

# Documented Code For glossaries v4.46

Nicola L.C. Talbot

Dickimaw Books

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

2020-03-19

This is the documented code for the `glossaries` package. This bundle comes with the following documentation:

`glossariesbegin.pdf` If you are a complete beginner, start with “The `glossaries` package: a guide for beginners”.

`glossary2glossaries.pdf` If you are moving over from the obsolete `glossary` package, read “Upgrading from the `glossary` package to the `glossaries` package”.

`glossaries-user.pdf` For the main user guide, read “`glossaries.sty` v4.46:  $\text{\LaTeX}2\text{e}$  Package to Assist Generating Glossaries”.

`mfirstuc-manual.pdf` The commands provided by the `mfistuc` package are briefly described in “`mfistuc.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.md** 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 Main Package Code</b>	<b>4</b>
1.1 Package Definition . . . . .	4
1.2 Package Options . . . . .	5
1.3 Predefined Text . . . . .	39
1.4 Xindy . . . . .	49
1.5 Loops and conditionals . . . . .	58
1.6 Defining new glossaries . . . . .	65
1.7 Defining new entries . . . . .	69
1.8 Resetting and unsetting entry flags . . . . .	95
1.9 Keeping Track of How Many Times an Entry Has Been Unset . . . . .	98
1.10 Loading files containing glossary entries . . . . .	103
1.11 Using glossary entries in the text . . . . .	104
1.12 Adding an entry to the glossary without generating text . . . . .	164
1.13 Creating associated files . . . . .	165
1.14 Writing information to associated files . . . . .	186
1.15 Glossary Entry Cross-References . . . . .	195
1.16 Displaying the glossary . . . . .	197
1.17 Acronyms . . . . .	225
1.18 Predefined acronym styles . . . . .	230
1.19 Predefined Glossary Styles . . . . .	261
1.20 Debugging Commands . . . . .	262
1.21 Compatibility with version 2.07 and below . . . . .	267
<b>2 Prefix Support (glossaries-prefix Code)</b>	<b>269</b>
<b>3 Glossary Styles</b>	<b>276</b>
3.1 Glossary hyper-navigation definitions (glossary-hypernav package) . . . . .	276
3.2 In-line Style (glossary-inline.sty) . . . . .	278
3.3 List Style (glossary-list.sty) . . . . .	281
3.4 Glossary Styles using longtable (the glossary-long package) . . . . .	284
3.5 Glossary Styles using longtable and booktabs (the glossary-longbooktabs) package . . . . .	290
3.6 Glossary Styles using longtable (the glossary-longragged package) . . . . .	295
3.7 Glossary Styles using multicol (glossary-mcols.sty) . . . . .	300
3.8 Glossary Styles using supertabular environment (glossary-super package) . . . . .	306
3.9 Glossary Styles using supertabular environment (glossary-superragged package) . . . . .	313
3.10 Tree Styles (glossary-tree.sty) . . . . .	319

<b>4 Backwards Compatibility</b>	<b>329</b>
4.1 <code>glossaries-compatible-207</code> . . . . .	329
4.2 <code>glossaries-compatible-307</code> . . . . .	335
<b>5 Accessibility Support (<code>glossaries-accsupp</code> Code)</b>	<b>349</b>
5.1 Defining Replacement Text . . . . .	351
5.2 Accessing Replacement Text . . . . .	355
5.3 Displaying the Glossary . . . . .	380
5.4 Acronyms . . . . .	381
5.5 Debugging Commands . . . . .	395
<b>6 Multi-Lingual Support</b>	<b>398</b>
6.1 Polyglossia Captions . . . . .	398
<b>Glossary</b>	<b>400</b>
<b>Change History</b>	<b>401</b>
<b>Index</b>	<b>427</b>

# 1 Main Package Code

## 1.1 Package Definition

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

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries}[2020/03/19 v4.46 (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*{\mfistuc}{\MakeTextUppercase}{\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 (this is now redundant as datatool-base loads 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{nlectdoc}{\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

\PprintChanges Now redefine \PrintChanges so that it uses the original theglossary environment.
23 \let\glsorg@PrintChanges\PrintChanges
24 \renewcommand{\PrintChanges}{%
25   \begingroup
26   \let\theglossary\glsorg@theglossary
27   \let\endtheglossary\glsorg@endtheglossary
28   \glsorg@PrintChanges
29   \endgroup
30 }
```

End of doc stuff.

```
31 \fi
```

## 1.2 Package Options

debug Switch on debug mode. This will also cancel the nowarn option. This is now a choice key.

```
32 \newif\if@gls@debug
33 \define@choicekey{glossaries.sty}{debug}[\gls@debug@val\gls@debug@nr]{%
34 {true,false,showtargets,showaccsupp}[true]{%
35 \ifcase\gls@debug@nr\relax
36   % debug=true
37   \@gls@debugtrue
38   \renewcommand*\GlossariesWarning[1]{%
39     \PackageWarning{glossaries}{##1}%
40   }%
41   \renewcommand*\GlossariesWarningNoLine[1]{%
42     \PackageWarningNoLine{glossaries}{##1}%
43   }%
44   \let\@glsshowtarget\@gobble
45   \PackageInfo{glossaries}{debug mode ON (nowarn option disabled)}%
46 \or
47   % debug=false
48   \@gls@debugfalse
49   \let\@glsshowtarget\@gobble
50   \let\@glsshowaccsupp\@gobblethree
51   \PackageInfo{glossaries}{debug mode OFF}%
52 }
```

```

52 \or
53   % debug=showtargets
54   \@gls@debugtrue
55   \renewcommand*{\GlossariesWarning}[1]{%
56     \PackageWarning{glossaries}{##1}%
57   }%
58   \renewcommand*{\GlossariesWarningNoLine}[1]{%
59     \PackageWarningNoLine{glossaries}{##1}%
60   }%
61   \PackageInfo{glossaries}{debug mode ON (nowarn option disabled)}%
62   \renewcommand{\@glsshowtarget}{\@glsshowtarget}%
63 \or
64   % debug=showaccsupp
65   \@gls@debugtrue
66   \renewcommand*{\GlossariesWarning}[1]{%
67     \PackageWarning{glossaries}{##1}%
68   }%
69   \renewcommand*{\GlossariesWarningNoLine}[1]{%
70     \PackageWarningNoLine{glossaries}{##1}%
71   }%
72   \PackageInfo{glossaries}{debug mode ON (nowarn option disabled)}%
73   \renewcommand{\@glsshowaccsupp}{\glsshowaccsupp}%
74 \fi
75 }

```

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

```

76 \newcommand*{\glsshowtarget}[1]{%
77   \ifmmode
78     \nfss@text{\glsshowtargetfont [#1]}%
79   \else
80     \ifinner

```

Grouping no longer required as new \@@glsshowtarget adds scoping but retained here in case any existing documents are using \glsshowtarget elsewhere.

```

81   {\glsshowtargetfont [#1]}%
82   \else
83     \glsshowtargetouter{#1}%
84   \fi
85 \fi
86 }

```

showtargetouter

```

87 \newcommand*{\glsshowtargetouter}[1]{%
88   \glsshowtargetsymbol\marginpar{\glsshowtargetsymbol\glsshowtargetfont #1}}%

```

howtargetsymbol

```

89 \newcommand*{\glsshowtargetsymbol}{\tiny$\triangleright$}
```

sshowtargetfont

```

90 \newcommand*{\glsshowtargetfont}{\ttfamily\footnotesize}
```

```

\@glsshowtarget debug=showtargets will redefine this.
91 \newcommand*{\@glsshowtarget}[1]{}

@@glsshowtarget Need to detokenize the label in the event that it contains awkward characters like underscores.
92 \newrobustcmd*{\@glsshowtarget}[1]{%
93   \begingroup
94   \protected@edef\gls@tmp{#1}%
95   \conelevel@sanitize\gls@tmp
96   \expandafter\glsshowtarget\expandafter{\gls@tmp}%
97   \endgroup
98 }

@glsshowaccsupp debug=showaccsupp will redefine this.
99 \newcommand*{\@glsshowaccsupp}[3]{}

\glsshowaccsupp Just use \@@glsshowtarget since it basically needs to do the same thing.
100 \newrobustcmd*{\glsshowaccsupp}[3]{%
101   \ifstrempty{#1}{%
102     {\@glsshowtarget{/#2 (#3)}%}
103     {\@glsshowtarget{/#2 (#3) [#1]}%}
104   }
}

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

gls@see@noindex
105 \newcommand*{\gls@see@noindex}{%
106   \PackageError{glossaries}{%
107     {‘\gls@xr@key’ key may only be used after \string\makeglossaries\space
108     or \string\makenoidxglossaries\space (or move
109     \string\newglossary\space
110     definitions into the preamble)}%
111     {You must use \string\makeglossaries\space
112     or \string\makenoidxglossaries\space before defining
113     any entries that have a ‘\gls@xr@key’ key. It may
114     be that the ‘see’ key has been written to the .glsdefs
115     file from the previous run, in which case you need to
116     move your definitions
117     to the preamble if you don’t want to use
118     \string\makeglossaries\space
119     or \string\makenoidxglossaries}%
120   }
}

seenoindex
121 \define@choicekey{glossaries.sty}{seenoindex}{%
122   [\gls@seenoindex@val\gls@seenoindex@nr]{error,warn,ignore}{%
123   \ifcase\gls@seenoindex@nr

```

```

124 \renewcommand*{\@gls@see@noindex}{%
125   \PackageError{glossaries}%
126   {`\gls@xr@key' key may only be used after \string\makeglossaries\space%
127   or \string\makenoidxglossaries\%}
128   {You must use \string\makeglossaries\space%
129   or \string\makenoidxglossaries\space before defining%
130   any entries that have a `\gls@xr@key' key}%
131 }%
132 \or
133 \renewcommand*{\@gls@see@noindex}{%
134   \GlossariesWarning{`\gls@xr@key' key ignored}%
135 }%
136 \or
137 \renewcommand*{\@gls@see@noindex}{}%
138 \fi
139 }

```

**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.

```
140 \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.

```
141 \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.

```

142 \ifcsundef{chapter}%
143   {\newcommand*{\@glossarysec}{section}}%
144   {\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.

```

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

```

Determine whether or not to use numbered sections.

**glossarysecstar**

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

**lossaryseclabel**

```
149 \newcommand*{\@glossaryseclabel}{}%
```

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

```
150 \newcommand*{\glsautoprefix}{}%
```

```

numberedsection
151 \define@choicekey{glossaries.sty}{numberedsection}%
152 [\gls@numberedsection@val\gls@numberedsection@nr]{%
153 false,nolabel,autolabel,nameref}[nolabel]{%
154 \ifcase\gls@numberedsection@nr\relax
155 \renewcommand*{\@glossarysecstar}{*}%
156 \renewcommand*{\@glossaryseclabel}{ }%
157 \or
158 \renewcommand*{\@glossarysecstar}{ }%
159 \renewcommand*{\@glossaryseclabel}{ }%
160 \or
161 \renewcommand*{\@glossarysecstar}{ }%
162 \renewcommand*{\@glossaryseclabel}{ }%
163 \label{\glsautoprefix@glo@type}}%
164 \or
165 \renewcommand*{\@glossarysecstar}{*}%
166 \renewcommand*{\@glossaryseclabel}{ }%
167 \protected@edef{@currentlabelname{\glossarytoctitle}}%
168 \label{\glsautoprefix@glo@type}}%
169 \fi
170 }

```

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
171 \@ifpackageloaded{classicthesis}%
172 {\newcommand*{\@glossary@default@style}{index}}%
173 {\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`.

```

174 \define@key{glossaries.sty}{style}{%
175 \def\@glossary@default@style{\#1}%
176 }

```

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
177 \newcommand*{\@gls@declareoption}[2]{%
178 \DeclareOptionX{\#1}{\#2}%
179 \DeclareOption{\#1}{\#2}%
180 }

```

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`

```
181 \newcommand*{\glossaryentrynumbers}[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 ignores its argument).

```
182 \@gls@declareoption{nonumberlist}{%
183   \renewcommand*{\glossaryentrynumbers}[1]{\gls@save@numberlist{#1}}%
184 }
```

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

```
185 \define@boolkey{glossaries.sty}[gls]{savenunderlist}[true]{}
186 \glssavenunderlistfalse
```

`eautonumberlist`

```
187 \newcommand*{\glo@seeautonumberlist}{}%
```

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

```
188 \@gls@declareoption{seeautonumberlist}{%
189   \renewcommand*{\glo@seeautonumberlist}{%
190     \def\glo@prefix{\glsnextpages}%
191   }%
192 }
```

`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.

```
193 \define@boolkey{glossaries.sty}[gls]{esclocations}[true]{}
194 \glssesclocationstrue
```

`\@gls@loadlong`

```
195 \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.

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

```

@gls@loadsuper The package isn't loaded if isn't installed.
197 \IfFileExists{supertabular.sty}{%
198   \newcommand*{\@gls@loadsuper}{\RequirePackage{glossary-super}}}{%
199   \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.
200 \@gls@declareoption{nosuper}{\renewcommand*{\@gls@loadsuper}{}}

@gls@loadlist
201 \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.
202 \@gls@declareoption{nolist}{%
203   \renewcommand*{\@gls@loadlist}{%
204     \ifdefstring{\@glossary@default@style}{list}{%
205       {\let\@glossary@default@style\relax}%
206     {}%
207   }%
208 }

@gls@loadtree
209 \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.
210 \@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).
211 \@gls@declareoption{nostyles}{%
212   \renewcommand*{\@gls@loadlong}{}%
213   \renewcommand*{\@gls@loadsuper}{}%
214   \renewcommand*{\@gls@loadlist}{}%
215   \renewcommand*{\@gls@loadtree}{}%
216   \let\@glossary@default@style\relax
217 }

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.)
218 \newcommand*{\glspostdescription}{%
219   \ifglsnopostdot\else.\spacefactor\sfcod\fi
220 }

```

nopostdot Boolean option to suppress post description dot

```
221 \define@boolkey{glossaries.sty}[gls]{nopostdot}[true]{}
222 \glsnopostrdotfalse
```

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

```
223 \define@boolkey{glossaries.sty}[gls]{nogroupskip}[true]{}
224 \glsnogroupskipfalse
```

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

```
225 \define@boolkey{glossaries.sty}[gls]{ucmark}[true]{}
226 \@ifclassloaded{memoir}{%
227 }%
228   \glsucmarktrue
229 }%
230 {%
231   \glsucmarkfalse
232 }
```

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

aryentrycounter

```
233 \newcommand*{\@gls@define@glossaryentrycounter}{%
234   \ifglsentrycounter
      Define the glossaryentry counter if it doesn't already exist.
    235   \ifundef\c@glossaryentry
    236     {%
    237       \ifx\@gls@counterwithin\@empty
    238         \newcounter{glossaryentry}%
    239       \else
    240         \newcounter{glossaryentry}[\@gls@counterwithin]%
    241       \fi
    242       \def\theHglossaryentry{\currentglossary.\theglossaryentry}%
    243     }%
    244     {}%
    245   \fi
  246 }
```

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.

```
247 \define@boolkey{glossaries.sty}[gls]{entrycounter}[true]{}
248 \glsentrycounterfalse
```

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

```

249 \define@key{glossaries.sty}{counterwithin}{%
250   \renewcommand*{\@gls@counterwithin}{\#1}%
251   \glsentrycountertrue
252   \@gls@define@glossaryentrycounter
253 }

```

s@counterwithin The default value is no parent counter:

```
254 \newcommand*{\@gls@counterwithin}{}%
```

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

subentrycounter

```
255 \newcommand{\@gls@define@glossarysubentrycounter}{%
```

Check if counter already defined.

```

256 \ifundef\c@glossarysubentry
257 {%
258   \ifglssubentrycounter
259     \ifglsentrycounter
260       \newcounter{glossarysubentry}[glossaryentry]%
261     \else
262       \newcounter{glossarysubentry}%
263     \fi

```

As with \theHglossaryentry, this starts with \currentglossary. to help avoid duplicate hyper targets.

```

264   \def\theHglossarysubentry{\currentglossary.\currentglssubentry.\theglossarysubentry}%
265   \fi
266 }%
267 {}%
268 }
```

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.

```
269 \define@boolkey{glossaries.sty}[gls]{subentrycounter}[true]{}
270 \glssubentrycounterfalse
```

default@sorttype Initialise default sort for \printnoidxglossary

```
271 \newcommand*{\@glo@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.

```

272 \define@choicekey{glossaries.sty}{sort}{standard,def,use,none}{%
273   \renewcommand*{\@glo@default@sorttype}{\#1}%
274   \csname @gls@setupsort@\#1\endcsname
275 }
```

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

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

```
276 \newcommand*{\glsprestandardsort}[3]{%
277   \glsdosanizesort
278 }
```

eck@sortallowed

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

upsort@standard Set up the macros for default sorting.

```
280 \newcommand*{\@gls@setupsort@standard}{%
  Store entry information when it's defined.
281   \def\do@glo@storeentry{\@glo@storeentry}%
  No count register required for standard sort.
282   \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.)
283   \def@\gls@defsort##1##2{%
284     \ifx\@glo@sort\@glsdefaultsort
285       \let\@glo@sort\@glo@name
286     \fi
287     \let\glsdosanizesort\gls@sanizesort
288     \glsprestandardsort{\@glo@sort}{##1}{##2}%
289     \expandafter\protected@xdef\csname glo##2@sort\endcsname{\@glo@sort}%
290   }%
```

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

```
291   \def@\gls@setsort##1{}%
  This sort option is allowed with \makeglossaries and \makenoidxglossaries.
292   \let@\glo@check@sortallowed@gobble
293 }
```

Set standard sort as the default:

```
294 \@gls@setupsort@standard
```

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

```
295 \newcommand*{\glssortnumberfmt}[1]{%
296   \ifnum#1<100000 0\fi
297   \ifnum#1<10000 0\fi
298   \ifnum#1<1000 0\fi
299   \ifnum#1<100 0\fi
```

```

300 \ifnum#1<10 0\fi
301 \number#1%
302 }

s@setupsort@def Set up the macros for order of definition sorting.
303 \newcommand*{\@gls@setupsort@def}{%
  Store entry information when it's defined.
304 \def\do@glo@storeentry{\@glo@storeentry}%
  Defined count register associated with the glossary.
305 \def\@gls@defsortcount##1{%
306   \expandafter\global
307   \expandafter\newcount\csname glossary@##1@sortcount\endcsname
308 }%

  Increment count register associated with the glossary and use as the sort key.
309 \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.
310 \ifcsundef{glossary@##1@sortcount}%
311 {\@gls@defsortcount{##1}}%
312 {}%
313 \expandafter\global\expandafter
314 \advance\csname glossary@##1@sortcount\endcsname by 1\relax
315 \expandafter\protected@xdef\csname glo@##2@sort\endcsname{%
316   \expandafter\glssortnumberfmt
317   {\csname glossary@##1@sortcount\endcsname}}%
318 }%

  Don't need to do anything when the entry is used.
319 \def\@gls@setsort##1{%
  This sort option is allowed with \makeglossaries and \makenoidxglossaries.
320 \let\@glo@check@sortallowed\@gobble
321 }

s@setupsort@use Set up the macros for order of use sorting.
322 \newcommand*{\@gls@setupsort@use}{%
  Don't store entry information when it's defined.
323 \let\do@glo@storeentry\@gobble
  Defined count register associated with the glossary.
324 \def\@gls@defsortcount##1{%
325   \expandafter\global
326   \expandafter\newcount\csname glossary@##1@sortcount\endcsname
327 }%

  Initialise the sort key to empty.
328 \def\@gls@defsort##1##2{%
329   \expandafter\gdef\csname glo@##2@sort\endcsname{}%
330 }%

```

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

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

Get the parent, if one exists

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

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

```
333 \ifx\@glo@parent\empty
```

```
334 \else
```

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

```
336 \fi
```

Set index information for this entry

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

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

```
339 \ifx\@gls@tmp\empty
```

```
340 \expandafter\global\expandafter
```

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

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

```
343 \expandafter\glossortnumberfmt
```

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

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

```
346 \fi
```

```
347 }%
```

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

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

```
349 }
```

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

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

Don't store entry index information.

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

No count register required for standard sort.

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

Don't modify sort value.

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

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

```
355 }%
```

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

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

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

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

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

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

```
360 }
```

\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.)

```
361 \newcommand*{\glsdefmain}{%
362   \if@gls@docloaded
363     \newglossary[glg2]{main}{gls2}{glo2}{\glossaryname}%
364   \else
365     \newglossary{main}{gls}{glo}{\glossaryname}%
366   \fi}
```

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

```
367 \newcommand*{\gls@tr@set@main@toctitle}{%
368   \translatelet{\glossarytoctitle}{Glossary}%
369 }%
370 }
```

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

```
371 \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

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

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

```
373 @gls@declareoption{nomain}{%
374   \let\glsdefaulttype\relax
375   \renewcommand*{\glsdefmain}{}%
376 }
```

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.

```
377 \define@boolkey{glossaries.sty}[gls]{acronym}[true]{%
378   \ifglsacronym
379     \renewcommand{\@gls@do@acronymsdef}{%
380       \DeclareAcronymList{acronym}%
381       \newglossary[alg]{acronym}{acr}{acn}{\acronymname}%
382     \renewcommand{\acronymtype}{acronym}%
383   }
```

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

```
383     \newcommand*{\gls@tr@set@acronym@toctitle}{%
384         \translatelet{\glossarytoctitle}{Acronyms}%
385     }%
386     }%
387 \else
388     \let\@gls@do@acronymsdef\relax
389 \fi
390 }
```

\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.

```
391 \AtBeginDocument{%
392     \ifglsacronym
393         \ifbool{glscompatible-3.07}{%
394             {}%
395             {}%
396             \providecommand*{\printacronyms}[1][]{%
397                 \printglossary[type=\acronymtype,#1]}%
398             }%
399     \fi
400 }
```

@do@acronymsdef Set default value

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

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

```
402 \@gls@declareoption{acronyms}{%
403     \glsacronymtrue
404     \def\@gls@do@acronymsdef{%
405         \DeclareAcronymList{acronym}%
406         \newglossary[alg]{acronym}{acr}{acn}{\acronymname}%
407         \renewcommand*{\acronymtype}{acronym}%
408     }%
409     }%
410 }
```

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

```
408     \newcommand*{\gls@tr@set@acronym@toctitle}{%
409         \translatelet{\glossarytoctitle}{Acronyms}%
410     }%
411 }
```

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

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

dtoacronymlists

```
414 \newcommand*{\@addtoacronymlists}[1]{%
```

```

415 \ifx\@glsacronymlists\@empty
416   \protected@xdef\@glsacronymlists{\#1}%
417 \else
418   \protected@xdef\@glsacronymlists{\@glsacronymlists,\#1}%
419 \fi
420 }

```

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

```

421 \newcommand*{\DeclareAcronymList}[1]{%
422   \glsIfListOfAcronyms{\#1}{}{\@addtoacronymlists{\#1}}%
423 }

```

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

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

```

424 \newcommand{\glsIfListOfAcronyms}[1]{%
425   \edef\@do@gls@islistofacronyms{%
426     \noexpand\@gls@islistofacronyms{\#1}{\@glsacronymlists}}%
427   \@do@gls@islistofacronyms
428 }

```

Internal command requires label and list to be expanded:

```

429 \newcommand{\@gls@islistofacronyms}[4]{%
430   \def\gls@islistofacronyms##1,#1,##2\end@gls@islistofacronyms{%
431     \def\@gls@before{##1}\def\@gls@after{##2}}%
432   \gls@islistofacronyms,#2,#1,\@nil\end@gls@islistofacronyms
433   \ifx\@gls@after\@nnil

```

Not found

```

434   #4%
435 \else

```

Found

```

436   #3%
437 \fi
438 }

```

`lsisacronymlist` Convenient boolean.

```
439 \newif\if@glsisacronymlist
```

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

```

440 \newcommand*{\gls@checkisacronymlist}[1]{%
441   \glsIfListOfAcronyms{\#1}%
442   {\@glsisacronymlisttrue}{\@glsisacronymlistfalse}%
443 }

```

`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.)

```
444 \newcommand*{\SetAcronymLists}[1]{%
445   \renewcommand*{\@glsacronymlists}{#1}%
446 }
```

`acronymlists`

```
447 \define@key{glossaries.sty}{acronymlists}{%
448   \DeclareAcronymList{#1}%
449 }
```

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`

```
450 \newcommand{\glscounter}{page}
```

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

```
451 \define@key{glossaries.sty}{counter}{%
452   \renewcommand*{\glscounter}{#1}%
453 }
```

`gls@nohyperlist`

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

`lareNoHyperList`

```
455 \newcommand*{\GlsDeclareNoHyperList}[1]{%
456   \ifdefempty{\gls@nohyperlist}%
457   {%
458     \renewcommand*{\gls@nohyperlist}{#1}%
459   }%
460   {%
461     \appto{\gls@nohyperlist}{, #1}%
462   }%
463 }
```

`nohypertypes`

```
464 \define@key{glossaries.sty}{nohypertypes}{%
465   \GlsDeclareNoHyperList{#1}%
466 }
```

`ossariesWarning` Prints a warning message.

```
467 \newcommand*{\GlossariesWarning}[1]{%
468   \PackageWarning{glossaries}{#1}%
469 }
```

```

esWarningNoLine Prints a warning message without the line number.
470 \newcommand*{\GlossariesWarningNoLine}[1]{%
471   \PackageWarningNoLine{glossaries}{#1}%
472 }

tentrieswarning Warn user that sorting may take a long time. This is actually an informational message rather
than a warning so just use \typeout.
473 \newcommand{\glosortentrieswarning}{%
474   \typeout{Using TeX to sort glossary entries---this may
475   take a while}%
476 }

nowarn Define package option to suppress warnings
477 \@gls@declareoption{nowarn}{%
478   \if@gls@debug
479     \GlossariesWarning{Warnings can't be suppressed in debug mode}%
480   \else
481     \renewcommand*{\GlossariesWarning}[1]{}%
482     \renewcommand*{\GlossariesWarningNoLine}[1]{}%
483     \renewcommand*{\glosortentrieswarning}{}%
484     \renewcommand*{\@gls@missinglang@warn}[2]{}%
485   \fi
486 }

issinglang@warn Missing language warning.
487 \newcommand*{\@gls@missinglang@warn}[2]{%
488   \PackageWarningNoLine{glossaries}{%
489     {No language module detected for '#1'.\MessageBreak
490     Language modules need to be installed separately.\MessageBreak
491     Please check on CTAN for a bundle called\MessageBreak
492     'glossaries-#2' or similar}%
493   }

nolangwarn Suppress warning if language support not found.
494 \@gls@declareoption{nolangwarn}{%
495   \renewcommand*{\@gls@missinglang@warn}[2]{}%
496 }

nonglossdefined Issue a warning if overriding \printglossary
497 \newcommand*{\@gls@warnnonglossdefined}{%
498   \GlossariesWarning{Overriding \string\printglossary}%
499 }

theglossdefined Issue a warning if overriding theglossary
500 \newcommand*{\@gls@warnontheglossdefined}{%
501   \GlossariesWarning{Overriding 'theglossary' environment}%
502 }

```

```

noredefwarn Suppress warning on redefinition of \printglossary
503 \@gls@declareoption{noredefwarn}{%
504   \renewcommand*\{@gls@warnonglossdefined}{}%
505   \renewcommand*\{@gls@warnontheglossdefined}{}%
506 }

```

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
507 \newcommand*\{@gls@sanitizedesc}{%
508 }

```

**\glssetexpandfield{<field>}**

Sets field to always expand.

```

509 \newcommand*\glssetexpandfield[1]{%
510   \csdef{gls@assign@#1@field}##1##2{%
511     \@@gls@expand@field{##1}{#1}{##2}%
512   }%
513 }

```

**\glssetnoexpandfield{<field>}**

Sets field to never expand.

```

514 \newcommand*\glssetnoexpandfield[1]{%
515   \csdef{gls@assign@#1@field}##1##2{%
516     \@@gls@noexpand@field{##1}{#1}{##2}%
517   }%
518 }

```

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

```
519 \glssetexpandfield{type}
```

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

```
520 \glssetnoexpandfield{desc}
```

escplural@field

```
521 \glssetnoexpandfield{descplural}
```

ls@sanitizename

```
522 \newcommand*\{@gls@sanitizename}{}%
```

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

```
523 \glssetnoexpandfield{name}
```

```

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

gn@symbol@field  Don't expand symbol by default.
525 \glssetnoexpandfield{symbol}

bolplural@field
526 \glssetnoexpandfield{symbolplural}

    Sanitizing stuff:

ls@sanitizesort
527 \newcommand*{\@gls@sanitizesort}{%
528   \ifglssanitizesort
529     \@@gls@sanitizesort
530   \else
531     \@@gls@nosanitizesort
532   \fi
533 }

ls@sanitizesort
534 \newcommand*{\@@gls@sanitizesort}{%
535   \onelevel@sanitize\glo@sort
536 }

@nosanitizesort
537 \newcommand*{\@@gls@nosanitizesort}{}{}

dx@sanitizesort  Remove braces around first character (if present) before sanitizing.
538 \newcommand*{\@gls@noidx@sanitizesort}{%
539   \ifdefvoid\glo@sort
540   {}%
541   {%
542     \expandafter\@gls@noidx@sanitizesort\glo@sort\gls@end@sanitizesort
543   }%
544 }
545 \def\@@gls@noidx@sanitizesort#1#2\gls@end@sanitizesort{%
546   \def\glo@sort{#1#2}%
547   \onelevel@sanitize\glo@sort
548 }

@nosanitizesort
549 \newcommand*{\@@gls@noidx@nosanitizesort}{%
550   \ifdefvoid\glo@sort
551   {}%
552   {%
553     \expandafter\@gls@noidx@no@sanitizesort\glo@sort\gls@end@sanitizesort
554   }%
}

```

```

555 }
556 \def\@gls@noidx@no@sanitizesort#1#2\gls@end@sanitizesort{%
557   \bgroup
558   \glsnoidxstripaccents
559   \protected@xdef\@glo@sort{#1#2}%
560   \egroup
561   \let\@glo@sort\@glo@sort
562 }

```

`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`. It's much better to use `xindy` or `bib2gls` with the correct language setting.

```

563 \newcommand*\glsnoidxstripaccents{%
564   \let\IeC\@firstofone
565   \let\add@accent\@secondoftwo
566   \let\text@composite\x\@secondoftwo
567   \let\tabackludge\@secondoftwo
568   \expandafter\def\csname \encodingdefault-cmd\endcsname##1##2##3{##3}%
569   \expandafter\def\csname OT1-cmd\endcsname##1##2##3{##3}%
570   \expandafter\def\csname T1-cmd\endcsname##1##2##3{##3}%
571   \expandafter\def\csname PD1-cmd\endcsname##1##2##3{##3}%
572   \let'\@firstofone
573   \let`@firstofone
574   \let^@firstofone
575   \let"@\firstofone
576   \let\u@\firstofone
577   \let\t@\firstofone
578   \let\d@\firstofone
579   \let\r@\firstofone
580   \let=@\firstofone
581   \let.\@\firstofone
582   \let.^@\firstofone
583   \let\v@\firstofone
584   \let\H@\firstofone
585   \let\c@\firstofone
586   \let\b@\firstofone

587   \let\aa\@secondoftwo
588   \def\AE{AE}%
589   \def\ae{ae}%
590   \def\OE{OE}%
591   \def\oe{oe}%
592   \def\AA{AA}%
593   \def\aa{aa}%
594   \def\L{L}%
595   \def\l{l}%
596   \def\O{O}%
597   \def\o{o}%

```

```

598 \def\SS{SS}%
599 \def\ss{ss}%
600 \def\th{th}%

601 \def\TH{TH}%
602 \def\dh{dh}%
603 \def\DH{DH}%
604 }

```

Need to check if the LaTeX kernel is at least version 2019/10/01 as that changes the way that UTF-8 characters expand.

```

605 \@ifl@t@r\fmtversion{2019/10/01}%
606 {%
607   \appto\glsnoidxstripaccents{\let\UTFviii@two@octets\UTFviii@two@octets@combine}%
608 }%
609 {}

```

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.

```

610 \define@boolkey[gls]{sanitize}{description}[true]{%
611   \GlossariesWarning{sanitize={description} package option deprecated}%
612   \ifgls@sanitize@description
613     \glssetnoexpandfield{desc}%
614     \glssetnoexpandfield{descplural}%
615   \else
616     \glssetexpandfield{desc}%
617     \glssetexpandfield{descplural}%
618   \fi
619 }

620 \define@boolkey[gls]{sanitize}{name}[true]{%
621   \GlossariesWarning{sanitize={name} package option deprecated}%
622   \ifgls@sanitize@name
623     \glssetnoexpandfield{name}%
624   \else
625     \glssetexpandfield{name}%
626   \fi
627 }

628 \define@boolkey[gls]{sanitize}{symbol}[true]{%
629   \GlossariesWarning{sanitize={symbol} package option deprecated}%
630   \ifgls@sanitize@symbol
631     \glssetnoexpandfield{symbol}%
632     \glssetnoexpandfield{symbolplural}%
633   \else
634     \glssetexpandfield{symbol}%
635     \glssetexpandfield{symbolplural}%
636   \fi
637 }

```

```

sanitizesort
638 \define@boolkey{glossaries.sty}[gls]{sanitizesort}[true]{%
639   \ifglssanitizesort
640     \glssetnoexpandfield{sortvalue}%
641     \renewcommand*{\@gls@noidx@setsanitizesort}{%
642       \glssanitizesorttrue
643       \glssetnoexpandfield{sortvalue}%
644     }%
645   \else
646     \glssetexpandfield{sortvalue}%
647     \renewcommand*{\@gls@noidx@setsanitizesort}{%
648       \glssanitizesortfalse
649       \glssetexpandfield{sortvalue}%
650     }%
651   \fi
652 }

      Default setting:

653 \glssanitizesorttrue
654 \glssetnoexpandfield{sortvalue}%

setsanitizesort Default behaviour for \makenoidxglossaries is sanitizesort=false.
655 \newcommand*{\@gls@noidx@setsanitizesort}{%
656   \glssanitizesortfalse
657   \glssetexpandfield{sortvalue}%
658 }

659 \define@choicekey[gls]{sanitize}{sort}{true, false}[true]{%
660   \setbool{glssanitizesort}{#1}%
661   \ifglssanitizesort
662     \glssetnoexpandfield{sortvalue}%
663   \else
664     \glssetexpandfield{sortvalue}%
665   \fi
666   \GlossariesWarning{sanitize={sort} package option
667   deprecated. Use sanitizesort instead}%
668 }

```

sanitize

```

669 \define@key{glossaries.sty}{sanitize}[description=true, symbol=true, name=true]{%
670   \ifthenelse{\equal{#1}{none}}{%
671     {%
672       \GlossariesWarning{sanitize package option deprecated}%
673       \glssetexpandfield{name}%
674       \glssetexpandfield{symbol}%
675       \glssetexpandfield{symbolplural}%
676       \glssetexpandfield{desc}%
677       \glssetexpandfield{descplural}%
678     }%

```

```

679  {%
680    \setkeys[gls]{sanitize}{#1}%
681  }%
682 }

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

\otranslatorhook \gls@notranslatorhook has been removed.

\s@usetranslator
684 \newcommand*\gls@usetranslator{%
  polyglossia tricks \ifpackageloaded into thinking that babel has been loaded, so check for
  polyglossia as well.
685   \ifpackageloaded{polyglossia}%
686   {%
687     \let\glsifusetranslator\secondoftwo
688   }%
689   {%
690     \ifpackageloaded{babel}%
691     {%
692       \IfFileExists{translator.sty}%
693       {%
694         \RequirePackage{translator}%
695         \let\glsifusetranslator\firstoftwo
696       }%
697       {}%
698     }%
699     {}%
700   }%
701 }

\dtranslatordict Checks if given translator dictionary has been loaded.
702 \newcommand{\glsifusedtranslatordict}[3]{%
703   \glsifusetranslator
704   {\ifcsdef{ver@glossaries-dictionary-\#1.dict}{\#2}{\#3}}%
705   {\#3}%
706 }

\notranslate Provide a synonym for translate=false that can be passed via the document class.
707 \glsdeclareoption{notranslate}{%
708   \glstranslatefalse
709   \let\gls@usetranslator\relax
710   \let\glsifusetranslator\secondoftwo
711 }

```

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

```
712 \define@choicekey{glossaries.sty}{translate}%
713   [\"gls@translate@val\"gls@translate@nr]%
714   {true,false,babel}[true]%
715 {%
716   \ifcase\gls@translate@nr\relax
717     \glstranslatetrue
718     \renewcommand*@\gls@usetranslator{%
719       \@ifpackageloaded{polyglossia}%
720       {%
721         \let\glsifusetranslator\@secondoftwo
722       }%
723       {%
724         \@ifpackageloaded{babel}%
725         {%
726           \IfFileExists{translator.sty}%
727             {%
728               \RequirePackage{translator}%
729               \let\glsifusetranslator\@firstoftwo
730             }%
731             {}%
732           }%
733           {}%
734         }%
735       }%
736     \or
737       \glstranslatefalse
738       \let@\gls@usetranslator\relax
739       \let\glsifusetranslator\@secondoftwo
740     \or
741       \glstranslatetrue
742       \let@\gls@usetranslator\relax
743       \let\glsifusetranslator\@secondoftwo
744     \fi
745 }
```

Set the default value:

```
746 \glstranslatefalse
747 \let\glsifusetranslator\@secondoftwo
748 \@ifpackageloaded{translator}%
749 {%
750   \glstranslatetrue
751   \let\glsifusetranslator\@firstoftwo
752 }%
753 {%
754   \@for\gls@thissty:=tracklang,babel,ngerman,polyglossia\do
755   {
756     \@ifpackageloaded{\gls@thissty}%
757     {%
```

```

758     \glstranslatetrue
759     \endfortrue
760   }%
761   {}%
762 }
763 }
```

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

```

764 \define@boolkey{glossaries.sty}[gls]{indexonlyfirst}[true]{}
765 \glsindexonlyfirstfalse
```

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

```

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

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

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

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

```

769 \define@boolkey{glossaries.sty}[glsacr]{footnote}[true]{%
770   \ifbool{glsacrdescription}{%
771     {}%
772     {}%
773     \renewcommand*{\@gls@sanitizedesc}{}{%
774   }%
775   \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}{%
776 }}
```

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

```

777 \define@boolkey{glossaries.sty}[glsacr]{description}[true]{%
778   \renewcommand*{\@gls@sanitizesymbol}{}{%
779   \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}{%
780 }}
```

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

```

781 \define@boolkey{glossaries.sty}[glsacr]{smallcaps}[true]{%
782   \renewcommand*{\@gls@sanitizesymbol}{}{%
783   \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}{%
784 }}
```

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

```

785 \define@boolkey{glossaries.sty}[glsacr]{smaller}[true]{%
786   \renewcommand*{\@gls@sanitizesymbol}{}{%
787   \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}{%
788 }}
```

```

dua Define \newacronym to always use the long forms (i.e. don't use acronyms)
789 \define@boolkey{glossaries.sty}[glsacr]{dua}[true]{%
790   \renewcommand*{\@gls@sanitizesymbol}{}%
791   \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
792 }

shotcuts Define acronym shortcuts.
793 \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.
794 \newcommand*{\glsorder}{word}

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

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

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

The default is makeindex:
799 \glsxindyfalse

makeindex Define package option to specify that makeindex will be used to sort the glossaries:
800 \gls@declareoption{makeindex}{\glsxindyfalse}

The xindy package option may have a value which in turn can be a key=value list. First de-
fine the keys for this sub-list. The boolean glsnumbers determines whether to automatically
add the glsnumbers letter group.
801 \define@boolkey[gls]{xindy}{glsnumbers}[true]{}
802 \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.)
803 \def\xdy@main@language{\languagename}%

Define key to set the language
804 \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.
805 \ifcsundef{\inputencodingname}{%
806   \def\gls@codepage{}{%
807   \def\gls@codepage{\inputencodingname}%
808 }%
Define a key to set the code page.
809 \define@key[gls]{xindy}{codepage}{\def\gls@codepage{\#1}}}

xindy Define package option to specify that xindy will be used to sort the glossaries:
810 \define@key{glossaries.sty}{xindy}[]{%
811   \glsxindytrue
812   \setkeys[gls]{xindy}{\#1}%
813 }

xindygloss Provide a synonym for xindy that can be passed via the document class options.
814 \@gls@declareoption{xindygloss}{%
815   \glsxindytrue
816 }

ndynoglsnumbers Provide a synonym for xindy=glsnumbers=false that can be passed via the document class
options.
817 \@gls@declareoption{xindynoglsnumbers}{%
818   \glsxindytrue
819   \gls@xindy@glsnumbersfalse
820 }

omakeglossaries
821 \providecommand{\@domakeglossaries}[1]{#1}

isablemakegloss Provide a way of disabling \makeglossaries. For example, if a class or package explicitly
uses \makeglossaries. This is a valueless option to allow it to be passed through the docu-
ment class option list.
822 \@gls@declareoption{isablemakegloss}{%
823   \ifdefequal{\makeglossaries}{\noexpand\makeglossaries}%
824   {%
825     \GlossariesWarning{Option ‘isablemakegloss’ has no effect}%
826     (\string\makenoidxglossaries\space has already been used)}%
827   }%
828   {%
829     \ifdefequal{\makeglossary}{\gobble}%
830     {%
831       \GlossariesWarning{Option ‘isablemakegloss’ has no effect}%
832       (\string\makeglossaries\space has already been used)}%
833     }%
834   {%
835     \renewcommand{\@domakeglossaries}[1]{%

```

```

836      \PackageInfo{glossaries}{\string\makeglossaries\space and
837      \string\makenoidxglossaries\space have been disabled}%
838  }%
839 }%
840 }%
841 }

estoremakegloss Cancel the effect of disablemakegloss.
842 \@gls@declareoption{restoremakegloss}{%
843   \ifdefequal\makeglossaries\@no@makeglossaries
844   {%
845     \GlossariesWarning{Option ‘restoremakegloss’ has no effect
846     (\string\makenoidxglossaries\space has already been used)}%
847   }%
848   {%
849     \ifdefequal\@makeglossary\@gobble
850     {%
851       \GlossariesWarning{Option ‘restoremakegloss’ has no effect
852       (\string\makeglossaries\space has already been used)}%
853     }%
854     {%
855       \PackageInfo{glossaries}{\string\makeglossaries\space and
856       \string\makenoidxglossaries\space have been restored}%
857       \let\@domakeglossaries\@firstofone
858     }%
859   }%
860 }

write@glslabels
861 \newcommand*{\@do@write@glslabels}{%
862   \AtEndDocument{\@@do@write@glslabels}%
863   \let\@do@write@glslabels\relax
864 }

write@glslabels
865 \newcommand*{\@@do@write@glslabels}{%
866   \newwrite\@gls@labelsfile
867   \immediate\openout\@gls@labelsfile=\jobname.glslabels
868   \forallglsentries[\@glo@types,\@ignored@glossaries]{\@glsentry}{%
869     \ifdefempty{\@glsentry}{}{\immediate\write\@gls@labelsfile{\@glsentry}}}}%
870   \immediate\closeout\@gls@labelsfile
871 }

writeglslabels This option will write all entry labels (including those in ignored glossaries) to the file
\jobname.glslabels. This file may be used by text editors for label auto-completion.
872 \@gls@declareoption{writeglslabels}{\@do@write@glslabels}

\ifglsautomake
873 \newif\ifglsautomake

```

```

gls@automake@nr
874 \newcommand{\gls@automake@nr}{1}

automake If this setting is on, automatically run makeindex/xindy at the end of the document. Must be used with \makeglossaries. Default is false. As from v4.42, this is now a choice rather than boolean key.
875 \define@choicekey{glossaries.sty}{automake}%
876   [\gls@automake@val\gls@automake@nr]{true,false,immediate}[true]{%
877   \ifnum\gls@automake@nr=1\relax
878     \glsautomakefalse
879   \else
880     \glsautomaketrue
881   \fi
882   \ifglsautomake
883     \renewcommand*{\@gls@doautomake}{%
884       \PackageError{glossaries}{You must use
885         \string\makeglossaries\space with automake=true}
886     }%
887       Either remove the automake=true setting or
888       add \string\makeglossaries\space to your document preamble.%
889     }%
890   }%
891   \else
892     \renewcommand*{\@gls@doautomake}{%
893   \fi
894 }
895 \glsautomakefalse

@gls@doautomake
896 \newcommand*{\@gls@doautomake}{}%
897 \AtEndDocument{\@gls@doautomake}

savewrites The savewrites package option is provided to save on the number of write registers.
898 \define@boolkey{glossaries.sty}[gls]{savewrites}[true]{%
899   \ifglssavewrites
900     \renewcommand*{\glswritefiles}{\@glswritefiles}%
901   \else
902     \let\glswritefiles\empty
903   \fi
904 }

Set default:
905 \glssavewritesfalse
906 \let\glswritefiles\empty

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

```

```

compatible-2.07
909 \define@boolkey{glossaries.sty}[gls]{compatible-2.07}[true]{%
  Also set 3.07 compatibility if this option is set.
910  \ifbool{glscompatible-2.07}{%
911    {%
912      \booltrue{glscompatible-3.07}%
913    }%
914  {}%
915 }
916 \boolfalse{glscompatible-2.07}

al@makeglossary Store the original definition.
917 \let\gls@original@makeglossary\makeglossary

original@glossary Store the original definition.
918 \let\gls@original@glossary\glossary

\makeglossary The \makeglossary command is redefined to be identical to \makeglossaries. (This is
done partly to reinforce the message that you must either use \makeglossary for all the
glossaries or for none of them, but is also a legacy from the old glossary package.)
919 \def\makeglossary{%
920  \GlossariesWarning{Use of \string\makeglossary\space with
921  glossaries.sty is \MessageBreak deprecated. Use \string\makeglossaries\space
922  instead. If you \MessageBreak need the original definition of
923  \string\makeglossary\space use \MessageBreak the package options
924  kernelglossredefs=false (to \MessageBreak restore the former definition of
925  \string\makeglossary) and \MessageBreak nomain (if the file extensions cause a
926  conflict)}%
927  \makeglossaries
928 }

erride@glossary
929 \newcommand*{\@gls@override@glossary}[1] [main]{%
930  \GlossariesWarning{Use of \string\glossary\space with
931  glossaries.sty is deprecated. \MessageBreak Indexing should be performed
932  with the user level \MessageBreak commands, such as \string\gls\space or
933  \string\glsadd. If you need the \MessageBreak original definition of
934  \string\glossary\space use the package \MessageBreak options
935  kernelglossredefs=false (to restore the \MessageBreak former definition of
936  \string\glossary) and nomain (if the \MessageBreak file extensions cause a
937  conflict)}%
938  \gls@glossary{#1}%
939 }

```

In v4.10, the redefinition of \glossary was removed since it was never intended as a user level command (and wasn't documented in the user manual), however it seems there are packages that have hacked the internal macros used by glossaries and no longer work with

this redefinition removed, so it's been restored in v4.11 but is not used at all by glossaries. (This may be removed or moved to a compatibility mode in future.) As from v4.41, the use of \glossary now triggers a warning. The package option kernelglossredefs=nowarn may be used to remove the warning, but it's better not to use \glossary.

