironment{theglossary}%  
8566 {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}l%  
8567 >{\raggedright}p{\glspagelistwidth}}}%  
8568 {\end{longtable}}%  
8569 }
```

sLTpenaltycheck

```
8570 \newcommand*{\glslTpenaltycheck}{%  
8571 \ifnum\outputpenalty=-50\vskip-\normalbaselineskip\relax\fi  
8572 }
```

enaltygroupskip

```
8573 \newcommand{\glspenaltygroupskip}{%  
8574 \noalign{\penalty-50\vskip\normalbaselineskip}}
```

restoreLToutput Provide a way of restoring \LT@output for the user.

```
8575 \let\@gls@org@LT@output\LT@output  
8576 \newcommand*{\glstoreLToutput}{\let\LT@output\@gls@org@LT@output}
```

This is David's patch, but I've replaced the hard-coded values with \glslTpenaltycheck to make it easier to adjust.

lspatchLToutput

```
8577 \newcommand*{\glspatchLToutput}{%  
8578 \renewcommand*{\LT@output}{%  
8579 \ifnum\outputpenalty < -\@Mi  
8580 \ifnum\outputpenalty > -\LT@end@pen  
8581 \LT@err{floats and marginpars not allowed in a longtable}\@ehc  
8582 \else  
8583 \setbox\z@\vbox{\unvbox\@cclv}%  
8584 \ifdim \ht\LT@lastfoot>\ht\LT@foot  
8585 \dimen@pagegoal  
8586 \advance\dimen@-\ht\LT@lastfoot  
8587 \ifdim\dimen@<\ht\z@  
8588 \setbox\@cclv\vbox{\unvbox\z@\copy\LT@foot\vss}%  
8589 \@makecol  
8590 \@outputpage  
8591 \setbox\z@\vbox{\box\LT@head\glslTpenaltycheck}%  
8592 \fi  
8593 \fi  
8594 \global\@colroom\@colht  
8595 \global\vsizel\@colht  
8596 {\unvbox\z@\box\ifvoid\LT@lastfoot\LT@foot\else\LT@lastfoot\fi}%  
8597 \fi  
8598 \else
```

```

8599 \setbox\@cclv\vbox{\unvbox\@cclv\copy\LT@foot\vss}%
8600 \@makecol
8601 \@outputpage
8602 \global\ysize\@colroom
8603 \copy\LT@head
8604 \glsLTpenaltycheck
8605 \nobreak
8606 \fi
8607 }%
8608 }

```

### 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.

```
8609 \ProvidesPackage{glossary-longragged}[2018/04/07 v4.37 (NLCT)]
```

Requires the package:

```
8610 \RequirePackage{array}
```

Requires the package:

```
8611 \RequirePackage{longtable}
```

`\glsdescwidth` This is a length that governs the width of the description column. This may have already been defined.

```

8612 \@ifundefined{glsdescwidth}{%
8613 \newlength\glsdescwidth
8614 \setlength{\glsdescwidth}{0.6\hsize}
8615 }{}

```

`lspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```

8616 \@ifundefined{glspagelistwidth}{%
8617 \newlength\glspagelistwidth
8618 \setlength{\glspagelistwidth}{0.1\hsize}
8619 }{}

```

`longragged` The longragged glossary style is like the long but uses ragged right formatting for the description column.

```
8620 \newglossarystyle{longragged}{%
```

Use longtable with two columns:

```

8621 \renewenvironment{theglossary}%
8622 {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}}}%
8623 {\end{longtable}}%

```

Do nothing at the start of the environment:

```
8624 \renewcommand*{\glossaryheader}{}%
```

No heading between groups:

```
8625 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries displayed in a row:

```
8626 \renewcommand{\glossentry}[2]{%
8627   \glstarget{##1}{\glstarget{##1}{\glossentryname{##1}} &
8628   \glossentrydesc{##1}\glspostdescription\space ##2%
8629   \tabularnewline
8630 }%
```

Sub entries displayed on the following row without the name:

```
8631 \renewcommand{\subglossentry}[3]{%
8632   &
8633   \glssubentryitem{##2}%
8634   \glstarget{##2}{\strut}\glossentrydesc{##2}%
8635   \glspostdescription\space ##3%
8636   \tabularnewline
8637 }%
```

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>)

```
8638 \ifglsgroupskip
8639 \renewcommand*{\glsgroupskip}{}%
8640 \else
8641 \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
8642 \fi
8643 }
```

`ongraggedborder` The `longraggedborder` style is like the above, but with horizontal and vertical lines:

```
8644 \newglossarystyle{longraggedborder}{%
```

Base it on the `glostylelongragged` style:

```
8645 \setglossarystyle{longragged}%
```

Use `longtable` with two columns with vertical lines between each column:

```
8646 \renewenvironment{theglossary}{%
8647   \begin{longtable}{|l|>{\raggedright}p{\glstdescwidth}|}%
8648   {\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
8649 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8650 }
```

`ongraggedheader` The `longraggedheader` style is like the `longragged` style but with a header:

```
8651 \newglossarystyle{longraggedheader}{%
```

Base it on the `glostylelongragged` style:

```
8652 \setglossarystyle{longragged}%
```

Set the table's header:

```
8653 \renewcommand*{\glossaryheader}{%
8654   \bfseries \entryname & \bfseries \descriptionname
```



```

8655 \tabularnewline\endhead}%
8656 }

```

`longraggedheaderborder` The `longraggedheaderborder` style is like the `longragged` style but with a header and border:

```

8657 \newglossarystyle{longraggedheaderborder}{%
      Base it on the glostylelongraggedborder style:
8658 \setglossarystyle{longraggedborder}%
      Set the table's header and add horizontal line to table's foot:
8659 \renewcommand*{\glossaryheader}{%
8660 \hline\bfseries \entryname & \bfseries \descriptionname
8661 \tabularnewline\hline
8662 \endhead
8663 \hline\endfoot}%
8664 }

```

`longragged3col` The `longragged3col` style is like `longragged` but with 3 columns

```

8665 \newglossarystyle{longragged3col}{%
      Use a longtable with 3 columns:
8666 \renewenvironment{theglossary}%
8667 {\begin{longtable}{l>{\raggedright}p{\glstdescwidth}%
8668 >{\raggedright}p{\glspagelistwidth}}}%
8669 {\end{longtable}}%

```

No table header:

```

8670 \renewcommand*{\glossaryheader}{}%

```

No headings between groups:

```

8671 \renewcommand*{\glsgroupheading}[1]{}%

```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```

8672 \renewcommand{\glossentry}[2]{%
8673 \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8674 \glossentrydesc{##1} & ##2\tabularnewline
8675 }%

```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```

8676 \renewcommand{\subglossentry}[3]{%
8677 &
8678 \glssubentryitem{##2}%
8679 \glstarget{##2}{\strut}\glossentrydesc{##2} &
8680 ##3\tabularnewline
8681 }%

```

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>)

```

8682 \ifglsnogroupskip
8683 \renewcommand*{\glsgroupskip}{}%

```

```

8684 \else
8685 \renewcommand*{\glsgroupskip}{ & & \tabularnewline}%
8686 \fi
8687 }

```

**ragged3colborder** The longragged3colborder style is like the longragged3col style but with a border:

```

8688 \newglossarystyle{longragged3colborder}{%
    Base it on the glostylelongragged3col style:
8689 \setglossarystyle{longragged3col}%
    Use a longtable with 3 columns with vertical lines around them:
8690 \renewenvironment{theglossary}%
8691 {\begin{longtable}{|l|>{\raggedright}p{\glsgdescwidth}|%
8692 >{\raggedright}p{\glspagelistwidth}|}%
8693 {\end{longtable}}%
    Place horizontal lines at the head and foot of the table:
8694 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8695 }

```

**ragged3colheader** The longragged3colheader style is like longragged3col but with a header row:

```

8696 \newglossarystyle{longragged3colheader}{%
    Base it on the glostylelongragged3col style:
8697 \setglossarystyle{longragged3col}%
    Set the table's header:
8698 \renewcommand*{\glossaryheader}{%
8699 \bfseries\entryname&\bfseries\descriptionname&
8700 \bfseries\pagelistname\tabularnewline\endhead}%
8701 }

```

**colheaderborder** The longragged3colheaderborder style is like the above but with a border

```

8702 \newglossarystyle{longragged3colheaderborder}{%
    Base it on the glostylelongragged3colborder style:
8703 \setglossarystyle{longragged3colborder}%
    Set the table's header and add horizontal line at table's foot:
8704 \renewcommand*{\glossaryheader}{%
8705 \hline
8706 \bfseries\entryname&\bfseries\descriptionname&
8707 \bfseries\pagelistname\tabularnewline\hline\endhead
8708 \hline\endfoot}%
8709 }

```

**altlongragged4col** The altlongragged4col style is like the altlong4col style defined in the package, except that ragged right formatting is used for the description and page list columns.

```

8710 \newglossarystyle{altlongragged4col}{%

```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8711 \renewenvironment{theglossary}%
8712   {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}l%
8713     >{\raggedright}p{\glspagelistwidth}}}%
8714   {\end{longtable}}%
```

No table header:

```
8715 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8716 \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):

```
8717 \renewcommand{\glossentry}[2]{%
8718   \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8719   \glossentrydesc{##1} & \glossentrysymbol{##1} &
8720   ##2\tabularnewline
8721 }%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
8722 \renewcommand{\subglossentry}[3]{%
8723   &
8724   \glssubentryitem{##2}%
8725   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8726   \glossentrysymbol{##2} & ##3\tabularnewline
8727 }%
```

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>)

```
8728 \ifglsgroupskip
8729   \renewcommand*{\glsgroupskip}{}%
8730 \else
8731   \renewcommand*{\glsgroupskip}{ & & \tabularnewline}%
8732 \fi
8733 }
```

agged4colheader The altlongragged4colheader style is like altlongragged4col but with a header row.

```
8734 \newglossarystyle{altlongragged4colheader}{%
```

Base it on the glostylealtlongragged4col style:

```
8735 \setglossarystyle{altlongragged4col}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8736 \renewenvironment{theglossary}%
8737   {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}l%
8738     >{\raggedright}p{\glspagelistwidth}}}%
8739   {\end{longtable}}%
```

Table has a header:

```
8740 \renewcommand*{\glossaryheader}{%
8741 \bfseries\entryname&\bfseries\descriptionname&
8742 \bfseries \symbolname&
8743 \bfseries\pagelistname\tabularnewline\endhead}%
8744 }
```

`ragged4colborder` The `altlongragged4colborder` style is like `altlongragged4col` but with a border.

```
8745 \newglossarystyle{altlongragged4colborder}{%
```

Base it on the `glostylealtlongragged4col` style:

```
8746 \setglossarystyle{altlongragged4col}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8747 \renewenvironment{theglossary}%
8748 {\begin{longtable}{|l|>\raggedright}p{\glsgdescwidth}|l|}%
8749 >\raggedright}p{\glspagelistwidth}|}}%
8750 {\end{longtable}}%
```

Add horizontal lines to the head and foot of the table:

```
8751 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8752 }
```

`colheaderborder` The `altlongragged4colheaderborder` style is like the above but with a header as well as a border.

```
8753 \newglossarystyle{altlongragged4colheaderborder}{%
```

Base it on the `glostylealtlongragged4col` style:

```
8754 \setglossarystyle{altlongragged4col}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8755 \renewenvironment{theglossary}%
8756 {\begin{longtable}{|l|>\raggedright}p{\glsgdescwidth}|l|}%
8757 >\raggedright}p{\glspagelistwidth}|}}%
8758 {\end{longtable}}%
```

Add table header and horizontal line at the table's foot:

```
8759 \renewcommand*{\glossaryheader}{%
8760 \hline\bfseries\entryname&\bfseries\descriptionname&
8761 \bfseries \symbolname&
8762 \bfseries\pagelistname\tabularnewline\hline\endhead
8763 \hline\endfoot}%
8764 }
```

### 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.

```
8765 \ProvidesPackage{glossary-mcols}[2018/04/07 v4.37 (NLCT)]
```

Required packages:

```
8766 \RequirePackage{multicol}
8767 \RequirePackage{glossary-tree}
```

`\indexspace` The are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```
8768 \providecommand{\indexspace}{%
8769   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
8770 }
```

`\glsmcols` Define macro in which to store the number of columns. (Defaults to 2.)

```
8771 \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.)

```
8772 \newglossarystyle{mcolindex}{%
8773   \setglossarystyle{index}%
8774   \renewenvironment{theglossary}%
8775     {%
8776       \begin{multicols}{\glsmcols}
8777       \setlength{\parindent}{0pt}%
8778       \setlength{\parskip}{0pt plus 0.3pt}%
8779       \let\item\glstreeitem
8780       \let\subitem\glstreesubitem
8781       \let\subsubitem\glstreesubsubitem
8782     }%
8783     {\end{multicols}}}%
8784 }
```

`mcolindexgroup` As `mcolindex` but has headings:

```
8785 \newglossarystyle{mcolindexgroup}{%
8786   \setglossarystyle{mcolindex}%
8787   \renewcommand*{\glsgroupheading}[1]{%
8788     \item\glstreegroupheaderfmt{\glsgrouptitle{##1}}\indexspace}%
8789 }
```

`indexhypergroup` The `mcolindexhypergroup` style is like the `mcolindexgroup` style but has hyper navigation.

```
8790 \newglossarystyle{mcolindexhypergroup}{%
```

Base it on the `glostylemcolindex` style:

```
8791 \setglossarystyle{mcolindex}{%
```

Put navigation links to the groups at the start of the glossary:

```
8792 \renewcommand*{\glossaryheader}{%
8793   \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.

```
8794 \renewcommand*{\glsgroupheading}[1]{%
8795   \item\glstreegroupheaderfmt
8796     {\glsnavigationhypertarget{##1}{\glsgrouptitle{##1}}}%
8797   \indexspace}%
8798 }
```

**colindexspannav** Similar to **colindexhypergroup**, but puts the navigation line in the optional argument of **multicols**.

```
8799 \newglossarystyle{colindexspannav}{%
8800   \setglossarystyle{index}%
8801   \renewenvironment{theglossary}%
8802     {%
8803       \begin{multicols}{\glsmcols}[\noindent\glstreenavigationfmt{\glsnavigation}]
8804       \setlength{\parindent}{0pt}%
8805       \setlength{\parskip}{0pt plus 0.3pt}%
8806       \let\item\glstreeitem}%
8807   {\end{multicols}}%
```

Add a heading for each group (with a target). The group's title is in bold followed by a vertical gap.

```
8808 \renewcommand*{\glsgroupheading}[1]{%
8809   \item\glstreegroupheaderfmt
8810     {\glsnavigationhypertarget{##1}{\glsgrouptitle{##1}}}%
8811   \indexspace}%
8812 }
```

**mcoltree** Multi-column index style. Same as the tree, but puts the glossary in multiple columns.

```
8813 \newglossarystyle{mcoltree}{%
8814   \setglossarystyle{tree}%
8815   \renewenvironment{theglossary}%
8816     {%
8817       \begin{multicols}{\glsmcols}
8818       \setlength{\parindent}{0pt}%
8819       \setlength{\parskip}{0pt plus 0.3pt}%
8820     }%
8821   {\end{multicols}}%
8822 }
```

**mcoltreegroup** Like the **mcoltree** style but the glossary groups have headings.

```
8823 \newglossarystyle{mcoltreegroup}{%
  Base it on the glostylemcoltree style:
8824   \setglossarystyle{mcoltree}%
```

Each group has a heading (in bold) followed by a vertical gap):

```
8825 \renewcommand{\glsgroupheading}[1]{\par
8826 \noindent\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}\par\indexspace}%
8827 }
```

**ltreehypergroup** The `mcoltreehypergroup` style is like the `treegroup` style, but has a set of links to the groups at the start of the glossary.

```
8828 \newglossarystyle{mcoltreehypergroup}{%
```

Base it on the `glostylemcoltree` style:

```
8829 \setglossarystyle{mcoltree}{%
```

Put navigation links to the groups at the start of the `theglossary` environment:

```
8830 \renewcommand*{\glossaryheader}{%
8831 \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%

```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
8832 \renewcommand*{\glsgroupheading}[1]{%
8833 \par\noindent
8834 \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8835 \indexspace}%
8836 }
```

**mcoltreespannav** Similar to the `mcoltreehypergroup` style but the navigation line is put in the optional argument of the `multicols` environment.

```
8837 \newglossarystyle{mcoltreespannav}{%
8838 \setglossarystyle{tree}%
8839 \renewenvironment{theglossary}%
8840 {%
8841 \begin{multicols}{\glsmcols}\noindent\glstreenavigationfmt{\glsnavigation}]
8842 \setlength{\parindent}{0pt}%
8843 \setlength{\parskip}{0pt plus 0.3pt}%
8844 }%
8845 {\end{multicols}}}%

```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
8846 \renewcommand*{\glsgroupheading}[1]{%
8847 \par\noindent
8848 \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8849 \indexspace}%
8850 }
```

**mcoltreenoname** Multi-column index style. Same as the `treenoname`, but puts the glossary in multiple columns.

```
8851 \newglossarystyle{mcoltreenoname}{%
8852 \setglossarystyle{treenoname}%
8853 \renewenvironment{theglossary}%
8854 {%

```

```

8855     \begin{multicols}{\glsmcols}
8856     \setlength{\parindent}{0pt}%
8857     \setlength{\parskip}{0pt plus 0.3pt}%
8858 }%
8859 {\end{multicols}}%
8860 }

```

**treenonamegroup** Like the `mcoltreenoname` style but the glossary groups have headings.

```

8861 \newglossarystyle{mcoltreenonamegroup}{%
    Base it on the glostylemcoltreenoname style:
8862 \setglossarystyle{mcoltreenoname}%
    Give each group a heading:
8863 \renewcommand{\glsgroupheading}[1]{\par
8864 \noindent\glstreegroupheaderfmt{\glsgrouptitle{##1}}\par\indexspace}%
8865 }

```

**onamehypergroup** The `mcoltreenonamehypergroup` style is like the `mcoltreenonamegroup` style, but has a set of links to the groups at the start of the glossary.

```

8866 \newglossarystyle{mcoltreenonamehypergroup}{%
    Base it on the glostylemcoltreenoname style:
8867 \setglossarystyle{mcoltreenoname}%
    Put navigation links to the groups at the start of the theglossary environment:
8868 \renewcommand*{\glossaryheader}{%
8869 \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
    Each group has a heading (in bold with a target) followed by a vertical gap):
8870 \renewcommand*{\glsgroupheading}[1]{%
8871 \par\noindent
8872 \glstreegroupheaderfmt{\glsnavigationtarget{##1}{\glsgrouptitle{##1}}}\par
8873 \indexspace}%
8874 }

```

**eenonamespannav** Similar to the `mcoltreenonamehypergroup` style but the navigation line is put in the optional argument of the `multicols` environment.

```

8875 \newglossarystyle{mcoltreenonamespannav}{%
8876 \setglossarystyle{treenoname}%
8877 \renewenvironment{theglossary}%
8878 {%
8879 \begin{multicols}{\glsmcols}[\noindent\glstreenavigationfmt{\glsnavigation}]
8880 \setlength{\parindent}{0pt}%
8881 \setlength{\parskip}{0pt plus 0.3pt}%
8882 }%
8883 {\end{multicols}}%
    Each group has a heading (in bold with a target) followed by a vertical gap):
8884 \renewcommand*{\glsgroupheading}[1]{%
8885 \par\noindent

```



```

8886 \glstreegroupheaderfmt{\glsnahypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8887 \indexspace}%
8888 }

```

**mcolalttree** Multi-column index style. Same as the alttree, but puts the glossary in multiple columns.

```

8889 \newglossarystyle{mcolalttree}{%
8890 \setglossarystyle{alttree}%
8891 \renewenvironment{theglossary}%
8892 {%
8893 \begin{multicols}{\glsmcols}
8894 \def\@gls@prevlevel{-1}%
8895 \mbox{}\par
8896 }%
8897 {\par\end{multicols}}}%
8898 }

```

**colalttreegroup** Like the mcolalttree style but the glossary groups have headings.

```

8899 \newglossarystyle{colalttreegroup}{%
    Base it on the glostylemcolalttree style:
9900 \setglossarystyle{mcolalttree}%
    Give each group a heading.
9901 \renewcommand{\glsgroupheading}[1]{\par
9902 \def\@gls@prevlevel{-1}%
9903 \hangindent0pt\relax
9904 \parindent0pt\relax
9905 \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}\par\indexspace}%
9906 }

```

**treehypergroup** The mcolalttreehypergroup style is like the mcolalttreegroup style, but has a set of links to the groups at the start of the glossary.

```

8907 \newglossarystyle{mcolalttreehypergroup}{%
    Base it on the glostylemcolalttree style:
8908 \setglossarystyle{mcolalttree}%
    Put the navigation links in the header
8909 \renewcommand*{\glossaryheader}{%
8910 \par
8911 \def\@gls@prevlevel{-1}%
8912 \hangindent0pt\relax
8913 \parindent0pt\relax
8914 \glstreenavigationfmt{\glsnavigation}\par\indexspace}%
    Put a hypertarget at the start of each group
8915 \renewcommand*{\glsgroupheading}[1]{%
8916 \par
8917 \def\@gls@prevlevel{-1}%
8918 \hangindent0pt\relax

```

```

8919 \parindent0pt\relax
8920 \glstreegroupheaderfmt{\glshnavhypertarget{##1}{\glsggetgrouptitle{##1}}}\par
8921 \indexspace}%
8922 }

```

`lalttreespannav` Similar to the `mcolalttreehypergroup` style but the navigation line is put in the optional argument of the `multicols` environment.

```

8923 \newglossarystyle{mcolalttreespannav}{%
8924 \setglossarystyle{alttree}%
8925 \renewenvironment{theglossary}%
8926 {%
8927 \begin{multicols}{\glsmcols}\noindent\glstreenavigationfmt{\glshnavigation}]
8928 \def\@gls@prevlevel{-1}%
8929 \mbox{}\par
8930 }%
8931 {\par\end{multicols}}}%

```

Put a hypertarget at the start of each group

```

8932 \renewcommand*{\glsgroupheading}[1]{%
8933 \par
8934 \def\@gls@prevlevel{-1}%
8935 \hangindent0pt\relax
8936 \parindent0pt\relax
8937 \glstreegroupheaderfmt{\glshnavhypertarget{##1}{\glsggetgrouptitle{##1}}}\par
8938 \indexspace}%
8939 }

```

### 3.8 Glossary Styles using supertabular environment (glossary-super package)

The glossary styles defined in the package use the `supertabular` environment.

```

8940 \ProvidesPackage{glossary-super}[2018/04/07 v4.37 (NLCT)]

```

Requires the package:

```

8941 \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.

```

8942 \@ifundefined{glsdescwidth}{%
8943 \newlength{glsdescwidth}
8944 \setlength{glsdescwidth}{0.6\hsize}
8945 }{}

```

`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.

```

8946 \@ifundefined{glspagelistwidth}{%
8947 \newlength{glspagelistwidth}
8948 \setlength{glspagelistwidth}{0.1\hsize}

```

8949 }{}

**super** The super glossary style uses the supertabular environment (it uses lengths defined in the package.)

8950 \newglossarystyle{super}{%

Put the glossary in a supertabular environment with two columns and no head or tail:

```
8951 \renewenvironment{theglossary}%  
8952 {\tablehead{}\tabletail{}}%  
8953 \begin{supertabular}{lp{\glsdescwidth}}%  
8954 {\end{supertabular}}%
```

Do nothing at the start of the table:

8955 \renewcommand\*{\glossaryheader}{}%

No group headings:

8956 \renewcommand\*{\glsgroupheading}[1]{}%

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```
8957 \renewcommand{\glossentry}[2]{%  
8958 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &  
8959 \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline  
8960 }%
```

Sub entries put in a row (no name, description and page list in second column):

```
8961 \renewcommand{\subglossentry}[3]{%  
8962 &  
8963 \glssubentryitem{##2}%  
8964 \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space  
8965 ##3\tabularnewline  
8966 }%
```

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>)

```
8967 \ifglsnogroupskip  
8968 \renewcommand*{\glsgroupskip}{}%  
8969 \else  
8970 \renewcommand*{\glsgroupskip}{& \tabularnewline}%  
8971 \fi  
8972 }
```

**superborder** The superborder style is like the above, but with horizontal and vertical lines:

8973 \newglossarystyle{superborder}{%

Base it on the glostylesuper style:

8974 \setglossarystyle{super}%

Put the glossary in a supertabular environment with two columns and a horizontal line in the head and tail:

```
8975 \renewenvironment{theglossary}%  
8976 {\tablehead{\hline}\tabletail{\hline}%
```

```

8977     \begin{supertabular}{|l|p{\glsdescwidth}|}%
8978     {\end{supertabular}}%
8979 }

```

**superheader** The superheader style is like the super style, but with a header:

```

8980 \newglossarystyle{superheader}{%
      Base it on the glostylesuper style:
8981   \setglossarystyle{super}%
      Put the glossary in a supertabular environment with two columns, a header and no tail:
8982 \renewenvironment{theglossary}%
8983   {\tablehead{\bfseries \entryname &
8984     \bfseries\descriptionname\tabularnewline}%
8985     \tabletail{}}%
8986   \begin{supertabular}{lp{\glsdescwidth}}%
8987   {\end{supertabular}}%
8988 }

```

**superheaderborder** The superheaderborder style is like the super style but with a header and border:

```

8989 \newglossarystyle{superheaderborder}{%
      Base it on the glostylesuper style:
8990   \setglossarystyle{super}%
      Put the glossary in a supertabular environment with two columns, a header and horizontal
      lines above and below the table:
8991   \renewenvironment{theglossary}%
8992     {\tablehead{\hline\bfseries \entryname &
8993       \bfseries \descriptionname\tabularnewline\hline}%
8994       \tabletail{\hline}}
8995     \begin{supertabular}{|l|p{\glsdescwidth}|}%
8996     {\end{supertabular}}%
8997 }

```

**super3col** The super3col style is like the super style, but with 3 columns:

```

8998 \newglossarystyle{super3col}{%
      Put the glossary in a supertabular environment with three columns and no head or tail:
8999   \renewenvironment{theglossary}%
9000     {\tablehead{}\tabletail{}}%
9001     \begin{supertabular}{lp{\glsdescwidth}p{\glspagelistwidth}}%
9002     {\end{supertabular}}%
      Do nothing at the start of the table:
9003   \renewcommand*{\glossaryheader}{}%
      No group headings:
9004   \renewcommand*{\glsgroupheading}[1]{}%

```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```

9005 \renewcommand{\glossentry}[2]{%
9006   \glentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
9007   \glossentrydesc{##1} & ##2\tabularnewline
9008 }%

```

Sub entries on a row (no name, description in second column, page list in last column):

```

9009 \renewcommand{\subglossentry}[3]{%
9010   &
9011   \glssubentryitem{##2}%
9012   \glstarget{##2}{\strut}\glossentrydesc{##2} &
9013   ##3\tabularnewline
9014 }%

```

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>)

```

9015 \ifglsgroupskip
9016   \renewcommand*{\glsgroupskip}{}%
9017 \else
9018   \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
9019 \fi
9020 }

```

**super3colborder** The super3colborder style is like the super3col style, but with a border:

```

9021 \newglossarystyle{super3colborder}{%

```

Base it on the glostylessuper3col style:

```

9022 \setglossarystyle{super3col}%

```

Put the glossary in a supertabular environment with three columns and a horizontal line in the head and tail:

```

9023 \renewenvironment{theglossary}%
9024   {\tablehead{\hline}\tabletail{\hline}%
9025   \begin{supertabular}{|l|p{\glsgdescwidth}|p{\glspagelistwidth|}}%
9026   {\end{supertabular}}%
9027 }

```

**super3colheader** The super3colheader style is like the super3col style but with a header row:

```

9028 \newglossarystyle{super3colheader}{%

```

Base it on the glostylessuper3col style:

```

9029 \setglossarystyle{super3col}%

```

Put the glossary in a supertabular environment with three columns, a header and no tail:

```

9030 \renewenvironment{theglossary}%
9031   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
9032   \bfseries\pagelistname\tabularnewline}\tabletail{}}%
9033   \begin{supertabular}{lp{\glsgdescwidth}p{\glspagelistwidth}}}%
9034   {\end{supertabular}}%
9035 }

```

colheaderborder The super3colheaderborder style is like the super3col style but with a header and border:

```
9036 \newglossarystyle{super3colheaderborder}{%
```

Base it on the glostylessuper3colborder style:

```
9037 \setglossarystyle{super3colborder}{%
```

Put the glossary in a supertabular environment with three columns, a header with horizontal lines and a horizontal line in the tail:

```
9038 \renewenvironment{theglossary}{%
9039   {\tablehead{\hline
9040     \bfseries\entryname&\bfseries\descriptionname&
9041     \bfseries\pagelistname\tabularnewline\hline}%
9042   \tabletail{\hline}%
9043   \begin{supertabular}{|l|p{\glsgdescwidth}|p{\glspagelistwidth}|}%
9044   {\end{supertabular}}}%
9045 }
```

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.

```
9046 \newglossarystyle{super4col}{%
```

Put the glossary in a supertabular environment with four columns and no head or tail:

```
9047 \renewenvironment{theglossary}{%
9048   {\tablehead{}\tabletail{}}%
9049   \begin{supertabular}{|l|l|l|l|}%
9050   \end{supertabular}}%
```

Do nothing at the start of the table:

```
9051 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9052 \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:

```
9053 \renewcommand{\glossentry}[2]{%
9054   \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
9055   \glossentrydesc{##1} &
9056   \glossentrysymbol{##1} & ##2\tabularnewline
9057   }%
```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```
9058 \renewcommand{\subglossentry}[3]{%
9059   &
9060   \glssubentryitem{##2}%
9061   \glstarget{##2}{\strut}\glossentrydesc{##2} &
9062   \glossentrysymbol{##2} & ##3\tabularnewline
9063   }%
```

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>)

```

9064 \ifglsgroupskip
9065   \renewcommand*{\glsgroupskip}{}%
9066 \else
9067   \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
9068 \fi
9069 }

```

**super4colheader** The super4colheader style is like the super4col but with a header row.

```

9070 \newglossarystyle{super4colheader}{%
  Base it on the glostylesuper4col style:
9071   \setglossarystyle{super4col}%
  Put the glossary in a supertabular environment with four columns, a header and no tail:
9072   \renewenvironment{theglossary}%
9073     {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
9074               \bfseries\symbolname &
9075               \bfseries\pagelistname\tabularnewline}%
9076     \tabletail{}}%
9077     \begin{supertabular}{l111}}}%
9078     {\end{supertabular}}}%
9079 }

```

**super4colborder** The super4colborder style is like the super4col but with a border.

```

9080 \newglossarystyle{super4colborder}{%
  Base it on the glostylesuper4col style:
9081   \setglossarystyle{super4col}%
  Put the glossary in a supertabular environment with four columns and a horizontal line in the
  head and tail:
9082   \renewenvironment{theglossary}%
9083     {\tablehead{\hline}\tabletail{\hline}%
9084     \begin{supertabular}{|l11|11|}}}%
9085     {\end{supertabular}}}%
9086 }

```

**colheaderborder** The super4colheaderborder style is like the super4col but with a header and border.

```

9087 \newglossarystyle{super4colheaderborder}{%
  Base it on the glostylesuper4col style:
9088   \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:
9089   \renewenvironment{theglossary}%
9090     {\tablehead{\hline\bfseries\entryname&\bfseries\descriptionname&
9091               \bfseries\symbolname &

```

```

9092      \bfseries\pagelistname\tabularnewline\hline}%
9093      \tabletail{\hline}%
9094      \begin{supertabular}{|l|l|l|l|}%
9095      {\end{supertabular}}}%
9096 }

```

**altsuper4col** The altsuper4col glossary style is like super4col but has provision for multiline descriptions.

```
9097 \newglossarystyle{altsuper4col}{%
```

Base it on the glostylesuper4col style:

```
9098 \setglossarystyle{super4col}%
```

Put the glossary in a supertabular environment with four columns and no head or tail:

```

9099 \renewenvironment{theglossary}%
9100 {\tablehead{}\tabletail{}}%
9101 \begin{supertabular}{lp{\glsgdescwidth}lp{\glspagelistwidth}}%
9102 {\end{supertabular}}}%
9103 }

```

**super4colheader** The altsuper4colheader style is like the altsuper4col but with a header row.

```
9104 \newglossarystyle{altsuper4colheader}{%
```

Base it on the glostylesuper4colheader style:

```
9105 \setglossarystyle{super4colheader}%
```

Put the glossary in a supertabular environment with four columns, a header and no tail:

```

9106 \renewenvironment{theglossary}%
9107 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
9108 \bfseries\symbolname &
9109 \bfseries\pagelistname\tabularnewline}\tabletail{}}%
9110 \begin{supertabular}{lp{\glsgdescwidth}lp{\glspagelistwidth}}%
9111 {\end{supertabular}}}%
9112 }

```

**super4colborder** The altsuper4colborder style is like the altsuper4col but with a border.

```
9113 \newglossarystyle{altsuper4colborder}{%
```

Base it on the glostylesuper4colborder style:

```
9114 \setglossarystyle{super4colborder}%
```

Put the glossary in a supertabular environment with four columns and a horizontal line in the head and tail:

```

9115 \renewenvironment{theglossary}%
9116 {\tablehead{\hline}\tabletail{\hline}%
9117 \begin{supertabular}%
9118 {lp{\glsgdescwidth}lp{\glspagelistwidth}}%
9119 {\end{supertabular}}}%
9120 }

```

**colheaderborder** The altsuper4colheaderborder style is like the altsuper4col but with a header and border.

```
9121 \newglossarystyle{altsuper4colheaderborder}{%
```



Base it on the `glostylessuper4colheaderborder` style:

```
9122 \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:

```
9123 \renewenvironment{theglossary}%
9124   {\tablehead{\hline
9125     \bfseries\entryname &
9126     \bfseries\descriptionname &
9127     \bfseries\symbolname &
9128     \bfseries\pagelistname\tabularnewline\hline}%
9129   \tabletail{\hline}%
9130   \begin{supertabular}%
9131     {lllp{\glsdescwidth}llp{\glspagelistwidth}}}%
9132   {\end{supertabular}}%
9133 }
```

### 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.

```
9134 \ProvidesPackage{glossary-superragged}[2018/04/07 v4.37 (NLCT)]
```

Requires the package:

```
9135 \RequirePackage{array}
```

Requires the package:

```
9136 \RequirePackage{supertabular}
```

`\glsdescwidth` This is a length that governs the width of the description column. This may already have been defined.

```
9137 \@ifundefined{glsdescwidth}{%
9138   \newlength\glsdescwidth
9139   \setlength{\glsdescwidth}{0.6\hsize}
9140 }{}
```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```
9141 \@ifundefined{glspagelistwidth}{%
9142   \newlength\glspagelistwidth
9143   \setlength{\glspagelistwidth}{0.1\hsize}
9144 }{}
```

`superragged` The `superragged` glossary style uses the `supertabular` environment.

```
9145 \newglossarystyle{superragged}{%
```

Put the glossary in a supertabular environment with two columns and no head or tail:

```
9146 \renewenvironment{theglossary}%
9147   {\tablehead{}\tabletail{}}%
9148   \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}}}%
9149   {\end{supertabular}}%
```

Do nothing at the start of the table:

```
9150 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9151 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```
9152 \renewcommand{\glossentry}[2]{%
9153   \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
9154   \glossentrydesc{##1}\glspostdescription\space ##2%
9155   \tabularnewline
9156 }%
```

Sub entries put in a row (no name, description and page list in second column):

```
9157 \renewcommand{\subglossentry}[3]{%
9158   &
9159   \glssubentryitem{##2}%
9160   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
9161   ##3%
9162   \tabularnewline
9163 }%
```

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>)

```
9164 \ifglsgroupskip
9165   \renewcommand*{\glsgroupskip}{}%
9166 \else
9167   \renewcommand*{\glsgroupskip}{& \tabularnewline}%
9168 \fi
9169 }
```

`\perraggedborder` The `superraggedborder` style is like the above, but with horizontal and vertical lines:

```
9170 \newglossarystyle{superraggedborder}{%
```

Base it on the `glostylesuperragged` style:

```
9171 \setglossarystyle{superragged}%
```

Put the glossary in a supertabular environment with two columns and a horizontal line in the head and tail:

```
9172 \renewenvironment{theglossary}%
9173   {\tablehead{\hline}\tabletail{\hline}%
9174   \begin{supertabular}{l|l>{\raggedright}p{\glsgdescwidth}|}%
9175   {\end{supertabular}}%
9176 }
```

`superraggedheader` The `superraggedheader` style is like the `super` style, but with a header:

```
9177 \newglossarystyle{superraggedheader}{%
```

Base it on the `glostylesuperragged` style:

```
9178 \setglossarystyle{superragged}{%
```

Put the glossary in a `supertabular` environment with two columns, a header and no tail:

```
9179 \renewenvironment{theglossary}{%
```

```
9180 {\tablehead{\bfseries \entryname & \bfseries \descriptionname
```

```
9181 \tabularnewline}%
```

```
9182 \tabletail{}}%
```

```
9183 \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}}%
```

```
9184 {\end{supertabular}}%
```

```
9185 }
```

`superraggedheaderborder` The `superraggedheaderborder` style is like the `superragged` style but with a header and border:

```
9186 \newglossarystyle{superraggedheaderborder}{%
```

Base it on the `glostylesuper` style:

```
9187 \setglossarystyle{superragged}{%
```

Put the glossary in a `supertabular` environment with two columns, a header and horizontal lines above and below the table:

```
9188 \renewenvironment{theglossary}{%
```

```
9189 {\tablehead{\hline\bfseries \entryname &
```

```
9190 \bfseries \descriptionname\tablearnewline\hline}%
```

```
9191 \tabletail{\hline}
```

```
9192 \begin{supertabular}{ll>{\raggedright}p{\glsgdescwidth}}%
```

```
9193 {\end{supertabular}}%
```

```
9194 }
```

`superragged3col` The `superragged3col` style is like the `superragged` style, but with 3 columns:

```
9195 \newglossarystyle{superragged3col}{%
```

Put the glossary in a `supertabular` environment with three columns and no head or tail:

```
9196 \renewenvironment{theglossary}{%
```

```
9197 {\tablehead{}\tabletail{}}%
```

```
9198 \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}%
```

```
9199 >{\raggedright}p{\glspagelistwidth}}%
```

```
9200 {\end{supertabular}}%
```

Do nothing at the start of the table:

```
9201 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9202 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
9203 \renewcommand{\glossentry}[2]{%
```

```
9204 \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
```

```
9205 \glossentrydesc{##1} &
```

```

9206     ##2\tabularnewline
9207 }%

```

Sub entries on a row (no name, description in second column, page list in last column):

```

9208 \renewcommand{\subglossentry}[3]{%
9209     &
9210     \glssubentryitem{##2}%
9211     \glstarget{##2}{\strut}\glossentrydesc{##2} &
9212     ##3\tabularnewline
9213 }%

```

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>)

```

9214 \ifglsnogroupskip
9215 \renewcommand*{\glsgroupskip}{}%
9216 \else
9217 \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
9218 \fi
9219 }

```

agged3colborder The superragged3colborder style is like the superragged3col style, but with a border:

```

9220 \newglossarystyle{superragged3colborder}{%

```

Base it on the glostypesuperragged3col style:

```

9221 \setglossarystyle{superragged3col}%

```

Put the glossary in a supertabular environment with three columns and a horizontal line in the head and tail:

```

9222 \renewenvironment{theglossary}%
9223 {\tablehead{\hline}\tabletail{\hline}%
9224 \begin{supertabular}{|l|>{\raggedright}p{\glsdescwidth}||%
9225 >{\raggedright}p{\glspagelistwidth}||}%
9226 {\end{supertabular}}%
9227 }

```

agged3colheader The superragged3colheader style is like the superragged3col style but with a header row:

```

9228 \newglossarystyle{superragged3colheader}{%

```

Base it on the glostypesuperragged3col style:

```

9229 \setglossarystyle{superragged3col}%

```

Put the glossary in a supertabular environment with three columns, a header and no tail:

```

9230 \renewenvironment{theglossary}%
9231 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
9232 \bfseries\pagelistname\tabularnewline}\tabletail{}}%
9233 \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}%
9234 >{\raggedright}p{\glspagelistwidth}}%
9235 {\end{supertabular}}%
9236 }

```

colheaderborder The superragged3colheaderborder style is like the superragged3col style but with a header and border:

```
9237 \newglossarystyle{superragged3colheaderborder}{%
```

Base it on the glostylesuperragged3colborder style:

```
9238 \setglossarystyle{superragged3colborder}{%
```

Put the glossary in a supertabular environment with three columns, a header with horizontal lines and a horizontal line in the tail:

```
9239 \renewenvironment{theglossary}{%
9240   {\tablehead{\hline
9241     \bfseries\entryname&\bfseries\descriptionname&
9242     \bfseries\pagelistname\tabularnewline\hline}%
9243   \tabletail{\hline}%
9244   \begin{supertabular}{|l|>{\raggedright}p{\glstdescwidth}|%
9245     >{\raggedright}p{\glspagelistwidth}|}%
9246   {\end{supertabular}}}%
9247 }
```

superragged4col The altsuperragged4col glossary style is like altsuper4col style in the package but uses ragged right formatting in the description and page list columns.

```
9248 \newglossarystyle{altsuperragged4col}{%
```

Put the glossary in a supertabular environment with four columns and no head or tail:

```
9249 \renewenvironment{theglossary}{%
9250   {\tablehead{}\tabletail{}%
9251   \begin{supertabular}{l>{\raggedright}p{\glstdescwidth}l%
9252     >{\raggedright}p{\glspagelistwidth}}}%
9253   {\end{supertabular}}}%

```

Do nothing at the start of the table:

```
9254 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9255 \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:

```
9256 \renewcommand{\glossentry}[2]{%
9257   \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
9258   \glossentrydesc{##1} &
9259   \glossentrysymbol{##1} & ##2\tabularnewline
9260 }%
```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```
9261 \renewcommand{\subglossentry}[3]{%
9262   &
9263   \glssubentryitem{##2}%
9264   \glstarget{##2}{\strut}\glossentrydesc{##2} &
9265   \glossentrysymbol{##2} & ##3\tabularnewline
9266 }%
```

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>)

```
9267 \ifglsgroupskip
9268 \renewcommand*{\glsgroupskip}{}%
9269 \else
9270 \renewcommand*{\glsgroupskip}{& & & \tabularnewline}%
9271 \fi
9272 }
```

agged4colheader The altsuperragged4colheader style is like the altsuperragged4col style but with a header row.

```
9273 \newglossarystyle{altsuperragged4colheader}{%
```

Base it on the glostylealtsuperragged4col style:

```
9274 \setglossarystyle{altsuperragged4col}{%
```

Put the glossary in a supertabular environment with four columns, a header and no tail:

```
9275 \renewenvironment{theglossary}%
9276 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
9277 \bfseries\symbolname &
9278 \bfseries\pagelistname\tabularnewline}\tabletail{}}%
9279 \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}l%
9280 >{\raggedright}p{\glspagelistwidth}}}%
9281 {\end{supertabular}}}%
9282 }
```

agged4colborder The altsuperragged4colborder style is like the altsuperragged4col style but with a border.

```
9283 \newglossarystyle{altsuperragged4colborder}{%
```

Base it on the glostylealtsuperragged4col style:

```
9284 \setglossarystyle{altsuper4col}{%
```

Put the glossary in a supertabular environment with four columns and a horizontal line in the head and tail:

```
9285 \renewenvironment{theglossary}%
9286 {\tablehead{\hline}\tabletail{\hline}%
9287 \begin{supertabular}%
9288 {l|>{\raggedright}p{\glsdescwidth}l|}%
9289 >{\raggedright}p{\glspagelistwidth}l|}%
9290 {\end{supertabular}}}%
9291 }
```

colheaderborder The altsuperragged4colheaderborder style is like the altsuperragged4col style but with a header and border.

```
9292 \newglossarystyle{altsuperragged4colheaderborder}{%
```

Base it on the glostylealtsuperragged4col style:

```
9293 \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:

```

9294 \renewenvironment{theglossary}%
9295   {\tablehead{\hline
9296     \bfseries\entryname &
9297     \bfseries\descriptionname &
9298     \bfseries\symbolname &
9299     \bfseries\pagelistname\tabularnewline\hline}%
9300   \tabletail{\hline}%
9301   \begin{supertabular}%
9302     {||>{\raggedright}p{\glsgdescwidth}||}%
9303     >{\raggedright}p{\glspagelistwidth}||}%
9304   {\end{supertabular}}%
9305 }

```

### 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.

```

9306 \ProvidesPackage{glossary-tree}[2018/04/07 v4.37 (NLCT)]

```

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```

9307 \providecommand{\indexspace}{%
9308   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
9309 }

```

`\glstreenamefmt` Format used to display the name in the tree styles. (This may be counteracted by `\glsglfont`.) This command was previously also used to format the group headings.