```
\glossary
940 \if@gls@docloaded
941 \else
942   \def\glossary{\@gls@override@glossary}
943 \fi
```

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

```
944 \define@choicekey{glossaries.sty}{kernelglossredefs}%
945 [\gls@debug@val\gls@debug@nr]{true,false,nowarn}[true]%
946 {%
947   \ifcase\gls@debug@nr\relax
948     \def\glossary{\@gls@override@glossary}%
949     \def\makeglossary{%
950       \GlossariesWarning{Use of \string\makeglossary\space with
951       glossaries.sty is deprecated. Use \string\makeglossaries\space
952       instead. If you need the original definition of
953       \string\makeglossary\space use the package options
954       kernelglossredefs=false (to prevent redefinition of
955       \string\makeglossary) and nomain (if the file extensions cause a
956       conflict)}%
957     \makeglossaries
958   }%
959 \or
960   \let\glossary\gls@original@glossary
961   \let\makeglossary\gls@original@makeglossary
962 \or
963   \def\makeglossary{\makeglossaries}%
964   \renewcommand*{\@gls@override@glossary}[1][main]{%
965     \gls@glossary{##1}%
966   }%
967 \fi
968 }
```

**symbols** Create a “symbols” glossary type

```
969 \@gls@declareoption{symbols}{%
970   \let@\gls@do@symbolsdef@\gls@symbolsdef
971 }
```

```

Default is not to define the symbols glossary:
972 \newcommand*{\@gls@do@symbolsdef}{}}

@gls@symbolsdef
973 \newcommand*{\@gls@symbolsdef}%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
974   \newglossary[slg]{symbols}{sls}{slo}{\glssymbolsgroupname}%
975   \newcommand*{\printsymbols}[1][]{\printglossary[type=symbols,##1]}%


Define hook to set the toc title when translator is in use.
976 \newcommand*{\gls@tr@set@symbols@toctitle}%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
977   \translatelet{\glossarytoctitle}{Symbols (glossaries)}%
978 }%
979 }%


numbers Create a “symbols” glossary type
980 \@gls@declareoption{numbers}%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
981   \let\@gls@do@numbersdef\@gls@numbersdef
982 }

Default is not to define the numbers glossary:
983 \newcommand*{\@gls@do@numbersdef}{}}

@gls@numbersdef
984 \newcommand*{\@gls@numbersdef}%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
985   \newglossary[nlg]{numbers}{nls}{nlo}{\glsnumbersgroupname}%
986   \newcommand*{\printnumbers}[1][]{\printglossary[type=numbers,##1]}%


Define hook to set the toc title when translator is in use.
987 \newcommand*{\gls@tr@set@numbers@toctitle}%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
988   \translatelet{\glossarytoctitle}{Numbers (glossaries)}%
989 }%
990 }%


index Create an “index” glossary type
991 \@gls@declareoption{index}%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
992   \ifx\@gls@do@indexdef\@empty
993     \let\@gls@do@indexdef\@gls@indexdef
994   \fi
995 }%


noglossaryindex Counteract index if it happens to be globally used in the document class.
996 \@gls@declareoption{noglossaryindex}%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
997   \let\@gls@do@indexdef\relax
998 }%


Default is not to define index glossary:
999 \newcommand*{\@gls@do@indexdef}{}}

```

```

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

1000 \newcommand*\@gls@indexdef}{%
1001   \newglossary[ilg]{index}{ind}{idx}{\indexname}%
1002   \newcommand*\printindex}[1][]{\printglossary[type=index,##1]}%
1003   \newcommand*\newterm}[2][]{%
1004     \newglossaryentry{##2}{%
1005       {type={index},name={##2},description={\nopostdesc},##1}}%
1006   \let\@gls@do@indexdef\relax
1007 }%

```

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

```

1008 \@for\CurrentOption :=\@declaredoptions\do{%
1009   \ifx\CurrentOption\empty
1010   \else
1011     \expandafter\@expandtwoargs
1012     \in@{\CurrentOption ,}{},\@classoptionslist,\@curroptions,}%
1013   \ifin@{%
1014     \use@option
1015     \expandafter\let\csname ds@\CurrentOption\endcsname\empty
1016   \fi
1017 \fi
1018 }

```

Now process options passed to the package:

```

1019 \ProcessOptionsX
Load backward compatibility stuff:
1020 \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:

```

1021 \disable@keys{glossaries.sty}{compatible-2.07,%
1022 xindy,xindygloss,xindynoglsnumbers,makeindex,%
1023 acronym,translate,notranslate,nolong,nosuper,notree,nostyles,%
1024 nomain,noglossaryindex}

```

Now define `\setupglossaries`:

```

1025 \newcommand*\@setupglossaries}[1]{%
1026   \renewcommand*\@gls@setacrstyle}{()}%
1027   \ifglsacrshortcuts
1028     \def\@gls@setupshortcuts{\glsacrshortcutstrue}%
1029   \else
1030     \def\@gls@setupshortcuts{%
1031       \ifglsacrshortcuts
1032         \DefineAcronymSynonyms
1033       \fi
1034     }%
1035   \fi
1036   \glsacrshortcutsfalse

```

```

1037 \let\@gls@do@numbersdef\relax
1038 \let\@gls@do@symbolssdef\relax
1039 \let\@gls@do@indexdef\relax
1040 \let\@gls@do@acronymsdef\relax
1041 \ifglsentrycounter
1042   \let\@gls@doentrycounterdef\relax
1043 \else
1044   \let\@gls@doentrycounterdef\@gls@define@glossaryentrycounter
1045 \fi
1046 \ifglssubentrycounter
1047   \let\@gls@dosubentrycounterdef\relax
1048 \else
1049   \let\@gls@dosubentrycounterdef\@gls@define@glossarysubentrycounter
1050 \fi
1051 \setkeys{glossaries.sty}{#1}%
1052 \@gls@setacrstyle
1053 \@gls@setupshortcuts
1054 \@gls@do@acronymsdef
1055 \@gls@do@numbersdef
1056 \@gls@do@symbolssdef
1057 \@gls@do@indexdef
1058 \@gls@doentrycounterdef
1059 \@gls@dosubentrycounterdef
1060 }

```

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}`).

```

1061 \ifthenelse{\equal{\glscounter}{section}}%
1062 {%
1063   \ifcsundef{chapter}{}%
1064   {%
1065     \let\@gls@old@chapter\@chapter
1066     \def\@chapter[#1]#2{\@gls@old@chapter[{-#1}]{}{#2}%
1067     \ifcsundef{hyperdef}{}{\hyperdef{section}{\thesection}{}{}}%
1068   }%
1069 }%
1070 {}

```

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

```
1071 \newcommand*{\@gls@onlypremakeg}{}%
```

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

```

\makeglossaries.

1072 \newcommand*{\@onlypremakeg}[1]{%
1073   \ifx\@gls@onlypremakeg\empty
1074     \def\@gls@onlypremakeg{\#1}%
1075   \else
1076     \expandafter\toks@\expandafter{\@gls@onlypremakeg}%
1077     \edef\@gls@onlypremakeg{\the\toks@\noexpand\#1}%
1078   \fi
1079 }

```

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

```

1080 \newcommand*{\@disable@onlypremakeg}{%
1081   \for@thiscs:=\@gls@onlypremakeg\do{%
1082     \expandafter\@disable@premakecs@\thiscs%
1083   }%

```

`\sable@premakecs` Disables the given command.

```

1084 \newcommand*{\@disable@premakecs}[1]{%
1085   \def#1{\PackageError{glossaries}{\string#1\space may only be
1086   used before \string\makeglossaries}{You can't use
1087   \string#1\space after \string\makeglossaries}%
1088 }

```

## 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
1089 \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
1090 \providecommand*{\acronymname}{Acronyms}

```

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

```

1091 \newcommand*{\glsettoctitle}[1]{%
1092   \def\glossarytoctitle{\csname @glotype\#1@title\endcsname}%

```

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

```

\entryname
1093 \providecommand*{\entryname}{Notation}

```

```

descriptionname
 1094 \providecommand*\{descriptionname}{Description}

\symbolname
 1095 \providecommand*\{\symbolname}{Symbol}

\pagelistname
 1096 \providecommand*\{\pagelistname}{Page List}

  Labels for makeindex's symbol and number groups:

symbolsgroupname
 1097 \providecommand*\{\glssymbolsgroupname}{Symbols}

numbersgroupname
 1098 \providecommand*\{\glsnumbersgroupname}{Numbers}

glspluralsuffix The default plural is formed by appending \glspluralsuffix to the singular form.
 1099 \newcommand*\{\glspluralsuffix}{s}

acrpluralsuffix Default plural suffix for acronyms
 1100 \newcommand*\{\glsacrpluralsuffix}{\glspluralsuffix}

acrpluralsuffix
 1101 \newcommand*\{\glsupacrpluralsuffix}{\glstextup{\glsacrpluralsuffix}{}}

\seename
 1102 \providecommand*\{\seename}{see}

\andname
 1103 \providecommand*\{\andname}{\&}

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

eGlossariesLang
 1104 \newcommand*\{\RequireGlossariesLang}[1]{%
 1105   \@ifundefined{ver@glossaries-\#1.ldf}{\input{glossaries-\#1.ldf}}{}%
 1106 }

sGlossariesLang
 1107 \newcommand*\{\ProvidesGlossariesLang}[1]{%
 1108   \ProvidesFile{glossaries-\#1.ldf}%
 1109 }

ssarytocaptions Does nothing if translator hasn't been loaded.
 1110 \newcommand*\{\addglossarytocaptions}[1]{}

```

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

```
1111 \ifglstranslate
      Load tracklang
1112   \RequirePackage{tracklang}
      Load translator if required.
1113   \@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.)

```
1114  \@ifpackageloaded{translator}
1115  {%
```

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 `\bblo@loaded` isn't simply `english`, then don't use the translator dictionaries.

```
1116  \ifboolexpr
1117  {
1118    test {\ifdefstring{\trans@languages}{English}}
1119    and not
1120    test {\ifdefstring{\bblo@loaded}{english}}
1121  }
1122  {%
1123    \let\glsifusetranslator\@secondoftwo
1124  }%
1125  {%
1126    \usedictionary{glossaries-dictionary}%
1127    \renewcommand*{\addglossarytocaptions}[1]{%
1128      \ifcsundef{captions#1}{%
1129        {%
1130          \expandafter\let\expandafter\@gls@tmp\csname captions#1\endcsname
1131          \expandafter\toks@\expandafter{\@gls@tmp
1132            \renewcommand*{\glossaryname}{\translate{Glossary}}%
1133          }%
1134          \expandafter\edef\csname captions#1\endcsname{\the\toks@}%
1135        }%
1136      }%
1137    }%
1138  }%
1139  {}%
```

Check for tracked languages

```
1140  \AnyTrackedLanguages
1141  {%
1142    \ForEachTrackedDialect{\this@dialect}{%
1143      \IfTrackedLanguageFileExists{\this@dialect}{%
```

```

1144     {glossaries-}%
1145     {.ldf}%
1146     {%
1147         \RequireGlossariesLang{\CurrentTrackedTag}%
1148     }%
1149     {%
1150         \gls@missinglang@warn\this@dialect\CurrentTrackedLanguage
1151     }%
1152     }%
1153 }%
1154 {}%

```

if using translator use translator interface.

```

1155 \glsifusettranslator
1156 {%
1157     \renewcommand*{\glssettoctitle}[1]{%
1158         \ifcsdef{gls@tr@set@#1@toctitle}%
1159             {%
1160                 \csuse{gls@tr@set@#1@toctitle}%
1161             }%
1162             {%
1163                 \def\glossarytoctitle{\csname @glotype@#1@title\endcsname}%
1164             }%
1165         }%
1166         \renewcommand*{\glossaryname}{\translate{Glossary}}%
1167         \renewcommand*{\acronymname}{\translate{Acronyms}}%
1168         \renewcommand*{\entryname}{\translate{Notation (glossaries)}}%
1169         \renewcommand*{\descriptionname}{%
1170             \translate{Description (glossaries)}}%
1171         \renewcommand*{\symbolname}{\translate{Symbol (glossaries)}}%
1172         \renewcommand*{\pagelistname}{%
1173             \translate{Page List (glossaries)}}%
1174         \renewcommand*{\glossymbolsgroupname}{%
1175             \translate{Symbols (glossaries)}}%
1176         \renewcommand*{\glossnumbersgroupname}{%
1177             \translate{Numbers (glossaries)}}%
1178     }{%
1179 \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.

```

1180 \DeclareRobustCommand*{\nopostdesc}{}

```

\@nopostdesc Suppress next description terminator.

```

1181 \newcommand*{\@nopostdesc}{%
1182     \let\org@glspostdescription\glspostdescription
1183     \def\glspostdescription{%
1184         \let\glspostdescription\org@glspostdescription}%
1185 }

```

```

\f@no@post@desc Used for comparison purposes.
1186 \newcommand*{\f@no@post@desc}{\nopostdesc}

\glspar Provide means of having a paragraph break in glossary entries
1187 \newcommand{\glspar}{\par}

\setStyleFile Sets the style file. The relevant extension is appended.
1188 \newcommand{\setStyleFile}[1]{%
1189   \renewcommand{\gls@istfilebase}{#1}%
}
Just in case \istfilename has been modified.
1190 \ifglsxindy
1191   \def\istfilename{\gls@istfilebase.xdy}
1192 \else
1193   \def\istfilename{\gls@istfilebase.ist}
1194 \fi
1195 }

This command only has an effect prior to using \makeglossaries.
1196 \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
1197 \ifglsxindy
1198   \def\istfilename{\gls@istfilebase.xdy}
1199 \else
1200   \def\istfilename{\gls@istfilebase.ist}
1201 \fi

\gls@istfilebase
1202 \newcommand*{\gls@istfilebase}{\jobname}

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

\@istfilename
1203 \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
1204 \newcommand*{\glscompositor}{.}

```

`\lsSetCompositor` Sets the compositor.

```
1205 \newcommand*{\glsSetCompositor}[1]{%
1206   \renewcommand*{\glscompositor}{#1}}
```

Only use before `\makeglossaries`

```
1207 \@onlypremakeg\glsSetCompositor
```

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

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

```
1208 \newcommand*{\@glsAlphaCompositor}{\glscompositor}
```

`\AlphaCompositor` Sets the alpha compositor.

```
1209 \ifglsxindy
1210   \newcommand*{\glsSetAlphaCompositor}[1]{%
1211     \renewcommand*{\@glsAlphaCompositor}{#1}}
1212 \else
1213   \newcommand*{\glsSetAlphaCompositor}[1]{%
1214     \glsnoxindywarning\glsSetAlphaCompositor}
1215 \fi
```

Can only be used before `\makeglossaries`

```
1216 \@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.

```
1217 \newcommand*{\gls@suffixF}{}%
```

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

```
1218 \newcommand*{\glsSetSuffixF}[1]{%
1219   \renewcommand*{\gls@suffixF}{#1}}
```

Only has an effect when used before `\makeglossaries`

```
1220 \@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.

```
1221 \newcommand*{\gls@suffixFF}{}%
```

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

```
1222 \newcommand*{\glsSetSuffixFF}[1]{%
1223   \renewcommand*{\gls@suffixFF}{#1}}%
```

`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".

```
1225 \ifcsundef{hyperlink}%
1226 {%
1227   \newcommand*{\glsnumberformat}[1]{#1}%
1228 }%
1229 {%
1230   \newcommand*{\glsnumberformat}[1]{\glshypernumber{#1}}%
1231 }
```

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
1232 \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
1233 \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
1234 \newcommand*{\glossarypreamble}{%
1235   \csuse{@glossarypreamble@\currentglossary}%
1236 }
```

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

Code provided by Michael Pock.

```
1237 \newcommand{\setglossarypreamble}[2][\glsdefaulttype]{%
1238   \ifglossaryexists*{#1}%
1239     {\csgdef{@glossarypreamble@#1}{#2}}%
```

```

1240 {\GlossariesWarning{Glossary '#1' is not defined}}%
1241 }

```

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

```

1242 \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.

```

1243 \newcommand*\glossarysection}[2] [\@gls@title]{%
1244   \def\@gls@title{\#2}%
1245   \ifcsundef{phantomsection}%
1246   {%
1247     \glossarysection{\#1}{\#2}%
1248   }%
1249   {%
1250     \p@glossarysection{\#1}{\#2}%
1251   }%
1252   \glsglossarymark{\glossarytoctitle}%
1253 }
```

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

```

1254 \ifcsundef{glossarymark}%
1255 {%
1256   \newcommand{\glsglossarymark}[1]{\glossarymark{\#1}}%
1257 }%
1258 {%
1259   \@ifclassloaded{memoir}%
1260   {%
1261     \newcommand{\glsglossarymark}[1]{%
1262       \ifglsucmark
1263         \markboth{\memUHead{\#1}}{\memUHead{\#1}}%
1264       \else
1265         \markboth{\#1}{\#1}%
1266       \fi
1267     }%
1268   }%
1269 }
```

```

1270 \newcommand{\glsglossarymark}[1]{%
1271   \ifglsucmark
1272     \mkboth{\mfirstucMakeUppercase{#1}}{\mfirstucMakeUppercase{#1}}%
1273   \else
1274     \mkboth{#1}{#1}%
1275   \fi
1276 }
1277 }
1278 }
```

\glossarymark Provided for backward compatibility:

```

1279 \providecommand{\glossarymark}[1]{%
1280   \ifglsucmark
1281     \mkboth{\mfirstucMakeUppercase{#1}}{\mfirstucMakeUppercase{#1}}%
1282   \else
1283     \mkboth{#1}{#1}%
1284   \fi
1285 }
```

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

```

1286 \newcommand*{\setglossarysection}[1]{%
1287 \setkeys{glossaries.sty}{section=#1}}
```

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

glossarysection

```

1288 \newcommand*{\@glossarysection}[2]{%
1289   \ifdefempty{\@glossarysecstar}
1290   {%
1291     \csname\@glossarysec\endcsname[#1]{#2}%
1292   }%
1293   {%
1294     \csname\@glossarysec\endcsname*{#2}%
1295     \@gls@toc{#1}{\@glossarysec}%
1296   }%
```

Do automatic labelling if required

```

1297 \@@glossaryseclabel
1298 }
```

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
1299 \newcommand*{\@p@glossarysection}[2]{%
1300   \glsclearpage
1301   \phantomsection
1302   \ifdefempty\@@glossarysecstar
1303   {%
1304     \csname\@@glossarysec\endcsname{#2}%
1305   }%
1306   {%
1307     \gls@toc{#1}{\@@glossarysec}%
1308     \csname\@@glossarysec\endcsname*{#2}%
1309   }%
1310   \@@glossaryseclabel
1311 }

Do automatic labelling if required

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.
1312 \newcommand*{\gls@doclearpage}{%
1313   \ifthenelse{\equal{\@@glossarysec}{chapter}}{%
1314   {%
1315     \ifcsundef{cleardoublepage}{%
1316     {%
1317       \clearpage
1318     }%
1319     {%
1320       \ifcsdef{if@openright}{%
1321       {%
1322         \if@openright
1323           \cleardoublepage
1324         \else
1325           \clearpage
1326         \fi
1327       }%
1328       {%
1329         \cleardoublepage
1330       }%
1331     }%
1332   }%
1333   {}%
1334 }

\glsclearpage This just calls \gls@doclearpage, but it makes it easier to have a user command so that the user can override it.
1335 \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
1336 \newcommand*{\@gls@toc}[2]{%
1337   \ifglstoc
1338     \ifglsnumberline
1339       \addcontentsline{toc}{#2}{\protect\numberline{}#1}%
1340     \else
1341       \addcontentsline{toc}{#2}{#1}%
1342     \fi
1343   \fi
1344 }
```

## 1.4 Xindy

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

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

```
1345 \newcommand*{\glsnoxindywarning}[1]{%
1346   \GlossariesWarning{Not in xindy mode --- ignoring \string#1}%
1347 }
```

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

```
1348 \newcommand*{\glsnomakeindexwarning}[1]{%
1349   \GlossariesWarning{Not in makeindex mode --- ignoring \string#1}%
1350 }
```

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

```
1351 \ifglsxindy
1352   \edef\@xdyattributes{\string"default\string"}%
1353 \fi
```

`dyattributelist` Comma-separated list of attributes.

```
1354 \ifglsxindy
1355   \edef\@xdyattributelist{}%
1356 \fi
```

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

```
1357 \ifglsxindy
1358   \def\@xdylocref{}%
1359 \fi
```

```

\@gls@ifinlist
1360 \newcommand*{\@gls@ifinlist}[4]{%
1361   \def\@do@ifinlist##1,#1,##2\end@doifinlist{%
1362     \def\@gls@listsuffix{##2}%
1363     \ifx\@gls@listsuffix\@empty
1364       #4%
1365     \else
1366       #3%
1367     \fi
1368   }%
1369   \@do@ifinlist,#2,#1,\end@doifinlist
1370 }

```

**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.

```

1371 \ifglsxindy
1372   \newcommand*{\@xdycounters}{\glscounter}
1373   \newcommand*\GlsAddXdyCounters[1]{%
1374     \@for\@gls@ctr:=#1\do{%

```

Check if already in list before adding.

```

1375   \edef\@do@addcounter{%
1376     \noexpand\@gls@ifinlist{\@gls@ctr}{\@xdycounters}{}}%
1377   {%
1378     \noexpand\edef\noexpand\@xdycounters{\@xdycounters,%
1379     \noexpand\@gls@ctr}%
1380   }%
1381   }%
1382   \@do@addcounter
1383 }
1384 }

```

Only has an effect before \writeist:

```

1385   \@onlypremakeg\GlsAddXdyCounters
1386 \else
1387   \newcommand*\GlsAddXdyCounters[1]{%
1388     \glsnoxindywarning\GlsAddXdyAttribute
1389   }
1390 \fi

```

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

```

1391 \newcommand*{\@disabled@glsaddxdycounters}{%
1392   \PackageError{glossaries}{\string\GlsAddXdyCounters\space
1393   can't be used after \string\GlsAddXdyAttribute}{Move all
1394   occurrences of \string\GlsAddXdyCounters\space before the first
1395   instance of \string\GlsAddXdyAttribute}%
1396 }

```

**AddXdyAttribute** Adds an attribute.

```

1397 \ifglsxindy

```

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

```
1398 \newcommand*\@glsaddxdyattribute[2]{%
  Add to xindy attribute list
```

```
1399 \edef\@xdyattributes{\@xdyattributes ^~J \string"#1\string" ^~J
1400   \string"#2#1\string"}%
```

Add to xindy markup location.

```
1401 \expandafter\toks@\expandafter{\@xdylocref}%
1402 \edef\@xdylocref{\the\toks@ ^~J%
1403   (markup-locref
1404     :open \string"\glstildechar n%
1405       \expandafter\string\csname glsX#2X#1\endcsname
1406       \string" ^~J
1407     :close \string"\string" ^~J
1408     :attr \string"#2#1\string")}%
```

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

```
1409 \expandafter\gdef\csname glsX#2X#1\endcsname##1##2{%
1410   \setentrycounter[##1]{##2}\csname #1\endcsname{##2}%
1411 }%
1412 }
```

High-level command:

```
1413 \newcommand*\GlsAddXdyAttribute[1]{%
```

Add to comma-separated attribute list

```
1414 \ifx\@xdyattributelist\empty
1415   \edef\@xdyattributelist{\#1}%
1416 \else
1417   \edef\@xdyattributelist{\@xdyattributelist,\#1}%
1418 \fi
```

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

```
1419 \for@\this@counter:=\xdycounters\do{%
1420   \protected@edef\gls@do@addxdyattribute{%
1421     \noexpand\@glsaddxdyattribute{\#1}{\this@counter}%
1422   }
1423   \gls@do@addxdyattribute
1424 }%
```

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

```
1425 \let\GlsAddXdyCounters\disabled@glsaddxdycounters
1426 }
```

Only has an effect before `\writeist`:

```
1427 \onlypremakeg\GlsAddXdyAttribute
1428 \else
1429 \newcommand*\GlsAddXdyAttribute[1]{%
1430   \glsnoxindywarning\GlsAddXdyAttribute}
1431 \fi
```

```

finedattributes Add known attributes for all defined counters
1432 \ifglsxindy
1433 \newcommand*{\@gls@addpredefinedattributes}{%
1434   \GlsAddXdyAttribute{glsnumberformat}%
1435   \GlsAddXdyAttribute{textrm}%
1436   \GlsAddXdyAttribute{textsf}%
1437   \GlsAddXdyAttribute{texttt}%
1438   \GlsAddXdyAttribute{textbf}%
1439   \GlsAddXdyAttribute{textmd}%
1440   \GlsAddXdyAttribute{textit}%
1441   \GlsAddXdyAttribute{textup}%
1442   \GlsAddXdyAttribute{textsl}%
1443   \GlsAddXdyAttribute{textsc}%
1444   \GlsAddXdyAttribute{emph}%
1445   \GlsAddXdyAttribute{glshypernumber}%
1446   \GlsAddXdyAttribute{hyperrm}%
1447   \GlsAddXdyAttribute{hypersf}%
1448   \GlsAddXdyAttribute{hypertt}%
1449   \GlsAddXdyAttribute{hyperbf}%
1450   \GlsAddXdyAttribute{hypermd}%
1451   \GlsAddXdyAttribute{hyperit}%
1452   \GlsAddXdyAttribute{hyperup}%
1453   \GlsAddXdyAttribute{hypersl}%
1454   \GlsAddXdyAttribute{hypersc}%
1455   \GlsAddXdyAttribute{hyperemph}%
1456   \GlsAddXdyAttribute{glsignore}%
1457 }
1458 \else
1459   \let\@gls@addpredefinedattributes\relax
1460 \fi

```

dyuseralphabets List of additional alphabets

```
1461 \def\@xdyuseralphabets{}
```

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

```

1462 \ifglsxindy
1463   \newcommand*{\GlsAddXdyAlphabet}[2]{%
1464     \edef\@xdyuseralphabets{%
1465       \@xdyuseralphabets ^^J
1466       (define-alphabet "#1" (#2))}%
1467   }%
1468   \newcommand*{\GlsAddXdyAlphabet}[2]{%
1469     \glsnoxindywarning\GlsAddXdyAlphabet}%
1470 \fi

```

This code is only required for xindy:

```
1471 \ifglsxindy
```

dy@locationlist List of predefined location names.

```
1472 \newcommand*{\@gls@xdy@locationlist}{%
1473     roman-page-numbers,%
1474     Roman-page-numbers,%
1475     arabic-page-numbers,%
1476     alpha-page-numbers,%
1477     Alpha-page-numbers,%
1478     Appendix-page-numbers,%
1479     arabic-section-numbers%
1480 }
```

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.

```
1481 \protected@edef{\gls@roman}{\romannumeral{0}\string"
1482     \string"roman-numbers-lowercase\string" :sep \string"}}%
1483 \onelevel@sanitize{\gls@roman
1484 \edef{\tmp{\string" \string"romannumbers-lowercase\string"
1485     :sep \string"}\%
1486 \onelevel@sanitize{\tmp
1487 \ifx{\tmp{\gls@roman
1488     \expandafter
1489     \edef{\csname{\gls@xdy@Lclass@roman-page-numbers}\endcsname{%
1490         \string"roman-numbers-lowercase\string"}%
1491     }%
1492 \else
1493     \expandafter
1494     \edef{\csname{\gls@xdy@Lclass@roman-page-numbers}\endcsname{%
1495         :sep \string"\@gls@roman\string"}%
1496     }%
1497 \fi
```

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

```
1498 \expandafter\def\csname{\gls@xdy@Lclass@Roman-page-numbers}\endcsname{%
1499     \string"roman-numbers-uppercase\string"}%
1500 }%
```

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

```
1501 \expandafter\def\csname{\gls@xdy@Lclass@arabic-page-numbers}\endcsname{%
1502     \string"arabic-numbers\string"}%
1503 }%
```

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

```
1504 \expandafter\def\csname{\gls@xdy@Lclass@alpha-page-numbers}\endcsname{%
1505     \string"alpha\string"}%
1506 }%
```

```

ha-page-numbers Upper case alphabetical (A, B, ...).
1507 \expandafter\def\csname @gls@xdy@Lclass@Alpha-page-numbers\endcsname{%
1508   \string"ALPHA\string"%
1509 }%

ix-page-numbers Appendix style locations (e.g. A-1, A-2, ..., B-1, B-2, ...). The separator is given by
\@glsAlphacompositor.
1510 \expandafter\def\csname @gls@xdy@Lclass@Appendix-page-numbers\endcsname{%
1511   \string"ALPHA\string"%
1512   :sep \string"\@glsAlphacompositor\string"%
1513   \string"arabic-numbers\string"%
1514 }%

section-numbers Section number style locations (e.g. 1.1, 1.2, ...). The compositor is given by \glscompositor.
1515 \expandafter\def\csname @gls@xdy@Lclass@arabic-section-numbers\endcsname{%
1516   \string"arabic-numbers\string"%
1517   :sep \string"\glscompositor\string"%
1518   \string"arabic-numbers\string"%
1519 }%

userlocationdefs List of additional location definitions (separated by ^^J)
1520 \def\@xdyuserlocationdefs{[]}

erlocationnames List of additional user location names
1521 \def\@xdyuserlocationnames{[]}

      End of xindy-only block:
1522 \fi

xdycrossrefhook Hook used after writing cross-reference class information.
1523 \ifglsxindy
1524 \newcommand\@xdycrossrefhook{}
1525 \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.)
1526 \ifglsxindy
1527 \newcommand*\@GlsAddXdyLocation[3] [] {%
1528   \def\@gls@tmp{\#1}%
1529   \ifx\@gls@tmp\empty
1530     \edef\@xdyuserlocationdefs{%
1531       \@xdyuserlocationdefs ^^J%
1532       (define-location-class \string"\#2\string"^^J\space\space
1533         \space(:sep \string"\{\}\glsopenbrace\string" #3
1534           :sep \string"\glsclosebrace\string"))
1535   }%

```

```

1536     \else
1537         \edef\@xdyuserlocationdefs{%
1538             \@xdyuserlocationdefs ^^J%
1539             (define-location-class \string"\#2\string"^^J\space\space
1540             \space(:sep "\glsopenbrace"
1541                 #1
1542                 :sep "\glsclosebrace\glsopenbrace" #3
1543                 :sep "\glsclosebrace"))
1544         }%
1545     \fi
1546     \edef\@xdyuserlocationnames{%
1547         \@xdyuserlocationnames^^J\space\space\space\space
1548         \string"\#2\string"}%
1549 }

```

Only has an effect before \writeist:

```

1550     \onlypremakeg\GlsAddXdyLocation
1551 \else
1552     \newcommand*\{\GlsAddXdyLocation}[2]{%
1553         \glsnoxindywarning\GlsAddXdyLocation}
1554 \fi

```

#### ationclassorder Define location class order

```

1555 \ifglsxindy
1556     \def\@xdylocationclassorder{^^J\space\space\space
1557         \string"roman-page-numbers\string"^^J\space\space\space
1558         \string"arabic-page-numbers\string"^^J\space\space\space
1559         \string"arabic-section-numbers\string"^^J\space\space\space
1560         \string"alpha-page-numbers\string"^^J\space\space\space
1561         \string"Roman-page-numbers\string"^^J\space\space\space
1562         \string"Alpha-page-numbers\string"^^J\space\space\space
1563         \string"Appendix-page-numbers\string"
1564         \@xdyuserlocationnames^^J\space\space\space
1565         \string"see\string"
1566     }
1567 \fi

```

Change the location order.

#### ationClassOrder

```

1568 \ifglsxindy
1569     \newcommand*\{\GlsSetXdyLocationClassOrder}[1]{%
1570         \def\@xdylocationclassorder{\#1}}
1571 \else
1572     \newcommand*\{\GlsSetXdyLocationClassOrder}[1]{%
1573         \glsnoxindywarning\GlsSetXdyLocationClassOrder}
1574 \fi

```

#### \@xdysortrules Define sort rules

```

1575 \ifglsxindy
1576   \def\@xdysortrules{}
1577 \fi

\GlsAddSortRule Add a sort rule
1578 \ifglsxindy
1579   \newcommand*\GlsAddSortRule[2]{%
1580     \expandafter\toks@\expandafter{\@xdysortrules}%
1581     \protected@edef\@xdysortrules{\the\toks@ ^^J
1582       (sort-rule \string"#1\string" \string"#2\string")}%
1583   }
1584 \else
1585   \newcommand*\GlsAddSortRule[2]{%
1586     \glsnoxindywarning\GlsAddSortRule}
1587 \fi

yrequiredstyles Define list of required styles (this should be a comma-separated list of xindy styles)
1588 \ifglsxindy
1589   \def\@xdyrequiredstyles{tex}
1590 \fi

\GlsAddXdyStyle Add a xindy style to the list of required styles
1591 \ifglsxindy
1592   \newcommand*\GlsAddXdyStyle[1]{%
1593     \edef\@xdyrequiredstyles{\@xdyrequiredstyles,#1}%
1594 \else
1595   \newcommand*\GlsAddXdyStyle[1]{%
1596     \glsnoxindywarning\GlsAddXdyStyle}
1597 \fi

GlsSetXdyStyles Reset the list of required styles
1598 \ifglsxindy
1599   \newcommand*\GlsSetXdyStyles[1]{%
1600     \edef\@xdyrequiredstyles{\#1}%
1601 \else
1602   \newcommand*\GlsSetXdyStyles[1]{%
1603     \glsnoxindywarning\GlsSetXdyStyles}
1604 \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.
1605 \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.
1606 \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. This uses the unstarred form of \ifglossaryexists because ignored glossaries can't be used with xindy.

```
1607 \ifglsxindy
1608   \newcommand*\GlsSetXdyLanguage[2] [\"glsdefaulttype]{%
1609     \ifglossaryexists{#1}{%
1610       \expandafter\def\csname @xdy@#1@language\endcsname{#2}%
1611     }{%
1612       \PackageError{glossaries}{Can't set language type for%
1613         glossary type '#1' --- no such glossary}{%
1614           You have specified a glossary type that doesn't exist}}}
1615 \else
1616   \newcommand*\GlsSetXdyLanguage[2] []{%
1617     \glsnoxindywarning\GlsSetXdyLanguage}
1618 \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.

```
1619 \def\@gls@codepage#1#2{}
```

sSetXdyCodePage Define command to set the code page.

```
1620 \ifglsxindy
1621   \newcommand*\GlsSetXdyCodePage[1]{%
1622     \renewcommand*\gls@codepage{#1}%
1623   }
```

Suggested by egreg:

```
1624   \AtBeginDocument{%
1625     \ifx\gls@codepage\empty
1626       \@ifpackageloaded{fontspec}{\def\gls@codepage{utf8}}{}%
1627     \fi
1628   }
1629 \else
1630   \newcommand*\GlsSetXdyCodePage[1]{%
1631     \glsnoxindywarning\GlsSetXdyCodePage}
1632 \fi
```

xdylettergroups Store letter group definitions.

```
1633 \ifglsxindy
1634   \ifgls@xindy@glsnumbers
1635     \def@\xdylettergroups{(define-letter-group
1636       \"string@glsnumbers\"string\"^J\"space\"space\"space
1637       :prefixes (\\"string\"0\"string\" \\"string\"1\"string"
1638       \\"string\"2\"string\" \\"string\"3\"string\" \\"string\"4\"string"
1639       \\"string\"5\"string\" \\"string\"6\"string\" \\"string\"7\"string"
1640       \\"string\"8\"string\" \\"string\"9\"string\")\"^J\"space\"space\"space
1641       \"@xdynumbergrouporder)}
1642   \else
```

```

1643     \def\@xdylettergroups{}
1644     \fi
1645 \fi

```

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

```

1646 \newcommand*\GlsAddLetterGroup[2]{%
1647   \expandafter\toks@\expandafter{\@xdylettergroups}%
1648   \protected@edef\@xdylettergroups{\the\toks@^J}%
1649   (define-letter-group \string"#1\string"^^J\space\space\space#2)}%
1650 }%

```

## 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.

```

1651 \newcommand*{\forallglossaries}[3][\@glo@types]{%
1652   \@for#2:=#1\do{\ifx#2\empty\else#3\fi}%
1653 }

```

`\forallacronyms`

```

1654 \newcommand*{\forallacronyms}[2]{%
1655   \@for#1:=\@glsacronymlists\do{\ifx#1\empty\else#2\fi}%
1656 }

```

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

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

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

```

1657 \newcommand*{\forglsentries}[3][\glsdefaulttype]{%
1658   \edef\@@glo@list{\csname glolist#1\endcsname}%
1659   \@for#2:=\@@glo@list\do{%
1660     {%
1661       \ifdefempty{#2}{}{#3}%
1662     }%
1663 }

```

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

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

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

```
1664 \newcommand*{\forallglsentries}[3][\@glo@types]{%
1665   \expandafter\forallglossaries\expandafter[\#1]{\@this@glo@}%
1666   {%
1667     \forglsentries[\@this@glo@]{#2}{#3}%
1668   }%
1669 }
```

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

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

where `⟨type⟩` is the glossary's label. The unstarred form will do `⟨false-text⟩` for ignored glossaries. The starred form will do `⟨true-text⟩` for ignored glossaries.

```
1670 \newcommand{\ifglossaryexists}{%
1671   \@ifstar\s@ifglossaryexists\@ifglossaryexists
1672 }
```

`fGLOSSARYexists` Unstarred form only tests the existence of non-ignored glossaries.

```
1673 \newcommand{\@ifglossaryexists}[3]{%
1674   \ifcsundef{@glotype@#1@out}{#3}{#2}%
1675 }
```

`fGLOSSARYexists` Starred form includes ignored glossaries.

```
1676 \newcommand{\s@ifglossaryexists}[3]{%
1677   \ifcsundef{glolist@#1}{#3}{#2}%
1678 }
```

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 redefining `\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`

```
1679 \newcommand*{\glsdetoklabel}[1]{#1}
```

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

```
\ifglsentryexists{\label}{\true}{\false}
```

where `\label` is the entry's label.

```
1680 \newcommand{\ifglsentryexists}[3]{%
1681   \ifcsundef{glo@\glsdetoklabel{\#1}@name}{\#3}{\#2}%
1682 }
```

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

```
\ifglsused{\label}{\true}{\false}
```

where `\label` is the entry's label. If true it will do `\true` otherwise it will do `\false`.

```
1683 \newcommand*\ifglsused[3]{%
1684   \ifbool{glo@\glsdetoklabel{\#1}@flag}{\#2}{\#3}%
1685 }
```

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

```
\glsdoifexists{\label}{\code}
```

Generate an error if entry specified by `\label` doesn't exists, otherwise do `\code`.

```
1686 \newcommand{\glsdoifexists}[2]{%
1687   \ifglsentryexists{\#1}{\#2}{%
1688     \PackageError{glossaries}{Glossary entry '\glsdetoklabel{\#1}'%
1689       has not been defined}{You need to define a glossary entry before you%
1690       can use it.}%
1691 }
```

`\glsdoifnoexists` `\glsdoifnoexists{\label}{\code}`

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

```
1692 \newcommand{\glsdoifnoexists}[2]{%
1693   \ifglsentryexists{\#1}{\#2}{%
1694     \PackageError{glossaries}{Glossary entry '\glsdetoklabel{\#1}' has already%
1695       been defined}{}}%
1696 }
```

```
\glsdoifexistsorwarn{\label}{\code}
```

Generate a warning if entry specified by `\label` doesn't exists, otherwise do `\code`.

```
1697 \newcommand{\glsdoifexistsorwarn}[2]{%
1698   \ifglsentryexists{\#1}{\#2}{%
1699     \GlossariesWarning{Glossary entry '\glsdetoklabel{\#1}'}}
```

```

1700      has not been defined}%
1701  }%
1702 }

```

sdoifexistsordo \glsdoifexistsordo{\label}{\code}{\undef code}

Generate an error and do `\undef code` if entry specified by `\label` doesn't exists, otherwise do `\code`.

```

1703 \newcommand{\glsdoifexistsordo}[3]{%
1704   \ifglsentryexists{#1}{#2}{%
1705     \PackageError{glossaries}{Glossary entry `\\glsdetoklabel{#1}'%
1706     has not been defined}{You need to define a glossary entry before you%
1707     can use it.}%
1708     #3%
1709   }%
1710 }

```

arynoexistsordo \doifglossarynoexistsordo{\label}{\code}{\else code}

If glossary given by `\label` doesn't exist do `\code` otherwise generate an error and do `\else code`.

```

1711 \newcommand{\doifglossarynoexistsordo}[3]{%
1712   \ifglossaryexists*{#1}{%
1713     {%
1714       \PackageError{glossaries}{Glossary type '#1' already exists}{%
1715         #3%
1716     }%
1717     {#2}%
1718 }

```

fglshaschildren \ifglshaschildren{\label}{\true part}{\false part}

This is inefficient as it has to search through all entries to find out which ones have the given entry as its parent. It's much easier to use `bib2gls` and get it to store the list of children that have been indexed (which is likely to be more useful).

```

1719 \newrobustcmd{\ifglshaschildren}[3]{%
1720   \glsdoifexists{#1}{%
1721     {%
1722       \def\do@glshaschildren{#3}%
1723       \edef\@gls@thislabel{\glsdetoklabel{#1}}%
1724       \expandafter\forglsentries\expandafter%
1725         [\csname glo@\@gls@thislabel \type\endcsname]%
1726         {\glo@label}%
1727     {%
1728       \let\cs\glo@parent\glo@\glo@label \parent}%

```

```

1729     \ifdefequal@\gls@thislabel\glo@parent
1730     {%
1731         \def\do@glshaschildren{#2}%
1732         \endfortrue
1733     }%
1734     {}%
1735 }%
1736     \do@glshaschildren
1737 }%
1738 }

```

\ifglshasparent \ifglshasparent{\langle label \rangle}{\langle true part \rangle}{\langle false part \rangle}

```

1739 \newcommand{\ifglshasparent}[3]{%
1740     \glsdoifexists{#1}%
1741     {%
1742         \ifcseempty{\glo@\glsdetoklabel{#1}@parent}{#3}{#2}%
1743     }%
1744 }

```

```

\ifglshasdesc \ifglshasdesc{\langle label \rangle}{\langle true part \rangle}{\langle false part \rangle}
1745 \newcommand*\ifglshasdesc[3]{%
1746     \ifcseempty{\glo@\glsdetoklabel{#1}@desc}%
1747     {#3}%
1748     {#2}%
1749 }

```

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

```

1750 \newcommand*\ifglsdescsuppressed[3]{%
1751     \ifcsequal{\glo@\glsdetoklabel{#1}@desc}{@no@post@desc}%
1752     {#2}%
1753     {#3}%
1754 }

```

```

\ifglshassymbol \ifglshassymbol{\langle label \rangle}{\langle true part \rangle}{\langle false part \rangle}
1755 \newrobustcmd*\ifglshassymbol[3]{%
1756     \letcs{\glo@symbol}{\glo@\glsdetoklabel{#1}@symbol}%
1757     \ifdefempty{\glo@symbol}%
1758     {#3}%
1759     {}%
1760     \ifdefequal{\glo@symbol}{\gls@default@value}%
1761     {#3}%
1762     {#2}%
1763 }%
1764 }

```

```

\ifglshaslong \ifglshaslong{\label}{\truepart}{\falsepart}
1765 \newrobustcmd*\ifglshaslong[3]{%
1766   \letcs{\glo@long}{\glsdetoklabel{#1}@long}%
1767   \ifdefempty{\glo@long}%
1768   {#3}%
1769   {%
1770     \ifdefequal{\glo@long}{\gls@default@value}%
1771     {#3}%
1772     {#2}%
1773   }%
1774 }

\ifglshasshort \ifglshasshort{\label}{\truepart}{\falsepart}
1775 \newrobustcmd*\ifglshasshort[3]{%
1776   \letcs{\glo@short}{\glsdetoklabel{#1}@short}%
1777   \ifdefempty{\glo@short}%
1778   {#3}%
1779   {%
1780     \ifdefequal{\glo@short}{\gls@default@value}%
1781     {#3}%
1782     {#2}%
1783   }%
1784 }

```

\ifglshasfield \ifglshasfield{\field}{\label}{\truepart}{\falsepart}

```

1785 \newrobustcmd*\ifglshasfield[4]{%
1786   \glsdoifexists{#2}%
1787   {%
1788     \letcs{\glo@thisvalue}{\glsdetoklabel{#2}@#1}%

```

First check supplied field label is defined.

```

1789   \ifdef{\glo@thisvalue}%
1790   {%

```

Is defined, so now check if empty.

```

1791     \ifdefempty{\glo@thisvalue}%
1792     {%

```

Is empty, so doesn't have field set.

```

1793     #4%
1794   }%
1795   {%

```

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

```

1796     \ifdefequal{\glo@thisvalue}{\gls@default@value}%
1797     {%

```

Value is set to the default value.

```
1798      #4%
1799      }%
1800      {%
```

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

```
1801      \let\glscurrentfieldvalue@glo@thisvalue
1802      #3%
1803      }%
1804      }%
1805      }%
1806      {%
```

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

```
1807      \gls@fetchfield{\gls@thisfield}{#1}%
If \gls@thisfield is defined, we've found a map. If not, the field supplied doesn't exist.
1808      \ifdef\gls@thisfield
1809      {%
```

Is defined, so now check if empty.

```
1810      \letcs{\glo@thisvalue}{\glsdetoklabel{#2}@gls@thisfield}%
1811      \ifdefempty\glo@thisvalue
1812      {%
```

Is empty so field hasn't been set.

```
1813      #4%
1814      }%
1815      {%
```

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

```
1816      \ifdefequal\glo@thisvalue\gls@default@value
1817      {%
```

Value is set to the default value.

```
1818      #4%
1819      }%
1820      {%
```

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

```
1821      \let\glscurrentfieldvalue@glo@thisvalue
1822      #3%
1823      }%
1824      }%
1825      }%
1826      {%
```

Not defined.

```
1827      \GlossariesWarning{Unknown entry field '#1'}%
1828      #4%
```

```

1829      }%
1830  }%
1831 }%
1832 }

rrentfieldvalue
1833 \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
1834 \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.
1835 \newcommand*{\gls@provide@newglossary}{%
1836   \protected@write\@auxout{}{\string\providecommand\string@glossary[4]{}{}}%
Only need to do this once.
1837 \let\gls@provide@newglossary\relax
1838 }

```

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

```

1839 \newcommand*{\defglsentryfmt}[2][\glsdefaulttype]{%
1840   \csgdef{gls@#1@entryfmt}{#2}%
1841 }

```

`\gls@doentryfmt`

```

1842 \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.

```

1843 \newcommand*{\gls@forbidtexext}[1]{%
1844   \ifboolexpr{test {\ifdefstring{#1}{tex}}%
1845     or test {\ifdefstring{#1}{TEX}}}}%
1846 {%
1847   \def#1{nottex}%
1848   \PackageError{glossaries}{%
1849     {Forbidden ‘.tex’ extension replaced with ‘.nottex’}}{%
1850     {I’m sorry, I can’t allow you to do something so reckless.\MessageBreak
1851       Don’t use ‘.tex’ as an extension for a temporary file.}}%
1852 }%

```

```

1853  {%
1854  }%
1855 }

\gls@gobbleopt Discard optional argument.

1856 \newcommand*{\gls@gobbleopt}{\new@ifnextchar[{\@gls@gobbleopt}{}}
1857 \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
1858 \newcommand*{\newglossary}{\@ifstar\s@newglossary\ns@newglossary}

```

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

```

1859 \newcommand*{\s@newglossary}[2]{%
1860   \ns@newglossary[#1-glg]{#1}{#1-gls}{#1-glo}{#2}%
1861 }

```

`\ns@newglossary` Define the unstarred version.

```

1862 \newcommand*{\ns@newglossary}[5][glg]{%
1863   \doifglossarynoexists{#2}{%
1864     {%

```

Check if default has been set

```

1865   \ifundef\glsdefaulttype{%
1866     {%
1867       \gdef\glsdefaulttype{#2}{%
1868     }{%

```

Add this to the list of glossary types:

```

1869   \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.

```

1870   \expandafter\gdef\csname glolist@#2\endcsname{,}%

```

Store the file extensions:

```
1871 \expandafter\edef\csname @gloctype@#2@log\endcsname{#1}%
1872 \expandafter\edef\csname @gloctype@#2@in\endcsname{#3}%
1873 \expandafter\edef\csname @gloctype@#2@out\endcsname{#4}%
1874 \expandafter\@gls@forbidtexext\csname @gloctype@#2@log\endcsname
1875 \expandafter\@gls@forbidtexext\csname @gloctype@#2@in\endcsname
1876 \expandafter\@gls@forbidtexext\csname @gloctype@#2@out\endcsname
```

Store the title:

```
1877 \expandafter\def\csname @gloctype@#2@title\endcsname{#5}%
1878 \gls@provide@newglossary
1879 \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.

```
1880 \ifcsundef{\gls@#2@entryfmt}%
1881 {%
1882   \defglsentryfmt[#2]{\glsentryfmt}%
1883 }%
1884 {}%
```

Define sort counter if required:

```
1885 \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.)

```
1886 \ifnextchar[\{\gls@setcounter{#2}\}%
1887   {\gls@setcounter{#2}[\glscounter]}%
1888 }%
1889 {%
1890   \gls@gobbleopt
1891 }%
1892 }
```

`\altnewglossary`

```
1893 \newcommand*{\altnewglossary}[3]{%
1894   \newglossary[#2-glg]{#1}{#2-gls}{#2-glo}{#3}%
1895 }
```

Only define new glossaries in the preamble:

```
1896 \onlypreamble{\newglossary}
```

Only define new glossaries before `\makeglossaries`

```
1897 \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 L<sup>A</sup>T<sub>E</sub>X, `\@newglossary` simply ignores its arguments.

```
\@newglossary  
1898 \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
```

```
1899 \def \@gls@setcounter#1[#2]{%  
1900   \expandafter\def\csname @glotype@#1@counter\endcsname{#2}%"
```

Add counter to xindy list, if not already added:

```
1901  \ifglsxindy  
1902    \GlsAddXdyCounters{#2}%"  
1903  \fi  
1904 }
```

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

```
@gls@getcounter
```

```
1905 \newcommand*{\@gls@getcounter}[1]{%  
1906   \csname @glotype@#1@counter\endcsname  
1907 }
```

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

```
1908 \glsdefmain
```

Define the “acronym” glossaries if required.

```
1909 \@gls@do@acronymsdef
```

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

```
1910 \@gls@do@symbolsdef
```

```
1911 \@gls@do@numbersdef
```

```
1912 \@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.

```
1913 \newcommand*{\newignoredglossary}[1]{%  
1914   \ifdefempty{\ignored@glossaries}{%  
1915     {}%  
1916     \edef{\ignored@glossaries}{#1}%"  
1917   }%  
1918   {}%  
1919   \eappto{\ignored@glossaries}{, #1}%"  
1920 }%  
1921 \csgdef{glolist@#1}{, }%"  
1922 \ifcsundef{\gls@#1@entryfmt}{%  
1923   {}%
```

```

1924     \defglsentryfmt [#1]{\glsentryfmt}%
1925   }%
1926   {}%
1927   \ifdefempty{\gls@nohyperlist}
1928   {%
1929     \renewcommand*{\gls@nohyperlist}{#1}%
1930   }%
1931   {}%
1932   \eappto{\gls@nohyperlist}{, #1}%
1933 }%
1934 }

```

ored@glossaries List of ignored glossaries.

```
1935 \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.

```

1936 \newcommand*{\ifignoredglossary}[3]{%
1937   \edef\gls@igtype{#1}%
1938   \expandafter\DTLifinlist\expandafter
1939   {\gls@igtype}{\ignored@glossaries}{#2}{#3}%
1940 }

```

## 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.

```

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

```

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

```

1944 \define@key{glossentry}{description}{%
1945 \def\glo@desc{#1}%
1946 }

```

```

scriptionplural
1947 \define@key{glossentry}{descriptionplural}{%
1948 \def\@glo@descplural{\#1}%
1949 }

sort The sort key needs to be sanitized here (the sort key is provided for makeindex's benefit, not for use in the document). The sort key is optional when defining a new glossary entry. If omitted, the value is given by <name> <description>.
1950 \define@key{glossentry}{sort}{%
1951 \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.
1952 \define@key{glossentry}{text}{%
1953 \def\@glo@text{\#1}%
1954 }

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.
1955 \define@key{glossentry}{plural}{%
1956 \def\@glo@plural{\#1}%
1957 }

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.
1958 \define@key{glossentry}{first}{%
1959 \def\@glo@first{\#1}%
1960 }

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.
1961 \define@key{glossentry}{firstplural}{%
1962 \def\@glo@firstplural{\#1}%
1963 }

s@default@value
1964 \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).

```

```

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

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

```

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

```

1971 \define@key{glossentry}{type}{%
1972 \def\@glo@type{\#1}}

```

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

```

1973 \define@key{glossentry}{counter}{%
1974 \ifcsundef{c@\#1}%
1975 {%
1976 \PackageError{glossaries}%
1977 {There is no counter called ‘#1’}%
1978 {%
1979 The counter key should have the name of a valid counter%
1980 as its value%
1981 }%
1982 }%
1983 {%
1984 \def\@glo@counter{\#1}%
1985 }%
1986 }

```

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

```

1987 \define@key{glossentry}{see}{%
1988 \gls@set@xr@key{see}{\@glo@see}{\#1}%
1989 }

```

```
\gls@set@xr@key{\gls@set@xr@key{\keyname}{\cs}{\value}}
```

Assign a cross-reference key.

```

1990 \newcommand*{\gls@set@xr@key}[3]{%
1991 \renewcommand*{\gls@xr@key}{\#1}%
1992 \gls@checkseeallowed%
1993 \def#2{\#3}%
1994 \glo@seeautonumberlist%
1995 }

```

```

\gls@xr@key
1996 \newcommand*{\gls@xr@key}{see}

```

```

checkseeallowed
1997 \newcommand*{\gls@checkseeallowed}{%
1998   \gls@see@noindex
1999 }

ed@preambleonly
2000 \newcommand*{\gls@checkseeallowed@preambleonly}{%
2001   \GlossariesWarning{glossaries}%
2002   {'\gls@xr@key' key doesn't have any effect when used in the document
2003   environment. Move the definition to the preamble
2004   after \string\makeglossaries\space
2005   or \string\makenoidxglossaries}%
2006 }

parent The parent key specifies the parent entry, if required.
2007 \define@key{glossentry}{parent}{%
2008 \def\@glo@parent{\#1} }

nonumberlist The nonumberlist key suppresses or activates the number list for the given entry.
2009 \define@choicekey{glossentry}{nonumberlist}{%
2010   [\gls@nonumberlist@val\gls@nonumberlist@nr]{true,false}[true]%
2011 }%
2012   \ifcase\gls@nonumberlist@nr\relax
2013     \def\@glo@prefix{\glsnonextpages}%
2014     \gls@savenonumberlist{true}%
2015   \else
2016     \def\@glo@prefix{\glsnextpages}%
2017     \gls@savenonumberlist{false}%
2018   \fi
2019 }

avenonumberlist 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.
2020 \newcommand*{\@gls@savenonumberlist}[1]{}

nitnonumberlist
2021 \newcommand*{\@gls@initnonumberlist}{}%

nitnonumberlist
2022 \newcommand*{\@gls@storenonumberlist}[1]{}

avenonumberlist Allow the nonumberlist value to be saved.
2023 \newcommand*{\@gls@enablesavenonumberlist}{%
2024   \renewcommand*{\@gls@initnonumberlist}{%
2025     \undef\@glo@nonumberlist

```

```

2026 }%
2027 \renewcommand*{\@gls@savenonumberlist}[1]{%
2028   \def\@glo@nonumberlist{##1}%
2029 }%
2030 \renewcommand*{\@gls@storenonumberlist}[1]{%
2031   \ifdef\@glo@nonumberlist
2032   {%
2033     \cslet{\glo@\glsdetoklabel{##1}@nonumberlist}{\@glo@nonumberlist}%
2034   }%
2035   {}%
2036 }%
2037 \appto\@gls@keymap{,{nonumberlist}{nonumberlist}}%
2038 }

```

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

user1

```

2039 \define@key{glossentry}{user1}{%
2040   \def\@glo@useri{#1}%
2041 }

```

user2

```

2042 \define@key{glossentry}{user2}{%
2043   \def\@glo@userii{#1}%
2044 }

```

user3

```

2045 \define@key{glossentry}{user3}{%
2046   \def\@glo@useriii{#1}%
2047 }

```

user4

```

2048 \define@key{glossentry}{user4}{%
2049   \def\@glo@useriv{#1}%
2050 }

```

user5

```

2051 \define@key{glossentry}{user5}{%
2052   \def\@glo@userv{#1}%
2053 }

```

user6

```

2054 \define@key{glossentry}{user6}{%
2055   \def\@glo@uservi{#1}%
2056 }

```

**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.

```

2057 \define@key{glossentry}{short}{%
2058   \def\@glo@short{\#1}%
2059 }

shortplural This key is provided for use by \newacronym.
2060 \define@key{glossentry}{shortplural}{%
2061   \def\@glo@shortpl{\#1}%
2062 }

long This key is provided for use by \newacronym.
2063 \define@key{glossentry}{long}{%
2064   \def\@glo@long{\#1}%
2065 }

longplural This key is provided for use by \newacronym.
2066 \define@key{glossentry}{longplural}{%
2067   \def\@glo@longpl{\#1}%
2068 }

\@glsnoname Define command to generate error if name key is missing.
2069 \newcommand*\@glsnoname{%
2070   \PackageError{glossaries}{name key required in
2071   \string\newglossaryentry\space for entry '\@glo@label'}{You
2072   haven't specified the entry name}%
}

\@glsnodec Define command to generate error if description key is missing.
2073 \newcommand*\@glsnodec{%
2074   \PackageError{glossaries}%
2075   {%
2076     description key required in \string\newglossaryentry\space
2077     for entry '\@glo@label'%
2078   }%
2079   {%
2080     You haven't specified the entry description%
2081   }%
2082 }%


\@glsdefaultplural Now obsolete. Don't use.
2083 \newcommand*\@glsdefaultplural{}{}

\@glsmissingnumberlist Define a command to generate warning when numberlist not set.
2084 \newcommand*\@gls@missingnumberlist[1]{%
2085   ??%
2086   \ifglssavename
2087     \GlossariesWarning{Missing number list for entry '#1'.
2088       Maybe makeglossaries + rerun required}%
2089   \else
2090     \PackageError{glossaries}%
}

```

```

2091 {Package option 'savenumberlist=true' required}%
2092 {%
2093   You must use the 'savenumberlist' package option
2094   to reference location lists.%}
2095 }%
2096 \fi
2097 }

@glsdefaultsort Define command to set default sort.
2098 \newcommand{\@glsdefaultsort}{\@glo@name}

\gls@level Register to increment entry levels.
2099 \newcount\gls@level

@noexpand@field
2100 \newcommand{\@gls@noexpand@field}[3]{%
2101   \expandafter\global\expandafter
2102     \let\csname glo@#1@#2\endcsname#3%
2103 }

noexpand@fields
2104 \newcommand{\@gls@noexpand@fields}[4]{%
2105   \ifcsdef{gls@assign@#3@field}%
2106   {%
2107     \ifdefequal{#4}{\@gls@default@value}%
2108     {%
2109       \edef\@gls@value{\expandonce{#1}}%
2110       \csuse{gls@assign@#3@field}{#2}{\@gls@value}%
2111     }%
2112     {%
2113       \csuse{gls@assign@#3@field}{#2}{#4}%
2114     }%
2115   }%
2116   {%
2117     \ifdefequal{#4}{\@gls@default@value}%
2118     {%
2119       \edef\@gls@value{\expandonce{#1}}%
2120       \@@gls@noexpand@field{#2}{#3}{\@gls@value}%
2121     }%
2122     {%
2123       \@@gls@noexpand@field{#2}{#3}{#4}%
2124     }%
2125   }%
2126 }

ls@expand@field
2127 \newcommand{\@gls@expand@field}[3]{%
2128   \expandafter

```

```

2129   \protected@xdef\csname glo@#1@#2\endcsname{#3}%
2130 }

s@expand@fields
2131 \newcommand{\@gls@expand@fields}[4]{%
2132   \ifcsdef{gls@assign@#3@field}{%
2133     {%
2134       \ifdefequal{#4}{\@gls@default@value}{%
2135         {%
2136           \edef\@gls@value{\expandonce{#1}}%
2137           \csuse{gls@assign@#3@field}{#2}{\@gls@value}%
2138         }%
2139         {%
2140           \expandafter\@gls@startswithexpandonce#4\relax\relax\gls@endcheck
2141         }%
2142           \@@gls@expand@field{#2}{#3}{#4}%
2143         }%
2144         {%
2145           \csuse{gls@assign@#3@field}{#2}{#4}%
2146         }%
2147       }%
2148     }%
2149     {%
2150       \ifdefequal{#4}{\@gls@default@value}{%
2151         {%
2152           \@@gls@expand@field{#2}{#3}{#1}%
2153         }%
2154         {%
2155           \@@gls@expand@field{#2}{#3}{#4}%
2156         }%
2157       }%
2158     }%
}

```

swithexpandonce

```

2159 \def\@gls@expandonce{\expandonce}
2160 \def\@gls@startswithexpandonce#1#2\gls@endcheck#3#4{%
2161   \def\@gls@tmp{#1}%
2162   \ifdefequal{\@gls@expandonce}{\@gls@tmp}{#3}{#4}%
2163 }

```

ls@assign@field

\gls@assign@field{\langle def value \rangle}{\langle label \rangle}{\langle field \rangle}{\langle tmp cs \rangle}

Assigns an entry field. Expansion performed by default (except for name, symbol and description where backward compatibility required). If  $\langle \text{tmp cs} \rangle$  is  $\langle \text{@gls@default@value} \rangle$ ,  $\langle \text{def value} \rangle$  is used instead.

```
2164 \let\gls@assign@field\@gls@expand@fields
```

glsexpandfields Fully expand values when assigning fields (except for specific fields that are overridden by `\glssetnoexpandfield`).  
2165 `\newcommand*{\glsexpandfields}{%`  
2166   `\let\gls@assign@field\@gls@expand@fields`  
2167 }

snoexpandfields Don't expand values when assigning fields (except for specific fields that are overridden by `\glssetexpandfield`).  
2168 `\newcommand*{\glsnoexpandfields}{%`  
2169   `\let\gls@assign@field\@gls@noexpand@fields`  
2170 }

ewglossaryentry Define `\newglossaryentry {<label>} {<key-val list>}`. There are two required fields in `<key-val list>`: name (or parent) and description. (See above.)  
2171 `\newrobustcmd{\newglossaryentry}[2]{%`  
    Check to see if this glossary entry has already been defined:  
2172   `\glsdoifnoexists{#1}%`  
2173   `{%`  
2174     `\gls@defglossaryentry{#1}{#2}%`  
2175   `}%`  
2176 }

ewglossaryentry The definition of `\newglossaryentry` is changed at the start of the document environment.  
The `see` key doesn't work for entries that have been defined in the document environment.  
2177 `\newcommand*{\gls@defdocnewglossaryentry}{%`  
2178   `\let\gls@checkseeallowed\gls@checkseeallowed@preambleonly`  
2179   `\let\newglossaryentry\new@glossaryentry`  
2180 }

deglossaryentry Like `\newglossaryentry` but does nothing if the entry has already been defined.  
2181 `\newrobustcmd{\provideglossaryentry}[2]{%`  
2182   `\ifglsentryexists{#1}%`  
2183   `{}%`  
2184   `\gls@defglossaryentry{#1}{#2}%`  
2185   `}%`  
2186 }

2187 }  
2188 `@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 L<sup>A</sup>T<sub>E</sub>X run. This means that any glossaries at the start of the document can access the entry information.  
2189 `\newrobustcmd{\new@glossaryentry}[2]{%`  
2190   `\ifundef\@gls@deffile`  
2191   `{%`  
2192     `\global\newwrite\@gls@deffile`  
2193     `\immediate\openout\@gls@deffile=\jobname.glsdefs`

```

2194 }%
2195 {}%
2196 \ifglsentryexists{#1}{}%
2197 {%
2198   \gls@defglossaryentry{#1}{#2}%
2199 }%
2200 \gls@writedef{#1}%
2201 }

```

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`.

```
2202 \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.

```
2203 \AtEndDocument{\ifdef{\gls@deffile}{\closeout\gls@deffile}{}}
```

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

```

2204 \newcommand*{\gls@begindocdefs}{}%
2205   \gls@enablesavenonumberlist
2206   \edef{\gls@restoreat}{\noexpand\catcode`\noexpand\@=\number\catcode`@\relax}%
2207   \makeatletter
2208   \InputIfFileExists{\jobname.glsdefs}{}{}%
2209   \gls@restoreat
2210   \undef{\gls@restoreat}
2211   \gls@defdocnewglossaryentry
2212 }

```

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

```

2213 \newcommand*{\gls@writedef}[1]{}%
2214   \immediate\write\gls@deffile
2215 {}%
2216   \string\ifglsentryexists{#1}{}\glspercentchar^~J%
2217   \expandafter\gobble\string\{\glspercentchar^~J%
2218   \string\gls@defglossaryentry{\glsdetoklabel{#1}}\glspercentchar^~J%
2219   \expandafter\gobble\string\{\glspercentchar%
2220 }

```

Write key value information:

```

2221 \cfor{\gls@map}{\gls@keymap}{\do}
2222 {}%
2223   \letcs{\glo@value}{\glsdetoklabel{#1}}\expandafter\secondoftwo\gls@map}%
2224   \ifdef{\glo@value}
2225   {}%
2226     \onelevel@sanitize{\glo@value}
2227     \immediate\write\gls@deffile
2228   {}%
2229     \expandafter\firstoftwo\gls@map

```

```

2230      =\expandafter\@gobble\string\{\glo@value\expandafter\@gobble\string\},%
2231      \glspercentchar
2232  }%
2233 }%
2234 {}%
2235 }%

```

Provide hook:

```

2236 \glswritedefhook
2237 \immediate\write\@gls@deffile
2238 {%
2239     \glspercentchar^~J%
2240     \expandafter\@gobble\string\}\glspercentchar^~J%
2241     \expandafter\@gobble\string\}\glspercentchar%
2242 }%
2243 }

```

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

```

2244 \newcommand*\{@gls@keymap}{%
2245   {name}{name},%
2246   {sort}{sortvalue},% unescaped sort value
2247   {type}{type},%
2248   {first}{first},%
2249   {firstplural}{firstpl},%
2250   {text}{text},%
2251   {plural}{plural},%
2252   {description}{desc},%
2253   {descriptionplural}{descplural},%
2254   {symbol}{symbol},%
2255   {symbolplural}{symbolplural},%
2256   {user1}{useri},%
2257   {user2}{userii},%
2258   {user3}{useriii},%
2259   {user4}{useriv},%
2260   {user5}{userv},%
2261   {user6}{uservi},%
2262   {long}{long},%
2263   {longplural}{longpl},%
2264   {short}{short},%
2265   {shortplural}{shortpl},%
2266   {counter}{counter},%
2267   {parent}{parent}%
2268 }

```

\@gls@fetchfield {\cs} {\field}

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

```
2269 \newcommand*\{@gls@fetchfield}[2]{%
```

```

Ensure user field name is fully expanded
2270 \edef\@gls@thisval{#2}%
Iterate through known mappings until we find the one for this field.
2271 \for\@gls@map:=\@gls@keymap\do{%
2272   \edef\@this@key{\expandafter\@firstoftwo\@gls@map}%
2273   \ifdefequal{\@this@key}{\@gls@thisval}%
2274   {%
Found it.

2275   \edef#1{\expandafter\@secondoftwo\@gls@map}%
Break out of loop.

2276   \endfortrue
2277 }%
2278 {}%
2279 }%
2280 }

```

lsaddstoragekey \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.

```

2281 \newcommand*{\glsaddstoragekey}{\ifstar{\glsaddstoragekey}{\glsaddstoragekey}}
Starred version switches on expansion for this key.

2282 \newcommand*{\sglsaddstoragekey}[1]{%
2283   \key@ifundefined{glossentry}{#1}%
2284   {%
2285     \expandafter\newcommand\expandafter*\expandafter
2286     {\csname gls@assign@\#1@field\endcsname}[2]{%
2287       \gls@expand@field{##1}{#1}{##2}%
2288     }%
2289   }%
2290   {}%
2291   \glsaddstoragekey{#1}%
2292 }

```

Unstarred version doesn't override default expansion.

```

2293 \newcommand*{\glsaddstoragekey}[3]{%
Check the specified key doesn't already exist.

2294 \key@ifundefined{glossentry}{#1}%
2295 {%
Set up the key.

2296 \define@key{glossentry}{#1}{\csdef{@glo@\#1}{##1}}%
2297 \appto{\gls@keymap}{, #1{#1}}%
Set the default value.

2298 \appto{\newglossaryentryprehook}{\csdef{@glo@\#1}{#2}}%

```

Assignment code.

```
2299     \appto{@newglossaryentryposthook}{%
2300         \letcs{\@glo@tmp}{\glo@#1}%
2301         \gls@assign@field{#2}{\glo@label}{#1}{\glo@tmp}%
2302     }%
```

Define the no-link commands.

```
2303     \newcommand*{#3}[1]{\gls@entry@field{##1}{#1}}%
2304     }%
2305     {%
2306     \PackageError{glossaries}{Key '#1' already exists}{}%
2307     }%
2308 }
```

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

Allow user to add their own custom keys.

```
2309 \newcommand*{\glsaddkey}{\ifstar{\sglsaddkey}{\glsaddkey}}
```

Starred version switches on expansion for this key.

```
2310 \newcommand*{\sglsaddkey}[1]{%
2311     \key@ifundefined{glossentry}{#1}{%
2312     {%
2313         \expandafter\newcommand\expandafter*\expandafter
2314             {\csname gls@assign@#1@field\endcsname}[2]{%
2315                 \gls@expand@field{##1}{#1}{##2}}%
2316             }%
2317     }%
2318     {}%
2319     \glsaddkey{#1}%
2320 }
```

Unstarred version doesn't override default expansion.

```
2321 \newcommand*{\glsaddkey}[7]{%
```

Check the specified key doesn't already exist.

```
2322 \key@ifundefined{glossentry}{#1}{%
2323     {%
```

Set up the key.

```
2324     \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
2325     \appto{\gls@keymap}{, {#1}{#1}}%
```

Set the default value.

```
2326 \appto{@newglossaryentryprehook}{\csdef{@glo@#1}{#2}}%
```

Assignment code.

```
2327 \appto{@newglossaryentryposthook}{%
2328     \letcs{\@glo@tmp}{\glo@#1}%
```

```

2329     \gls@assign@field{#2}{\glo@label}{#1}{\glo@tmp}%
2330 }

```

Define the no-link commands.

```

2331 \newcommand*{#3}[1]{\gls@entry@field{##1}{#1}%
2332 \newcommand*{#4}[1]{\Gls@entry@field{##1}{#1}%

```

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

```

2333 \ifcsdef{\gls@user@#1@}%
2334 {%
2335     \PackageError{glossaries}%
2336     {Can't define '\string#5' as helper command
2337      '\expandafter\string\csname \gls@user@#1@\endcsname' already exists}%
2338 }%
2339 }%
2340 {%
2341 
2342     \expandafter\newcommand\expandafter*\expandafter
2343         {\csname \gls@user@#1\endcsname}[2] []{%
2344             \new@ifnextchar[%
2345                 {\csuse{\gls@user@#1@}{##1}{##2}}%
2346                 {\csuse{\gls@user@#1@}{##1}{##2}[]}}%
2347         \csdef{\gls@user@#1@}##1##2[##3]{%
2348             \gls@field@link{##1}{##2}{##3}##3}%
2349 }%
2350 \newrobustcmd*{#5}{%
2351     \expandafter\gls@hyp@opt\csname \gls@user@#1\endcsname}%
2352 }%

```

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

```

2352 \ifcsdef{\Gls@user@#1@}%
2353 {%
2354     \PackageError{glossaries}%
2355     {Can't define '\string#6' as helper command
2356      '\expandafter\string\csname \Gls@user@#1@\endcsname' already exists}%
2357 }%
2358 }%
2359 {%
2360 
2361     \expandafter\newcommand\expandafter*\expandafter
2362         {\csname \Gls@user@#1\endcsname}[2] []{%
2363             \new@ifnextchar[%
2364                 {\csuse{\Gls@user@#1@}{##1}{##2}}%
2365                 {\csuse{\Gls@user@#1@}{##1}{##2}[]}}%
2366         \csdef{\Gls@user@#1@}##1##2[##3]{%
2367             \gls@field@link{##1}{##2}{##3}##3}%
2368 }%
2369 \newrobustcmd*{#6}{%
2370     \expandafter\gls@hyp@opt\csname \Gls@user@#1\endcsname}%

```

Finally the all caps version:

```
2371 \ifcsdef{@GLS@user@#1@}{%
2372   {%
2373     \PackageError{glossaries}{%
2374       {Can't define '\string#7' as helper command
2375       '\expandafter\string\csname @GLS@user@#1@\endcsname' already exists}%
2376     {}%
2377   }%
2378   {}%
2379 
2380   \expandafter\newcommand\expandafter*\expandafter
2381     {\csname @GLS@user@#1\endcsname}[2] []{%
2382       \new@ifnextchar[%
2383         {\csuse{@GLS@user@#1@}{##1}{##2}}%
2384         {\csuse{@GLS@user@#1@}{##1}{##2}[]}}%
2385       \csdef{@GLS@user@#1@}##1##2[##3]{%
2386         \@gls@field@link{##1}{##2}{\mfirstucMakeUppercase{##3{##2}##3}}%
2387       }%
2388       \newrobustcmd*{#7}{%
2389         \expandafter\@gls@hyp@opt\csname @GLS@user@#1\endcsname}%
2390       }%
2391     {}%
2392     \PackageError{glossaries}{Key '#1' already exists}{}%
2393   }%
2394 }
```

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

```
2395 \newcommand{\glsfieldxdef}[3]{%
2396   \glsdoifexists{#1}{%
2397     {%
2398       \edef@glo@label{\glsdetoklabel{#1}}%
2399       \ifcsdef{glo@`@glo@label@#2}{%
2400         {%
2401           \protected@csxdef{glo@`@glo@label@#2}{#3}}%
2402         }%
2403       {}%
2404       \PackageError{glossaries}{Key '#2' doesn't exist}{}%
2405     }%
2406   }%
2407 }
```

```
\glsfieldedef{\glsfieldedef{\label}{\field}{\definition}}
```

```
2408 \newcommand{\glsfielddef}[3]{%
2409   \glsdoifexists{#1}%
2410 {%
2411   \edef\@glo@label{\glsdetoklabel{#1}}%
2412   \ifcsdef{glo@\@glo@label}{#2}%
2413 {%
2414     \protected@csedef{glo@\@glo@label}{#2}{#3}%
2415   }%
2416 {%
2417   \PackageError{glossaries}{Key '#2' doesn't exist}{}%
2418 }%
2419 }%
2420 }
```

```
\glsfieldgdef{\langle label \rangle}{\langle field \rangle}{\langle definition \rangle}
```

```

2421 \newcommand{\glsfieldgdef}[3]{%
2422   \glsdoifexists{#1}%
2423   {%
2424     \edef\@glo@label{\glsdetoklabel{#1}}%
2425     \ifcsdef{\glo@\@glo@label}{#2}%
2426     {%
2427       \expandafter\gdef\csname glo@\@glo@label \endcsname{#3}%
2428     }%
2429     {%
2430       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2431     }%
2432   }%
2433 }

```

```
\glsfielddef{\langle label \rangle}{\langle field \rangle}{\langle definition \rangle}
```

```
2434 \newcommand{\glsfielddef}[3]{%
2435   \glsdoifexists{#1}{%
2436     {%
2437       \edef\@glo@label{\glsdetoklabel{#1}}%
2438       \ifcsdef{\glo@\@glo@label}{#2}{%
2439         {%
2440           \expandafter\def\csname glo@\@glo@label\endcsname{#3}%
2441         }%
2442         {%
2443           \PackageError{glossaries}{Key '#2' doesn't exist}{}%
2444         }%
2445       }%
2446     }%
```

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

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

```
2447 \newcommand{\glsfieldfetch}[3]{%
2448   \glsdoifexists{#1}%
2449   {%
2450     \edef\@glo@\label{\glsdetoklabel{#1}}%
2451     \ifcsdef{glo@\@glo@\label}{#2}%
2452     {%
2453       \letcs#3{glo@\@glo@\label}{#2}%
2454     }%
2455     {%
2456       \PackageError{glossaries}{Key '#2' doesn't exist}{}%
2457     }%
2458   }%
2459 }
```

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

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

```
2460 \newcommand{\ifglsfieldeq}[5]{%
2461   \glsdoifexists{#1}%
2462   {%
2463     \edef\@glo@\label{\glsdetoklabel{#1}}%
2464     \ifcsdef{glo@\@glo@\label}{#2}%
2465     {%
2466       \ifcsstring{glo@\@glo@\label}{#2}{#3}{#4}{#5}%
2467     }%
2468     {%
2469       \PackageError{glossaries}{Key '#2' doesn't exist}{}%
2470     }%
2471   }%
2472 }
```

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

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

```
2473 \newcommand{\ifglsfielddefeq}[5]{%
2474   \glsdoifexists{#1}%
2475   {%
2476     \edef\@glo@\label{\glsdetoklabel{#1}}%
2477     \ifcsdef{glo@\@glo@\label}{#2}%
2478     {%
2479       \expandafter\ifdefstrequal
2480         \csname glo@\@glo@\label\endcsname{#3}{#4}{#5}%
2481     }%
2482 }
```

```

2482     {%
2483         \PackageError{glossaries}{Key '#2' doesn't exist}{}
2484     }%
2485 }%
2486 }

```

```
\ifglsfieldcseq{\label}{\field}{\csname}{\true}{\false}
```

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

```

2487 \newcommand{\ifglsfieldcseq}[5]{%
2488     \glsdoifexists{#1}%
2489     {%
2490         \edef\@glo@label{\glsdetoklabel{#1}}%
2491         \ifcsdef{glo@\@glo@label}{#2}%
2492         {%
2493             \ifcsstrequal{glo@\@glo@label}{#2}{#3}{#4}{#5}%
2494         }%
2495     {%
2496         \PackageError{glossaries}{Key '#2' doesn't exist}{}
2497     }%
2498 }%
2499 }

```

`glswritedefhook`

```
2500 \newcommand*{\glswritedefhook}{}%
```

`gls@assign@desc`

```

2501 \newcommand*{\gls@assign@desc}[1]{%
2502     \gls@assign@field{}{#1}{desc}{\@glo@desc}%
2503     \gls@assign@field{\@glo@desc}{#1}{descplural}{\@glo@descplural}%
2504 }

```

`ewglossaryentry`

```

2505 \newcommand{\longnewglossaryentry}[3]{%
2506     \glsdoifnoexists{#1}%
2507     {%
2508         \bgroup
2509             \let\@org@newglossaryentryprehook\@newglossaryentryprehook
2510             \long\def\@newglossaryentryprehook{%
2511                 \long\def\@glo@desc{#3}\leavevmode\nskip\nopostdesc}%
2512                 \org@newglossaryentryprehook
2513             }%
2514             \renewcommand*{\gls@assign@desc}[1]{%
2515                 \global\cslet{\glo@\glsdetoklabel{#1}@desc}{\@glo@desc}%
2516                 \global\cslet{\glo@\glsdetoklabel{#1}@descplural}{\@glo@desc}%
2517             }%
2518             \gls@defglossaryentry{#1}{#2}%
2519         \egroup

```

```
2520  }
2521 }
```

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

```
2522 \onlypreamble{\longnewglossaryentry}
```

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

```
2523 \newcommand{\longprovideglossaryentry}[3]{%
2524   \ifglsentryexists{#1}{\%
2525     {\longnewglossaryentry{#1}{#2}{#3}}%
2526   }%
2527 \onlypreamble{\longprovideglossaryentry}
```

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

Defines a new entry without checking if it already exists.

```
2528 \newcommand{\gls@defglossaryentry}[2]{%
```

Prevent any further use of `\GlsSetQuote`:

```
2529 \let\GlsSetQuote\gls@nosetquote
```

Store label

```
2530 \edef@glo@label{\glsdetoklabel{#1}}%
```

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

```
2531 \let\glslabel@glo@label
```

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

```
2532 \let@glo@name@glsnoname
2533 \let@glo@desc@glsnodedesc

2534 \let@glo@descplural@gls@default@value
2535 \let@glo@type@gls@default@value
2536 \let@glo@symbol@gls@default@value

2537 \let@glo@symbolplural@gls@default@value
2538 \let@glo@text@gls@default@value
2539 \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.)

```
2540 \let@glo@first@gls@default@value
2541 \let@glo@firstplural@gls@default@value
```

Set the default sort:

```
2542 \let@glo@sort@gls@default@value
```

Set the default counter:

```
2543 \let\@glo@counter\@gls@default@value
2544 \def\@glo@see{}%
2545 \def\@glo@parent{}%
2546 \def\@glo@prefix{}%
```

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

```
2547 \@gls@initnonumberlist
2548 \def\@glo@useri{}%
2549 \def\@glo@userii{}%
2550 \def\@glo@useriii{}%
2551 \def\@glo@useriv{}%
2552 \def\@glo@userv{}%
2553 \def\@glo@uservi{}%
2554 \def\@glo@short{}%
2555 \def\@glo@shortpl{}%
2556 \def\@glo@long{}%
2557 \def\@glo@longpl{}%
```

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

```
2558 @newglossaryentryprehook
```

Extract key-val information from third parameter:

```
2559 \setkeys{glossentry}{#2}%
```

Check there is a default glossary.

```
2560 \ifundef\glsdefaulttype
2561 {%
2562   \PackageError{glossaries}%
2563   {No default glossary type (have you used ‘nomain’ by mistake?)}%
2564   {If you use package option ‘nomain’ you must define
2565    a new glossary before you can define entries}%
2566 }%
2567 {}%
```

Assign type. This must be fully expandable

```
2568 \gls@assign@field{\glsdefaulttype}{\@glo@label}{type}{\@glo@type}%
2569 \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.

```
2570 \ifcsundef{glolist@\@glo@type}%
2571 {%
2572   \PackageError{glossaries}%
2573   {Glossary type ‘\@glo@type’ has not been defined}%
2574   {You need to define a new glossary type, before making entries
2575    in it}%
2576 }%
```

```

2576  }%
2577  {%
    Check if it's an ignored glossary
2578      \ifignoredglossary\@glo@type
2579      {%
        The description may be omitted for an entry in an ignored glossary.
2580          \ifx\@glo@desc\@glsnodec
2581              \let\@glo@desc\@empty
2582              \fi
2583          }%
2584          {%
2585          }%
2586          \protected@edef\@glolist@{\csname glolist@\@glo@type\endcsname}%
2587          \expandafter\xdef\csname glolist@\@glo@type\endcsname{%
2588              \@glolist@{\@glo@label},}%
2589      }%
2590
        Initialise level to 0.
2591 \gls@level=0\relax
    Has this entry been assigned a parent?
2592 \ifx\@glo@parent\@empty
        Doesn't have a parent. Set \glo@<label>@parent to empty.
2593 \else
    Has a parent. Check to ensure this entry isn't its own parent.
2594 \ifdefequal\@glo@label\@glo@parent%
2595 {%
        \PackageError{glossaries}{Entry '\@glo@label' can't be its own parent}{}%
2596         \def\@glo@parent{}%
2597         \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%
2598     }%
2599     {%
2600     }%
2601
        Check the parent exists:
2602 \ifglsentryexists{\@glo@parent}%
2603 {%
        Parent exists. Set \glo@<label>@parent.
2604 \expandafter\xdef\csname glo@\@glo@label @parent\endcsname{%
2605             \@glo@parent}%
2606
        Determine level.
2607 \gls@level=\csname glo@\@glo@parent @level\endcsname\relax
2608 \advance\gls@level by 1\relax
    If name hasn't been specified, use same as the parent name
2609 \ifx\@glo@name\@glsnoname
2610     \expandafter\let\expandafter\@glo@name
2611         \csname glo@\@glo@parent @name\endcsname

```

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

```
2610      \ifx\@glo@plural\@gls@default@value
2611          \expandafter\let\expandafter\@glo@plural
2612              \csname glo@\@glo@parent @plural\endcsname
2613      \fi
2614  \fi
2615 }%
2616 {%
```

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

```
2617      \PackageError{glossaries}%
2618  {%
2619      Invalid parent '\@glo@parent'
2620      for entry '\@glo@label' - parent doesn't exist%
2621 }%
2622 {%
2623     Parent entries must be defined before their children%
2624 }%
2625     \def\@glo@parent{}%
2626     \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%
2627 }%
2628 }%
2629 \fi
```

Set the level for this entry

```
2630 \expandafter\xdef\csname glo@\@glo@label @level\endcsname{\number\gls@level}%
```

Define commands associated with this entry:

```
2631 \gls@assign@field{\@glo@name}{\@glo@label}{sortvalue}{\@glo@sort}%
2632 \letcs\@glo@sort{\glo@\@glo@label}{sortvalue}%
2633 \gls@assign@field{\@glo@name}{\@glo@label}{text}{\@glo@text}%
2634 \expandafter\gls@assign@field\expandafter
2635     {\csname glo@\@glo@label @text\endcsname\glspluralsuffix}%
2636     {\@glo@label}{plural}{\@glo@plural}%
2637 \expandafter\gls@assign@field\expandafter
2638     {\csname glo@\@glo@label @text\endcsname}%
2639     {\@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.