```

9310 \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.

```

9311 \newcommand*{\glstreegroupheaderfmt}[1]{\glstreenamefmt{#1}}

```

`\glstreenavigationfmt` Format used to display the navigation header in the tree styles.

```

9312 \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.

```

9313 \ifdef\@idxitem
9314 {\newcommand{\glstreeitem}{\@idxitem}}
9315 {\newcommand{\glstreeitem}{\par\hangindent40\p@}}

```

`\glstreesubitem` Level 1 item used in index style.

```
9316 \ifdef\subitem
9317 {\let\glstreesubitem\subitem}
9318 {\newcommand\glstreesubitem{\glstreeitem\hspace*{20\p@}}}
```

`\glstreesubsubitem` Level 1 item used in index style.

```
9319 \ifdef\subsubitem
9320 {\let\glstreesubsubitem\subsubitem}
9321 {\newcommand\glstreesubsubitem{\glstreeitem\hspace*{30\p@}}}
```

`\glstreepredesc` Allow the user to adjust the space before the description (except for the `alttree` style).

```
9322 \newcommand{\glstreepredesc}{\space}
```

`\glstreechildpredesc` Allow the user to adjust the space before the description for sub-entries (except for the `treename` and `alttree` style).

```
9323 \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.

```
9324 \newglossarystyle{index}{%
```

Set the paragraph indentation and skip and define `\item` to be the same as that used by `theindex`:

```
9325 \renewenvironment{theglossary}%
9326 {\setlength{\parindent}{0pt}}%
9327 {\setlength{\parskip}{0pt plus 0.3pt}}%
9328 \let\item\glstreeitem
9329 \let\subitem\glstreesubitem
9330 \let\subsubitem\glstreesubsubitem
9331 }%
```

```
9332 {\par}%
```

Do nothing at the start of the environment:

```
9333 \renewcommand*{\glossaryheader}{}%
```

No group headers:

```
9334 \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.

```
9335 \renewcommand*{\glossentry}[2]{%
9336 \item\glstreeitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
9337 \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
9338 \glstreepredesc \glossentrydesc{##1}\glspostdescription\space ##2%
9339 }%
```



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.

```

9340 \renewcommand{\subglossentry}[3]{%
9341   \ifcase##1\relax
9342     % level 0
9343     \item
9344   \or
9345     % level 1
9346     \subitem
9347     \glssubentryitem{##2}%
9348   \else
9349     % all other levels
9350     \subsubitem
9351   \fi
9352   \glstreenamfmt{\glstarget{##2}{\glossentryname{##2}}}%
9353   \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
9354   \glstreechildpredesc\glossentrydesc{##2}\glspostdescription\space ##3%
9355 }%
```

Vertical gap between groups is the same as that used by indices:

```

9356 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}}
```

**indexgroup** The `indexgroup` style is like the `index` style but has headings.

```

9357 \newglossarystyle{indexgroup}{%
```

Base it on the `glostyleindex` style:

```

9358 \setglossarystyle{index}%
```

Add a heading for each group. This puts the group's title in bold followed by a vertical gap.

```

9359 \renewcommand*{\glsgroupheading}[1]{%
9360   \item\glstreegroupheaderfmt{\glsgrouptitle{##1}}%
9361   \indexspace
9362 }%
9363 }
```

**indexhypergroup** The `indexhypergroup` style is like the `indexgroup` style but has hyper navigation.

```

9364 \newglossarystyle{indexhypergroup}{%
```

Base it on the `glostyleindex` style:

```

9365 \setglossarystyle{index}%
```

Put navigation links to the groups at the start of the glossary:

```

9366 \renewcommand*{\glossaryheader}{%
9367   \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.

```

9368 \renewcommand*{\glsgroupheading}[1]{%
9369   \item\glstreegroupheaderfmt

```

```

9370      {\glshnavhypertarget{##1}{\glshgetgrouptitle{##1}}}%
9371      \indexspace}%
9372 }

```

**tree** The tree glossary style is similar in style to the index style, but can have arbitrary levels.

```

9373 \newglossarystyle{tree}{%

```

Set the paragraph indentation and skip:

```

9374 \renewenvironment{theglossary}%
9375 {\setlength{\parindent}{0pt}%
9376 \setlength{\parskip}{0pt plus 0.3pt}}%
9377 {}%

```

Do nothing at the start of the theglossary environment:

```

9378 \renewcommand*{\glossaryheader}{}%

```

No group headings:

```

9379 \renewcommand*{\glsgroupheading}[1]{}%

```

Main (level 0) entries: name in bold, followed by symbol in brackets (if it exists), the description and the page list:

```

9380 \renewcommand{\glossentry}[2]{%
9381 \hangindent0pt\relax
9382 \parindent0pt\relax
9383 \glshentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
9384 \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
9385 \glstreepredesc\glossentrydesc{##1}\glspostdescription\space##2\par
9386 }%

```

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.

```

9387 \renewcommand{\subglossentry}[3]{%
9388 \hangindent##1\glstreeindent\relax
9389 \parindent##1\glstreeindent\relax
9390 \ifnum##1=1\relax
9391 \glssubentryitem{##2}%
9392 \fi
9393 \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
9394 \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
9395 \glstreechildpredesc\glossentrydesc{##2}\glspostdescription\space ##3\par
9396 }%

```

Vertical gap between groups is the same as that used by indices:

```

9397 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}

```

**treegroup** Like the tree style but the glossary groups have headings.

```

9398 \newglossarystyle{treegroup}{%

```

Base it on the glostyletree style:

```

9399 \setglossarystyle{tree}%

```

Each group has a heading (in bold) followed by a vertical gap):

```
9400 \renewcommand{\glsgroupheading}[1]{\par
9401 \noindent\glstreegroupheaderfmt{\glsgrouptitle{##1}}\par
9402 \indexspace}%
9403 }
```

**treehypergroup** The **treehypergroup** style is like the **treegroup** style, but has a set of links to the groups at the start of the glossary.

```
9404 \newglossarystyle{treehypergroup}{%
```

Base it on the **glostyletree** style:

```
9405 \setglossarystyle{tree}%
```

Put navigation links to the groups at the start of the **theglossary** environment:

```
9406 \renewcommand*{\glossaryheader}{%
9407 \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
9408 \renewcommand*{\glsgroupheading}[1]{%
9409 \par\noindent
9410 \glstreegroupheaderfmt
9411 {\glsnavigationtarget{##1}{\glsgrouptitle{##1}}}\par
9412 \indexspace}%
9413 }
```

**\glstreeindent** Length governing left indent for each level of the tree style.

```
9414 \newlength\glstreeindent
9415 \setlength{\glstreeindent}{10pt}
```

**treenoname** The **treenoname** glossary style is like the **tree** style, but doesn't print the name or symbol for sub-levels.

```
9416 \newglossarystyle{treenoname}{%
```

Set the paragraph indentation and skip:

```
9417 \renewenvironment{theglossary}%
9418 {\setlength{\parindent}{0pt}%
9419 \setlength{\parskip}{0pt plus 0.3pt}}%
9420 {}%
```

No header:

```
9421 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9422 \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.

```
9423 \renewcommand{\glossentry}[2]{%
9424 \hangindent0pt\relax
9425 \parindent0pt\relax
9426 \glstryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
```

```

9427 \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
9428 \glstreepredesc\glossentrydesc{##1}\glspostdescription\space##2\par
9429 }%

```

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.

```

9430 \renewcommand{\subglossentry}[3]{%
9431 \hangindent##1\glstreeindent\relax
9432 \parindent##1\glstreeindent\relax
9433 \ifnum##1=1\relax
9434 \glssubentryitem{##2}%
9435 \fi
9436 \glstarget{##2}{\strut}%
9437 \glossentrydesc{##2}\glspostdescription\space##3\par
9438 }%

```

Vertical gap between groups is the same as that used by indices:

```

9439 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
9440 }

```

`treenonamegroup` Like the `treenoname` style but the glossary groups have headings.

```

9441 \newglossarystyle{treenonamegroup}{%
  Base it on the glostyletreenoname style:
9442 \setglossarystyle{treenoname}%
  Give each group a heading:
9443 \renewcommand{\glsgroupheading}[1]{\par
9444 \noindent\glstreegroupheaderfmt
9445 {\glsgrouptitle{##1}}\par\indexspace}%
9446 }

```

`onamehypergroup` The `treenonamehypergroup` style is like the `treenonamegroup` style, but has a set of links to the groups at the start of the glossary.

```

9447 \newglossarystyle{treenonamehypergroup}{%
  Base it on the glostyletreenoname style:
9448 \setglossarystyle{treenoname}%
  Put navigation links to the groups at the start of the theglossary environment:
9449 \renewcommand*{\glossaryheader}{%
9450 \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
  Each group has a heading (in bold with a target) followed by a vertical gap):
9451 \renewcommand*{\glsgroupheading}[1]{%
9452 \par\noindent
9453 \glstreegroupheaderfmt
9454 {\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}\par
9455 \indexspace}%
9456 }

```

`esttoplevelname` Find the widest name over all parentless entries in the given glossary or glossaries.

```

9457 \newrobustcmd*{\glsfindwidesttoplevelname}[1][\@glo@types]{%
9458   \dimen@=0pt\relax
9459   \gls@tmplen=0pt\relax
9460   \forallglossaries[#1]{\@gls@type}%
9461   {%
9462     \forallglsentries[\@gls@type]{\@glo@label}%
9463     {%
9464       \ifglsahasparent{\@glo@label}%
9465       }%
9466       {%
9467         \settowidth{\dimen@}%
9468           {\glstreenamfmt{\glsentryname{\@glo@label}}}%
9469         \ifdim\dimen@>\gls@tmplen
9470           \gls@tmplen=\dimen@
9471           \letcs{\@glswidestname}{glo\glsdetoklabel{\@glo@label}@name}%
9472         \fi
9473       }%
9474     }%
9475   }%
9476 }
```

`\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.

```

9477 \newcommand*{\glssetwidest}[2][0]{%
9478   \expandafter\def\csname @glswidestname\romannumeral#1\endcsname{%
9479     #2}%
9480 }
```

`\@glswidestname` Initialise `\@glswidestname`.

```

9481 \newcommand*{\@glswidestname}{}
```

`\glstreenamebox` Used by the alttree style to create the box for the name and associated information.

```

9482 \newcommand*{\glstreenamebox}[2]{%
9483   \makebox[#1][l]{#2}%
9484 }
```

`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`.

```

9485 \newglossarystyle{alttree}{%
```

Redefine theglossary environment.

```

9486 \renewenvironment{theglossary}%
9487   {\def\@gls@prevlevel{-1}%
9488    \mbox{}\par}%
9489   {\par}%

```

Set the header and group headers to nothing.

```

9490 \renewcommand*{\glossaryheader}{}%
9491 \renewcommand*{\glsgroupheading}[1]{}%

```

Redefine the way that the level 0 entries are displayed.

```
9492 \renewcommand{\glossentry}[2]{%
9493   \ifnum\@gls@prevlevel=0\relax
9494   \else
```

Find out how big the indentation should be by measuring the widest entry.

```
9495     \settowidth{\glstreeindent}{\glstreenamefmt{\@glswidestname\space}}%
9496   \fi
```

Set the hangindent and paragraph indent.

```
9497   \hangindent\glstreeindent
9498   \parindent\glstreeindent
```

Put the name to the left of the paragraph block.

```
9499   \makebox[0pt][r]{\glstreenamebox{\glstreeindent}{%
9500     \glstryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}}}%
```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
9501   \ifglshassymbol{##1}{(\glossentrysymbol{##1})\space}{}%
```

Do the description followed by the description terminator and location list.

```
9502   \glossentrydesc{##1}\glspostdescription \space ##2\par
```

Set the previous level to 0.

```
9503   \def\@gls@prevlevel{0}%
9504 }%
```

Redefine the way sub-entries are displayed.

```
9505 \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.

```
9506   \ifnum##1=1\relax
9507     \glssubentryitem{##2}%
9508   \fi
```

If the level hasn't changed, keep the same settings, otherwise adjust `\glstreeindent` accordingly.

```
9509   \ifnum\@gls@prevlevel=##1\relax
9510   \else
```

Compute the widest entry for this level, or for level 0 if not defined for this level. Store in `\gls@tmplen`

```
9511     \@ifundefined{@glswidestname\romannumeral##1}{%
9512       \settowidth{\gls@tmplen}{\glstreenamefmt{\@glswidestname\space}}{%
9513       \settowidth{\gls@tmplen}{\glstreenamefmt{%
9514         \csname @glswidestname\romannumeral##1\endcsname\space}}}%
```

Determine if going up or down a level

```
9515   \ifnum\@gls@prevlevel<##1\relax
```

Depth has increased, so add the width of the widest entry to `\glstreeindent`.

```
9516      \setlength\glstreeindent\gls@tmplen
9517      \addtolength\glstreeindent\parindent
9518      \parindent\glstreeindent
9519      \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`.

```
9520      \@ifundefined{glswidestname\romannumeral\@gls@prevlevel}{%
9521      \settowidth{\glstreeindent}{\glstreenamfmt{%
9522      \@glswidestname\space}}}{%
9523      \settowidth{\glstreeindent}{\glstreenamfmt{%
9524      \csname @glswidestname\romannumeral\@gls@prevlevel
9525      \endcsname\space}}}{%
```

Subtract this length from the previous level's paragraph indent and set to `\glstreeindent`.

```
9526      \addtolength\parindent{-\glstreeindent}%
9527      \setlength\glstreeindent\parindent
9528      \fi
9529      \fi
```

Set the hanging indentation.

```
9530      \hangindent\glstreeindent
```

Put the name to the left of the paragraph block

```
9531      \makebox[0pt][r]{\glstreenambox{\gls@tmplen}{%
9532      \glstreenamfmt{\glstarget{##2}{\glossentryname{##2}}}}}%
```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
9533      \ifglshassymbol{##2}{(\glossentrysymbol{##2})\space}{}%
```

Do the description followed by the description terminator and location list.

```
9534      \glossentrydesc{##2}\glspostdescription\space ##3\par
```

Set the previous level macro to the current level.

```
9535      \def\@gls@prevlevel{##1}%
9536      }%
```

Vertical gap between groups is the same as that used by indices:

```
9537      \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
9538 }
```

`almtreegroup` Like the `almtree` style but the glossary groups have headings.

```
9539 \newglossarystyle{almtreegroup}{%
```

Base it on the `glostylealmtree` style:

```
9540 \setglossarystyle{almtree}%
```

Give each group a heading.

```
9541 \renewcommand{\glsgroupheading}[1]{\par
9542 \def\@gls@prevlevel{-1}%
9543 \hangindent0pt\relax
```

```

9544     \parindent0pt\relax
9545     \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}}%
9546     \par\indexspace}%
9547 }

```

alttreehypergroup The alttreehypergroup style is like the alttreegroup style, but has a set of links to the groups at the start of the glossary.

```

9548 \newglossarystyle{alttreehypergroup}{%
    Base it on the glostylealttree style:
9549   \setglossarystyle{alttree}%
    Put the navigation links in the header
9550   \renewcommand*{\glossaryheader}{%
9551     \par
9552     \def\@gls@prevlevel{-1}%
9553     \hangindent0pt\relax
9554     \parindent0pt\relax
9555     \glstreenavigationfmt{\glsnavigation}\par\indexspace}%
    Put a hypertarget at the start of each group
9556   \renewcommand*{\glsgroupheading}[1]{%
9557     \par
9558     \def\@gls@prevlevel{-1}%
9559     \hangindent0pt\relax
9560     \parindent0pt\relax
9561     \glstreegroupheaderfmt
9562     {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
9563     \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.

```
9564 \NeedsTeXFormat{LaTeX2e}
9565 \ProvidesPackage{glossaries-compatible-207}[2018/04/07 v4.37 (NLCT)]
```

**AddXdyAttribute** Adds an attribute in old format.

```
9566 \ifglsxindy
9567   \renewcommand*\GlsAddXdyAttribute[1]{%
9568     \edef\@xdyattributes{\@xdyattributes ^^J \string"#1\string"}%
9569     \expandafter\toks@\expandafter{\@xdylocref}%
9570     \edef\@xdylocref{\the\toks@ ^^J%
9571       (markup-locref
9572       :open \string"\string~n\string\setentrycounter
9573         {\noexpand\glscounter}%
9574         \expandafter\string\csname#1\endcsname
9575         \expandafter\@gobble\string\{\string" ^^J
9576         :close \string"\expandafter\@gobble\string\}\string" ^^J
9577         :attr \string"#1\string")}}}
```

Only has an effect before `\writeist`:

```
9578 \fi
```

**sAddXdyCounters**

```
9579 \renewcommand*\GlsAddXdyCounters[1]{%
9580   \GlossariesWarning{\string\GlsAddXdyCounters\space not available
9581     in compatibility mode.}%
9582 }
```

Add predefined attributes

```
9583 \GlsAddXdyAttribute{glsnumberformat}
9584 \GlsAddXdyAttribute{textrm}
9585 \GlsAddXdyAttribute{textsf}
9586 \GlsAddXdyAttribute{texttt}
9587 \GlsAddXdyAttribute{textbf}
9588 \GlsAddXdyAttribute{textmd}
9589 \GlsAddXdyAttribute{textit}
9590 \GlsAddXdyAttribute{textup}
9591 \GlsAddXdyAttribute{textsl}
```

```

9592 \GlsAddXdyAttribute{textsc}
9593 \GlsAddXdyAttribute{emph}
9594 \GlsAddXdyAttribute{glshypernumber}
9595 \GlsAddXdyAttribute{hyperrm}
9596 \GlsAddXdyAttribute{hypersf}
9597 \GlsAddXdyAttribute{hypertt}
9598 \GlsAddXdyAttribute{hyperbf}
9599 \GlsAddXdyAttribute{hypermd}
9600 \GlsAddXdyAttribute{hyperit}
9601 \GlsAddXdyAttribute{hyperup}
9602 \GlsAddXdyAttribute{hypersl}
9603 \GlsAddXdyAttribute{hypersc}
9604 \GlsAddXdyAttribute{hyperemph}

```

sAddXdyLocation Restore v2.07 definition:

```

9605 \ifglxindy
9606   \renewcommand*{\GlsAddXdyLocation}[2]{%
9607     \edef\@xdyuserlocationdefs{%
9608       \@xdyuserlocationdefs ^^J%
9609       (define-location-class \string"#1\string"^^J\space\space
9610       \space(#2))
9611     }%
9612     \edef\@xdyuserlocationnames{%
9613       \@xdyuserlocationnames^^J\space\space\space
9614       \string"#1\string"}%
9615   }
9616 \fi

```

\@do@wrglossary

```

9617 \renewcommand{\@do@wrglossary}[1]{%
  Determine whether to use xindy or makeindex syntax
9618 \ifglxindy
  Need to determine if the formatting information starts with a ( or ) indicating a range.
9619   \expandafter\@glo@check@mkidxrangechar\@glsnumberformat\@nil
9620   \def\@glo@range{}%
9621   \expandafter\if\@glo@prefix(\relax
9622     \def\@glo@range{:open-range}%
9623   \else
9624     \expandafter\if\@glo@prefix)\relax
9625     \def\@glo@range{:close-range}%
9626   \fi
9627 \fi

  Get the location and escape any special characters
9628   \protected@edef\@glslocref{\theglentrycounter}%
9629   \@gls@checkmkidxchars\@glslocref

  Write to the glossary file using xindy syntax.
9630   \glossary[\csname glo@#1@type\endcsname]{%

```

```

9631 (indexentry :tkey (\csname glo@#1@index\endcsname)
9632   :locoref \string"\@glslocoref\string" %
9633   :attr \string"\@glo@suffix\string" \@glo@range
9634 )
9635 }%
9636 \else
    Convert the format information into the format required for makeindex
9637 \@set@glo@numformat\@glo@numfmt\@gls@counter\@glsnumberformat
    Write to the glossary file using makeindex syntax.
9638 \glossary[\csname glo@#1@type\endcsname]{%
9639 \string\glossaryentry{\csname glo@#1@index\endcsname
9640   \@gls@encapchar\@glo@numfmt}{\theglsentrycounter}}%
9641 \fi
9642 }

```

t@glo@numformat Only had 3 arguments in v2.07

```

9643 \def\@set@glo@numformat#1#2#3{%
9644   \expandafter\@glo@check@mkidxrangechar#3\@nil
9645   \protected@edef#1{%
9646     \@glo@prefix setentrycounter[]{#2}%
9647     \expandafter\string\csname\@glo@suffix\endcsname
9648   }%
9649   \@gls@checkmkidxchars#1%
9650 }

```

\writeist Redefine \writeist back to the way it was in v2.07, but change \istfile to \glswrite.

```

9651 \ifglxindy
9652   \def\writeist{%
9653     \openout\glswrite=\istfilename
9654     \write\glswrite{;; xindy style file created by the glossaries
9655       package in compatible-2.07 mode}%
9656     \write\glswrite{;; for document '\jobname' on
9657       \the\year-\the\month-\the\day}%
9658     \write\glswrite{^^J; required styles^^J}
9659     \@for\@xdystyle:=\@xdyrequiredstyles\do{%
9660       \ifx\@xdystyle\@empty
9661       \else
9662         \protected@write\glswrite{{(require
9663           \string"\@xdystyle.xdy\string")}}%
9664       \fi
9665     }%
9666     \write\glswrite{^^J%
9667       ; list of allowed attributes (number formats)^^J}%
9668     \write\glswrite{(define-attributes ((\@xdyattributes)))}%
9669     \write\glswrite{^^J; user defined alphabets^^J}%
9670     \write\glswrite{\@xdyuseralphabets}%
9671     \write\glswrite{^^J; location class definitions^^J}%
9672     \protected@edef\@gls@roman{\@roman{0}\string"

```

```

9673     \string"roman-numbers-lowercase\string" :sep \string"}}%
9674 \@onelevel@sanitize\@gls@roman
9675 \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
9676     :sep \string"}}%
9677 \@onelevel@sanitize\@tmp
9678 \ifx\@tmp\@gls@roman
9679     \write\glswrite{(define-location-class
9680         \string"roman-page-numbers\string"^^J\space\space\space
9681         (\string"roman-numbers-lowercase\string")
9682         :min-range-length \@glsminrange)}}%
9683 \else
9684     \write\glswrite{(define-location-class
9685         \string"roman-page-numbers\string"^^J\space\space\space
9686         (:sep "\@gls@roman")
9687         :min-range-length \@glsminrange)}}%
9688 \fi
9689 \write\glswrite{(define-location-class
9690     \string"Roman-page-numbers\string"^^J\space\space\space
9691     (\string"roman-numbers-uppercase\string")
9692     :min-range-length \@glsminrange)}}%
9693 \write\glswrite{(define-location-class
9694     \string"arabic-page-numbers\string"^^J\space\space\space
9695     (\string"arabic-numbers\string")
9696     :min-range-length \@glsminrange)}}%
9697 \write\glswrite{(define-location-class
9698     \string"alpha-page-numbers\string"^^J\space\space\space
9699     (\string"alpha\string")
9700     :min-range-length \@glsminrange)}}%
9701 \write\glswrite{(define-location-class
9702     \string"Alpha-page-numbers\string"^^J\space\space\space
9703     (\string"ALPHA\string")
9704     :min-range-length \@glsminrange)}}%
9705 \write\glswrite{(define-location-class
9706     \string"Appendix-page-numbers\string"^^J\space\space\space
9707     (\string"ALPHA\string"
9708     :sep \string"\@glsAlphacompositor\string"
9709     \string"arabic-numbers\string")
9710     :min-range-length \@glsminrange)}}%
9711 \write\glswrite{(define-location-class
9712     \string"arabic-section-numbers\string"^^J\space\space\space
9713     (\string"arabic-numbers\string"
9714     :sep \string"\glscompositor\string"
9715     \string"arabic-numbers\string")
9716     :min-range-length \@glsminrange)}}%
9717 \write\glswrite{^^J; user defined location classes}%
9718 \write\glswrite{\@xdyuserlocationdefs}%
9719 \write\glswrite{^^J; define cross-reference class^^J}%
9720 \write\glswrite{(define-crossref-class \string"see\string"
9721     :unverified )}%

```

```

9722 \write\glswrite{(markup-crossref-list
9723 :class \string"see\string"^^J\space\space\space
9724 :open \string"\string\glssseeformat\string"
9725 :close \string"{}\string")}%
9726 \write\glswrite{^^J; define the order of the location classes}%
9727 \write\glswrite{(define-location-class-order
9728 (\@xdylocationclassorder))}%
9729 \write\glswrite{^^J; define the glossary markup^^J}%
9730 \write\glswrite{(markup-index^^J\space\space\space
9731 :open \string"\string
9732 \glossarysection[\string\glossarytoctitle]{\string
9733 \glossarytitle}\string\glossarypreamble\string~n\string\begin
9734 {theglossary}\string\glossaryheader\string~n\string" ^^J\space
9735 \space\space:close \string"\expandafter\@gobble
9736 \string%\string~n\string
9737 \end{theglossary}\string\glossarypostamble
9738 \string~n\string" ^^J\space\space\space
9739 :tree)}}%
9740 \write\glswrite{(markup-letter-group-list
9741 :sep \string"\string\glsgroupskip\string~n\string")}%
9742 \write\glswrite{(markup-indexentry
9743 :open \string"\string\relax \string\glssresetentrylist
9744 \string~n\string")}%
9745 \write\glswrite{(markup-locclass-list :open
9746 \string"\glsoopenbrace\string\glossaryentrynumbers
9747 \glsoopenbrace\string\relax\space \string"^^J\space\space\space
9748 :sep \string", \string"
9749 :close \string"\glsclosebrace\glsclosebrace\string")}%
9750 \write\glswrite{(markup-locref-list
9751 :sep \string"\string\delimN\space\string")}%
9752 \write\glswrite{(markup-range
9753 :sep \string"\string\delimR\space\string")}%
9754 \@onelevel@sanitize\gls@suffixF
9755 \@onelevel@sanitize\gls@suffixFF
9756 \ifx\gls@suffixF\@empty
9757 \else
9758 \write\glswrite{(markup-range
9759 :close "\gls@suffixF" :length 1 :ignore-end)}%
9760 \fi
9761 \ifx\gls@suffixFF\@empty
9762 \else
9763 \write\glswrite{(markup-range
9764 :close "\gls@suffixFF" :length 2 :ignore-end)}%
9765 \fi
9766 \write\glswrite{^^J; define format to use for locations^^J}%
9767 \write\glswrite{\@xdylocref}%
9768 \write\glswrite{^^J; define letter group list format^^J}%
9769 \write\glswrite{(markup-letter-group-list
9770 :sep \string"\string\glsgroupskip\string~n\string")}%

```

```

9771 \write\glswrite{^^J; letter group headings^^J}%
9772 \write\glswrite{(markup-letter-group
9773   :open-head \string"\string\glsgroupheading
9774   \glsoopenbrace\string"^^J\space\space\space
9775   :close-head \string"\glsclosebrace\string")}%
9776 \write\glswrite{^^J; additional letter groups^^J}%
9777 \write\glswrite{\@xdylettergroups}%
9778 \write\glswrite{^^J; additional sort rules^^J}
9779 \write\glswrite{\@xdysortrules}%
9780 \noist}
9781 \else
9782 \edef\@gls@actualchar{\string?}
9783 \edef\@gls@encapchar{\string|}
9784 \edef\@gls@levelchar{\string!}
9785 \edef\@gls@quotechar{\string"}
9786 \def\writeist{\relax
9787   \openout\glswrite=\istfilename
9788   \write\glswrite{\expandafter\@gobble\string\% makeindex style file
9789     created by the glossaries package}
9790   \write\glswrite{\expandafter\@gobble\string\% for document
9791     '\jobname' on \the\year-\the\month-\the\day}
9792   \write\glswrite{actual '\@gls@actualchar'}
9793   \write\glswrite{encap '\@gls@encapchar'}
9794   \write\glswrite{level '\@gls@levelchar'}
9795   \write\glswrite{quote '\@gls@quotechar'}
9796   \write\glswrite{keyword \string"\string\glossaryentry\string"}
9797   \write\glswrite{preamble \string"\string\glossarysection[\string
9798     \glossarytoctitle]{\string\glossarytitle}\string
9799     \glossarypreamble\string\n\string\begin{theglossary}\string
9800     \glossaryheader\string\n\string"}
9801   \write\glswrite{postamble \string"\string%\string\n\string
9802     \end{theglossary}\string\glossarypostamble\string\n
9803     \string"}
9804   \write\glswrite{group_skip \string"\string\glsgroupskip\string\n
9805     \string"}
9806   \write\glswrite{item_0 \string"\string%\string\n\string"}
9807   \write\glswrite{item_1 \string"\string%\string\n\string"}
9808   \write\glswrite{item_2 \string"\string%\string\n\string"}
9809   \write\glswrite{item_01 \string"\string%\string\n\string"}
9810   \write\glswrite{item_x1
9811     \string"\string\relax \string\glsresetentrylist\string\n
9812     \string"}
9813   \write\glswrite{item_12 \string"\string%\string\n\string"}
9814   \write\glswrite{item_x2
9815     \string"\string\relax \string\glsresetentrylist\string\n
9816     \string"}
9817   \write\glswrite{delim_0 \string"\string\{\string
9818     \glossaryentrynumbers\string\{\string\relax \string"}
9819   \write\glswrite{delim_1 \string"\string\{\string

```

```

9820      \glossaryentrynumbers\string\{\string\relax \string}
9821      \write\glswrite{delim_2 \string"\string\{\string
9822      \glossaryentrynumbers\string\{\string\relax \string}
9823      \write\glswrite{delim_t \string"\string\}\string\}\string}
9824      \write\glswrite{delim_n \string"\string\delimN \string}
9825      \write\glswrite{delim_r \string"\string\delimR \string}
9826      \write\glswrite{headings_flag 1}
9827      \write\glswrite{heading_prefix
9828      \string"\string\glsgroupheading\string\{\string}
9829      \write\glswrite{heading_suffix
9830      \string"\string\}\string\relax
9831      \string\glsgroupresetentrylist \string}
9832      \write\glswrite{symhead_positive \string"glssymbols\string}
9833      \write\glswrite{numhead_positive \string"glslnumbers\string}
9834      \write\glswrite{page_compositor \string"glscpositor\string}
9835      \@gls@escbsdq\gls@suffixF
9836      \@gls@escbsdq\gls@suffixFF
9837      \ifx\gls@suffixF\@empty
9838      \else
9839      \write\glswrite{suffix_2p \string"\gls@suffixF\string}
9840      \fi
9841      \ifx\gls@suffixFF\@empty
9842      \else
9843      \write\glswrite{suffix_3p \string"\gls@suffixFF\string}
9844      \fi
9845      \noist
9846    }
9847  \fi

```

\noist

```

9848 \renewcommand*{\noist}{\let\writeist\relax}

```

## 4.2 glossaries-compatible-307

```

9849 \NeedsTeXFormat{LaTeX2e}
9850 \ProvidesPackage{glossaries-compatible-307}[2018/04/07 v4.37 (NLCT)]

```

Compatibility macros for predefined glossary styles:

`\atglossarystyle` Defines a compatibility glossary style.

```

9851 \newcommand{\compatglossarystyle}[2]{%
9852   \ifcsundef{@glscompstyle@#1}%
9853   {%
9854     \csdef{@glscompstyle@#1}{#2}%
9855   }%
9856   {%
9857     \PackageError{glossaries}{Glossary compatibility style ‘#1’ is already defined}{}%
9858   }%
9859 }

```

Backward compatible inline style.

```

9860 \compatglossarystyle{inline}{%
9861   \renewcommand{\glossaryentryfield}[5]{%
9862     \glsinlinedopostchild
9863     \gls@inlinesep
9864     \def\glo@desc{##3}%
9865     \def\@no@post@desc{\nopostdesc}%
9866     \glsentryitem{##1}\glsinlinenameformat{##1}{##2}%
9867     \ifx\glo@desc\@no@post@desc
9868       \glsinlineemptydescformat{##4}{##5}%
9869     \else
9870       \ifstrepty{##3}%
9871       {\glsinlineemptydescformat{##4}{##5}}%
9872       {\glsinlinedescformat{##3}{##4}{##5}}%
9873     \fi
9874     \ifglshaschildren{##1}%
9875     {%
9876       \glsresetsubentrycounter
9877       \glsinlineparentchildseparator
9878       \def\gls@inlinesubsep{}%
9879       \def\gls@inlinepostchild{\glsinlinepostchild}%
9880     }%
9881   }%
9882   \def\gls@inlinesep{\glsinlineseparator}%
9883 }%
```

Sub-entries display description:

```

9884 \renewcommand{\glossarysubentryfield}[6]{%
9885   \gls@inlinesubsep%
9886   \glsinlinesubnameformat{##2}{##3}%
9887   \glssubentryitem{##2}\glsinlinesubdescformat{##4}{##5}{##6}%
9888   \def\gls@inlinesubsep{\glsinlinesubseparator}%
9889 }%
9890 }
```

Backward compatible list style.

```

9891 \compatglossarystyle{list}{%
9892   \renewcommand*{\glossaryentryfield}[5]{%
9893     \item[\glsentryitem{##1}\glstarget{##1}{##2}]
9894     ##3\glspostdescription\space ##5}%
9895 }
```

Sub-entries continue on the same line:

```

9895 \renewcommand*{\glossarysubentryfield}[6]{%
9896   \glssubentryitem{##2}%
9897   \glstarget{##2}{\strut}##4\glspostdescription\space ##6.}%
9898 }
```

Backward compatible listgroup style.

```

9899 \compatglossarystyle{listgroup}{%
9900   \csuse{@glscompstyle@list}%
9901 }%
```



Backward compatible listhypergroup style.

```
9902 \compatglossarystyle{listhypergroup}{%
9903   \csuse{@glscompstyle@list}%
9904 }%
```

Backward compatible altlist style.

```
9905 \compatglossarystyle{altlist}{%
9906   \renewcommand*{\glossaryentryfield}[5]{%
9907     \item[\glsentryitem{##1}\glstarget{##1}{##2}]%
9908       \mbox{}\par\nobreak\@afterheading
9909       ##3\glspostdescription\space ##5}%
9910   \renewcommand{\glossarysubentryfield}[6]{%
9911     \par
9912     \glssubentryitem{##2}%
9913     \glstarget{##2}{\strut}##4\glspostdescription\space ##6}%
9914 }%
```

Backward compatible altlistgroup style.

```
9915 \compatglossarystyle{altlistgroup}{%
9916   \csuse{@glscompstyle@altlist}%
9917 }%
```

Backward compatible altlisthypergroup style.

```
9918 \compatglossarystyle{altlisthypergroup}{%
9919   \csuse{@glscompstyle@altlist}%
9920 }%
```

Backward compatible listdotted style.

```
9921 \compatglossarystyle{listdotted}{%
9922   \renewcommand*{\glossaryentryfield}[5]{%
9923     \item[\makebox[\glslistdottedwidth][l]{%
9924       \glsentryitem{##1}\glstarget{##1}{##2}%
9925       \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##3}%
9926   \renewcommand*{\glossarysubentryfield}[6]{%
9927     \item[\makebox[\glslistdottedwidth][l]{%
9928       \glssubentryitem{##2}%
9929       \glstarget{##2}{##3}%
9930       \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##4}%
9931 }%
```

Backward compatible sublistdotted style.

```
9932 \compatglossarystyle{sublistdotted}{%
9933   \csuse{@glscompstyle@listdotted}%
9934   \renewcommand*{\glossaryentryfield}[5]{%
9935     \item[\glsentryitem{##1}\glstarget{##1}{##2}]}%
9936 }%
```

Backward compatible long style.

```
9937 \compatglossarystyle{long}{%
9938   \renewcommand*{\glossaryentryfield}[5]{%
9939     \glsentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
9940   \renewcommand*{\glossarysubentryfield}[6]{%
9941     \par
9942     \glssubentryitem{##2}%
9943     \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
9944 }%
```

```

9941      &
9942      \glssubentryitem{##2}%
9943      \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
9944 }%

```

Backward compatible longborder style.

```

9945 \compatglossarystyle{longborder}{%
9946   \csuse{@glscmpstyle@long}%
9947 }%

```

Backward compatible longheader style.

```

9948 \compatglossarystyle{longheader}{%
9949   \csuse{@glscmpstyle@long}%
9950 }%

```

Backward compatible longheaderborder style.

```

9951 \compatglossarystyle{longheaderborder}{%
9952   \csuse{@glscmpstyle@long}%
9953 }%

```

Backward compatible long3col style.

```

9954 \compatglossarystyle{long3col}{%
9955   \renewcommand*{\glossaryentryfield}[5]{%
9956     \glstentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\\}%
9957   \renewcommand*{\glossarysubentryfield}[6]{%
9958     &
9959     \glssubentryitem{##2}%
9960     \glstarget{##2}{\strut}##4 & ##6\\}%
9961 }%

```

Backward compatible long3colborder style.

```

9962 \compatglossarystyle{long3colborder}{%
9963   \csuse{@glscmpstyle@long3col}%
9964 }%

```

Backward compatible long3colheader style.

```

9965 \compatglossarystyle{long3colheader}{%
9966   \csuse{@glscmpstyle@long3col}%
9967 }%

```

Backward compatible long3colheaderborder style.

```

9968 \compatglossarystyle{long3colheaderborder}{%
9969   \csuse{@glscmpstyle@long3col}%
9970 }%

```

Backward compatible long4col style.

```

9971 \compatglossarystyle{long4col}{%
9972   \renewcommand*{\glossaryentryfield}[5]{%
9973     \glstentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%
9974   \renewcommand*{\glossarysubentryfield}[6]{%
9975     &
9976     \glssubentryitem{##2}%

```

```

9977 \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%
9978 }%

Backward compatible long4colheader style.
9979 \compatglossarystyle{long4colheader}{%
9980 \csuse{@glscompstyle@long4col}}%
9981 }%

Backward compatible long4colborder style.
9982 \compatglossarystyle{long4colborder}{%
9983 \csuse{@glscompstyle@long4col}}%
9984 }%

Backward compatible long4colheaderborder style.
9985 \compatglossarystyle{long4colheaderborder}{%
9986 \csuse{@glscompstyle@long4col}}%
9987 }%

Backward compatible altlong4col style.
9988 \compatglossarystyle{altlong4col}{%
9989 \csuse{@glscompstyle@long4col}}%
9990 }%

Backward compatible altlong4colheader style.
9991 \compatglossarystyle{altlong4colheader}{%
9992 \csuse{@glscompstyle@long4col}}%
9993 }%

Backward compatible altlong4colborder style.
9994 \compatglossarystyle{altlong4colborder}{%
9995 \csuse{@glscompstyle@long4col}}%
9996 }%

Backward compatible altlong4colheaderborder style.
9997 \compatglossarystyle{altlong4colheaderborder}{%
9998 \csuse{@glscompstyle@long4col}}%
9999 }%

Backward compatible long style.
10000 \compatglossarystyle{longragged}{%
10001 \renewcommand*{\glossaryentryfield}[5]{%
10002 \glssentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
10003 \tabularnewline}%
10004 \renewcommand*{\glossarysubentryfield}[6]{%
10005 &
10006 \glssubentryitem{##2}%
10007 \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
10008 \tabularnewline}%
10009 }%

Backward compatible longraggedborder style.
10010 \compatglossarystyle{longraggedborder}{%
10011 \csuse{@glscompstyle@longragged}}%
10012 }%

```

Backward compatible longraggedheader style.

```
10013 \compatglossarystyle{longraggedheader}{%  
10014 \csuse{@glscompstyle@longragged}%  
10015 }%
```

Backward compatible longraggedheaderborder style.

```
10016 \compatglossarystyle{longraggedheaderborder}{%  
10017 \csuse{@glscompstyle@longragged}%  
10018 }%
```

Backward compatible longragged3col style.

```
10019 \compatglossarystyle{longragged3col}{%  
10020 \renewcommand*{\glossaryentryfield}[5]{%  
10021 \glstarget{##1}{\glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%  
10022 \renewcommand*{\glossarysubentryfield}[6]{%  
10023 &  
10024 \glssubentryitem{##2}%  
10025 \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%  
10026 }%
```

Backward compatible longragged3colborder style.

```
10027 \compatglossarystyle{longragged3colborder}{%  
10028 \csuse{@glscompstyle@longragged3col}%  
10029 }%
```

Backward compatible longragged3colheader style.

```
10030 \compatglossarystyle{longragged3colheader}{%  
10031 \csuse{@glscompstyle@longragged3col}%  
10032 }%
```

Backward compatible longragged3colheaderborder style.

```
10033 \compatglossarystyle{longragged3colheaderborder}{%  
10034 \csuse{@glscompstyle@longragged3col}%  
10035 }%
```

Backward compatible altlongragged4col style.

```
10036 \compatglossarystyle{altlongragged4col}{%  
10037 \renewcommand*{\glossaryentryfield}[5]{%  
10038 \glstarget{##1}{\glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%  
10039 \renewcommand*{\glossarysubentryfield}[6]{%  
10040 &  
10041 \glssubentryitem{##2}%  
10042 \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%  
10043 }%
```

Backward compatible altlongragged4colheader style.

```
10044 \compatglossarystyle{altlongragged4colheader}{%  
10045 \csuse{@glscompstyle@altlong4col}%  
10046 }%
```

Backward compatible altlongragged4colborder style.

```
10047 \compatglossarystyle{altlongragged4colborder}{%
```

```
10048 \csuse{@glscompstyle@altlong4col}%
10049 }%
```

Backward compatible altlongragged4colheaderborder style.

```
10050 \compatglossarystyle{altlongragged4colheaderborder}{%
10051 \csuse{@glscompstyle@altlong4col}%
10052 }%
```

Backward compatible index style.

```
10053 \compatglossarystyle{index}{%
10054 \renewcommand*{\glossaryentryfield}[5]{%
10055 \item\glsentryitem{##1}\textbf{\glstarget{##1}{##2}}%
10056 \ifx\relax##4\relax
10057 \else
10058 \space{##4}%
10059 \fi
10060 \space ##3\glspostdescription \space ##5}%
10061 \renewcommand*{\glossarysubentryfield}[6]{%
10062 \ifcase##1\relax
10063 % level 0
10064 \item
10065 \or
10066 % level 1
10067 \subitem
10068 \glssubentryitem{##2}%
10069 \else
10070 % all other levels
10071 \subsubitem
10072 \fi
10073 \textbf{\glstarget{##2}{##3}}%
10074 \ifx\relax##5\relax
10075 \else
10076 \space{##5}%
10077 \fi
10078 \space##4\glspostdescription\space ##6}%
10079 }%
```

Backward compatible indexgroup style.

```
10080 \compatglossarystyle{indexgroup}{%
10081 \csuse{@glscompstyle@index}%
10082 }%
```

Backward compatible indexhypergroup style.

```
10083 \compatglossarystyle{indexhypergroup}{%
10084 \csuse{@glscompstyle@index}%
10085 }%
```

Backward compatible tree style.

```
10086 \compatglossarystyle{tree}{%
10087 \renewcommand{\glossaryentryfield}[5]{%
10088 \hangindent0pt\relax
```

```

10089 \parindent0pt\relax
10090 \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
10091 \ifx\relax##4\relax
10092 \else
10093 \space(##4)%
10094 \fi
10095 \space ##3\glspostdescription \space ##5\par}%
10096 \renewcommand{\glossarysubentryfield}[6]{%
10097 \hangindent##1\glstreeindent\relax
10098 \parindent##1\glstreeindent\relax
10099 \ifnum##1=1\relax
10100 \glssubentryitem{##2}%
10101 \fi
10102 \textbf{\glstarget{##2}{##3}}%
10103 \ifx\relax##5\relax
10104 \else
10105 \space(##5)%
10106 \fi
10107 \space##4\glspostdescription\space ##6\par}%
10108 }%

```

Backward compatible treegroup style.

```

10109 \compatglossarystyle{treegroup}{%
10110 \csuse{@glscmpstyle@tree}%
10111 }%

```

Backward compatible treehypergroup style.

```

10112 \compatglossarystyle{treehypergroup}{%
10113 \csuse{@glscmpstyle@tree}%
10114 }%