```
2640 \ifx\@glo@first\@gls@default@value
2641     \expandafter\gls@assign@field\expandafter
2642         {\csname glo@\@glo@label @plural\endcsname}%
2643         {\@glo@label}{firstpl}{\@glo@firstplural}%
2644 \else
2645     \expandafter\gls@assign@field\expandafter
2646         {\csname glo@\@glo@label @first\endcsname\glspluralsuffix}%
2647         {\@glo@label}{firstpl}{\@glo@firstplural}%
2648 \fi
2649 \ifcsundef{@glotype@\@glo@type @counter}%

```

```

2650  {%
2651    \def\@glo@defaultcounter{\glscounter}%
2652  }%
2653  {%
2654    \letcs\@glo@defaultcounter{@glotype@\@glo@type \counter}%
2655  }%
2656  \gls@assign@field{\@glo@defaultcounter}{\@glo@label}{counter}{\@glo@counter}%
2657  \gls@assign@field{}{\@glo@label}{useri}{\@glo@useri}%
2658  \gls@assign@field{}{\@glo@label}{userii}{\@glo@userii}%
2659  \gls@assign@field{}{\@glo@label}{useriii}{\@glo@useriii}%
2660  \gls@assign@field{}{\@glo@label}{useriv}{\@glo@useriv}%
2661  \gls@assign@field{}{\@glo@label}{userv}{\@glo@userv}%
2662  \gls@assign@field{}{\@glo@label}{uservi}{\@glo@uservi}%
2663  \gls@assign@field{}{\@glo@label}{short}{\@glo@short}%
2664  \gls@assign@field{}{\@glo@label}{shortpl}{\@glo@shortpl}%
2665  \gls@assign@field{}{\@glo@label}{long}{\@glo@long}%
2666  \gls@assign@field{}{\@glo@label}{longpl}{\@glo@longpl}%
2667  \ifx\@glo@name\glsnoname
2668    \glsnoname
2669    \let\@gloname\@gls@default@value
2670  \fi
2671  \gls@assign@field{}{\@glo@label}{name}{\@glo@name}%

```

Set default numberlist if not defined:

```

2672  \ifcsundef{glo@\@glo@label @numberlist}%
2673  {%
2674    \csxdef{glo@\@glo@label @numberlist}{}%
2675    \noexpand\@gls@missingnumberlist{\@glo@label}}%
2676  }%
2677  {}%

```

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

```
2678  \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.

```

2679  \def\@glo@@desc{\@glo@first}%
2680  \ifx\@glo@desc\@glo@@desc
2681    \let\@glo@desc\@glo@first
2682  \fi
2683  \ifx\@glo@desc\glsnodesc
2684    \glsnodesc
2685    \let\@glodesc\@gls@default@value
2686  \fi
2687  \gls@assign@desc{\@glo@label}%

```

Set the sort key for this entry:

```

2688  \gls@defsort{\@glo@type}{\@glo@label}%
2689  \def\@glo@@symbol{\@glo@text}%
2690  \ifx\@glo@symbol\@glo@@symbol

```

```

2691     \let\@glo@symbol\@glo@text
2692     \fi
2693     \gls@assign@field{\relax}{\@glo@label}{symbol}{\@glo@symbol}%
2694     \expandafter
2695         \gls@assign@field\expandafter
2696             {\csname glo@\@glo@label \symbol\endcsname}%
2697             {\@glo@label}{symbolplural}{\@glo@symbolplural}%

```

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

```

2698     \expandafter\xdef\csname glo@\@glo@label @flagfalse\endcsname{%
2699         \noexpand\global
2700             \noexpand\let\expandafter\noexpand
2701                 \csname ifglo@\@glo@label @flag\endcsname\noexpand\iffalse
2702             }%
2703     \expandafter\xdef\csname glo@\@glo@label @flagtrue\endcsname{%
2704         \noexpand\global
2705             \noexpand\let\expandafter\noexpand
2706                 \csname ifglo@\@glo@label @flag\endcsname\noexpand\iftrue
2707             }%
2708     \csname glo@\@glo@label @flagfalse\endcsname

```

Sort out any cross-referencing if required.

```
2709     \@glo@autosee
```

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

```

2710     \ifignoredglossary\@glo@type
2711     {%
2712         \csdef{glo@\@glo@label @index}{}%
2713     }
2714     {%
2715         \do@glo@storeentry{\@glo@label}%
2716     }%

```

Define entry counters if enabled:

```
2717     \@newglossaryentry@defcounters
```

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

```

2718     \@newglossaryentry@posthook
2719 }
```

\@glo@autosee Automatically implement \glssee.

```

2720 \newcommand*{\@glo@autosee}{{%
2721     \ifdefvoid{\@glo@see}{}{%
2722     {%
2723         \protected@edef{\do@glssee}{%
2724             \noexpand\@gls@fixbraces\noexpand\@glo@list\@glo@see\noexpand\@nil
2725             \noexpand\expandafter\noexpand\@glssee\noexpand\@glo@list{\@glo@label}}%
2726         \do@glssee
2727     }%
2728     \@glo@autoseehook

```

```

2729 }%}

glo@autoseehook
2730 \newcommand*{\@glo@autoseehook}{}}

aryentryprehook Allow extra information to be added to glossary entries:
2731 \newcommand*{\@newglossaryentryprehook}{}}

ryentryposthook Allow extra information to be added to glossary entries:
2732 \newcommand*{\@newglossaryentryposthook}{}}

try@defcounters
2733 \newcommand*{\@newglossaryentry@defcounters}{}}

\glsmoveentry Moves entry whose label is given by first argument to the glossary named in the second argument.
2734 \newcommand*{\glsmoveentry}[2]{%
2735   \edef\@glo@thislabel{\glsmovetoklabel{#1}}%
2736   \edef\glo@type{\csname glo@\@glo@thislabel @type\endcsname}%
2737   \def\glo@list{,}%
2738   \forglsentries[\glo@type]{\glo@label}{%
2739     {%
2740       \ifdefeqequal\@glo@thislabel\glo@label
2741         {}{\eappto\glo@list{\glo@label,}}%
2742     }%
2743   \cslet{glo@list@\glo@type}{\glo@list}%
2744   \csdef{glo@\@glo@thislabel @type}{#2}%
2745 }

```

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

```

2746 \ifglsxindy
2747   \newcommand*{\@glossaryentryfield}{\string\\glossentry}
2748 \else
2749   \newcommand*{\@glossaryentryfield}{\string\glossentry}
2750 \fi

```

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

```

2751 \ifglsxindy
2752   \newcommand*{\@glossarysubentryfield}{%
2753     \string\\subglossentry}
2754 \else
2755   \newcommand*{\@glossarysubentryfield}{%
2756     \string\subglossentry}
2757 \fi

```

```
@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.)

```
2758 \newcommand{\@glo@storeentry}[1]{%
```

Escape makeindex/xindy special characters in the label:

```
2759 \edef\@glo@esclabel{#1}%
```

```
2760 \@gls@checkmkidxchars\@glo@esclabel
```

Get the sort string and escape any special characters

```
2761 \protected\edef\@glo@sort{\csname glo@\#1@sort\endcsname}%
```

```
2762 \@gls@checkmkidxchars\@glo@sort
```

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

```
2763 \@gls@checkmkidxchars\@glo@prefix
```

Get the parent, if one exists

```
2764 \edef\@glo@parent{\csname glo@\#1@parent\endcsname}%
```

Write the information to the glossary file.

```
2765 \ifglsxindy
```

Store using xindy syntax.

```
2766 \ifx\@glo@parent\empty
```

Entry doesn't have a parent

```
2767 \expandafter\protected\def\csname glo@\#1@index\endcsname{%
```

```
2768 (\string"\@glo@sort\string" %
```

```
2769 \string"\@glo@prefix\@glossaryentryfield{\@glo@esclabel}\string") %
```

```
2770 }%
```

```
2771 \else
```

Entry has a parent

```
2772 \expandafter\protected\def\csname glo@\#1@index\endcsname{%
```

```
2773 \csname glo@\@glo@parent @index\endcsname
```

```
2774 (\string"\@glo@sort\string" %
```

```
2775 \string"\@glo@prefix\@glossarysubentryfield
```

```
2776 {\csname glo@\#1@level\endcsname}{\@glo@esclabel}\string") %
```

```
2777 }%
```

```
2778 \fi
```

```
2779 \else
```

Store using makeindex syntax.

```
2780 \ifx\@glo@parent\empty
```

Sanitize \@glo@prefix

```
2781 \onelevel@sanitize\@glo@prefix
```

Entry doesn't have a parent

```
2782     \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2783         \@glo@sort@gls@actualchar@glo@prefix
2784         \@glossaryentryfield{\@glo@esclabel}%
2785     }%
2786 }
```

Entry has a parent

```
2787     \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2788         \csname glo@\@glo@parent \index\endcsname\@gls@levelchar
2789         \@glo@sort@gls@actualchar@glo@prefix
2790         \@glossarysubentryfield
2791         {\csname glo@#1@level\endcsname}{\@glo@esclabel}%
2792     }%
2793     \fi
2794 \fi
2795 }
```

## 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`

```
2796 \AtBeginDocument{%
2797   \@ifpackageloaded{amsmath}{%
2798     {\let\gls@ifnotmeasuring@gls@ifnotmeasuring}{%
2799       {}%
2800     }%
2801     \newcommand*{\@gls@ifnotmeasuring}[1]{%
2802       \ifmeasuring@
2803       \else
2804         #1%
2805       \fi
2806     }%
2807     \newcommand*\gls@ifnotmeasuring[1]{#1}
```

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

```
2808 \def\@gls@patchtabularx#1\hbox#2#3{!!{%
2809   \def\TX@trial##1{#1\hbox{\let\glsunset@gobble#2}#3}%
2810 }%
2811 \newcommand*\glspatchtabularx{%
2812   \ifdef\TX@trial
2813   {%
2814     \expandafter\@gls@patchtabularx\TX@trial{##1}!!%
```

```
2815   \let\glspatchtabularx\relax
2816 }%
2817 {}%
2818 }
```

\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.

```
2819 \newcommand*{\glsreset}[1]{%
2820   \gls@ifnotmeasuring
2821   {%
2822     \glsdoifexists{#1}%
2823     {%
2824       \glsreset{#1}%
2825     }%
2826   }%
2827 }
```

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

```
2828 \newcommand*{\glslocalreset}[1]{%
2829   \gls@ifnotmeasuring
2830   {%
2831     \glsdoifexists{#1}%
2832     {%
2833       \glslocalreset{#1}%
2834     }%
2835   }%
2836 }
```

\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.

```
2837 \newcommand*{\glsunset}[1]{%
2838   \gls@ifnotmeasuring
2839   {%
2840     \glsdoifexists{#1}%
2841     {%
2842       \glsunset{#1}%
2843     }%
2844   }%
2845 }
```

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

```
2846 \newcommand*{\glslocalunset}[1]{%
2847   \gls@ifnotmeasuring
2848   {%
2849     \glsdoifexists{#1}%
2850     {%
2851       \glslocalunset{#1}%
2852     }%
```

```

2853  }%
2854 }

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

2855 \newcommand*{\@glslocalunset}{\@glslocalunset}

@@glslocalunset Local unset without checks.

2856 \newcommand*{\@@glslocalunset}[1]{%
2857   \expandafter\let\csname ifglo@\glsdetoklabel{#1}@flag\endcsname\iftrue
2858 }

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

2859 \newcommand*{\@glsunset}{\@glsunset}

\@@glsunset Global unset without checks.

2860 \newcommand*{\@@glsunset}[1]{%
2861   \expandafter\global\csname glo@\glsdetoklabel{#1}@flagtrue\endcsname
2862 }

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

2863 \newcommand*{\@glslocalreset}{\@glslocalreset}

@@glslocalreset Local reset without checks.

2864 \newcommand*{\@@glslocalreset}[1]{%
2865   \expandafter\let\csname ifglo@\glsdetoklabel{#1}@flag\endcsname\iffalse
2866 }

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

2867 \newcommand*{\@glsreset}{\@glsreset}

\@@glsreset Global reset without checks.

2868 \newcommand*{\@@glsreset}[1]{%
2869   \expandafter\global\csname glo@\glsdetoklabel{#1}@flagfalse\endcsname
2870 }

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

\glsresetall[<glossary-list>]

\glsresetall

2871 \newcommand*{\glsresetall}[1][\@glo@types]{%
2872   \forallglsentries[#1]{\glsentry}%
2873   {%
2874     \glsreset{\glsentry}%
2875   }%
2876 }

```

As above, but with only a local effect:

```
lslocalresetall
2877 \newcommand*{\glslocalresetall}[1] [\\@glo@types]{%
2878   \\forallglsentries[#1]{\\glsentry}%
2879   {%
2880     \\glslocalreset{\\glsentry}%
2881   }%
2882 }
```

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

```
\glsunsetall
2883 \newcommand*{\glsunsetall}[1] [\\@glo@types]{%
2884   \\forallglsentries[#1]{\\glsentry}%
2885   {%
2886     \\glsunset{\\glsentry}%
2887   }%
2888 }
```

As above, but with only a local effect:

```
lslocalunsetall
2889 \newcommand*{\glslocalunsetall}[1] [\\@glo@types]{%
2890   \\forallglsentries[#1]{\\glsentry}%
2891   {%
2892     \\glslocalunset{\\glsentry}%
2893   }%
2894 }
```

## 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 L<sup>A</sup>T<sub>E</sub>X counter or even an explicit T<sub>E</sub>X count register but is just a macro. Any of the commands that use \\glsunset or \\glslocalunset, such as \\gls, will automatically increment this value. Commands that don’t modify the first use flag (such as \\glstext or \\glsentrytext) don’t modify this value.

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

```
2895 \\newcommand*{\\@newglossaryentry@defcounters}{%
2896   \\csdef{glo@\\@glo@label}{currcount}{0}%
2897   \\csdef{glo@\\@glo@label}{prevcount}{0}%
2898 }
```

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

```
2899 \newcommand{\glsenableentrycount}{%
  Enable new entry fields.
2900 \let\@newglossaryentry\defcounters\@@newglossaryentry\defcounters
  Disable \newglossaryentry in the document environment.
2901 \renewcommand{\gls@defdocnewglossaryentry}{%
2902   \renewcommand{\newglossaryentry}[2]{%
2903     \PackageError{glossaries}{\string\newglossaryentry\space
2904       may only be used in the preamble when entry counting has
2905       been activated}{If you use \string\glsenableentrycount\space
2906       you must place all entry definitions in the preamble not in
2907       the document environment}%
2908   }%
2909 }%
```

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

```
2910 \newcommand{\glsentrycurrcount}[1]{%
2911   \ifcsundef{\glo@\glsdetoklabel{\##1}@currcount}{%
2912     {0}{\gls@entry@field{\##1}{currcount}}%
2913   }%
2914 \newcommand{\glsentryprevcount}[1]{%
2915   \ifcsundef{\glo@\glsdetoklabel{\##1}@prevcount}{%
2916     {0}{\gls@entry@field{\##1}{prevcount}}%
2917   }%
```

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

```
2918 \renewcommand{\@glsunset}[1]{%
2919   \@@glsunset{\##1}%
2920   \gls@increment@currcount{\##1}%
2921 }%
2922 \renewcommand{\@glslocalunset}[1]{%
2923   \@@glslocalunset{\##1}%
2924   \gls@local@increment@currcount{\##1}%
2925 }%
2926 \renewcommand{\@glsreset}[1]{%
2927   \@@glsreset{\##1}%
2928   \csgdef{\glo@\glsdetoklabel{\##1}@currcount}{0}%
2929 }%
2930 \renewcommand{\@glslocalreset}[1]{%
2931   \@@glslocalreset{\##1}%
2932   \csdef{\glo@\glsdetoklabel{\##1}@currcount}{0}%
2933 }%
```

Alter behaviour of \cgls. (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.)

```
2934 \def\@cgls{\##1\##2\##3}{%
2935   \ifnum\glsentryprevcount{\##2}=1\relax
2936     \cglsformat{\##2}{\##3}%
}
```

```

2937     \glsunset{##2}%
2938     \else
2939         \gls@{##1}{##2}[##3]%
2940     \fi
2941 }%

```

Similarly for the analogous commands. No case change plural:

```

2942 \def\cglsp{\##1##2[##3]{%
2943     \ifnum\glsentryprevcount{##2}=1\relax
2944         \cglspformat{##2}{##3}%
2945         \glsunset{##2}%
2946     \else
2947         \glspl{\##1}{##2}[##3]%
2948     \fi
2949 }%

```

First letter uppercase singular:

```

2950 \def\cGls{\##1##2[##3]{%
2951     \ifnum\glsentryprevcount{##2}=1\relax
2952         \cGlsformat{##2}{##3}%
2953         \glsunset{##2}%
2954     \else
2955         \Gls{\##1}{##2}[##3]%
2956     \fi
2957 }%

```

First letter uppercase plural:

```

2958 \def\cGlspl{\##1##2[##3]{%
2959     \ifnum\glsentryprevcount{##2}=1\relax
2960         \cGlsplformat{##2}{##3}%
2961         \glsunset{##2}%
2962     \else
2963         \Glspl{\##1}{##2}[##3]%
2964     \fi
2965 }%

```

Write information to aux file at the end of the document

```
2966 \AtEndDocument{\gls@write@entrycounts}%
```

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

```

2967 \renewcommand*\gls@entry@count}[2]{%
2968     \csgdef{\glo@\glsdetoklabel{##1}@prevcount}{##2}%
2969 }%

```

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

```

2970 \let\glsenableentrycount\relax
2971 }
2972 \onlypreamble\glsenableentrycount

```

ement@currcount

```

2973 \newcommand*{\@gls@increment@currcount}[1]{%
2974   \csxdef{glo@\glsdetoklabel{#1}@currcount}{%
2975     \number\numexpr\glsentrycurrcount{#1}+1}%
2976 }

ement@currcount
2977 \newcommand*{\@gls@local@increment@currcount}[1]{%
2978   \csedef{glo@\glsdetoklabel{#1}@currcount}{%
2979     \number\numexpr\glsentrycurrcount{#1}+1}%
2980 }

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.)
2981 \newcommand*{\@gls@write@entrycounts}{%
2982   \immediate\write\auxout
2983   {\string\providecommand*{\string@gls@entry@count}[2]{}}
2984   \forallglsentries{\glsentry}{%
2985     \ifglsused{\glsentry}{%
2986       \immediate\write\auxout
2987       {\string@gls@entry@count{\glsentry}{\glsentrycurrcount{\glsentry}}}}
2988     {}%
2989   }%
2990 }

gls@entry@count Default behaviour is to ignore arguments. Activated by \glsenableentrycount.
2991 \newcommand*{\@gls@entry@count}[2]{}

\cglsc 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.)
2992 \newrobustcmd*{\cglsc}{\gls@hyp@opt@\cglsc}

@cglsc Defined the un-starred form. Need to determine if there is a final optional argument
2993 \newcommand*{\@cglsc}[2][]{%
2994   \new@ifnextchar[{\@cglsc@{#1}{#2}}{\@cglsc@{#1}{#2}[]}}
2995 }

@cglsc@ Read in the final optional argument. This defaults to same behaviour as \gls but issues a
warning.
2996 \def{\@cglsc@#1#2[#3]}{%
2997   \GlossariesWarning{\string\cglsc\space is defaulting to
2998   \string\gls\space since you haven't enabled entry counting}%
2999   \gls@{#1}{#2}[]#3}%
3000 }

\cglscformat Format used by \cglsc if entry only used once on previous run. The first argument is the label,
the second argument is the insert text.

```

```

3001 \newcommand*{\cGlsformat}[2]{%
3002   \ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}#2%
3003 }

\cGls Define command that works like \Gls but behaves differently if the entry count function is
enabled. (If not enabled, it behaves the same as \Gls but issues a warning.)
3004 \newrobustcmd*{\cGls}{\@gls@hyp@opt\@cGls}

\@cGls Defined the un-starred form. Need to determine if there is a final optional argument
3005 \newcommand*{\@cGls}[2][]{%
3006   \new@ifnextchar[{\@cGls@{\#1}{\#2}}{\@cGls@{\#1}{\#2}[]}%
3007 }

\@cGls@ Read in the final optional argument. This defaults to same behaviour as \Gls but issues a
warning.
3008 \def\@cGls@#1#2[#3]{%
3009   \GlossariesWarning{\string\cGls\space is defaulting to
3010   \string\Gls\space since you haven't enabled entry counting}%
3011 \@Gls@{\#1}{\#2}[#3]%
3012 }

\cGlsformat Format used by \cGls if entry only used once on previous run. The first argument is the label,
the second argument is the insert text.
3013 \newcommand*{\cGlsformat}[2]{%
3014   \ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}#2%
3015 }

\cglsp Define command that works like \glspl but behaves differently if the entry count function
is enabled. (If not enabled, it behaves the same as \glspl but issues a warning.)
3016 \newrobustcmd*{\cglsp}{\@gls@hyp@opt\@cglsp}

\@cglsp Defined the un-starred form. Need to determine if there is a final optional argument
3017 \newcommand*{\@cglsp}[2][]{%
3018   \new@ifnextchar[{\@cglsp@{\#1}{\#2}}{\@cglsp@{\#1}{\#2}[]}%
3019 }

\@cglsp@ Read in the final optional argument. This defaults to same behaviour as \glspl but issues a
warning.
3020 \def\@cglsp@#1#2[#3]{%
3021   \GlossariesWarning{\string\cglsp\space is defaulting to
3022   \string\glspl\space since you haven't enabled entry counting}%
3023 \@glspl@{\#1}{\#2}[#3]%
3024 }

\cglspformat Format used by \cglsp if entry only used once on previous run. The first argument is the
label, the second argument is the insert text.
3025 \newcommand*{\cglspformat}[2]{%
3026   \ifglshaslong{#1}{\glsentrylongpl{#1}}{\glsentryfirstplural{#1}}#2%
3027 }

```

```
\cGlspl Define command that works like \Glspl but behaves differently if the entry count function  
is enabled. (If not enabled, it behaves the same as \Glspl but issues a warning.)  
3028 \newrobustcmd*\{cGlspl\}{\gls@hyp@opt@cGlspl}
```

```
\@cGlspl Defined the un-starred form. Need to determine if there is a final optional argument  
3029 \newcommand*\{@cGlspl}[2] [] {  
3030   \new@ifnextchar[\{@cGlspl@{\#1}{\#2}\}{\@cGlspl@{\#1}{\#2}[]}%  
3031 }
```

```
\@cGlspl@ Read in the final optional argument. This defaults to same behaviour as \Glspl but issues a  
warning.  
3032 \def\@cGlspl@#1#2[#3]{%  
3033   \GlossariesWarning{\string\cGlspl\space is defaulting to  
3034     \string\Glspl\space since you haven't enabled entry counting}%  
3035   \@Glspl@{\#1}{\#2}[#3]%
```

```
3036 }  
  
\cGlsplformat Format used by \cGlspl if entry only used once on previous run. The first argument is the  
label, the second argument is the insert text.
```

```
3037 \newcommand*\cGlsplformat[2]{%  
3038   \ifglshaslong{\#1}{\Glsentrylongpl{\#1}}{\Glsentryfirstplural{\#1}}#2%  
3039 }
```

## 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>

```
\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  
3040 \newcommand*\loadglsentries[2][\gls@default]{%  
3041   \let\gls@default\glsdefaulttype  
3042   \def\glsdefaulttype{\#1}\input{\#2}%  
3043   \let\glsdefaulttype\gls@default  
3044 }  
  
\loadglsentries can only be used in the preamble:  
3045 \only\loadglsentries}
```

<sup>1</sup>and any other valid L<sup>A</sup>T<sub>E</sub>X code that can be used in the preamble.

## 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`

3046 `\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

3047 `\newcommand*\{\glsentryfmt\}{%`  
3048   `\@@gls@default@entryfmt\glsdisplayfirst\glsdisplay`  
3049 `}`

Format that provides backwards compatibility:

3050 `\newcommand*\{\@@gls@default@entryfmt\}[2]{%`  
3051   `\ifdefempty\glscustomtext`  
3052   `{%`  
3053    `\glsifplural`  
3054   `{%`

Plural form

3055   `\glscapscase`  
3056   `{%`

Don’t adjust case

3057   `\ifglsused\glslabel`  
3058   `{%`

Subsequent use

3059   `#2{\glsentryplural{\glslabel}}%`  
3060    `\glsentrydescplural{\glslabel}}%`  
3061    `{\glsentrysymbolplural{\glslabel}}{\glsinsert}%`  
3062   `}%`  
3063   `{%`

First use

3064   `#1{\glsentryfirstplural{\glslabel}}%`  
3065    `\glsentrydescplural{\glslabel}}%`  
3066    `{\glsentrysymbolplural{\glslabel}}{\glsinsert}%`  
3067   `}%`  
3068   `{%`  
3069   `{%`

### Make first letter upper case

```
3070     \ifglsused\glslabel
3071     {%
```

Subsequent use. (Expansion was used in version 3.07 and below in case the name wasn't the first thing to be displayed, but now the user can sort out the upper casing in `\defglsentryfmt`, which avoids the issues caused by fragile commands.)

```
3072     \ifbool{glscompatible-3.07}%
3073     {%
3074         \protected@edef@glo@etext{%
3075             #2{\glsentryplural{\glslabel}}%
3076             {\glsentrydescplural{\glslabel}}%
3077             {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
3078         \xmakefirstuc@glo@etext
3079     }%
3080     {%
3081         #2{\Glsentryplural{\glslabel}}%
3082             {\glsentrydescplural{\glslabel}}%
3083             {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
3084     }%
3085     {%
3086     }%
```

### First use

```
3087     \ifbool{glscompatible-3.07}%
3088     {%
3089         \protected@edef@glo@etext{%
3090             #1{\glsentryfirstplural{\glslabel}}%
3091                 {\glsentrydescplural{\glslabel}}%
3092                 {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
3093         \xmakefirstuc@glo@etext
3094     }%
3095     {%
3096         #1{\Glsentryfirstplural{\glslabel}}%
3097             {\glsentrydescplural{\glslabel}}%
3098             {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
3099     }%
3100     {%
3101     }%
3102     {%
```

### Make all upper case

```
3103     \ifglsused\glslabel
3104     {%
```

### Subsequent use

```
3105     \mfirstucMakeUppercase{#2{\glsentryplural{\glslabel}}%
3106         {\glsentrydescplural{\glslabel}}%
3107             {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
3108     }%
3109     {%
```

First use

```
3110      \mfirstucMakeUppercase{#1{\glsentryfirstplural{\glslabel}}}%  
3111          {\glsentrydescplural{\glslabel}}%  
3112          {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%  
3113      }%  
3114  }%  
3115 }%  
3116 {%
```

Singular form

```
3117     \glscapscase  
3118     {%
```

Don't adjust case

```
3119     \ifglsused\glslabel  
3120     {%
```

Subsequent use

```
3121     #2{\glsentrytext{\glslabel}}%  
3122         {\glsentrydesc{\glslabel}}%  
3123         {\glsentrysymbol{\glslabel}}{\glsinsert}}%  
3124     }%  
3125     {%
```

First use

```
3126     #1{\glsentryfirst{\glslabel}}%  
3127         {\glsentrydesc{\glslabel}}%  
3128         {\glsentrysymbol{\glslabel}}{\glsinsert}}%  
3129     }%  
3130     }%  
3131     {%
```

Make first letter upper case

```
3132     \ifglsused\glslabel  
3133     {%
```

Subsequent use

```
3134     \ifbool{glscompatible-3.07}{%  
3135     {%
```

\protected@edef\@glo@etext{%

```
3136     #2{\glsentrytext{\glslabel}}%  
3137         {\glsentrydesc{\glslabel}}%  
3138         {\glsentrysymbol{\glslabel}}{\glsinsert}}%  
3139     \xmakefirstuc\@glo@etext  
3140 }%  
3141 }%  
3142 {%
```

#2{\Glsentrytext{\glslabel}}%  
3143 {\glsentrydesc{\glslabel}}%  
3144 {\glsentrysymbol{\glslabel}}{\glsinsert}}%  
3145 }%  
3146 }%  
3147 }%  
3148 {%

### First use

```
3149      \ifbool{glscompatible-3.07}{%
3150      {%
3151          \protected@edef{\glo@etext}{%
3152              #1{\glsentryfirst{\glslabel}}{%
3153                  {\glsentrydesc{\glslabel}}{%
3154                      {\glsentrysymbol{\glslabel}}{\glsinsert}}{%
3155                          \xmakefirstuc{\glo@etext}%
3156                      }%
3157                  }%
3158                  #1{\Glsentryfirst{\glslabel}}{%
3159                      {\glsentrydesc{\glslabel}}{%
3160                          {\glsentrysymbol{\glslabel}}{\glsinsert}}{%
3161                      }%
3162                  }%
3163              }%
3164          }%
```

### Make all upper case

```
3165      \ifglsused{\glslabel}{%
3166      {%
```

### Subsequent use

```
3167      \mfirstucMakeUppercase{#2{\glsentrytext{\glslabel}}}{%
3168          {\glsentrydesc{\glslabel}}{%
3169              {\glsentrysymbol{\glslabel}}{\glsinsert}}{%
3170          }%
3171      }%
```

### First use

```
3172      \mfirstucMakeUppercase{#1{\glsentryfirst{\glslabel}}}{%
3173          {\glsentrydesc{\glslabel}}{%
3174              {\glsentrysymbol{\glslabel}}{\glsinsert}}{%
3175          }%
3176      }%
3177  }%
3178 }%
3179 {%
```

### Custom text provided in \glsdisp

```
3180  \ifglsused{\glslabel}{%
3181  {%
```

### Subsequent use

```
3182      #2{\glscustomtext}{%
3183          {\glsentrydesc{\glslabel}}{%
3184              {\glsentrysymbol{\glslabel}}{}{%
3185          }%
3186      }%
```

### First use

```
3187      #1{\glscustomtext}{%
```

```

3188      {\glsentrydesc{\glslabel}}%
3189      {\glsentrysymbol{\glslabel}}{}%
3190      }%
3191  }%
3192 }

\glsgenentryfmt Define a generic format that just uses the first, text, plural or first plural keys (or the custom text) with the insert text appended.

3193 \newcommand*\glsgenentryfmt{%
3194   \ifdefempty\glscustomtext
3195   {%
3196     \glsifplural
3197   }%
}

Plural form

3198   \glscapscase
3199   {%

Don't adjust case

3200   \ifglsused\glslabel
3201   {%

Subsequent use

3202   \glsentryplural{\glslabel}\glsinsert
3203   }%
3204   {%

First use

3205   \glsentryfirstplural{\glslabel}\glsinsert
3206   }%
3207   }%
3208   {%

Make first letter upper case

3209   \ifglsused\glslabel
3210   {%

Subsequent use.

3211   \Glsentryplural{\glslabel}\glsinsert
3212   }%
3213   {%

First use

3214   \Glsentryfirstplural{\glslabel}\glsinsert
3215   }%
3216   }%
3217   {%

Make all upper case

3218   \ifglsused\glslabel
3219   {%

```

Subsequent use

```
3220      \mfirstucMakeUppercase
3221          {\glsentryplural{\glslabel}\glsinsert}%
3222      }%
3223      {%
```

First use

```
3224      \mfirstucMakeUppercase
3225          {\glsentryfirstplural{\glslabel}\glsinsert}%
3226      }%
3227      }%
3228      }%
3229      {%
```

Singular form

```
3230      \glscapscase
3231      {%
```

Don't adjust case

```
3232      \ifglsused\glslabel
3233      {%
```

Subsequent use

```
3234      \glsentrytext{\glslabel}\glsinsert
3235      }%
3236      {%
```

First use

```
3237      \glsentryfirst{\glslabel}\glsinsert
3238      }%
3239      }%
3240      {%
```

Make first letter upper case

```
3241      \ifglsused\glslabel
3242      {%
```

Subsequent use

```
3243      \Glsentrytext{\glslabel}\glsinsert
3244      }%
3245      {%
```

First use

```
3246      \Glsentryfirst{\glslabel}\glsinsert
3247      }%
3248      }%
3249      {%
```

Make all upper case

```
3250      \ifglsused\glslabel
3251      {%
```

### Subsequent use

```
3252      \mfirstucMakeUppercase{\glsentrytext{\glslabel}\glsinsert}%
3253      }%
3254      {%
```

### First use

```
3255      \mfirstucMakeUppercase{\glsentryfirst{\glslabel}\glsinsert}%
3256      }%
3257      }%
3258      }%
3259      }%
3260      {%
```

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

```
3261      \glscustomtext\glsinsert
3262      }%
3263 }
```

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

```
3264 \newcommand*\glsgenacfmt}{%
3265 \ifdefempty\glscustomtext
3266 {%
3267 \ifglsused\glslabel
3268 {%
```

### Subsequent use:

```
3269      \glsifplural
3270      {%
```

### Subsequent plural form:

```
3271      \glscapscase
3272      {%
```

### Subsequent plural form, don't adjust case:

```
3273      \acronymfont{\glsentryshortpl{\glslabel}}\glsinsert
3274      }%
3275      {%
```

### Subsequent plural form, make first letter upper case:

```
3276      \acronymfont{\Glsentryshortpl{\glslabel}}\glsinsert
3277      }%
3278      {%
```

### Subsequent plural form, all caps:

```
3279      \mfirstucMakeUppercase
3280      {\acronymfont{\glsentryshortpl{\glslabel}}\glsinsert}%
3281      }%
3282      }%
3283      {%
```

Subsequent singular form

```
3284      \glscapscase
3285      {%
```

Subsequent singular form, don't adjust case:

```
3286      \acronymfont{\glsentryshort{\glslabel}}\glsinsert
3287      }%
3288      {%
```

Subsequent singular form, make first letter upper case:

```
3289      \acronymfont{\Glsentryshort{\glslabel}}\glsinsert
3290      }%
3291      {%
```

Subsequent singular form, all caps:

```
3292      \mfirstucMakeUppercase
3293      {\acronymfont{\glsentryshort{\glslabel}}\glsinsert}%
3294      }%
3295      }%
3296      }%
3297      {%
```

First use:

```
3298      \glsifplural
3299      {%
```

First use plural form:

```
3300      \glscapscase
3301      {%
```

First use plural form, don't adjust case:

```
3302      \genplacrfullformat{\glslabel}{\glsinsert}%
3303      }%
3304      {%
```

First use plural form, make first letter upper case:

```
3305      \Genplacrfullformat{\glslabel}{\glsinsert}%
3306      }%
3307      {%
```

First use plural form, all caps:

```
3308      \mfirstucMakeUppercase
3309      {\genplacrfullformat{\glslabel}{\glsinsert}}%
3310      }%
3311      }%
3312      {%
```

First use singular form

```
3313      \glscapscase
3314      {%
```

First use singular form, don't adjust case:

```
3315      \genacrfullformat{\glslabel}{\glsinsert}%
```

```
3316      }%
3317      {%
```

First use singular form, make first letter upper case:

```
3318      \Genacrfullformat{\glslabel}{\glsinsert}%
3319      }%
3320      {%
```

First use singular form, all caps:

```
3321      \mfirstucMakeUppercase
3322      {\genacrfullformat{\glslabel}{\glsinsert}}%
3323      }%
3324      }%
3325      }%
3326      }%
3327      {%
```

User supplied text.

```
3328      \glscustomtext
3329      }%
3330 }
```

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

The full format used by \glsgenacfmt (singular).

```
3331 \newcommand*{\genacrfullformat}[2]{%
3332   \glsentrylong{\#1}\#2\space
3333   (\protect\firstracronymfont{\glsentryshort{\#1}})%
3334 }
```

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

As above but makes the first letter upper case.

```
3335 \newcommand*{\Genacrfullformat}[2]{%
3336   \protected@edef\gls@text{\genacrfullformat{\#1}{\#2}}%
3337   \xmakefirstuc\gls@text
3338 }
```

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

The full format used by \glsgenacfmt (plural).

```
3339 \newcommand*{\genplacrfullformat}[2]{%
3340   \glsentrylongpl{\#1}\#2\space
3341   (\protect\firstracronymfont{\glsentryshortpl{\#1}})%
3342 }
```

```
\Genplacrfullformat{\label}{\insert}
```

As above but makes the first letter upper case.

```
3343 \newcommand*{\Genplacrfullformat}[2]{%
3344   \protected@edef\gls@text{\genplacrfullformat{\#1}{\#2}}%
3345   \xmakefirstuc\gls@text
3346 }
```

`glsdisplayfirst` Deprecated. Kept for backward compatibility.

```
3347 \newcommand*{\glsdisplayfirst}[4]{#1#4}
```

`\glsdisplay` Deprecated. Kept for backward compatibility.

```
3348 \newcommand*{\glsdisplay}[4]{#1#4}
```

`\defglsdisplay` Deprecated. Kept for backward compatibility.

```
3349 \newcommand*{\defglsdisplay}[2][\glsdefaulttype]{%
3350   \GlossariesWarning{\string\defglsdisplay\space is now obsolete.\^J
3351   Use \string\defglsentryfmt\space instead}%
3352   \expandafter\def\csname gls@\#1@display\endcsname##1##2##3##4{\#2}%
3353   \edef@\gls@doentrydef{%
3354     \noexpand\defglsentryfmt[\#1]{%
3355       \noexpand\ifcsdef{gls@\#1@displayfirst}{%
3356         {%
3357           \noexpand\@@gls@default@entryfmt
3358           {\noexpand\csuse{gls@\#1@displayfirst}}%
3359           {\noexpand\csuse{gls@\#1@display}}%
3360         }%
3361         {%
3362           \noexpand\@@gls@default@entryfmt
3363             {\noexpand\glsdisplayfirst}%
3364             {\noexpand\csuse{gls@\#1@display}}%
3365           }%
3366         }%
3367       }%
3368     \gls@doentrydef
3369   }}
```

`glsdisplayfirst` Deprecated. Kept for backward compatibility.

```
3370 \newcommand*{\defglsdisplayfirst}[2][\glsdefaulttype]{%
3371   \GlossariesWarning{\string\defglsdisplayfirst\space is now obsolete.\^J
3372   Use \string\defglsentryfmt\space instead}%
3373   \expandafter\def\csname gls@\#1@displayfirst\endcsname##1##2##3##4{\#2}%
3374   \edef@\gls@doentrydef{%
3375     \noexpand\defglsentryfmt[\#1]{%
3376       \noexpand\ifcsdef{gls@\#1@display}{%
3377         {%
3378           \noexpand\@@gls@default@entryfmt
3379             {\noexpand\csuse{gls@\#1@displayfirst}}%
```

```

3380      {\noexpand\csuse{gls@#1@display}}%
3381  }%
3382  {%
3383      \noexpand\@gls@default@entryfmt
3384          {\noexpand\csuse{gls@#1@displayfirst}}%
3385          {\noexpand\glsdisplay}%
3386      }%
3387  }%
3388 }%
3389 \gls@doentrydef
3390 }

```

## 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 L<sup>A</sup>T<sub>E</sub>X 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.

```

3391 \define@key{glslink}{counter}{%
3392   \ifcsundef{c@#1}{%
3393     {%
3394       \PackageError{glossaries}{%
3395         {There is no counter called ‘#1’}}%
3396     {%
3397       The counter key should have the name of a valid counter
3398       as its value}%
3399   }%
3400 }%
3401 {%
3402   \def\gls@counter{#1}%
3403 }%
3404 }

```

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.

```

3405 \define@key{glslink}{format}{%
3406   \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.

```
3407 \define@boolkey{glslink}{hyper}[true]{}
```

Initialise hyper key.

```
3408 \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.

```
3409 \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.

```
3410 \newcommand*{\glslinkvar}[3]{#1}
```

\glsifhyper Now deprecated.

```
3411 \newcommand*{\glsifhyper}[2]{%
3412   \glslinkvar{#1}{#2}{#1}%
3413   \GlossariesWarning{\string\glsifhyper\space is deprecated. Did%
3414   you mean \string\glsifhyperon\space or \string\glslinkvar?}%
3415 }
```

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

```
3416 \newcommand*{\@gls@hyp@opt}[1]{%
3417   \let\glslinkvar\@firstofthree
3418   \let\@gls@hyp@opt@cs\relax
3419   \@ifstar{\s@gls@hyp@opt}%
3420   {\@ifnextchar+\@firstoftwo{\p@gls@hyp@opt}{#1}}%
3421 }
```

\s@gls@hyp@opt Starred version

```
3422 \newcommand*{\s@gls@hyp@opt}[1][]{%
3423   \let\glslinkvar\@secondofthree
3424   \@gls@hyp@opt@cs[hyper=false,#1]}
```

\p@gls@hyp@opt Plus version

```
3425 \newcommand*{\p@gls@hyp@opt}[1][]{%
3426   \let\glslinkvar\@thirdofthree
3427   \@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
3428 \newrobustcmd*\glslink{%
3429   \@gls@hyp@opt\@gls@link
3430 }
```

`\@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.

```
3431 \newcommand*\@gls@link[3][]{%
3432   \glsdoifexistsor{\#2}{%
3433     {%
3434       \let\do@gls@link@checkfirsthyper\relax
3435       \gls@link[\#1]{\#2}{\#3}%
3436     }{}}
```

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

```
3437   \glstextformat{\#3}%
3438 }
```

  

```
3439 \glspostlinkhook
3440 }
```

`glspostlinkhook`

```
3441 \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`.

```
3442 \newcommand*\@gls@link@checkfirsthyper{%
3443   \ifglsused{\glslabel}{%
3444     {%
3445   }{%
3446     {}}
```

```

3447 \gls@checkisacronymlist\glstype
3448 \ifglshyperfirst
3449   \if@glsisacronymlist
3450     \ifglsacrfootnote
3451       \KV@glslink@hyperfalse
3452     \fi
3453   \fi
3454 \else
3455   \KV@glslink@hyperfalse
3456 \fi
3457 }%

```

Allow user to hook into this

```

3458 \glslinkcheckfirsthyperhook
3459 }

```

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

```

3460 \newcommand*\glslinkcheckfirsthyperhook{}}

```

`linkpostsetkeys`

```

3461 \newcommand*\glslinkpostsetkeys{}}

```

`\glsifhyperon` Check the value of the hyper key:

```

3462 \newcommand{\glsifhyperon}[2]{\ifKV@glslink@hyper#1\else#2\fi}

```

`ablehyperinlist` Disable hyperlink if in the “nohyper” list.

```

3463 \newcommand*\do@glsdisablehyperinlist{}%
3464   \expandafter\DTLifinlist\expandafter{\glstype}{\@gls@nohyperlist}%
3465   {\KV@glslink@hyperfalse}{}%
3466 }

```

`\lt@glslink@opts` Hook to set default options for `\@glslink`.

```

3467 \newcommand*\@gls@setdefault@glslink@opts{}}

```

`\@gls@link`

```

3468 \def\@gls@link[#1]#2#3{%

```

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

```

3469   \leavevmode
3470   \edef\glslabel{\glsdetoklabel{#2}}%

```

Save options in `\@gls@link@opts` and label in `\@gls@link@label`

```

3471   \def\@gls@link@opts{#1}%
3472   \let\@gls@link@label\glslabel
3473   \def\@glsnumberformat{\glsnumberformat}%
3474   \edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%

```

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

```

3475   \edef\glstype{\csname glo@\glslabel @type\endcsname}%

```

Save original setting

3476 \let\org@ifKV@glslink@hyper\ifKV@glslink@hyper

Set defaults:

3477 \gls@setdefault@glslink@opts

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

3478 \do@glsdisablehyperinlist

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.

3479 \do@gls@link@checkfirsthyper

3480 \setkeys{glslink}{#1}%

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

3481 \glslinkpostsetkeys

Store the entry's counter in \the\glsentrycounter

3482 \gls@saveentrycounter

Define sort key if necessary:

3483 \gls@setsort{\glslabel}%

(De-tok'ing done by \do@wrglossary)

3484 \do@wrglossary{#2}%

3485 \ifKV@glslink@hyper

3486 \glslink{\glolinkprefix\glslabel}{\glstextformat{#3}}%

3487 \else

3488 \glsdonohyperlink{\glolinkprefix\glslabel}{\glstextformat{#3}}%

3489 \fi

Restore original setting

3490 \let\ifKV@glslink@hyper\org@ifKV@glslink@hyper

3491 }

\glolinkprefix

3492 \newcommand\*{\glolinkprefix}[1]

\glsentrycounter Set default value of entry counter

3493 \def\glsentrycounter{\glscounter}%

\aveentrycounter Need to check if using equation counter in align environment:

3494 \newcommand\*{\gls@saveentrycounter}{%

3495 \def\gls@Hcounter{}%

Are we using equation counter?

3496 \ifthenelse{\equal{\gls@counter}{equation}}%

3497 {

If we're in align environment, `\xatlevel@` will be defined. (Can't test for `\@currenvir` as may be inside an inner environment.)

```
3498 \ifcsundef{xatlevel@}%
3499 {%
3500   \edef\the\glsglsentrycounter{\expandafter\noexpand
3501     \csname the\@glsglsentrycounter\endcsname}%
3502 }%
3503 {%
3504   \ifx\xatlevel@\empty
3505     \edef\the\glsglsentrycounter{\expandafter\noexpand
3506       \csname the\@glsglsentrycounter\endcsname}%
3507   \else
3508     \savecounters@
3509     \advance\c@equation by 1\relax
3510     \edef\the\glsglsentrycounter{\csname the\@glsglsentrycounter\endcsname}%

```

Check if hyperref version of this counter

```
3511 \ifcsundef{theH\@glsglsentrycounter}%
3512 {%
3513   \def\@glsglsentrycounter{\the\glsglsentrycounter}%
3514 }%
3515 {%
3516   \def\@glsglsentrycounter{\csname theH\@glsglsentrycounter\endcsname}%
3517 }%
3518 \protected@edef\theH\glsglsentrycounter{\@glsglsentrycounter}%
3519 \restorecounters@
3520 \fi
3521 }%
3522 }%
3523 {%
```

Not using equation counter so no special measures:

```
3524 \edef\the\glsglsentrycounter{\expandafter\noexpand
3525   \csname the\@glsglsentrycounter\endcsname}%
3526 }%
```

Check if hyperref version of this counter

```
3527 \ifx\@glsglsentrycounter\empty
3528   \ifcsundef{theH\@glsglsentrycounter}%
3529   {%
3530     \def\theH\glsglsentrycounter{\the\glsglsentrycounter}%
3531   }%
3532   {%
3533     \protected@edef\theH\glsglsentrycounter{\expandafter\noexpand
3534       \csname theH\@glsglsentrycounter\endcsname}%
3535   }%
3536 \fi
3537 }
```

`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.

```
3538 \def\@set@glo@numformat#1#2#3#4{%
3539   \expandafter\@glo@check@mkidxrangechar#3\@nil
3540   \protected@edef#1{%
3541     \@glo@prefix setentrycounter[#4]{#2}%
3542     \expandafter\string\csname\@glo@suffix\endcsname
3543   }%
3544   \@gls@checkmkidxchars#1%
3545 }
```

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.

```
3546 \def\@glo@check@mkidxrangechar#1#2\@nil{%
3547 \if#1(\relax
3548   \def\@glo@prefix{()%
3549   \if\relax#2\relax
3550     \def\@glo@suffix{glsnumberformat}%
3551   \else
3552     \def\@glo@suffix{#2}%
3553   \fi
3554 \else
3555   \if#1)\relax
3556     \def\@glo@prefix{}%
3557     \if\relax#2\relax
3558       \def\@glo@suffix{glsnumberformat}%
3559     \else
3560       \def\@glo@suffix{#2}%
3561     \fi
3562   \else
3563     \def\@glo@prefix{}\def\@glo@suffix{#1#2}%
3564   \fi
3565 \fi}
```

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

```
3566 \newcommand*{\@gls@escbsdq}[1]{%
3567   \def\@gls@checkedmkidx{}%
3568   \let\gls@xdystring=\relax
3569   \onelevel@sanitize\gls@xdystring
3570   \edef\do@gls@xdycheckbackslash{%
3571     \noexpand\@gls@xdycheckbackslash\gls@xdystring\noexpand\@nil
3572     \@backslashchar\@backslashchar\noexpand\@null}%
3573   \do@gls@xdycheckbackslash
3574   \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdystring}%
3575   \def\@gls@checkedmkidx{}%
```

```

3576 \expandafter\@gls@xdycheckquote\gls@xdystring\@nil""\null
3577 \expandafter\@gls@updatechecked\@gls@checkedmidx{\gls@xdystring}%
  Unsanitize \gls@numberpage, \gls@alphpage, \gls@Alphpage and \glsromanpage (thanks
  to David Carlise for the suggestion.)
3578 \@for\@gls@tmp:=\gls@protected@pagefmts\do
3579 {%
3580   \edef\@gls@sanitized@tmp{\expandafter\gobble\string\\expandonce\@gls@tmp}%
3581   \onelevel@sanitize\@gls@sanitized@tmp
3582   \edef\gls@dosubst{%
3583     \noexpand\DTLsubstituteall\noexpand\gls@xdystring
3584     {\@gls@sanitized@tmp}{\expandonce\@gls@tmp}%
3585   }%
3586   \gls@dosubst
3587 }%

```

Assign to required control sequence

```

3588 \let#1=\gls@xdystring
3589 }%

```

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

#### checkmidxchars

```

3590 \newcommand{\@gls@checkmidxchars}[1]{%
3591   \ifglsxindy
3592     \@gls@escbsdq{#1}%
3593   \else
3594     \def\@gls@checkedmidx{}%
3595     \expandafter\@gls@checkquote#1\@nil""\null
3596     \expandafter\@gls@updatechecked\@gls@checkedmidx{#1}%
3597     \def\@gls@checkedmidx{}%
3598     \expandafter\@gls@checkescquote#1\@nil\""\null
3599     \expandafter\@gls@updatechecked\@gls@checkedmidx{#1}%
3600     \def\@gls@checkedmidx{}%
3601     \expandafter\@gls@checkescactual#1\@nil\?\?\null
3602     \expandafter\@gls@updatechecked\@gls@checkedmidx{#1}%
3603     \def\@gls@checkedmidx{}%
3604     \expandafter\@gls@checkactual#1\@nil??\null
3605     \expandafter\@gls@updatechecked\@gls@checkedmidx{#1}%
3606     \def\@gls@checkedmidx{}%
3607     \expandafter\@gls@checkbar#1\@nil||\null
3608     \expandafter\@gls@updatechecked\@gls@checkedmidx{#1}%
3609     \def\@gls@checkedmidx{}%
3610     \expandafter\@gls@checkescbar#1\@nil\\|\|\null
3611     \expandafter\@gls@updatechecked\@gls@checkedmidx{#1}%
3612     \def\@gls@checkedmidx{}%
3613     \expandafter\@gls@checklevel#1\@nil!!\null
3614     \expandafter\@gls@updatechecked\@gls@checkedmidx{#1}%
3615   \fi
3616 }%

```

Update the control sequence and strip trailing \@nil:

s@updatechecked

3617 \def\@gls@updatechecked#1\@nil#2{\def#2{#1}}

\@gls@tmpb Define temporary token

3618 \newtoks\@gls@tmpb

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

3619 \def\@gls@checkquote#1"#2"#3\null{%

3620 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%

3621 \toks@={#1}%

3622 \ifx\null#2\null

3623 \ifx\null#3\null

3624 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%

3625 \def\@gls@checkquote{\relax}%

3626 \else

3627 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@

3628 \@gls@quotechar\@gls@quotechar\@gls@quotechar\@gls@quotechar}%

3629 \def\@gls@checkquote{\@gls@checkquote#3\null}%

3630 \fi

3631 \else

3632 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@

3633 \@gls@quotechar\@gls@quotechar}%

3634 \ifx\null#3\null

3635 \def\@gls@checkquote{\@gls@checkquote#2""\null}%

3636 \else

3637 \def\@gls@checkquote{\@gls@checkquote#2"#3\null}%

3638 \fi

3639 \fi

3640 \@@gls@checkquote

3641 }

s@checkescquote Do the same for \":

3642 \def\@gls@checkescquote#1"#2"#3\null{%

3643 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%

3644 \toks@={#1}%

3645 \ifx\null#2\null

3646 \ifx\null#3\null

3647 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%

3648 \def\@gls@checkescquote{\relax}%

3649 \else

3650 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@

3651 \@gls@quotechar\string\"@\gls@quotechar

3652 \@gls@quotechar\string\"@\gls@quotechar}%

3653 \def\@gls@checkescquote{\@gls@checkescquote#3\null}%

3654 \fi

3655 \else

3656 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@

```

3657     \@gls@quotechar\string\"@\gls@quotechar}%
3658     \ifx\null#3\null
3659         \def@\gls@checkescquote{\gls@checkescquote#2\""\null}%
3660     \else
3661         \def@\gls@checkescquote{\gls@checkescquote#2"#3\null}%
3662     \fi
3663 \fi
3664 \@@gls@checkescquote
3665 }

```

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

```

3666 \def@\gls@checkescactual#1\?#2\?#3\null{%
3667   @gls@tmpb=\expandafter{\gls@checkedmkidx}%
3668   \toks@={#1}%
3669   \ifx\null#2\null
3670     \ifx\null#3\null
3671       \edef@\gls@checkedmkidx{\the@gls@tmpb\the\toks@}%
3672       \def@\gls@checkescactual{\relax}%
3673     \else
3674       \edef@\gls@checkedmkidx{\the@gls@tmpb\the\toks@%
3675       \@gls@quotechar\string\"@\gls@actualchar%
3676       \@gls@quotechar\string\"@\gls@actualchar}%
3677       \def@\gls@checkescactual{\gls@checkescactual#3\null}%
3678     \fi
3679   \else
3680     \edef@\gls@checkedmkidx{\the@gls@tmpb\the\toks@%
3681     \@gls@quotechar\string\"@\gls@actualchar}%
3682     \ifx\null#3\null
3683       \def@\gls@checkescactual{\gls@checkescactual#2\??\?\\null}%
3684     \else
3685       \def@\gls@checkescactual{\gls@checkescactual#2\?#3\null}%
3686     \fi
3687   \fi
3688 \@@gls@checkescactual
3689 }

```

gls@checkescbar Similarly for \|:

```

3690 \def@\gls@checkescbar#1\|#2\|#3\null{%
3691   @gls@tmpb=\expandafter{\gls@checkedmkidx}%
3692   \toks@={#1}%
3693   \ifx\null#2\null
3694     \ifx\null#3\null
3695       \edef@\gls@checkedmkidx{\the@gls@tmpb\the\toks@}%
3696       \def@\gls@checkescbar{\relax}%
3697     \else
3698       \edef@\gls@checkedmkidx{\the@gls@tmpb\the\toks@%
3699       \@gls@quotechar\string\"@\gls@encapchar%
3700       \@gls@quotechar\string\"@\gls@encapchar}%
3701       \def@\gls@checkescbar{\gls@checkescbar#3\null}%

```

```

3702     \fi
3703 \else
3704   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3705     \@gls@quotechar\string\"@\gls@encapchar}%
3706   \ifx\null#3\null
3707     \def\@@gls@checkescbar{\@gls@checkescbar#2\|\|\null}%
3708   \else
3709     \def\@@gls@checkescbar{\@gls@checkescbar#2\|#3\null}%
3710   \fi
3711 \fi
3712 \@@gls@checkescbar
3713 }

```

s@checkesclevel Similarly for \!:

```

3714 \def\@gls@checkesclevel#1\!#2\!#3\null{%
3715   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3716   \toks@={#1}%
3717   \ifx\null#2\null
3718     \ifx\null#3\null
3719       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3720       \def\@@gls@checkesclevel{\relax}%
3721     \else
3722       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3723         \@gls@quotechar\string\"@\gls@levelchar
3724         \@gls@quotechar\string\"@\gls@levelchar}%
3725       \def\@@gls@checkesclevel{\@gls@checkesclevel#3\null}%
3726     \fi
3727   \else
3728     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3729       \@gls@quotechar\string\"@\gls@levelchar}%
3730     \ifx\null#3\null
3731       \def\@@gls@checkesclevel{\@gls@checkesclevel#2\!!\!#3\null}%
3732     \else
3733       \def\@@gls@checkesclevel{\@gls@checkesclevel#2\!#3\null}%
3734     \fi
3735   \fi
3736 \@@gls@checkesclevel
3737 }

```

\@gls@checkbar and for |:

```

3738 \def\@gls@checkbar#1|#2|#3\null{%
3739   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3740   \toks@={#1}%
3741   \ifx\null#2\null
3742     \ifx\null#3\null
3743       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3744       \def\@@gls@checkbar{\relax}%
3745     \else
3746       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}

```

```

3747     \@gls@quotechar \@gls@encapchar \@gls@quotechar \@gls@encapchar}%
3748     \def\@@gls@checkbar{\@gls@checkbar#3\null}%
3749     \fi
3750 \else
3751     \edef\@gls@checkedmidx{\the\@gls@tmpb\the\toks@
3752         \@gls@quotechar \@gls@encapchar}%
3753     \ifx\null#3\null
3754         \def\@@gls@checkbar{\@gls@checkbar#2|\| \null}%
3755     \else
3756         \def\@@gls@checkbar{\@gls@checkbar#2|\#3\null}%
3757     \fi
3758 \fi
3759 \@@gls@checkbar
3760 }

```

@gls@checklevel and for !:

```

3761 \def\@gls@checklevel#1!#2!#3\null{%
3762     \@gls@tmpb=\expandafter{\@gls@checkedmidx}%
3763     \toks@={#1}%
3764     \ifx\null#2\null
3765         \ifx\null#3\null
3766             \edef\@gls@checkedmidx{\the\@gls@tmpb\the\toks@}%
3767             \def\@@gls@checklevel{\relax}%
3768         \else
3769             \edef\@gls@checkedmidx{\the\@gls@tmpb\the\toks@
3770                 \@gls@quotechar \@gls@levelchar \@gls@quotechar \@gls@levelchar}%
3771             \def\@@gls@checklevel{\@gls@checklevel#3\null}%
3772         \fi
3773     \else
3774         \edef\@gls@checkedmidx{\the\@gls@tmpb\the\toks@
3775             \@gls@quotechar \@gls@levelchar}%
3776         \ifx\null#3\null
3777             \def\@@gls@checklevel{\@gls@checklevel#2!!\null}%
3778         \else
3779             \def\@@gls@checklevel{\@gls@checklevel#2!#3\null}%
3780         \fi
3781     \fi
3782 \@@gls@checklevel
3783 }

```

gls@checkactual and for ?:

```

3784 \def\@gls@checkactual#1?#2?#3\null{%
3785     \@gls@tmpb=\expandafter{\@gls@checkedmidx}%
3786     \toks@={#1}%
3787     \ifx\null#2\null
3788         \ifx\null#3\null
3789             \edef\@gls@checkedmidx{\the\@gls@tmpb\the\toks@}%
3790             \def\@@gls@checkactual{\relax}%
3791         \else

```

```

3792     \edef\@gls@checkedmidx{\the\@gls@tmpb\the\toks@%
3793         \@gls@quotechar\@gls@actualchar\@gls@quotechar\@gls@actualchar}%
3794     \def\@gls@checkactual{\@gls@checkactual#3\null}%
3795     \fi
3796   \else
3797     \edef\@gls@checkedmidx{\the\@gls@tmpb\the\toks@%
3798         \@gls@quotechar\@gls@actualchar}%
3799     \ifx\null#3\null
3800       \def\@gls@checkactual{\@gls@checkactual#2??\null}%
3801     \else
3802       \def\@gls@checkactual{\@gls@checkactual#2#3\null}%
3803     \fi
3804   \fi
3805 \@@gls@checkactual
3806 }

```

s@xdycheckquote As before but for use with xindy

```

3807 \def\@gls@xdycheckquote#1"#2"#3\null{%
3808   \@gls@tmpb=\expandafter{\@gls@checkedmidx}%
3809   \toks@={#1}%
3810   \ifx\null#2\null
3811     \ifx\null#3\null
3812       \edef\@gls@checkedmidx{\the\@gls@tmpb\the\toks@}%
3813       \def\@gls@xdycheckquote{\relax}%
3814     \else
3815       \edef\@gls@checkedmidx{\the\@gls@tmpb\the\toks@%
3816           \string\"}\string"}%
3817       \def\@gls@xdycheckquote{\@gls@xdycheckquote#3\null}%
3818     \fi
3819   \else
3820     \edef\@gls@checkedmidx{\the\@gls@tmpb\the\toks@%
3821           \string"}%
3822     \ifx\null#3\null
3823       \def\@gls@xdycheckquote{\@gls@xdycheckquote#2""\null}%
3824     \else
3825       \def\@gls@xdycheckquote{\@gls@xdycheckquote#2#3\null}%
3826     \fi
3827   \fi
3828 \@@gls@xdycheckquote
3829 }

```

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

```

3830 \edef\def@gls@xdycheckbackslash{%
3831   \noexpand\def\noexpand\@gls@xdycheckbackslash##1\@backslashchar
3832   ##2\@backslashchar##3\noexpand\null{%
3833   \noexpand\@gls@tmpb=\noexpand\expandafter
3834   {\noexpand\@gls@checkedmidx}%
3835   \noexpand\toks@={##1}%
3836   \noexpand\ifx\noexpand\null##2\noexpand\null

```

```

3837 \noexpand\ifx\noexpand\null##3\noexpand\null
3838   \noexpand\edef\noexpand\@gls@checkedmkidx{%
3839     \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
3840   \noexpand\def\noexpand\@@gls@xdycheckbackslash{\relax}%
3841 \noexpand\else
3842   \noexpand\edef\noexpand\@gls@checkedmkidx{%
3843     \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@%
3844     \@backslashchar@backslashchar@backslashchar@backslashchar}%
3845 \noexpand\def\noexpand\@@gls@xdycheckbackslash{%
3846   \noexpand\@gls@xdycheckbackslash##3\noexpand\null}%
3847 \noexpand\fi
3848 \noexpand\else
3849   \noexpand\edef\noexpand\@gls@checkedmkidx{%
3850     \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@%
3851     \@backslashchar@backslashchar}%
3852 \noexpand\ifx\noexpand\null##3\noexpand\null
3853   \noexpand\def\noexpand\@@gls@xdycheckbackslash{%
3854     \noexpand\@gls@xdycheckbackslash##2\@backslashchar%
3855     \@backslashchar\noexpand\null}%
3856 \noexpand\else
3857   \noexpand\def\noexpand\@@gls@xdycheckbackslash{%
3858     \noexpand\@gls@xdycheckbackslash##2\@backslashchar%
3859     ##3\noexpand\null}%
3860 \noexpand\fi
3861 \noexpand\fi
3862 \noexpand\@@gls@xdycheckbackslash
3863 }%
3864 }

```

Now go ahead and define \@gls@xdycheckbackslash

```
3865 \def@gls@xdycheckbackslash
```

lsdohypertarget

```

3866 \newlength\gls@tmpalen
3867 \newcommand*\glsdohypertarget[2]{%
3868   \glsshowtarget{#1}%
3869   \settoheight{\gls@tmpalen}{#2}%
3870   \raisebox{\gls@tmpalen}{\hypertarget{#1}{}}#2%
3871 }

```

\glsdohyperlink

```

3872 \newcommand*\glsdohyperlink[2]{%
3873   \glsshowtarget{#1}%
3874   \hyperlink{#1}{#2}%
3875 }

```

lsdonohyperlink

```
3876 \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.

```
3877 \ifcsundef{hyperlink}{%
3878 {%
3879   \let\@glslink\glsdonohyperlink
3880 }%
3881 {%
3882   \let\@glslink\glsdohyperlink
3883 }
```

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

```
3884 \ifcsundef{hypertarget}{%
3885 {%
3886   \let\@glstarget\@secondoftwo
3887 }%
3888 {%
3889   \let\@glstarget\glsdohypertarget
3890 }
```

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

\glsdisablehyper

```
3891 \newcommand{\glsdisablehyper}{%
3892   \KV@glslink@hyperfalse
3893   \let\@glslink\glsdonohyperlink
3894   \let\@glstarget\@secondoftwo
3895 }
```

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

\glsenablehyper

```
3896 \newcommand{\glsenablehyper}{%
3897   \KV@glslink@hypertrue
3898   \let\@glslink\glsdohyperlink
3899   \let\@glstarget\glsdohypertarget
3900 }
```

Provide some convenience commands if not already defined:

```
3901 \providecommand{\@firstofthree}[3]{#1}
3902 \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
3903 \newrobustcmd*{\gls}{\@gls@hyp@opt@gls}
```

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

```
\@gls
3904 \newcommand*{\@gls}[2][]{%
3905   \new@ifnextchar[\{\@gls@{\#1}{\#2}\}{\@gls@{\#1}{\#2}[]}%
3906 }
```

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

```
3907 \def \@gls@#1#2[#3]{%
3908   \glsdoifexists{#2}%
3909   {%
3910     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
3911     \let\glsifplural\@secondoftwo
3912     \let\glscapscase\@firstofthree
3913     \let\glscustomtext\@empty
3914     \def\glsinsert{#3}%

```

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

```
3915   \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.

```
3916   \@gls@link[#1]{#2}{\@glo@text}%

```

Indicate that this entry has now been used

```
3917   \ifKV@glslink@local
3918     \glslocalunset{#2}%
3919   \else
3920     \glsunset{#2}%
3921   \fi
3922 }%
3923 \glspostlinkhook
3924 }
```

\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
3925 \newrobustcmd*\{\Gls\}{\@gls@hyp@opt\@Gls}
```

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

```
3926 \newcommand*{\@Gls}[2][]{%
3927   \new@ifnextchar[{\@Gls@{\#1}{\#2}}{\@Gls@{\#1}{\#2}[]}}%
3928 }
```

\@Gls@ Read in the final optional argument:

```
3929 \def\@Gls@#1#2[#3]{%
3930   \glsdoifexists{#2}%
3931   {%
3932     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
3933     \let\glsifplural\@secondoftwo
3934     \let\glscapscase\@secondofthree
3935     \let\glscustomtext\@empty
3936     \def\glsinsert{#3}%
3937 }
```

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

```
3937 \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.

```
3938 \gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3939 \ifKV@glslink@local
3940   \glslocalunset{#2}%
3941 \else
3942   \glsunset{#2}%
3943 \fi
3944 }%
3945 \glspostlinkhook
3946 }
```

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

```
\GLS
3947 \newrobustcmd*\{\GLS\}{\@gls@hyp@opt\@GLS}
```

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

```
3948 \newcommand*{\@GLS}[2][]{%
3949   \new@ifnextchar[{\@GLS@{\#1}{\#2}}{\@GLS@{\#1}{\#2}[]}}%
3950 }
```

\@GLS@ Read in the final optional argument:

```
3951 \def\@GLS@#1#2[#3]{%
3952   \glsdoifexists{#2}%
3953   {%
3954     \let\do@gls@link@checkfirsthyper\gls@link@checkfirsthyper
3955     \let\glsifplural\@secondoftwo
3956     \let\glscapscase\@thirdofthree
3957     \let\glscustomtext\@empty
3958     \def\glsinsert{#3}%
3959   }
```

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

```
3959   \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%
3960 }
```

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.

```
3960   \@gls@link[#1]{#2}{\@glo@text}%
3961 }
```

Indicate that this entry has now been used

```
3961   \ifKV@glslink@local
3962     \glslocalunset{#2}%
3963   \else
3964     \glsunset{#2}%
3965   \fi
3966 }%
3967 \glspostlinkhook
3968 }
```

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

\glspl

```
3969 \newrobustcmd*\glspl{\gls@hyp@opt\glspl}
```

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

```
3970 \newcommand*\glspl[2][]{%
3971   \new@ifnextchar[\glspl@#1]{\glspl@#1[]}{\glspl@#1[]}%
3972 }
```

\@glspl@ Read in the final optional argument:

```
3973 \def\@glspl@#1#2[#3]{%
3974   \glsdoifexists{#2}%
3975   {%
3976     \let\do@gls@link@checkfirsthyper\gls@link@checkfirsthyper
3977     \let\glsifplural\@firstoftwo
3978     \let\glscapscase\@firstofthree
3979     \let\glscustomtext\@empty
3980     \def\glsinsert{#3}%
3981   }
```

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

```
3981 \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.

```
3982 \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3983 \ifKV@glslink@local
3984   \glslocalunset{#2}%
3985 \else
3986   \glsunset{#2}%
3987 \fi
3988 }%
```

```
3989 \glspostlinkhook
3990 }
```

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

`\Glspl`

```
3991 \newrobustcmd*\Glspl{\@gls@hyp@opt\Glspl}
```

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

```
3992 \newcommand*\Glspl[2][]{%
3993   \new@ifnextchar[\Glspl@{#1}{#2}]{\Glspl@{#1}{#2}[]}{%
3994 }
```

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

```
3995 \def\@Glspl@#1#2[#3]{%
3996   \glsdoifexists{#2}%
3997   {%
3998     \let\do@gls@link@checkfirsthyper\gls@link@checkfirsthyper
3999     \let\glsifplural\firstoftwo
4000     \let\glscapscase\secondofthree
4001     \let\glscustomtext\empty
4002     \def\glsinsert{#3}%

```

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

```
4003 \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.

```
4004 \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
4005 \ifKV@glslink@local
4006   \glslocalunset{#2}%
4007 \else
4008   \glsunset{#2}%
4009 \fi
4010 }%
4011 \glspostlinkhook
4012 }
```

\GLSp1 behaves like \glspl except that all the link text is converted to uppercase.

\GLSp1

```
4013 \newrobustcmd*\{\GLSp1\}{\gls@hyp@opt\GLSp1}
```

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

```
4014 \newcommand*\{\GLSp1}[2][]{%
4015   \new@ifnextchar[\{\GLSp1@{#1}{#2}\}{\GLSp1@{#1}{#2}}[] }%
4016 }
```

\@GLSp1 Read in the final optional argument:

```
4017 \def\@GLSp1@#1#2[#3]{%
4018   \glsdoifexists{#2}%
4019 {%
4020   \let\do@gls@link@checkfirsthyper\gls@link@checkfirsthyper
4021   \let\glsifplural\firstoftwo
4022   \let\glscapscase\thirdofthree
4023   \let\glscustomtext\empty
4024   \def\glsinsert{#3}%
}
```

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

```
4025 \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.

```
4026 \@gls@link[#1]{#2}{\glo@text}%
```

Indicate that this entry has now been used

```
4027 \ifKV@glslink@local
4028   \glslocalunset{#2}%
4029 \else
4030   \glsunset{#2}%
4031 \fi
4032 }%
4033 \glspostlinkhook
4034 }
```

```
\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:

```
4035 \newrobustcmd*\glsdisp{\gls@hyp@opt@glsdisp}
```

Defined the un-starred form.

```
\@glsdisp
```

```
4036 \newcommand*\@glsdisp[3] []{  
4037   \glsdoifexists{#2}{%  
4038     \let\do@gls@link@checkfirsthyper@gls@link@checkfirsthyper  
4039     \let\glsifplural@\secondoftwo  
4040     \let\glscapscase@\firstofthree  
4041     \def\glscustomtext{#3}%  
4042     \def\glsinsert{}%
```

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

```
4043   \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%
```

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

```
4044   \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
4045   \ifKV@glslink@local  
4046     \glslocalunset{#2}%  
4047   \else  
4048     \glsunset{#2}%  
4049   \fi  
4050 }%  
4051 \glspostlinkhook  
4052 }
```

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

```
4053 \newcommand*\@gls@link@nocheckfirsthyper{}
```

```
@gls@field@link
```

```
4054 \newcommand{\@gls@field@link}[3]{  
4055   \glsdoifexists{#2}{%  
4056   {  
4057     \let\do@gls@link@checkfirsthyper@gls@link@nocheckfirsthyper  
4058     \@gls@link[#1]{#2}{#3}%  
4059   }%
```

```
4060 \glspostlinkhook  
4061 }
```

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

### \glstext

```
4062 \newrobustcmd*\{\glstext\}{\gls@hyp@opt\glstext}
```

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

```
4063 \newcommand*\{\glstext\}[2][]{%
```

```
4064 \new@ifnextchar[\{\gls@text@{\#1}{\#2}\}{\gls@text@{\#1}{\#2}[]}]
```

Read in the final optional argument:

```
4065 \def\{\glstext@{\#1}{\#3}}{%
```

```
4066 \gls@field@link{\#1}{\#2}{\glsentrytext{\#2}{\#3}}%
```

```
4067 }
```

\GLStext behaves like \glstext except the text is converted to uppercase.

### \GLStext

```
4068 \newrobustcmd*\{\GLStext\}{\gls@hyp@opt\GLStext}
```

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

```
4069 \newcommand*\{\GLStext\}[2][]{%
```

```
4070 \new@ifnextchar[\{\gls@text@{\#1}{\#2}\}{\gls@text@{\#1}{\#2}[]}]
```

Read in the final optional argument:

```
4071 \def\{\GLStext@{\#1}{\#3}}{%
```

```
4072 \gls@field@link{\#1}{\#2}{\mfirstucMakeUppercase{\glsentrytext{\#2}{\#3}}}%
```

```
4073 }
```

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

### \Glstext

```
4074 \newrobustcmd*\{\Glstext\}{\gls@hyp@opt\Glstext}
```

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

```
4075 \newcommand*\{\Glstext\}[2][]{%
```

```
4076 \new@ifnextchar[\{\gls@text@{\#1}{\#2}\}{\gls@text@{\#1}{\#2}[]}]
```

Read in the final optional argument:

```
4077 \def\{\Glstext@{\#1}{\#3}}{%
```

```
4078 \gls@field@link{\#1}{\#2}{\Glsentrytext{\#2}{\#3}}%
```

```
4079 }
```

\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

```
4080 \newrobustcmd*\{\glsfirst\}{\gls@hyp@opt\glsfirst}
```

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

```
4081 \newcommand*{\glsfirst}[2] []{%
4082   \new@ifnextchar[{\@glsfirst@{\#1}{\#2}}{\@glsfirst@{\#1}{\#2}[]}}}
```

Read in the final optional argument:

```
4083 \def\@glsfirst@#1#2[#3]{%
4084   \@gls@field@link{\#1}{\#2}{\glsentryfirst{\#2}\#3}%
4085 }
```

\Glsfirst behaves like \glsfirst except it displays the first letter in uppercase.

\Glsfirst

```
4086 \newrobustcmd*{\Glsfirst}{\gls@hyp@opt\@Glsfirst}
```

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

```
4087 \newcommand*{\@Glsfirst}[2] []{%
4088   \new@ifnextchar[{\@Glsfirst@{\#1}{\#2}}{\@Glsfirst@{\#1}{\#2}[]}}}
```

Read in the final optional argument:

```
4089 \def\@Glsfirst@#1#2[#3]{%
4090   \@gls@field@link{\#1}{\#2}{\Glsentryfirst{\#2}\#3}%
4091 }
```

\GLSfirst behaves like \Glsfirst except it displays the text in uppercase.

\GLSfirst

```
4092 \newrobustcmd*{\GLSfirst}{\gls@hyp@opt\@GLSfirst}
```

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

```
4093 \newcommand*{\@GLSfirst}[2] []{%
4094   \new@ifnextchar[{\@GLSfirst@{\#1}{\#2}}{\@GLSfirst@{\#1}{\#2}[]}}}
```

Read in the final optional argument:

```
4095 \def\@GLSfirst@#1#2[#3]{%
4096   \@gls@field@link{\#1}{\#2}{\mfirstucMakeUppercase{\glsentryfirst{\#2}\#3}}%
4097 }
```

\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

```
4098 \newrobustcmd*{\glsplural}{\gls@hyp@opt\@glsplural}
```

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

```
4099 \newcommand*{\@glsplural}[2] []{%
4100   \new@ifnextchar[{\@glsplural@{\#1}{\#2}}{\@glsplural@{\#1}{\#2}[]}}}
```

Read in the final optional argument:

```
4101 \def\@glsplural@#1#2[#3]{%
4102   \@gls@field@link{\#1}{\#2}{\glsentryplural{\#2}\#3}%
4103 }
```

\Glsplural behaves like \glsplural except that the first letter is converted to uppercase.

```

\Glsplural
4104 \newrobustcmd*\{\Glsplural\}{\gls@hyp@opt\Glsplural}

Defined the un-starred form. Need to determine if there is a final optional argument
4105 \newcommand*\{@Glsplural}[2] []{%
4106   \new@ifnextchar[{\@Glsplural@{\#1}{\#2}}{\@Glsplural@{\#1}{\#2}[]}}}

Read in the final optional argument:
4107 \def\@Glsplural@#1#2[#3]{%
4108   \gls@field@link{\#1}{\#2}{\Glsentryplural{\#2}{#3}}%
4109 }

\GLSplural behaves like \glsplural except that the text is converted to uppercase.

\GLSplural
4110 \newrobustcmd*\{\GLSplural\}{\gls@hyp@opt\GLSplural}

Defined the un-starred form. Need to determine if there is a final optional argument
4111 \newcommand*\{@GLSplural}[2] []{%
4112   \new@ifnextchar[{\@GLSplural@{\#1}{\#2}}{\@GLSplural@{\#1}{\#2}[]}}}

Read in the final optional argument:
4113 \def\@GLSplural@#1#2[#3]{%
4114   \gls@field@link{\#1}{\#2}{\mfirstucMakeUppercase{\glsentryplural{\#2}{#3}}}}%
4115 }

\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
4116 \newrobustcmd*\{\glsfirstplural\}{\gls@hyp@opt\glsfirstplural}

Defined the un-starred form. Need to determine if there is a final optional argument
4117 \newcommand*\{@glsfirstplural}[2] []{%
4118   \new@ifnextchar[{\@glsfirstplural@{\#1}{\#2}}{\@glsfirstplural@{\#1}{\#2}[]}}}

Read in the final optional argument:
4119 \def\@glsfirstplural@#1#2[#3]{%
4120   \gls@field@link{\#1}{\#2}{\glsentryfirstplural{\#2}{#3}}%
4121 }

\Glsfirstplural behaves like \glsfirstplural except that the first letter is converted
to uppercase.

\Glsfirstplural
4122 \newrobustcmd*\{\Glsfirstplural\}{\gls@hyp@opt\Glsfirstplural}

Defined the un-starred form. Need to determine if there is a final optional argument
4123 \newcommand*\{@Glsfirstplural}[2] []{%
4124   \new@ifnextchar[{\@Glsfirstplural@{\#1}{\#2}}{\@Glsfirstplural@{\#1}{\#2}[]}}}

```

Read in the final optional argument:

```
4125 \def\@Glsfirstplural@#1#2[#3]{%
4126   \gls@field@link{#1}{#2}{\Glsentryfirstplural{#2}#3}%
4127 }
```

\GLSfirstplural behaves like \glsfirstplural except that the link text is converted to uppercase.

\GLSfirstplural

```
4128 \newrobustcmd*\{\GLSfirstplural\}{\gls@hyp@opt\@GLSfirstplural}
```

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

```
4129 \newcommand*\{@GLSfirstplural}[2][]{%
4130   \new@ifnextchar[\{@GLSfirstplural@{#1}{#2}\}{\@GLSfirstplural@{#1}{#2}[]}}
```

Read in the final optional argument:

```
4131 \def\@GLSfirstplural@#1#2[#3]{%
4132   \gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryfirstplural{#2}#3}}%
4133 }
```

\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

```
4134 \newrobustcmd*\{\glsname\}{\gls@hyp@opt\@glsname}
```

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

```
4135 \newcommand*\{@glsname}[2][]{%
4136   \new@ifnextchar[\{@glsname@{#1}{#2}\}{\glsname@{#1}{#2}[]}}
```

Read in the final optional argument:

```
4137 \def\@glsname@#1#2[#3]{%
4138   \gls@field@link{#1}{#2}{\glsentryname{#2}#3}%
4139 }
```

\Glsname behaves like \glsname except that the first letter is converted to uppercase.

\Glsname

```
4140 \newrobustcmd*\{\Glsname\}{\gls@hyp@opt\@Glsname}
```

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

```
4141 \newcommand*\{@Glsname}[2][]{%
4142   \new@ifnextchar[\{@Glsname@{#1}{#2}\}{\Glsname@{#1}{#2}[]}}
```

Read in the final optional argument:

```
4143 \def\@Glsname@#1#2[#3]{%
4144   \gls@field@link{#1}{#2}{\Glsentryname{#2}#3}%
4145 }
```

\GLSname behaves like \glsname except that the link text is converted to uppercase.

\GLSname

```
4146 \newrobustcmd*\{\GLSname\}{\gls@hyp@opt\@GLSname}
```

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

```
4147 \newcommand*{\@GLSname}[2] []{%
4148   \new@ifnextchar[{\@GLSname@{\#1}{\#2}}{\@GLSname@{\#1}{\#2}[]}}%
```

Read in the final optional argument:

```
4149 \def\@GLSname@#1#2[#3]{%
4150   \@gls@field@link{\#1}{\#2}{\mfirstucMakeUppercase{\glsentryname{\#2}}#3}}%
4151 }
```

\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

```
4152 \newrobustcmd*{\glsdesc}{\@gls@hyp@opt\glsdesc}
```

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

```
4153 \newcommand*{\@glsdesc}[2] []{%
4154   \new@ifnextchar[{\@glsdesc@{\#1}{\#2}}{\@glsdesc@{\#1}{\#2}[]}}%
```

Read in the final optional argument:

```
4155 \def\@glsdesc@#1#2[#3]{%
4156   \@gls@field@link{\#1}{\#2}{\glsentrydesc{\#2}}#3}%
4157 }
```

\Glsdesc behaves like \glsdesc except that the first letter is converted to uppercase.

\Glsdesc

```
4158 \newrobustcmd*{\Glsdesc}{\@gls@hyp@opt\Glsdesc}
```

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

```
4159 \newcommand*{\@Glsdesc}[2] []{%
4160   \new@ifnextchar[{\@Glsdesc@{\#1}{\#2}}{\@Glsdesc@{\#1}{\#2}[]}}%
```

Read in the final optional argument:

```
4161 \def\@Glsdesc@#1#2[#3]{%
4162   \@gls@field@link{\#1}{\#2}{\Glsentrydesc{\#2}}#3}%
4163 }
```

\GLSdesc behaves like \glsdesc except that the link text is converted to uppercase.

\GLSdesc

```
4164 \newrobustcmd*{\GLSdesc}{\@gls@hyp@opt\GLSdesc}
```

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

```
4165 \newcommand*{\@GLSdesc}[2] []{%
4166   \new@ifnextchar[{\@GLSdesc@{\#1}{\#2}}{\@GLSdesc@{\#1}{\#2}[]}}%
```

Read in the final optional argument:

```
4167 \def\@GLSdesc@#1#2[#3]{%
4168   \@gls@field@link{\#1}{\#2}{\mfirstucMakeUppercase{\glsentrydesc{\#2}}#3}}%
4169 }
```

\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
4170 \newrobustcmd*{\glsdescplural}{\gls@hyp@opt@glsdescplural}

Define the un-starred form. Need to determine if there is a final optional argument
4171 \newcommand*{\glsdescplural}[2][]{%
4172   \new@ifnextchar[{\glsdescplural@{\#1}{\#2}}{\glsdescplural@{\#1}{\#2}[]}}}

Read in the final optional argument:
4173 \def\glsdescplural@#1#2[#3]{%
4174   \gls@field@link{\#1}{\#2}{\glsentrydescplural{\#2}{#3}}%
4175 }

\Glsdescplural behaves like \glsdescplural except that the first letter is converted to
uppercase.

\Glsdescplural
4176 \newrobustcmd*{\Glsdescplural}{\gls@hyp@opt@Glsdescplural}

Define the un-starred form. Need to determine if there is a final optional argument
4177 \newcommand*{\Glsdescplural}[2][]{%
4178   \new@ifnextchar[{\Glsdescplural@{\#1}{\#2}}{\Glsdescplural@{\#1}{\#2}[]}}}

Read in the final optional argument:
4179 \def\Glsdescplural@#1#2[#3]{%
4180   \gls@field@link{\#1}{\#2}{\Glsentrydescplural{\#2}{#3}}%
4181 }

\GLSdescplural behaves like \glsdescplural except that the link text is converted to
uppercase.

\GLSdescplural
4182 \newrobustcmd*{\GLSdescplural}{\gls@hyp@opt@GLSdescplural}

Define the un-starred form. Need to determine if there is a final optional argument
4183 \newcommand*{\GLSdescplural}[2][]{%
4184   \new@ifnextchar[{\GLSdescplural@{\#1}{\#2}}{\GLSdescplural@{\#1}{\#2}[]}}}

Read in the final optional argument:
4185 \def\GLSdescplural@#1#2[#3]{%
4186   \gls@field@link{\#1}{\#2}{\mfirstucMakeUppercase{\glsentrydescplural{\#2}{#3}}}}%
4187 }

\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
4188 \newrobustcmd*{\glssymbol}{\gls@hyp@opt@glssymbol}

Defined the un-starred form. Need to determine if there is a final optional argument
4189 \newcommand*{\glssymbol}[2][]{%
4190   \new@ifnextchar[{\glssymbol@{\#1}{\#2}}{\glssymbol@{\#1}{\#2}[]}}}

```

Read in the final optional argument:

```
4191 \def\@glssymbol@#1#2[#3]{%
4192   \gls@field@link{#1}{#2}{\glsentrysymbol{#2}#3}%
4193 }
```

\Glssymbol behaves like \glssymbol except that the first letter is converted to uppercase.

\Glssymbol

```
4194 \newrobustcmd*\{\Glssymbol\}{\gls@hyp@opt\@Glssymbol}
```

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

```
4195 \newcommand*{\@Glssymbol}[2][]{%
4196   \new@ifnextchar[{\@Glssymbol@#1}{#2}}{\@Glssymbol@#1}{#2}[])}
```

Read in the final optional argument:

```
4197 \def\@Glssymbol@#1#2[#3]{%
4198   \gls@field@link{#1}{#2}{\glsentrysymbol{#2}#3}%
4199 }
```

\GLSsymbol behaves like \glssymbol except that the link text is converted to uppercase.

\GLSsymbol

```
4200 \newrobustcmd*\{\GLSsymbol\}{\gls@hyp@opt\@GLSsymbol}
```

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

```
4201 \newcommand*{\@GLSsymbol}[2][]{%
4202   \new@ifnextchar[{\@GLSsymbol@#1}{#2}}{\@GLSsymbol@#1}{#2}[])}
```

Read in the final optional argument:

```
4203 \def\@GLSsymbol@#1#2[#3]{%
4204   \gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrysymbol{#2}#3}}%
4205 }
```

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

glssymbolplural

```
4206 \newrobustcmd*\{\glssymbolplural\}{\gls@hyp@opt\@glssymbolplural}
```

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

```
4207 \newcommand*{\@glssymbolplural}[2][]{%
4208   \new@ifnextchar[{\@glssymbolplural@#1}{#2}}{\@glssymbolplural@#1}{#2}[])}
```

Read in the final optional argument:

```
4209 \def\@glssymbolplural@#1#2[#3]{%
4210   \gls@field@link{#1}{#2}{\glsentrysymbolplural{#2}#3}%
4211 }
```

\Glssymbolplural behaves like \glssymbolplural except that the first letter is converted to uppercase.

Glssymbolplural

```
4212 \newrobustcmd*\{\Glssymbolplural\}{\gls@hyp@opt\@Glssymbolplural}
```

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

```
4213 \newcommand*{\@Glssymbolplural}[2] [] {%
4214   \new@ifnextchar[{\@Glssymbolplural@{\#1}{\#2}}{\@Glssymbolplural@{\#1}{\#2}[]}}%
```

Read in the final optional argument:

```
4215 \def\@Glssymbolplural@#1#2[#3]{%
4216   \gls@field@link{\#1}{\#2}{\glsentrysymbolplural{\#2}#3}}%
4217 }
```

\GLSsymbolplural behaves like \glssymbolplural except that the link text is converted to uppercase.

## GLSsymbolplural

```
4218 \newrobustcmd*{\GLSsymbolplural}{\gls@hyp@opt\@GLSsymbolplural}
```

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

```
4219 \newcommand*{\@GLSsymbolplural}[2] [] {%
4220   \new@ifnextchar[{\@GLSsymbolplural@{\#1}{\#2}}{\@GLSsymbolplural@{\#1}{\#2}[]}}%
```

Read in the final optional argument:

```
4221 \def\@GLSsymbolplural@#1#2[#3]{%
4222   \gls@field@link{\#1}{\#2}{\mfirstucMakeUppercase{\glsentrysymbolplural{\#2}#3}}%
4223 }
```

\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

```
4224 \newrobustcmd*{\glsuseri}{\gls@hyp@opt\glsuseri}
```

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

```
4225 \newcommand*{\@glsuseri}[2] [] {%
4226   \new@ifnextchar[{\@glsuseri@{\#1}{\#2}}{\@glsuseri@{\#1}{\#2}[]}}%
```

Read in the final optional argument:

```
4227 \def\@glsuseri@#1#2[#3]{%
4228   \gls@field@link{\#1}{\#2}{\glsentryuseri{\#2}#3}}%
4229 }
```

\Glsuseri behaves like \glsuseri except that the first letter is converted to uppercase.

## \Glsuseri

```
4230 \newrobustcmd*{\Glsuseri}{\gls@hyp@opt\Glsuseri}
```

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

```
4231 \newcommand*{\@Glsuseri}[2] [] {%
4232   \new@ifnextchar[{\@Glsuseri@{\#1}{\#2}}{\@Glsuseri@{\#1}{\#2}[]}}%
```

Read in the final optional argument:

```
4233 \def\@Glsuseri@#1#2[#3]{%
4234   \gls@field@link{\#1}{\#2}{\Glsentryuseri{\#2}#3}}%
4235 }
```

\GLSuseri behaves like \glsuseri except that the link text is converted to uppercase.

### \GLSuseri

```
4236 \newrobustcmd*\{\GLSuseri\}{\gls@hyp@opt\GLSuseri}
```

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

```
4237 \newcommand*\{@GLSuseri}[2][]{%
```

```
4238 \new@ifnextchar[{\@GLSuseri@{\#1}{\#2}}{\@GLSuseri@{\#1}{\#2}[]}]}
```

Read in the final optional argument:

```
4239 \def\@GLSuseri@#1#2[#3]{%
```

```
4240 \gls@field@link{\#1}{\#2}{\mfirstucMakeUppercase{\glsentryuseri{\#2}\#3}}%
```

```
4241 }
```

\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

```
4242 \newrobustcmd*\{\glsuserii\}{\gls@hyp@opt\glsuserii}
```

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

```
4243 \newcommand*\{@glsuserii}[2][]{%
```

```
4244 \new@ifnextchar[{\@glsuserii@{\#1}{\#2}}{\@glsuserii@{\#1}{\#2}[]}]}
```

Read in the final optional argument:

```
4245 \def\@glsuserii@#1#2[#3]{%
```

```
4246 \gls@field@link{\#1}{\#2}{\glsentryuserii{\#2}\#3}}%
```

```
4247 }
```

\Glsuserii behaves like \glsuserii except that the first letter is converted to uppercase.