```

Backward compatible treenoname style.

```

10115 \compatglossarystyle{treenoname}{%
10116 \renewcommand{\glossaryentryfield}[5]{%
10117 \hangindent0pt\relax
10118 \parindent0pt\relax
10119 \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
10120 \ifx\relax##4\relax
10121 \else
10122 \space(##4)%
10123 \fi
10124 \space ##3\glspostdescription \space ##5\par}%
10125 \renewcommand{\glossarysubentryfield}[6]{%
10126 \hangindent##1\glstreeindent\relax
10127 \parindent##1\glstreeindent\relax
10128 \ifnum##1=1\relax
10129 \glssubentryitem{##2}%
10130 \fi
10131 \glstarget{##2}{\strut}%
10132 ##4\glspostdescription\space ##6\par}%
10133 }%

```

Backward compatible treenonamegroup style.

```
10134 \compatglossarystyle{treenonamegroup}{%
10135   \csuse{@glscompstyle@treenoname}%
10136 }%
```

Backward compatible treenonamehypergroup style.

```
10137 \compatglossarystyle{treenonamehypergroup}{%
10138   \csuse{@glscompstyle@treenoname}%
10139 }%
```

Backward compatible alttree style.

```
10140 \compatglossarystyle{alttree}{%
10141   \renewcommand{\glossaryentryfield}[5]{%
10142     \ifnum\@gls@prevlevel=0\relax
10143     \else
10144       \settowidth{\glstreeindent}{\textbf{\@glswidestname\space}}%
10145       \hangindent\glstreeindent
10146       \parindent\glstreeindent
10147     \fi
10148     \makebox[0pt][r]{\makebox[\glstreeindent][l]{%
10149       \glssentryitem{##1}\textbf{\glstarget{##1}{##2}}}%
10150     \ifx\relax##4\relax
10151     \else
10152       (##4)\space
10153     \fi
10154     ##3\glspostdescription \space ##5\par
10155     \def\@gls@prevlevel{0}%
10156   }%
10157   \renewcommand{\glossarysubentryfield}[6]{%
10158     \ifnum##1=1\relax
10159       \glssubentryitem{##2}%
10160     \fi
10161     \ifnum\@gls@prevlevel=##1\relax
10162     \else
10163       \@ifundefined{@glswidestname\romannumeral##1}{%
10164         \settowidth{\gls@tmplen}{\textbf{\@glswidestname\space}}{%
10165         \settowidth{\gls@tmplen}{\textbf{%
10166           \csname @glswidestname\romannumeral##1\endcsname\space}}}%
10167       \ifnum\@gls@prevlevel<##1\relax
10168         \setlength\glstreeindent\gls@tmplen
10169         \addtolength\glstreeindent\parindent
10170         \parindent\glstreeindent
10171       \else
10172         \@ifundefined{@glswidestname\romannumeral\@gls@prevlevel}{%
10173         \settowidth{\glstreeindent}{\textbf{%
10174           \@glswidestname\space}}{%
10175         \settowidth{\glstreeindent}{\textbf{%
10176           \csname @glswidestname\romannumeral\@gls@prevlevel
10177             \endcsname\space}}}%
10178         \addtolength\parindent{-\glstreeindent}%

```

```

10179      \setlength\glstreeindent\parindent
10180      \fi
10181      \fi
10182      \hangindent\glstreeindent
10183      \makebox[0pt][r]{\makebox[\glstemplen][l]{%
10184        \textbf{\glstarget{##2}{##3}}}%
10185      \ifx##5\relax\relax
10186      \else
10187        (##5)\space
10188      \fi
10189      ##4\glspostdescription\space ##6\par
10190      \def\@gls@prevlevel{##1}%
10191    }%
10192 }%

```

Backward compatible alttreegroup style.

```

10193 \compatglossarystyle{alttreegroup}{%
10194   \csuse{@glscompstyle@alttree}%
10195 }%

```

Backward compatible alttreehypergroup style.

```

10196 \compatglossarystyle{alttreehypergroup}{%
10197   \csuse{@glscompstyle@alttree}%
10198 }%

```

Backward compatible mcolindex style.

```

10199 \compatglossarystyle{mcolindex}{%
10200   \csuse{@glscompstyle@index}%
10201 }%

```

Backward compatible mcolindexgroup style.

```

10202 \compatglossarystyle{mcolindexgroup}{%
10203   \csuse{@glscompstyle@index}%
10204 }%

```

Backward compatible mcolindexhypergroup style.

```

10205 \compatglossarystyle{mcolindexhypergroup}{%
10206   \csuse{@glscompstyle@index}%
10207 }%

```

Backward compatible mcoltree style.

```

10208 \compatglossarystyle{mcoltree}{%
10209   \csuse{@glscompstyle@tree}%
10210 }%

```

Backward compatible mcoltreegroup style.

```

10211 \compatglossarystyle{mcolindextreegroup}{%
10212   \csuse{@glscompstyle@tree}%
10213 }%

```

Backward compatible mcoltreehypergroup style.

```

10214 \compatglossarystyle{mcolindextreehypergroup}{%

```



```
10215 \csuse{@glscompstyle@tree}%
10216 }%
```

Backward compatible mcoltreenoname style.

```
10217 \compatglossarystyle{mcoltreenoname}{%
10218 \csuse{@glscompstyle@tree}%
10219 }%
```

Backward compatible mcoltreenonamegroup style.

```
10220 \compatglossarystyle{mcoltreenonamegroup}{%
10221 \csuse{@glscompstyle@tree}%
10222 }%
```

Backward compatible mcoltreenonamehypergroup style.

```
10223 \compatglossarystyle{mcoltreenonamehypergroup}{%
10224 \csuse{@glscompstyle@tree}%
10225 }%
```

Backward compatible mcolalttree style.

```
10226 \compatglossarystyle{mcolalttree}{%
10227 \csuse{@glscompstyle@alttree}%
10228 }%
```

Backward compatible mcolalttreegroup style.

```
10229 \compatglossarystyle{mcolalttreegroup}{%
10230 \csuse{@glscompstyle@alttree}%
10231 }%
```

Backward compatible mcolalttreehypergroup style.

```
10232 \compatglossarystyle{mcolalttreehypergroup}{%
10233 \csuse{@glscompstyle@alttree}%
10234 }%
```

Backward compatible superragged style.

```
10235 \compatglossarystyle{superragged}{%
10236 \renewcommand*{\glossaryentryfield}[5]{%
10237 \glsentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
10238 \tabularnewline}%
10239 \renewcommand*{\glossarysubentryfield}[6]{%
10240 &
10241 \glssubentryitem{##2}%
10242 \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
10243 \tabularnewline}%
10244 }%
```

Backward compatible superraggedborder style.

```
10245 \compatglossarystyle{superraggedborder}{%
10246 \csuse{@glscompstyle@superragged}%
10247 }%
```

Backward compatible superraggedheader style.

```
10248 \compatglossarystyle{superraggedheader}{%
10249 \csuse{@glscompstyle@superragged}%
10250 }%
```

Backward compatible superraggedheaderborder style.

```
10251 \compatglossarystyle{superraggedheaderborder}{%
10252   \csuse{@glscompstyle@superragged}%
10253 }%
```

Backward compatible superragged3col style.

```
10254 \compatglossarystyle{superragged3col}{%
10255   \renewcommand*{\glossaryentryfield}[5]{%
10256     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%
10257   \renewcommand*{\glossarysubentryfield}[6]{%
10258     &
10259     \glssubentryitem{##2}%
10260     \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%
10261 }%
```

Backward compatible superragged3colborder style.

```
10262 \compatglossarystyle{superragged3colborder}{%
10263   \csuse{@glscompstyle@superragged3col}%
10264 }%
```

Backward compatible superragged3colheader style.

```
10265 \compatglossarystyle{superragged3colheader}{%
10266   \csuse{@glscompstyle@superragged3col}%
10267 }%
```

Backward compatible superragged3colheaderborder style.

```
10268 \compatglossarystyle{superragged3colheaderborder}{%
10269   \csuse{@glscompstyle@superragged3col}%
10270 }%
```

Backward compatible altsuperragged4col style.

```
10271 \compatglossarystyle{altsuperragged4col}{%
10272   \renewcommand*{\glossaryentryfield}[5]{%
10273     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%
10274   \renewcommand*{\glossarysubentryfield}[6]{%
10275     &
10276     \glssubentryitem{##2}%
10277     \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%
10278 }%
```

Backward compatible altsuperragged4colheader style.

```
10279 \compatglossarystyle{altsuperragged4colheader}{%
10280   \csuse{@glscompstyle@altsuperragged4col}%
10281 }%
```

Backward compatible altsuperragged4colborder style.

```
10282 \compatglossarystyle{altsuperragged4colborder}{%
10283   \csuse{@glscompstyle@altsuperragged4col}%
10284 }%
```

Backward compatible altsuperragged4colheaderborder style.

```
10285 \compatglossarystyle{altsuperragged4colheaderborder}{%
```

```
10286 \csuse{@glscompstyle@altsuperragged4col}%
10287 }%
```

Backward compatible super style.

```
10288 \compatglossarystyle{super}{%
10289   \renewcommand*{\glossaryentryfield}[5]{%
10290     \glentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
10291   \renewcommand*{\glossarysubentryfield}[6]{%
10292     &
10293     \glssubentryitem{##2}%
10294     \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
10295 }%
```

Backward compatible superborder style.

```
10296 \compatglossarystyle{superborder}{%
10297   \csuse{@glscompstyle@super}%
10298 }%
```

Backward compatible superheader style.

```
10299 \compatglossarystyle{superheader}{%
10300   \csuse{@glscompstyle@super}%
10301 }%
```

Backward compatible superheaderborder style.

```
10302 \compatglossarystyle{superheaderborder}{%
10303   \csuse{@glscompstyle@super}%
10304 }%
```

Backward compatible super3col style.

```
10305 \compatglossarystyle{super3col}{%
10306   \renewcommand*{\glossaryentryfield}[5]{%
10307     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\\}%
10308   \renewcommand*{\glossarysubentryfield}[6]{%
10309     &
10310     \glssubentryitem{##2}%
10311     \glstarget{##2}{\strut}##4 & ##6\\}%
10312 }%
```

Backward compatible super3colborder style.

```
10313 \compatglossarystyle{super3colborder}{%
10314   \csuse{@glscompstyle@super3col}%
10315 }%
```

Backward compatible super3colheader style.

```
10316 \compatglossarystyle{super3colheader}{%
10317   \csuse{@glscompstyle@super3col}%
10318 }%
```

Backward compatible super3colheaderborder style.

```
10319 \compatglossarystyle{super3colheaderborder}{%
10320   \csuse{@glscompstyle@super3col}%
10321 }%
```

Backward compatible super4col style.

```
10322 \compatglossarystyle{super4col}{%
10323   \renewcommand*{\glossaryentryfield}[5]{%
10324     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%
10325   \renewcommand*{\glossarysubentryfield}[6]{%
10326     &
10327     \glssubentryitem{##2}%
10328     \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%
10329 }%
```

Backward compatible super4colheader style.

```
10330 \compatglossarystyle{super4colheader}{%
10331   \csuse{@glscompstyle@super4col}%
10332 }%
```

Backward compatible super4colborder style.

```
10333 \compatglossarystyle{super4colborder}{%
10334   \csuse{@glscompstyle@super4col}%
10335 }%
```

Backward compatible super4colheaderborder style.

```
10336 \compatglossarystyle{super4colheaderborder}{%
10337   \csuse{@glscompstyle@super4col}%
10338 }%
```

Backward compatible altsuper4col style.

```
10339 \compatglossarystyle{altsuper4col}{%
10340   \csuse{@glscompstyle@super4col}%
10341 }%
```

Backward compatible altsuper4colheader style.

```
10342 \compatglossarystyle{altsuper4colheader}{%
10343   \csuse{@glscompstyle@super4col}%
10344 }%
```

Backward compatible altsuper4colborder style.

```
10345 \compatglossarystyle{altsuper4colborder}{%
10346   \csuse{@glscompstyle@super4col}%
10347 }%
```

Backward compatible altsuper4colheaderborder style.

```
10348 \compatglossarystyle{altsuper4colheaderborder}{%
10349   \csuse{@glscompstyle@super4col}%
10350 }%
```

## 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.

```
10351 \NeedsTeXFormat{LaTeX2e}
```

Package version number now in line with main glossaries package number.

```
10352 \ProvidesPackage{glossaries-accsupp}[2018/04/07 v4.37 (NLCT)]
```

```
10353 Experimental glossaries accessibility]
```

Pass all options to glossaries:

```
10354 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
10355 \ProcessOptions
```

This package should be loaded before glossaries-extra, so complain if that has already been loaded.

```
10356 \@ifpackageloaded{glossaries-extra}
```

```
10357 {%
```

If the accsupp option was used, \glsxtr@doaccsupp will have been set, otherwise it will be empty.

```
10358 \ifx\glsxtr@doaccsupp\empty
```

```
10359 \GlossariesWarning{The ‘glossaries-accsupp’
```

```
10360 package has been loaded\MessageBreak
```

```
10361 after the ‘glossaries-extra’ package. This\MessageBreak
```

```
10362 can cause a failure to integrate both packages. \MessageBreak
```

```
10363 Either use the ‘accsupp’ option when you load\MessageBreak
```

```
10364 ‘glossaries-extra’ or load ‘glossaries-accsupp’\MessageBreak
```

```
10365 before loading ‘glossaries-extra’}%
```

```
10366 \fi
```

```
10367 }
```

```
10368 {}
```

tibleglossentry Override style compatibility macros:

```
10369 \def\compatibleglossentry#1#2{%
```

```
10370 \toks@{#2}%
```

```
10371 \protected@edef\do@glossentry{%
```

```
10372 \noexpand\accsuppglossaryentryfield{#1}%
```

```
10373 {\noexpand\glsnamefont
```

```
10374 {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@name\endcsname}}%
```

```

10375     {\expandafter\expandonce\csname glo@glstetoklabel{#1}@desc\endcsname}%
10376     {\expandafter\expandonce\csname glo@glstetoklabel{#1}@symbol\endcsname}%
10377     {\the\toks@}%
10378   }%
10379   \@do@glossentry
10380 }

```

lesubglossentry

```

10381 \def\compatiblesubglossentry#1#2#3{%
10382   \toks@{#3}%
10383   \protected@edef\@do@subglossentry{%
10384     \noexpand\accsuppglossarysubentryfield{\number#1}%
10385     {#2}%
10386     {\noexpand\glsnamefont
10387       {\expandafter\expandonce\csname glo@glstetoklabel{#2}@name\endcsname}}%
10388     {\expandafter\expandonce\csname glo@glstetoklabel{#2}@desc\endcsname}%
10389     {\expandafter\expandonce\csname glo@glstetoklabel{#2}@symbol\endcsname}%
10390     {\the\toks@}%
10391   }%
10392   \@do@subglossentry
10393 }

```

Required packages:

```

10394 \RequirePackage{glossaries}
10395 \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:

```

10396 \define@key{glossentry}{access}{%
10397   \def\@glo@access{#1}%
10398 }

```

**textaccess** The replacement text corresponding to the text key:

```

10399 \define@key{glossentry}{textaccess}{%
10400   \def\@glo@textaccess{#1}%
10401 }

```

**firstaccess** The replacement text corresponding to the first key:

```

10402 \define@key{glossentry}{firstaccess}{%
10403   \def\@glo@firstaccess{#1}%
10404 }

```

pluralaccess The replacement text corresponding to the plural key:

```

10405 \define@key{glossentry}{pluralaccess}{%
10406   \def\@glo@pluralaccess{#1}%
10407 }
```

firstpluralaccess The replacement text corresponding to the firstplural key:

```

10408 \define@key{glossentry}{firstpluralaccess}{%
10409   \def\@glo@firstpluralaccess{#1}%
10410 }
```

symbolaccess The replacement text corresponding to the symbol key:

```

10411 \define@key{glossentry}{symbolaccess}{%
10412   \def\@glo@symbolaccess{#1}%
10413 }
```

symbolpluralaccess The replacement text corresponding to the symbolplural key:

```

10414 \define@key{glossentry}{symbolpluralaccess}{%
10415   \def\@glo@symbolpluralaccess{#1}%
10416 }
```

descriptionaccess The replacement text corresponding to the description key:

```

10417 \define@key{glossentry}{descriptionaccess}{%
10418   \def\@glo@descaccess{#1}%
10419 }
```

descriptionpluralaccess The replacement text corresponding to the descriptionplural key:

```

10420 \define@key{glossentry}{descriptionpluralaccess}{%
10421   \def\@glo@descpluralaccess{#1}%
10422 }
```

shortaccess The replacement text corresponding to the short key:

```

10423 \define@key{glossentry}{shortaccess}{%
10424   \def\@glo@shortaccess{#1}%
10425 }
```

shortpluralaccess The replacement text corresponding to the shortplural key:

```

10426 \define@key{glossentry}{shortpluralaccess}{%
10427   \def\@glo@shortpluralaccess{#1}%
10428 }
```

longaccess The replacement text corresponding to the long key:

```

10429 \define@key{glossentry}{longaccess}{%
10430   \def\@glo@longaccess{#1}%
10431 }
```

longpluralaccess The replacement text corresponding to the longplural key:

```

10432 \define@key{glossentry}{longpluralaccess}{%
10433   \def\@glo@longpluralaccess{#1}%
10434 }
```

There are no equivalent keys for the user1...user6 keys. The replacement text would have to be explicitly put in the value, e.g., user1={\glsacccsupp{inches}{in}}.

Append these new keys to \@gls@keymap:

```
10435 \appto\@gls@keymap{,%
10436   {access}{access},%
10437   {textaccess}{textaccess},%
10438   {firstaccess}{firstaccess},%
10439   {pluralaccess}{pluralaccess},%
10440   {firstpluralaccess}{firstpluralaccess},%
10441   {symbolaccess}{symbolaccess},%
10442   {symbolpluralaccess}{symbolpluralaccess},%
10443   {descaccess}{descaccess},%
10444   {descpluralaccess}{descpluralaccess},%
10445   {shortaccess}{shortaccess},%
10446   {shortpluralaccess}{shortpluralaccess},%
10447   {longaccess}{longaccess},%
10448   {longpluralaccess}{longpluralaccess}}%
10449 }
```

\@gls@noaccess Indicates that no replacement text has been provided.

```
10450 \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):

```
10451 \let\@gls@oldnewglossaryentryprehook\@newglossaryentryprehook
10452 \renewcommand*{\@newglossaryentryprehook}{%
10453   \@gls@oldnewglossaryentryprehook
10454   \def\@glo@access{\@glo@symbol}}%
```

Initialise the other keys:

```
10455 \def\@glo@textaccess{\@glo@access}%
10456 \def\@glo@firstaccess{\@glo@access}%
10457 \def\@glo@pluralaccess{\@glo@textaccess}%
10458 \def\@glo@firstpluralaccess{\@glo@pluralaccess}%
10459 \def\@glo@symbolaccess{\relax}%
10460 \def\@glo@symbolpluralaccess{\@glo@symbolaccess}%
10461 \def\@glo@descaccess{\relax}%
10462 \def\@glo@descpluralaccess{\@glo@descaccess}%
10463 \def\@glo@shortaccess{\relax}%
10464 \def\@glo@shortpluralaccess{\@glo@shortaccess}%
10465 \def\@glo@longaccess{\relax}%
10466 \def\@glo@longpluralaccess{\@glo@longaccess}%
10467 }
```

Add to the end hook:

```
10468 \let\@gls@oldnewglossaryentryposthook\@newglossaryentryposthook
10469 \renewcommand*{\@newglossaryentryposthook}{%
10470   \@gls@oldnewglossaryentryposthook}
```



Store the access information:

```

10471 \expandafter
10472 \protected@xdef\csname glo@\@glo@label @access\endcsname{%
10473 \@glo@access}%
10474 \expandafter
10475 \protected@xdef\csname glo@\@glo@label @textaccess\endcsname{%
10476 \@glo@textaccess}%
10477 \expandafter
10478 \protected@xdef\csname glo@\@glo@label @firstaccess\endcsname{%
10479 \@glo@firstaccess}%
10480 \expandafter
10481 \protected@xdef\csname glo@\@glo@label @pluralaccess\endcsname{%
10482 \@glo@pluralaccess}%
10483 \expandafter
10484 \protected@xdef\csname glo@\@glo@label @firstpluralaccess\endcsname{%
10485 \@glo@firstpluralaccess}%
10486 \expandafter
10487 \protected@xdef\csname glo@\@glo@label @symbolaccess\endcsname{%
10488 \@glo@symbolaccess}%
10489 \expandafter
10490 \protected@xdef\csname glo@\@glo@label @symbolpluralaccess\endcsname{%
10491 \@glo@symbolpluralaccess}%
10492 \expandafter
10493 \protected@xdef\csname glo@\@glo@label @descaccess\endcsname{%
10494 \@glo@descaccess}%
10495 \expandafter
10496 \protected@xdef\csname glo@\@glo@label @descpluralaccess\endcsname{%
10497 \@glo@descpluralaccess}%
10498 \expandafter
10499 \protected@xdef\csname glo@\@glo@label @shortaccess\endcsname{%
10500 \@glo@shortaccess}%
10501 \expandafter
10502 \protected@xdef\csname glo@\@glo@label @shortpluralaccess\endcsname{%
10503 \@glo@shortpluralaccess}%
10504 \expandafter
10505 \protected@xdef\csname glo@\@glo@label @longaccess\endcsname{%
10506 \@glo@longaccess}%
10507 \expandafter
10508 \protected@xdef\csname glo@\@glo@label @longpluralaccess\endcsname{%
10509 \@glo@longpluralaccess}%
10510 }

```

## 5.2 Accessing Replacement Text

`\glsentryaccess` Get the value of the access key for the entry with the given label:

```

10511 \newcommand*{\glsentryaccess}[1]{%
10512 \@gls@entry@field{#1}{access}%
10513 }

```

entrytextaccess Get the value of the textaccess key for the entry with the given label:

```
10514 \newcommand*{\glsentrytextaccess}[1]{%
10515   \@gls@entry@field{#1}{textaccess}%
10516 }
```

entryfirstaccess Get the value of the firstaccess key for the entry with the given label:

```
10517 \newcommand*{\glsentryfirstaccess}[1]{%
10518   \@gls@entry@field{#1}{firstaccess}%
10519 }
```

entrypluralaccess Get the value of the pluralaccess key for the entry with the given label:

```
10520 \newcommand*{\glsentrypluralaccess}[1]{%
10521   \@gls@entry@field{#1}{pluralaccess}%
10522 }
```

entryfirstpluralaccess Get the value of the firstpluralaccess key for the entry with the given label:

```
10523 \newcommand*{\glsentryfirstpluralaccess}[1]{%
10524   \csname glo@#1@firstpluralaccess\endcsname
10525 }
```

entrysymbolaccess Get the value of the symbolaccess key for the entry with the given label:

```
10526 \newcommand*{\glsentrysymbolaccess}[1]{%
10527   \@gls@entry@field{#1}{symbolaccess}%
10528 }
```

entrysymbolpluralaccess Get the value of the symbolpluralaccess key for the entry with the given label:

```
10529 \newcommand*{\glsentrysymbolpluralaccess}[1]{%
10530   \@gls@entry@field{#1}{symbolpluralaccess}%
10531 }
```

entrydescaccess Get the value of the descriptionaccess key for the entry with the given label:

```
10532 \newcommand*{\glsentrydescaccess}[1]{%
10533   \@gls@entry@field{#1}{descaccess}%
10534 }
```

entrydescpluralaccess Get the value of the descriptionpluralaccess key for the entry with the given label:

```
10535 \newcommand*{\glsentrydescpluralaccess}[1]{%
10536   \@gls@entry@field{#1}{descaccess}%
10537 }
```

entryshortaccess Get the value of the shortaccess key for the entry with the given label:

```
10538 \newcommand*{\glsentryshortaccess}[1]{%
10539   \@gls@entry@field{#1}{shortaccess}%
10540 }
```

entryshortpluralaccess Get the value of the shortpluralaccess key for the entry with the given label:

```
10541 \newcommand*{\glsentryshortpluralaccess}[1]{%
10542   \@gls@entry@field{#1}{shortpluralaccess}%
10543 }
```

entrylongaccess Get the value of the longaccess key for the entry with the given label:

```
10544 \newcommand*{\glsentrylongaccess}[1]{%
10545   \@gls@entry@field{#1}{longaccess}%
10546 }
```

ongpluralaccess Get the value of the longpluralaccess key for the entry with the given label:

```
10547 \newcommand*{\glsentrylongpluralaccess}[1]{%
10548   \@gls@entry@field{#1}{longpluralaccess}%
10549 }
```

\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.)

```
10550 \newcommand*{\glsaccsupp}[2]{%
10551   \BeginAccSupp{ActualText=#1}#2\EndAccSupp{}%
10552 }
```

\xglsaccsupp Fully expands replacement text before calling \glsaccsupp

```
10553 \newcommand*{\xglsaccsupp}[2]{%
10554   \protected@edef\@gls@replacementtext{#1}%
10555   \expandafter\glsaccsupp\expandafter{\@gls@replacementtext}{#2}%
10556 }
```

@access@display

```
10557 \newcommand*{\@gls@access@display}[2]{%
10558   \protected@edef\@glo@access{#2}%
10559   \ifx\@glo@access\@gls@noaccess
10560     #1%
10561   \else
10562     \xglsaccsupp{\@glo@access}{#1}%
10563   \fi
10564 }
```

meaccessdisplay Displays the first argument with the accessibility text for the entry with the label given by the second argument (if set).

```
10565 \DeclareRobustCommand*{\glsnameaccessdisplay}[2]{%
10566   \@gls@access@display{#1}{\glsentryaccess{#2}}%
10567 }
```

xtaccessdisplay As above but for the textaccess replacement text.

```
10568 \DeclareRobustCommand*{\glsstextaccessdisplay}[2]{%
10569   \@gls@access@display{#1}{\glsentrytextaccess{#2}}%
10570 }
```

alaccessdisplay As above but for the pluralaccess replacement text.

```
10571 \DeclareRobustCommand*{\glspluralaccessdisplay}[2]{%
10572   \@gls@access@display{#1}{\glsentrypluralaccess{#2}}%
10573 }
```

staccessdisplay As above but for the firstaccess replacement text.

```

10574 \DeclareRobustCommand*\glfirstaccessdisplay}[2]{%
10575   \@gls@access@display{#1}{\glentryfirstaccess{#2}}%
10576 }

```

alaccessdisplay As above but for the firstpluralaccess replacement text.

```

10577 \DeclareRobustCommand*\glfirstpluralaccessdisplay}[2]{%
10578   \@gls@access@display{#1}{\glentryfirstpluralaccess{#2}}%
10579 }

```

olaccessdisplay As above but for the symbolaccess replacement text.

```

10580 \DeclareRobustCommand*\glssymbolaccessdisplay}[2]{%
10581   \@gls@access@display{#1}{\glentrysymbolaccess{#2}}%
10582 }

```

alaccessdisplay As above but for the symbolpluralaccess replacement text.

```

10583 \DeclareRobustCommand*\glssymbolpluralaccessdisplay}[2]{%
10584   \@gls@access@display{#1}{\glentrysymbolpluralaccess{#2}}%
10585 }

```

onaccessdisplay As above but for the descriptionaccess replacement text.

```

10586 \DeclareRobustCommand*\glsdescriptionaccessdisplay}[2]{%
10587   \@gls@access@display{#1}{\glentrydescaccess{#2}}%
10588 }

```

alaccessdisplay As above but for the descriptionpluralaccess replacement text.

```

10589 \DeclareRobustCommand*\glsdescriptionpluralaccessdisplay}[2]{%
10590   \@gls@access@display{#1}{\glentrydescpluralaccess{#2}}%
10591 }

```

rtaccessdisplay As above but for the shortaccess replacement text.

```

10592 \DeclareRobustCommand*\glsshortaccessdisplay}[2]{%
10593   \@gls@access@display{#1}{\glentryshortaccess{#2}}%
10594 }

```

alaccessdisplay As above but for the shortpluralaccess replacement text.

```

10595 \DeclareRobustCommand*\glsshortpluralaccessdisplay}[2]{%
10596   \@gls@access@display{#1}{\glentryshortpluralaccess{#2}}%
10597 }

```

ngaccessdisplay As above but for the longaccess replacement text.

```

10598 \DeclareRobustCommand*\glslongaccessdisplay}[2]{%
10599   \@gls@access@display{#1}{\glentrylongaccess{#2}}%
10600 }

```

alaccessdisplay As above but for the longpluralaccess replacement text.

```

10601 \DeclareRobustCommand*\glslongpluralaccessdisplay}[2]{%
10602   \@gls@access@display{#1}{\glentrylongpluralaccess{#2}}%
10603 }

```

`\glsaccessdisplay` Gets the replacement text corresponding to the named key given by the first argument and calls the appropriate command defined above.

```

10604 \DeclareRobustCommand*\glsaccessdisplay}[3]{%
10605   \@ifundefined{gls#1accessdisplay}%
10606   {%
10607     \PackageError{glossaries-accsupp}{No accessibility support
10608       for key ‘#1’}{}%
10609   }%
10610   {%
10611     \csname gls#1accessdisplay\endcsname{#2}{#3}%
10612   }%
10613 }

```

`\default@entryfmt` Redefine the default entry format to use accessibility information

```

10614 \renewcommand*\@@gls@default@entryfmt}[2]{%
10615   \ifdefempty\glscustomtext
10616   {%
10617     \glsifplural
10618     {%

```

Plural form

```

10619     \glscapscase
10620     {%

```

Don't adjust case

```

10621     \ifglsused\glslabel
10622     {%

```

Subsequent use

```

10623     #2{\glspluralaccessdisplay
10624       {\glsentryplural{\glslabel}}{\glslabel}}%
10625     {\glsdescriptionpluralaccessdisplay
10626       {\glsentrydescplural{\glslabel}}{\glslabel}}%
10627     {\glsymbolpluralaccessdisplay
10628       {\glsentrysymbolplural{\glslabel}}{\glslabel}}
10629     {\glsinsert}%
10630   }%
10631   {%

```

First use

```

10632     #1{\glsfirstpluralaccessdisplay
10633       {\glsentryfirstplural{\glslabel}}{\glslabel}}%
10634     {\glsdescriptionpluralaccessdisplay
10635       {\glsentrydescplural{\glslabel}}{\glslabel}}%
10636     {\glsymbolpluralaccessdisplay
10637       {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10638     {\glsinsert}%
10639   }%
10640   }%
10641   {%

```

Make first letter upper case

```
10642      \ifglsused\glslabel
10643      {%
```

Subsequent use.

```
10644      #2{\glspluralaccessdisplay
10645          {\Glsentryplural{\glslabel}}{\glslabel}}%
10646          {\glsdescriptionpluralaccessdisplay
10647          {\glsentrydescplural{\glslabel}}{\glslabel}}%
10648          {\glssymbolpluralaccessdisplay
10649          {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10650          {\glsinsert}}%
10651      }%
10652      {%
```

First use

```
10653      #1{\glsfirstpluralaccessdisplay
10654          {\Glsentryfirstplural{\glslabel}}{\glslabel}}%
10655          {\glsdescriptionpluralaccessdisplay
10656          {\glsentrydescplural{\glslabel}}{\glslabel}}%
10657          {\glssymbolpluralaccessdisplay
10658          {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10659          {\glsinsert}}%
10660      }%
10661      }%
10662      {%
```

Make all upper case

```
10663      \ifglsused\glslabel
10664      {%
```

Subsequent use

```
10665      \MakeUppercase{%
10666      #2{\glspluralaccessdisplay
10667          {\glsentryplural{\glslabel}}{\glslabel}}%
10668          {\glsdescriptionpluralaccessdisplay
10669          {\glsentrydescplural{\glslabel}}{\glslabel}}%
10670          {\glssymbolpluralaccessdisplay
10671          {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10672          {\glsinsert}}}%
10673      }%
10674      {%
```

First use

```
10675      \MakeUppercase{%
10676      #1{\glsfirstpluralaccessdisplay
10677          {\glsentryfirstplural{\glslabel}}{\glslabel}}%
10678          {\glsdescriptionpluralaccessdisplay
10679          {\glsentrydescplural{\glslabel}}{\glslabel}}%
10680          {\glssymbolpluralaccessdisplay
10681          {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
```

```

10682         {\glsinsert}}}%
10683     }%
10684 }%
10685 }%
10686 {%

```

#### Singular form

```

10687     \glscapscase
10688     {%

```

#### Don't adjust case

```

10689     \ifglsused\glslabel
10690     {%

```

#### Subsequent use

```

10691     #2{\glstextaccessdisplay
10692         {\glsentrytext{\glslabel}}{\glslabel}}%
10693     {\glsdescriptionaccessdisplay
10694         {\glsentrydesc{\glslabel}}{\glslabel}}%
10695     {\glssymbolaccessdisplay
10696         {\glsentrysymbol{\glslabel}}{\glslabel}}%
10697     {\glsinsert}}%
10698 }%
10699 {%

```

#### First use

```

10700     #1{\glsfirstaccessdisplay
10701         {\glsentryfirst{\glslabel}}{\glslabel}}%
10702     {\glsdescriptionaccessdisplay
10703         {\glsentrydesc{\glslabel}}{\glslabel}}%
10704     {\glssymbolaccessdisplay
10705         {\glsentrysymbol{\glslabel}}{\glslabel}}%
10706     {\glsinsert}}%
10707 }%
10708 }%
10709 {%

```

#### Make first letter upper case

```

10710     \ifglsused\glslabel
10711     {%

```

#### Subsequent use

```

10712     #2{\glstextaccessdisplay
10713         {\Glsentrytext{\glslabel}}{\glslabel}}%
10714     {\glsdescriptionaccessdisplay
10715         {\glsentrydesc{\glslabel}}{\glslabel}}%
10716     {\glssymbolaccessdisplay
10717         {\glsentrysymbol{\glslabel}}{\glslabel}}%
10718     {\glsinsert}}%
10719 }%
10720 {%

```

#### First use

```

10721      #1{\glsfirstaccessdisplay
10722          {\Glsentryfirst{\glslabel}}{\glslabel}}%
10723          {\glsdescriptionaccessdisplay
10724              {\glsentrydesc{\glslabel}}{\glslabel}}%
10725          {\glsymbolaccessdisplay
10726              {\glsentrysymbol{\glslabel}}{\glslabel}}%
10727          {\glsinsert}}%
10728      }%
10729  }%
10730  {%

```

#### Make all upper case

```

10731      \ifglsused\glslabel
10732      {%

```

#### Subsequent use

```

10733      \MakeUppercase{%
10734          #2{\glstextaccessdisplay
10735              {\glsentrytext{\glslabel}}{\glslabel}}%
10736              {\glsdescriptionaccessdisplay
10737                  {\glsentrydesc{\glslabel}}{\glslabel}}%
10738                  {\glsymbolaccessdisplay
10739                      {\glsentrysymbol{\glslabel}}{\glslabel}}%
10740                      {\glsinsert}}}%
10741      }%
10742      {%

```

#### First use

```

10743      \MakeUppercase{%
10744          #1{\glsfirstaccessdisplay
10745              {\glsentryfirst{\glslabel}}{\glslabel}}%
10746              {\glsdescriptionaccessdisplay
10747                  {\glsentrydesc{\glslabel}}{\glslabel}}%
10748                  {\glsymbolaccessdisplay
10749                      {\glsentrysymbol{\glslabel}}{\glslabel}}%
10750                      {\glsinsert}}}%
10751      }%
10752  }%
10753  }%
10754  }%
10755  {%

```

#### Custom text provided in \glsdisp

```

10756      \ifglsused{\glslabel}%
10757      {%

```

#### Subsequent use

```

10758      #2{\glscustomtext}%
10759      {\glsdescriptionaccessdisplay
10760          {\glsentrydesc{\glslabel}}{\glslabel}}%

```



```

10761      {\glssymbolaccessdisplay
10762      {\glentrysymbol{\glslabel}}{\glslabel}}%
10763      {\glsinsert}%
10764  }%
10765  {%

```

#### First use

```

10766      #1{\glscustomtext}%
10767      {\glsdescriptionaccessdisplay
10768      {\glentrydesc{\glslabel}}{\glslabel}}%
10769      {\glssymbolaccessdisplay
10770      {\glentrysymbol{\glslabel}}{\glslabel}}%
10771      {\glsinsert}%
10772  }%
10773  }%
10774 }

```

`\glsgenentryfmt`   Redefine to use accessibility information.

```

10775 \renewcommand*{\glsgenentryfmt}{%
10776   \ifdefempty\glscustomtext
10777   {%
10778     \glsifplural
10779     {%

```

#### Plural form

```

10780     \glscapscase
10781     {%

```

#### Don't adjust case

```

10782     \ifglused\glslabel
10783     {%

```

#### Subsequent use

```

10784     \glspluralaccessdisplay
10785     {\glentryplural{\glslabel}}{\glslabel}%
10786     \glsinsert
10787   }%
10788   {%

```

#### First use

```

10789     \glsfirstpluralaccessdisplay
10790     {\glentryfirstplural{\glslabel}}{\glslabel}%
10791     \glsinsert
10792   }%
10793   }%
10794   {%

```

#### Make first letter upper case

```

10795     \ifglused\glslabel
10796     {%

```

Subsequent use.

```
10797      \glspluralaccessdisplay
10798      {\Glsentryplural{\glslabel}}{\glslabel}%
10799      \glsinsert
10800      }%
10801      {%
```

First use

```
10802      \glsfirstpluralaccessdisplay
10803      {\Glsentryfirstplural{\glslabel}}{\glslabel}%
10804      \glsinsert
10805      }%
10806      }%
10807      {%
```

Make all upper case

```
10808      \ifglsused\glslabel
10809      {%
```

Subsequent use

```
10810      \glspluralaccessdisplay
10811      {\mfirstucMakeUppercase{\glsentryplural{\glslabel}}}%
10812      {\glslabel}%
10813      \mfirstucMakeUppercase{\glsinsert}%
10814      }%
10815      {%
```

First use

```
10816      \glsfirstpluralaccessdisplay
10817      {\mfirstucMakeUppercase{\glsentryfirstplural{\glslabel}}}%
10818      {\glslabel}%
10819      \mfirstucMakeUppercase{\glsinsert}%
10820      }%
10821      }%
10822      }%
10823      {%
```

Singular form

```
10824      \glscapscase
10825      {%
```

Don't adjust case

```
10826      \ifglsused\glslabel
10827      {%
```

Subsequent use

```
10828      \glstextaccessdisplay{\glsentrytext{\glslabel}}{\glslabel}%
10829      \glsinsert
10830      }%
10831      {%
```

First use

```
10832      \glsfirstaccessdisplay{\glsentryfirst{\glslabel}}{\glslabel}%
10833      \glsinsert
10834      }%
10835      }%
10836      {%
```

Make first letter upper case

```
10837      \ifglsused\glslabel
10838      {%
```

Subsequent use

```
10839      \glstextaccessdisplay{\Glsentrytext{\glslabel}}{\glslabel}%
10840      \glsinsert
10841      }%
10842      {%
```

First use

```
10843      \glsfirstaccessdisplay{\Glsentryfirst{\glslabel}}{\glslabel}%
10844      \glsinsert
10845      }%
10846      }%
10847      {%
```

Make all upper case

```
10848      \ifglsused\glslabel
10849      {%
```

Subsequent use

```
10850      \glstextaccessdisplay
10851      {\mfirstucMakeUppercase{\glsentrytext{\glslabel}}}{\glslabel}%
10852      \mfirstucMakeUppercase{\glsinsert}%
10853      }%
10854      {%
```

First use

```
10855      \glsfirstaccessdisplay
10856      {\mfirstucMakeUppercase{\glsentryfirst{\glslabel}}}{\glslabel}%
10857      \mfirstucMakeUppercase{\glsinsert}%
10858      }%
10859      }%
10860      }%
10861      }%
10862      {%
```

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.

```
10863      \glscustomtext\glsinsert
10864      }%
10865      }
```

`\glsgenacfmt`   Redefine to include accessibility information.

```
10866 \renewcommand*{\glsgenacfmt}{%
10867   \ifdefempty\glscustomtext
10868     {%
10869       \ifglused\glslabel
10870       {%
```

Subsequent use:

```
10871     \glsifplural
10872     {%
```

Subsequent plural form:

```
10873     \glscapscase
10874     {%
```

Subsequent plural form, don't adjust case:

```
10875     \acronymfont
10876     {\glsshortpluralaccessdisplay
10877      {\glentryshortpl{\glslabel}}{\glslabel}}%
10878     \glsinsert
10879   }%
10880   {%
```

Subsequent plural form, make first letter upper case:

```
10881     \acronymfont
10882     {\glsshortpluralaccessdisplay
10883      {\Glsentryshortpl{\glslabel}}{\glslabel}}%
10884     \glsinsert
10885   }%
10886   {%
```

Subsequent plural form, all caps:

```
10887     \mfirstucMakeUppercase
10888     {\acronymfont
10889      {\glsshortpluralaccessdisplay
10890       {\glentryshortpl{\glslabel}}{\glslabel}}%
10891      \glsinsert}%
10892   }%
10893   }%
10894   {%
```

Subsequent singular form

```
10895     \glscapscase
10896     {%
```

Subsequent singular form, don't adjust case:

```
10897     \acronymfont
10898     {\glsshortaccessdisplay{\glentryshort{\glslabel}}{\glslabel}}%
10899     \glsinsert
10900   }%
10901   {%
```

Subsequent singular form, make first letter upper case:

```
10902      \acronymfont
10903      {\glsshortaccessdisplay{\Glsentryshort{\glslabel}}{\glslabel}}%
10904      \glsinsert
10905      }%
10906      {%
```

Subsequent singular form, all caps:

```
10907      \mfirstucMakeUppercase
10908      {\acronymfont{%
10909      \glsshortaccessdisplay{\Glsentryshort{\glslabel}}{\glslabel}}%
10910      \glsinsert}%
10911      }%
10912      }%
10913      }%
10914      {%
```

First use:

```
10915      \glsifplural
10916      {%
```

First use plural form:

```
10917      \glscapscase
10918      {%
```

First use plural form, don't adjust case:

```
10919      \genplacrfullformat{\glslabel}{\glsinsert}%
10920      }%
10921      {%
```

First use plural form, make first letter upper case:

```
10922      \Genplacrfullformat{\glslabel}{\glsinsert}%
10923      }%
10924      {%
```

First use plural form, all caps:

```
10925      \mfirstucMakeUppercase
10926      {\genplacrfullformat{\glslabel}{\glsinsert}}%
10927      }%
10928      }%
10929      {%
```

First use singular form

```
10930      \glscapscase
10931      {%
```

First use singular form, don't adjust case:

```
10932      \genacrfullformat{\glslabel}{\glsinsert}%
10933      }%
10934      {%
```

First use singular form, make first letter upper case:

```
10935      \Genacrfullformat{\glslabel}{\glsinsert}%
10936      }%
10937      {%
```

First use singular form, all caps:

```
10938      \mfirstucMakeUppercase
10939      {\genacrfullformat{\glslabel}{\glsinsert}}%
10940      }%
10941      }%
10942      }%
10943      }%
10944      {%
```

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.

```
10945      \glscustomtext
10946      }%
10947 }
```

**enacrfullformat** Redefine to include accessibility information.

```
10948 \renewcommand*{\genacrfullformat}[2]{%
10949   \glslongaccessdisplay{\glsentrylong{#1}}{#1}#2\space
10950   (\glsshortaccessdisplay{\protect\firstacronymfont{\glsentryshort{#1}}}{#1}}%
10951 }
```

**enacrfullformat** Redefine to include accessibility information.

```
10952 \renewcommand*{\Genacrfullformat}[2]{%
10953   \glslongaccessdisplay{\Glsentrylong{#1}}{#1}#2\space
10954   (\glsshortaccessdisplay{\protect\firstacronymfont{\Glsentryshort{#1}}}{#1}}%
10955 }
```

**placrfullformat** Redefine to include accessibility information.

```
10956 \renewcommand*{\genplacrfullformat}[2]{%
10957   \glslongpluralaccessdisplay{\glsentrylongpl{#1}}{#1}#2\space
10958   (\glsshortpluralaccessdisplay
10959     {\protect\firstacronymfont{\glsentryshortpl{#1}}}{#1}}%
10960 }
```

**placrfullformat** Redefine to include accessibility information.

```
10961 \renewcommand*{\Genplacrfullformat}[2]{%
10962   \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}#2\space
10963   (\glsshortpluralaccessdisplay
10964     {\protect\firstacronymfont{\Glsentryshortpl{#1}}}{#1}}%
10965 }
```