### \Glsuserii

```
4248 \newrobustcmd*\{\Glsuserii\}{\gls@hyp@opt\Glsuserii}
```

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

```
4249 \newcommand*\{@Glsuserii}[2][]{%
```

```
4250 \new@ifnextchar[{\@Glsuserii@{\#1}{\#2}}{\@Glsuserii@{\#1}{\#2}[]}]}
```

Read in the final optional argument:

```
4251 \def\@Glsuserii@#1#2[#3]{%
```

```
4252 \gls@field@link{\#1}{\#2}{\Glsentryuserii{\#2}\#3}}%
```

```
4253 }
```

\GLSuserii behaves like \glsuserii except that the link text is converted to uppercase.

### \GLSuserii

```
4254 \newrobustcmd*\{\GLSuserii\}{\gls@hyp@opt\GLSuserii}
```

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

```
4255 \newcommand*\{@GLSuserii}[2][]{%
```

```
4256 \new@ifnextchar[{\@GLSuserii@{\#1}{\#2}}{\@GLSuserii@{\#1}{\#2}[]}]}
```

Read in the final optional argument:

```
4257 \def\@GLSuseriii@#1#2[#3]{%
4258   \gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseriii{#2}}#3}%
4259 }
```

\glsuseriii behaves like \gls except it always uses the value given by the user3 key and it doesn't mark the entry as used.

\glsuserii

```
4260 \newrobustcmd*\glsuseriii{\gls@hyp@opt\glsuseriii}
```

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

```
4261 \newcommand*\@glsuseriii[2][]{%
4262   \new@ifnextchar[\glsuseriii@#1]{\glsuseriii@#1[]}{\glsuseriii@#1#2[]}}
```

Read in the final optional argument:

```
4263 \def\@glsuseriii@#1#2[#3]{%
4264   \gls@field@link{#1}{#2}{\glsentryuseriii{#2}}#3}%
4265 }
```

\Glsuseriii behaves like \glsuseriii except that the first letter is converted to uppercase.

\Glsuserii

```
4266 \newrobustcmd*\Glsuseriii{\gls@hyp@opt\Glsuseriii}
```

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

```
4267 \newcommand*\@Glsuseriii[2][]{%
4268   \new@ifnextchar[\Glsuseriii@#1]{\Glsuseriii@#1[]}{\Glsuseriii@#1#2[]}}
```

Read in the final optional argument:

```
4269 \def\@Glsuseriii@#1#2[#3]{%
4270   \gls@field@link{#1}{#2}{\Glsentryuseriii{#2}}#3}%
4271 }
```

\GLSuseriii behaves like \glsuseriii except that the link text is converted to uppercase.

\GLSuserii

```
4272 \newrobustcmd*\GLSuseriii{\gls@hyp@opt\GLSuseriii}
```

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

```
4273 \newcommand*\@GLSuseriii[2][]{%
4274   \new@ifnextchar[\GLSuseriii@#1]{\GLSuseriii@#1[]}{\GLSuseriii@#1#2[]}}
```

Read in the final optional argument:

```
4275 \def\@GLSuseriii@#1#2[#3]{%
4276   \gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseriii{#2}}#3}%
4277 }
```

\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
4278 \newrobustcmd*\{\glsuseriv\}{\gls@hyp@opt@glsuseriv}

    Define the un-starred form. Need to determine if there is a final optional argument
4279 \newcommand*{\glsuseriv}[2][]{%
4280   \new@ifnextchar[{\glsuseriv@{\#1}{\#2}}{\glsuseriv@{\#1}{\#2}[]}]}

    Read in the final optional argument:
4281 \def\glsuseriv@#1#2[#3]{%
4282   \gls@field@link{\#1}{\#2}{\glsentryuseriv{\#2}{\#3}}%
4283 }

    \Glsuseriv behaves like \glsuseriv except that the first letter is converted to uppercase.

\Glsuseriv
4284 \newrobustcmd*\{\Glsuseriv\}{\gls@hyp@opt\Glsuseriv}

    Define the un-starred form. Need to determine if there is a final optional argument
4285 \newcommand*{\Glsuseriv}[2][]{%
4286   \new@ifnextchar[{\Glsuseriv@{\#1}{\#2}}{\Glsuseriv@{\#1}{\#2}[]}]}

    Read in the final optional argument:
4287 \def\Glsuseriv@#1#2[#3]{%
4288   \gls@field@link{\#1}{\#2}{\Glsentryuseriv{\#2}{\#3}}%
4289 }

    \GLSuseriv behaves like \glsuseriv except that the link text is converted to uppercase.

\GLSuseriv
4290 \newrobustcmd*\{\GLSuseriv\}{\gls@hyp@opt\GLSuseriv}

    Define the un-starred form. Need to determine if there is a final optional argument
4291 \newcommand*{\GLSuseriv}[2][]{%
4292   \new@ifnextchar[{\GLSuseriv@{\#1}{\#2}}{\GLSuseriv@{\#1}{\#2}[]}]}

    Read in the final optional argument:
4293 \def\GLSuseriv@#1#2[#3]{%
4294   \gls@field@link{\#1}{\#2}{\mfirstucMakeUppercase{\glsentryuseriv{\#2}{\#3}}}}%
4295 }

    \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
4296 \newrobustcmd*\{\glsuserv\}{\gls@hyp@opt@glsuserv}

    Define the un-starred form. Need to determine if there is a final optional argument
4297 \newcommand*{\glsuserv}[2][]{%
4298   \new@ifnextchar[{\glsuserv@{\#1}{\#2}}{\glsuserv@{\#1}{\#2}[]}]}

    Read in the final optional argument:
4299 \def\glsuserv@#1#2[#3]{%
4300   \gls@field@link{\#1}{\#2}{\glsentryuserv{\#2}{\#3}}%
4301 }

```

\Glsuserv behaves like \glsuserv except that the first letter is converted to uppercase.

#### \Glsuserv

```
4302 \newrobustcmd*\{\Glsuserv\}{\gls@hyp@opt\Glsuserv}
```

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

```
4303 \newcommand*{\Glsuserv}[2][]{%
```

```
4304 \new@ifnextchar[{\Glsuserv@{#1}{#2}}{\Glsuserv@{#1}{#2}[]}]
```

Read in the final optional argument:

```
4305 \def\Glsuserv@#1#2[#3]{%
```

```
4306   \gls@field@link{#1}{#2}{\Glsentryuserv{#2}#3}%
```

```
4307 }
```

\GLSuserv behaves like \glsuserv except that the link text is converted to uppercase.

#### \GLSuserv

```
4308 \newrobustcmd*\{\GLSuserv\}{\gls@hyp@opt\GLSuserv}
```

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

```
4309 \newcommand*{\GLSuserv}[2][]{%
```

```
4310 \new@ifnextchar[{\GLSuserv@{#1}{#2}}{\GLSuserv@{#1}{#2}[]}]
```

Read in the final optional argument:

```
4311 \def\GLSuserv@#1#2[#3]{%
```

```
4312   \gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentryuserv{#2}#3}}%
```

```
4313 }
```

\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

```
4314 \newrobustcmd*\{\glsuservi\}{\gls@hyp@opt\glsuservi}
```

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

```
4315 \newcommand*{\glsuservi}[2][]{%
```

```
4316 \new@ifnextchar[{\glsuservi@{#1}{#2}}{\glsuservi@{#1}{#2}[]}]
```

Read in the final optional argument:

```
4317 \def\glsuservi@#1#2[#3]{%
```

```
4318   \gls@field@link{#1}{#2}{\Glsentryuservi{#2}#3}%
```

```
4319 }
```

\Glsuservi behaves like \glsuservi except that the first letter is converted to uppercase.

#### \Glsuservi

```
4320 \newrobustcmd*\{\Glsuservi\}{\gls@hyp@opt\Glsuservi}
```

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

```
4321 \newcommand*{\Glsuservi}[2][]{%
```

```
4322 \new@ifnextchar[{\Glsuservi@{#1}{#2}}{\Glsuservi@{#1}{#2}[]}]
```

Read in the final optional argument:

```
4323 \def\@Glsuservi@#1#2[#3]{%
4324   \gls@field@link{#1}{#2}{\Glsentryuservi{#2}#3}%
4325 }
```

\GLSuservi behaves like \glsuservi except that the link text is converted to uppercase.

\GLSuservi

```
4326 \newrobustcmd*\{\GLSuservi\}{\gls@hyp@opt\GLSuservi}
```

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

```
4327 \newcommand*{\@GLSuservi}[2][]{%
4328   \new@ifnextchar[\{\@GLSuservi@#1}{\@GLSuservi@#1}[] ]}%
```

Read in the final optional argument:

```
4329 \def\@GLSuservi@#1#2[#3]{%
4330   \gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuservi{#2}#3}}%
4331 }
```

Now deal with acronym related keys. First the short form:

\acrshort

```
4332 \newrobustcmd*\{\acrshort\}{\gls@hyp@opt\ns@acrshort}
```

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

```
4333 \newcommand*{\ns@acrshort}[2][]{%
4334   \new@ifnextchar[\{\@acrshort{#1}{#2}\}{\@acrshort{#1}{#2}[] }%
```

```
4335 }
```

Read in the final optional argument:

```
4336 \def\@acrshort#1#2[#3]{%
4337   \glsdoifexists{#2}%
4338   {%
4339     \let\do@gls@link@checkfirsthyper\gls@link@nocheckfirsthyper
4340     \let\glsifplural\@secondoftwo
4341     \let\glscapscase\@firstofthree
4342     \let\glsinsert\@empty
4343     \def\glscustomtext{%
4344       \acronymfont{\glsentryshort{#2}}#3%
4345     }%
4346 }
```

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

```
4346   \gls@link[#1]{#2}{\csname gls@\glstype\entryfmt\endcsname}%
4347 }%
4348 \glspostlinkhook
4349 }
```

\Acrshort

```
4350 \newrobustcmd*\{\Acrshort\}{\gls@hyp@opt\ns@Acrshort}
```

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

```
4351 \newcommand*{\ns@Acrshort}[2] [] {%
4352   \new@ifnextchar[{\@Acrshort{#1}{#2}}{\@Acrshort{#1}{#2}[] }%
4353 }
```

Read in the final optional argument:

```
4354 \def\@Acrshort#1#2[#3]{%
4355   \glsdoifexists{#2}%
4356   {%
4357     \let\do@gls@link@checkfirsthyper\gls@link@nocheckfirsthyper
4358     \def\glslabel{#2}%
4359     \let\glsifplural@\secondoftwo
4360     \let\glscapscase@\secondofthree
4361     \let\glsinsert@\empty
4362     \def\glscustomtext{%
4363       \acronymfont{\glsentryshort{#2}}#3%
4364     }%
4365 }
```

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

```
4365   \gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4366 }
4367 \glspostlinkhook
4368 }
```

## \ACRshort

```
4369 \newrobustcmd*{\ACRshort}{\gls@hyp@opt\ns@ACRshort}
```

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

```
4370 \newcommand*{\ns@ACRshort}[2] [] {%
4371   \new@ifnextchar[{\@ACRshort{#1}{#2}}{\@ACRshort{#1}{#2}[] }%
4372 }
```

Read in the final optional argument:

```
4373 \def\@ACRshort#1#2[#3]{%
4374   \glsdoifexists{#2}%
4375   {%
4376     \let\do@gls@link@checkfirsthyper\gls@link@nocheckfirsthyper
4377     \def\glslabel{#2}%
4378     \let\glsifplural@\secondoftwo
4379     \let\glscapscase@\thirdofthree
4380     \let\glsinsert@\empty
4381     \def\glscustomtext{%
4382       \mfirstrucMakeUppercase{\acronymfont{\glsentryshort{#2}}#3}%
4383     }%
4384 }
```

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

```
4384     \gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4385 }
4386 \glspostlinkhook
4387 }
```

Short plural:

\acrshortpl

```
4388 \newrobustcmd*\acrshortpl{\gls@hyp@opt\ns@acrshortpl}
```

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

```
4389 \newcommand*\ns@acrshortpl[2][]{%
4390   \new@ifnextchar[\@acrshortpl[#1]{#2}{\@acrshortpl[#1]{#2}[]}}%
4391 }
```

Read in the final optional argument:

```
4392 \def\acrshortpl#1#2[#3]{%
4393   \glsdoifexists{#2}%
4394   {%
4395     \let\do@gls@link@checkfirsthyper\gls@link@nocheckfirsthyper
4396     \def\glslabel{#2}%
4397     \let\glsifplural\@firstoftwo
4398     \let\glscapscase\@firstofthree
4399     \let\glsinsert\@empty
4400     \def\glscustomtext{%
4401       \acronymfont{\glsentryshortpl{#2}}#3%
4402     }%
4403   }
```

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

```
4404     \gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4405   }
4406 }
```

\Acrshortpl

```
4407 \newrobustcmd*\Acrshortpl{\gls@hyp@opt\ns@Acrshortpl}
```

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

```
4408 \newcommand*\ns@Acrshortpl[2][]{%
4409   \new@ifnextchar[\@Acrshortpl[#1]{#2}{\@Acrshortpl[#1]{#2}[]}}%
4410 }
```

Read in the final optional argument:

```
4411 \def\Acrshortpl#1#2[#3]{%
4412   \glsdoifexists{#2}%
4413   {%
```

```

4414 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
4415 \def\glslabel{#2}%
4416 \let\glsifplural@\firstoftwo
4417 \let\glscapscase@\secondofthree
4418 \let\glsinsert@\empty
4419 \def\glscustomtext{%
4420   \acronymfont{\Glsentryshortpl{#2}}#3%
4421 }%

```

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

```

4422 \gls@link[#1]{#2}{\csname gls@\glstype \entryfmt\endcsname}%
4423 }%

```

```

4424 \glspostlinkhook
4425 }

```

## \ACRshortpl

```
4426 \newrobustcmd*\ACRshortpl{\gls@hyp@opt\ns@ACRshortpl}
```

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

```

4427 \newcommand*{\ns@ACRshortpl}[2][]{%
4428   \new@ifnextchar[\@ACRshortpl{#1}{#2}]{\@ACRshortpl{#1}{#2}[]}{%
4429 }

```

Read in the final optional argument:

```

4430 \def\@ACRshortpl#1#2[#3]{%
4431   \glsdoifexists{#2}%
4432   {%

```

```

4433 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
4434 \def\glslabel{#2}%
4435 \let\glsifplural@\firstoftwo
4436 \let\glscapscase@\thirdofthree
4437 \let\glsinsert@\empty
4438 \def\glscustomtext{%
4439   \mfirstrucMakeUppercase{\acronymfont{\glsentryshortpl{#2}}#3}%
4440 }%

```

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

```

4441 \gls@link[#1]{#2}{\csname gls@\glstype \entryfmt\endcsname}%
4442 }%

```

```

4443 \glspostlinkhook
4444 }

```

## \acrlong

```
4445 \newrobustcmd*\acrlong{\gls@hyp@opt\ns@acrlong}
```

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

```
4446 \newcommand*{\ns@acrlong}[2] [] {%
4447   \new@ifnextchar[{\@\acrlong{#1}{#2}}{\@\acrlong{#1}{#2}[] }%
4448 }
```

Read in the final optional argument:

```
4449 \def\@acrlong#1#2[#3]{%
4450   \glsdoifexists{#2}%
4451   {%
4452     \let\do@gls@link@checkfirsthyper\gls@link@nocheckfirsthyper
4453     \def\glslabel{#2}%
4454     \let\glsifplural\@secondoftwo
4455     \let\glscapscase\@firstofthree
4456     \let\glsinsert\@empty}
```

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

```
4457   \def\glscustomtext{%
4458     \glsentrylong{#2}#3%
4459   }%
```

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

```
4460   \gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4461 }%
4462 \glspostlinkhook
4463 }
```

## \Acrlong

```
4464 \newrobustcmd*{\Acrlong}{\gls@hyp@opt\ns@Acrlong}
```

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

```
4465 \newcommand*{\ns@Acrlong}[2] [] {%
4466   \new@ifnextchar[{\@\Acrlong{#1}{#2}}{\@\Acrlong{#1}{#2}[] }%
4467 }
```

Read in the final optional argument:

```
4468 \def\@Acrlong#1#2[#3]{%
4469   \glsdoifexists{#2}%
4470   {%
4471     \let\do@gls@link@checkfirsthyper\gls@link@nocheckfirsthyper
4472     \def\glslabel{#2}%
4473     \let\glsifplural\@secondoftwo
4474     \let\glscapscase\@secondofthree
4475     \let\glsinsert\@empty}
```

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

```
4476 \def\glscustomtext{%
4477   \Glsentrylong{\#2}\#3%
4478 }%
4481 Call \@gls@link. Note that \@gls@link sets \glstype.
4482 }%
4483 \newrobustcmd*\ACRlong{\gls@hyp@opt\ns@ACRlong}
```

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

```
4484 \newcommand*\ns@ACRlong[2][]{%
4485   \new@ifnextchar[\{@ACRlong{\#1}{\#2}\}{\@ACRlong{\#1}{\#2}[]}}%
4486 }
```

Read in the final optional argument:

```
4487 \def\@ACRlong#1#2[#3]{%
4488   \glsdoifexists{\#2}%
4489 }%
4490 \let\do@gls@link@checkfirsthyper\gls@link@nocheckfirsthyper
4491 \def\glslabel{\#2}%
4492 \let\glsifplural\@secondoftwo
4493 \let\glscapscase\@thirdofthree
4494 \let\glsinsert\@empty
```

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

```
4495 \def\glscustomtext{%
4496   \mfirstrucMakeUppercase{\Glsentrylong{\#2}\#3}%
4497 }%
```

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

```
4498 \newrobustcmd*\ns@ACRlong{\gls@hyp@opt\ns@ACRlong}%
4499 }%
4500 \glspostlinkhook
4501 }
```

Short plural:

```
\acrlongpl
4502 \newrobustcmd*\acrlongpl{\gls@hyp@opt\ns@ACRlong}
```

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

```
4503 \newcommand*{\ns@acrlongpl}[2] []{%
4504   \new@ifnextchar[{\@\acrlongpl{#1}{#2}}{\@\acrlongpl{#1}{#2}[] }%
4505 }
```

Read in the final optional argument:

```
4506 \def\@acrlongpl#1#2[#3]{%
4507   \glsdoifexists{#2}%
4508   {%
4509     \let\do@gls@link@checkfirsthyper\gls@link@nocheckfirsthyper
4510     \def\glslabel{#2}%
4511     \let\glsifplural@\firstoftwo
4512     \let\glscapscase@\firstofthree
4513     \let\glsinsert@\empty
```

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

```
4514   \def\glscustomtext{%
4515     \glsentrylongpl{#2}#3%
4516   }%
```

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

```
4517   \gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4518 }%
4519 \glspostlinkhook
4520 }
```

## \Acrlongpl

```
4521 \newrobustcmd*{\Acrlongpl}{\gls@hyp@opt\ns@Acrlongpl}
```

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

```
4522 \newcommand*{\ns@Acrlongpl}[2] []{%
4523   \new@ifnextchar[{\@\Acrlongpl{#1}{#2}}{\@\Acrlongpl{#1}{#2}[] }%
4524 }
```

Read in the final optional argument:

```
4525 \def\@Acrlongpl#1#2[#3]{%
4526   \glsdoifexists{#2}%
4527   {%
4528     \let\do@gls@link@checkfirsthyper\gls@link@nocheckfirsthyper
4529     \def\glslabel{#2}%
4530     \let\glsifplural@\firstoftwo
4531     \let\glscapscase@\secondofthree
4532     \let\glsinsert@\empty
```

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

```
4533 \def\glscustomtext{%
4534   \Glsentrylongpl{\#2}\#3%
4535 }%
4536   \gls@link[\#1]{\#2}{\csname gls@\glstype @entryfmt\endcsname}%
4537 }%
4538 \glspostlinkhook
4539 }
```

## \ACRlongpl

```
4540 \newrobustcmd*\ACRlongpl{\gls@hyp@opt\ns@ACRlongpl}
```

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

```
4541 \newcommand*\ns@ACRlongpl[2][]{%
4542   \new@ifnextchar[\{\ns@ACRlongpl{\#1}{\#2}\}{\ns@ACRlongpl{\#1}{\#2}}[]]%
4543 }
```

Read in the final optional argument:

```
4544 \def\ACRlongpl#1#2[#3]{%
4545   \glsdoifexists{\#2}%
4546   {%
4547     \let\do@gls@link@checkfirsthyper\gls@link@nocheckfirsthyper
4548     \def\glslabel{\#2}%
4549     \let\glsifplural\@firstoftwo
4550     \let\glscapscase\@thirdofthree
4551     \let\glsinsert\@empty
```

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

```
4552 \def\glscustomtext{%
4553   \mfirstrucMakeUppercase{\Glsentrylongpl{\#2}\#3}%
4554 }%
```

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

```
4555   \gls@link[\#1]{\#2}{\csname gls@\glstype @entryfmt\endcsname}%
4556 }%
4557 \glspostlinkhook
4558 }
```

## 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}
```

```
4559 \newcommand*{\@gls@entry@field}[2]{%
4560   \csname glo@\glsdetoklabel{#1}@#2\endcsname
4561 }
```

`lsetentryfield` `\glsletentryfield{\cs}{\label}{\field}`

```
4562 \newcommand*{\glsletentryfield}[3]{%
4563   \letcs{\#1}{glo@\glsdetoklabel{#2}@#3}%
4564 }
```

`Gls@entry@field` Generic first letter uppercase version.

```
\@Gls@entry@field{\label}{\field}
```

```
4565 \newcommand*{\@Gls@entry@field}[2]{%
4566   \glsdoifexistsor{\#1}%
4567   {%
4568     \letcs{\glo@text}{glo@\glsdetoklabel{#1}@#2}%
4569     \ifdef{\glo@text}%
4570     {%
4571       \xmakefirstuc{\glo@text}%
4572     }%
4573     {%
4574       ??\PackageError{glossaries}{The field ‘#2’ doesn’t exist for glossary
4575         entry ‘\glsdetoklabel{#1}’}{Check you have correctly spelt the entry
4576         label and the field name}%
4577     }%
4578   }%
4579   {%
4580     ??%
4581   }%
4582 }
```

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
4583 \newcommand*{\glsentryname}[1]{\gls@entry@field{#1}{name}}
```

```
\Glsentryname
4584 \newrobustcmd*{\Glsentryname}[1]{%
4585   \gls@entryname{#1}%
4586 }
```

\@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:

```
4587 \newcommand*{\@Gls@entryname}[1]{%
4588   \gls@entry@field{#1}{name}%
4589 }
```

\s@acronymname Now the behaviour when \setacronymstyle is used:

```
4590 \newcommand*{\@Gls@acronymname}[1]{%
4591   \ifglshaslong{#1}%
4592   {%
4593     \letcs{\glo@text}{\glsdetoklabel{#1}{name}}%
\noexpand\gls@getbody is defined by \firststuc (which used to be part of glossaries).
```

```
4594   \expandafter\gls@getbody\glo@text{}@nil
4595   \expandafter\ifx\gls@body\glsentrylong\relax
4596     \expandafter\Glsentrylong\gls@rest
4597   \else
4598     \expandafter\ifx\gls@body\glsentryshort\relax
4599       \expandafter\Glsentryshort\gls@rest
4600     \else
4601       \expandafter\ifx\gls@body\acronymfont\relax
```

Temporarily make \glsentryshort behave like \Glsentryshort. (This is on the assumption that the argument of \acronymfont is \glsentryshort{\label}, as that's the behaviour of the predefined acronym styles.) This is scoped to localise the effect of the assignment.

```
4602   {%
4603     \let\glsentryshort\Glsentryshort
4604     \glo@text
4605   }%
4606   \else
4607     \expandafter\ifx\gls@body\glsshortaccessdisplay\relax
4608     {%
4609       \let\glsentryshort\Glsentryshort
4610       \glo@text
4611     }%
4612   \else
4613     \xmakefirststuc{\glo@text}%
4614   \fi
4615 \fi
4616 \fi
```

```

4617     \fi
4618 }%
4619 {%

```

Not an acronym

```

4620     \Gls@entry@field{#1}{name}%
4621 }%
4622 }

```

Get the entry description (as specified by the description when the entry was defined). The argument is the label associated with the entry. Note that unless you used `description=false` in the `sanitize` package option you may get unexpected results if the description key contained any commands.

```
\glsentrydesc
4623 \newcommand*{\glsentrydesc}[1]{\Gls@entry@field{#1}{desc}}
```

```
\Glsentrydesc
4624 \newrobustcmd*{\Glsentrydesc}[1]{%
4625   \Gls@entry@field{#1}{desc}%
4626 }
```

Plural form:

```
entrydescplural
4627 \newcommand*{\glsentrydescplural}[1]{%
4628   \Gls@entry@field{#1}{descplural}%
4629 }
```

```
entrydescplural
4630 \newrobustcmd*{\Glsentrydescplural}[1]{%
4631   \Gls@entry@field{#1}{descplural}%
4632 }
```

Get the entry text, as specified by the text key when the entry was defined. The argument is the label associated with the entry:

```
\glsentrytext
4633 \newcommand*{\glsentrytext}[1]{\Gls@entry@field{#1}{text}}
```

```
\Glsentrytext
4634 \newrobustcmd*{\Glsentrytext}[1]{%
4635   \Gls@entry@field{#1}{text}%
4636 }
```

Get the plural form:

```
\glsentryplural  
4637 \newcommand*{\glsentryplural}[1]{%  
4638   \gls@entry@field{#1}{plural}}%  
4639 }
```

```
\Glsentryplural  
4640 \newrobustcmd*{\Glsentryplural}[1]{%  
4641   \gls@entry@field{#1}{plural}}%  
4642 }
```

Get the symbol associated with this entry. The argument is the label associated with the entry.

```
\glsentrysymbol  
4643 \newcommand*{\glsentrysymbol}[1]{%  
4644   \gls@entry@field{#1}{symbol}}%  
4645 }
```

```
\Glsentrysymbol  
4646 \newrobustcmd*{\Glsentrysymbol}[1]{%  
4647   \gls@entry@field{#1}{symbol}}%  
4648 }
```

Plural form:

```
trysymbolplural  
4649 \newcommand*{\glsentrysymbolplural}[1]{%  
4650   \gls@entry@field{#1}{symbolplural}}%  
4651 }
```

```
trysymbolplural  
4652 \newrobustcmd*{\Glsentrysymbolplural}[1]{%  
4653   \gls@entry@field{#1}{symbolplural}}%  
4654 }
```

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  
4655 \newcommand*{\glsentryfirst}[1]{%  
4656   \gls@entry@field{#1}{first}}%  
4657 }
```

```
\Glsentryfirst  
4658 \newrobustcmd*{\Glsentryfirst}[1]{%  
4659   \gls@entry@field{#1}{first}}%  
4660 }
```

Get the plural form (as specified by the `firstplural` key when the entry was defined).

```

ntryfirstplural
4661 \newcommand*{\glsentryfirstplural}[1]{%
4662   \@gls@entry@field{#1}{firstpl}%
4663 }

ntryfirstplural
4664 \newrobustcmd*{\Glsentryfirstplural}[1]{%
4665   \@Gls@entry@field{#1}{firstpl}%
4666 }

sentrytitlecase
4667 \newrobustcmd*{\@glsentrytitlecase}[2]{%
4668   \glsdoifexists{#1}%
4669   {%
4670     \glsfieldfetch{#1}{#2}{\@gls@value}%
4671     \xcapitalisewords{\@gls@value}%
4672   }%
4673 }
4674 \ifdef\texorpdfstring
4675 {
4676   \newcommand*{\glsentrytitlecase}[2]{%
4677     \texorpdfstring
4678     {\@glsentrytitlecase{#1}{#2}}%
4679     {\@gls@entry@field{#1}{#2}}%
4680   }
4681 }
4682 {
4683   \newcommand*{\glsentrytitlecase}[2]{\@glsentrytitlecase{#1}{#2}}
4684 }

```

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

```
\glsentrytype
4685 \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
4686 \newcommand*{\glsentrysort}[1]{%
4687   \@gls@entry@field{#1}{sort}%
4688 }
```

\glsentryparent Expands to the label of the entry's parent.

```
4689 \newcommand*{\glsentryparent}[1]{%
4690   \@gls@entry@field{#1}{parent}%
4691 }
```

```

\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.
4692 \newcommand*{\glsentryuseri}[1]{%
4693   \gls@entry@field{#1}{useri}%
4694 }

\Glsentryuseri
4695 \newrobustcmd*{\Glsentryuseri}[1]{%
4696   \gls@entry@field{#1}{useri}%
4697 }

\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.
4698 \newcommand*{\glsentryuserii}[1]{%
4699   \gls@entry@field{#1}{userii}%
4700 }

\Glsentryuserii
4701 \newrobustcmd*{\Glsentryuserii}[1]{%
4702   \gls@entry@field{#1}{userii}%
4703 }

\glsentryuseriii Get the third user key (as specified by the user3 when the entry was defined). The argument
is the label associated with the entry.
4704 \newcommand*{\glsentryuseriii}[1]{%
4705   \gls@entry@field{#1}{useriii}%
4706 }

\Glsentryuseriii
4707 \newrobustcmd*{\Glsentryuseriii}[1]{%
4708   \gls@entry@field{#1}{useriii}%
4709 }

\glsentryuseriv Get the fourth user key (as specified by the user4 when the entry was defined). The argument
is the label associated with the entry.
4710 \newcommand*{\glsentryuseriv}[1]{%
4711   \gls@entry@field{#1}{useriv}%
4712 }

\Glsentryuseriv
4713 \newrobustcmd*{\Glsentryuseriv}[1]{%
4714   \gls@entry@field{#1}{useriv}%
4715 }

\glsentryuserv Get the fifth user key (as specified by the user5 when the entry was defined). The argument is
the label associated with the entry.
4716 \newcommand*{\glsentryuserv}[1]{%
4717   \gls@entry@field{#1}{userv}%
4718 }

```

```

\Glsentryuserv
4719 \newrobustcmd*\{\Glsentryuserv\}[1]{%
4720   \Gls@entry@field{#1}{userv}%
4721 }

\glsentryuservi Get the sixth user key (as specified by the user6 when the entry was defined). The argument is the label associated with the entry.
4722 \newcommand*\{\glsentryuservi\}[1]{%
4723   \gls@entry@field{#1}{uservi}%
4724 }

\Glsentryuservi
4725 \newrobustcmd*\{\Glsentryuservi\}[1]{%
4726   \Gls@entry@field{#1}{uservi}%
4727 }

\glsentryshort Get the short key (as specified by the short the entry was defined). The argument is the label associated with the entry.
4728 \newcommand*\{\glsentryshort\}[1]{\gls@entry@field{#1}{short}{}}

\Glsentryshort
4729 \newrobustcmd*\{\Glsentryshort\}[1]{%
4730   \Gls@entry@field{#1}{short}%
4731 }

\glsentryshortpl Get the short plural key (as specified by the shortplural the entry was defined). The argument is the label associated with the entry.
4732 \newcommand*\{\glsentryshortpl\}[1]{\gls@entry@field{#1}{shortpl}{}}

\Glsentryshortpl
4733 \newrobustcmd*\{\Glsentryshortpl\}[1]{%
4734   \Gls@entry@field{#1}{shortpl}%
4735 }

\glsentrylong Get the long key (as specified by the long the entry was defined). The argument is the label associated with the entry.
4736 \newcommand*\{\glsentrylong\}[1]{\gls@entry@field{#1}{long}{}}

\Glsentrylong
4737 \newrobustcmd*\{\Glsentrylong\}[1]{%
4738   \Gls@entry@field{#1}{long}%
4739 }

\glsentrylongpl Get the long plural key (as specified by the longplural the entry was defined). The argument is the label associated with the entry.
4740 \newcommand*\{\glsentrylongpl\}[1]{\gls@entry@field{#1}{longpl}{}}

```

```

\Glsentrylongpl
4741 \newrobustcmd*\{\Glsentrylongpl\}[1]{%
4742   \Gls@entry@field{#1}{longpl}%
4743 }

Short cut macros to access full form:

\glsentryfull
4744 \newcommand*\{\glsentryfull\}[1]{%
4745   \acrfullformat{\Glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%
4746 }

\Glsentryfull
4747 \newrobustcmd*\{\Glsentryfull\}[1]{%
4748   \acrfullformat{\Glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%
4749 }

\glsentryfullpl
4750 \newcommand*\{\glsentryfullpl\}[1]{%
4751   \acrfullformat{\Glsentrylongpl{#1}}{\acronymfont{\glsentryshortpl{#1}}}%
4752 }

\Glsentryfullpl
4753 \newrobustcmd*\{\Glsentryfullpl\}[1]{%
4754   \acrfullformat{\Glsentrylongpl{#1}}{\acronymfont{\glsentryshortpl{#1}}}%
4755 }

entrynumberlist Displays the number list as is.
4756 \newcommand*\{\glsentrynumberlist\}[1]{%
4757   \glsdoifexists{#1}%
4758   {%
4759     \Gls@entry@field{#1}{numberlist}%
4760   }%
4761 }

splaynumberlist Formats the number list for the given entry label. Doesn't work with hyperref.
4762 \@ifpackageloaded{hyperref} {%
4763   \newcommand*\{\glsdisplaynumberlist\}[1]{%
4764     \GlossariesWarning
4765     {%
4766       \string\glsdisplaynumberlist\space
4767       doesn't work with hyperref.^^JUsing
4768       \string\glsentrynumberlist\space instead%
4769     }%
4770     \glsentrynumberlist{#1}%
4771   }%
4772 }%
4773 {%

```

```

4774 \newcommand*{\glsdisplaynumberlist}[1]{%
4775   \glsdoifexists{#1}%
4776   {%
4777     \bgroup
4778       \edef\@glo@label{\glsdetoklabel{#1}}%
4779       \let\@org@glsnumberformat\glsnumberformat
4780       \def\glsnumberformat##1{##1}%
4781       \protected@edef\the@numberlist{%
4782         \csname glo@\@glo@label @numberlist\endcsname}%
4783       \def\@gls@numlist@sep{}%
4784       \def\@gls@numlist@nextsep{}%
4785       \def\@gls@numlist@lastsep{}%
4786       \def\@gls@thislist{}%
4787       \def\@gls@donext@def{}%
4788       \renewcommand\do[1]{%
4789         \protected@edef\@gls@thislist{%
4790           \@gls@thislist
4791           \noexpand\@gls@numlist@sep
4792             ##1%
4793         }%
4794         \let\@gls@numlist@sep\@gls@numlist@nextsep
4795         \def\@gls@numlist@nextsep{\glsnumlistsep}%
4796         \gls@donext@def
4797         \def\@gls@donext@def{%
4798           \def\@gls@numlist@lastsep{\glsnumlistlastsep}%
4799         }%
4800       }%
4801       \expandafter \glsnumlistparser \expandafter{\the@numberlist}%
4802       \let\@gls@numlist@sep\@gls@numlist@lastsep
4803       \gls@thislist
4804     \egroup
4805   }%
4806 }
4807 }

\glsnumlistsep
4808 \newcommand*{\glsnumlistsep}{, }

snumlistlastsep
4809 \newcommand*{\glsnumlistlastsep}{ \& }


```

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

```

4810 \newcommand*{\glshyperlink}[2][\glsentrytext{\@glo@label}]{%
4811   \def\@glo@label{#2}%
4812   \glslink{\glolinkprefix\glsdetoklabel{#2}}{#1}

```

## 1.12 Adding an entry to the glossary without generating text

The following keys are provided for \glsadd and \glsaddall:

```
4813 \define@key{glossadd}{counter}{\def\@gls@counter{\#1}}
4814 \define@key{glossadd}{format}{\def\@glsnumberformat{\#1}}
```

This key is only used by \glsaddall:

```
4815 \define@key{glossadd}{types}{\def\@glo@type{\#1}}
```

```
\glsadd[<options>]{label}
```

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

\glsadd

```
4816 \newrobustcmd*\glsadd[2][]{%
```

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

```
4817   \@gls@adjustmode
4818   \glsdoifexists{\#2}%
4819   {%
4820     \def\@glsnumberformat{\glsnumberformat}%
4821     \edef\@gls@counter{\csname glo@\glsdetoklabel{\#2}@counter\endcsname}%
4822     \setkeys{glossadd}{#1}%

```

Store the entry's counter in \theglsentrycounter

```
4823   \@gls@saveentrycounter
```

Define sort key if necessary:

```
4824   \@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.

```
4825   \@@do@wrglossary{\#2}%
4826 }%
4827 }
```

@gls@adjustmode

```
4828 \newcommand*\@gls@adjustmode(){}
4829 \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
4830 \newrobustcmd*\{\glsaddall\}[1] []{%
4831   \edef\@glo@type{\@glo@types}%
4832   \setkeys{glossadd}{#1}%
4833   \forallglsentries[\@glo@type]{\@glo@entry}{%
4834     \glsadd[#1]{\@glo@entry}%
4835   }%
4836 }

```

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.

```

4837 \newrobustcmd*\{\glsaddallunused\}[1][\@glo@types]{%
4838   \forallglsentries[#1]{\@glo@entry}%
4839   {%
4840     \ifglsused{\@glo@entry}{}{\glsadd[format=glsignore]{\@glo@entry}}%
4841   }%
4842 }

```

\glsignore  
4843 \newcommand\*\{\glsignore\}[1]{}

## 1.13 Creating associated files

The `\writeist` command creates the associated customized `.ist` makeindex style file. While defining this command, some characters have their catcodes temporarily changed to ensure they get written to the `.ist` file correctly. The `makeindex` actual character (usually `\@`) is redefined to be a `?`, to allow internal commands to be written to the glossary file output file.

The special characters are stored in `\@gls@actualchar`, `\@gls@encapchar`, `\@gls@levelchar` and `\@gls@quotechar` to make them easier to use later, but don't change these values, because the characters are encoded in the command definitions that are used to escape the special characters (which means that the user no longer needs to worry about `makeindex` special characters).

The symbols and numbers label for group headings are hardwired into the `.ist` file as `glssymbols` and `glsnumbers`, the group titles can be translated (so that `\glssymbols{groupname}` replaces `glssymbols` and `\glsnumbers{groupname}` replaces `glsnumbers`) using the command `\glsgetgrouptitle` which is defined in `.`. This is done to prevent any problem characters in `\glssymbols{groupname}` and `\glsnumbers{groupname}` from breaking hyperlinks.

\glsopenbrace Define `\glsopenbrace` to make it easier to write an opening brace to a file.

```
4844 \edef\glsopenbrace{\expandafter\@gobble\string\{}
```

```

\glsclosebrace Define \glsclosebrace to make it easier to write an opening brace to a file.
4845 \edef\glsclosebrace{\expandafter\@gobble\string\{}}

\glsbackslash Define \glsbackslash to make it easier to write a backslash to a file.
4846 \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.
4847 \edef\glsquote#1{\string"#1\string"}

\glspercentchar Define \glspercentchar to make it easier to write a percent character to a file.
4848 \edef\glspercentchar{\expandafter\@gobble\string\%}

\glstildechar Define \glstildechar to make it easier to write a tilde character to a file.
4849 \edef\glstildechar{\string~}

@glsfirstletter Define the first letter to come after the digits 0,...,9. Only required for xindy.
4850 \ifglsxindy
4851   \newcommand*{\@glsfirstletter}{A}
4852 \fi

tterAfterDigits Sets the first letter to come after the digits 0,...,9. The starred version sanitizes.
4853 \newcommand*{\GlsSetXdyFirstLetterAfterDigits}{%
4854   \@ifstar\s@GlsSetXdyFirstLetterAfterDigits\@GlsSetXdyFirstLetterAfterDigits}
4855 \ifglsxindy
4856   \newcommand*{\@GlsSetXdyFirstLetterAfterDigits}[1]{%
4857     \renewcommand*{\@glsfirstletter}{#1}}
4858   \newcommand*{\s@GlsSetXdyFirstLetterAfterDigits}[1]{%
4859     \renewcommand*{\@glsfirstletter}{#1}%
4860     \onelevel@sanitize\@glsfirstletter
4861   }
4862 \else
4863   \newcommand*{\@GlsSetXdyFirstLetterAfterDigits}[1]{%
4864     \glsnoxindywarning\GlsSetXdyFirstLetterAfterDigits}
4865   \newcommand*{\s@GlsSetXdyFirstLetterAfterDigits}{%
4866     \GlsSetXdyFirstLetterAfterDigits
4867   }
4868 \fi

umbergrouporder Specifies the order of the number group.
4869 \ifglsxindy
4870   \newcommand*{\xdynumbergrouporder}{:before \string"\@glsfirstletter\string"}
4871 \fi

umberGroupOrder Sets the relative location of the number group. The starred version sanitizes.
4872 \newcommand*{\GlsSetXdyNumberGroupOrder}[1]{%
4873   \@ifstar\s@GlsSetXdyNumberGroupOrder\@GlsSetXdyNumberGroupOrder
4874 }

```

```

4875 \ifglsxindy
4876   \newcommand*{\@GlsSetXdyNumberGroupOrder}[1]{%
4877     \renewcommand*{\@xdynumbergrouporder}{#1}%
4878   }
4879   \newcommand*{\s@GlsSetXdyNumberGroupOrder}[1]{%
4880     \renewcommand*{\@xdynumbergrouporder}{#1}%
4881     \onelevel@sanitize\@xdynumbergrouporder
4882   }
4883 \else
4884   \newcommand*{\@GlsSetXdyNumberGroupOrder}[1]{%
4885     \glsnoxindywarning\GlsSetXdyNumberGroupOrder}
4886   \newcommand*{\s@GlsSetXdyNumberGroupOrder}{%
4887     \@GlsSetXdyNumberGroupOrder}
4888 \fi

```

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

```
4889 \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.

```

4890 \ifglsxindy
4891   \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
4892     \renewcommand*{\@glsminrange}{#1}%
4893   }
4894   \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
4895     \glsnoxindywarning\GlsSetXdyMinRangeLength}
4896 \fi

```

\writeist

```
4897 \ifglsxindy
  Code to use if xindy is required.
```

```
4898 \def\writeist{%
  Define write register if not already defined
```

```
4899   \ifundefined{\glswrite}{\newwrite\glswrite}{}%
```

Update attributes list

```
4900   \gls@addpredefinedattributes
```

Open the file.

```
4901   \openout\glswrite=\listfilename
```

Write header comment at the start of the file

```
4902   \write\glswrite{;; xindy style file created by the glossaries
4903     package}%
4904   \write\glswrite{;; for document '\jobname' on
4905     \the\year-\the\month-\the\day}%
```

Specify the required styles

```
4906   \write\glswrite{^^J; required styles^^J}
```

```

4907  \@for\xdystyle:=\xdyrequiredstyles\do{%
4908      \ifx\xdystyle\empty
4909      \else
4910          \protected@write\glswrite{}{(require
4911              \string"\xdystyle.xdy\string")}%
4912      \fi
4913  }%

```

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

```

4914  \write\glswrite{^^J%
4915      ; list of allowed attributes (number formats)^^J}%
4916  \write\glswrite{(\define-attributes ((\xdyattributes))}%

```

Define any additional alphabets

```

4917  \write\glswrite{^^J; user defined alphabets^^J}%
4918  \write\glswrite{\xdyuseralphabets}%

```

Define location classes.

```

4919  \write\glswrite{^^J; location class definitions^^J}%

```

As from version 3.0, locations are now specified as {*Hprefix*} {*number*}, so need to add all possible combinations of location types.

```

4920  \@for@gls@classI:=\glsxdy@locationlist\do{%

```

Case where *Hprefix* is empty:

```

4921      \protected@write\glswrite{}{(\define-location-class
4922          \string"\gls@classI\string"^^J\space\space\space
4923          (
4924              :sep "{}{"
4925              \csname@glsxdy@Lclass@\gls@classI\endcsname\space
4926              :sep "}""
4927          )
4928          ^^J\space\space\space
4929          :min-range-length \glsminrange^^J%
4930      )
4931  }%

```

Nested iteration over all classes:

```

4932  {%
4933      \@for@gls@classII:=\glsxdy@locationlist\do{%
4934          \protected@write\glswrite{}{(\define-location-class
4935              \string"\gls@classII-\gls@classI\string"
4936              ^^J\space\space\space
4937              (
4938                  :sep "{}"
4939                  \csname@glsxdy@Lclass@\gls@classII\endcsname\space
4940                  :sep "}"{"
4941                  \csname@glsxdy@Lclass@\gls@classI\endcsname\space
4942                  :sep "}""
4943              )
4944              ^^J\space\space\space
4945              :min-range-length \glsminrange^^J%

```

```

4946      )
4947      }%
4948      }%
4949      }%
4950      }%

```

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

```

4951 \write\glswrite{^^J; user defined location classes}%
4952 \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.)

```

4953 \write\glswrite{^^J; define cross-reference class^^J}%
4954 \write\glswrite{(define-crossref-class \string"see"\string"
4955 :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.)

```

4956 \write\glswrite{(markup-crossref-list
4957   :class \string"see"\string"^^J\space\space\space
4958   :open \string"\string\glsseeformat\string"
4959   :close \string"{}"\string")}%

```

Provide hook to write extra material here (used by glossaries-extra to define a seealso class).

```

4960 \@xdycrossrefhook

```

List the order to sort the classes.

```

4961 \write\glswrite{^^J; define the order of the location classes}%
4962 \write\glswrite{(define-location-class-order
4963   (\@xdylocationclassorder))}%

```

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

```

4964 \write\glswrite{^^J; define the glossary markup^^J}%
4965 \write\glswrite{(markup-index^^J\space\space\space
4966   :open \string"\string
4967   \glossarysection[\string\glossarytoctitle]{\string
4968   \glossarytitle}\string}\string\glossarypreamble}%

```

Add all the xindy-only macro definitions (needed to prevent errors in the event that the user changes from xindy to makeindex)

```

4969 \@for@\this@ctr:=\@xdycounters\do{%
4970   {%
4971     \@for@\this@attr:=\@xdyattributelist\do{%
4972       \protected@write\glswrite{}{\string\providecommand*%
4973         \expandafter\string
4974         \csname glsX@\this@ctr X@\this@attr\endcsname[2]%
4975       {%

```

```

4976          \string\setentrycounter
4977              [\expandafter\@gobble\string\#1]{\@this@\ctr}%
4978          \expandafter\string
4979              \csname\@this@\attr\endcsname
4980                  {\expandafter\@gobble\string\#2}%
4981          }%
4982      }%
4983  }%
4984 }%
4985 }%

```

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

```

4986 \write\glswrite{%
4987     \string\begin
4988     {theglossary}\string\glossaryheader\glstildechar n\string" ^\J\space
4989     \space\space:close \string"\glspcentchar\glstildechar n\string"
4990         \end{theglossary}\string\glossarypostamble
4991         \glstildechar n\string" ^\J\space\space\space
4992         :tree)}%

```

Specify what to put between letter groups

```

4993 \write\glswrite{(\markup-letter-group-list
4994     :sep \string"\string\glsgroupskip\glstildechar n\string")}%

```

Specify what to put between entries

```

4995 \write\glswrite{(\markup-indexentry
4996     :open \string"\string\relax \string\glsresetentrylist
4997         \glstildechar n\string")}%

```

Specify how to format entries

```

4998 \write\glswrite{(\markup-locclass-list :open
4999     \string"\glsopenbrace\string\glossaryentrynumbers
5000     \glsopenbrace\string\relax\space \string"^\J\space\space\space
5001     :sep \string", \string"
5002     :close \string"\glsclosebrace\glsclosebrace\string")}%

```

Specify how to separate location numbers

```

5003 \write\glswrite{(\markup-locref-list
5004     :sep \string"\string\delimN\space\string")}%

```

Specify how to indicate location ranges

```

5005 \write\glswrite{(\markup-range
5006     :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.

```

5007 \Onelevel@sanitize\gls@suffixF
5008 \Onelevel@sanitize\gls@suffixFF
5009 \ifx\gls@suffixF\empty
5010 \else
5011     \write\glswrite{(\markup-range

```

```

5012      :close "\gls@suffixF" :length 1 :ignore-end}%
5013  \fi
5014  \ifx\gls@suffixFF\@empty
5015  \else
5016      \write\glswrite{(markup-range
5017          :close "\gls@suffixFF" :length 2 :ignore-end}%
5018  \fi

Specify how to format locations.

5019  \write\glswrite{^^J; define format to use for locations^^J}%
5020  \write\glswrite{\@xdylocref}%

Specify how to separate letter groups.

5021  \write\glswrite{^^J; define letter group list format^^J}%
5022  \write\glswrite{(markup-letter-group-list
5023      :sep \string"\string\glsgroupskip\glstildechar n\string")}%

Define letter group headings.

5024  \write\glswrite{^^J; letter group headings^^J}%
5025  \write\glswrite{(markup-letter-group
5026      :open-head \string"\string\glsgrouthead
5027      \glssopenbrace\string"^^J\space\space\space
5028      :close-head \string"\glsclosebrace\string")}%

Define additional letter groups.

5029  \write\glswrite{^^J; additional letter groups^^J}%
5030  \write\glswrite{\@xdylettergroups}%

Define additional sort rules

5031  \write\glswrite{^^J; additional sort rules^^J}%
5032  \write\glswrite{\@xdysortrules}%

Hook for any additional information:

5033  \gls@writeisthook

Close the style file

5034  \closeout\glswrite

Suppress any further calls.

5035  \let\writeist\relax
5036  }
5037 \else

Code to use if makeindex is required.

5038  \edef\gls@actualchar{\string?}
5039  \edef\gls@encapchar{\string!}
5040  \edef\gls@levelchar{\string!}
5041  \edef\gls@quotechar{\string"}%
5042  \let\GlsSetQuote\gls@nosetquote
5043  \def\writeist{\relax
5044  \ifundefined{\glswrite}{\newwrite\glswrite}{}{\relax
5045  \openout\glswrite=\istfilename
5046  \write\glswrite{\glspercentchar\space makeindex style file

```

```

5047     created by the glossaries package}
5048 \write\glswrite{\glspercentchar\space for document
5049   '\jobname' on \the\year-\the\month-\the\day}
5050 \write\glswrite{actual '@gls@actualchar'}
5051 \write\glswrite{encap '@gls@encapchar'}
5052 \write\glswrite{level '@gls@levelchar'}
5053 \write\glswrite{quote '@gls@quotechar'}
5054 \write\glswrite{keyword "string"\string\\glossaryentry\string"}
5055 \write\glswrite{preamble "string"\string\\glossarysection[\string
5056   \\glossarytoctitle]{\string\\glossarytitle}\string
5057   \\glossarypreamble\string\n\string\\begin{theglossary}\string
5058   \\glossaryheader\string\n\string"}
5059 \write\glswrite{postamble "string"\string\\%\\string\n\string
5060   \\end{theglossary}\string\\glossarypostamble\string\n
5061   \string"}
5062 \write\glswrite{group_skip "string"\string\\glsgroupskip\string\n
5063   \string"}
5064 \write\glswrite{item_0 "string"\string\\%\\string\n\string"}
5065 \write\glswrite{item_1 "string"\string\\%\\string\n\string"}
5066 \write\glswrite{item_2 "string"\string\\%\\string\n\string"}
5067 \write\glswrite{item_01 "string"\string\\%\\string\n\string"}
5068 \write\glswrite{item_x1
5069   "string"\string\\relax \string\\glsresetentrylist\string\n
5070   \string"}
5071 \write\glswrite{item_12 "string"\string\\%\\string\n\string"}
5072 \write\glswrite{item_x2
5073   "string"\string\\relax \string\\glsresetentrylist\string\n
5074   \string"}
5075 \write\glswrite{delim_0 "string"\string{\string
5076   \\glossaryentrynumbers\string{\string\\relax \string"}}
5077 \write\glswrite{delim_1 "string"\string{\string
5078   \\glossaryentrynumbers\string{\string\\relax \string"}}
5079 \write\glswrite{delim_2 "string"\string{\string
5080   \\glossaryentrynumbers\string{\string\\relax \string"}}
5081 \write\glswrite{delim_t "string"\string{}\\string{}\\string"}
5082 \write\glswrite{delim_n "string"\string\\delimN \string"}
5083 \write\glswrite{delim_r "string"\string\\delimR \string"}
5084 \write\glswrite{headings_flag 1}
5085 \write\glswrite{heading_prefix
5086   "string"\string\\glsgroupheading\string{\string"
5087 \write\glswrite{heading_suffix
5088   "string"\string{}\\string\\relax
5089   \string\\glsresetentrylist \string"}
5090 \write\glswrite{symhead_positive "string"\glossysymbols\string"}
5091 \write\glswrite{numhead_positive "string"\glossynumbers\string"}
5092 \write\glswrite{page_compositor "string"\glosscompositor\string"}
5093 @_gls@escbsdq@gls@suffixF
5094 @_gls@escbsdq@gls@suffixFF
5095 \ifx\gls@suffixF@\empty
```

```

5096     \else
5097         \write\glswrite{suffix_2p \string"\gls@suffixF\string"}
5098     \fi
5099     \ifx\gls@suffixFF\empty
5100     \else
5101         \write\glswrite{suffix_3p \string"\gls@suffixFF\string"}
5102     \fi

```

Hook for any additional information:

```
5103     \gls@writeisthook
```

Close the file and disable \writeist.

```

5104     \closeout\glswrite
5105     \let\writeist\relax
5106 }
5107 \fi

```

**SetWriteIstHook** Allow user to append information to the style file.

```

5108 \newcommand*{\GlsSetWriteIstHook}[1]{\renewcommand*{\gls@writeisthook}{#1}}
5109 \onlypremakeg{\GlsSetWriteIstHook}

```

**\gls@writeisthook**

```
5110 \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.

```

5111 \ifglsxindy
5112     \newcommand*{\GlsSetQuote}[1]{\glsnomakeindexwarning{\GlsSetQuote}}
5113     \newcommand*{\gls@nosetquote}[1]{\glsnomakeindexwarning{\GlsSetQuote}}
5114 \else
5115     \newcommand*{\GlsSetQuote}[1]{\edef\@gls@quotechar{\string#1}%

```

If German is in use, set the extra makeindex option so makeglossaries can pick it up.

```

5116     \ifpackageloaded{tracklang}%
5117     {%
5118         \IfTrackedLanguage{german}%
5119         {%
5120             \def\@gls@extramakeindexopts{-g}%
5121         }%
5122         {}%
5123     }%
5124     {}%

```

Need to redefine \gls@checkquote

```

5125     \edef\@gls@docheckquotedef{%
5126         \noexpand\def\noexpand\@gls@checkquote####1#1####2#1####3\noexpand\null{%
5127             \noexpand\@gls@tmpb=\noexpand\expandafter{\noexpand\@gls@checkedmkidx}%
5128             \noexpand\toks@={####1}%
5129             \noexpand\ifx\noexpand\null##2\noexpand\null
5130                 \noexpand\ifx\noexpand\null##3\noexpand\null

```

```

5131 \noexpand\edef\noexpand@gls@checkedmkidx{%
5132   \noexpand\the\noexpand@gls@tmpb\noexpand\the\noexpand\toks@}%
5133 \noexpand\def\noexpand@@gls@checkquote{\noexpand\relax}%
5134 \noexpand\else
5135   \noexpand\edef\noexpand@gls@checkedmkidx{%
5136     \noexpand\the\noexpand@gls@tmpb\noexpand\the\noexpand\toks@%
5137     \noexpand@gls@quotechar\noexpand@gls@quotechar}%
5138     \noexpand@gls@quotechar\noexpand@gls@quotechar}%
5139   \noexpand\def\noexpand@@gls@checkquote{%
5140     \noexpand@gls@checkquote####3\noexpand\null}%
5141   \noexpand\fi
5142 \noexpand\else
5143   \noexpand\edef\noexpand@gls@checkedmkidx{%
5144     \noexpand\the\noexpand@gls@tmpb\noexpand\the\noexpand\toks@%
5145     \noexpand@gls@quotechar\noexpand@gls@quotechar}%
5146   \noexpand\ifx\noexpand\null####3\noexpand\null
5147     \noexpand\def\noexpand@@gls@checkquote{%
5148       \noexpand@gls@checkquote####2#1#1\noexpand\null}%
5149   \noexpand\else
5150     \noexpand\def\noexpand@gls@checkquote{%
5151       \noexpand@gls@checkquote####2#1####3\noexpand\null}%
5152     \noexpand\fi
5153   \noexpand\fi
5154   \noexpand@@gls@checkquote
5155 }%
5156 }%
5157 \gls@docheckquotedef
5158 \edef@gls@docheckquotedef{%
5159   \noexpand\renewcommand{\noexpand@gls@checkmkidxchars}[1]{%
5160     \noexpand\def\noexpand@gls@checkedmkidx{}%
5161     \noexpand\expandafter\noexpand@gls@checkquote####1\noexpand@nil
5162     #1#1\noexpand\null
5163     \noexpand\expandafter\noexpand@gls@updatechecked
5164     \noexpand@gls@checkedmkidx{####1}%
5165     \noexpand\def\noexpand@gls@checkedmkidx{}%
5166     \noexpand\expandafter\noexpand@gls@checkescquote####1\noexpand@nil
5167     \expandonce{\csname#1\endcsname}\expandonce{\csname#1\endcsname}%
5168     \noexpand\null
5169     \noexpand\expandafter\noexpand@gls@updatechecked
5170     \noexpand@gls@checkedmkidx{####1}%
5171     \noexpand\def\noexpand@gls@checkedmkidx{}%
5172     \noexpand\expandafter\noexpand@gls@checkescactual####1\noexpand@nil
5173     \noexpand?\noexpand?\noexpand\null
5174     \noexpand\expandafter\noexpand@gls@updatechecked
5175     \noexpand@gls@checkedmkidx{####1}%
5176     \noexpand\def\noexpand@gls@checkedmkidx{}%
5177     \noexpand\expandafter\noexpand@gls@checkactual####1\noexpand@nil
5178     \noexpand?\noexpand?\noexpand\null
5179     \noexpand\expandafter\noexpand@gls@updatechecked

```

```

5180      \noexpand\@gls@checkedmkidx{####1}%
5181      \noexpand\def\noexpand\@gls@checkedmkidx{}%
5182      \noexpand\expandafter\noexpand\@gls@checkbar####1\noexpand\@nil
5183          \noexpand|\noexpand|\noexpand\@nil
5184      \noexpand\expandafter\noexpand\@gls@updatechecked
5185          \noexpand\@gls@checkedmkidx{####1}%
5186      \noexpand\def\noexpand\@gls@checkedmkidx{}%
5187      \noexpand\expandafter\noexpand\@gls@checkescbar####1\noexpand\@nil
5188          \noexpand|\noexpand|\noexpand\@nil
5189      \noexpand\expandafter\noexpand\@gls@updatechecked
5190          \noexpand\@gls@checkedmkidx{####1}%
5191      \noexpand\def\noexpand\@gls@checkedmkidx{}%
5192      \noexpand\expandafter\noexpand\@gls@checklevel####1\noexpand\@nil
5193          \noexpand!\noexpand!\noexpand\@nil
5194      \noexpand\expandafter\noexpand\@gls@updatechecked
5195          \noexpand\@gls@checkedmkidx{####1}%
5196      }%
5197  }%
5198 \@gls@docheckquotedef
5199 \edef\@gls@docheckquotedef{%
5200     \noexpand\def\noexpand\@gls@checkescquote####1%
5201         \expandonce{\csname#1\endcsname}####2\expandonce{\csname#1\endcsname}%
5202             ####3\noexpand\@nil{%
5203             \noexpand\@gls@tmpb=\noexpand\expandafter{\noexpand\@gls@checkedmkidx}%
5204                 \noexpand\toks@={####1}%
5205                 \noexpand\ifx\noexpand\@nil####2\noexpand\@nil
5206                     \noexpand\ifx\noexpand\@nil####3\noexpand\@nil
5207                         \noexpand\edef\noexpand\@gls@checkedmkidx{%
5208                             \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
5209                         \noexpand\def\noexpand\@gls@checkescquote{\noexpand\relax}%
5210                         \noexpand\else
5211                             \noexpand\edef\noexpand\@gls@checkedmkidx{%
5212                                 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@%
5213                                     \noexpand\@gls@quotechar\noexpand\string\expandonce{%
5214                                         \csname#1\endcsname}\noexpand\@gls@quotechar
5215                                         \noexpand\@gls@quotechar\noexpand\string\expandonce{%
5216                                             \csname#1\endcsname}\noexpand\@gls@quotechar}%
5217                                         \noexpand\def\noexpand\@gls@checkescquote{%
5218                                             \noexpand\@gls@checkescquote####3\noexpand\@nil}%
5219                                         \noexpand\fi
5220                                         \noexpand\else
5221                                         \noexpand\edef\noexpand\@gls@checkedmkidx{%
5222                                             \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@%
5223                                                 \noexpand\@gls@quotechar\noexpand\string
5224                                                 \expandonce{\csname#1\endcsname}\noexpand\@gls@quotechar}%
5225                                         \noexpand\ifx\noexpand\@nil####3\noexpand\@nil
5226                                         \noexpand\def\noexpand\@gls@checkescquote{%
5227                                             \noexpand\@gls@checkescquote####2\expandonce{\csname#1\endcsname}%
5228                                             \expandonce{\csname#1\endcsname}\noexpand\@nil}%

```

```

5229     \noexpand\else
5230         \noexpand\def\noexpand\@gls@checkescquote{%
5231             \noexpand\@gls@checkescquote####2\expandonce{\csname#1\endcsname}%
5232             #####3\noexpand\null}%
5233         \noexpand\fi
5234     \noexpand\fi
5235     \noexpand\@gls@checkescquote
5236     }%
5237 }%
5238 \gls@docheckquotedef
5239 }
5240 \newcommand*{\gls@nosetquote}[1]{\PackageError{glossaries}%
5241   {\string\GlsSetQuote\space not permitted here}%
5242   {Move \string\GlsSetQuote\space earlier in the preamble, as
5243    soon as possible after glossaries.sty has been loaded}}
5244 \fi

```

#### ramakeindexopts

```
5245 \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
5246 \newcommand{\noist}{%
  Update attributes list
5247   \gls@addpredefinedattributes
5248   \let\writeist\relax
5249 }
```

`\@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 (with just the base `glossaries` package). You either need to have a `\makeglossaries` 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`. `glossaries-extra` allows for a hybrid approach.

`\@makeglossary` Unstarred form of `\ifglossaryexists` is used as `\@makeglossary` can't be used with an ignored glossary.

```
5250 \newcommand*{\@makeglossary}[1]{%
5251   \ifglossaryexists{#1}%
5252     {%
```

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

```
5253 \ifglssavewrites
5254   \expandafter\newtoks\csname glo@#1@filetok\endcsname
5255 \else
5256   \expandafter\newwrite\csname glo@#1@file\endcsname
5257   \expandafter\@glsopenfile\csname glo@#1@file\endcsname{#1}%
5258 \fi
5259 \@gls@renewglossary
5260 \writeisit
5261 }%
5262 {%
5263   \PackageError{glossaries}%
5264   {Glossary type ‘#1’ not defined}%
5265   {New glossaries must be defined before using \string\makeglossaries}%
5266 }%
5267 }
```

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

```
5268 \newcommand*{\@glsopenfile}[2]{%
5269   \immediate\openout#1=\jobname.\csname @gloctype@#2@out\endcsname
5270   \PackageInfo{glossaries}{Writing glossary file
5271     \jobname.\csname @gloctype@#2@out\endcsname}%
5272 }
```

\@closegls

```
5273 \newcommand*{\@closegls}[1]{%
5274   \closeout\csname glo@#1@file\endcsname
5275 }
```

\@gls@automake Unstarred form of \ifglossaryexists is used as \@gls@automake can't be used with an ignored glossary.

```
5276 \ifglsxindy
5277 \newcommand*{\@gls@automake}[1]{%
5278   \ifglossaryexists{#1}
5279   {%
5280     \@closegls{#1}%
5281     \ifdefstring{\glsorder}{letter}{%
5282       {\def\@gls@order{-M ord/letorder }}%
5283       {\let\@gls@order\empty}%
5284     \ifcsondef{\xdy@#1@language}{%
5285       {\let\@gls@langmod\xdy@main@language}%
5286       {\let\@gls@langmod{\xdy@#1@language}}%
5287     \edef\@gls@dothiswrite{\noexpand\write18{xindy
5288       -I xindy
5289       \@gls@order
5290       -L \@gls@langmod\space
5291       -M \gls@istfilebase\space}
```

```

5292     -C \gls@codepage\space
5293     -t \jobname.\csuse{@glotype@#1@log}
5294     -o \jobname.\csuse{@glotype@#1@in}
5295     \jobname.\csuse{@glotype@#1@out}}}%
5296   }%
5297   \gls@dothiswrite
5298 }%
5299 {%
5300   \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
5301 }%
5302 }
5303 \else
5304 \newcommand*{\gls@automake}[1]{%
5305   \ifglossaryexists{#1}
5306   {%
5307     \closegls{#1}%
5308     \ifdefstring{\glsorder}{letter}%
5309       {\def\gls@order{-1}}%
5310       {\let\gls@order\empty}%
5311     \edef\gls@dothiswrite{\noexpand\write18{makeindex \gls@order
5312       -s \listfilename\space
5313       -t \jobname.\csuse{@glotype@#1@log}
5314       -o \jobname.\csuse{@glotype@#1@in}
5315       \jobname.\csuse{@glotype@#1@out}}}%
5316   }%
5317   \gls@dothiswrite
5318 }%
5319 {%
5320   \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
5321 }%
5322 }
5323 \fi

```

omake@immediate Unstarred form of \ifglossaryexists is used as \gls@automake@immediate can't be used with an ignored glossary.

```

5324 \ifglsxindy
5325 \newcommand*{\gls@automake@immediate}[1]{%
5326   \ifglossaryexists{#1}
5327   {%
5328     \IfFileExists{\jobname.\csuse{@glotype@#1@out}}{%
5329       {%
5330         \ifdefstring{\glsorder}{letter}%
5331           {\def\gls@order{-M ord/letorder}}%
5332           {\let\gls@order\empty}%
5333         \ifcsondef{\xdy@#1@language}{%
5334           {\let\gls@langmod\xdy@main@language}%
5335           {\let\csondef{\xdy@#1@language}}%
5336         \edef\gls@dothiswrite{\noexpand\immediate\noexpand\write18{xindy
5337           -I xindy}

```

```

5338     \gls@order
5339     -L \gls@langmod\space
5340     -M \gls@istfilebase\space
5341     -C \gls@codepage\space
5342     -t \jobname.\csuse{@glotype@#1@log}
5343     -o \jobname.\csuse{@glotype@#1@in}
5344     \jobname.\csuse{@glotype@#1@out}}%
5345   }%
5346   \gls@dothiswrite
5347 }%
5348 {\GlossariesWarning{can't automake '#1': \jobname.\csuse{@glotype@#1@out}}
5349   doesn't exist. Rerun may be required}}%
5350 }%
5351 {%
5352   \GlossariesWarning{Can't make glossary '#1', it doesn't exist}}%
5353 }%
5354 }
5355 \else
5356 \newcommand*{\gls@automake@immediate}[1]{%
5357   \ifglossaryexists{#1}
5358   {%
5359     \IfFileExists{\jobname.\csuse{@glotype@#1@out}}{%
5360       {%
5361         \ifdefstring{\glsorder}{letter}{%
5362           {\def\gls@order{-1}}{%
5363             {\let\gls@order\empty}{%
5364               \edef\gls@dothiswrite{\noexpand\immediate\noexpand\write18{makeindex \gls@order
5365                 -s \listfilename\space
5366                 -t \jobname.\csuse{@glotype@#1@log}
5367                 -o \jobname.\csuse{@glotype@#1@in}
5368                 \jobname.\csuse{@glotype@#1@out}}}}%
5369         }%
5370         \gls@dothiswrite
5371       }%
5372       {\GlossariesWarning{can't automake '#1': \jobname.\csuse{@glotype@#1@out}}
5373         doesn't exist. Rerun may be required}}%
5374     }%
5375   {%
5376     \GlossariesWarning{Can't make glossary '#1', it doesn't exist}}%
5377   }%
5378 }
5379 \fi

```

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

```
5380 \newcommand*{\warn@nomakeglossaries}{}%
```

Only use this if warning if `\printglossary` has been used without `\makeglossaries`

```
5381 \newcommand*{\warn@nomakeglossaries}{\warn@nomakeglossaries}
```

```

omake@immediate
5382 \newcommand{\@gls@@automake@immediate}{%
5383   \ifnum\gls@automake@nr=2\relax
5384     \@for\@gls@type:=\@glo@types\do{%
5385       \ifdefempty{\@gls@type}{}{%
5386         {\@gls@automake@immediate{\@gls@type}}{}}%
5387     }%
5388     \glsautomakefalse
5389     \renewcommand*\@gls@doautomake{}{%
5390   \fi
5391 }

```

\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
5392 \newcommand*\@makeglossaries{}{%
5393   \@domakeglossaries
5394   {}{%
      If automake=immediate setting is on, use the shell escape now.
      5395   \@gls@@automake@immediate
      Define the write used for style file also used for all other output files if savewrites=true.
      5396   \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.
          5397   \protected@write\@auxout{}{\string\providecommand\string@glsorder[1]{}}
          5398   \protected@write\@auxout{}{\string\providecommand\string@istfilename[1]{}}
          If \@@gls@extramakeindexopts has been defined, write it:
          5399   \ifundef\@@gls@extramakeindexopts
          5400   {}{%
          5401   {}{%
          5402     \protected@write\@auxout{}{\string\providecommand
          5403       \string@gls@extramakeindexopts[1]{}}
          5404     \protected@write\@auxout{}{\string@gls@extramakeindexopts
          5405       {\@@gls@extramakeindexopts}}{}}%
          5406   }%
          Write the name of the style file to the aux file (needed by makeglossaries)
          5407   \protected@write\@auxout{}{\string@istfilename{\istfilename}}{%
          5408   \protected@write\@auxout{}{\string@glsorder{\glsorder}}{%

```

Iterate through each glossary type and activate it.

```

5409   \@for\@glo@type:=\@glo@types\do{%
5410     \ifthenelse{\equal{\@glo@type}{}{}}{%
5411       \makeglossary{\@glo@type}}{%
5412     }%

```

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

```
5413 \renewcommand*\newglossary[4] []{%
5414 \PackageError{glossaries}{New glossaries
5415 must be created before \string\makeglossaries}{You need
5416 to move \string\makeglossaries\space after all your
5417 \string\newglossary\space commands}}%
```

Any subsequent instances of this command should have no effect. The deprecated \makeglossary is not redefined here as it either implements \makeglossaries or has been restored to its original definition (in which case it shouldn't be changed).

```
5418 \let\@makeglossary\@gobble
5419 \let\makeglossaries\relax
```

Disable all commands that have no effect after \makeglossaries

```
5420 \c@disable@onlypremakeg
```

Allow see key:

```
5421 \let\gls@checkseeallowed\relax
```

Suppress warning about no \makeglossaries

```
5422 \let\warn@nomakeglossaries\relax
```

Activate warning about missing \printglossary

```
5423 \def\warn@noprintglossary{%
5424 \ifdefstring{\@glo@types}{,}{%
5425 {%
5426 \GlossariesWarningNoLine{No glossaries have been defined}%
5427 }%
5428 {%
5429 \GlossariesWarningNoLine{No \string\printglossary\space
5430 or \string\printglossaries\space
5431 found. ^J(Remove \string\makeglossaries\space if you
5432 don't want any glossaries.) ^JThis document will not
5433 have a glossary}%
5434 }%
5435 }%
```

Declare list parser for \glsdisplaynumberlist

```
5436 \ifglssavenumberlist
5437 \edef\@gls@dodeflistparser{\noexpand\DeclareListParser
5438 {\noexpand\glsnumlistparser}{\delimN}}%
5439 \@gls@dodeflistparser
5440 \fi
```

Prevent user from also using \makenoidxglossaries

```
5441 \let\makenoidxglossaries\@no@makeglossaries
```

Prohibit sort key in printgloss family:

```
5442 \renewcommand*{\@printgloss@setsort}{%
5443 \let\@glo@assign@sortkey\@glo@no@assign@sortkey
5444 }%
```

Check the automake setting:

```
5445 \ifglsautomake
5446   \renewcommand*\@gls@doautomake}{%
5447     \@for\@gls@type:=\@glo@types\do{%
5448       \ifdefempty{\@gls@type}{}{%
5449         {\@gls@automake{\@gls@type}}{%
5450       }%
5451     }%
5452   }\fi
```

Check the sort setting:

```
5453 \@glo@check@sortallowed\makeglossaries
5454 }%
5455 }
```

Must occur in the preamble:

```
5456 \@onlypreamble{\makeglossaries}
```

\glswrite The definition of \glswrite has now been moved to \makeglossaries so that it's only defined if needed.

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

```
5457 \AtEndDocument{%
5458   \warn@nomakeglossaries
5459   \warn@noprintglossary
5460 }
```

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

```
5461 \newcommand*{\makenoidxglossaries}{%
5462   \@domakeglossaries
5463 }
```

Redefine empty glossary warning:

```
5464 \renewcommand{\@gls@noref@warn}[1]{%
5465   \GlossariesWarning{Empty glossary for
5466   \string\printnoidxglossary[type={##1}].
5467   Rerun may be required (or you may have forgotten to use
5468   commands like \string\gls)}%
5469 }
```

Don't escape makeindex/xindy characters:

```
5470 \let\@gls@checkmkidxchars\@gobble
```

Don't escape locations:

```
5471 \glsesclocationsfalse
```

Write glossary information to aux instead of glossary files

```
5472 \let\@@do@@wrglossary\gls@noidxglossary
```

Switch on group headings that use the character code:

```
5473 \let\@gls@getgroupitle\@gls@noidx@getgroupitle
```

Allow see key:

```
5474 \let\gls@checkseeallowed\relax
```

Redefine cross-referencing macro:

```
5475 \renewcommand{\do@seeglossary}[2]{%
5476   \edef\gls@label{\glsdetoklabel{\#1}}%
5477   \protected@write\auxout{}{%
5478     \string\gls@reference
5479     {\csname glo@\gls@label\endcsname}%
5480     {\gls@label}%
5481     {%
5482       \string\glsseeformat##2{}%
5483     }%
5484   }%
5485 }
```

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.

```
5486 \AtBeginDocument
5487 {%
5488   \write\auxout{\string\providecommand\string\gls@reference[3]{}}
5489 }%
```

Change warning about no glossaries

```
5490 \def\warn@noprintglossary{%
5491   \GlossariesWarningNoLine{No \string\printnoidxglossary\space
5492   or \string\printnoidxglossaries ~~J
5493   found. (Remove \string\makenoidxglossaries\space if you
5494   don't want any glossaries.)~~JThis document will not have a glossary}%
5495 }
```

Suppress warning about no \makeglossaries

```
5496 \let\warn@nomakeglossaries\relax
```

Prevent user from also using \makeglossaries

```
5497 \let\makeglossaries\@no@makeglossaries
```

Allow sort key in printgloss family:

```
5498 \renewcommand*\@printgloss@setsort{%
5499   \let\glo@assign@sortkey\@glo@assign@sortkey
```

Initialise default sort order:

```
5500 \def\glo@sorttype{\glo@default@sorttype}%
5501 }
```

All entries must be defined in the preamble:

```
5502 \renewcommand*\new@glossaryentry[2]{%
5503   \PackageError{glossaries}{Glossary entries must be
5504   defined in the preamble~~Jwhen you use
5505   \string\makenoidxglossaries}%
5506   {Either move your definitions to the preamble or use
5507   \string\makeglossaries}%
5508 }
```

```

Redefine \glsentrynumberlist
5509  \renewcommand*{\glsentrynumberlist}[1]{%
5510    \letcs{\@gls@loclist}{\glo@\glsdetoklabel{##1}@loclist}%
5511    \ifdef{\gls@loclist}
5512    {%
5513      \glsnoidxloclist{\@gls@loclist}%
5514    }%
5515    {%
5516      ??\glsdoifexists{##1}%
5517      {%
5518        \GlossariesWarning{Missing location list for ‘##1’. Either
5519          a rerun is required or you haven’t referenced the entry}%
5520      }%
5521    }%
5522  }%

```

Redefine \glsdisplaynumberlist

```

5523  \renewcommand*{\glsdisplaynumberlist}[1]{%
5524    \letcs{\@gls@loclist}{\glo@\glsdetoklabel{##1}@loclist}%
5525    \ifdef{\gls@loclist}
5526    {%
5527      \def{\gls@noidxloclist@sep}{%
5528        \def{\gls@noidxloclist@sep}{%
5529          \def{\gls@noidxloclist@sep}{%
5530            \glsnumlistsep
5531          }%
5532          \def{\gls@noidxloclist@finalsep}{\glsnumlistlastsep}%
5533        }%
5534      }%
5535      \def{\gls@noidxloclist@finalsep}{%
5536        \def{\gls@noidxloclist@prev}{%
5537          \forlistloop{\glsnoidxdisplayloclisthandler}{\@gls@loclist}%
5538          \@gls@noidxloclist@finalsep
5539          \@gls@noidxloclist@prev
5540        }%
5541      }%
5542      ??\glsdoifexists{##1}%
5543      {%
5544        \GlossariesWarning{Missing location list for ‘##1’. Either
5545          a rerun is required or you haven’t referenced the entry}%
5546      }%
5547    }%
5548  }%

```

Provide a generic way of iterating through the number list:

```

5549  \renewcommand*{\glsnumberlistloop}[3]{%
5550    \letcs{\@gls@loclist}{\glo@\glsdetoklabel{##1}@loclist}%
5551    \let{\gls@org@glsnoidxdisplayloc}{\glsnoidxdisplayloc}
5552    \let{\gls@org@glsseefORMAT}{\glsseefORMAT}
5553    \let{\glsnoidxdisplayloc##2}{\relax}

```

```

5554 \let\glsseefORMAT##3\relax
5555 \ifdef@\gls@loclist
5556 {%
5557   \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
5558 }%
5559 {%
5560   ??\glsdoifexists{##1}%
5561   {%
5562     \GlossariesWarning{Missing location list for ‘##1’. Either
5563       a rerun is required or you haven’t referenced the entry}%
5564   }%
5565 }%
5566 \let\glsnoidxdisplayloc\@gls@org@glsnoidxdisplayloc
5567 \let\glsseefORMAT\@gls@org@glsseefORMAT
5568 }%

```

Modify sanitize sort function

```

5569 \let\@@gls@sanitizesort\@gls@noidx@sanitizesort
5570 \let\@@gls@nosanitizesort\@gls@noidx@nosanitizesort
5571 \@gls@noidx@setsanitizesort

```

Check sort option allowed.

```

5572 \@glo@check@sortallowed\makenoidxglossaries
5573 }%
5574 }

```

Preamble-only command:

```
5575 \@onlypreamble{\makenoidxglossaries}
```

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

```

5576 \newcommand*\glsnumberlistloop[2]{%
5577   \PackageError{glossaries}{\string\glsnumberlistloop\space
5578     only works with \string\makenoidxglossaries}{}%
5579 }

```

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

```

5580 \newcommand*\glsnoidxnumberlistloophandler[1]{%
5581   #1%
5582 }

```

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

```

5583 \newcommand*\@no@makeglossaries{}%
5584 \PackageError{glossaries}{You can’t use both
5585 \string\makeglossaries\space and \string\makenoidxglossaries}%
5586 {Either use one or other (or none) of those commands but not both
5587 together.}%
5588 }

```

```

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

5589 \newcommand{\@gls@noref@warn}[1]{%
5590   \GlossariesWarning{\string\maketoidxglossaries\space
5591     is required to make \string\printnoidxglossary[type=\#1] work}%
5592 }

```

## 1.14 Writing information to associated files

`s@noidxglossary` Write the glossary information to the aux file (for the ‘noidx’ method):

```

5593 \newcommand*\@gls@noidxglossary}{%
5594   \protected@write\auxout{}{%
5595     \string\@gls@reference
5596     {\csname glo@\@gls@label \type\endcsname}%
5597     {\@gls@label}%
5598     {\string\glsnoidxdisplayloc
5599       {\@glo@counterprefix}%
5600       {\@gls@counter}%
5601       {\@glsnumberformat}%
5602       {\@glslocref}%
5603     }%
5604   }%
5605 }

```

`\istfile` Deprecated.

```
5606 \providecommand\istfile{\glswrite}
```

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

```

5607 \AtEndDocument{%
5608   \glswritefiles
5609 }

```

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

```
5610 \newcommand*\@glswritefiles}{%
```

Iterate through all the glossaries.

```
5611 \forallglossaries{\@glo@type}{%
```

Check for empty glossaries (patch provided by Patrick Häcker)

```

5612   \ifcsundef{\glo@\@glo@type \filetok}{%
5613     {%
5614       \def\gls@tmp{}%
5615     }%
5616     {%
5617       \edef\gls@tmp{\expandafter\the
5618         \csname glo@\@glo@type \filetok\endcsname}%
5619     }%
5620     \ifx\gls@tmp\empty
5621       \ifx\@glo@type\glsdefaulttype

```

```

5622     \GlossariesWarning{Glossary '\@glo@type' has no
5623         entries.^^JRemember to use package option 'nomain' if
5624 you
5625         don't want to^^Juse the main glossary}%
5626 \else
5627     \GlossariesWarning{Glossary '\@glo@type' has no
5628         entries}%
5629 \fi
5630 \else
5631     \glsopenfile{\glswrite}{\@glo@type}%
5632     \immediate\write\glswrite{%
5633         \expandafter\the
5634         \csname glo@\@glo@type \filetok\endcsname}%
5635     \immediate\closeout\glswrite
5636 \fi
5637 }%
5638 }

```

As from v4.10, the `\glossary` command isn't used by the `glossaries` package. Since the user isn't expected to use this command (as `glossaries` takes care of the particular format required for `makeindex/xindy`) there's no need for a user level command. Using a custom internal command prevents any conflict with other packages (and with the `\mark` mechanism).

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

```

\gls@glossary
5639 \newcommand*{\gls@glossary}[1]{%
5640     \gls@glossary{#1}%
5641 }

```

`\@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 `\glspercentchar` would be needed with the sort sanitization switched off.) This means that it's safe to simply ignore the second argument.

```
5642 \newcommand*{\@gls@glossary}[2]{%
5643   \if@gls@debug
5644     \PackageInfo{glossaries}{wrglossary(#1)(#2)}%
5645   \fi
5646 }
```

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`

```
5647 \newcommand{\@gls@renewglossary}{%
5648   \gdef\@gls@glossary##1{\@bsphack\begingroup\gls@wrglossary{##1}}%
5649   \let\@gls@renewglossary\@empty
5650 }
```

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`

```
5651 \newcommand*{\gls@wrglossary}[2]{%
5652   \ifglsavewrites
5653     \protected@edef\@gls@tmp{\the\csname glo@#1@filetok\endcsname#2}%
5654     \expandafter\global\expandafter\csname glo@#1@filetok\endcsname
5655       \expandafter{\@gls@tmp^J}%
5656   \else
5657     \ifcscdef{glo@#1@file}%
5658       \%
5659         \expandafter\protected@write\csname glo@#1@file\endcsname{%
5660           \gls@disablepagerefexpansion}%
5661       \%
5662       \%
5663         \ifignoredglossary{#1}{}%
5664         \%
5665           \GlossariesWarning{No file defined for glossary '#1'}%
5666         \%
5667       \%
5668   \fi
5669   \endgroup\@esphack
5670 }
```

`\@do@wrglossary`

```
5671 \newcommand*{\@do@wrglossary}[1]{%
5672   \glswriteentry{#1}{\@do@wrglossary{#1}}%
5673 }
```

\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.

```
5674 \newcommand*\glswriteentry[2]{%
5675   \ifglsindexonlyfirst
5676     \ifglsused{#1}{}{#2}%
5677   \else
5678     #2%
5679   \fi
5680 }
```

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

```
5681 \newcommand{\gls@protected@pagefmts}{\gls@numberpage,\gls@alphpage,%
5682   \gls@Alphpage,\gls@romanpage,\gls@Romanpage,\gls@arabicpage}
```

\agerefexpansion

```
5683 \newcommand*\gls@disablepagerefexpansion{%
5684   \cfor@gls@this:=\gls@protected@pagefmts\do
5685   {%
5686     \expandafter\let\gls@this\relax
5687   }%
5688 }
```

\gls@alphpage

```
5689 \newcommand*\gls@alphpage{\@alph\c@page}
```

\gls@Alphpage

```
5690 \newcommand*\gls@Alphpage{\@Alpha\c@page}
```

\gls@numberpage

```
5691 \newcommand*\gls@numberpage{\number\c@page}
```

\gls@arabicpage

```
5692 \newcommand*\gls@arabicpage{\@arabic\c@page}
```

\gls@romanpage

```
5693 \newcommand*\gls@romanpage{\romannumeral\c@page}
```

\gls@Romanpage

```
5694 \newcommand*\gls@Romanpage{\@Roman\c@page}
```

\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 ( $\langle csname \rangle \c@page$  must be valid).

```

5695 \newcommand*{\glsaddprotectedpagefmt}[1]{%
5696   \eappto{\gls@protected@pagefmts}{\expandonce{\csname gls#1page\endcsname}}%
5697   \csedef{gls#1page}{\expandonce{\csname#1\endcsname}\noexpand\c@page}%
5698   \eappto{\@wrglossarynumberhook}{%
5699     \noexpand\let\expandonce{\csname org@gls#1\endcsname}%
5700     \expandonce{\csname#1\endcsname}%
5701     \noexpand\def\expandonce{\csname#1\endcsname}{%
5702       \noexpand\@wrglossary@pageformat
5703         \expandonce{\csname gls#1page\endcsname}%
5704         \expandonce{\csname org@gls#1\endcsname}%
5705     }%
5706   }%
5707 }

```

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

```

5708 \newcommand*{\@wrglossarynumberhook}{}

```

`sary@pageformat`

```

5709 \newcommand{\@wrglossary@pageformat}[3]{%
5710   \ifx#3\c@page #1\else #2#3\fi
5711 }

```

`@@do@wrglossary` Write the glossary entry in the appropriate format.

```

5712 \newcommand*{\@@do@wrglossary}[1]{%
5713   \ifglsesclocations
5714     \@@do@esc@wrglossary{#1}%
5715   \else
5716     \@@do@noesc@wrglossary{#1}%
5717   \fi
5718 }

```

`oesc@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.

```

5719 \newcommand*{\@@do@noesc@wrglossary}[1]{%

```

Don't fully expand yet.

```

5720   \expandafter\def\expandafter\@glslocref\expandafter{\the\glstentrycounter}%
5721   \expandafter\def\expandafter\@glsHlocref\expandafter{\the\Hglstentrycounter}%

```

Find the prefix if `\@glsHlocref` and `\@glslocref` aren't the same.

```

5722   \ifx\@glsHlocref\@glslocref
5723     \def\@glo@counterprefix{}%
5724   \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 `\the\glstentrycounter` and `\the\Hglstentrycounter` at this point, even if it hasn't been updated yet. The page number is not expected to occur in the prefix.)

```

5725   \protected@edef{\do@gls@getcounterprefix}{\noexpand@gls@getcounterprefix
5726     {\glslocref}{\glsHlocref}%
5727   }%
5728   \do@gls@getcounterprefix
5729 \fi
      De-tok label if required.
5730 \edef{\gls@label}{\glsdetoklabel{#1}}%
      Write the information to file:
5731 \@@do@wrglossary
5732 }

```

`owprimitivemods` Conditional to determine whether or not `\@@do@esc@wrglossary` should be allowed to temporarily redefine `\the` and `\number`.

```

5733 \newif\ifglswallowprimitivemods
5734 \glswallowprimitivemodstrue

```

`@esc@wrglossary` Write the glossary entry in the appropriate format. (Need to set `\glsnumberformat` and `\gls@counter` prior to use.) The argument is the entry's label. This is far more complicated with `xindy` than with other indexing methods. There are two necessary but conflicting requirements with `xindy`:

1. all backslashes in the location must be escaped;
2. `\c@page` can't be prematurely expanded.

(With `makeindex` there's the remote possibility that the page compositor is a `makeindex` special character, so that would also need to be escaped.)

For example, suppose `\thepage` is defined as

```
\renewcommand{\thepage}{\tally{page}}
\newcommand{\tally}[1]{\tallynum{\expandafter\the\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}{[\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 `\gls@swallowprimitivemodsfalse`) 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 `\gls@tallypage` 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 `\gls@hypernumber` to create a hyperlink for a given counter (like `\hyperpage` but for an arbitrary counter).

```
5735 \newcommand*{\@@do@esc@wrglossary}{[1]{% please read documented code!
5736   \begingroup
```

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

```
5737   \let\gls@orgthe\the
5738   \let\gls@orgnumber\number
5739   \let\gls@orgarabic@\arabic
5740   \let\gls@orgromannumeral\romannumeral
5741   \let\gls@orgalph@\alph
5742   \let\gls@orgAlpha@\Alpha
5743   \let\gls@orgRoman@\Roman
```

Redefine:

```
5744   \ifgls@swallowprimitivemods
```

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.