**\@acrshort**

```
10966 \def\@acrshort#1#2[#3]{%
10967   \glsdoifexists{#2}%
```

```

10968 {%
10969   \let\do@gl@s@link@checkfirsthyper\relax

10970   \let\gl@sifplural\@secondoftwo
10971   \let\gl@scapscase\@firstofthree
10972   \let\gl@sinsert\@empty
10973   \def\glscustomtext{%
10974     \acronymfont{\glsshortaccessdisplay{\glentryshort{#2}}{#2}}#3%
10975   }%

   Call \@gl@s@link
10976   \@gl@s@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10977 }%

10978 \glspostlinkhook
10979 }

```

\@Acrshort

```

10980 \def\@Acrshort#1#2[#3]{%
10981   \glstoifexists{#2}%
10982   {%
10983     \let\do@gl@s@link@checkfirsthyper\relax

10984     \let\gl@sifplural\@secondoftwo
10985     \let\gl@scapscase\@secondofthree
10986     \let\gl@sinsert\@empty
10987     \def\glscustomtext{%
10988       \acronymfont{\glsshortaccessdisplay{\Glsentryshort{#2}}{#2}}#3%
10989     }%

     Call \@gl@s@link
10990     \@gl@s@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10991   }%

10992   \glspostlinkhook
10993 }

```

\@ACRshort

```

10994 \def\@ACRshort#1#2[#3]{%
10995   \glstoifexists{#2}%
10996   {%
10997     \let\do@gl@s@link@checkfirsthyper\relax

10998     \let\gl@sifplural\@secondoftwo
10999     \let\gl@scapscase\@thirdofthree
11000     \let\gl@sinsert\@empty
11001     \def\glscustomtext{%
11002       \acronymfont{\glsshortaccessdisplay
11003         {\MakeUppercase{\glentryshort{#2}}}{#2}}#3%
11004     }%

```

```

    Call \@gls@link
11005     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
11006   }%

11007   \glspostlinkhook
11008 }

\@acrlong
11009 \def\@acrlong#1#2[#3]{%
11010   \glsdoifexists{#2}%
11011   {%
11012     \let\do@gls@link@checkfirsthyper\relax

11013     \let\glsifplural\@secondoftwo
11014     \let\glscapscase\@firstofthree
11015     \let\glsinsert\@empty
11016     \def\glscustomtext{%
11017       \acronymfont{\glslongaccessdisplay{\glsentrylong{#2}}{#2}}#3%
11018     }%

    Call \@gls@link
11019     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
11020   }%

11021   \glspostlinkhook
11022 }

\@Acrlong
11023 \def\@Acrlong#1#2[#3]{%
11024   \glsdoifexists{#2}%
11025   {%
11026     \let\do@gls@link@checkfirsthyper\relax

11027     \let\glsifplural\@secondoftwo
11028     \let\glscapscase\@firstofthree
11029     \let\glsinsert\@empty
11030     \def\glscustomtext{%
11031       \acronymfont{\glslongaccessdisplay{\Glsentrylong{#2}}{#2}}#3%
11032     }%

    Call \@gls@link
11033     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
11034   }%

11035   \glspostlinkhook
11036 }

\@ACRlong
11037 \def\@ACRlong#1#2[#3]{%
11038   \glsdoifexists{#2}%
11039   {%
11040     \let\do@gls@link@checkfirsthyper\relax

```



```

11041 \let\glsifplural\@secondoftwo
11042 \let\glsifscapscase\@firstofthree
11043 \let\glsinsert\@empty
11044 \def\glscustomtext{%
11045     \acronymfont{\glslongaccessdisplay{%
11046         \MakeUppercase{\glsentrylong{#2}}}{#2}#3}%
11047     }%

    Call \@gls@link
11048     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
11049 }%

11050 \glspostlinkhook
11051 }

```

## 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.

```

11052 \renewcommand*{\glossentryname}[1]{%
11053     \glsdoifexists{#1}%
11054     {%
11055         \glsnamefont{\glsnameaccessdisplay{\glsentryname{#1}}{#1}}%
11056     }%
11057 }

11058 \renewcommand*{\glossentryname}[1]{%
11059     \glsdoifexists{#1}%
11060     {%
11061         \glsnamefont{\glsnameaccessdisplay{\Glsentryname{#1}}{#1}}%
11062     }%
11063 }

11064 \renewcommand*{\glossentrydesc}[1]{%
11065     \glsdoifexists{#1}%
11066     {%
11067         \glsdescriptionaccessdisplay{\glsentrydesc{#1}}{#1}%
11068     }%
11069 }

11070 \renewcommand*{\Glossentrydesc}[1]{%
11071     \glsdoifexists{#1}%
11072     {%
11073         \glsdescriptionaccessdisplay{\Glsentrydesc{#1}}{#1}%
11074     }%
11075 }

```

```

11076 \renewcommand*{\glossentrysymbol}[1]{%
11077   \glsdoifexists{#1}%
11078   {%
11079     \glssymbolaccessdisplay{\glsentrysymbol{#1}}{#1}%
11080   }%
11081 }

11082 \renewcommand*{\Glossentrysymbol}[1]{%
11083   \glsdoifexists{#1}%
11084   {%
11085     \glssymbolaccessdisplay{\Glsentrysymbol{#1}}{#1}%
11086   }%
11087 }

```

ssaryentryfield

```

11088 \newcommand*{\accsuppglossaryentryfield}[5]{%
11089   \glossaryentryfield{#1}%
11090   {\glsnameaccessdisplay{#2}{#1}}%
11091   {\glsdescriptionaccessdisplay{#3}{#1}}%
11092   {\glssymbolaccessdisplay{#4}{#1}}{#5}%
11093 }

```

rysubentryfield

```

11094 \newcommand*{\accsuppglossarysubentryfield}[6]{%
11095   \glossarysubentryfield{#1}{#2}%
11096   {\glsnameaccessdisplay{#3}{#2}}%
11097   {\glsdescriptionaccessdisplay{#4}{#2}}%
11098   {\glssymbolaccessdisplay{#5}{#2}}{#6}%
11099 }

```

## 5.4 Acronyms

Redefine acronym styles provided by glossaries:

long-short    *<long>* (*<short>*) acronym style.

```

11100 \renewacronymstyle{long-short}%
11101 {%

```

Check for long form in case this is a mixed glossary.

```

11102   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
11103 }%
11104 {%
11105   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
11106   \renewcommand*{\genacrfullformat}[2]{%
11107     \glslongaccessdisplay{\glsentrylong{##1}}{##1}##2\space
11108     (\glsshortaccessdisplay
11109       {\protect\firstacronymfont{\glsentryshort{##1}}}{##1})%
11110   }%
11111   \renewcommand*{\Genacrfullformat}[2]{%

```

```

11112 \glslongaccessdisplay{\Glsentrylong{##1}}{##1}##2\space
11113 (\glsshortaccessdisplay
11114   {\protect\firstacronymfont{\glsentryshort{##1}}{##1}})%
11115 }%
11116 \renewcommand*{\genplacrfullformat}[2]{%
11117   \glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}##2\space
11118   (\glsshortpluralaccessdisplay
11119     {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}})%
11120 }%
11121 \renewcommand*{\Genplacrfullformat}[2]{%
11122   \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}##2\space
11123   (\glsshortpluralaccessdisplay
11124     {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}})%
11125 }%
11126 \renewcommand*{\acronymentry}[1]{%
11127   \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}{##1}}
11128 \renewcommand*{\acronymsort}[2]{##1}%
11129 \renewcommand*{\acronymfont}[1]{##1}%
11130 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
11131 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11132 }

```

short-long (*short*) (*long*) acronym style.

```

11133 \renewacronymstyle{short-long}%
11134 {%

```

Check for long form in case this is a mixed glossary.

```

11135 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
11136 }%
11137 {%
11138 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
11139 \renewcommand*{\genacrfullformat}[2]{%
11140   \glsshortaccessdisplay
11141     {\protect\firstacronymfont{\glsentryshort{##1}}{##1}##2\space
11142     (\glslongaccessdisplay{\glsentrylong{##1}}{##1}})%
11143 }%
11144 \renewcommand*{\Genacrfullformat}[2]{%
11145   \glsshortaccessdisplay
11146     {\protect\firstacronymfont{\Glsentryshort{##1}}{##1}##2\space
11147     (\glslongaccessdisplay{\glsentrylong{##1}}{##1}})%
11148 }%
11149 \renewcommand*{\genplacrfullformat}[2]{%
11150   \glsshortpluralaccessdisplay
11151     {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}##2\space
11152     (\glslongpluralaccessdisplay
11153       {\glsentrylongpl{##1}}{##1}})%
11154 }%
11155 \renewcommand*{\Genplacrfullformat}[2]{%
11156   \glsshortpluralaccessdisplay
11157     {\protect\firstacronymfont{\Glsentryshortpl{##1}}{##1}##2\space

```

```

11158 (\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1})%
11159 }%
11160 \renewcommand*{\acronymentry}[1]{%
11161   \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}%
11162 \renewcommand*{\acronymsort}[2]{##1}%
11163 \renewcommand*{\acronymfont}[1]{##1}%
11164 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
11165 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11166 }

```

long-short-desc *<long>* (*<short>*) acronym style that has an accompanying description (which the user needs to supply).

```

11167 \renewacronymstyle{long-short-desc}%
11168 {%
11169   \GlsUseAcrEntryDisplayStyle{long-short}%
11170 }%
11171 {%
11172   \GlsUseAcrStyleDefs{long-short}%
11173   \renewcommand*{\GenericAcronymFields}{}%
11174   \renewcommand*{\acronymsort}[2]{##2}%
11175   \renewcommand*{\acronymentry}[1]{%
11176     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11177     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11178 }

```

g-sc-short-desc *<long>* (\textsc{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

11179 \renewacronymstyle{long-sc-short-desc}%
11180 {%
11181   \GlsUseAcrEntryDisplayStyle{long-sc-short}%
11182 }%
11183 {%
11184   \GlsUseAcrStyleDefs{long-sc-short}%
11185   \renewcommand*{\GenericAcronymFields}{}%
11186   \renewcommand*{\acronymsort}[2]{##2}%
11187   \renewcommand*{\acronymentry}[1]{%
11188     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11189     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11190 }

```

g-sm-short-desc *<long>* (\textsmaller{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

11191 \renewacronymstyle{long-sm-short-desc}%
11192 {%
11193   \GlsUseAcrEntryDisplayStyle{long-sm-short}%
11194 }%
11195 {%
11196   \GlsUseAcrStyleDefs{long-sm-short}%
11197   \renewcommand*{\GenericAcronymFields}{}%

```

```

11198 \renewcommand*{\acronymsort}[2]{##2}%
11199 \renewcommand*{\acronymentry}[1]{%
11200   \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11201   (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11202 }

```

short-long-desc    *<short>* (*<long>*) acronym style that has an accompanying description (which the user needs to supply).

```

11203 \renewacronymstyle{short-long-desc}%
11204 {%
11205   \GlsUseAcrEntryDispStyle{short-long}%
11206 }%
11207 {%
11208   \GlsUseAcrStyleDefs{short-long}%
11209   \renewcommand*{\GenericAcronymFields}{}%
11210   \renewcommand*{\acronymsort}[2]{##2}%
11211   \renewcommand*{\acronymentry}[1]{%
11212     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11213     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11214 }

```

short-long-desc    *<long>* (*\textsc{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

11215 \renewacronymstyle{sc-short-long-desc}%
11216 {%
11217   \GlsUseAcrEntryDispStyle{sc-short-long}%
11218 }%
11219 {%
11220   \GlsUseAcrStyleDefs{sc-short-long}%
11221   \renewcommand*{\GenericAcronymFields}{}%
11222   \renewcommand*{\acronymsort}[2]{##2}%
11223   \renewcommand*{\acronymentry}[1]{%
11224     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11225     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11226 }

```

short-long-desc    *<long>* (*\textsmaller{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

11227 \renewacronymstyle{sm-short-long-desc}%
11228 {%
11229   \GlsUseAcrEntryDispStyle{sm-short-long}%
11230 }%
11231 {%
11232   \GlsUseAcrStyleDefs{sm-short-long}%
11233   \renewcommand*{\GenericAcronymFields}{}%
11234   \renewcommand*{\acronymsort}[2]{##2}%
11235   \renewcommand*{\acronymentry}[1]{%
11236     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11237     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%

```

11238 }

dua *<long>* only acronym style.

11239 \renewacronymstyle{dua}%

11240 {%

Check for long form in case this is a mixed glossary.

11241 \ifdefempty\glscustomtext

11242 {%

11243 \ifglshaslong{\glslabel}%

11244 {%

11245 \glsifplural

11246 {%

Plural form:

11247 \glscapscase

11248 {%

Plural form, don't adjust case:

11249 \gslongpluralaccessdisplay{\glentrylongpl{\glslabel}}{\glslabel}%

11250 \glsinsert

11251 }%

11252 {%

Plural form, make first letter upper case:

11253 \gslongpluralaccessdisplay{\Glentrylongpl{\glslabel}}{\glslabel}%

11254 \glsinsert

11255 }%

11256 {%

Plural form, all caps:

11257 \gslongpluralaccessdisplay

11258 {\mfirstucMakeUppercase{\glentrylongpl{\glslabel}}}{\glslabel}%

11259 \mfirstucMakeUppercase{\glsinsert}%

11260 }%

11261 }%

11262 {%

Singular form

11263 \glscapscase

11264 {%

Singular form, don't adjust case:

11265 \gslongaccessdisplay{\glentrylong{\glslabel}}{\glslabel}\glsinsert

11266 }%

11267 {%

Subsequent singular form, make first letter upper case:

11268 \gslongaccessdisplay{\Glentrylong{\glslabel}}{\glslabel}\glsinsert

11269 }%

11270 {%

Subsequent singular form, all caps:

```

11271      \glslongaccessdisplay
11272      {\mfirstucMakeUppercase
11273       {\glsentrylong{\glslabel}\glsinsert}}{\glslabel}%
11274      \mfirstucMakeUppercase{\glsinsert}%
11275      }%
11276      }%
11277      }%
11278      {%

```

Not an acronym:

```

11279      \glsgenentryfmt
11280      }%
11281      }%
11282      {\glscustomtext\glsinsert}%
11283      }%
11284      {%
11285      \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
11286      \renewcommand*{\acrfullfmt}[3]{%
11287        \glslink[##1]{##2}{%
11288          \glslongaccessdisplay{\glsentrylong{##2}}{##2}##3\space
11289          (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}})%
11290      \renewcommand*{\Acrfullfmt}[3]{%
11291        \glslink[##1]{##2}{%
11292          \glslongaccessdisplay{\Glsentrylong{##2}}{##2}##3\space
11293          (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}})%
11294      \renewcommand*{\ACRfullfmt}[3]{%
11295        \glslink[##1]{##2}{%
11296          \glslongaccessdisplay
11297          {\mfirstucMakeUppercase{\glsentrylong{##2}}{##2}##3\space
11298          (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}})%
11299      \renewcommand*{\acrfullplfmt}[3]{%
11300        \glslink[##1]{##2}{%
11301          \glslongpluralaccessdisplay
11302          {\glsentrylongpl{##2}}{##2}##3\space
11303          (\glsshortpluralaccessdisplay
11304          {\acronymfont{\glsentryshortpl{##2}}}{##2}})%
11305      \renewcommand*{\Acrfullplfmt}[3]{%
11306        \glslink[##1]{##2}{%
11307          \glslongpluralaccessdisplay
11308          {\Glsentrylongpl{##2}}{##2}##3\space
11309          (\glsshortpluralaccessdisplay
11310          {\acronymfont{\glsentryshortpl{##2}}}{##2}})%
11311      \renewcommand*{\ACRfullplfmt}[3]{%
11312        \glslink[##1]{##2}{%
11313          \glslongpluralaccessdisplay
11314          {\mfirstucMakeUppercase{\glsentrylongpl{##2}}{##2}##3\space
11315          (\glsshortpluralaccessdisplay
11316          {\acronymfont{\glsentryshortpl{##2}}}{##2}})%
11317      \renewcommand*{\glsentryfull}[1]{%

```

```

11318 \glslongaccessdisplay{\glsentrylong{##1}}\space
11319 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})%
11320 }%
11321 \renewcommand*{\Glsentryfull}[1]{%
11322 \glslongaccessdisplay{\Glsentrylong{##1}}{##1}\space
11323 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})%
11324 }%
11325 \renewcommand*{\glsentryfullpl}[1]{%
11326 \glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}\space
11327 (\glsshortpluralaccessdisplay{\acronymfont{\glsentryshortpl{##1}}}{##1})%
11328 }%
11329 \renewcommand*{\Glsentryfullpl}[1]{%
11330 \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}\space
11331 (\glsshortpluralaccessdisplay{\acronymfont{\glsentryshortpl{##1}}}{##1})%
11332 }%
11333 \renewcommand*{\acronymentry}[1]{%
11334 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}%
11335 \renewcommand*{\acronymsort}[2]{##1}%
11336 \renewcommand*{\acronymfont}[1]{##1}%
11337 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11338 }

```

dua-desc *<long>* only acronym style with user-supplied description.

```

11339 \renewacronymstyle{dua-desc}%
11340 {%
11341 \GlsUseAcrEntryDispStyle{dua}%
11342 }%
11343 {%
11344 \GlsUseAcrStyleDefs{dua}%
11345 \renewcommand*{\GenericAcronymFields}{}%
11346 \renewcommand*{\acronymentry}[1]{%
11347 \glslongaccessdisplay{\acronymfont{\glsentrylong{##1}}}{##1}}%
11348 \renewcommand*{\acronymsort}[2]{##2}%
11349 }%

```

footnote *<short>*\footnote{*<long>*} acronym style.

```

11350 \renewacronymstyle{footnote}%
11351 {%
11352 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
11353 }%
11354 {%
11355 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%

```

Need to ensure hyperlinks are switched off on first use:

```

11356 \glshyperfirstfalse
11357 \renewcommand*{\genacrfullformat}[2]{%
11358 \glsshortaccessdisplay
11359 {\protect\firstacronymfont{\glsentryshort{##1}}}{##1}##2%

```



```

11360 \protect\footnote{\glslongaccessdisplay{\glsentrylong{##1}}{##1}}%
11361 }%
11362 \renewcommand*{\Genacrfullformat}[2]{%
11363 \glsshortaccessdisplay
11364   {\firstacronymfont{\Glsentryshort{##1}}{##1}##2%
11365 \protect\footnote{\glslongaccessdisplay{\glsentrylong{##1}}{##1}}%
11366 }%
11367 \renewcommand*{\genplacrfullformat}[2]{%
11368 \glsshortpluralaccessdisplay
11369   {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}##2%
11370 \protect\footnote{\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}}%
11371 }%
11372 \renewcommand*{\Genplacrfullformat}[2]{%
11373 \glsshortpluralaccessdisplay
11374   {\protect\firstacronymfont{\Glsentryshortpl{##1}}{##1}##2%
11375 \protect\footnote{\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}}%
11376 }%
11377 \renewcommand*{\acronymentry}[1]{%
11378 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}{##1}}%
11379 \renewcommand*{\acronymsort}[2]{##1}%
11380 \renewcommand*{\acronymfont}[1]{##1}%
11381 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%

```

Don't use footnotes for \acrfull:

```

11382 \renewcommand*{\acrfullfmt}[3]{%
11383 \glslink{##1}{##2}{%
11384 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}{##2}##3\space
11385 (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}}%
11386 \renewcommand*{\Acrfullfmt}[3]{%
11387 \glslink{##1}{##2}{%
11388 \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##2}}{##2}##3\space
11389 (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}}%
11390 \renewcommand*{\ACRfullfmt}[3]{%
11391 \glslink{##1}{##2}{%
11392 \glsshortaccessdisplay
11393   {\mfirstucMakeUppercase
11394   {\acronymfont{\glsentryshort{##2}}{##2}##3\space
11395   (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}}%
11396 \renewcommand*{\acrfullplfmt}[3]{%
11397 \glslink{##1}{##2}{%
11398 \glsshortpluralaccessdisplay
11399   {\acronymfont{\glsentryshortpl{##2}}{##2}##3\space
11400   (\glslongpluralaccessdisplay{\glsentrylongpl{##2}}{##2})}}}%
11401 \renewcommand*{\Acrfullplfmt}[3]{%
11402 \glslink{##1}{##2}{%
11403 \glsshortpluralaccessdisplay
11404   {\acronymfont{\Glsentryshortpl{##2}}{##2}##3\space
11405   (\glslongpluralaccessdisplay{\glsentrylongpl{##2}}{##2})}}}%
11406 \renewcommand*{\ACRfullplfmt}[3]{%
11407 \glslink{##1}{##2}{%

```

```

11408      \glsshortpluralaccessdisplay
11409      {\mfirstucMakeUppercase
11410      {\acronymfont{\glentryshortpl{##2}}}{##2}##3\space
11411      (\glslongpluralaccessdisplay{\glentrylongpl{##2}}{##2}}}%

  Similarly for \glentryfull etc:
11412  \renewcommand*{\glentryfull}[1]{%
11413    \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}}{##1}\space
11414    (\glslongaccessdisplay{\glentrylong{##1}}{##1}})%
11415  \renewcommand*{\Glsentryfull}[1]{%
11416    \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##1}}}{##1}\space
11417    (\glslongaccessdisplay{\glentrylong{##1}}{##1}})%
11418  \renewcommand*{\glentryfullpl}[1]{%
11419    \glsshortpluralaccessdisplay
11420    {\acronymfont{\glentryshortpl{##1}}}{##1}\space
11421    (\glslongpluralaccessdisplay{\glentrylongpl{##1}}{##1}})%
11422  \renewcommand*{\Glsentryfullpl}[1]{%
11423    \glsshortpluralaccessdisplay
11424    {\acronymfont{\Glsentryshortpl{##1}}}{##1}\space
11425    (\glslongpluralaccessdisplay{\glentrylongpl{##1}}{##1}})%
11426 }

```

footnote-sc \textsc{<short>}\footnote{<long>} acronym style.

```

11427 \renewacronymstyle{footnote-sc}%
11428 {%
11429   \GlsUseAcrEntryDispStyle{footnote}%
11430 }%
11431 {%
11432   \GlsUseAcrStyleDefs{footnote}%
11433   \renewcommand{\acronymentry}[1]{%
11434     \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}}{##1}}
11435   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
11436   \renewcommand*{\acrpluralsuffix}{\glstextup{\glspluralsuffix}}%
11437 }%

```

footnote-sm \textsmaller{<short>}\footnote{<long>} acronym style.

```

11438 \renewacronymstyle{footnote-sm}%
11439 {%
11440   \GlsUseAcrEntryDispStyle{footnote}%
11441 }%
11442 {%
11443   \GlsUseAcrStyleDefs{footnote}%
11444   \renewcommand{\acronymentry}[1]{%
11445     \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}}{##1}}
11446   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
11447   \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11448 }%

```

footnote-desc <short>\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

11449 \renewacronymstyle{footnote-desc}%
11450 {%
11451   \GlsUseAcrEntryDisplayStyle{footnote}%
11452 }%
11453 {%
11454   \GlsUseAcrStyleDefs{footnote}%
11455   \renewcommand*{\GenericAcronymFields}{}%
11456   \renewcommand*{\acronymsort}[2]{##2}%
11457   \renewcommand*{\acronymentry}[1]{%
11458     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11459     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11460 }

```

ootnote-sc-desc \textsc{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

11461 \renewacronymstyle{footnote-sc-desc}%
11462 {%
11463   \GlsUseAcrEntryDisplayStyle{footnote-sc}%
11464 }%
11465 {%
11466   \GlsUseAcrStyleDefs{footnote-sc}%
11467   \renewcommand*{\GenericAcronymFields}{}%
11468   \renewcommand*{\acronymsort}[2]{##2}%
11469   \renewcommand*{\acronymentry}[1]{%
11470     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11471     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11472 }

```

ootnote-sm-desc \textsmaller{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

11473 \renewacronymstyle{footnote-sm-desc}%
11474 {%
11475   \GlsUseAcrEntryDisplayStyle{footnote-sm}%
11476 }%
11477 {%
11478   \GlsUseAcrStyleDefs{footnote-sm}%
11479   \renewcommand*{\GenericAcronymFields}{}%
11480   \renewcommand*{\acronymsort}[2]{##2}%
11481   \renewcommand*{\acronymentry}[1]{%
11482     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11483     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11484 }

```

Use \newacronymhook to modify the key list to set the access text to the long version by default.

```

11485 \renewcommand*{\newacronymhook}{%
11486   \edef\@gls@keylist{shortaccess=\the\glslongtok,%
11487     \the\glskeylisttok}%
11488   \expandafter\glskeylisttok\expandafter{\@gls@keylist}%

```

11489 }

1tNewAcronymDef Modify default style to use access text:

```
11490 \renewcommand*{\DefaultNewAcronymDef}{%
11491   \edef\@do@newglossaryentry{%
11492     \noexpand\newglossaryentry{\the\glslabeltok}%
11493     {%
11494       type=\acronymtype,%
11495       name={\the\glsshorttok},%
11496       description={\the\glslongtok},%
11497       descriptionaccess=\relax,%
11498       text={\the\glsshorttok},%
11499       access={\noexpand\@glo@textaccess},%
11500       sort={\the\glsshorttok},%
11501       short={\the\glsshorttok},%
11502       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11503       shortaccess={\the\glslongtok},%
11504       long={\the\glslongtok},%
11505       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11506       descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11507       first={\noexpand\glslongaccessdisplay
11508         {\the\glslongtok}{\the\glslabeltok}\space
11509         (\noexpand\glsshortaccessdisplay
11510         {\the\glsshorttok}{\the\glslabeltok})},%
11511       plural={\the\glsshorttok\acrpluralsuffix},%
11512       firstplural={\noexpand\glslongpluralaccessdisplay
11513         {\noexpand\@glo@longpl}{\the\glslabeltok}\space
11514         (\noexpand\glsshortpluralaccessdisplay
11515         {\noexpand\@glo@shortpl}{\the\glslabeltok})},%
11516       firstaccess=\relax,%
11517       firstpluralaccess=\relax,%
11518       textaccess={\noexpand\@glo@shortaccess},%
11519       \the\glskeylisttok
11520     }%
11521   }%
11522   \let\@org@gls@assign@firstpl\gls@assign@firstpl
11523   \let\@org@gls@assign@plural\gls@assign@plural
11524   \let\@org@gls@assign@descplural\gls@assign@descplural
11525   \def\gls@assign@firstpl##1##2{%
11526     \@@gls@expand@field{##1}{firstpl}{##2}%
11527   }%
11528   \def\gls@assign@plural##1##2{%
11529     \@@gls@expand@field{##1}{plural}{##2}%
11530   }%
11531   \def\gls@assign@descplural##1##2{%
11532     \@@gls@expand@field{##1}{descplural}{##2}%
11533   }%
11534   \@do@newglossaryentry
11535   \let\gls@assign@firstpl\@org@gls@assign@firstpl
```

```

11536 \let\gls@assign@plural\@org@gls@assign@plural
11537 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11538 }

```

teNewAcronymDef

```

11539 \renewcommand*{\DescriptionFootnoteNewAcronymDef}{%
11540 \edef\@do@newglossaryentry{%
11541 \noexpand\newglossaryentry{\the\glslabeltok}%
11542 {%
11543 type=\acronymtype,%
11544 name={\noexpand\acronymfont{\the\glsshorttok}},%
11545 sort={\the\glsshorttok},%
11546 text={\the\glsshorttok},%
11547 short={\the\glsshorttok},%
11548 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11549 shortaccess={\the\glslongtok},%
11550 long={\the\glslongtok},%
11551 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11552 access={\noexpand\@glo@textaccess},%
11553 plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11554 symbol={\the\glslongtok},%
11555 symbolplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11556 firstpluralaccess=\relax,
11557 textaccess={\noexpand\@glo@shortaccess},%
11558 \the\glskeylisttok
11559 }%
11560 }%
11561 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11562 \let\@org@gls@assign@plural\gls@assign@plural
11563 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
11564 \def\gls@assign@firstpl##1##2{%
11565 \@@gls@expand@field{##1}{firstpl}{##2}%
11566 }%
11567 \def\gls@assign@plural##1##2{%
11568 \@@gls@expand@field{##1}{plural}{##2}%
11569 }%
11570 \def\gls@assign@symbolplural##1##2{%
11571 \@@gls@expand@field{##1}{symbolplural}{##2}%
11572 }%
11573 \do@newglossaryentry
11574 \let\gls@assign@plural\@org@gls@assign@plural
11575 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11576 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11577 }

```

onNewAcronymDef

```

11578 \renewcommand*{\DescriptionNewAcronymDef}{%
11579 \edef\@do@newglossaryentry{%
11580 \noexpand\newglossaryentry{\the\glslabeltok}%

```

```

11581 {%
11582     type=\acronymtype,%
11583     name={\noexpand
11584         \acrnameformat{\the\glsshorttok}{\the\glslongtok}},%
11585     access={\noexpand\@glo@textaccess},%
11586     sort={\the\glsshorttok},%
11587     short={\the\glsshorttok},%
11588     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11589     shortaccess={\the\glslongtok},%
11590     long={\the\glslongtok},%
11591     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11592     first={\the\glslongtok},%
11593     firstaccess=\relax,
11594     firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11595     text={\the\glsshorttok},%
11596     textaccess={\the\glslongtok},%
11597     plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11598     symbol={\noexpand\@glo@text},%
11599     symbolaccess={\noexpand\@glo@textaccess},%
11600     symbolplural={\noexpand\@glo@plural},%
11601     firstpluralaccess=\relax,
11602     textaccess={\noexpand\@glo@shortaccess},%
11603     \the\glskeylisttok}%
11604 }%
11605 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11606 \let\@org@gls@assign@plural\gls@assign@plural
11607 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
11608 \def\gls@assign@firstpl##1##2{%
11609     \@gls@expand@field{##1}{firstpl}{##2}%
11610 }%
11611 \def\gls@assign@plural##1##2{%
11612     \@gls@expand@field{##1}{plural}{##2}%
11613 }%
11614 \def\gls@assign@symbolplural##1##2{%
11615     \@gls@expand@field{##1}{symbolplural}{##2}%
11616 }%
11617 \do@newglossaryentry
11618 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11619 \let\gls@assign@plural\@org@gls@assign@plural
11620 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11621 }

```

teNewAcronymDef

```

11622 \renewcommand*{\FootnoteNewAcronymDef}{%
11623     \edef\@do@newglossaryentry{%
11624         \noexpand\newglossaryentry{\the\glslabeltok}%
11625         {%
11626             type=\acronymtype,%
11627             name={\noexpand\acronymfont{\the\glsshorttok}},%

```

```

11628     sort={\the\glsshorttok},%
11629     text={\the\glsshorttok},%
11630     textaccess={\the\glslongtok},%
11631     access={\noexpand\@glo@textaccess},%
11632     plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11633     short={\the\glsshorttok},%
11634     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11635     long={\the\glslongtok},%
11636     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11637     description={\the\glslongtok},%
11638     descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11639     \the\glskeylisttok
11640   }%
11641 }%
11642 \let\@org@gls@assign@plural\gls@assign@plural
11643 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11644 \let\@org@gls@assign@descplural\gls@assign@descplural
11645 \def\gls@assign@firstpl##1##2{%
11646   \@@gls@expand@field{##1}{firstpl}{##2}%
11647 }%
11648 \def\gls@assign@plural##1##2{%
11649   \@@gls@expand@field{##1}{plural}{##2}%
11650 }%
11651 \def\gls@assign@descplural##1##2{%
11652   \@@gls@expand@field{##1}{descplural}{##2}%
11653 }%
11654 \do@newglossaryentry
11655 \let\gls@assign@plural\@org@gls@assign@plural
11656 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11657 \let\gls@assign@descplural\@org@gls@assign@descplural
11658 }

```

# 11NewAcronymDef

```

11659 \renewcommand*{\SmallNewAcronymDef}{%
11660   \edef\@do@newglossaryentry{%
11661     \noexpand\newglossaryentry{\the\glslabeltok}%
11662     {%
11663       type=\acronymtype,%
11664       name={\noexpand\acronymfont{\the\glsshorttok}},%
11665       access={\noexpand\@glo@symbolaccess},%
11666       sort={\the\glsshorttok},%
11667       short={\the\glsshorttok},%
11668       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11669       shortaccess={\the\glslongtok},%
11670       long={\the\glslongtok},%
11671       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11672       text={\noexpand\@glo@short},%
11673       textaccess={\noexpand\@glo@shortaccess},%
11674       plural={\noexpand\@glo@shortpl},%

```

```

11675     first={\the\glslongtok},%
11676     firstaccess=\relax,
11677     firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11678     description={\noexpand\@glo@first},%
11679     descriptionplural={\noexpand\@glo@firstplural},%
11680     symbol={\the\glsshorttok},%
11681     symbolaccess={\the\glslongtok},%
11682     symbolplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11683     \the\glskeylisttok
11684   }%
11685 }%
11686 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11687 \let\@org@gls@assign@plural\gls@assign@plural
11688 \let\@org@gls@assign@descplural\gls@assign@descplural
11689 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
11690 \def\gls@assign@firstpl##1##2{%
11691   \@@gls@expand@field{##1}{firstpl}{##2}%
11692 }%
11693 \def\gls@assign@plural##1##2{%
11694   \@@gls@expand@field{##1}{plural}{##2}%
11695 }%
11696 \def\gls@assign@descplural##1##2{%
11697   \@@gls@expand@field{##1}{descplural}{##2}%
11698 }%
11699 \def\gls@assign@symbolplural##1##2{%
11700   \@@gls@expand@field{##1}{symbolplural}{##2}%
11701 }%
11702 \do@newglossaryentry
11703 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11704 \let\gls@assign@plural\@org@gls@assign@plural
11705 \let\gls@assign@descplural\@org@gls@assign@descplural
11706 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11707 }

```

The following are kept for compatibility with versions before 3.0:

sshortaccesskey

```

11708 \newcommand*{\glsshortaccesskey}{\glsshortkey access}%

```

pluralaccesskey

```

11709 \newcommand*{\glsshortpluralaccesskey}{\glsshortpluralkey access}%

```

lslongaccesskey

```

11710 \newcommand*{\glslongaccesskey}{\glslongkey access}%

```

pluralaccesskey

```

11711 \newcommand*{\glslongpluralaccesskey}{\glslongpluralkey access}%

```



## 5.5 Debugging Commands

owglonameaccess

```
11712 \newcommand*{\showglonameaccess}[1]{%
11713   \expandafter\show\csname glo@\glsdetoklabel{#1}@textaccess\endcsname
11714 }
```

owglotextaccess

```
11715 \newcommand*{\showglotextaccess}[1]{%
11716   \expandafter\show\csname glo@\glsdetoklabel{#1}@textaccess\endcsname
11717 }
```

glopluralaccess

```
11718 \newcommand*{\showglopluralaccess}[1]{%
11719   \expandafter\show\csname glo@\glsdetoklabel{#1}@pluralaccess\endcsname
11720 }
```

wglofirstaccess

```
11721 \newcommand*{\showwlofirstaccess}[1]{%
11722   \expandafter\show\csname glo@\glsdetoklabel{#1}@firstaccess\endcsname
11723 }
```

rstpluralaccess

```
11724 \newcommand*{\showrstpluralaccess}[1]{%
11725   \expandafter\show\csname glo@\glsdetoklabel{#1}@rstpluralaccess\endcsname
11726 }
```

glosymbolaccess

```
11727 \newcommand*{\showglosymbolaccess}[1]{%
11728   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolaccess\endcsname
11729 }
```

bolpluralaccess

```
11730 \newcommand*{\showbolpluralaccess}[1]{%
11731   \expandafter\show\csname glo@\glsdetoklabel{#1}@bolpluralaccess\endcsname
11732 }
```

owglodescaccess

```
11733 \newcommand*{\showglodescaccess}[1]{%
11734   \expandafter\show\csname glo@\glsdetoklabel{#1}@descaccess\endcsname
11735 }
```

escpluralaccess

```
11736 \newcommand*{\showescpluralaccess}[1]{%
11737   \expandafter\show\csname glo@\glsdetoklabel{#1}@escpluralaccess\endcsname
11738 }
```

wgloshortaccess

```
11739 \newcommand*{\showgloshortaccess}[1]{%  
11740   \expandafter\show\csname glo@\glsdetoklabel{#1}@shortaccess\endcsname  
11741 }
```

ortpluralaccess

```
11742 \newcommand*{\showgloshortpluralaccess}[1]{%  
11743   \expandafter\show\csname glo@\glsdetoklabel{#1}@shortpluralaccess\endcsname  
11744 }
```

owglolongaccess

```
11745 \newcommand*{\showglolongaccess}[1]{%  
11746   \expandafter\show\csname glo@\glsdetoklabel{#1}@longaccess\endcsname  
11747 }
```

ongpluralaccess

```
11748 \newcommand*{\showglolongpluralaccess}[1]{%  
11749   \expandafter\show\csname glo@\glsdetoklabel{#1}@longpluralaccess\endcsname  
11750 }
```

## 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.

```
11751 \NeedsTeXFormat{LaTeX2e}
11752 \ProvidesPackage{glossaries-babel}[2018/04/07 v4.37 (NLCT)]
```

Load tracklang to obtain language settings.

```
11753 \RequirePackage{tracklang}
11754 \let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
11755 \AnyTrackedLanguages
11756 {%
11757   \ForEachTrackedDialect{\this@dialect}{%
11758     \IfTrackedLanguageFileExists{\this@dialect}%
11759       {glossaries-}% prefix
11760       {.ldf}%
11761       {%
11762         \RequireGlossariesLang{\CurrentTrackedTag}%
11763       }%
11764       {%
11765         \PackageWarningNoLine{glossaries}%
11766           {No language module detected for ‘\this@dialect’.\MessageBreak
11767             Language modules need to be installed separately.\MessageBreak
11768             Please check on CTAN for a bundle called\MessageBreak
11769             ‘glossaries-\CurrentTrackedLanguage’ or similar}%
11770       }%
11771     }%
11772   }%
11773 }
```

### 6.1 Polyglossia Captions

Language support has now been split off into independent language modules.

```
11774 \NeedsTeXFormat{LaTeX2e}
11775 \ProvidesPackage{glossaries-polyglossia}[2018/04/07 v4.37 (NLCT)]
```

Load tracklang to obtain language settings.

```
11776 \RequirePackage{tracklang}
11777 \let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
11778 \AnyTrackedLanguages
```

```

11779 {%
11780     \ForEachTrackedDialect{\this@dialect}{%
11781         \IfTrackedLanguageFileExists{\this@dialect}%
11782         {glossaries-}% prefix
11783         {.ldf}%
11784         {%
11785             \RequireGlossariesLang{\CurrentTrackedTag}%
11786         }%
11787         {%
11788             \PackageWarningNoLine{glossaries}%
11789             {No language module detected for ‘\this@dialect’.\MessageBreak
11790             Language modules need to be installed separately.\MessageBreak
11791             Please check on CTAN for a bundle called\MessageBreak
11792             ‘glossaries-\CurrentTrackedLanguage’ or similar}%
11793         }%
11794     }%
11795 }%
11796 {}%

```

# Glossary

`makeindex` An indexing application. [9](#), [12](#), [28](#), [29](#), [178](#)

`xindy` An flexible indexing application with multilingual support written in Perl. [9](#), [12](#), [28](#), [29](#), [178](#)

# Change History

1.01 (2007-05-17)	numberline: numberline option added . . . 7
General: Added range facility in format key . . . . . 113	
\writeist: Added spaces after \delimN and \delimR in ist file . . . . . 160	
1.04 (2007-08-03)	1.12 (2008-03-08)
General: Added \glstextformat . . . . . 97	\@GLSpl: now uses
1.05 (2007-08-10)	\glentrydescplural and
\glossarysection: added \@mkboth to \glossarysection . . . . . 40	\glentrysymbolplural instead of
\gls@defglossaryentry: Changed the default value of the sort key to just the value of the name key . . . . . 81	\glentrydesc and
1.07 (2007-09-13)	\glentrysymbol . . . . . 126
\@gls@link: fixed bug caused by \theglentrycounter setting the page number too soon . . . . . 111	\@Glspl@: now uses
\glsadd: fixed bug caused by \theglentrycounter setting the page number too soon . . . . . 157	\glentrydescplural and
1.08 (2007-10-13)	\glentrysymbolplural instead of
General: Added babel support . . . . . 34	\glentrydesc and
listgroup: changed listgroup style to use \glsgetgrouptitle . . . . . 274	\glentrysymbol . . . . . 125
altlistgroup: changed altlistgroup style to use \glsgetgrouptitle . . . . . 275	General: added check for \hypertarget separate to \hyperlink (memoir defines \hyperlink but not \hypertarget) . . . . . 121
1.1 (2008-02-22)	descriptionplural: new . . . . . 63
\@glossarysection: numbered sections and auto label added . . . . . 41	\gls@defglossaryentry: Changed default first plural to be first key with s appended (was text key with s appended) . . . . . 81
\@gls@tmpb: changed \toksdef to \newtoks . . . . . 115	descriptionplural support added . . . . . 80
\@gls@toc: numberline added . . . . . 42	symbolplural support added . . . . . 80
\@p@glossarysection: numbered sections and auto label added . . . . . 41	\Glsentrydescplural: New . . . . . 150
General: amsgen now loaded (\new@ifnextchar needed) . . . . . 4	\glentrydescplural: New . . . . . 150
translate: translate option added . . . . . 25	\Glsentrysymbolplural: New . . . . . 151
\setglossarysection: new . . . . . 41	\glentrysymbolplural: New . . . . . 151
numberedsection: numberedsection package option added . . . . . 8	\SetDescriptionFootnoteAcronymStyle: Added \protect before \footnote and \glslink . . . . . 240
	\SetFootnoteAcronymStyle: Added \protect before \footnote and \glslink . . . . . 246
	symbolplural: new . . . . . 64

1.13 (2008-05-10)	
General: fixed bug that ignored 3rd parameter .....	128–135
\ACRfullpl: new .....	221
\Acrfullpl: new .....	221
\acrfullpl: new .....	220
\acrpluralsuffix: New .....	218
\gls@defglossaryentry: Changed default first value .....	81
Changed default firstplural value .....	81
Removed restriction on only using \newglossaryentry in the preamble	86
\newacronym: Removed restriction on only using \newacronym in the preamble .....	218
1.14 (2008-06-17)	
\@gls@hypergroup: new .....	269
General: added nonumberlist key to \printglossary .....	204
added numberedsection key to \printglossary .....	202
\firstacronymfont: new .....	221
\glsautoprefix: new .....	7
\glsnavhyperlink: changed \edef to \protected@edef .....	268
\glsnavhypertarget: added write to aux file .....	268
\glsnavigation: changed to only use labels for groups that are present ..	270
1.15 (2008-08-15)	
\@gls@link: added \glslabel .....	111
\gls@defglossaryentry: check for \@glo@first in description .....	84
check for \@glo@text in symbol .....	85
\gls@hypergroup: new .....	269
\glsnavhypertarget: added check if rerun required .....	268
\glssettoctitle: new .....	33
\printglossary: changed the way the TOC title is set .....	188
1.16 (2008-08-27)	
\@GLS@: Test glossary type is \acronymtype in addition to checking if footnote option has been used .....	124
\@GLSpl: Test glossary type is \acronymtype in addition to checking if footnote option has been used .....	126
\@Gls@: Test glossary type is \acronymtype in addition to checking if footnote option has been used .....	123
\@Glspl@: Test glossary type is \acronymtype in addition to checking if footnote option has been used .....	126
\@gls@: Test glossary type is \acronymtype in addition to checking if footnote option has been used .....	122
\@glsdisp: 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 .....	125
\@glstarget: raised the hypertarget so the target text doesn't scroll off the top of the page .....	121
\gls@defglossaryentry: Changed def to let .....	81
1.17 (2008-12-26)	
\@do@esc@wrglossary: new .....	182
\@do@seeglossary: new .....	186
\@glo@storeentry: new .....	87
\@gls@glossary: changed definition to use \index instead of \@index ....	178
\@glsdefaultplural: new .....	67
\@glsdefaultsort: new .....	68
\@glshypernumber: new .....	215
\@glsnoname: new .....	67
\@glsnonextpages: new .....	205
General: added xindy support .....	28
parent: new .....	65
see: new .....	64
\gls@defglossaryentry: added nonumberlist key .....	81
added parent key .....	81
added see key .....	81
Stored main part of entry format when entry is defined .....	85
\gls@suffixF: new .....	38
\gls@suffixFF: new .....	38
\gls@wrglossary: modified to allow for xindy support .....	179

\glshyperlink: new .....	156	\SetDescriptionFootnoteAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	240
\glshypernumber: modified to allow material to be attached to location	215	\SetFootnoteAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller .....	246
\glshnavhyperlink: replaced \hyperlink to \@glslink .....	268	\SetSmallAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller .....	249
\glshnavhypertarget: replaced \hypertarget to \@glstarget ...	268	2.01 (2009 May 30) \@glsl@link: moved \@do@wrglossary before term is displayed to prevent unwanted whatsit .....	111
\glsssee: new .....	187	\forall glossaries: replaced \ifthenelse with \ifx .....	52
\glssseeformat: new .....	187	\forall glossentries: replaced \ifthenelse with \ifx .....	52
\glssSetSuffixF: new .....	38	\glssdefmain: new .....	15
\glssSetSuffixFF: new .....	38	\glssdescwidth: changed \linewidth to \hsize .....	276, 298
\ifglssxindy: new .....	28	\glsslistdottedwidth: changed \linewidth to \hsize .....	276
\istfilename: added xindy support ...	37	\glsspagelistwidth: changed \linewidth to \hsize .....	276, 298
\newglossarystyle: made \newglossarystyle long .....	214	nomain: added nomain package option	15
\nopostdesc: new .....	36	\writeist: removed item_02 - no such makeindex key .....	164
nonumberlist: new .....	65	2.02 (2007-07-13) \@printglossary: suppressed warning globally rather than locally .....	191
\printglossary: added check to determine if \printglossary is already defined .....	188	2.02 (2009-07-13) \glossarysection: changed \@mkboth to \glossarymark .....	40
added print language to aux file .....	188	\glssglossarymark: New .....	40
order: order package option added	27	2.03 (2009-09-23) \@GLS@: Added check for hyperfirst	124
\writeist: added xindy support	160	\@GLSpl: Added check for hyperfirst	126
1.18 (2009-01-14) \@glsl@loadlist: new .....	10	\@GLs@: Added check for hyperfirst	123
\@glsl@loadlong: new .....	9	\@GLspl@: Added check for hyperfirst	126
\@glsl@loadsuper: new .....	10	\@glsl@: Added check for hyperfirst	122
\@glsl@loadtree: new .....	10	\@glsl@@link: new .....	109
\glsl@defglossaryentry: Changed default value of sort to \@glsldefaultsort .....	81	\@glsl@link: added \leavevmode	111
moved sort sanitization to \newglossaryentry .....	85	Moved entry existence check to avoid duplicate code .....	111
\glstarget: new .....	208	\@glsl@disp: Added check for hyperfirst	127
\oldacronym: new .....	217	\@glspl@: Added check for hyperfirst	125
nolist: new .....	10	\glssglossarymark: Added check to see if it's already defined .....	40
nolong: new .....	9	hyperfirst: new .....	26
sort: moved sanitization to \newglossaryentry .....	63		
nostyles: new .....	10		
nosuper: new .....	10		
notree: new .....	10		
1.19 (2009-03-02) \glsclearpage: new .....	42		
\glssdisp: new .....	127		
\SetDescriptionAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	244		



2.04 (2009-11-10)	
\@GLS@: Changed test to check if glossary type has been identified as a list of acronyms .....	124
\@GLSpl: Changed test to check if glossary type has been identified as a list of acronyms .....	126
\@GLs@: Changed test to check if glossary type has been identified as a list of acronyms .....	123
\@GLspl@: Changed test to check if glossary type has been identified as a list of acronyms .....	126
\@glossaryentryfield: new .....	86
\@glossarysubentryfield: new .....	87
\@gls@: Changed test to check if glossary type has been identified as a list of acronyms .....	122
\@glsacronymlists: new .....	16
\@glsdisp: 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 .....	125
\@newglossaryentryposthook: new ..	86
\@newglossaryentryprehook: new ...	86
acronymlists: new .....	18
\DeclareAcronymList: new .....	17
\DefineAcronymSynonyms: new .....	235
\gls@defglossaryentry: added userl-6 keys .....	81
\glsadd: fixed bug that ignored counter	157
\Glsentryuseri: new .....	152
\glsentryuseri: new .....	152
\Glsentryuserii: new .....	153
\glsentryuserii: new .....	152
\Glsentryuseriii: new .....	153
\glsentryuseriii: new .....	153
\Glsentryuseriv: new .....	153
\glsentryuseriv: new .....	153
\Glsentryuserv: new .....	153
\glsentryuserv: new .....	153
\Glsentryuservi: new .....	153
\glsentryuservi: new .....	153
\ns@newglossary: added check to determine if \gls@<type>@display and \gls@<type>@displayfirst have been defined. ....	60
\SetAcronymLists: new .....	18
\SetDefaultAcronymDisplayStyle: new .....	236
\SetDefaultAcronymStyle: new ....	237
\SetDescriptionAcronymDisplayStyle: new .....	242
\SetDescriptionDUAAcronymDisplayStyle: new .....	240
\SetDescriptionFootnoteAcronymDisplayStyle: new .....	238
\SetDUADisplayStyle: new .....	250
\SetFootnoteAcronymDisplayStyle: new .....	244
\SetSmallAcronymDisplayStyle: new	247
2.05 (2010-02-06)	
\@glsdisp: Added closing brace. Patch provided by Sergiu Dotenco .....	127
Removed spurious brace. Patch provided by Sergiu Dotenco .....	127
\writeist: Added \string before opening and closing braces. Patch provided by Segiu Dotenco .....	165
2.06 (2010-06-14)	
\altnewglossary: new .....	61
\CustomAcronymFields: new .....	252
\CustomNewAcronymDef: new .....	252
\SetCustomDisplayStyle: new .....	252
\SetCustomStyle: new .....	253
2.07 (2010-07-10)	
General: glsadd format key stored in \@glsnumberformat (was mistakenly stored in \@glo@format) .....	156
3.0 (2010-07-12)	
\@makeglossary: Added check for savewrites .....	169
\gls@wrglossary: modified to take into account savewrites .....	179
3.0 (2010/03/31)	
\@set@glo@numformat: added 4th argument .....	113
3.0 (2011-04-02)	
\@do@esc@wrglossary: added check for hyper location prefix .....	184
modified to use new format .....	182
\@@glossarysec: replaced \@ifundefined with \ifcsundef ...	7
\@do@seeglossary: Sanitize and escape cross-referencing information ....	186
\@gls@counterwithin: new .....	11

<code>\@gls@ifinlist</code> : new .....	43	<code>\glsadd</code> : added	
<code>\@gls@link</code> : added		<code>\@gls@saveentrycounter</code> .....	157
<code>\@gls@saveentrycounter</code> .....	111	<code>\GlsAddXdyCounters</code> : new .....	44
added <code>\@gls@setsort</code> .....	111	<code>\glentrycounterlabel</code> : new .....	207
<code>\@gls@saveentrycounter</code> : new .....	112	<code>\glentryitem</code> : new .....	207
<code>\@gls@setupsort@def</code> : new .....	13	<code>\Glentrylong</code> : new .....	154
<code>\@gls@setupsort@standard</code> : new ....	12	<code>\glentrylong</code> : new .....	154
<code>\@gls@setupsort@use</code> : new .....	13	<code>\Glentrylongpl</code> : new .....	154
<code>\@gls@xdy@locationlist</code> : new .....	46	<code>\glentrylongpl</code> : new .....	154
<code>\@glslink</code> : replaced <code>\@ifundefined</code>		<code>\Glentryshort</code> : new .....	154
with <code>\ifcsundef</code> .....	121	<code>\glentryshort</code> : new .....	154
<code>\@glsnextpages</code> : new .....	205	<code>\Glentryshortpl</code> : new .....	154
<code>\@print@glossary</code> : replaced		<code>\glentryshortpl</code> : new .....	154
<code>\@ifundefined</code> with <code>\ifcsundef</code> ..	191	<code>\glsgetgrouptitle</code> : replaced	
<code>\@printglossary</code> : added		<code>\@ifundefined</code> with <code>\ifcsundef</code> ..	211
<code>\currentglossary</code> .....	190	<code>\gls glossarymark</code> : replaced	
added <code>\glsnextpages</code> .....	190	<code>\@ifundefined</code> with <code>\ifcsundef</code> ..	40
make toctitle default to title .....	190	<code>\glshyperlink</code> : changed default from	
<code>\@xdyattributelist</code> : new .....	43	<code>\glentryname</code> to <code>\glentrytext</code> ..	156
General: added prefix to hyperlink ....	216	<code>\glshypernumber</code> : replaced	
etoolbox now loaded .....	4	<code>\@ifundefined</code> with <code>\ifcsundef</code> ..	215
replaced <code>\@ifundefined</code> with		<code>\glsnumberformat</code> : replaced	
<code>\ifcsundef</code> .....	32, 34, 107, 202	<code>\@ifundefined</code> with <code>\ifcsundef</code> ..	38
<code>\acrfootnote</code> : new .....	238	<code>\glsrefentry</code> : new .....	207
<code>\ACRfull</code> : added starred version .....	220	<code>\glsresetsubentrycounter</code> : new ...	206
<code>\Acrfull</code> : added starred version .....	219	<code>\glsseeitem</code> : hyperlink uses	
<code>\acrfull</code> : added starred version .....	219	<code>\glsseeitemformat</code> instead of	
<code>\ACRfullpl</code> : added starred version ...	221	<code>\glentryname</code> .....	188
<code>\Acrfullpl</code> : added starred version ...	221	<code>\glsseeitemformat</code> : new .....	188
<code>\acrfullpl</code> : added starred version ...	220	<code>\glssortnumberfmt</code> : new .....	12
<code>\acrlinkfootnote</code> : new .....	238	<code>\glsstepentry</code> : new .....	206
<code>\acrno linkfootnote</code> : new .....	238	<code>\glsstepsubentry</code> : new .....	206
<code>save writes</code> : new .....	29	<code>\glssubentrycounterlabel</code> : new ...	207
<code>see</code> : added <code>\@glo@seeautonumberlist</code> ..	64	<code>\glssubentryitem</code> : new .....	207
<code>seeautonumberlist</code> : new .....	9	<code>theglossary</code> : replaced <code>\@ifundefined</code>	
<code>\glossarysection</code> : replaced		with <code>\ifcsundef</code> .....	208
<code>\@ifundefined</code> with <code>\ifcsundef</code> ..	40	<code>short</code> : new .....	67
<code>\glossarystyle</code> : replaced		<code>shortplural</code> : new .....	67
<code>\@ifundefined</code> with <code>\ifcsundef</code> ..	213	<code>\ifglossaryexists</code> : replaced	
<code>\gls@codepage</code> : replaced		<code>\@ifundefined</code> with <code>\ifcsundef</code> ..	53
<code>\@ifundefined</code> with <code>\ifcsundef</code> ..	28	<code>\ifglentryexists</code> : replaced	
<code>\gls@defglossaryentry</code> : added		<code>\@ifundefined</code> with <code>\ifcsundef</code> ..	53
<code>\@gls@defsort</code> .....	85	<code>\istfile</code> : deprecated .....	177
added short and long keys .....	81	<code>glossaryentry</code> : new .....	205
replaced <code>\@ifundefined</code> with		<code>glossarysubentry</code> : new .....	206
<code>\ifcsundef</code> .....	82	<code>\newglossaryentry</code> : replaced	
<code>\gls@doclearpage</code> : replaced		<code>\DeclareRobustCommand</code> with	
<code>\@ifundefined</code> with <code>\ifcsundef</code> ..	42	<code>\newrobustcmd</code> .....	70

<code>\newglossarystyle:replaced</code>		<code>\showglouseriii:new</code>	255
<code>\@ifundefined with \ifcsundef</code>	214	<code>\showglouseriv:new</code>	256
<code>\ns@newglossary:added</code>		<code>\showglouserv:new</code>	256
<code>\@gls@defsortcount</code>	60	<code>\showglouservi:new</code>	256
replaced <code>\@ifundefined with</code>		<code>subentrycounter:new</code>	11
<code>\ifcsundef</code>	60	<code>\writeist:added xindy-only macro</code>	
<code>entrycounter:new</code>	11	definitions to glossary open tag	162
<code>\oldacronym:replaced \@ifundefined</code>		modified to support new format	160
with <code>\ifcsundef</code>	217	3.01 (2011-04-12)	
<code>compatible-2.07:compatible-2.07</code>		<code>\@glswritefiles:added check for</code>	
option added	29	empty glossaries	177
<code>long:new</code>	67	<code>General:made robust</code>	124
<code>longplural:new</code>	67	<code>\ACRfull:made robust</code>	220
<code>nonumberlist:now boolean</code>	65	<code>\Acrfull:made robust</code>	219
<code>sort:new</code>	11	<code>\acrfull:made robust</code>	219
<code>counter:replaced \@ifundefined with</code>		<code>\acrfullformat:removed</code>	
<code>\ifcsundef</code>	64	<code>\acronymfont</code> as it should already be	
<code>counterwithin:new</code>	11	set in the second argument.	219
<code>\printglossary:replaced</code>		<code>\ACRfullpl:made robust</code>	221
<code>\@ifundefined with \ifcsundef</code>	188	<code>\Acrfullpl:made robust</code>	221
<code>\SetDescriptionFootnoteAcronymDisplayStyle</code>		<code>\acrfullpl:made robust</code>	220
expanded options link options	238	<code>\ACRlong:made robust</code>	145
<code>\setentrycounter:added optional</code>		<code>\Acrlong:made robust</code>	144
argument	212	<code>\acrlong:made robust</code>	144
<code>\showacronymlists:new</code>	258	<code>\ACRlongpl:made robust</code>	147
<code>\showglocounter:new</code>	255	<code>\Acrlongpl:made robust</code>	146
<code>\showgloDESC:new</code>	256	<code>\acrlongpl:made robust</code>	146
<code>\showgloDESCplural:new</code>	256	<code>\ACRshort:made robust</code>	141
<code>\showglofirst:new</code>	254	<code>\Acrshort:made robust</code>	141
<code>\showglofirstpl:new</code>	255	<code>\acrshort:made robust</code>	140
<code>\showgloflag:new</code>	258	<code>\ACRshortpl:made robust</code>	143
<code>\showgloindex:new</code>	257	<code>\Acrshortpl:made robust</code>	142
<code>\showglolevel:new</code>	254	<code>\acrshortpl:made robust</code>	142
<code>\showglongame:new</code>	256	<code>\Gls:made robust</code>	123
<code>\showgloparent:new</code>	254	<code>\glsadd:made robust</code>	157
<code>\showgloplural:new</code>	254	<code>\glsaddall:made robust</code>	157
<code>\showglosort:new</code>	257	<code>\GLSdesc:made robust</code>	132
<code>\showglossaries:new</code>	258	<code>\Glsdesc:made robust</code>	132
<code>\showglossarycounter:new</code>	259	<code>\glsdesc:made robust</code>	132
<code>\showglossaryentries:new</code>	259	<code>\GLSdescplural:made robust</code>	133
<code>\showglossaryin:new</code>	258	<code>\Glsdescplural:made robust</code>	133
<code>\showglossaryout:new</code>	259	<code>\glsdescplural:made robust</code>	133
<code>\showglossarytitle:new</code>	259	<code>\glsfirst:made robust</code>	129
<code>\showglosymbol:new</code>	257	<code>\GLSfirstplural:made robust</code>	131
<code>\showglosymbolplural:new</code>	257	<code>\Glsfirstplural:made robust</code>	131
<code>\showglotext:new</code>	254	<code>\glsfirstplural:made robust</code>	130
<code>\showglotype:new</code>	255	<code>\glslink:made robust</code>	109
<code>\showglouserI:new</code>	255	<code>\GLSname:made robust</code>	132
<code>\showglouserII:new</code>	255	<code>\Glsname:made robust</code>	131

\glsname: made robust .....	131	\@printglossary: add a way to fetch	
\GLSpl: made robust .....	126	current entry label .....	190
\Glspl: made robust .....	125	savenumberlist: new .....	9
\glspl: made robust .....	124	ucmark: new .....	11
\GLSplural: made robust .....	130	\gls@defglossaryentry: added	
\GLSsymbol: made robust .....	134	numberlist element .....	84
\Glsymbol: made robust .....	134	\gls@save@numberlist: new .....	188
\glssymbol: made robust .....	134	\gls@wrglossary: added check for	
\GLSsymbolplural: made robust .....	135	glossary file defined .....	179
\Glsymbolplural: made robust .....	135	\glsdisplaynumberlist: new .....	155
\glssymbolplural: made robust .....	134	\glsentrycounter: set default value ..	112
\Glstext: made robust .....	128	\Glsentryfull: fixed bug (replaced	
\glstext: made robust .....	128	\glsentryshortpl with	
\GLSuseri: made robust .....	136	\glsentryshort) .....	154
\Glsuseri: made robust .....	136	\glsentryfullpl: fixed bug (replaced	
\glsuseri: made robust .....	135	\glsentryshort with	
\GLSuserii: made robust .....	137	\glsentryshortpl) .....	155
\Glsuserii: made robust .....	136	\glsentrynumberlist: new .....	155
\glsuserii: made robust .....	136	\glsmoveentry: new .....	86
\GLSuseriii: made robust .....	137	\glsresetsubentrycounter: new ...	206
\Glsuseriii: made robust .....	137	\ifglshaschildren: new .....	55
\glsuseriii: made robust .....	137	\ifglshasparent: new .....	55
\GLSuseriv: made robust .....	138	\makeglossaries: added list parser ..	172
\Glsuseriv: made robust .....	138	indexonlyfirst: new .....	26
\glsuseriv: made robust .....	138	\renewglossarystyle: new .....	214
\GLSuserv: made robust .....	139	\showglossaryentries: fixed misspelt	
\Glsuserv: made robust .....	139	command .....	259
\glsuserv: made robust .....	138	\SmallNewAcronymDef: fixed broken	
\GLSuservi: made robust .....	140	short and long plural .....	248
\Glsuservi: made robust .....	140	3.03 (2012/09/21)	
\glsuservi: made robust .....	139	\@gls@sanitizesort: new .....	21
3.02 (2012-05-19)		\@gls@setupsort@standard: used	
\glsnumlistlastsep: new .....	156	\@gls@sanitizesort .....	12
\glsnumlistsep: new .....	156	\@printglossary: allow title to override	
3.02 (2012-05-21)		default toctitle .....	189
\@do@wrglossary: changed		General: allow title to set toctitle .....	202
\@glslocref to		\glsinlinedescformat: new .....	272
\theglsentrycounter .....	185	\glsinlineemptydescformat: new ..	272
\@do@wrglossary: changed		\glsinlinenameformat: new .....	272
\@do@wr@glossary to test for		\glsinlinepostchild: new .....	272
indexonlyfirst option; put old		\glsinlinesubdescformat: new ....	272
\@do@wr@glossary code into		\glsinlinesubnameformat: new ....	272
\@do@wrglossary .....	180	\glspostinline: replaced “.” with	
\@gls@missingnumberlist: new .....	68	\glspostdescription .....	272
\@glswritefiles: added check for		list: added check for glsnogroupskip ..	274
existence of token in case		altlongragged4col: added check for	
\makeglossaries has been		glsnogroupskip .....	291
omitted .....	177	altsuperragged4col: added check for	
		glsnogroupskip .....	310

alttree: added check for glsnogroupskip .....	319	\gls@disablepagerefexpansion: new	180
index: added check for glsnogroupskip	313	\gls@numberpage: new .....	180
nogroupskip: new .....	11	\gls@protected@pagefmts: new ....	180
long: added check for glsnogroupskip	277	\gls@romanpage: new .....	180
long3col: added check for glsnogroupskip .....	279	\glsdefmain: added check for doc package .....	15
long4col: added check for glsnogroupskip .....	280	\glsorg@endtheglossary: new .....	5
longragged: added check for glsnogroupskip .....	288	\glsorg@theglossary: new .....	5
longragged3col: added check for glsnogroupskip .....	289	\PrintChanges: new .....	5
nopostdot: new .....	11	3.05 (2013-04-21)	
tree: added check for glsnogroupskip	314	\@do@esc@wrglossary: add Roman case. Fixed bugs in the else statements .....	183
treenoname: added check for glsnogroupskip .....	316	\@gls@link: added check for “nohypertypes” .....	111
super: added check for glsnogroupskip	299	mcocalttree: replaced ‘2’ with \glsmcols .....	297
super3col: added check for glsnogroupskip .....	301	mcocalindex: replaced ‘2’ with \glsmcols	293
super4col: added check for glsnogroupskip .....	303	mcocalindexspannav: replaced ‘2’ with \glsmcols .....	294
superragged: added check for glsnogroupskip .....	306	mcocaltree: replaced ‘2’ with \glsmcols	294
superragged3col: added check for glsnogroupskip .....	308	mcocaltreenoname: replaced ‘2’ with \glsmcols .....	296
3.04 (2012-11-11)		mcocaltreesspannav: replaced ‘2’ with \glsmcols .....	295
altlist: replaced \newline with paragraph break .....	274	\gls@protected@pagefmts: added Roman to list .....	180
3.04 (2012-11-18)		\gls@Romanpage: new .....	180
\@do@wrglossary: changed \theglsentrycounter back to \@glslocref .....	185	\glsgetgrouplabel: fixed bug (typo in \equal) .....	212
\@do@esc@wrglossary: modified to compensate for possible incorrect page number .....	183	\nopostdesc: made robust .....	36
\@gls@escbsdq: unsanitize \gls@numberpage, \gls@alphpage, \gls@Alphpage and \gls@romanpage .....	114	3.05 (2013/04/21)	
\@print@glossary: Moved aux write to end of document to prevent unwanted whatsit occurring here. . .	191	\@gls@nohyperlist: new .....	18
General: Added check for doc package added datatool-base as a required package .....	4	\GlsDeclareNoHyperList: new .....	18
added local key .....	108	nohypertypes: new .....	18
\gls@Alphpage: new .....	180	3.06 (2013/06/17)	
\gls@alphpage: new .....	180	\@xdy@main@language: Changed back to using \language name .....	28
		\findrootlanguage: Obsoleted .....	50
		3.07 (2013-07-05)	
		\@gls@link: fixed bug that failed to find entry in list .....	111
		\glossarypreamble: modified to work with \setglossarypreamble .....	39
		\gls@docclearpage: added check for openright .....	42
		\glspostdescription: Added spacefactor code .....	10

\GlsSetXdyCodePage: Added check for fontspec .....	51	\glseelist: made robust .....	187
\SetDescriptionAcronymDisplayStyle: now using \glsdoparenifnotempty	242	\ifglshasdesc: new .....	55
\setglossarypreamble: new .....	39	\ifglshassymbol: new .....	56
3.08a (2013-08-30)		altlongragged4col: updated to use \glossentry and \subglossentry	291
list: updated list style to use \glossentry and \subglossentry	273	alttree: updated to use \glossentry and \subglossentry .....	318
listdotted: updated listdotted style to use \glossentry and \subglossentry .....	275	index: added paragraph break at end of environment .....	312
altlist: updated altlist style to use \glossentry and \subglossentry	274	updated to use \glossentry and \subglossentry .....	312
inline: updated inline style to use \glossentry and \subglossentry	271	long: updated to use \glossentry and \subglossentry .....	277
3.08a (2013-09-28)		longragged: updated to use \glossentry and \subglossentry	288
\@glo@storeentry: no longer need to check for special characters in any of the fields other than sort .....	87	longragged3col: updated to use \glossentry and \subglossentry	289
updated for \glossentry .....	87	tree: updated to use \glossentry and \subglossentry .....	314
\@glossaryentryfield: switched to \glossentry .....	86	\setglossarystyle: new .....	213
\@glossarysubentryfield: switched to \subglossentry .....	87	\setglossentrycompatibility: new	210
General: added nogroupskip key to \printglossary .....	203	superragged: updated to use \glossentry and \subglossentry	306
removed definition of \@glossaryentryfield .....	361	3.09a (2013-10-09)	
removed definition of \@glossarysubentryfield .....	361	\@gls@assign@symbolplural@field: new .....	21
\compatibleglossentry: new .....	208	\@gls@default@value: new .....	64
\compatiblesubglossentry: new ...	210	\Glsentrydesc: made robust .....	150
\glossaryentryfield: deprecated ...	210	\Glsentrydescplural: made robust ..	150
\Glossentrydesc: new .....	209	\Glsentryfirst: made robust .....	151
\glossentrydesc: new .....	209	\Glsentryfirstplural: made robust .	151
\Glossentryname: new .....	209	\Glsentryfull: made robust .....	154
\glossentryname: new .....	209	\Glsentryfullpl: made robust .....	155
\Glossentrysymbol: new .....	209	\Glsentrylong: made robust .....	154
\glossentrysymbol: new .....	209	\Glsentrylongpl: made robust .....	154
\gls@assign@desc@field: new .....	20	\Glsentryname: made robust .....	149
\gls@assign@descplural@field: new	20	\Glsentryplural: made robust .....	150
\gls@assign@field: new .....	70	\Glsentryshort: made robust .....	154
\gls@ifnotmeasuring: new .....	88	\Glsentryshortpl: made robust .....	154
\glsaddallunused: new .....	157	\Glsentrysymbol: made robust .....	151
\glsexpandfields: new .....	70	\Glsentrysymbolplural: made robust	151
\glsnoexpandfields: new .....	70	\Glsentrytext: made robust .....	150
\glssee: made robust .....	187	\Glsentryuseri: made robust .....	152
\glsseeformat: made robust .....	187	\Glsentryuserii: made robust .....	153
\glsseeitem: made robust .....	188	\Glsentryuseriii: made robust .....	153
		\Glsentryuseriv: made robust .....	153
		\Glsentryuserv: made robust .....	153
		\Glsentryuservi: made robust .....	153

\glstextup: new .....	218	\@Gls@: add \glsifplural,	
\ifglshassymbol: changed test to check		\glscapscase, \glscustomtext and	
for \@gls@default@symbol .....	56	\glsinsert .....	123
3.10a (2013-09-28)		change to using \glentryfmt style	
\gls@assign@type@field: new .....	20	commands .....	123
3.10a (2013-10-13)		removed \makefirstuc (now dealt	
\@gls@keymap: new .....	72	with in \glentryfmt) .....	123
\@gls@provide@newglossary: new ...	58	\@Glspl@: add \glsifplural,	
\@gls@writedef: new .....	71	\glscapscase, \glscustomtext and	
\@glsdefaultplural: Obsolete .....	67	\glsinsert .....	125
\@glsnodesc: new .....	67	change to using \glentryfmt style	
\@print@glossary: Added		commands .....	126
providecommand code to aux file ..	192	removed \makefirstuc (now dealt	
\gls@defglossaryentry: Changed to		with in \glentryfmt) .....	126
using \@gls@default@value .....	81	\@acrlong: added \glslabel,	
new .....	80	\glsifplural, \glscapscase,	
\gls.writedefhook: new .....	79	\glsinsert and \glscustomtext	360
\makeglossaries: Added		\@acrshort: added \glslabel,	
providecommand code to aux file ..	171	\glsifplural, \glscapscase,	
\new@glossaryentry: new .....	71	\glsinsert and \glscustomtext	359
\ns@newglossary: added		\@gls@: add \glslabel, \glsifplural,	
\@gls@provide@newglossary ....	60	\glscapscase, \glscustomtext and	
3.11a (2013-10-15)		\glsinsert .....	122
\@ACRlong: added \glslabel,		change to using \glentryfmt style	
\glsifplural, \glscapscase,		commands .....	122
\glsinsert and \glscustomtext	361	\@gls@noexpand@fields: Fixed bug	
\@ACRshort: added \glslabel,		expand replaced with noexpand ....	68
\glsifplural, \glscapscase,		\@glsdisp: add \glslabel,	
\glsinsert and \glscustomtext	359	\glsifplural, \glscapscase,	
\@Acrlong: added \glslabel,		\glscustomtext and \glsinsert	127
\glsifplural, \glscapscase,		change to using \glentryfmt style	
\glsinsert and \glscustomtext	360	commands .....	127
\@Acrshort: added \glslabel,		\@Glspl@: add \glslabel,	
\glsifplural, \glscapscase,		\glsifplural, \glscapscase,	
\glsinsert and \glscustomtext	359	\glscustomtext and \glsinsert	125
\@GLS@: add \glslabel, \glsifplural,		change to using \glentryfmt style	
\glscapscase, \glscustomtext and		commands .....	125
\glsinsert .....	124	General: added \glslabel,	
change to using \glentryfmt style		\glsifplural, \glscapscase,	
commands .....	124	\glsinsert and	
removed \MakeUppercase (now		\glscustomtext .....	140–147
moved to \glentryfmt) .....	124	changed to just use	
\@GLSpl: add \glslabel,		\Glsentrydescplural .....	133
\glsifplural, \glscapscase,		changed to just use	
\glscustomtext and \glsinsert	126	\glentrydescplural .....	133
change to using \glentryfmt style		changed to just use \Glsentrydesc .	132
commands .....	126	changed to just use	
removed \MakeUppercase as now		\glentrydesc .....	132, 133
dealt with in \glentryfmt .....	126		

changed to just use		<code>\glsdisplay</code> : obsoleted	106
<code>\Glsentryfirstplural</code>	131	<code>\glsdisplayfirst</code> : obsoleted	106
changed to just use		<code>\glsgenentryfmt</code> : new	101
<code>\glsentryfirstplural</code>	130, 131	<code>\glsgetgrouptitle</code> : Added check in	
changed to just use <code>\Glsentryfirst</code>	129	case non-Latin alphabet in use	211
changed to just use <code>\glsentryfirst</code>	129	<code>\gls glossarymark</code> : replaced	
changed to just use <code>\Glsentryname</code>	132	<code>\MakeUppercase</code> with	
changed to just use		<code>\mfirstucMakeUppercase</code>	40
<code>\glsentryname</code>	131, 132	<code>\glsnavigation</code> : switched to using	
changed to just use <code>\Glsentryplural</code>	130	<code>\@gls@getgrouptitle</code>	270
changed to just use <code>\glsentryplural</code>	130	<code>\ifglshasdesc</code> : replaced <code>\ifdefempty</code>	
changed to just use		with <code>\ifcsempy</code>	55
<code>\Glsentrysymbolplural</code>	135	<code>\ifglshaslong</code> : new	56
changed to just use		<code>\ifglshasshort</code> : new	56
<code>\glsentrysymbolplural</code>	135	<code>\ifglshassymbol</code> : replaced	
changed to just use <code>\Glsentrysymbol</code>	134	<code>\ifdefempty</code> with <code>\ifcsempy</code>	56
changed to just use <code>\glsentrysymbol</code>	134	<code>\ifglused</code> : replaced <code>\ifthenelse</code> with	
Changed to just use <code>\Glsentrytext</code>	129	<code>\ifbool</code>	53
changed to just use <code>\glsentrytext</code>	128	<code>\longnewglossaryentry</code> : new	79
changed to just use		<code>\ns@newglossary</code> : replaced	
<code>\Glsentryuseriii</code>	137	<code>\glsdisplay</code> and	
changed to just use		<code>\glsdisplayfirst</code> with	
<code>\glsentryuseriii</code>	137, 138	<code>\glsentryfmt</code>	60
changed to just use <code>\Glsentryuserii</code>	136	compatible-3.07: <code>cnew</code>	29
changed to just use		<code>\SetCustomDisplayStyle</code> : updated to	
<code>\glsentryuserii</code>	136, 137	use <code>\defglsglsentryfmt</code>	252
changed to just use <code>\Glsentryuseriv</code>	138	<code>\SetDefaultAcronymDisplayStyle</code> :	
changed to just use <code>\glsentryuseriv</code>	138	changed to use <code>\defglsglsentryfmt</code>	236
changed to just use <code>\Glsentryuseri</code>	136	<code>\SetDescriptionAcronymDisplayStyle</code> :	
changed to just use		updated to use <code>\defglsglsentryfmt</code>	242
<code>\glsentryuseri</code>	135, 136	<code>\SetDescriptionDUAAcronymDisplayStyle</code> :	
changed to just use <code>\Glsentryuservi</code>	140	updated to use <code>\defglsglsentryfmt</code>	240
changed to just use		<code>\SetDescriptionFootnoteAcronymDisplayStyle</code> :	
<code>\glsentryuservi</code>	139, 140	updated to use <code>\defglsglsentryfmt</code>	238
changed to just use <code>\Glsentryuserv</code>	139	<code>\SetDUADisplayStyle</code> : updated to use	
changed to just use <code>\glsentryuserv</code>	139	<code>\defglsglsentryfmt</code>	250
Now requires <code>textcase</code>	4	<code>\SetFootnoteAcronymDisplayStyle</code> :	
acronymlists: replaced		updated to use <code>\defglsglsentryfmt</code>	244
<code>\@addtoacronymlists</code> with		<code>\SetSmallAcronymDisplayStyle</code> :	
<code>\DeclareAcronymList</code>	18	updated to use <code>\defglsglsentryfmt</code>	247
<code>\defglsglsdisplay</code> : obsoleted	106	<code>\setupglossaries</code> : new	31
<code>\defglsglsdisplayfirst</code> : obsoleted	107	<code>\showglo long</code> : new	257
<code>\defglsglsentryfmt</code> : new	59	<code>\showglo short</code> : new	257
<code>\forglsglsentries</code> : replaced <code>\ifx</code> with		numbers: new	30
<code>\ifdefempty</code>	52	symbols: new	30
<code>\gls@assign@desc</code> : new	79	3.12a (2013-10-16)	
<code>\gls@defglossaryentry</code> : Fixed default		<code>\gls@defglossaryentry</code> : added	
counter if none supplied	84	<code>\glslabel</code>	80
<code>\gls@doentryfmt</code> : new	59	<code>\glsaddkey</code> : new	74



3.13a (2013-11-05)		long: switched to \tabularnewline ..	277
\@gls@assign@symbol@field: changed to use \glsetnoexpandfield ....	21	long3col: switched to \tabularnewline .....	278
\@gls@assign@symbolplural@field: changed to use \glsetnoexpandfield .....	21	long3colheader: switched to \tabularnewline .....	279
\@gls@link: removed \relax .....	111	long3colheaderborder: switched to \tabularnewline .....	279
\@gls@notranslatorhook: new .....	24	long4col: switched to \tabularnewline .....	280
\@gls@setupsort@standard: moved \@gls@santizesort to \glsprestandardsort .....	12	long4colheader: switched to \tabularnewline .....	280
ucmark: added check for memoir .....	11	longheader: switched to \tabularnewline .....	278
see: added \gls@checkseeallowed ...	64	longheaderborder: switched to \tabularnewline .....	278
\glossarysection: changed \glossarymark to \glsglossarymark .....	40	\SetFootnoteAcronymDisplayStyle: fixed missing argument bug .....	245
\glossarystyle: fixed bug caused by using \ifdef instead of \ifcsdef ..	213	super: switched to \tabularnewline ..	299
\gls@assign@desc@field: changed to use \glsetnoexpandfield .....	20	super3col: switched to \tabularnewline .....	301
\gls@assign@descplural@field: changed to use \glsetnoexpandfield .....	20	super3colheader: switched to \tabularnewline .....	301
\gls@assign@name@field: changed to use \glsetnoexpandfield .....	20	super4col: switched to \tabularnewline .....	302
\gls@assign@type@field: changed to use \glsetexpandfield .....	20	super4colheader: switched to \tabularnewline .....	303
\gls@checkseeallowed: new .....	65	super4colheaderborder: switched to \tabularnewline .....	303
\glsaddallunused: set default to \@glo@types .....	157	superheader: switched to \tabularnewline .....	300
\Glsentryfull: changed to use \acrfullformat .....	154	superheaderborder: switched to \tabularnewline .....	300
\glentryfull: changed to use \acrfullformat .....	154		
\Glsentryfullpl: changed to use \acrfullformat .....	155	3.14a (2013-11-12)	
\glentryfullpl: changed to use \acrfullformat .....	155	\@glswritefiles: renamed \glswritefiles to \@glswritefiles and used “savewrites” option to set \glswritefiles .....	177
\glsglossarymark: renamed \glossarymark to \glsglossarymark to avoid conflict with memoir .....	40	General: new .....	261
\glsprestandardsort: new .....	12	acronyms: new .....	16
\glsetexpandfield: new .....	20	\gls@defglossaryentry: added check for existence of default glossary ....	81
\glsetnoexpandfield: new .....	20	set the default for firstplural to be the value of plural .....	84
altsuper4colheader: switched to \tabularnewline .....	304	xindygloss: new .....	28
altsuper4colheaderborder: switched to \tabularnewline .....	305	\longprovideglossaryentry: new ...	80
		compatible-2.07: added check for 2.07 before setting 3.07 compatibility ....	29
		notranslate: new .....	25

\provideglossaryentry:new .....	70	index:new .....	30
4.0 (2013-11-14)		\newacronymstyle:new .....	224
\gls@defglossaryentry: added check		long-sc-short:new .....	227
for first key .....	84	long-sc-short-desc:new .....	228
super: fixed typo in \subglossentry		long-short:new .....	225
(\glossentrydesc) .....	299	long-short-desc:new .....	228
4.01 (2013-11-16)		long-sm-short:new .....	227
General: fixed non-value options so that		long-sm-short-desc:new .....	229
they can be passed to document class .	8	long-sp-short-desc:new .....	228
\CustomAcronymFields: inserted		footnote:new .....	232
missing comma .....	252	footnote-desc:new .....	234
4.02 (2013-12-05)		footnote-sc:new .....	233
\@acrfull: now using \acrfullfmt ..	219	footnote-sc-desc:new .....	234
\@gls@indexdef:new .....	30	footnote-sm:new .....	234
\@gls@numbersdef:new .....	30	footnote-sm-desc:new .....	234
\@gls@symbolsdef:new .....	30	\setacronymstyle:new .....	224
General: Removed \acronymfont .	144–147	\SetDescriptionAcronymDisplayStyle:	
\ACRfullfmt:new .....	220	Moved check for empty custom text to	
\Acrfullfmt:new .....	220	prevent unwanted parenthetical	
\acrfullfmt:new .....	219	material .....	242
\ACRfullplfmt:new .....	221	\SetDescriptionFootnoteAcronymDisplayStyle:	
\Acrfullplfmt:new .....	221	Moved check for empty custom text to	
\acrfullplfmt:new .....	220	prevent unwanted parenthetical	
\acronymentry:new .....	223	material .....	238
sanitize: fixed bug that caused an error		\SetFootnoteAcronymDisplayStyle:	
here .....	24	Moved check for empty custom text to	
sc-short-long:new .....	227	prevent unwanted parenthetical	
sc-short-long-desc:new .....	229	material .....	244
\Genacrfullformat:new .....	105	\SetGenericNewAcronym:new .....	222
\genacrfullformat:new .....	105	\SetSmallAcronymDisplayStyle:	
\GenericAcronymFields:new .....	223	Moved check for empty custom text to	
\Genplacrfullformat:new .....	106	prevent unwanted parenthetical	
\genplacrfullformat:new .....	106	material .....	247
\Glsentryfull: bug fix: added missing		dua:new .....	230
\acronymfont .....	154	dua-desc:new .....	232
\glsentryfull: bug fix: added missing		numberedsection: added nameref	
\acronymfont .....	154	option .....	8
\Glsentryfullpl: bug fix: added		4.02 (2013-13-05)	
missing \acronymfont .....	155	\makeglossaries: made preamble only	173
\glsentryfullpl: bug fix: added		4.03 (2014-01-17)	
missing \acronymfont .....	155	General: changed default to \@empty	
\glsngenacfmt:new .....	103	instead of \relax .....	29
\GlsUseAcrEntryDisplayStyle:new ...	225	4.03 (2014-01-20)	
\GlsUseAcrStyleDefs:new .....	225	\@do@esc@wrglossary: added	
short-long:new .....	226	\glsdetoklabel .....	184
short-long-desc:new .....	229	\@do@noesc@wrglossary: added	
xindynoglsnumbers:new .....	28	\glsdetoklabel .....	182
sm-short-long:new .....	227	\@ACRlong: removed \glslabel	
sm-short-long-desc:new .....	229	(defined in \@gls@link) .....	361

\@ACRshort: removed \glslabel (defined in \@gls@link) .....	359	\Genplacrfullformat: redefined to use accessibility information .....	358
\@Acrlong: removed \glslabel (defined in \@gls@link) .....	360	\genplacrfullformat: redefined to use accessibility information .....	358
\@Acrshort: removed \glslabel (defined in \@gls@link) .....	359	\glossentryname: added \glsdetoklabel .....	209
\@GLS@: removed \glslabel (defined in \@gls@link) .....	124	\gls@defglossaryentry: added \glsdetoklabel .....	80
\@GLSpl: removed \glslabel (defined in \@gls@link) .....	126	replaced #1 with \@glo@label .....	82
\@Gls@: removed \glslabel (defined in \@gls@link) .....	123	replaced \ifthenelse with \ifdefequal .....	82
\@Gls@entry@field: new .....	148	\glsadd: added \glsdetoklabel ....	157
\@Glspl@: removed \glslabel (defined in \@gls@link) .....	125	\glsaddkey: switched to using \@gls@field@link .....	75
\@acrlong: removed \glslabel (defined in \@gls@link) .....	360	\glsdetoklabel: new .....	53
\@acrshort: removed \glslabel (defined in \@gls@link) .....	359	\glsdisplaynumberlist: added \glsdetoklabel .....	155
\@gls@: removed \glslabel (defined in \@gls@link) .....	122	\glsdoifexistsorwarn: new .....	54
\@gls@access@display: new .....	347	\glsentryaccess: switched to using \@gls@entry@field .....	345
\@gls@entry@field: new .....	148	\glsentrydescaccess: switched to using \@gls@entry@field .....	346
\@gls@fetchfield: new .....	73	\glsentrydescpluralaccess: switched to using \@gls@entry@field .....	346
\@gls@field@link: new .....	128	\glsentryfirstaccess: switched to using \@gls@entry@field .....	346
\@gls@link: added \glsdetoklabel .	111	\glsentryfirstplural: added \glsdetoklabel .....	151
moved \@gls@link@opts and \@gls@link@label to \@gls@link	111	\glsentrylongaccess: switched to using \@gls@entry@field .....	347
\@gls@writedef: added \glsdetoklabel .....	71	\glsentrylongpluralaccess: switched to using \@gls@entry@field .....	347
\@glsdisp: removed \glslabel (defined in \@gls@link) .....	127	\glsentrypluralaccess: switched to using \@gls@entry@field .....	346
\@Glspl@: removed \glslabel (defined in \@gls@link) .....	125	\glsentryshortaccess: switched to using \@gls@entry@field .....	346
\@printglossary: added \glsdetoklabel .....	190	\glsentryshortpluralaccess: switched to using \@gls@entry@field .....	346
General: removed \glslabel (defined in \@gls@link) .....	140	\glsentrysymbolaccess: switched to using \@gls@entry@field .....	346
sc-short-long-desc: redefined to use accessibility information .....	365	\glsentrysymbolpluralaccess: switched to using \@gls@entry@field .....	346
\compatibleglossentry: added \glsdetoklabel .....	341	\glsentrytextaccess: switched to using \@gls@entry@field .....	346
\compatiblesubglossentry: added \glsdetoklabel .....	342	\glsngenacfmt: redefined to use accessibility information .....	356
\Genacrfullformat: redefined to use accessibility information .....	358		
\genacrfullformat: redefined to use accessibility information .....	358		

<code>\glsgenentryfmt</code> : redefined to use accessibility information . . . . .	353	<code>sm-short-long-desc</code> : redefined to use accessibility information . . . . .	365
<code>\glshyperlink</code> : added <code>\glsdetoklabel</code> . . . . .	156	<code>long-sc-short-desc</code> : redefined to use accessibility information . . . . .	364
<code>\glslocalreset</code> : added <code>\glsdetoklabel</code> . . . . .	89	<code>long-short</code> : redefined to use accessibility information . . . . .	362
<code>\glslocalunset</code> : added <code>\glsdetoklabel</code> . . . . .	90	<code>long-short-desc</code> : redefined to use accessibility information . . . . .	364
<code>\glsmoveentry</code> : added <code>\glsdetoklabel</code> . . . . .	86	<code>long-sm-short-desc</code> : redefined to use accessibility information . . . . .	364
replaced <code>\ifthenelse</code> with <code>\ifdefequal</code> . . . . .	86	<code>footnote</code> : redefined to use accessibility information . . . . .	368
<code>\glsrefentry</code> : added <code>\glsdetoklabel</code>	207	<code>footnote-desc</code> : redefined to use accessibility information . . . . .	370
<code>\glsreset</code> : added <code>\glsdetoklabel</code> . . .	89	<code>footnote-sc</code> : redefined to use accessibility information . . . . .	370
<code>\glsseelist</code> : added <code>\expandafter</code> commands . . . . .	187	<code>footnote-sc-desc</code> : redefined to use accessibility information . . . . .	371
<code>\glsstepentry</code> : added <code>\glsdetoklabel</code> . . . . .	206	<code>footnote-sm</code> : redefined to use accessibility information . . . . .	370
<code>\glsstepsubentry</code> : added <code>\glsdetoklabel</code> . . . . .	206	<code>footnote-sm-desc</code> : redefined to use accessibility information . . . . .	371
<code>\glsunset</code> : added <code>\glsdetoklabel</code> . . .	89	<code>\renewacronymstyle</code> : new . . . . .	224
<code>short-long</code> : commented spurious EOL	227	<code>\showglocounter</code> : added <code>\glsdetoklabel</code> . . . . .	255
redefined to use accessibility information . . . . .	363	<code>\showglodesc</code> : added <code>\glsdetoklabel</code>	256
<code>short-long-desc</code> : redefined to use accessibility information . . . . .	365	<code>\showglodescaccess</code> : added <code>\glsdetoklabel</code> . . . . .	377
<code>\ifglsdescsuppressed</code> : added <code>\glsdetoklabel</code> . . . . .	56	<code>\showglodescplural</code> : added <code>\glsdetoklabel</code> . . . . .	257
fixed typo . . . . .	56	<code>\showglodescpluralaccess</code> : added <code>\glsdetoklabel</code> . . . . .	377
<code>\ifglsentryexists</code> : added <code>\glsdetoklabel</code> . . . . .	53	<code>\showglofirst</code> : added <code>\glsdetoklabel</code> . . . . .	255
<code>\ifglshaschildren</code> : added <code>\glsdetoklabel</code> . . . . .	55	<code>\showglofirstaccess</code> : added <code>\glsdetoklabel</code> . . . . .	377
<code>\ifglshasdesc</code> : added <code>\glsdetoklabel</code> . . . . .	55	<code>\showglofirstpl</code> : added <code>\glsdetoklabel</code> . . . . .	255
<code>\ifglshasfield</code> : new . . . . .	57	<code>\showglofirstpluralaccess</code> : added <code>\glsdetoklabel</code> . . . . .	377
<code>\ifglshaslong</code> : added <code>\glsdetoklabel</code> . . . . .	56	<code>\showgloflag</code> : added <code>\glsdetoklabel</code>	258
<code>\ifglshasparent</code> : added <code>\glsdetoklabel</code> . . . . .	55	<code>\showgloindex</code> : added <code>\glsdetoklabel</code> . . . . .	258
<code>\ifglshasshort</code> : added <code>\glsdetoklabel</code> . . . . .	56	<code>\showglolevel</code> : added <code>\glsdetoklabel</code> . . . . .	254
<code>\ifglshassymbol</code> : added <code>\glsdetoklabel</code> . . . . .	56	<code>\showglolong</code> : added <code>\glsdetoklabel</code>	257
replaced <code>\ifcempty</code> with <code>\ifdefempty</code> and replaced <code>\ifx</code> with <code>\ifdefequal</code> . . . . .	56	<code>\showglolongaccess</code> : added <code>\glsdetoklabel</code> . . . . .	378
<code>\ifglshused</code> : added <code>\glsdetoklabel</code> . .	53		

\showglolongpluralaccess: added		redefined to use accessibility	
\glsdetoklabel .....	378	information .....	368
\showglongname: added \glsdetoklabel	256	4.04 (2014-03-04)	
\showglongnameaccess: added		\@gls@getcounterprefix: added	
\glsdetoklabel .....	377	warning if no prefix can be formed .	186
\showgloparent: added		4.04 (2014-03-06)	
\glsdetoklabel .....	254	\@gls@noidx@nosanitizesort: new .	21
\showgloplural: added		\@gls@noidx@sanitizesort: new ...	21
\glsdetoklabel .....	254	\@gls@nosanitizesort: new .....	21
\showglopluralaccess: added		\@gls@sanitizesort: new .....	21
\glsdetoklabel .....	377	\@glo@addchildren: new .....	193
\showgloshort: added		\@glo@do@sortentries: new .....	194
\glsdetoklabel .....	257	\@glo@grabfirst: new .....	199
\showgloshortaccess: added		\@glo@sortedinsert: new .....	194
\glsdetoklabel .....	378	\@glo@sortentries: new .....	193
\showgloshortpluralaccess: added		\@glo@sorthandler@case: new .....	195
\glsdetoklabel .....	378	\@glo@sorthandler@letter: new ...	195
\showglosort: added \glsdetoklabel	257	\@glo@sorthandler@nocase: new ...	195
\showglosymbol: added		\@glo@sorthandler@word: new .....	194
\glsdetoklabel .....	257	\@glo@sortmacro@case: new .....	196
\showglosymbolaccess: added		\@glo@sortmacro@def: new .....	197
\glsdetoklabel .....	377	\@glo@sortmacro@def@do: new .....	197
\showglosymbolplural: added		\@glo@sortmacro@letter: new .....	196
\glsdetoklabel .....	257	\@glo@sortmacro@nocase: new .....	197
\showglosymbolpluralaccess: added		\@glo@sortmacro@standard: new ...	196
\glsdetoklabel .....	377	\@glo@sortmacro@use: new .....	198
\showglotext: added \glsdetoklabel	254	\@glo@sortmacro@word: new .....	195
\showglotextaccess: added		\@gls@noidx@do: new .....	199
\glsdetoklabel .....	377	\@gls@noidx@getgrouptitle: new ..	212
\showglotype: added \glsdetoklabel	255	\@gls@noref@warn: new .....	176
\showglouseri: added		\@gls@reference: new .....	201
\glsdetoklabel .....	255	\@gls@warnonglossdefined: new ....	19
\showglouserii: added		\@gls@warnontheGLOSSdefined: new .	19
\glsdetoklabel .....	255	\@no@makeglossaries: new .....	176
\showglouseriii: added		\@print@glossary: new .....	191
\glsdetoklabel .....	256	\@print@noidx@glossary: new .....	198
\showglouseriv: added		\@printgloss@setsort: new .....	189
\glsdetoklabel .....	256	\@printglossary: new .....	189
\showglouserv: added		General: added sort key to printgloss	
\glsdetoklabel .....	256	group .....	204
\showglouservi: added		\compatibleglossentry: changed	
\glsdetoklabel .....	256	\newcommand to \def as is may or	
dua: fixed bug in \acrfullfmt .....	231	may not be defined .....	341
fixed bug in \Acrfullplfmt .....	231	\compatiblesubglossentry: changed	
fixed bug in \acrfullplfmt .....	231	\newcommand to \def as is may or	
redefined to use accessibility		may not be defined .....	342
information .....	366	\defglsdisplayfirst: fixed unwanted	
dua-desc: commented spurious EOL ..	232	space .....	107
		\glo@grabfirst: new .....	198

\gls@defglossaryentry: replaced \ifx with \ifdefvoid .....	85	4.08 (2014-07-30)	
\glsnoidxdisplayloc: new .....	201	\@ACRlong: added	
\glsnoidxdisplaylocclishandler: new .....	201	\do@gls@link@checkfirsthyper	360
\glsnoidxloclist: new .....	200	\@ACRshort: added	
\glsnoidxlocclishandler: new ....	201	\do@gls@link@checkfirsthyper	359
\glsnoidxstripaccents: new .....	22	\@Acrlong: added	
alttree: moved hangindent and parindent assignments outside level test .....	318	\do@gls@link@checkfirsthyper	359
\makeglossaries: Moved definition of \glswrite to \makeglossaries ..	171	\@GLS@: moved \glsifhyper .....	124
\makenoidxglossaries: new .....	173	moved check for first use to \@gls@link .....	124
\printglossary: changed to use new \@printglossary .....	188	\@GLSpl: moved \glsifhyper .....	126
\printnoidxglossaries: new .....	189	moved check for first use to \@gls@link .....	126
\printnoidxglossary: new .....	189	\@Gls@: moved \glsifhyper .....	123
\showgloclolist: new .....	258	moved check for first use to \@gls@link .....	123
\warn@noprintglossary: Activate warning in \makeglossaries ....	188	\@Glspl@: moved \glsifhyper .....	126
\writeist: checked for definition of \glswrite .....	160, 164	moved check for first use to \@gls@link .....	126
4.06 (2014-03-12)		\@acrlong: added	
\@GLS@: added \glsifhyper .....	124	\do@gls@link@checkfirsthyper	360
\@GLSpl: added \glsifhyper .....	126	\@acrshort: added	
\@Gls@: added \glsifhyper .....	123	\do@gls@link@checkfirsthyper	358
\@Glspl@: added \glsifhyper .....	126	\@closegls: new .....	169
\@gls@: added \glsifhyper .....	122	\@gls@: moved \glsifhyper .....	122
\@gls@numbersdef: added hook to set toc title .....	30	moved check for first use to \@gls@link .....	122
\@gls@symbolsdef: added hook to set toc title .....	30	\@gls@automake: new .....	170
\@glsdisp: added \glsifhyper ....	127	\@gls@doautomake: new .....	29
\@glspl@: added \glsifhyper ....	125	\@gls@field@link: added assignment of \do@gls@link@checkfirsthyper	128
General: added \glsifhyper ....	140–147	\@gls@forbidtexext: new .....	59
acronym: added hook to set toc title ....	16	\@gls@hyp@opt: new .....	108
acronyms: added hook to set toc title ...	16	\@gls@link: removed redundancy ....	111
\glsdefmain: added hook to set toc title	15	renamed \gls@type to \glstype ...	111
4.07 (2014-04-04)		\@gls@link@checkfirsthyper: new .	110
\@glossarysection: added optional argument when using unstarred version .....	41	\@glsdisp: moved \glsifhyper .....	127
\@gls@noidx@do: added \global in case it's used in a tabular-like style ....	199	moved check for first use to \@gls@link .....	127
\Acrfullplfmt: fixed no case change bug .....	221	\@glspl@: moved \glsifhyper .....	125
\glsletentryfield: new .....	148	moved check for first use to \@gls@link .....	125
		\@ignored@glossaries: new .....	62
		General: added entrycounter option to printgloss family .....	203

added nopostdot option to printgloss family .....	203	removed \@sglstext .....	128
added subentrycounter option to printgloss family .....	204	removed \@sGLSuseriii .....	137
explicitly initialise hyper key .....	108	removed \@sGlsuseriii .....	137
moved \glsifhyper .....	140–147	removed \@sglsuseriii .....	137
removed \@sACRlongpl .....	147	removed \@sGLSuserii .....	137
removed \@sAcrlongpl .....	146	removed \@sGlsuserii .....	136
removed \@sacrlongpl .....	146	removed \@sGLSuseriv .....	138
removed \@sACRlong .....	145	removed \@sGlsuseriv .....	138
removed \@sAcrlong .....	144	removed \@sglsuseriv .....	138
removed \@sacrlong .....	144	removed \@sGLSuseri .....	136
removed \@sACRshortpl .....	143	removed \@sGlsuseri .....	136
removed \@sAcrshortpl .....	143	removed \@sglsuseri .....	135
removed \@sacrshortpl .....	142	removed \@sGLSservi .....	140
removed \@sACRshort .....	141	removed \@sGlsservi .....	140
removed \@sAcrshort .....	141	removed \@sglsuseri .....	139
removed \@sacrshort .....	140	removed \@sGLSserv .....	139
removed \@sgls@link .....	109	removed \@sGlsuseri .....	139
removed \@sGLSdescplural .....	133	removed \@sglsuseri .....	139
removed \@sGlsdescplural .....	133	removed \@sGLS .....	124
removed \@sglsdescplural .....	133	removed \@sGls .....	123
removed \@sGLSdesc .....	133	removed \@sgls .....	122
removed \@sGlsdesc .....	132	removed \@thirdofthree (defined in kernel) .....	122
removed \@sglsdesc .....	132	removed sPGLS .....	266
removed \@sglsdisp .....	127	removed sPgls .....	264
removed \@sGLSfirstplural .....	131	removed spgls .....	263
removed \@sGlsfirstplural .....	131	removed sPGLSpl .....	266
removed \@sglsfirstplural .....	130	removed sPglspl .....	265
removed \@sGLSfirst .....	129	removed spglspl .....	264
removed \@sGlsfirst .....	129	\ACRfull: removed \s@ACRfull .....	220
removed \@sglsfirst .....	129	switched to using \@gls@hyp@opt ..	220
removed \@sGLSname .....	132	\Acrfull: removed \s@Acrfull .....	219
removed \@sGlsname .....	131	switched to using \@gls@hyp@opt ..	219
removed \@sglsname .....	131	\acrfull: removed \@sacrfull .....	219
removed \@sGLSplural .....	130	switched to using \@gls@hyp@opt ..	219
removed \@sGlsplural .....	130	\ACRfullpl: removed \s@ACRfullpl ..	221
removed \@sglsplural .....	130	switched to using \@gls@hyp@opt ..	221
removed \@sGLSpl .....	126	\Acrfullpl: removed \s@Acrfullpl ..	221
removed \@sGlspl .....	125	switched to using \@gls@hyp@opt ..	221
removed \@sglspl .....	124	\acrfullpl: removed \s@acrfullpl ..	220
removed \@sGLSsymbolplural .....	135	switched to using \@gls@hyp@opt ..	220
removed \@sGlsymbolplural .....	135	\ACRlong: switched to using \@gls@hyp@opt .....	145
removed \@sglsymbolplural .....	134	\Acrlong: switched to using \@gls@hyp@opt .....	144
removed \@sGLSsymbol .....	134	\acrlong: switched to using \@gls@hyp@opt .....	144
removed \@sGlsymbol .....	134		
removed \@sglsymbol .....	134		
removed \@sGLStext .....	128		
removed \@sGlstext .....	129		

\ACRlongpl: switched to using \@gls@hyp@opt .....	147	\glsenablehyper: added \KV@glslink@hypertrue to definition .....	121
\Acrlongpl: switched to using \@gls@hyp@opt .....	146	\GLSfirst: switched to using \@gls@hyp@opt .....	129
\acrlongpl: switched to using \@gls@hyp@opt .....	146	\Glsfirst: switched to using \@gls@hyp@opt .....	129
\ACRshort: switched to using \@gls@hyp@opt .....	141	\glsfirst: switched to using \@gls@hyp@opt .....	129
\Acrshort: switched to using \@gls@hyp@opt .....	141	\GLSfirstplural: switched to using \@gls@hyp@opt .....	131
\acrshort: switched to using \@gls@hyp@opt .....	140	\Glsfirstplural: switched to using \@gls@hyp@opt .....	131
\ACRshorttpl: switched to using \@gls@hyp@opt .....	143	\glsfirstplural: switched to using \@gls@hyp@opt .....	130
\Acrshorttpl: switched to using \@gls@hyp@opt .....	142	\glsifhyper: deprecated .....	108
\acrshorttpl: switched to using \@gls@hyp@opt .....	142	\glslink: switched to using \@gls@hyp@opt .....	109
\forallacronyms: new .....	52	\glslinkcheckfirsthyperhook: new	110
\GLS: switched to using \@gls@hyp@opt	124	\glslinkvar: new .....	108
\Gls: switched to using \@gls@hyp@opt	123	\GLSname: switched to using \@gls@hyp@opt .....	132
\gls: switched to using \@gls@hyp@opt	122	\Glsname: switched to using \@gls@hyp@opt .....	131
\gls@defglossaryentry: added check for ignored glossary .....	82	\glsname: switched to using \@gls@hyp@opt .....	131
\gls@istfilebase: new .....	37	\GLSpl: switched to using \@gls@hyp@opt .....	126
\glsaddkey: removed \@sGLS@user@<key> .....	76	\Glspl: switched to using \@gls@hyp@opt .....	125
removed \@sGls@user@<key> .....	75	\glspl: switched to using \@gls@hyp@opt .....	124
removed \@sgls@user@<key> .....	75	\GLSplural: switched to using \@gls@hyp@opt .....	130
switched to using \@gls@hyp@opt	75, 76	\Glsplural: switched to using \@gls@hyp@opt .....	130
\GLSdesc: switched to using \@gls@hyp@opt .....	132	\glsplural: switched to using \@gls@hyp@opt .....	130
\Glsdesc: switched to using \@gls@hyp@opt .....	132	\glsspace: new .....	219
\glsdesc: switched to using \@gls@hyp@opt .....	132	\GLSsymbol: switched to using \@gls@hyp@opt .....	134
\GLSdescplural: switched to using \@gls@hyp@opt .....	133	\Glsymbol: switched to using \@gls@hyp@opt .....	134
\Glsdescplural: switched to using \@gls@hyp@opt .....	133	\glsymbol: switched to using \@gls@hyp@opt .....	134
\glsdescplural: switched to using \@gls@hyp@opt .....	133	\GLSsymbolplural: switched to using \@gls@hyp@opt .....	135
\glsdisablehyper: added \KV@glslink@hyperfalse to definition .....	121	\Glsymbolplural: switched to using \@gls@hyp@opt .....	135
\glsdisp: switched to using \@gls@hyp@opt .....	127		
\glsdohyperlink: new .....	121		
\glsdohypertarget: new .....	120		



\glssymbolplural: switched to using \@gls@hyp@opt .....	134	\newglossary: added starred version ..	59
\GLStext: switched to using \@gls@hyp@opt .....	128	\newignoredglossary: new .....	62
\Glstext: switched to using \@gls@hyp@opt .....	128	\ns@newglossary: added \@glotype@<name>@log .....	60
\glstext: switched to using \@gls@hyp@opt .....	128	new .....	60
\glstreenamefmt: new .....	311	\p@gls@hyp@opt: new .....	109
\GLSuseri: switched to using \@gls@hyp@opt .....	136	\PGLS: changed to use \@gls@hyp@opt	266
\Glsuseri: switched to using \@gls@hyp@opt .....	136	\Pgls: changed to use \@gls@hyp@opt	264
\glsuseri: switched to using \@gls@hyp@opt .....	135	\pgls: changed to use \@gls@hyp@opt	263
\GLSuserii: switched to using \@gls@hyp@opt .....	137	\PGLSpl: changed to use \@gls@hyp@opt .....	266
\Glsuserii: switched to using \@gls@hyp@opt .....	136	\Pglspl: changed to use \@gls@hyp@opt .....	265
\glsuserii: switched to using \@gls@hyp@opt .....	136	\pglspl: changed to use \@gls@hyp@opt .....	264
\GLSuseriii: switched to using \@gls@hyp@opt .....	137	\s@gls@hyp@opt: new .....	109
\Glsuseriii: switched to using \@gls@hyp@opt .....	137	\s@newglossary: new .....	59
\glsuseriii: switched to using \@gls@hyp@opt .....	137	automake: new .....	29
\GLSuseriv: switched to using \@gls@hyp@opt .....	138	4.09 (2014-08-12)	
\Glsuseriv: switched to using \@gls@hyp@opt .....	138	\glsaddkey: fixed bug in user commands	75
\glsuseriv: switched to using \@gls@hyp@opt .....	138	4.10 (2014-08-27)	
\GLSuseriv: switched to using \@gls@hyp@opt .....	138	\@Gls@acentryname: new .....	149
\GLSuseriv: switched to using \@gls@hyp@opt .....	138	\@Gls@entryname: new .....	149
\GLSuseriv: switched to using \@gls@hyp@opt .....	138	\@gls@glossary: Renamed \@glossary to \@gls@glossary .....	178
\GLSuseriv: switched to using \@gls@hyp@opt .....	138	\glspercentchar: new .....	158
\GLSuseriv: switched to using \@gls@hyp@opt .....	138	\glstildechar: new .....	158
\GLSuseriv: switched to using \@gls@hyp@opt .....	138	alttree: moved space after symbol	318, 319
\GLSuseriv: switched to using \@gls@hyp@opt .....	138	4.11 (2014-09-01)	
\GLSuseriv: switched to using \@gls@hyp@opt .....	138	\@do@esc@wrglossary: added hook ..	184
\GLSuseriv: switched to using \@gls@hyp@opt .....	138	sanitize: none option .....	24
\GLSuseriv: switched to using \@gls@hyp@opt .....	138	\gls@wrglossary: renamed from \@wrglossary to \gls@wrglossary	179
\GLSuseriv: switched to using \@gls@hyp@opt .....	138	\glsaddprotectedpagefmt: new ....	181
\GLSuseriv: switched to using \@gls@hyp@opt .....	138	\glsbackslash: new .....	158
\GLSuseriv: switched to using \@gls@hyp@opt .....	138	4.12 (2014-11-22)	
\GLSuseriv: switched to using \@gls@hyp@opt .....	138	\@gls@addpredefinedattributes: Added glsignore attribute .....	46
\GLSuseriv: switched to using \@gls@hyp@opt .....	138	\@gls@adjustmode: new .....	157
\GLSuseriv: switched to using \@gls@hyp@opt .....	138	\@gls@notranslatorhook: removed ...	24
\GLSuseriv: switched to using \@gls@hyp@opt .....	138	\@gls@toc: added \protect to \numberline .....	42
\GLSuseriv: switched to using \@gls@hyp@opt .....	138	\@gls@usetranslator: new .....	24
\GLSuseriv: switched to using \@gls@hyp@opt .....	138	\glsacrpluralsuffix: new .....	34
\GLSuseriv: switched to using \@gls@hyp@opt .....	138	\glsadd: added check for vertical mode	157
\GLSuseriv: switched to using \@gls@hyp@opt .....	138	\glsaddallunused: replaced @gobble with glsignore .....	157
\GLSuseriv: switched to using \@gls@hyp@opt .....	138	\glsifusedtranslatordict: new ....	25
\GLSuseriv: switched to using \@gls@hyp@opt .....	138	\glsignore: new .....	158
\ifignoredglossary: new .....	62		
altlongragged4col: fixed bug that displayed description instead of symbol .....	291		

\glsupacrpluralsuffix:new .....	34	4.16 (2015-06-18)	
\ProvidesGlossariesLang:new .....	34	\glsaddstoragekey:new .....	73
\RequireGlossariesLang:new .....	34	4.16 (2015-07-08)	
4.13 (2015-02-03)		\@ACRlong: added \glspostlinkhook	361
\indexspace:new .....	273, 293, 311	\@ACRshort: added \glspostlinkhook	360
4.14 (2015-02-28)		\@Acrlong: added \glspostlinkhook	360
\@glslocalreset:new .....	90	\@Acrshort: added \glspostlinkhook	359
\@glslocalunset:new .....	90	\@GLS@: added \glspostlinkhook ...	124
\@glsreset:new .....	91	\@GLSpl: added \glspostlinkhook ..	127
\@glsunset:new .....	90	\@Gls@: added \glspostlinkhook ...	124
\@newglossaryentry@defcounters:		\@Glspl@: added \glspostlinkhook ..	126
new .....	92	\@acrlong: added \glspostlinkhook	360
\cGls:new .....	95	\@acrshort: added \glspostlinkhook	359
\cGls@:new .....	95	\@gls@: added \glspostlinkhook ...	123
\cGlspl@:new .....	96	\@gls@@link: added	
\cgls:new .....	95	\glspostlinkhook .....	110
\cgls@:new .....	95	\@gls@field@link: added	
\cglspl@:new .....	96	\glspostlinkhook .....	128
\cglspl@:new .....	96	\@gls@link: moved definition of	
\gls@entry@count:new .....	94	\glsifhyperon outside of this	
\gls@increment@currcount:new ...	94	macro .....	111
\gls@local@increment@currcount:		\@glsdisp: added \glspostlinkhook	127
new .....	94	\@glspl@: added \glspostlinkhook ..	125
\gls@write@entrycounts:new .....	94	General: added \glspostlinkhook	141–147
\glslocalreset:new .....	90	\glsacspace:new .....	226
\glslocalunset:new .....	90	\glsadd: changed \@do@wrglossary to	
\glsreset:new .....	90	\@do@wrglossary .....	157
\glsunset:new .....	90	\glsfielddef:new .....	77
\@newglossaryentry@defcounters:		\glsfieldedef:new .....	77
new .....	86	\glsfieldfetch:new .....	78
\cGls:new .....	95	\glsfieldgdef:new .....	77
\cgls:new .....	94	\glsfieldxdef:new .....	76
\cGlsformat:new .....	95	\glsifhyperon: moved definition of	
\cglsformat:new .....	95	\glsifhyperon .....	110
\cGlspl:new .....	96	\glslinkpostsetkeys:new .....	110
\cglspl:new .....	95	\glspostlinkhook:new .....	110
\cGlsplformat:new .....	96	\glswriteentry:new .....	180
\cglsplformat:new .....	96	\ifglsfieldcseq:new .....	79
\gls@defdocnewglossaryentry:new ..	70	\ifglsfielddefeq:new .....	79
\glsenableentrycount:new .....	92	\ifglsfielddeq:new .....	78
\glslocalreset: switched to		long-sp-short:new .....	225
\@glslocalreset .....	89	\showglofield:new .....	258
\glslocalunset: switched to		4.18 (2015-09-09)	
\@glslocalunset .....	90	General: split mfirstuc into separate	
\glsreset: switched to \@glsreset ...	89	bundle .....	4
\glsunset: switched to \@glsunset ...	89	4.19 (2015-10-31)	
4.15 (2015-03-16)		\glsstreenamebox:new .....	317
General: bug fix replaced \@glo@type		4.19 (2015-11-22)	
with \glstype .....	147	\@gls@link@nocheckfirsthyper:new	128

\@gls@preglossaryhook: new .....	189	\glsfindwidesttoplevelname: new .	317
\@printglossary: added		\glslistgroupheaderfmt: new .....	273
\@gls@preglossaryhook .....	191	\glslistnavigationitem: new .....	273
\do@glsglisablehyperinlist: new ..	110	\glstreegroupheaderfmt: new .....	311
\doifglossarynoexistsordo: new ...	55	\glstreenavigationfmt: new .....	311
\gls@gobbleopt: new .....	59	\ifglswrallowprimitivemods: new .	182
\glsglsoifexistsordo: new .....	54	list: fixed missing space before	
4.20 (2015-11-30)		description .....	273
\@gls@link: added		long: fixed typo in \glossentrydesc .	277
\@gls@setdefault@glslink@opts	111	super4col: fixed bug in \glossentry .	302
added \glsglsonohyperlink when		4.23 (2016-04-30)	
hyperlink is suppressed .....	111	\glsglcurrentfieldvalue: new .....	58
\@gls@setdefault@glslink@opts:		\ifglshasfield: added	
new .....	110	\glsglcurrentfieldvalue .....	57, 58
\glsgl@checkseeallowed@preambleonly:		altlongragged4col: check for	
new .....	65	nogroupskip changed .....	291
\glsglsonohyperlink: new .....	121	altsuperragged4col: check for	
4.21 (2016-01-24)		nogroupskip changed .....	310
\@printglossary: warn if no style has		long: check for nogroupskip changed ..	277
been set .....	189	long-booktabs: check for nogroupskip	
General: changed checkfirsthyper		changed .....	283
assignment .....	140–147	long3col: check for nogroupskip	
\glossarystyle: set default style if not		changed .....	279
already set .....	213	long3col-booktabs: check for	
\glsglLTpenaltycheck: new .....	286	nogroupskip changed .....	284
\glsglpatchLToutput: new .....	286	long4col: check for nogroupskip	
\glsglspenaltygroupskip: new .....	286	changed .....	280
altlong4col-booktabs: new .....	284	long4col-booktabs: check for	
altlongragged4col-booktabs: new .	285	nogroupskip changed .....	284
long-booktabs: new .....	283	longragged: check for nogroupskip	
long3col-booktabs: new .....	283	changed .....	288
long4col-booktabs: new .....	284	longragged3col: check for nogroupskip	
longragged-booktabs: new .....	285	changed .....	289
longragged3col-booktabs: new ....	285	super: check for nogroupskip changed .	299
\setglossarystyle: set default style if		super3col: check for nogroupskip	
not already set .....	213	changed .....	301
4.22 (2016-04-19)		super4col: check for nogroupskip	
\@do@esc@wrglossary: added check		changed .....	303
for \@arabic .....	183	superragged: check for nogroupskip	
added test to allow temporary primitive		changed .....	306
modifications and added arabic case	183	superragged3col: check for	
mcolalttreespannav: new .....	298	nogroupskip changed .....	308
mcolindexspannav: new .....	294	4.24 (2016-05-27)	
mcoltreenonamespannav: new .....	296	\@gls@extramakeindexopts: new ...	168
mcoltreespannav: new .....	295	\@gls@glossary: added check for debug	
\glsgl@arabicpage: new .....	180	mode .....	178
\glsgl@protected@pagefmts: added		\@gls@see@noindex: new .....	6
arabic to list .....	180	debug: new .....	5
\glsglentrytitlecase: new .....	152	seenoinindex: new .....	6

\glsnomakeindexwarning: new .....	43	\@xdylocationclassorder: bug fix:	
\GlsSetQuote: new .....	166	changed \edef to \def .....	49
\GlsSetWriteIstHook: new .....	165	\glosortentrieswarning: new .....	19
4.25 (2016-06-09)		\gls@set@xr@key: new .....	65
\@gls@enablesavenonumberlist: new	66	\gls@xr@key: new .....	65
\@gls@initnonnumberlist: new .....	66	\GlsAddXdyLocation: bug fix: changed	
\@gls@savenonumberlist: new .....	65	#1 to #2 .....	49
4.26 (2016-10-12)		\glsnoidxstripaccents: added \a ...	22
\@glossary@default@style: added		added \TH, \dh and \DH .....	22
check for classicthesis .....	8	4.31 (2017-08-10)	
mcolindex: replaced \@idxitem with		nolist: added check for “list” style .....	10
\glstreeitem .....	293	4.31 (2017-09-10)	
mcolindexspannav: replaced \@idxitem		style: changed \renewcommand to \def .	8
with \glstreeitem .....	294	4.32 (2017-08-24)	
\glstreechildpredesc: new .....	312	\@glsnavhypertarget: new .....	268
\glstreeitem: new .....	311	\@glsshowtarget: new .....	6
\glstreepredesc: new .....	312	\glsshowtarget: new .....	6
\glstreesubitem: new .....	312	4.33 (2017-09-20)	
\glstreesubsubitem: new .....	312	\@do@esc@wrglossary: added	
4.28 (2017-01-07)		\gls@the and \gls@number .....	183
\glspatchtabularx: new .....	89	renamed from	
4.29 (2017-01-19)		\@do@esc@wrglossary .....	182
\@gls@noidx@do: current letter group		\@do@noesc@wrglossary: new .....	181
assignment made global .....	200	\@do@wrglossary: changed to check	
\@print@noidx@glossary: moved		for esclocations .....	181
definition of		\@gls@missinglang@warn: new .....	19
\@gls@currentlettergroup outside		\GlsSetXdyFirstLetterAfterDigits:	
of the glossary environment .....	198	added starred version .....	159
General: added check for		\GlsSetXdyNumberGroupOrder: new .	159
\@glxtr@doaccsupp .....	341	esclocations: new .....	9
\glsnavhyperlinkname: new .....	268	4.34 (2017-11-03)	
4.30 (2017-06-11)		mcolalttreespannav: removed spurious	
\@glo@autosee: new .....	86	space .....	298
\@glo@autoseehook: new .....	86	\glsshowtarget: modified to check for	
\@glo@check@sortallowed: new .....	12	math mode and inner .....	6
\@gls@noidx@do: letter group		4.35 (2017-11-14)	
assignment made global .....	200	\glsadd: added \@gls@setsort (in case	
\@gls@setupsort@def: added check for		of sort=use) .....	157
register .....	13	4.36 (2018-03-07)	
\@gls@setupsort@none: new .....	14	\@gls@glossary: removed \index ...	179
\@xdycrossrefhook: new .....	48	4.37 (2018-04-07)	
		\gls@begindocdefs: new .....	71

# 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	
\!	117
\"	22, 114, 116, 117, 119
\#	162
\%	158, 164, 165, 325, 326
\&	34, 156
\'	22
\.	10, 22
\=	22
\?	114, 116, 167
\@	71
\@@delimN	215
\@@do@@wrglossary	173, 182, 184
\@@do@esc@wrglossary	181
\@@do@noesc@wrglossary	181
\@@do@wrglossary	157, 180
\@@glo@assign@sortkey	174
\@@glo@list	52
\@@glo@sort	22
\@@glo@type	189
\@@glossarysec	7, 41, 42
\@@glossaryseclabel	8, 41, 42, 203
\@@glossarysecstar	8, 41, 202, 203
\@@gls@checkactual	119
\@@gls@checkbar	118
\@@gls@checkescactual	116
\@@gls@checkescbar	117
\@@gls@checkesclevel	117, 118
\@@gls@checkescquote	116, 168
\@@gls@checklevel	118, 119
\@@gls@checkquote	115, 166, 167
\@@gls@default@entryfmt	97, 106, 107
\@@gls@expand@field	20, 69, 73, 74, 237, 239, 241, 243, 246, 248–250, 372–376
\@@gls@extramakeindexopts	166, 171
\@@gls@fixbraces	186
\@@gls@noexpand@field	20, 68, 69
\@@gls@noidx@no@sanitizesort	21, 22
\@@gls@noidx@nosanitizesort	176
\@@gls@nosanitizesort	21, 176
\@@gls@sanitizesort	21, 176
\@@gls@xdycheckbackslash	120
\@@gls@xdycheckquote	119, 120
\@@gls@localreset	90, 93
\@@gls@localunset	90, 93
\@@gls@reset	90, 93
\@@gls@unset	90, 92
\@@newglossaryentry@defcounters	92
\@@this@glo@	53
\@ACRfull	220
\@ACRfullpl	221
\@ACRlong	145, 220
\@ACRlongpl	147, 221
\@ACRshort	141, 220
\@ACRshortpl	143, 221
\@Acrfull	219
\@Acrfullpl	221
\@Acrlong	144, 220
\@Acrlongpl	146, 221
\@Acrshort	141
\@Acrshortpl	143
\@Alpha	180, 183, 184
\@GLS	124
\@GLS@	124, 266
\@GLSdesc	132, 133
\@GLSdesc@	133
\@GLSdescplural	133
\@GLSdescplural@	133
\@GLSfirst	129
\@GLSfirst@	129
\@GLSfirstplural	131
\@GLSfirstplural@	131
\@GLSname	132
\@GLSname@	132

\@GLSpl	126	\@Glsuseri	136
\@GLSpl@	126, 267	\@Glsuseri@	136
\@GLSplural	130	\@Glsuserii	136
\@GLSplural@	130	\@Glsuserii@	136
\@GLSsymbol	134	\@Glsuseriii	137
\@GLSsymbol@	134	\@Glsuseriii@	137
\@GLSsymbolplural	135	\@Glsuseriv	138
\@GLSsymbolplural@	135	\@Glsuseriv@	138
\@GLStext	128	\@Glsuserv	139
\@GLStext@	128	\@Glsuserv@	139
\@GLSuseri	136	\@Glsuservi	140
\@GLSuseri@	136	\@Glsuservi@	140
\@GLSuserii	137	\@Mi	286
\@GLSuserii@	137	\@PGLS	266
\@GLSuseriii	137	\@PGLS@	266
\@GLSuseriii@	137, 138	\@PGLSpl	266
\@GLSuseriv	138	\@PGLSpl@	266
\@GLSuseriv@	138	\@Pgls	264
\@GLSuserv	139	\@Pgls@	264
\@GLSuserv@	139	\@Pglspl	265
\@GLSuservi	140	\@Pglspl@	265
\@GLSuservi@	140	\@Roman	180, 183, 184
\@Gls	123	\@acrfull	219
\@Gls@	93, 95, 123, 265	\@acrfullpl	220
\@Gls@acrentryname	222	\@acrlong	144, 219
\@Gls@entry@field	75, 149–154	\@acrlongpl	146, 220
\@Gls@entryname	149, 222	\@acrshort	140, 219, 220
\@GlsSetXdyFirstLetterAfterDigits	159	\@acrshortpl	142, 220, 221
\@GlsSetXdyNumberGroupOrder	159	\@addtoacronymlists	16, 17
\@Glsdesc	132	\@after	17
\@Glsdesc@	132	\@afterheading	274, 329
\@Glsdescplural	133	\@alph	180, 183, 184
\@Glsdescplural@	133	\@arabic	180, 183, 184
\@Glsfirst	129	\@auxout	58,
\@Glsfirst@	129		60, 94, 171, 173, 174, 177, 188, 192, 269
\@Glsfirstplural	131	\@backslashchar	114, 120
\@Glsfirstplural@	131	\@before	17
\@Glsname	131	\@bsphack	179
\@Glsname@	131, 132	\@cGls	95
\@Glspl	125	\@cGls@	93, 95
\@Glspl@	93, 96, 125, 265, 266	\@cGlspl	96
\@Glsplural	130	\@cGlspl@	93, 96
\@Glsplural@	130	\@cclv	286, 287
\@Glsymbol	134	\@cgls	94
\@Glsymbol@	134	\@cgls@	93, 95
\@Glsymbolplural	135	\@cglspl	95
\@Glsymbolplural@	135	\@cglspl@	93, 96
\@Glstext	128, 129	\@chapter	32
\@Glstext@	129	\@classoptionslist	31

\@closegls .....	170	\@glo@autoseehook .....	86
\@colht .....	286	\@glo@check@mkidxrangechar .....	113, 185, 322, 323
\@colroom .....	286, 287	\@glo@check@sortallowed ..	12–14, 173, 176
\@currentlabelname .....	8, 203	\@glo@childlist .....	193
\@curroptions .....	31	\@glo@counter .....	64, 81, 84
\@declaredoptions .....	31	\@glo@counterprefix .....	177, 181, 184–186, 213, 216
\@delimN .....	215	\@glo@default@sorttype ..	11, 174, 195–197
\@delimR .....	215	\@glo@defaultcounter .....	84
\@disable@onlypremakeg .....	172	\@glo@desc .....	63, 79, 80, 82, 85
\@disable@premakecs .....	32	\@glo@descaccess .....	343–345
\@disabled@gl\$addxdycounters .....	45	\@glo@descplural .....	63, 79, 80
\@do@addcounter .....	44	\@glo@descpluralaccess .....	343–345
\@do@auxoutstuff .....	191, 192	\@glo@do@sortentries .....	193
\@do@glossentry .....	208, 341, 342	\@glo@entry .....	157, 158
\@do@gl\$@getcounterprefix .....	182, 184	\@glo@entryprefix .....	261
\@do@gl\$@islistofacronyms .....	17	\@glo@entryprefixfirst .....	261
\@do@gl\$see .....	86	\@glo@entryprefixfirstplural ..	261, 262
\@do@ifinlist .....	43, 44	\@glo@entryprefixplural .....	261
\@do@newglossaryentry .....	222, 223, 237, 239, 241, 243–246, 248–253, 372–376	\@glo@esclabel .....	87, 88
\@do@seeglossary .....	173, 187	\@glo@etext .....	98, 100
\@do@subglossentry .....	210, 342	\@glo@first .....	63, 81, 83–85, 248, 376
\@do@wrglossary .....	111	\@glo@firstaccess .....	342, 344, 345
\@do@writeaux@info .....	188	\@glo@firstplural .....	64, 81, 84, 376
\@ehc .....	286	\@glo@firstpluralaccess .....	343–345
\@empty ...	14, 16, 29, 31, 32, 43, 45, 48, 51, 52, 82, 87, 88, 112, 113, 122–126, 140– 147, 160, 163, 165, 170, 177, 179, 185, 186, 203, 205, 213, 238, 240, 242, 244– 247, 249, 251, 253, 323, 325, 327, 359–361	\@glo@grabfirst .....	199
\@end@fixbraces .....	186, 187	\@glo@label .....	67, 74– 80, 82–86, 92, 155, 156, 261, 262, 317, 345
\@endfortrue .....	26, 55, 73, 269	\@glo@list .....	86
\@esphack .....	179	\@glo@long .....	56, 67, 81, 84
\@expandtwoargs .....	31	\@glo@longaccess .....	343–345
\@firstofone .....	22	\@glo@longpl .....	67, 81, 84, 237, 239, 241, 243, 245, 248, 250, 372
\@firstofthree .....	109, 122, 125, 127, 140, 142, 144, 146, 359–361	\@glo@longpluralaccess .....	343–345
\@firstoftwo .....	25, 26, 72, 73, 109, 125, 126, 142, 143, 146, 147	\@glo@name .....	12, 63, 68, 80, 83, 84
\@for .....	26, 31, 32, 44, 45, 52, 72, 73, 114, 160–162, 171, 172, 180, 187, 193, 224, 237, 240, 242, 244, 246, 249, 251, 253, 269, 270, 323	\@glo@no@assign@sortkey .....	172
\@glo@@desc .....	84, 85	\@glo@nonumberlist .....	66
\@glo@@symbol .....	85	\@glo@numfmt .....	185, 323
\@glo@access .....	342, 344, 345, 347	\@glo@parent .....	14, 65, 81–83, 87, 88, 194
\@glo@addchildren .....	193, 197	\@glo@plural .....	63, 81, 83, 374
\@glo@assign@sortkey .....	172, 174, 204	\@glo@pluralaccess .....	343–345
\@glo@autosee .....	85	\@glo@prefix 9, 65, 81, 87, 88, 113, 185, 322, 323	
		\@glo@range .....	185, 322, 323
		\@glo@see .....	64, 81, 86
		\@glo@seeautonumberlist .....	9, 65
		\@glo@short .....	56, 67, 81, 84, 375
		\@glo@shortaccess .....	343–345, 372–375

<code>\@glo@shortpl</code> .....	67, 81, 84,	<code>\@gls@Hcounter</code> .....	112, 113
237, 239, 241, 243, 245, 248, 250, 372, 375		<code>\@gls@ReturnAfterFi</code> .....	216
<code>\@glo@shortpluralaccess</code> .....	343–345	<code>\@gls@access@display</code> .....	347, 348
<code>\@glo@sort</code> ...	12, 14, 21, 22, 63, 81, 83, 87, 88	<code>\@gls@actualchar</code> ....	88, 116, 119, 164, 326
<code>\@glo@sortedinsert</code> .....	194	<code>\@gls@addpredefinedattributes</code> .	160, 169
<code>\@glo@sortentries</code> .....	196, 197	<code>\@gls@adjustmode</code> .....	157
<code>\@glo@sorthandler@case</code> .....	196	<code>\@gls@automake</code> .....	172
<code>\@glo@sorthandler@letter</code> .....	196	<code>\@gls@between</code> .....	270
<code>\@glo@sorthandler@nocase</code> .....	197	<code>\@gls@body</code> .....	149
<code>\@glo@sorthandler@word</code> .....	196	<code>\@gls@checkactual</code> .....	115, 167
<code>\@glo@sortinghandler</code> .....	193, 194	<code>\@gls@checkbar</code> .....	115, 167
<code>\@glo@sortinglist</code> .....	193, 194, 197	<code>\@gls@checkedmkidx</code> ....	114–120, 166–168
<code>\@glo@sorttype</code> .....	174, 198, 199, 205	<code>\@gls@checkescactual</code> .....	114, 167
<code>\@glo@storeentry</code> .....	12–14	<code>\@gls@checkescbar</code> .....	115, 167
<code>\@glo@sufffix</code> .....	113, 185, 323	<code>\@gls@checkescquote</code> .....	114, 167, 168
<code>\@glo@symbol</code> .....		<code>\@gls@checklevel</code> .....	115, 167
....	56, 64, 80, 85, 242, 243, 247, 248, 344	<code>\@gls@checkmkidxchars</code> .....	
<code>\@glo@symbolaccess</code> .....	343–345, 375	....	87, 113, 167, 173, 184, 186, 322, 323
<code>\@glo@symbolplural</code> .....	64, 80, 85	<code>\@gls@checkquote</code> .....	114, 166, 167
<code>\@glo@symbolpluralaccess</code> .....	343–345	<code>\@gls@classI</code> .....	161
<code>\@glo@text</code> .....	63,	<code>\@gls@classII</code> .....	161
80, 83, 85, 122–127, 148, 149, 243, 262, 374		<code>\@gls@codepage</code> .....	192
<code>\@glo@textaccess</code> ...	342, 344, 345, 372–375	<code>\@gls@counter</code> .....	
<code>\@glo@thislabel</code> .....	86	108, 111–113, 156, 157, 177, 185, 186, 323	
<code>\@glo@thislettergrp</code> .....	199, 200	<code>\@gls@counterwithin</code> ....	11, 203, 205, 206
<code>\@glo@thisvalue</code> .....	57, 58	<code>\@gls@ctr</code> .....	44
<code>\@glo@tmp</code> .....	74, 75, 185	<code>\@gls@currentlettergroup</code> .....	198, 200
<code>\@glo@type</code> .....	8, 14, 64,	<code>\@gls@debugfalse</code> .....	5
80, 82, 84, 85, 156, 157, 171, 177, 178,		<code>\@gls@debugtrue</code> .....	5
189–194, 197, 198, 202, 203, 222, 238,		<code>\@gls@declareoption</code> .....	
240, 242, 244, 246, 249, 251, 253, 268, 270		....	9, 10, 15, 16, 19, 20, 25, 28, 30
<code>\@glo@types</code> 52, 60, 91, 157, 171, 172, 258, 317		<code>\@gls@default</code> .....	97
<code>\@glo@useri</code> .....	66, 81, 84	<code>\@gls@default@value</code> .....	
<code>\@glo@userii</code> .....	66, 81, 84	....	56–58, 68, 69, 80, 81, 83–85, 247, 261
<code>\@glo@useriii</code> .....	66, 81, 84	<code>\@gls@deffile</code> .....	71, 72
<code>\@glo@useriv</code> .....	66, 81, 84	<code>\@gls@defsort</code> .....	12–14, 85
<code>\@glo@userv</code> .....	66, 81, 84	<code>\@gls@defsortcount</code> .....	12–14, 60
<code>\@glo@uservi</code> .....	67, 81, 84	<code>\@gls@do@acronymsdef</code> ....	15, 16, 31, 32, 61
<code>\@glodesc</code> .....	85	<code>\@gls@do@indexdef</code> .....	30–32, 61
<code>\@glolist@</code> .....	82	<code>\@gls@do@numbersdef</code> .....	30–32, 61
<code>\@gloname</code> .....	84	<code>\@gls@do@symbolsdef</code> .....	30, 61
<code>\@glossary@default@style</code> 8, 10, 189, 213, 254		<code>\@gls@do@symbolssdef</code> .....	31, 32
<code>\@glossaryentryfield</code> .....	87, 88	<code>\@gls@doautomake</code> .....	29, 172
<code>\@glossarysection</code> .....	40	<code>\@gls@docheckquotedef</code> .....	166–168
<code>\@glossarystyle</code> .....	189, 190, 202	<code>\@gls@docloadedfalse</code> .....	4
<code>\@glossarysubentryfield</code> .....	88	<code>\@gls@docloadedtrue</code> .....	4
<code>\@gls</code> .....	122	<code>\@gls@dodeflistparser</code> .....	172
<code>\@gls@</code> .....	93, 95, 122, 264, 265	<code>\@gls@doentrydef</code> .....	106, 107
<code>\@gls@@link</code> .....	109	<code>\@gls@dolast</code> .....	187



\@gls@donext .....	187	\@gls@loadlist .....	10, 253
\@gls@donext@def .....	155, 156	\@gls@loadlong .....	9, 10, 253
\@gls@dotothiswrite .....	170	\@gls@loadsuper .....	10, 253
\@gls@elem .....	269	\@gls@loadtree .....	10, 254
\@gls@enablesavenonumberlist .....	71	\@gls@local@increment@currcount ....	93
\@gls@encapchar .....		\@gls@loclist .....	174, 175, 199, 200
..... 117, 118, 164, 185, 186, 323, 326		\@gls@map .....	72, 73
\@gls@entry@count .....	94	\@gls@missinglang@warn .....	19, 35
\@gls@entry@field .....		\@gls@missingnumberlist .....	84
..... 74, 75, 92, 149–155, 345–347		\@gls@noaccess .....	347
\@gls@escbsdq .....	114, 165, 327	\@gls@noexpand@fields .....	70
\@gls@expand@fields .....	69, 70	\@gls@nohyperlist .....	18, 62, 110
\@gls@expandonce .....	69	\@gls@noidx@do .....	198
\@gls@extramakeindexopts .....	171	\@gls@noidx@getgrouptitle .....	173
\@gls@fetchfield .....	57	\@gls@noidx@sanitizesort .....	21, 176
\@gls@field@link .....	75, 76, 128–140	\@gls@noidx@setsanitizesort ....	23, 176
\@gls@firsttok .....	198, 199	\@gls@noidx@loclist@finalsep .....	175
\@gls@fixbraces .....	86	\@gls@noidx@loclist@prev .....	175, 201
\@gls@forbidtexext .....	60	\@gls@noidx@loclist@sep .....	175, 201
\@gls@get@counterprefix .....	185, 186	\@gls@noref@warn .....	173, 198
\@gls@getbody .....	149	\@gls@numberlink .....	215, 216
\@gls@getcounterprefix .....	182, 184	\@gls@numbersdef .....	30
\@gls@getgrouptitle .....	173, 211, 270	\@gls@numlist@lastsep .....	155, 156
\@gls@glossary .....	178, 179	\@gls@numlist@nextsep .....	155, 156
\@gls@gobbleopt .....	59	\@gls@numlist@sep .....	155, 156
\@gls@grptitle .....	211, 268, 270	\@gls@old@chapter .....	32
\@gls@hyp@opt .....	75,	\@gls@oldnewglossaryentryposthook .	344
76, 94–96, 109, 122–147, 219–221, 263–266		\@gls@oldnewglossaryentryprehook ..	344
\@gls@hyp@opt@cs .....	109	\@gls@onlypremakeg .....	32
\@gls@hypergroup .....	269	\@gls@order .....	170
\@gls@ifinlist .....	44	\@gls@org@LT@output .....	286
\@gls@ifnotmeasuring .....	88	\@gls@org@glsnoidx@displayloc ..	175, 176
\@gls@igtype .....	62	\@gls@org@glsseeformat .....	175, 176
\@gls@increment@currcount .....	92	\@gls@patchtabularx .....	89
\@gls@indexdef .....	30	\@gls@preglossaryhook .....	191
\@gls@initnonumberlist .....	66, 81	\@gls@prevlevel .	297, 298, 317–320, 335, 336
\@gls@islistofacronyms .....	17	\@gls@provide@newglossary .....	60
\@gls@keylist .....	371	\@gls@quotechar .	115–119, 164, 166, 168, 326
\@gls@keymap .....	66, 72–75, 261, 344	\@gls@reference .....	174, 177
\@gls@label .....	173, 174, 177, 182, 184, 185	\@gls@removespaces .....	216
\@gls@langmod .....	170	\@gls@renewglossary .....	169
\@gls@levelchar .....	88, 117, 118, 164, 326	\@gls@replacementtext .....	347
\@gls@link ..	109, 122–128, 140–147, 359–361	\@gls@rest .....	149
\@gls@link@checkfirsthyper ....	122–127	\@gls@restreat .....	71
\@gls@link@label .....	111, 239, 245	\@gls@roman .....	47, 323, 324
\@gls@link@nocheckfirsthyper	128, 140–147	\@gls@sanitized@tmp .....	114
\@gls@link@opts .....	111, 239, 245	\@gls@sanitizedesc .....	27
\@gls@list .....	269, 270	\@gls@sanitizesort .....	12
\@gls@listsuffix .....	43	\@gls@sanitizesymbol .....	27

\@gls@saveentrycounter	111, 157	\@glsdescplural	133
\@gls@savenonumberlist	65, 66	\@glsdescplural@	133
\@gls@see@noindex	7, 65	\@glsdisp	127
\@gls@setacrstyle	27, 31	\@glsentry	91, 94
\@gls@setcounter	60	\@glsentrytitlecase	152
\@gls@setdefault@glslink@opts	111	\@glsfirst	129
\@gls@setsort	12–14, 111, 157	\@glsfirst@	129
\@gls@setupshortcuts	31, 32	\@glsfirstletter	159
\@gls@sort	200	\@glsfirstplural	130
\@gls@sort@A	194, 195	\@glsfirstplural@	130
\@gls@sort@B	195	\@glshypernumber	215
\@gls@startswithexpandonce	69	\@glsisacronymlistfalse	17
\@gls@storenonumberlist	66, 84	\@glsisacronymlisttrue	17
\@gls@symbolsdef	30	\@glslink	111, 121, 122, 156, 268
\@gls@this	180	\@glslocalreset	89, 93
\@gls@thisHloc	185	\@glslocalunset	90, 93
\@gls@thisfield	57	\@glslocref	177, 181, 182, 184, 185, 322, 323
\@gls@thislabel	55, 187, 197	\@glsminrange	160, 161, 324
\@gls@thislist	155, 156	\@glsname	131
\@gls@thisloc	185	\@glsname@	131
\@gls@thisval	73	\@glsnavhypertarget	268
\@gls@title	40	\@glsnextpages	190
\@gls@tmp	14, 35, 48, 69, 114, 179, 269, 270	\@glsnodesc	80, 82, 85
\@gls@tmpb	115–120, 166, 168	\@glsnoname	80, 83, 84
\@gls@toc	41, 42	\@glsnonextpages	190
\@gls@type	172, 224, 237, 240, 242, 244, 246, 249, 251, 253, 317	\@glsnumberformat	108, 111, 156, 157, 177, 185, 322, 323
\@gls@updatechecked	114, 115, 167	\@glsopenfile	169, 178
\@gls@usetranslator	25, 26, 34	\@glsorder	171
\@gls@value	68, 69, 152	\@glspl	124
\@gls@warnonglossdefined	20, 188	\@glspl@	93, 96, 124, 264–266
\@gls@warnontheGLOSSdefined	20, 208	\@glsplural	130
\@gls@write@entrycounts	94	\@glsplural@	130
\@gls@writedef	71	\@glsreset	89, 93
\@gls@writeisthook	164, 165	\@glssee	86, 187
\@gls@xdy@locationlist	161	\@glsshowtarget	5, 6, 121
\@gls@xdycheckbackslash	114	\@glsymbol	134
\@gls@xdycheckquote	114	\@glsymbol@	134
\@gls@xref	186	\@glsymbolplural	134
\@gls@Alphacompositor	38, 48, 324	\@glsymbolplural@	134, 135
\@gls@Hlocref	181, 182, 184	\@gls@target	121, 122, 208, 269
\@gls@acronymlists	16–18, 52, 222, 224, 237, 238, 240, 242, 244, 246, 249, 251, 253, 258	\@gls@text	128
\@gls@addkey	74	\@gls@text@	128
\@gls@addstoragekey	73	\@glsunset	90, 92
\@gls@addxdyattribute	44, 45	\@glsuseri	135
\@gls@defaultsort	12	\@glsuseri@	135
\@gls@desc	132	\@glsuserii	136
\@gls@desc@	132	\@glsuserii@	136
		\@glsuseriii	137

\@glsuseriii@	137	\@org@gls@assign@plural	237, 239, 241, 243–246, 248–251, 372–376
\@glsuseriv	138	\@org@gls@assign@symbolplural	237, 239–241, 243, 244, 248–251, 373, 374, 376
\@glsuseriv@	138	\@org@gls@numberformat	155
\@glsuserv	138, 139	\@org@newglossaryentryprehook	80
\@glsuserv@	139	\@outputpage	286, 287
\@glsuservi	139	\@p@glossarysection	40
\@glsuservi@	139	\@pgls	263
\@glswidestname	317–319, 335	\@pgls@	263
\@glswritefiles	29	\@pglspl	264
\@glsxtr@doaccsupp	341	\@pglspl@	264
\@gobble	5, 12–14, 71, 72, 89, 114, 158, 162, 173, 321, 325, 326	\@plus	273, 293, 311
\@idxitem	311	\@print@glossary	188
\@ifclassloaded	4, 11, 40	\@print@noidx@glossary	189
\@ifnextchar	60, 109	\@printgloss@setsort	172, 174, 190
\@ifpackageloaded	4, 8, 24–26, 34, 51, 88, 155, 166, 341	\@printglossary	188, 189
\@ifstar	59, 73, 74, 109, 159, 217	\@roman	47, 323
\@ifundefined	34, 269, 276, 287, 298, 305, 318, 319, 335, 349	\@secondofthree	109, 122, 123, 125, 141, 143, 145, 147, 359
\@ignored@glossaries	62	\@secondoftwo	22, 25, 26, 35, 72, 73, 121–124, 127, 140–142, 144, 145, 359–361, 379
\@input@	191	\@set@glo@numformat	185, 323
\@istfilename	171	\@sglsaddkey	74
\@makecol	286, 287	\@sglsaddstoragekey	73
\@makeglossary	171, 172	\@thirdofthree	109, 124, 126, 142, 143, 145, 147, 359
\@minus	273, 293, 311	\@this@attr	162
\@mkboth	40, 41	\@this@childlabel	193
\@newglossary	58, 60	\@this@counter	45
\@newglossaryentry@defcounters	85, 92	\@this@ctr	162
\@newglossaryentryposthook	74, 75, 86, 261, 344	\@this@key	73
\@newglossaryentryprehook	74, 75, 80, 81, 261, 344	\@this@label	193
\@nil	17, 86, 113–115, 149, 167, 185, 186, 198–200, 215, 216, 322, 323	\@this@cs	32
\@nnil	17, 187	\@tmp	47, 324
\@no@makeglossaries	172, 174	\@use@option	31
\@no@post@desc	328	\@warn@nomakeglossaries	171, 192
\@nopostdesc	190	\@wrglossary@pageformat	181
\@onelevel@sanitize	21, 47, 72, 88, 114, 159, 163, 186, 188, 199, 324, 325	\@wrglossarynumberhook	181, 184
\@onlypreamble	61, 70, 80, 94, 97, 173, 176	\@xdy@main@language	28, 170, 192
\@onlypremakeg	37, 38, 44, 45, 49, 61, 165	\@xdy@attributelist	45, 162
\@org@glossaryentrynumbers	190, 191	\@xdy@attributes	44, 160, 321, 323
\@org@gls@assign@descplural	237, 246, 248–251, 372, 375, 376	\@xdy@counters	44, 45, 162
\@org@gls@assign@firstpl	237, 239–241, 243, 244, 246, 248–251, 372–376	\@xdy@crossrefhook	162
		\@xdy@language	192
		\@xdy@lettergroups	52, 164, 326
		\@xdy@locationclassorder	49, 162, 325
		\@xdy@locref	45, 163, 321, 325
		\@xdy@numbergrouporder	51, 159

<code>\@xdyrequiredstyles</code> .....	50, 160, 323	<code>\acronymsort</code> .....	222, 225–230, 232–235, 363–365, 368, 369, 371
<code>\@xdysortrules</code> .....	50, 164, 326	<code>\acronymtype</code> .....	15, 16, 222, 224, 237–246, 248–253, 372–375
<code>\@xdystyle</code> .....	160, 323	<code>\acrpluralsuffix</code> .....	223, 225– 228, 232–234, 237–241, 243–246, 248– 250, 252, 253, 363, 364, 368–370, 372–376
<code>\@xdyuseralphabets</code> .....	46, 160, 323	<code>\Acrshort</code> .....	235
<code>\@xdyuserlocationdefs</code> ...	48, 161, 322, 324	<code>\acrshort</code> .....	235
<code>\@xdyuserlocationnames</code> .....	49, 322	<code>\Acrshortpl</code> .....	235
<code>\@xfor@nextelement</code> .....	187	<code>\acrshortpl</code> .....	235
<code>\\</code> .....	86, 87, 114, 158, 164, 165, 215, 216, 326, 327, 329–331, 339, 340	<code>\addcontentsline</code> .....	43
<code>\{</code> .....	71, 72, 158, 165, 321, 326, 327	<code>\addglossarytocaptions</code> .....	35
<code>\}</code> .....	72, 158, 165, 321, 327	<code>\addtolength</code> .....	319, 335
<code>\^</code> .....	22	<code>\advance</code> .....	13, 14, 83, 112, 286
<code>\‘</code> .....	22	<code>\AE</code> .....	22
<code>\ </code> .....	115, 117, 167	<code>\ae</code> .....	22
<code>\~</code> .....	22	<code>amsgen package</code> .....	4, 107
<b>A</b>		<code>amsmath package</code> .....	88
<code>\a</code> .....	22	<code>\andname</code> .....	187
<code>\AA</code> .....	22	<code>\AnyTrackedLanguages</code> .....	35, 379
<code>\aa</code> .....	22	<code>\appto</code> .....	18, 66, 74, 75, 261, 344
<code>accsupp package</code> .....	341	<code>array package</code> .....	283, 287, 305
<code>\accsuppglossaryentryfield</code> .....	341	<code>article class</code> .....	185
<code>\accsuppglossarysubentryfield</code> .....	342	<code>\AtBeginDocument</code> ....	16, 51, 71, 88, 157, 174
<code>\acrfootnote</code> .....	239, 245	<code>\AtEndDocument</code> .....	29, 71, 94, 173, 177, 191, 192, 269
<code>\Acrfull</code> .....	236	<b>B</b>	
<code>\acrfull</code> .....	236	<code>\b</code> .....	22
<code>\ACRfullfmt</code> ....	220, 223, 231, 233, 367, 369	<code>babel package</code> .....	24, 33, 35, 50
<code>\Acrfullfmt</code> ....	220, 223, 231, 233, 367, 369	<code>\begin</code> ...	162, 198, 273, 277–282, 285–311, 325
<code>\acrfullfmt</code> ....	219, 223, 231, 233, 367, 369	<code>\BeginAccSupp</code> .....	347
<code>\acrfullformat</code> ....	154, 155, 219, 237, 252	<code>\begingroup</code> .....	5, 179, 183, 216
<code>\Acrfullpl</code> .....	236	<code>\bfseries</code> .....	278–281, 283, 284, 288–290, 292, 300–305, 307–311
<code>\acrfullpl</code> .....	236	<code>\bgroup</code> .....	22, 79, 155, 190, 193
<code>\ACRfullplfmt</code> ...	221, 223, 231, 233, 367, 369	<code>bib2gls</code> .....	182
<code>\Acrfullplfmt</code> ...	221, 223, 231, 233, 367, 369	<code>booktabs package</code> .....	282–285
<code>\acrfullplfmt</code> ...	220, 223, 231, 233, 367, 369	<code>\boolean</code> .....	251
<code>\acrlinkfootnote</code> .....	238	<code>\boolfalse</code> .....	29
<code>\acrlinkfullformat</code> .....	219–221	<code>\booltrue</code> .....	29
<code>\Acrlong</code> .....	235	<code>\bottomrule</code> .....	283, 284
<code>\acrlong</code> .....	235	<code>\box</code> .....	286
<code>\Acrlongpl</code> .....	235	<b>C</b>	
<code>\acrlongpl</code> .....	235	<code>\c</code> .....	22
<code>\acrnameformat</code> .....	243, 374	<code>\c@equation</code> .....	112
<code>\acronymentry</code> .....	222, 225–230, 232–235, 363–365, 368–371	<code>\c@glossarysubentry</code> .....	204
<code>\acronymfont</code> .....	104, 140–143, 149, 154, 155, 221–223, 225–235, 238–240, 242, 244–249, 356, 357, 359–361, 363–365, 367–371, 373–375	<code>\c@page</code> .....	180, 181, 183, 184
<code>\acronymname</code> .....	15, 16, 36		

<code>\catcode</code> .....	71	<code>\DeclareAcronymList</code> ....	15, 16, 18, 222, 224, 238, 240, 242, 244, 246, 249, 251, 253
<code>\cGls</code> .....	95	<code>\DeclareListParser</code> .....	172
<code>\cglS</code> .....	95	<code>\DeclareOption</code> .....	8, 261, 341
<code>\cGlsformat</code> .....	93	<code>\DeclareOptionX</code> .....	8
<code>\cglSformat</code> .....	93	<code>\DeclareRobustCommand</code> .....	
<code>\cGlspl</code> .....	96	.....	36, 187, 188, 247, 347–349
<code>\cglSpl</code> .....	96	<code>\def</code> .....	8, 9, 12–14, 17, 21, 22, 27, 28, 31–33, 36, 37, 40, 43, 46–51, 55, 59– 61, 63–67, 69, 78, 80–86, 89, 93, 95–97, 106–108, 111–120, 122–147, 155–157, 160, 164, 166–168, 170, 172, 174, 175, 177, 181, 183–187, 189, 190, 193, 197– 199, 201–206, 212–217, 219–222, 237– 244, 246–251, 253, 261, 263–267, 270– 272, 297, 298, 317–320, 322, 323, 326, 328, 335, 336, 341–344, 358–361, 372–376
<code>\cGlsplformat</code> .....	93	<code>\def@glS@xdycheckbackslash</code> .....	120
<code>\cglSplformat</code> .....	93	<code>\DefaultNewAcronymDef</code> .....	238
<code>\char</code> .....	212	<code>\defglSentryfmt</code> .....	60, 62, 106, 107, 224, 236, 238, 241, 242, 244, 247, 250, 252
classcthesis package .....	8	<code>\define@boolkey</code> 7, 9, 11, 15, 23, 26–29, 108, 204	
<code>\cleardoublepage</code> .....	42	<code>\define@choicekey</code> .....	5–8, 11, 24, 25, 27, 65, 202–204
<code>\clearpage</code> .....	42	<code>\define@key</code> .....	8, 11, 18, 24, 28, 63–67, 74, 75, 107, 108, 156, 202, 204, 261, 342, 343
<code>\closeout</code> .....	71, 164, 165, 170, 178	<code>\DefineAcronymSynonyms</code> .....	31, 236
<code>\compatglossarystyle</code> .....	328–340	<code>\delimN</code> .....	163, 172, 201, 215, 216, 325
<code>\compatibleglossentry</code> .....	210	<code>\delimR</code> .....	163, 215, 325
<code>\compatiblesubglossentry</code> .....	210	<code>\DescriptionDUANewAcronymDef</code> .....	242
<code>\copy</code> .....	286, 287	<code>\DescriptionFootnoteNewAcronymDef</code> .	240
<code>\count@</code> .....	199	<code>\descriptionname</code> .....	36, 278–281, 283, 284, 288–290, 292, 300–305, 307–311
<code>\csdef</code> .....	20, 74–76, 85, 86, 92, 93, 193, 194, 214, 224, 327	<code>\DescriptionNewAcronymDef</code> .....	244
<code>\csedef</code> .....	94, 181	<code>\DH</code> .....	22
<code>\csgdef</code> .....	39, 59, 62, 93, 94, 188, 202	<code>\dh</code> .....	22
<code>\cslet</code> .....	66, 80, 86, 197	<code>\dimen@</code> .....	226, 286, 317
<code>\csname</code> 11–14, 31, 33, 35, 36, 41, 42, 45, 47, 48, 51, 52, 55, 60, 61, 68, 69, 73–79, 82– 88, 90, 91, 106, 107, 111–113, 122–127, 140–148, 155, 157, 161, 162, 167–170, 174, 177–179, 181, 185, 186, 189–192, 194, 202, 208, 210, 213, 214, 217, 254– 262, 269, 270, 317–319, 321–323, 335, 341, 342, 345, 346, 349, 359–361, 377, 378		<code>\disable@keys</code> .....	31
<code>\csshow</code> .....	258	<code>\do</code> .....	26, 31, 32, 44, 45, 52, 72, 73, 114, 155, 160–162, 171, 172, 180, 187, 193, 224, 237, 240, 242, 244, 246, 249, 251, 253, 269, 270, 323
<code>\csuse</code> .....	36, 39, 59, 68, 69, 75, 76, 106, 107, 170, 194, 196, 198, 199, 201, 203, 204, 213, 225, 262, 328–340	<code>\do@glo@storeentry</code> .....	12–14, 85
<code>\csxdef</code> .....	84, 94	<code>\do@glS@link@checkfirsthyper</code> .....	109, 111, 122–128, 140–147, 359, 360
<code>\currentglossary</code> .....	39, 190, 203, 206	<code>\do@glS@xdycheckbackslash</code> .....	114
<code>\currentglssubentry</code> .....	204, 206, 207	<code>\do@glSdisablehyperinlist</code> .....	111
<code>\CurrentOption</code> .....	31, 261, 341	<code>\do@glshaschildren</code> .....	55
<code>\CurrentTrackedLanguage</code> .....	35, 379, 380	doc package .....	4, 5, 15
<code>\CurrentTrackedTag</code> .....	35, 379, 380		
<code>\CustomAcronymFields</code> .....	253		
<code>\CustomNewAcronymDef</code> .....	253		

## D

<code>\d</code> .....	22
datatool package .....	194
<code>\day</code> .....	160, 164, 323, 326

<code>\doifglossarynoexistsordo</code> .....	60	<code>\endgroup</code> .....	5, 179, 184, 216
<code>\dtl@ifsingle</code> .....	212	<code>\endhead</code> .	277–279, 281, 283, 284, 288–290, 292
<code>\dtl@insertinto</code> .....	194	<code>\endtheglossary</code> .....	5
<code>\dtl@sortresult</code> .....	195	<code>\entryname</code> .....	36, 278–281, 283, 284, 288–290, 292, 300–305, 307–311
<code>\dtlcompare</code> .....	195	<code>\equal</code> .....	24, 32, 42, 112, 171, 212, 269
<code>\dtlicompare</code> .....	195	equation (counter) .....	112
<code>\DTLifinlist</code> .....	62, 110	etoolbox package .....	4
<code>\DTLifint</code> .....	212	<code>\expandafter</code> .....	12–14, 21, 31, 32, 35, 45, 47, 48, 50–53, 55, 60–62, 68, 69, 71–79, 82–91, 106, 107, 110, 112– 120, 149, 156, 158, 162, 166–169, 177– 181, 183–185, 187, 190, 194, 199, 200, 208–210, 214, 216, 217, 239, 245, 254– 259, 261, 262, 269, 270, 317, 321–323, 325, 326, 341, 342, 345, 347, 371, 377, 378
<code>\dtlletterindexcompare</code> .....	195	<code>\expandonce</code> .....	68, 69, 114, 167, 168, 181, 195, 208, 210, 222, 237, 239, 241, 243, 245, 248, 250, 341, 342
<code>\DTLsubstituteall</code> .....	114		
<code>\dtlwordindexcompare</code> .....	195		
<code>\DUANewAcronymDef</code> .....	251		
<b>E</b>			
<code>\eappto</code> .....	62, 86, 181		
<code>\edef</code> .....	14, 17, 32, 35, 43–50, 52, 55, 60, 62, 68, 69, 71, 73, 76–80, 82, 86, 87, 106, 107, 111, 112, 114–120, 155, 157, 158, 164, 166–168, 170, 172, 173, 177, 182, 184, 185, 188, 191–195, 199, 204, 207, 212, 216, 217, 237, 239, 241, 243, 245, 248, 250, 268, 321, 322, 324, 326, 371–375		
<code>\egroup</code> .....	22, 80, 156, 191, 194		
<code>\else</code> .....	6, 10, 14–17, 19, 21, 23, 24, 29, 31, 32, 36–38, 40–52, 65, 68, 82, 84, 87–89, 93, 110–120, 123– 127, 149, 159, 160, 163–166, 168–170, 177–181, 183–187, 190, 199, 203, 204, 206–208, 213, 215, 216, 226, 240, 242, 244, 246, 247, 249–251, 254, 269, 274, 277, 279, 280, 283, 284, 286, 288, 290, 291, 299, 301, 303, 306, 308, 310, 313, 314, 316, 318, 319, 322–328, 333–336, 347		
<code>\emph</code> .....	187, 217		
<code>\empty</code> .....	216, 341		
<code>\end</code> .....	162, 198, 273, 277–282, 285–311, 325		
<code>\end@doifinlist</code> .....	43, 44		
<code>\end@getprefix</code> .....	185, 186		
<code>\end@glis@islistofacronyms</code> .....	17		
<code>\EndAccSupp</code> .....	347		
<code>\endcsname</code> .....	11– 14, 31, 33, 35, 36, 41, 42, 45, 47, 48, 51, 52, 55, 60, 61, 68, 69, 73–79, 82– 88, 90, 91, 106, 107, 111–113, 122–127, 140–148, 155, 157, 161, 162, 167–170, 174, 177–179, 181, 185, 186, 189–192, 194, 202, 208, 210, 213, 214, 217, 254– 262, 269, 270, 317–319, 321–323, 335, 341, 342, 345, 346, 349, 359–361, 377, 378		
<code>\endfoot</code> .	277–279, 281, 283, 284, 288–290, 292		
		<b>F</b>	
		<code>\fi</code> .....	5–8, 10, 12–17, 19, 21, 23, 24, 26, 29, 31, 32, 36–38, 40–52, 61, 65, 68, 82–85, 87–89, 93, 94, 110–120, 123–127, 149, 157, 159, 160, 163, 165–169, 171, 172, 178–188, 190, 192, 199, 203, 204, 206–208, 213– 216, 226, 236, 238, 240, 242, 244, 246, 247, 249–254, 260, 269, 274, 277, 279, 280, 283, 284, 286–288, 290, 291, 299, 301, 303, 306, 308, 310, 313, 314, 316– 319, 321–325, 327, 328, 333–336, 341, 347
		file types	
		<code>.aux</code> .....	191
		<code>.glo</code> .....	87
		<code>.ist</code> .....	158, 168, 169
		<code>.toc</code> .....	42
		<code>.xdy</code> .....	37
		<code>glo</code> .....	259
		<code>\firstacronymfont</code> .....	
		.....	105, 106, 225–227, 232, 233, 238, 243, 245, 248, 358, 362–364, 368, 369
		<code>\footnote</code> .....	232, 233, 238, 369
		<code>\FootnoteNewAcronymDef</code> .....	246
		<code>\forall glossaries</code> .....	53, 177, 189, 317
		<code>\forall glsentries</code> .....	91, 94, 157, 158
		<code>\ForEachTrackedDialect</code> .....	35, 379, 380
		<code>\for glsentries</code> .....	53, 55, 86, 197, 317
		<code>\forlistcsloop</code> .....	193, 198
		<code>\forlistloop</code> .....	175, 201

## G

garamondx package .....	218	
\gdef .....	13, 45, 60, 77, 82, 83, 179, 205, 269	
\Genacrfullformat .....		
	105, 223, 225, 226, 232, 358, 362, 363, 369	
\genacrfullformat .....	105, 106,	
	223, 225, 226, 232, 357, 358, 362, 363, 368	
\GenericAcronymFields ..	223, 225, 226,	
	228–232, 234, 235, 362–365, 367, 368, 371	
\Genplacrfullformat .....		
	.... 105, 223, 225–227, 232, 357, 363, 369	
\genplacrfullformat .....		
	104–106, 223, 225, 226, 232, 357, 363, 369	
\glo@desc .....	328	
\glo@do@compare .....	195	
\glo@grabfirst .....	200	
\glo@label .....	55, 86	
\glo@list .....	86	
\glo@name .....	209	
\glo@parent .....	55	
\glo@type .....	86	
\glo@value .....	72	
\global .....	13, 14, 68, 71, 80,	
	85, 90, 91, 179, 191, 199, 200, 205, 286, 287	
\glolinkprefix .....	111, 156, 208	
\glosortentrieswarning .....	19, 193	
glossareentry (counter) .....	206	
glossaries package .....		
	... 30, 50, 51, 160, 253, 261, 273, 321, 341	
glossaries-accsupp package .....	86, 87, 341	
glossaries-extra package .....	71, 162, 341	
\GlossariesWarning .....	5,	
	7, 19, 23, 24, 39, 43, 54, 58, 65, 68, 95,	
	96, 106–108, 155, 170, 171, 173, 175,	
	176, 179, 186, 189, 191, 210, 213, 321, 341	
\GlossariesWarningNoLine .....		
	..... 5, 6, 19, 172, 174, 177, 192, 269	
\glossary .....	322, 323	
glossary package .....	1, 217	
glossary styles:		
altlist .....	274, 275, 329	
altlistgroup .....	275, 329	
altlisthypergroup .....	275, 329	
altlong4col .....	281, 282, 290, 331	
altlong4col-booktabs .....	284, 286	
altlong4colborder .....	282, 331	
altlong4colheader .....	282, 284, 331	
altlong4colheaderborder ....	282, 331	
altlongragged4col ...	285, 290–292, 332	
altlongragged4col-booktabs .....	285	
altlongragged4colborder ....	292, 332	
altlongragged4colheader ....	291, 332	
altlongragged4colheaderborder	292, 333	
altsuper4col .....	304, 309, 340	
altsuper4colborder .....	304, 340	
altsuper4colheader .....	304, 340	
altsuper4colheaderborder ...	304, 340	
altsuperragged4col .....	309, 310, 338	
altsuperragged4colborder ...	310, 338	
altsuperragged4colheader ...	310, 338	
altsuperragged4colheaderborder .		
	..... 310, 338	
alttree .....	297, 312, 317, 319, 335	
alttreegroup .....	320, 336	
alttreehypergroup .....	320, 336	
index .....	8, 293, 311–314, 333	
indexgroup .....	313, 333	
indexhypergroup .....	313, 333	
inline .....	328	
list .....	8, 10, 273–275, 328	
listdotted .....	275, 276, 329	
listgroup .....	274, 328	
listhypergroup .....	274, 329	
long .....	277, 278, 283, 287, 329, 331	
long-booktabs .....	283, 285	
long3col .....	278, 279, 283, 330	
long3col-booktabs .....	283, 285	
long3colborder .....	279, 330	
long3colheader .....	279, 283, 330	
long3colheaderborder .....	279, 330	
long4col .....	280, 281, 284, 330	
long4col-booktabs .....	284	
long4colborder .....	281, 331	
long4colheader .....	280, 284, 331	
long4colheaderborder .....	281, 331	
longborder .....	277, 330	
longheader .....	277, 283, 330	
longheaderborder .....	278, 330	
longragged .....	285, 287–289	
longragged-booktabs .....	285	
longragged3col .....	285, 289, 290, 332	
longragged3col-booktabs .....	285	
longragged3colborder .....	290, 332	
longragged3colheader .....	290, 332	
longragged3colheaderborder .	290, 332	
longraggedborder .....	288, 331	
longraggedheader .....	288, 332	
longraggedheaderborder .....	289, 332	



<code>mcolalttree</code> .....	<a href="#">297</a> , <a href="#">337</a>	<code>\glossaryentryfield</code> .....	
<code>mcolalttreegroup</code> .....	<a href="#">297</a> , <a href="#">337</a>	.....	<a href="#">208</a> , <a href="#">328–335</a> , <a href="#">337–340</a> , <a href="#">362</a>
<code>mcolalttreehypergroup</code> ...	<a href="#">297</a> , <a href="#">298</a> , <a href="#">337</a>	<code>\glossaryentrynumbers</code> .....	
<code>mcolindex</code> .....	<a href="#">293</a> , <a href="#">336</a>	.....	<a href="#">9</a> , <a href="#">163</a> , <a href="#">190</a> , <a href="#">191</a> , <a href="#">200</a> , <a href="#">204</a> , <a href="#">205</a> , <a href="#">325</a>
<code>mcolindexgroup</code> .....	<a href="#">293</a> , <a href="#">336</a>	<code>\glossaryheader</code> .....	
<code>mcolindexhypergroup</code> .....	<a href="#">293</a> , <a href="#">294</a> , <a href="#">336</a>	.....	<a href="#">162</a> , <a href="#">198</a> , <a href="#">271</a> , <a href="#">273–275</a> , <a href="#">277–</a>
<code>mcoltree</code> .....	<a href="#">294</a> , <a href="#">336</a>		<a href="#">281</a> , <a href="#">283</a> , <a href="#">284</a> , <a href="#">287–293</a> , <a href="#">295–297</a> , <a href="#">299</a> ,
<code>mcoltreegroup</code> .....	<a href="#">336</a>		<a href="#">300</a> , <a href="#">302</a> , <a href="#">306</a> , <a href="#">307</a> , <a href="#">309</a> , <a href="#">312–317</a> , <a href="#">320</a> , <a href="#">325</a>
<code>mcoltreehypergroup</code> .....	<a href="#">295</a> , <a href="#">336</a>	<code>\glossarymark</code> .....	<a href="#">40</a>
<code>mcoltreenoname</code> .....	<a href="#">296</a> , <a href="#">337</a>	<code>\glossaryname</code> .....	<a href="#">15</a> , <a href="#">35</a> , <a href="#">36</a>
<code>mcoltreenonamegroup</code> .....	<a href="#">296</a> , <a href="#">337</a>	<code>\glossarypostamble</code> .....	<a href="#">162</a> , <a href="#">198</a> , <a href="#">325</a>
<code>mcoltreenonamehypergroup</code> ...	<a href="#">296</a> , <a href="#">337</a>	<code>\glossarypreamble</code> .....	<a href="#">162</a> , <a href="#">198</a> , <a href="#">325</a>
<code>sublistdotted</code> .....	<a href="#">329</a>	<code>\glossarysection</code> .....	<a href="#">162</a> , <a href="#">198</a> , <a href="#">325</a>
<code>super</code> .....	<a href="#">299</a> , <a href="#">300</a> , <a href="#">307</a> , <a href="#">339</a>	<code>glossarysubentry (counter)</code> ....	<a href="#">11</a> , <a href="#">206</a> , <a href="#">207</a>
<code>super3col</code> .....	<a href="#">300–302</a> , <a href="#">339</a>	<code>\glossarysubentryfield</code> .....	
<code>super3colborder</code> .....	<a href="#">301</a> , <a href="#">339</a>	.....	<a href="#">210</a> , <a href="#">328–335</a> , <a href="#">337–340</a> , <a href="#">362</a>
<code>super3colheader</code> .....	<a href="#">301</a> , <a href="#">339</a>	<code>\glossarytitle</code> ..	<a href="#">162</a> , <a href="#">189</a> , <a href="#">190</a> , <a href="#">198</a> , <a href="#">202</a> , <a href="#">325</a>
<code>super3colheaderborder</code> .....	<a href="#">302</a> , <a href="#">339</a>	<code>\glossarytoctitle</code> .....	<a href="#">8</a> , <a href="#">15</a> , <a href="#">16</a> ,
<code>super4col</code> .....	<a href="#">302–304</a> , <a href="#">340</a>		<a href="#">30</a> , <a href="#">33</a> , <a href="#">36</a> , <a href="#">40</a> , <a href="#">162</a> , <a href="#">189</a> , <a href="#">198</a> , <a href="#">202</a> , <a href="#">203</a> , <a href="#">325</a>
<code>super4colborder</code> .....	<a href="#">303</a> , <a href="#">340</a>	<code>\glossentry</code> .....	<a href="#">87</a> , <a href="#">190</a> , <a href="#">191</a> , <a href="#">200</a> , <a href="#">210</a> ,
<code>super4colheader</code> .....	<a href="#">303</a> , <a href="#">340</a>		<a href="#">271</a> , <a href="#">273–278</a> , <a href="#">280</a> , <a href="#">288</a> , <a href="#">289</a> , <a href="#">291</a> , <a href="#">299</a> ,
<code>super4colheaderborder</code> .....	<a href="#">303</a> , <a href="#">340</a>		<a href="#">301</a> , <a href="#">302</a> , <a href="#">306</a> , <a href="#">307</a> , <a href="#">309</a> , <a href="#">312</a> , <a href="#">314</a> , <a href="#">315</a> , <a href="#">318</a>
<code>superborder</code> .....	<a href="#">299</a> , <a href="#">339</a>	<code>\Glossentrydesc</code> .....	<a href="#">361</a>
<code>superheader</code> .....	<a href="#">300</a> , <a href="#">339</a>	<code>\glossentrydesc</code> .....	
<code>superheaderborder</code> .....	<a href="#">300</a> , <a href="#">339</a>		<a href="#">271–278</a> , <a href="#">280</a> , <a href="#">288</a> , <a href="#">289</a> , <a href="#">291</a> , <a href="#">299</a> , <a href="#">301</a> ,
<code>superragged</code> .....	<a href="#">305</a> , <a href="#">307</a> , <a href="#">337</a>		<a href="#">302</a> , <a href="#">306–309</a> , <a href="#">312–314</a> , <a href="#">316</a> , <a href="#">318</a> , <a href="#">319</a> , <a href="#">361</a>
<code>superragged3col</code> .....	<a href="#">307–309</a> , <a href="#">338</a>	<code>\glossentryname</code> .....	
<code>superragged3colborder</code> .....	<a href="#">308</a> , <a href="#">338</a>		<a href="#">271–278</a> , <a href="#">280</a> , <a href="#">288</a> , <a href="#">289</a> , <a href="#">291</a> , <a href="#">299</a> , <a href="#">301</a> ,
<code>superragged3colheader</code> .....	<a href="#">308</a> , <a href="#">338</a>		<a href="#">302</a> , <a href="#">306</a> , <a href="#">307</a> , <a href="#">309</a> , <a href="#">312–315</a> , <a href="#">318</a> , <a href="#">319</a> , <a href="#">361</a>
<code>superragged3colheaderborder</code> .....	<a href="#">309</a> , <a href="#">338</a>	<code>\Glossentrysymbol</code> .....	<a href="#">362</a>
<code>superraggedborder</code> .....	<a href="#">306</a> , <a href="#">337</a>	<code>\glossentrysymbol</code> .....	<a href="#">271</a> , <a href="#">272</a> , <a href="#">280</a> ,
<code>superraggedheader</code> .....	<a href="#">307</a> , <a href="#">337</a>		<a href="#">291</a> , <a href="#">302</a> , <a href="#">309</a> , <a href="#">312–314</a> , <a href="#">316</a> , <a href="#">318</a> , <a href="#">319</a> , <a href="#">362</a>
<code>superraggedheaderborder</code> ....	<a href="#">307</a> , <a href="#">338</a>	<code>\Gls</code> .....	<a href="#">95</a> , <a href="#">217</a> , <a href="#">236</a>
<code>tree</code> .....	<a href="#">294</a> , <a href="#">314</a> , <a href="#">315</a> , <a href="#">317</a> , <a href="#">333</a>	<code>\gls</code> .....	<a href="#">95</a> , <a href="#">173</a> , <a href="#">207</a> , <a href="#">217</a> , <a href="#">236</a>
<code>treegroup</code> .....	<a href="#">295</a> , <a href="#">315</a> , <a href="#">334</a>	<code>\gls@Alphpage</code> .....	<a href="#">180</a> , <a href="#">184</a>
<code>treehypergroup</code> .....	<a href="#">315</a> , <a href="#">334</a>	<code>\gls@alphpage</code> .....	<a href="#">180</a> , <a href="#">184</a>
<code>treenoname</code> .....	<a href="#">295</a> , <a href="#">312</a> , <a href="#">315</a> , <a href="#">316</a> , <a href="#">334</a>	<code>\gls@arabicpage</code> .....	<a href="#">180</a> , <a href="#">184</a>
<code>treenonamegroup</code> .....	<a href="#">316</a> , <a href="#">335</a>	<code>\gls@assign@desc</code> .....	<a href="#">80</a> , <a href="#">85</a>
<code>treenonamehypergroup</code> .....	<a href="#">316</a> , <a href="#">335</a>	<code>\gls@assign@descplural</code> .....	
<code>glossary-hypernav package</code> .....	<a href="#">158</a>	.....	<a href="#">237</a> , <a href="#">246</a> , <a href="#">248–251</a> , <a href="#">372</a> , <a href="#">375</a> , <a href="#">376</a>
<code>glossary-list package</code> .....	<a href="#">8</a> , <a href="#">10</a> , <a href="#">273</a>	<code>\gls@assign@field</code> .....	
<code>glossary-long package</code> ....	<a href="#">9</a> , <a href="#">276</a> , <a href="#">290</a> , <a href="#">298</a> , <a href="#">299</a>	.....	<a href="#">70</a> , <a href="#">74</a> , <a href="#">75</a> , <a href="#">79</a> , <a href="#">82–85</a> , <a href="#">261</a> , <a href="#">262</a>
<code>glossary-longragged package</code> .....	<a href="#">287</a>	<code>\gls@assign@firstpl</code> .....	<a href="#">237</a> ,
<code>glossary-mcols package</code> .....	<a href="#">292</a>		<a href="#">239–241</a> , <a href="#">243</a> , <a href="#">244</a> , <a href="#">246</a> , <a href="#">248–251</a> , <a href="#">372–376</a>
<code>glossary-super package</code> ..	<a href="#">10</a> , <a href="#">276</a> , <a href="#">298</a> , <a href="#">305</a> , <a href="#">309</a>	<code>\gls@assign@plural</code> .....	
<code>glossary-superragged package</code> .....	<a href="#">305</a>		<a href="#">237</a> , <a href="#">239</a> , <a href="#">241</a> , <a href="#">243–246</a> , <a href="#">248–251</a> , <a href="#">372–376</a>
<code>glossary-tree package</code> .....	<a href="#">10</a> , <a href="#">311</a>	<code>\gls@assign@symbolplural</code> .....	<a href="#">237</a> ,
<code>\glossaryentry</code> .....	<a href="#">185</a> , <a href="#">186</a> , <a href="#">323</a>		<a href="#">239–241</a> , <a href="#">243</a> , <a href="#">244</a> , <a href="#">248–251</a> , <a href="#">373</a> , <a href="#">374</a> , <a href="#">376</a>
<code>glossaryentry (counter)</code> .....	<a href="#">11</a> , <a href="#">206</a> , <a href="#">207</a>	<code>\gls@begindocdefs</code> .....	<a href="#">71</a>



<code>\gls@checkisacronymlist</code> .....	110	<code>\gls@tmp</code> .....	177, 247
<code>\gls@checkseeallowed</code> .....	65, 70, 172, 173	<code>\gls@tmplen</code> .....	120, 121, 317–319, 335, 336
<code>\gls@checkseeallowed@preambleonly</code> ..	70	<code>\gls@tr@set@acronym@toctitle</code> .....	16
<code>\gls@codepage</code> .....	51, 170, 192	<code>\gls@tr@set@main@toctitle</code> .....	15
<code>\gls@defdocnewglossaryentry</code> .....	71, 92	<code>\gls@tr@set@numbers@toctitle</code> .....	30
<code>\gls@defglossaryentry</code> .....	70, 71, 80	<code>\gls@tr@set@symbols@toctitle</code> .....	30
<code>\gls@disablepagerefexpansion</code> ..	179, 184	<code>\gls@wrglossary</code> .....	179
<code>\gls@do@addxdyattribute</code> .....	45	<code>\gls@xdystring</code> .....	114
<code>\gls@docclearpage</code> .....	42	<code>\gls@xindy@glsnumbersfalse</code> .....	28
<code>\gls@dosubst</code> .....	114	<code>\gls@xindy@glsnumberstrue</code> .....	28
<code>\gls@dotocitle</code> .....	190, 202	<code>\gls@xr@key</code> .....	6, 7, 65
<code>\gls@end@sanitizesort</code> .....	21, 22	<code>\glsaccsupp</code> .....	347
<code>\gls@endcheck</code> .....	69	<code>\glsacronymtrue</code> .....	16
<code>\gls@glossary</code> .....	178, 185, 186	<code>\glsacrpluralsuffix</code> .....	
<code>\gls@gobbleopt</code> .....	60	.....	34, 218, 227, 228, 232–234, 238
<code>\gls@grplabel</code> .....	268	<code>\glsacrshortcutsfalse</code> .....	31
<code>\gls@hypergrouprerun</code> .....	269	<code>\glsacrshortcutstrue</code> .....	31
<code>\gls@ifnotmeasuring</code> .....	89, 90	<code>\glsacspace</code> .....	226, 228
<code>\gls@inlinepostchild</code> .....	271, 272, 328	<code>\glsadd</code> .....	157, 158
<code>\gls@inlinesep</code> .....	271, 328	<code>\glsadd options</code>	
<code>\gls@inlinesubsep</code> .....	271, 272, 328	counter .....	156
<code>\gls@islistofacronyms</code> .....	17	format .....	156, 214
<code>\gls@istfilebase</code> .....	36, 37, 170	<code>\glsaddall options</code>	
<code>\gls@label</code> .....	217	types .....	156, 157
<code>\gls@level</code> .....	82, 83, 199, 200	<code>\GlsAddXdyAttribute</code> .....	44–46, 321, 322
<code>\gls@noidxglossary</code> .....	173	<code>\GlsAddXdyCounters</code> .....	44, 45, 61
<code>\gls@nosetquote</code> .....	80, 164, 166, 168	<code>\glsautomakefalse</code> .....	29
<code>\gls@number</code> .....	183, 184	<code>\glsautoprefix</code> .....	8, 203
<code>\gls@numberpage</code> .....	180, 183	<code>\glscapscase</code> .....	97, 99, 101–
<code>\gls@org@glossaryentryfield</code> ...	190, 191	105, 122–127, 140–147, 230, 243, 247,	
<code>\gls@org@glossarysubentryfield</code>	190, 191	349, 351, 353, 354, 356, 357, 359–361, 366	
<code>\gls@org@insert</code> .....	242, 245, 247	<code>\glscclearpage</code> .....	41
<code>\gls@orgAlph</code> .....	183, 184	<code>\glscclosebrace</code> ....	48, 49, 163, 164, 325, 326
<code>\gls@orgalph</code> .....	183, 184	<code>\glscompositor</code> ....	37, 38, 48, 165, 324, 327
<code>\gls@orgarabic</code> .....	183, 184	<code>\glscounter</code> .....	18, 32, 44, 60, 84, 112, 321
<code>\gls@orgnumber</code> .....	183	<code>\glscurrententrylabel</code> .....	188, 191
<code>\gls@orgRoman</code> .....	183, 184	<code>\glscurrentfieldvalue</code> .....	57, 58
<code>\gls@orgromannumeral</code> .....	183, 184	<code>\glscustomtext</code> .....	
<code>\gls@orgthe</code> .....	183	..	97, 101, 103, 105, 122–127, 140–147,
<code>\gls@protected@pagefmts</code> ....	114, 180, 181	230, 231, 238, 239, 242–245, 247, 248,	
<code>\gls@Romanpage</code> .....	180, 184	349, 352, 353, 355, 356, 358–361, 366, 367	
<code>\gls@romanpage</code> .....	180, 184	<code>\GlsDeclareNoHyperList</code> .....	18
<code>\gls@save@numberlist</code> .....	9	<code>\glsdefaulttype</code> .....	15, 39, 51,
<code>\gls@set@xr@key</code> .....	64	52, 59, 60, 81, 82, 97, 106, 107, 177, 188, 189	
<code>\gls@suffixF</code> .....	38, 163, 165, 325, 327	<code>\glsdefmain</code> .....	15, 61
<code>\gls@suffixFF</code> .....	38, 163, 165, 325, 327	<code>\glsdescriptionaccessdisplay</code> .....	
<code>\gls@text</code> .....	106	.....	351–353, 361, 362
<code>\gls@the</code> .....	183	<code>\glsdescriptionpluralaccessdisplay</code>	
<code>\gls@thissty</code> .....	26	.....	349, 350

<code>\glsdescwidth</code> .....	277–	<code>\Glsentrylong</code> .....	95, 145, 149,
	279, 281, 282, 285–292, 299–302, 304–311		154, 225, 226, 231, 358, 360, 363, 366–368
<code>\glsdetoklabel</code> .....	53–	<code>\glsentrylong</code> ....	95, 105, 144, 145, 149,
	57, 66, 71, 72, 76–80, 86, 90–94, 111,		154, 225, 226, 228–235, 245, 358, 360–371
	148, 149, 155–157, 173–175, 182, 184,	<code>\glsentrylongaccess</code> .....	348
	191, 194, 195, 199, 200, 202–204, 206,	<code>\Glsentrylongpl</code> .....	96, 147,
	207, 209, 254–258, 317, 341, 342, 377, 378		155, 225, 226, 230–232, 358, 363, 366–368
<code>\glsdisplay</code> .....	97, 107	<code>\glsentrylongpl</code> .....	
<code>\glsdisplayfirst</code> .....	97, 106		..... 96, 106, 146, 147, 155, 225–227,
<code>\glsdisplaynumberlist</code> .....	175		230–233, 245, 252, 358, 363, 364, 366–370
<code>\glsdohyperlink</code> .....	121, 122	<code>\glsentrylongpluralaccess</code> .....	348
<code>\glsdohypertarget</code> .....	121, 122	<code>\Glsentryname</code> .....	132, 209, 361
<code>\glsdoifexists</code> .....		<code>\glsentryname</code> .....	131, 132, 317, 361
	.. 55, 57, 76–79, 89, 90, 122–128, 140–	<code>\glsentrynumberlist</code> .....	155, 174
	147, 155, 157, 175, 176, 263–267, 358–362	<code>\Glsentryplural</code> ....	98, 102, 130, 350, 354
<code>\glsdoifexistsordo</code> .....	109, 148	<code>\glsentryplural</code> .....	
<code>\glsdoifexistsorwarn</code> .....	202, 209		.. 98, 99, 101, 102, 130, 349, 350, 353, 354
<code>\glsdoifnoexists</code> .....	70, 79	<code>\glsentrypluralaccess</code> .....	347
<code>\glsdonohyperlink</code> .....	111, 121	<code>\Glsentryprefix</code> .....	265
<code>\glsdosanitizesort</code> .....	12	<code>\glsentryprefix</code> .....	263, 266
<code>\glsentryaccess</code> .....	347	<code>\Glsentryprefixfirst</code> .....	265
<code>\glsentrycounter</code> .....	213, 216	<code>\glsentryprefixfirst</code> .....	264, 266
<code>\glsentrycounterfalse</code> .....	11	<code>\Glsentryprefixfirstplural</code> .....	266
<code>\glsentrycounterlabel</code> .....	203, 207	<code>\glsentryprefixfirstplural</code> ....	264, 267
<code>\glsentrycountertrue</code> .....	11	<code>\Glsentryprefixplural</code> .....	265
<code>\glsentrycurrcount</code> .....	92, 94	<code>\glsentryprefixplural</code> .....	264, 267
<code>\Glsentrydesc</code> .....	132, 209, 361	<code>\glsentryprevcount</code> .....	92, 93
<code>\glsentrydesc</code> .....		<code>\Glsentryshort</code> .....	104, 141,
	..... 99–101, 132, 133, 209, 351–353, 361		149, 226, 232, 233, 357–359, 363, 369, 370
<code>\glsentrydescaccess</code> .....	348	<code>\glsentryshort</code> 104, 105, 140, 142, 149, 154,	
<code>\Glsentrydescplural</code> .....	133		223, 225–235, 356–359, 362–365, 367–371
<code>\glsentrydescplural</code> ..	98, 99, 133, 349, 350	<code>\glsentryshortaccess</code> .....	348
<code>\glsentrydescpluralaccess</code> .....	348	<code>\Glsentryshortpl</code> .....	
<code>\Glsentryfirst</code> ...	95, 100, 103, 129, 352, 355		..... 104, 143, 227, 233, 356, 363, 369, 370
<code>\glsentryfirst</code> 95, 99–103, 129, 351, 352, 355		<code>\glsentryshortpl</code> .....	
<code>\glsentryfirstaccess</code> .....	348		..... 104, 106, 142, 143, 155, 225,
<code>\Glsentryfirstplural</code> .....			226, 231–233, 252, 356, 358, 363, 367–370
	..... 96, 99, 102, 131, 350, 354	<code>\glsentryshortpluralaccess</code> .....	348
<code>\glsentryfirstplural</code> .....	96,	<code>\Glsentrysymbol</code> .....	134, 209, 362
	98, 99, 101, 102, 130, 131, 349, 350, 353, 354	<code>\glsentrysymbol</code> .....	99–
<code>\glsentryfirstpluralaccess</code> .....	348		101, 134, 209, 239, 243, 247, 351–353, 362
<code>\glsentryfmt</code> .....	60, 62	<code>\glsentrysymbolaccess</code> .....	348
<code>\Glsentryfull</code> .....	223, 231, 233, 368, 370	<code>\Glsentrysymbolplural</code> .....	135
<code>\glsentryfull</code> .....	223, 231, 233, 367, 370	<code>\glsentrysymbolplural</code> .....	
<code>\Glsentryfullpl</code> ....	223, 232, 233, 368, 370		..... 98, 99, 135, 239, 242, 247, 349, 350
<code>\glsentryfullpl</code> ....	223, 232, 233, 368, 370	<code>\glsentrysymbolpluralaccess</code> .....	348
<code>\glsentryitem</code> 203, 204, 271, 273–278, 280,		<code>\Glsentrytext</code> .....	100, 103, 129, 351, 355
	288, 289, 291, 299, 301, 302, 306, 307,	<code>\glsentrytext</code> .....	99, 100,
	309, 312, 314, 315, 318, 328–335, 337–340		102, 103, 128, 156, 188, 351, 352, 354, 355

<code>\glsentrytextaccess</code> .....	347	<code>\glsinlineparentchildseparator</code>	271, 328
<code>\glsentrytype</code> .....	82	<code>\glsinlinepostchild</code> .....	271, 328
<code>\Glsentryuseri</code> .....	136	<code>\glsinlineseparator</code> .....	271, 328
<code>\glsentryuseri</code> .....	135, 136	<code>\glsinlinesubdescformat</code> .....	272, 328
<code>\Glsentryuserii</code> .....	136	<code>\glsinlinesubnameformat</code> .....	271, 328
<code>\glsentryuserii</code> .....	136, 137	<code>\glsinlinesubseparator</code> .....	272, 328
<code>\Glsentryuseriii</code> .....	137	<code>\glsinsert</code> .....	98–
<code>\glsentryuseriii</code> .....	137, 138		105, 122–127, 140–147, 230, 231, 239,
<code>\Glsentryuseriv</code> .....	138		242, 243, 245, 247, 248, 349–361, 366, 367
<code>\glsentryuseriv</code> .....	138	<code>\glskeylisttok</code> .....	
<code>\Glsentryuserv</code> .....	139		222, 223, 237–246, 248–251, 253, 371–376
<code>\glsentryuserv</code> .....	139	<code>\glslabel</code> .....	80, 98–105, 110, 111, 141–
<code>\Glsentryuservi</code> .....	140		147, 225, 226, 230–232, 238, 239, 242,
<code>\glsentryuservi</code> .....	139, 140		243, 245, 247, 349–358, 362, 363, 366–368
<code>\glsesclocationstrue</code> .....	9	<code>\glslabeltok</code>	222, 237–246, 248–253, 372–375
<code>\glsfieldfetch</code> .....	152	<code>\glslink</code> .....	223, 231, 233, 238, 367, 369
<code>\glsfirstaccessdisplay</code> .....	351, 352, 355	<code>\glslink options</code>	
<code>\glsfirstpluralaccessdisplay</code> .....		counter .....	107, 122, 260
	349, 350, 353, 354	format .....	108, 122, 214
<code>\glsfirstpluralacessdisplay</code> .....	354	hyper .....	108, 110, 122
<code>\glsgenacfmt</code> ....	225, 226, 232, 362, 363, 368	local .....	108
<code>\glsgenentryfmt</code> .....		<code>\glslinkcheckfirsthyperhook</code> .....	110
	225, 226, 231, 232, 236, 238, 241,	<code>\glslinkpostsetkeys</code> .....	111
	242, 245, 247, 250, 252, 362, 363, 367, 368	<code>\glslinkvar</code> .....	108, 109
<code>\glsgetgroupitle</code> .....		<code>\glslistdottedwidth</code> .....	275, 276, 329
	270, 274, 275, 293–298, 313–316, 320	<code>\glslistgroupheaderfmt</code> .....	274, 275
<code>\gls glossarymark</code> .....	40	<code>\glslistnavigationitem</code> .....	274, 275
<code>\gls groupheading</code>	164, 200, 271, 273–275,	<code>\glslocalreset</code> .....	91
	277, 278, 280, 288, 289, 291, 293–300,	<code>\glslocalunset</code> .....	91, 123–127
	302, 306, 307, 309, 312–317, 319, 320, 326	<code>\gls longaccessdisplay</code> .....	358, 360–372
<code>\gls groupskip</code> ...	163, 200, 272, 274, 277,	<code>\gls longkey</code> .....	376
	279, 280, 283, 284, 288–291, 299, 301,	<code>\gls longpluralaccessdisplay</code> .....	
	303, 306, 308, 310, 313, 314, 316, 319, 325		358, 363, 364, 366–370, 372
<code>\glshyperfirstfalse</code> .....	232, 368	<code>\gls longpluralkey</code> .....	376
<code>\glshyperfirsttrue</code> .....	26	<code>\gls longtok</code> .....	
<code>\glshyperlink</code> .....	188		222, 223, 225, 226, 231, 232, 237–
<code>\glshypernavsep</code> .....	270		246, 248–253, 362, 363, 367, 368, 371–376
<code>\glshypernumber</code> .....	39, 216, 217	<code>\glsLTpenaltycheck</code> .....	286, 287
<code>\glsifhyperon</code> .....	108	<code>\gls mcols</code> .....	293–298
<code>\glsIfListOfAcronyms</code> .....	17	<code>\glsnameaccessdisplay</code> .....	361, 362
<code>\glsifplural</code> .....	97, 101, 103,	<code>\glsnamefont</code> .....	208–210, 341, 342, 361
	104, 122–127, 140–147, 230, 239, 242,	<code>\glsnavhyperlink</code> .....	270
	245, 247, 349, 353, 356, 357, 359–361, 366	<code>\glsnavhyperlinkname</code> .....	268, 269
<code>\glsifusetranslator</code> .....	25, 26, 35, 379	<code>\glsnavhypertarget</code> .....	
<code>\glsindexonlyfirstfalse</code> .....	26		274, 275, 294–298, 314–316, 320
<code>\glsinlinedescformat</code> .....	271, 328	<code>\gls navigation</code> .....	
<code>\glsinlinedopostchild</code> .....	271, 328		274, 275, 293–298, 313, 315, 316, 320
<code>\glsinlineemptydescformat</code> .....	271, 328	<code>\gls nextpages</code> .....	9, 65, 190
<code>\glsinlinenameformat</code> .....	271, 328	<code>\glsnogroupskipfalse</code> .....	11

<code>\glsnoidxdisplayloc</code> .....	175–177	<code>\glsseelastsep</code> .....	187
<code>\glsnoidxdisplayloclisthandler</code> ....	175	<code>\glsseelist</code> .....	187
<code>\glsnoidxloclist</code> .....	175, 200	<code>\glsseesep</code> .....	187
<code>\glsnoidxloclisthandler</code> .....	201	<code>\glssetexpandfield</code> .....	20, 23, 24
<code>\glsnoidxnumberlistloophandler</code> ....	175	<code>\glssetnoexpandfield</code> .....	20, 21, 23, 24
<code>\glsnoidxstripaccents</code> .....	22	<code>\GlsSetQuote</code> .....	80, 164
<code>\glsnomakeindexwarning</code> .....	166	<code>\glssettoctitle</code> .....	35, 190
<code>\glsnonextpages</code> .....	65, 190	<code>\glsshortaccessdisplay</code> .....	356–359, 362–365, 367–372
<code>\glsnopostdotfalse</code> .....	11	<code>\glsshortkey</code> .....	376
<code>\glsnoxindywarning</code> 38, 44–46, 49–51, 159, 160		<code>\glsshortpluralaccessdisplay</code> .....	356, 358, 363, 367–370, 372
<code>\glsnumberformat</code> .....	155	<code>\glsshortpluralkey</code> .....	376
<code>\glsnumberlistloop</code> .....	175	<code>\glsshorttok</code> .....	222, 223, 237–246, 248–253, 372–376
<code>\glsnumbersgroupname</code> .....	30, 36, 212	<code>\glsshowtarget</code> .....	6
<code>\glsnumlistlastsep</code> .....	156, 175	<code>\glssortnumberfmt</code> .....	13, 14
<code>\glsnumlistparser</code> .....	156, 172	<code>\glsspace</code> .....	219
<code>\glsnumlistsep</code> .....	156, 175	<code>\glstepentry</code> .....	203, 207
<code>\glsopenbrace</code> ....	48, 49, 163, 164, 325, 326	<code>\glstepsubentry</code> .....	204, 207
<code>\glsorder</code> .....	27, 170, 171, 196	<code>\glssubentrycounterfalse</code> .....	11
<code>\glsorg@endtheglossary</code> .....	5	<code>\glssubentrycounterlabel</code> .....	204, 207
<code>\glsorg@PrintChanges</code> .....	5	<code>\glssubentryitem</code> .....	204, 272, 273, 275–278, 280, 288, 289, 291, 299, 301, 302, 306, 308, 309, 313, 314, 316, 318, 328–335, 337–340
<code>\glsorg@theglossary</code> .....	5	<code>\glssymbolaccessdisplay</code> ....	351–353, 362
<code>\glspagelistwidth</code> 278, 279, 281, 282, 285, 286, 289–292, 300–302, 304, 305, 307–311		<code>\glssymbolpluralaccessdisplay</code> .	349, 350
<code>\glspatchLToutput</code> .....	283–285	<code>\glssymbolsgroupname</code> .....	30, 36, 212
<code>\glspenaltygroupskip</code> .....	283, 284	<code>\glstarget</code> .....	210, 211, 272–278, 280, 288, 289, 291, 299, 301, 302, 306–309, 312–316, 318, 319, 328–340
<code>\glsperscentchar</code> .....	71, 72, 162, 164	<code>\glstextaccessdisplay</code> ..	351, 352, 354, 355
<code>\Glspl</code> .....	96, 236	<code>\glstextformat</code> .....	109, 111
<code>\glspl</code> .....	96, 236	<code>\glstextup</code> .....	34, 370
<code>\glspluralaccessdisplay</code> 349, 350, 353, 354		<code>\glstildechar</code> .....	45, 162, 163
<code>\glspluralsuffix</code> .....	34, 83, 84, 225–227, 363, 364, 368–370	<code>\glstranslatefalse</code> .....	25, 26
<code>\glspostdescription</code> .....	36, 272–275, 277, 288, 299, 306, 312–314, 316, 318, 319, 328–331, 333–337, 339	<code>\glstranslatetrue</code> .....	25, 26
<code>\glspostinline</code> .....	271	<code>\glstreechildpredesc</code> .....	313, 314
<code>\glspostlinkhook</code> .....	110, 123–128, 141–147, 359–361	<code>\glstreegroupheaderfmt</code> .....	293–298, 313, 315, 316, 320
<code>\glsprestandardsort</code> .....	12	<code>\glstreeindent</code> ..	314, 316, 318, 319, 334–336
<code>\glsreset</code> .....	91	<code>\glstreeitem</code> .....	293, 294, 312
<code>\glsresetentrycounter</code> .....	203, 206	<code>\glstreenamebox</code> .....	318, 319
<code>\glsresetentrylist</code> ....	163, 198, 205, 325	<code>\glstreenamefmt</code> .....	311–315, 317–319
<code>\glsresetsubentrycounter</code> 204, 207, 271, 328		<code>\glstreenavigationfmt</code> .....	293–298, 313, 315, 316, 320
<code>\glssanitizesortfalse</code> .....	23, 24	<code>\glstreepredesc</code> .....	312, 314, 316
<code>\glssanitizesorttrue</code> .....	23, 24	<code>\glstreesubitem</code> .....	293, 312
<code>\glssavenumberlistfalse</code> .....	9		
<code>\glssavewritesfalse</code> .....	29		
<code>\glsseeformat</code> .....	162, 174–176, 325		
<code>\glsseesep</code> .....	187		
<code>\glsseesepformat</code> .....	188		

<code>\glstreesubsubitem</code> .....	293, 312	<code>\ifcsstrequal</code> .....	79
<code>\glstype</code> .	110, 111, 122–127, 140–147, 359–361	<code>\ifcsstring</code> .....	78
<code>\glsucmarkfalse</code> .....	11	<code>\ifcsundef</code> .....	7, 13, 28, 32, 35, 38, 40, 42, 53, 60, 62, 64, 82, 84, 92, 107, 112, 113, 121, 170, 177, 188, 191, 194, 202, 208, 212–215, 217, 224, 270, 327
<code>\glsucmarktrue</code> .....	11	<code>\ifdef</code> .....	57, 66, 71, 72, 89, 108, 148, 152, 174, 175, 218, 311, 312
<code>\glsunset</code> .....	89, 91, 93, 123–127	<code>\ifdefempty</code> .....	18, 41, 52, 56, 57, 62, 97, 101, 103, 172, 199, 200, 222, 224, 230, 238, 242, 244, 247, 349, 353, 356, 366
<code>\glsupacrpluralsuffix</code> .....	227, 234, 240, 244, 246, 249	<code>\ifdefequal</code> ....	55–58, 68, 69, 73, 82, 86, 200
<code>\GlsUseAcrEntryDispStyle</code> .....	224, 227–230, 232–234, 364, 365, 368, 370, 371	<code>\ifdefstrequal</code> .....	79
<code>\GlsUseAcrStyleDefs</code> .....	224, 227–230, 232–235, 364, 365, 368, 370, 371	<code>\ifdefstring</code> .....	10, 35, 59, 170, 172, 195–197, 199, 201
<code>\glswrallowprimitivemodstrue</code> .....	182	<code>\ifdefvoid</code> .....	21, 86, 199, 200
<code>\glswrite</code> ...	160–165, 171, 177, 178, 323–327	<code>\ifdim</code> .....	226, 286, 317
<code>\glswritedefhook</code> .....	72	<code>\iffalse</code> .....	85, 90
<code>\glswriteentry</code> .....	180	<code>\IfFileExists</code> .....	10, 25, 191
<code>\glswritefiles</code> .....	29, 177	<code>\ifglossaryexists</code> ..	39, 51, 55, 169, 170, 190
<code>\glsxindyfalse</code> .....	28	<code>\ifgls@sanitize@description</code> .....	23
<code>\glsxindytrue</code> .....	28	<code>\ifgls@sanitize@name</code> .....	23
<b>H</b>			
<code>\H</code> .....	22	<code>\ifgls@sanitize@symbol</code> .....	23
<code>\hangindent</code> .....	297, 298, 311, 314–316, 318–320, 333–336	<code>\ifgls@xindy@glslnumbers</code> .....	51
<code>\hbox</code> .....	89, 275, 276, 329	<code>\ifglsacrdescription</code> .....	251
<code>\hfill</code> .....	275, 276, 329	<code>\ifglsacrdua</code> .....	240, 247, 249, 251, 252
<code>\hline</code> ...	277–279, 281, 288–290, 292, 299–311	<code>\ifglsacrfootnote</code> .....	110, 251
<code>\hsize</code> .....	276, 287, 298, 305	<code>\ifglsacrronym</code> .....	15, 16
<code>\hspace</code> .....	312	<code>\ifglsacrshortcuts</code> .....	31, 236
<code>\hss</code> .....	275, 276, 329	<code>\ifglsacrsmalldcaps</code> .	240, 241, 244, 246, 249
<code>\ht</code> .....	286	<code>\ifglsacrsmaller</code> .....	240, 242, 244, 246
<code>\hyperdef</code> .....	32	<code>\ifglsautomake</code> .....	29, 172
<code>\hyperlink</code> .....	108, 121, 216	<code>\ifglsdescsuppressed</code> .....	271
<code>hyperref</code> package .....	185, 188, 215, 260	<code>\ifglsentrycounter</code> .....	203–207
<code>\hypertarget</code> .....	121	<code>\ifglsentryexists</code> .....	54, 70, 71, 80, 82
<b>I</b>			
<code>\IeC</code> .....	22	<code>\ifglsesclocations</code> .....	181
<code>\if</code> .....	113, 185, 322	<code>\ifglshaschildren</code> .....	271, 328
<code>\if@endfor</code> .....	269	<code>\ifglshasdesc</code> .....	271
<code>\if@gl@debug</code> .....	5, 19, 179	<code>\ifglshaslong</code> .....	95, 96, 149, 225, 226, 230, 232, 245, 362, 363, 366, 368
<code>\if@gl@docloaded</code> .....	4, 15, 178	<code>\ifglshasparent</code> .....	194, 199, 317
<code>\if@gl@isacronymlist</code> .....	110	<code>\ifglshasprefix</code> .....	265
<code>\if@openright</code> .....	42	<code>\ifglshasprefixfirst</code> .....	265
<code>\ifbool</code> .....	16, 27, 29, 53, 98, 100	<code>\ifglshasprefixfirstplural</code> .....	265
<code>\ifboolexpr</code> .....	35, 59, 212	<code>\ifglshasprefixplural</code> .....	265
<code>\ifcase</code> .....	5, 6, 8, 25, 65, 202, 313, 333	<code>\ifglshassymbol</code> .....	238, 242, 247, 312–314, 316, 318, 319
<code>\ifcsdef</code> .....	25, 36, 42, 68, 69, 75– 79, 106, 107, 179, 193, 196–198, 213, 224	<code>\ifglshyperfirst</code> .....	110
<code>\ifcsempy</code> .....	55, 56, 263	<code>\ifglslindexonlyfirst</code> .....	180
<code>\ifcsequal</code> .....	56		

<code>\ifglsnogroupskip</code> .....	274, 277, 279, 280, 283, 284, 288, 289, 291, 299, 301, 303, 306, 308, 310, 313, 314, 316, 319
<code>\ifglsnonumberlist</code> .....	204
<code>\ifglsnopostdot</code> .....	10
<code>\ifglslnumberline</code> .....	43
<code>\ifglssanitizesort</code> .....	21, 23, 24
<code>\ifglssavenumberlist</code> .....	68, 172, 188
<code>\ifglssavewrites</code> .....	29, 169, 179
<code>\ifglssubentrycounter</code> .....	204, 206, 207
<code>\ifglstoc</code> .....	42
<code>\ifglstranslate</code> .....	34
<code>\ifglsucmark</code> .....	40, 41
<code>\ifglused</code> .....	94, 98–103, 110, 158, 180, 238, 242, 245, 247, 263–267, 349–356
<code>\ifglswrallowprimitivemods</code> .....	183
<code>\ifglsxindy</code> .....	.. 36–38, 43–46, 48–51, 61, 86, 87, 114, 159, 160, 166, 170, 184, 186, 191, 321–323
<code>\ifignoredglossary</code> .....	82, 85, 179
<code>\ifin@</code> .....	31
<code>\ifinlistcs</code> .....	197, 202
<code>\ifinner</code> .....	6
<code>\ifKV@glslink@hyper</code> .....	110, 111
<code>\ifKV@glslink@local</code> .....	123–127
<code>\ifmeasuring@</code> .....	88
<code>\ifmmode</code> .....	6
<code>\ifnum</code> 12, 13, 93, 199, 286, 314, 316, 318, 334, 335	
<code>\ifstrempy</code> .....	328
<code>\ifstrequal</code> .....	212
<code>\ifthenelse</code> 24, 32, 42, 112, 171, 212, 251, 269	
<code>\IfTrackedLanguage</code> .....	166
<code>\IfTrackedLanguageFileExists</code> 35, 379, 380	
<code>\iftrue</code> .....	85, 90
<code>\ifundef</code> .....	60, 71, 81, 160, 164, 171, 204
<code>\ifvmode</code> .....	157
<code>\ifvoid</code> .....	286
<code>\ifx</code> .....	12, 14, 16, 17, 31, 32, 43, 45, 47, 48, 51, 52, 82– 85, 87, 88, 112, 113, 115–120, 149, 160, 163, 165, 166, 168, 177, 181, 183–187, 189, 190, 203, 205, 213, 215, 216, 238, 240, 242, 244, 246, 247, 249, 251, 253, 254, 323–325, 327, 328, 333–336, 341, 347
<code>\immediate</code> .....	71, 72, 94, 169, 178, 192
<code>\in@</code> .....	31
<code>\indexname</code> .....	30
<code>\indexspace</code> .	274, 293–298, 313–316, 319, 320
<code>\input</code> .....	34, 97
<code>\inputencodingname</code> .....	28
<code>\InputIfFileExists</code> .....	71
<code>\istfilename</code> .	36, 160, 164, 170, 171, 323, 326
<code>\item</code> 273–276, 293, 294, 312, 313, 328, 329, 333	
<b>J</b>	
<code>\jobname</code> 37, 71, 160, 164, 169, 170, 191, 323, 326	
<b>K</b>	
<code>\key@ifundefined</code> .....	73, 74
<code>\KV@glslink@hyperfalse</code> .....	108, 110, 121
<code>\KV@glslink@hypertrue</code> .....	108, 121
<b>L</b>	
<code>\L</code> .....	22
<code>\l</code> .....	22
<code>\label</code> .....	8, 203, 204, 206, 207
<code>\language name</code> .....	28
<code>\leaders</code> .....	275, 276, 329
<code>\leavevmode</code> .....	80, 111
<code>\let</code> .....	5, 10, 12–16, 22, 25, 26, 29– 32, 35, 36, 45, 46, 57, 58, 68, 70, 80–85, 88–90, 92, 94, 97, 109, 111, 114, 121– 128, 140–147, 149, 155, 156, 164, 165, 169, 170, 172–176, 179–181, 183, 187, 189–191, 200, 202, 205, 210, 222, 235– 237, 239–251, 261, 269, 270, 286, 293, 294, 312, 327, 344, 359–361, 372–376, 379
<code>\letcs</code> .....	55– 57, 72, 74, 75, 78, 83, 84, 148, 149, 170, 174, 175, 193–195, 199, 200, 209, 212, 317
<code>link text</code> .....	<u>97</u>
<code>\listcsadd</code> .....	197
<code>\listcsgadd</code> .....	202
<code>\listcsxadd</code> .....	193
<code>\listead</code> .....	197
<code>\loadglsentries</code> .....	97
<code>\long</code> .....	80, 216
<code>\longnewglossaryentry</code> .....	80
<code>longtable package</code> .....	276, 283, 287
<code>\LT@end@open</code> .....	286
<code>\LT@err</code> .....	286
<code>\LT@foot</code> .....	286, 287
<code>\LT@head</code> .....	286, 287
<code>\LT@lastfoot</code> .....	286
<code>\LT@output</code> .....	286
<b>M</b>	
<code>\makeatletter</code> .....	71, 191
<code>\makebox</code> ....	275, 276, 317–319, 329, 335, 336

makeglossaries	27, 37, 50, 51, 59, 166, 171, 191	\newcounter	203–206
\makeglossaries	6, 7, 29, 33, 65, 173, 174, 176, 192	\newenvironment	208
\makeglossary	169, 172	\newglossary	15, 16, 30, 61, 171
makeindex	381	\newglossaryentry	6, 31, 67, 70, 92, 222, 237, 239, 241, 243, 245, 248, 250, 252, 372–375
makeindex	9, 12, 28, 29, 33, 37, 39, 43, 59, 61, 63, 88, 113, 116, 158, 162, 164, 166, 169, 178, 182–185, 211, 322, 323	\newglossaryentry options	
delim_n	39	access	344, 345
delim_r	39	counter	64
page_compositor	37	description	27, 62, 63, 67, 70, 80, 132, 150, 218, 246, 343
special characters	114, 115, 158	descriptionaccess	346, 348
\makenoidxglossaries	6, 7, 65, 172, 176	descriptionplural	133, 343
\MakeTextUppercase	4	descriptionpluralaccess	346, 348
\MakeUppercase	350, 352, 359, 361	first	63, 84, 122, 129, 151, 244, 249, 342
\marginpar	6	firstaccess	346, 348
\markboth	40	firstplural	63, 130, 151, 343
\mbox	157, 274, 297, 298, 317, 329	firstpluralaccess	346, 348
memoir class	178	format	160
\memUHead	40	long	103, 154, 343
\MessageBreak	19, 59, 189, 190, 341, 379, 380	longaccess	347, 348
mfistuc package	1	longplural	154, 343
\mfistucMakeUppercase	4, 40, 41, 76, 99–105, 128–140, 142, 143, 145, 147, 223, 230, 231, 233, 243, 248, 266, 267, 354–358, 366, 367, 369, 370	longpluralaccess	347, 348
\midrule	283, 284	name	62, 63, 67, 70, 80, 131, 148, 188, 342
\month	160, 164, 323, 326	nonumberlist	65, 66
multicol package	292	parent	65, 70
		plural	63, 84, 129, 343
		pluralaccess	346, 347
		prefix	261
		prefixfirst	261
		prefixfirstplural	262
		prefixplural	262
		see	6, 9, 64, 70, 172, 173
		short	103, 154, 343
		shortaccess	346, 348
		shortplural	154, 343
		shortpluralaccess	346, 348
		sort	63, 152, 179, 211
		symbol	62, 64, 134, 240, 241, 244, 249, 280, 302, 342–344
		symbolaccess	346, 348
		symbolplural	134, 343
		symbolpluralaccess	346, 348
		text	63, 122, 128, 150, 240, 244, 342
		textaccess	346, 347
		type	15, 64, 97, 152
		user1	135, 152, 344
		user2	136, 152
		user3	137, 153
N			
\n	164, 165, 326		
\NeedsTeXFormat	4, 261, 321, 327, 341, 379		
\new@glossaryentry	70, 174		
\new@ifnextchar	59, 75, 76, 95, 96, 122–126, 128–147, 219–221, 263–266		
\newacronym	217, 222, 238, 240, 242, 244, 246, 249, 251, 253		
\newacronymhook	222, 238, 240, 242, 244, 246, 249, 251, 253, 371		
\newacronymstyle	225–230, 232–234		
\newcommand	6–22, 24–27, 29–34, 36–46, 48–62, 64–80, 86–92, 94–97, 101, 103, 105–110, 112, 114, 120–160, 165, 166, 168–171, 173, 176–181, 183–189, 191, 193–199, 201, 205–214, 216–226, 235–259, 262–266, 268–270, 272, 273, 286, 293, 311, 312, 317, 327, 345–347, 362, 376–378		
\newcount	13, 68		







sanitizesort .....	20	toctitle .....	202
savewrites .....	29, 385	type .....	15, 188, 202
false .....	169	\printindex .....	30
true .....	171, 177	\printnoidxglossaries .....	174
section .....	7, 41	\printnoidxglossary .....	173, 174, 176, 189, 196, 197, 205
sort		\printnoidxglossary options	
def .....	11, 12	sort .....	204
none .....	11	\printnumbers .....	30
standard .....	11	\printsymbols .....	30
use .....	11, 12, 404	\ProcessOptions .....	261, 341
style .....	8, 253, 254	\ProcessOptionsX .....	31
subentrycounter .....	204, 206	\protect .....	43, 105, 106, 225–227, 232, 233, 239, 243, 245, 358, 362, 363, 368, 369
toc .....	7	\protected@edef .....	8, 45, 47, 50, 52, 82, 86, 87, 98, 100, 106, 112, 113, 155, 179, 182, 184, 203, 208, 210, 213, 222, 247, 252, 262, 268, 322, 323, 341, 342, 347
true .....	7	\protected@write .....	58, 60, 160–162, 171, 173, 177, 179, 188, 269, 323
translate .....	25	\protected@xdef .....	12–14, 17, 22, 69, 87, 88, 184, 345
false .....	25	\providecommand .....	16, 33, 34, 41, 58, 94, 122, 162, 171, 174, 192, 208, 210, 273, 293, 311
translator .....	24	\ProvidesFile .....	34
xindy .....	28, 162, 260	\ProvidesPackage .....	4, 261, 268, 270, 273, 276, 282, 287, 292, 298, 305, 311, 321, 327, 341, 379
\PackageError .....	6, 7, 14, 29, 33, 44, 51, 54, 55, 59, 64, 67, 68, 74–79, 81–83, 92, 107, 148, 168, 169, 171, 174, 176, 196–198, 202, 205, 213, 214, 224, 240–242, 247, 249, 250, 327, 349		
\PackageInfo .....	5, 6, 169, 179		
\PackageWarning .....	5, 18		
\PackageWarningNoLine ...	5, 6, 19, 379, 380		
\pagegoal .....	286		
\pagelistname .....	36, 279, 281, 283, 284, 290, 292, 301–305, 308–311		
\par .....	36, 210, 211, 273–275, 293, 295–298, 311, 312, 314–320, 329, 334–336		
\parindent .....	293–298, 312, 314–316, 318–320, 334–336		
\parskip .....	293–296, 312, 314, 315		
\PassOptionsToPackage .....	261, 341		
\penalty .....	286		
\phantomsection .....	41		
polyglossia package .....	24, 34		
\printglossaries .....	172		
\printglossary .....	16, 19, 30, 172, 189, 205		
\printglossary options			
entrycounter .....	203		
nogroupskip .....	203		
nonumberlist .....	204		
nopostdot .....	203		
numberedsection .....	202		
style .....	202		
subentrycounter .....	204		
title .....	202		

## R

\r .....	22
\raggedright .....	285–292, 306–311
\raisebox .....	121
\ref .....	207
\refstepcounter .....	203, 204, 206, 207
\relax .....	5, 8, 10, 13–16, 25, 26, 31, 46, 58, 64, 65, 69, 71, 82, 83, 85, 89, 93, 94, 109, 112–120, 149, 163–166, 168, 169, 172–175, 180, 184–187, 189, 191, 199, 202, 212, 213, 254, 269, 273, 286, 293, 297, 298, 311, 313–320, 322, 325–327, 333–336, 344, 359, 360, 372–374, 376
\renewacronymstyle ..	362–366, 368, 370, 371
\renewcommand ..	4–11, 14–16, 18–20, 23, 25, 27, 29, 31, 35–38, 51, 62, 65, 66, 80, 92–94, 155, 157, 159, 160, 165, 167, 171–175, 178, 191, 192, 202–204, 222, 223, 225–235, 238, 240, 242, 244, 246, 249, 251, 253, 271–281, 283, 284, 286–303,

306–310, 312–322, 327–335, 337–340, 344, 349, 353, 356, 358, 361–365, 367–375	\settowidth ..... 226, 317–319, 335
\renewenvironment ..... 208, 270, 273, 277–282, 285–312, 314, 315, 317	\sfcode ..... 10
\RequireGlossariesLang ..... 35, 379, 380	\show ..... 254–259, 377, 378
\RequirePackage ..... .... 4, 9, 10, 25, 31, 34, 253, 260, 261, 276, 282, 283, 287, 293, 298, 305, 342, 379	\small ..... 6
\restorecounters@ ..... 112	\SmallNewAcronymDef ..... 249
\romannumeral ... 180, 183, 184, 317–319, 335	\space ..... 6, 7, 29, 33, 44, 48, 49, 51, 52, 65, 67, 92, 95, 96, 105–108, 155, 161–164, 168, 170– 172, 174, 176, 187, 190, 192, 203, 204, 207, 213, 219, 225–235, 243, 247, 270, 272–275, 277, 288, 299, 306, 312–314, 316, 318, 319, 321, 322, 324–326, 328– 331, 333–337, 339, 358, 362–365, 367–372
<b>S</b>	\spacefactor ..... 10
\s@glshyp@opt ..... 109	\SS ..... 22
\s@GlsSetXdyFirstLetterAfterDigits 159	\ss ..... 22
\s@GlsSetXdyNumberGroupOrder ..... 159	\string ..... 6, 7, 14, 19, 29, 33, 43–45, 47–52, 58, 60, 65, 67, 71, 72, 75, 76, 86–88, 92, 94–96, 106–108, 113, 114, 116, 117, 119, 155, 158–166, 168, 169, 171–174, 176, 177, 185, 186, 190, 192, 196, 197, 205, 210, 211, 213, 269, 321–327
\s@newglossary ..... 59	\strut ..... 211, 274– 278, 280, 288, 289, 291, 299, 301, 302, 306, 308, 309, 316, 328–332, 334, 337–340
\savecounters@ ..... 112	\subglossentry ..... .... 87, 190, 191, 200, 210, 211, 271, 273, 275–278, 280, 288, 289, 291, 299, 301, 302, 306, 308, 309, 313, 314, 316, 318
\seenname ..... 187	\subitem ..... 293, 312, 313, 333
\SetAcronymStyle ..... 27	subsubitem ..... 293, 312, 313, 333
\setbool ..... 24	supertabular package ..... 10, 253, 298, 305
\setbox ..... 286, 287	\symbolname ..... . 36, 280, 281, 284, 292, 303–305, 310, 311
\setcounter ..... 203, 204, 206	<b>T</b>
\SetCustomDisplayStyle ..... 253	\t ..... 22
\SetDefaultAcronymDisplayStyle 237, 238	\tablehead ..... 299–311
\SetDefaultAcronymStyle ..... 251	\tabletail ..... 299–311
\SetDescriptionAcronymDisplayStyle 244	\tabularnewline ..... 277–281, 283, 284, 288–292, 299–311, 331, 332, 337, 338
\SetDescriptionAcronymStyle ..... 251	\texorpdfstring ..... 152
\SetDescriptionDUAAcronymDisplayStyle ..... 242	\textbar ..... 270
\SetDescriptionDUAAcronymStyle ... 251	\textbf ..... 210, 216, 311, 333–336
\SetDescriptionFootnoteAcronymDisplayStyle ..... 240	textcase package ..... 4
\SetDescriptionFootnoteAcronymStyle 251	\textit ..... 217
\SetDUADisplayStyle ..... 251	\textmd ..... 216
\SetDUASyle ..... 252	\textrm ..... 216
\setentrycounter ..... 45, 162, 201, 321	\textsc .. 217, 227, 233, 240, 244, 246, 249, 370
\SetFootnoteAcronymDisplayStyle ... 246	
\SetFootnoteAcronymStyle ..... 251	
\SetGenericNewAcronym ..... 224	
\setglossarystyle ..... 190, 213, 254, 274–286, 288–310, 313–316, 319, 320	
\setglossentrycompatibility ... 202, 213	
\setkeys .. 24, 28, 31, 41, 81, 111, 157, 190, 222, 238, 240, 242, 244, 246, 249, 251, 253	
\setlength ..... 276, 287, 293– 296, 298, 305, 312, 314, 315, 319, 335, 336	
\SetSmallAcronymDisplayStyle ..... 249	
\SetSmallAcronymStyle ..... 252	
\settoheight ..... 121	