```
5745   \def\gls@the##1{%
5746     \ifx##1\c@page \gls@numberpage\else\gls@orgthe##1\fi}%
5747   \def\the{\expandafter\gls@the}%
5748   \def\gls@number##1{%
5749     \ifx##1\c@page \gls@numberpage\else\gls@orgnumber##1\fi}%
5750   \def\number{\expandafter\gls@number}%
5751   \fi
5752   \def\arabic##1{%
5753     \ifx##1\c@page \gls@arabicpage\else\gls@orgarabic##1\fi}%
5754   \def\romannumeral##1{%
```

```

5755     \ifx##1\c@page \gls@romanpage\else\gls@orgromannumeral##1\fi}%
5756     \def\@Roman##1{%
5757         \ifx##1\c@page \gls@Romanpage\else\gls@orgRoman##1\fi}%
5758     \def\@alph##1{%
5759         \ifx##1\c@page \gls@alphpage\else\gls@orgalph##1\fi}%
5760     \def\@Alph##1{%
5761         \ifx##1\c@page \gls@Alphpage\else\gls@orgAlph##1\fi}%

```

Add hook to allow for other number formats:

```
5762     \wrsglossarynumberhook
```

Prevent expansion:

```
5763     \gls@disablepagerefexpansion
```

Now store location in \@glslocref:

```
5764     \protected\xdef\@glslocref{\the\glsentrycounter}%

```

```
5765     \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.

```
5766     \gls@checkmkidxchars\@glslocref
```

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

```

5767     \expandafter\ifx\the\glsentrycounter\the\glsentrycounter\relax
5768         \def\@glo@counterprefix{}%
5769     \else
5770         \protected\edef\@glsHlocref{\the\glsentrycounter}%
5771         \gls@checkmkidxchars\@glsHlocref
5772         \edef\@do@gls@getcounterprefix{\noexpand\gls@getcounterprefix
5773             {\@glslocref}{\@glsHlocref}}%
5774         }%
5775         \@do@gls@getcounterprefix
5776     \fi

```

De-tok label if required

```
5777     \edef\@gls@label{\glsdetoklabel{#1}}%
```

Write the information to file:

```
5778     \@@do@@wrglossary
```

```
5779 }
```

`\@@do@@wrglossary`

```
5780 \newcommand*\@@do@@wrglossary}{%
```

Determine whether to use `xindy` or `makeindex` syntax

```
5781     \ifglsxindy
```

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

```

5782     \expandafter\glo@check@mkidxrangefchar\glsnumberformat\@nil
5783     \def\@glo@range{}%
5784     \expandafter\if\@glo@prefix(\relax
5785         \def\@glo@range{:open-range}%
5786     \else

```

```

5787     \expandafter\if\@glo@prefix)\relax
5788         \def\@glo@range{:close-range}%
5789     \fi
5790 \fi
```

Write to the glossary file using xindy syntax.

```

5791 \gls@glossary{\csname glo@\gls@label @type\endcsname}{%
5792   (indexentry :tkey (\csname glo@\gls@label @index\endcsname)
5793   :locref \string"\{@glo@counterprefix}\{@glslocref}\string" %
5794   :attr \string"\@gls@counter\@glo@suffix\string"
5795   \@glo@range
5796   )
5797 }%
5798 \else
```

Convert the format information into the format required for makeindex

```

5799 \set@glo@numformat{\glo@numfmt}{\gls@counter}{\glsnumberformat}%
5800 {\glo@counterprefix}%
```

Write to the glossary file using makeindex syntax.

```

5801 \gls@glossary{\csname glo@\gls@label @type\endcsname}{%
5802   \string\glossaryentry{\csname glo@\gls@label @index\endcsname
5803   \@gls@encapchar\glo@numfmt}\{@glslocref}%
5804 \fi
5805 }
```

`\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.)

```

5806 \newcommand*\gls@getcounterprefix[2]{%
5807   \edef\@gls@thisloc{\#1}\edef\@gls@thisHloc{\#2}%
5808   \ifx\@gls@thisloc\@gls@thisHloc
5809     \def\@glo@counterprefix{}%
5810   \else
5811     \def\@gls@get@counterprefix##1.#1##2\end@getprefix{%
5812       \def\@glo@tmp{\#2}%
5813       \ifx\@glo@tmp\empty
5814         \def\@glo@counterprefix{}%
5815       \else
5816         \def\@glo@counterprefix{\#1}%
5817       \fi
5818     }%
5819   \gls@get@counterprefix#2.#1\end@getprefix
5820 }
```

Warn if no prefix can be formed.

```

5820 \ifx\@glo@counterprefix\empty
5821   \GlossariesWarning{Hyper target '#2' can't be formed by}
```

```

5822     prefixing^^Jlocation '#1'. You need to modify the
5823     definition of \string\theH\@gls@counter^^Jotherwise you
5824     will get the warning: "'name{\@gls@counter.#1}' has been^^J
5825     referenced but does not exist"}%
5826   \fi
5827 \fi
5828 }

```

## 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.

```

5829 \newcommand{\@do@seeglossary}[2]{%
5830 \def\@gls@xref[#2]{%
5831 \@onelevel@sanitize\@gls@xref
5832 \@gls@checkmkidxchars\@gls@xref
5833 \ifglsxindy
5834   \gls@glossary{\csname glo@#1@type\endcsname}{%
5835     (indexentry
5836       :tkey (\csname glo@#1@index\endcsname)
5837       :xref (\string"\@gls@xref\string")
5838       :attr \string"see\string"
5839     )
5840   }%
5841 \else
5842   \gls@glossary{\csname glo@#1@type\endcsname}{%
5843     \string\glossaryentry{\csname glo@#1@index\endcsname
5844     \@gls@encapchar glsseeformat\@gls@xref}{Z}}%
5845 \fi
5846 }

```

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

```

5847 \def\@gls@fixbraces#1#2#3\@nil{%
5848   \ifx#2[\relax
5849     \@gls@fixbraces#1#2#3\@end@fixbraces
5850   \else
5851     \def#1{{#2#3}}%
5852   \fi
5853 }

```

`@@gls@fixbraces`

```

5854 \def\@@gls@fixbraces#1[#2]#3\@end@fixbraces{%
5855   \def#1{[#2]{#3}}%
5856 }

```

`\glssee` \glssee{*label*} {*cross-ref list*}

```

5857 \newrobustcmd*{\glssee}[3][\seename]{%
5858   \do@seeglossary{#2}{[#1]{#3}}}
5859 \newcommand*{\glssee}[3][\seename]{%
5860   \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.

```

5861 \newrobustcmd*{\glsseeformat}[3][\seename]{%
5862   \emph{#1} \glsseelist{#2}}

```

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

```

5863 \newrobustcmd*{\glsseelist}[1]{%

```

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

```

5864 \let\@gls@dolast\relax

```

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

```

5865 \let\@gls@donext\relax

```

Iterate through the labels

```

5866 \cfor\@gls@thislabel:=#1\do{%

```

Check if on last iteration of loop

```

5867 \ifx\@x\@nextelement\@nnil

```

```

5868   \gls@dolast

```

```

5869 \else

```

```

5870   \gls@donext

```

```

5871 \fi

```

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

```

5872 \expandafter\glsseeitem\expandafter{\@gls@thislabel}%

```

Update separators

```

5873 \let\@gls@dolast\glsseelastsep

```

```

5874 \let\@gls@donext\glsseesep

```

```

5875 }%

```

```

5876 }

```

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

```

5877 \newcommand*{\glsseelastsep}{\space\andname\space}

```

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

```

5878 \newcommand*{\glsseesep}{, }

```

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

```

5879 \newrobustcmd*{\glsseeitem}[1]{\glshyperlink[\glsseeitemformat{#1}]{#1}}

```

\glsseeitemformat As from v3.0, default is to use \glsentrytext instead of \glsentryname. (To avoid problems with the name key being sanitized, although this is no longer a problem now.)

```

5880 \newcommand*{\glsseeitemformat}[1]{\glsentrytext{#1}}

```

## 1.16 Displaying the glossary

An individual glossary is displayed in the text using `\printglossary[<key-val list>]`. If the type key is omitted, the default glossary is displayed. The optional argument can be used to specify an alternative glossary, and can also be used to set the style, title and entry in the table of contents. Available keys are defined below.

`save@numberlist` Provide command to store number list.

```
5881 \newcommand*{\gls@save@numberlist}[1]{%
5882   \ifglssavename@numberlist
5883     \toks@{\#1}%
5884     \edef\@do@writeaux@info{%
5885       \noexpand\csgdef{glo@\glscurrententrylabel}{\numberlist}{\the\toks@}%
5886     }%
5887     \onelevel@sanitize\@do@writeaux@info
5888     \protected@write\@auxout{}{\@do@writeaux@info}%
5889   \fi
5890 }
```

`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.)

```
5891 \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.

```
5892 \ifcsundef{printglossary}{}%
5893 {%
  If \printglossary is already defined, issue a warning and undefine it.
5894   \gls@warnonglossdefined
5895   \undef\printglossary
5896 }
```

Neither `\printglossary` nor `\printnoidxglossary` can work with an ignored glossary (since ignored glossaries normally suppress indexing and there's no associated file for `makeindex`/`xindy` to process). However `\printunsrtglossary` can be used with an ignored glossary, so provide a command to warn if the glossary doesn't exist or is an ignored glossary.

`xists@noignored`

```
5897 \newcommand*{\@printgloss@checkexists@noignored}[2]{%
5898   \ifglossaryexists{\#1}%
5899   {%
5900     \ifignoredglossary{\#1}%
5901       {\GlossariesWarning{Glossary '#1' is an ignored glossary}}%
5902       {\GlossariesWarning{Glossary '#1' doesn't exist}}%
5903     }%
5904   }%
5905 }
```

ts@allowignored For use with \printunsrtglossary.

```
5906 \newcommand*{\@printgloss@checkexists@allowignored}[2]{%
5907   \s@ifglossaryexists{#1}%
5908   {#2}%
5909   {\GlossariesWarning{Glossary '#1' doesn't exist}}%
5910 }
```

oss@checkexists

```
5911 \let\@printgloss@checkexists\@printgloss@checkexists@noignored
```

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

```
5912 \newcommand*{\printglossary}[1][type=\glsdefaulttype]{%
5913   \let\@printgloss@checkexists\@printgloss@checkexists@noignored
5914   \@printglossary{#1}{\@print@glossary}%
5915 }
```

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

```
5916 \newcommand*{\printglossaries}{%
5917   \forallglossaries{\@glo@type}{\printglossary[type=\@glo@type]}%
5918 }
```

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

```
5919 \newcommand*{\printnoidxglossary}[1][type=\glsdefaulttype]{%
5920   \let\@printgloss@checkexists\@printgloss@checkexists@noignored
5921   \@printglossary{#1}{\@printnoidx@glossary}%
5922 }
```

noidxglossaries Analogous to \printglossaries

```
5923 \newcommand*{\printnoidxglossaries}{%
5924   \forallglossaries{\@glo@type}{\printnoidxglossary[type=\@glo@type]}%
5925 }
```

ntgloss@setsort Initialise to do nothing.

```
5926 \newcommand*{\@printgloss@setsort}{}%
```

preglossaryhook

```
5927 \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. This is also used by glossaries-extra's \printunsrtglossary which may be used with an ignored glossary.

```
5928 \newcommand{\@printglossary}[2]{%
  Set up defaults.
  5929  \def\@glo@type{\glsdefaulttype}%
  5930  \def\glossarytitle{\csname @glotype@\@glo@type @title\endcsname}%
  5931  \def\glossarytoctitle{\glossarytitle}%
  5932  \let\org@glossarytitle\glossarytitle

  5933  \def\@glossarystyle{%
    5934    \ifx\@glossary@default@style\relax
      \GlossariesWarning{No default glossary style provided \MessageBreak
        for the glossary '\@glo@type'. \MessageBreak
        Using deprecated fallback. \MessageBreak
        To fix this set the style with \MessageBreak
        \string\setglossarystyle\space or use the \MessageBreak
        style key=value option}%
    5941    \fi
  5942  }%
  5943  \def\gls@dotocitle{\glssettoctitle{\@glo@type}}%
```

Store current value of \glossaryentrynumbers. (This may be changed via the optional argument)

```
5944  \let\org@glossaryentrynumbers\glossaryentrynumbers
```

Localise the effects of the optional argument

```
5945  \bgroup
```

Activate or deactivate sort key:

```
5946  \@printgloss@setsort
```

Determine settings specified in the optional argument.

```
5947  \setkeys{printgloss}{#1}%
```

Does the glossary exist?

```
5948  \@printgloss@checkexists{\@glo@type}%
5949  {%
```

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)

```
5950  \ifx\glossarytitle\org@glossarytitle
5951  \else
5952  \expandafter\let\csname @glotype@\@glo@type @title\endcsname
5953  \glossarytitle
5954  \fi
```

Allow a high-level user command to indicate the current glossary

```
5955  \let\currentglossary\@glo@type
```

Enable individual number lists to be suppressed.

```
5956 \let\org@glossaryentrynumbers\glossaryentrynumbers  
5957 \let\glsnonextpages\@glsnonextpages
```

Enable individual number list to be activated:

```
5958 \let\glsnextpages\@glsnextpages
```

Enable suppression of description terminators.

```
5959 \let\nopostdesc\@nopostdesc
```

Set up the entry for the TOC

```
5960 \gls@dotocitle
```

Set the glossary style

```
5961 \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):

```
5962 \let\gls@org@glossaryentryfield\glossentry  
5963 \let\gls@org@glossarysubentryfield\subglossentry  
5964 \renewcommand{\glossentry}[1]{%  
5965   \xdef\glscurrententrylabel{\glsdetoklabel{##1}}%  
5966   \gls@org@glossaryentryfield{##1}%  
5967 }%  
5968 \renewcommand{\subglossentry}[2]{%  
5969   \xdef\glscurrententrylabel{\glsdetoklabel{##2}}%  
5970   \gls@org@glossarysubentryfield{##1}{##2}%  
5971 }%  
5972 \glossaryhook
```

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

```
5973 #2%  
5974 }%
```

End the current scope

```
5975 \egroup
```

Reset \glossaryentrynumbers

```
5976 \global\let\glossaryentrynumbers\org@glossaryentrynumbers
```

Suppress warning about no \printglossary

```
5977 \global\let\warn@noprintglossary\relax  
5978 }
```

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

```
5979 \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.)

```
5980 \makeatletter
```

Input the glossary file, if it exists.

```
5981  \cinput{\jobname.\csname @glo@type @in\endcsname}%
```

If the glossary file doesn't exist, do `\null`. (This ensures that the page is shipped out and all write commands are done.) This might produce an empty page, but at this point the document isn't complete, so it shouldn't matter.

```
5982  \IfFileExists{\jobname.\csname @glo@type @in\endcsname}%
5983  {}%
5984  {\null}%
```

If `xindy` is being used, need to write the language dependent information to the `.aux` file for `makeglossaries`.

```
5985  \ifglsxindy
5986    \ifcsundef{@xdy@\glo@type @language}%
5987    {}%
5988    \edef@do@auxoutstuff{%
5989      \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.

```
5990      \noexpand\immediate\noexpand\write\@auxout{%
5991        \string\providetoken\string\@xdylanguage[2]{}%
5992      \noexpand\immediate\noexpand\write\@auxout{%
5993        \string\@xdylanguage{\glo@type}{\xdy@main@language}}%
5994      }%
5995    }%
5996  }%
5997  {}%
5998  \edef@do@auxoutstuff{%
5999    \noexpand\AtEndDocument{%
6000      \noexpand\immediate\noexpand\write\@auxout{%
6001        \string\providetoken\string\@xdylanguage[2]{}%
6002      \noexpand\immediate\noexpand\write\@auxout{%
6003        \string\@xdylanguage{\glo@type}{\csname @xdy@\glo@type
6004          @language\endcsname}}%
6005      }%
6006    }%
6007  }%
6008  \do@auxoutstuff
6009  \edef@do@auxoutstuff{%
6010    \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.

```
6011  \noexpand\immediate\noexpand\write\@auxout{%
6012    \string\providetoken\string@gls@codepage[2]{}%
6013  \noexpand\immediate\noexpand\write\@auxout{%
6014    \string@gls@codepage{\glo@type}{\gls@codepage}}%
6015  }%
6016 }%
```

```

6017     \cdo@auxoutstuff
6018 \fi
    Activate warning if \makeglossaries hasn't been used.
6019 \renewcommand*{\@warn@nomakeglossaries}{%
6020     \GlossariesWarningNoLine{\string\makeglossaries\space
6021     hasn't been used,^^Jthe glossaries will not be updated}%
6022 }%
6023 }

```

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
6024 \newcommand*{\@glo@sortentries}[2]{%
6025     \glosortentrieswarning
6026     \def\@glo@sortinglist{}%
6027     \def\@glo@sortinghandler{\#1}%
6028     \edef\@glo@ctype{\#2}%
6029     \forlistcsloop{\@glo@do@sortentries}{\glsref{\#2}}%
6030     \csdef{\glsref{\#2}}{}%
6031     \@for\@this@label:=\@glo@sortinglist\do{%

```

Has this entry already been added?

```

6032     \xifinlistcs{\@this@label}{\glsref{\#2}}%
6033     {}%
6034     {}%
6035     \listcsxadd{\glsref{\#2}}{\@this@label}%
6036     }%
6037     \ifcsdef{\glo@sortingchildren}{\@this@label}{}%
6038     {}%
6039     \glo@addchildren{\#2}{\@this@label}%
6040     }%
6041     {}%
6042 }%
6043 }

```

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

```

6044 \newcommand*{\@glo@addchildren}[2]{%
Scope to allow nesting.
6045 \bgroup
6046     \letcs{\glo@childlist}{\glo@sortingchildren{\#2}}%

```

```
6047     \@for\@this@childlabel:=\@glo@childlist\do
6048     {%
```

Check this label hasn't already been added.

```
6049     \xifinlistcs{\@this@childlabel}{@glsref@#1}%
6050     {}%
6051     {%
6052         \listcsxadd{@glsref@#1}{\@this@childlabel}%
6053     }%
```

Does this child have children?

```
6054     \ifcsdef{@glo@sortingchildren@\@this@childlabel}%
6055     {%
6056         \@glo@addchildren{#1}{\@this@childlabel}%
6057     }%
6058     {%
6059     }%
6060     }%
6061 \egroup
6062 }
```

@do@sortentries

```
6063 \newcommand*{\@glo@do@sortentries}[1]{%
6064     \ifglshasparent{#1}%
6065     {%
```

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

```
6066     \edef\@glo@parent{\csuse{\glo@glsdetoklabel{#1}@parent}}%
6067     \ifcsundef{@glo@sortingchildren@\@glo@parent}%
6068     {%
6069         \csdef{@glo@sortingchildren@\@glo@parent}{}%
6070     }%
6071     {}%
6072     \expandafter\@glo@sortedinsert
6073     \csname @glo@sortingchildren@\@glo@parent\endcsname{#1}%

```

Has the parent been added?

```
6074     \xifinlistcs{\@glo@parent}{@glsref@\@glo@type}%
6075     {%
```

Yes, it has so do nothing.

```
6076     }%
6077     {%
```

No, it hasn't so add it now.

```
6078     \expandafter\@glo@do@sortentries\expandafter{\@glo@parent}%
6079     }%
6080 }%
6081 {%
6082     \@glo@sortedinsert{\@glo@sortinglist}{#1}%
6083 }%
6084 }
```

```
lo@sortedinsert \@glo@sortedinsert{\list}{\entry_label}
```

Insert into list.

```
6085 \newcommand*{\@glo@sortedinsert}[2]{%
6086   \dtl@insertinto{\#2}{\#1}{\@glo@sorthandler}%
6087 }%
```

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

```
6088 \newcommand*{\@glo@sorthandler@word}[2]{%
6089   \letcs@gls@sort@A{\glo@glsdetoklabel{\#1}@sort}%
6090   \letcs@gls@sort@B{\glo@glsdetoklabel{\#2}@sort}%
6091   \edef\glo@do@compare{%
6092     \noexpand\dtlwordindexcompare{\noexpand\dtl@sortresult}%
6093     {\expandonce@gls@sort@B}%
6094     {\expandonce@gls@sort@A}%
6095   }%
6096   \glo@do@compare
6097 }
```

thandler@letter

```
6098 \newcommand*{\@glo@sorthandler@letter}[2]{%
6099   \letcs@gls@sort@A{\glo@glsdetoklabel{\#1}@sort}%
6100   \letcs@gls@sort@B{\glo@glsdetoklabel{\#2}@sort}%
6101   \edef\glo@do@compare{%
6102     \noexpand\dtlletterindexcompare{\noexpand\dtl@sortresult}%
6103     {\expandonce@gls@sort@B}%
6104     {\expandonce@gls@sort@A}%
6105   }%
6106   \glo@do@compare
6107 }
```

orthandler@case Case-sensitive sort.

```
6108 \newcommand*{\@glo@sorthandler@case}[2]{%
6109   \letcs@gls@sort@A{\glo@glsdetoklabel{\#1}@sort}%
6110   \letcs@gls@sort@B{\glo@glsdetoklabel{\#2}@sort}%
6111   \edef\glo@do@compare{%
6112     \noexpand\dtlcompare{\noexpand\dtl@sortresult}%
6113     {\expandonce@gls@sort@B}%
6114     {\expandonce@gls@sort@A}%
6115   }%
6116   \glo@do@compare
6117 }
```

thandler@nocase Case-insensitive sort.

```

6118 \newcommand*{\@glo@sorthandler@nocase}[2]{%
6119   \letcs\@gls@sort@A{\glo@glsdetoklabel{#1}@sort}%
6120   \letcs\@gls@sort@B{\glo@glsdetoklabel{#2}@sort}%
6121   \edef\glo@do@compare{%
6122     \noexpand\dtlicompare{\noexpand\dtl@sortresult}%
6123     {\expandonce\@gls@sort@B}%
6124     {\expandonce\@gls@sort@A}%
6125   }%
6126   \glo@do@compare
6127 }

@sortmacro@word Sort macro for 'word'
6128 \newcommand*{\@glo@sortmacro@word}[1]{%
6129   \ifdefstring{\@glo@default@sorttype}{standard}{%
6130     {%
6131       \@glo@sortentries{\@glo@sorthandler@word}{#1}%
6132     }%
6133   }%
6134   \PackageError{glossaries}{Conflicting sort options:^^J
6135     \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
6136     \string\printnoidxglossary[sort=word]}{}%
6137 }%
6138 }

@sortmacro@letter Sort macro for 'letter'
6139 \newcommand*{\@glo@sortmacro@letter}[1]{%
6140   \ifdefstring{\@glo@default@sorttype}{standard}{%
6141     {%
6142       \@glo@sortentries{\@glo@sorthandler@letter}{#1}%
6143     }%
6144   }%
6145   \PackageError{glossaries}{Conflicting sort options:^^J
6146     \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
6147     \string\printnoidxglossary[sort=letter]}{}%
6148 }%
6149 }

@tmacro@standard Sort macro for 'standard'. (Use either 'word' or 'letter' order.)
6150 \newcommand*{\@glo@sortmacro@standard}[1]{%
6151   \ifdefstring{\@glo@default@sorttype}{standard}{%
6152     {%
6153       \ifcsdef{\glo@sorthandler@\glsorder}{}{%
6154         {%
6155           \@glo@sortentries{\csuse{\glo@sorthandler@\glsorder}}{#1}%
6156         }%
6157       }%
6158       \PackageError{glossaries}{Unknown sort handler '\glsorder'}{}%
6159     }%
6160   }%

```

```

6161  {%
6162    \PackageError{glossaries}{Conflicting sort options:^^J
6163      \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
6164      \string\printnoidxglossary[sort=standard]}{}%
6165  }%
6166 }

@sortmacro@case Sort macro for 'case'
6167 \newcommand*{\@glo@sortmacro@case}[1]{%
6168   \ifdefstring{\@glo@default@sorttype}{standard}{%
6169     {%
6170       \@glo@sortentries{\@glo@sorthandler@case}{#1}%
6171     }%
6172   }%
6173   \PackageError{glossaries}{Conflicting sort options:^^J
6174     \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
6175     \string\printnoidxglossary[sort=case]}{}%
6176 }%
6177 }

@sortmacro@nocase Sort macro for 'nocase'
6178 \newcommand*{\@glo@sortmacro@nocase}[1]{%
6179   \ifdefstring{\@glo@default@sorttype}{standard}{%
6180     {%
6181       \@glo@sortentries{\@glo@sorthandler@nocase}{#1}%
6182     }%
6183   }%
6184   \PackageError{glossaries}{Conflicting sort options:^^J
6185     \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
6186     \string\printnoidxglossary[sort=nocase]}{}%
6187 }%
6188 }

o@sortmacro@def Sort macro for 'def'. The order of definition is given in \glolist@<type>.
6189 \newcommand*{\@glo@sortmacro@def}[1]{%
6190   \def\@glo@sortinglist{}%
6191   \forglsentries[#1]{\@gls@thislabel}{%
6192     {%
6193       \xifinlistcs{\@gls@thislabel}{\@glsref@#1}%
6194       {%
6195         \listeadadd{\@glo@sortinglist}{\@gls@thislabel}%
6196       }%
6197     }%
6198   }%
6199 }%
6200 \cslet{\glsref@#1}{\@glo@sortinglist}%
6201 }

```

ortmacro@def@do This won't include parent entries that haven't been referenced.

```
6202 \newcommand*{\@glo@sortmacro@def@do}[1]{%
6203   \ifinlistcs{#1}{\glsref@\glo@type}%
6204   {}%
6205   {}%
6206   \listcsadd{\glsref@\glo@type}{#1}%
6207 }%
6208 \ifcsdef{\glo@sortingchildren@#1}%
6209 {}%
6210   \glo@addchildren{\glo@type}{#1}%
6211 }%
6212 {}%
6213 }
```

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

```
6214 \newcommand*{\@glo@sortmacro@use}[1]{}
```

cnidx@glossary Glossary handler for \printnidxglossary which doesn't use an indexing application. Since \printnidxglossary 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.

```
6215 \newcommand*{\@print@nidx@glossary}{%
6216   \ifcsdef{\glsref@\glo@type}%
6217   {}%
```

Sort the entries:

```
6218   \ifcsdef{\glo@sortmacro@\glo@sorttype}%
6219   {}%
6220     \csuse{\glo@sortmacro@\glo@sorttype}{\glo@type}%
6221   }%
6222   {}%
6223     \PackageError{glossaries}{Unknown sort handler '\glo@sorttype'}{}%
6224 }%
```

Do the glossary heading and preamble

```
6225 \glossarysection[\glossarytoctitle]{\glossarytitle}%
6226 \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.

```
6227 \def\gls@currentlettergroup{}%
6228 \begin{theglossary}%
6229 \glossaryheader
6230 \glsresetentrylist
```

Iterate through the entries.

```
6231 \forlistcsloop{\@gls@noidx@do}{\@glsref@\@glo@type}%
```

Finally end the glossary and do the postamble:

```
6232 \end{theglossary}%
6233 \glossarypostamble
6234 }%
6235 {%
6236 \@gls@noref@warn{\@glo@type}%
6237 }%
6238 }
```

\glo@grabfirst

```
6239 \def\glo@grabfirst#1#2\@nil{%
6240 \def\@gls@firsttok{#1}%
6241 \ifdefempty\@gls@firsttok
6242 {%
6243 \def\@glo@thislettergrp{0}%
6244 }%
6245 {%
```

Sanitize it:

```
6246 @onellevel@sanitize\@gls@firsttok
```

Fetch the first letter:

```
6247 \expandafter\glo@grabfirst\@gls@firsttok{}{}\@nil
6248 }%
6249 }
```

\@glo@grabfirst

```
6250 \def\@glo@grabfirst#1#2\@nil{%
6251 \ifdefempty\@glo@thislettergrp
6252 {%
6253 \def\@glo@thislettergrp{glssymbols}%
6254 }%
6255 {%
6256 \count@=\uccode`#1\relax
6257 \ifnum\count@=0\relax
6258 \def\@glo@thislettergrp{glssymbols}%
6259 \else
6260 \ifdefstring\@glo@sorttype{case}%
6261 {%
6262 \count@=\#1\relax
6263 }%
6264 {%
6265 }%
6266 \edef\@glo@thislettergrp{\the\count@}%
6267 \fi
6268 }%
6269 }
```

```

\@gls@noidx@do Handler for list iteration used by \print@noidx@glossary. The argument is the entry label.
This only allows one sublevel.
6270 \newcommand{\@gls@noidx@do}[1]{%
  Get this entry's location list
6271   \global\let\cs{\\gls@loclist}{\glo@\glsdetoklabel{#1}@loclist}%
  Does this entry have a parent?
6272   \ifglshasparent{#1}%
6273   {%
    Has a parent.
6274     \gls@level=\csuse{\glo@\glsdetoklabel{#1}@level}\relax
6275     \ifdefvoid{\gls@loclist}%
6276     {%
6277       \subglossentry{\gls@level}{#1}{}%
6278     }%
6279     {%
6280       \subglossentry{\gls@level}{#1}%
6281     }%
6282     \glossaryentrynumbers{\glsnoidxloclist{\gls@loclist}}%
6283   }%
6284 }%
6285 {%
6286 }%
Doesn't have a parent Get this entry's sort key
6287 \let\cs{\gls@sort}{\glo@\glsdetoklabel{#1}@sort}%
Fetch the first letter:
6288 \expandafter\glo@grabfirst\gls@sort{}{}@\nil
6289 \ifdefequal{\glo@thislettergrp}{\gls@currentlettergroup}%
6290 {%
6291 }%
Do the group header:
6292 \ifdefempty{\gls@currentlettergroup}{}%
6293 {%
The group skip may start a new scope, so make a global assignment.
6294 \global\let\glo@thislettergrp\glo@thislettergrp
6295   \glsgroupskip
6296 }%
6297   \glsgroupheading{\glo@thislettergrp}%
6298 }%
6299 \global\let\gls@currentlettergroup\glo@thislettergrp

Do this entry:
6300 \ifdefvoid{\gls@loclist}%
6301 {%
6302   \glossentry{#1}{}%

```

```

6303    }%
6304    {%
6305        \glossentry{#1}%
6306        {%
6307            \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
6308        }%
6309    }%
6310 }%
6311 }

```

`\glsnoidxloclist{<list cs>}`

Display location list.

```

6312 \newcommand*{\glsnoidxloclist}[1]{%
6313     \def\@gls@noidxloclist@sep{}%
6314     \def\@gls@noidxloclist@prev{}%
6315     \forlistloop{\glsnoidxloclisthandler}{#1}%
6316 }

```

`xloclisthandler` Handler for location list iterator.

```

6317 \newcommand*{\glsnoidxloclisthandler}[1]{%
6318     \ifdefstring{\@gls@noidxloclist@prev}{#1}%
6319     {%

```

Same as previous location so skip.

```

6320     }%
6321     {%
6322         \@gls@noidxloclist@sep
6323         #1%
6324         \def\@gls@noidxloclist@sep{\delimN}%
6325         \def\@gls@noidxloclist@prev{#1}%
6326     }%
6327 }

```

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

```

6328 \newcommand*{\glsnoidxdisplayloclisthandler}[1]{%
6329     \ifdefstring{\@gls@noidxloclist@prev}{#1}%
6330     {%

```

Same as previous location so skip.

```

6331     }%
6332     {%
6333         \@gls@noidxloclist@sep
6334         \@gls@noidxloclist@prev
6335         \def\@gls@noidxloclist@prev{#1}%
6336     }%
6337 }

```

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

Display a location in the location list.

```
6338 \newcommand*\glsnoidxdisplayloc[4]{%
6339   \setentrycounter[#1]{#2}%
6340   \csuse{#3}{#4}%
6341 }
```

```
\@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.

```
6342 \newcommand*{\@gls@reference}[3]{%
```

Add to label list

```
6343  \glsdoifexistsorwarn{#2}%
6344  {%
6345    \ifcsgundef{\glsref@#1}{\csgdef{\glsref@#1}{}{}}%
6346    \ifinlistcs{#2}{\glsref@#1}%
6347    {}%
6348    {\listcsgadd{\glsref@#1}{#2}}%
```

Add to location list

```
6349  \ifcsgundef{\glo@\glsdetoklabel{#2}@loclist}{%
6350    {\csgdef{\glo@\glsdetoklabel{#2}@loclist}{}{}}%
6351    {}%
6352    {\listcsgadd{\glo@\glsdetoklabel{#2}@loclist}{#3}}%
6353  }%
6354 }
```

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

```
6355 \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`.

```
6356 \define@key{printgloss}{title}{%
6357   \def@glossarytitle{#1}%
6358   \let@gls@dotocitle\relax
6359 }
```

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

```
6360 \define@key{printgloss}{toctitle}{%
6361   \def@glossarytoctitle{#1}%
6362   \let@gls@dotocitle\relax
6363 }
```

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

```
6364 \define@key{printgloss}{style}{%
```

```

6365 \ifcsundef{@glsstyle@#1}%
6366 {%
6367   \PackageError{glossaries}%
6368   {Glossary style '#1' undefined}{}%
6369 }%
6370 {%
6371   \def\@glossarystyle{\setglossentrycompatibility
6372     \csname @glsstyle@#1\endcsname}%
6373 }%
6374 }

```

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

```

6375 \define@choicekey{printgloss}{numberedsection}%
6376   [\gls@numberedsection@val\gls@numberedsection@nr]%
6377   {false,nolabel,autolabel,nameref}[nolabel]%
6378 {%
6379   \ifcase\gls@numberedsection@nr\relax
6380     \renewcommand*\@glossarysecstar}{*}%
6381     \renewcommand*\@glossaryseclabel}{}
6382 \or
6383   \renewcommand*\@glossarysecstar}{}
6384   \renewcommand*\@glossaryseclabel}{}
6385 \or
6386   \renewcommand*\@glossarysecstar}{}
6387   \renewcommand*\@glossaryseclabel}{\label{\glsautoprefix@glo@type}}%
6388 \or
6389   \renewcommand*\@glossarysecstar}{*}%
6390   \renewcommand*\@glossaryseclabel}{%
6391     \protected@edef\@currentlabelname{\glossarytoctitle}%
6392     \label{\glsautoprefix@glo@type}}%
6393 \fi
6394 }

```

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

```

6395 \define@choicekey{printgloss}{nogroupskip}{true,false}[true]{%
6396   \csuse{glsnogroupskip#1}%
6397 }

```

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

```

6398 \define@choicekey{printgloss}{nopostdot}{true,false}[true]{%
6399   \csuse{glsnopostdot#1}%
6400 }

```

`nterLabelPrefix` Make it easier to redefine the label prefix.

```

6401 \newcommand*\GlsEntryCounterLabelPrefix{\glsentry-}

```

The conditionals have been moved inside the appropriate commands to make it easier for the user to redefine them in the preamble and selectively switch the counter display on and

off. Previously the helper commands were redefined by the entrycounter option, which would counteract any earlier customisation.

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

```
6402 \define@choicekey{printgloss}{entrycounter}{true,false}[true]{%
6403   \csuse{glsentrycounter#1}%
6404   \gls@define@glossaryentrycounter
6405 }
```

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.

```
6406 \define@choicekey{printgloss}{subentrycounter}{true,false}[true]{%
6407   \csuse{glssubentrycounter#1}%
6408   \gls@define@glossarysubentrycounter
6409 }
```

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

```
6410 \define@boolkey{printgloss}[gls]{nonumberlist}[true]{%
6411 \ifglsnonumberlist
6412   \def\glossaryentrynumbers##1{}%
6413 \else
6414   \def\glossaryentrynumbers##1{##1}%
6415 \fi}
```

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

```
6416 \define@key{printgloss}{sort}{\glo@assign@sortkey{#1}}
```

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

```
6417 \newcommand*{\glo@no@assign@sortkey}[1]{%
6418   \PackageError{glossaries}{‘sort’ key not permitted with
6419   \string\printglossary}%
6420   {The ‘sort’ key may only be used with \string\printnoidxglossary}%
6421 }
```

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

```
6422 \newcommand*{\glo@sortkey}[1]{%
6423   \def\glo@sorttype{#1}%
6424 }
```

`@glsnonextpages` Suppresses the next number list only. Global assignments required as it may not occur in the same level of grouping as the next numberlist. (For example, if `\glsnonextpages` is placed in the entry's description and 3 column tabular style glossary is used.) `\org@glossaryentrynumbers` needs to be set at the start of each glossary, in the event that `\glossaryentrynumber` is re-defined.

```
6425 \newcommand*{\glsnonextpages}{%
6426   \gdef\glossaryentrynumbers##1{%
6427     \glsresetentrylist
6428   }%
6429 }
```

\@glsnextpages Activate the next number list only. Global assignments required as it may not occur in the same level of grouping as the next numberlist. (For example, if \glsnextpages is place in the entry's description and 3 column tabular style glossary is used.) \org@glossaryentrynumbers needs to be set at the start of each glossary, in the event that \glossaryentrynumber is re-defined.

```
6430 \newcommand*{\@glsnextpages}{%
6431   \gdef\glossaryentrynumbers##1{%
6432     ##1\glsresetentrylist}}
```

sresetentrylist Resets \glossaryentrynumbers

```
6433 \newcommand*{\glsresetentrylist}{%
6434   \global\let\glossaryentrynumbers\org@glossaryentrynumbers}
```

\glsnonextpages Outside of \printglossary this does nothing.

```
6435 \newcommand*{\glsnonextpages}{}{}
```

\glsnextpages Outside of \printglossary this does nothing.

```
6436 \newcommand*{\glsnextpages}{}{}
```

Process entrycounter and then subentrycounter options (this ensures the sub-counter can pick up the main counter as the master if required):

```
6437 \@gls@define@glossaryentrycounter
6438 \@gls@define@glossarysubentrycounter
```

subentrycounter Resets the glossarysubentry counter.

```
6439 \newcommand*{\glsresetsubentrycounter}{%
6440   \ifglssubentrycounter
6441     \setcounter{glossarysubentry}{0}%
6442   \fi
6443 }
```

subentrycounter Resets the glossaryentry counter.

```
6444 \newcommand*{\glsresetentrycounter}{%
6445   \ifglsentrycounter
6446     \setcounter{glossaryentry}{0}%
6447   \fi
6448 }
```

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

```
6449 \newcommand*{\glsstepentry}[1]{%
6450   \ifglsentrycounter
6451     \refstepcounter{glossaryentry}%
6452     \label{\GlsEntryCounterLabelPrefix\glsdetoklabel{#1}}%
6453   \fi
6454 }
```

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

```
6455 \newcommand*{\glsstepsubentry}[1]{%
6456   \ifglssubentrycounter
6457     \edef\currentglssubentry{\glsdetoklabel{#1}}%
6458     \refstepcounter{glossarysubentry}%
6459     \label{\GlsEntryCounterLabelPrefix\currentglssubentry}%
6460   \fi
6461 }
```

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

```
6462 \newcommand*{\glsrefentry}[1]{%
6463   \ifglsentrycounter
6464     \ref{\GlsEntryCounterLabelPrefix\glsdetoklabel{#1}}%
6465   \else
6466     \ifglssubentrycounter
6467       \ref{\GlsEntryCounterLabelPrefix\glsdetoklabel{#1}}%
6468     \else
6469       \gls{#1}%
6470     \fi
6471   \fi
6472 }
```

`trycounterlabel` Defines how to display the glossaryentry counter.

```
6473 \newcommand*{\glsentrycounterlabel}{%
6474   \ifglsentrycounter
6475     \the glossaryentry.\space
6476   \fi
6477 }
```

`trycounterlabel` Defines how to display the glossarysubentry counter.

```
6478 \newcommand*{\glssubentrycounterlabel}{%
6479   \ifglssubentrycounter
6480     \the glossarysubentry)\space
6481   \fi
6482 }
```

`\glsentryitem` Step and display glossaryentry counter, if appropriate.

```
6483 \newcommand*{\glsentryitem}[1]{%
6484   \ifglsentrycounter
6485     \glsstepentry{#1}\glsentrycounterlabel
6486   \else
6487     \glsresetsubentrycounter
6488   \fi
6489 }
```

`glssubentryitem` Step and display glossarysubentry counter, if appropriate.

```
6490 \newcommand*{\glssubentryitem}[1]{%
```

```

6491 \ifglssubentrycounter
6492   \glsstepsubentry{#1}\glssubentrycounterlabel
6493 \fi
6494 }

```

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

```

6495 \ifcsundef{theglossary}%
6496 {%
6497   \newenvironment{theglossary}{}{}%
6498 }%
6499 {%
6500   \gls@warnonthe glossdefined
6501   \renewenvironment{theglossary}{}{}%
6502 }

```

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
6503 \newcommand*{\glossaryheader}{}

```

```

\glstarget {\glstarget{\langle label \rangle}{\langle name \rangle}}

```

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

```

6504 \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.

```

\glossentry{\langle label \rangle}{\langle page-list \rangle}

```

```

6505 \providecommand*{\compatibleglossentry}[2]{%
6506   \toks@{\#2}%
6507   \protected@edef\@do@glossentry{\noexpand\glossaryentryfield{\#1}%
6508     {\noexpand\glsnamefont
6509       {\expandafter\expandonce\csname glo@\#1@name\endcsname}}%
6510     {\expandafter\expandonce\csname glo@\#1@desc\endcsname}}%
6511     {\expandafter\expandonce\csname glo@\#1@symbol\endcsname}}%
6512   {\the\toks@}%

```

```

6513  }%
6514  \do@glossentry
6515 }

\glossentryname
6516 \newcommand*{\glossentryname}[1]{%
6517   \glsdoifexistsorwarn{#1}%
6518   {%
6519     \letcs{\glo@name}{\glsdetoklabel{#1}@name}%
6520     \expandafter\glsnamefont\expandafter{\glo@name}%
6521   }%
6522 }

\Glossentryname
6523 \newcommand*{\Glossentryname}[1]{%
6524   \glsdoifexistsorwarn{#1}%
6525   {%
6526     \glsnamefont{\Glsentryname{#1}}%
6527   }%
6528 }

\glossentrydesc
6529 \newcommand*{\glossentrydesc}[1]{%
6530   \glsdoifexistsorwarn{#1}%
6531   {%
6532     \glsentrydesc{#1}%
6533   }%
6534 }

\Glossentrydesc
6535 \newcommand*{\Glossentrydesc}[1]{%
6536   \glsdoifexistsorwarn{#1}%
6537   {%
6538     \Glsentrydesc{#1}%
6539   }%
6540 }

\glossentrysymbol
6541 \newcommand*{\glossentrysymbol}[1]{%
6542   \glsdoifexistsorwarn{#1}%
6543   {%
6544     \glsentrysymbol{#1}%
6545   }%
6546 }

\glossentrysymbol
6547 \newcommand*{\Glossentrysymbol}[1]{%
6548   \glsdoifexistsorwarn{#1}%

```

```

6549  {%
6550    \Glsentrysymbol{#1}%
6551  }%
6552 }

```

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

```

6553 \providecommand*\compatiblesubglossentry[3]{%
6554   \toks@{#3}%
6555   \protected@edef\do@subglossentry{\noexpand\glossarysubentryfield{\number#1}%
6556   {#2}%
6557     {\noexpand\glsnamefont
6558       {\expandafter\expandonce\csname glo@#2@name\endcsname}%
6559       {\expandafter\expandonce\csname glo@#2@desc\endcsname}%
6560       {\expandafter\expandonce\csname glo@#2@symbol\endcsname}%
6561       {\the\toks@}%
6562     }%
6563   \do@subglossentry
6564 }

```

`rycompatibility`

```

6565 \newcommand*\setglossentrycompatibility{%
6566   \let\glossentry\compatibleglossentry
6567   \let\subglossentry\compatiblesubglossentry
6568 }
6569 \setglossentrycompatibility

```

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

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

```

6570 \newcommand{\glossaryentryfield}[5]{%
6571   \GlossariesWarning
6572   {Deprecated use of \string\glossaryentryfield.^^J
6573     I recommend you change to \string\glossentry.^^J
6574     If you've just upgraded, try removing your gls auxiliary
6575     files^^J and recompile}%
6576   \noindent\textrm{\bfseries\glstarget{#1}{#2}} #4 #3. #5\par}

```

`\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.)

```
6577 \newcommand*{\glossarysubentryfield}[6]{%
6578   \GlossariesWarning
6579   {Deprecated use of \string\glossarysubentryfield.^^J
6580   I recommend you change to \string\subglossentry.^^J
6581   If you've just upgraded, try removing your gls auxiliary
6582   files^^J and recompile}%
6583   \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
6584 \newcommand*{\glsgroupskip}{}%
```

Each of the 28 glossary groups described above is preceded by a group heading. This is formatted by the command `\glsgrouphereading` which takes one argument which is the *label* assigned to that group (not the title). The corresponding labels are: `glossymbols`, `glsnumbers`, A, ..., Z. Glossary styles must redefine this command. (In between groups, `\glsgrouphereading` comes immediately after `\glsgroupskip`.)

```
\glsgrouphereading
6585 \newcommand*{\glsgrouphereading}[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 `glossymbols` produces `\glossymbolsgroupname`, the group labelled `glsnumbers` produces `\glsnumbersgroupname` and all the other groups simply produce their label. As mentioned above, the group labels are: `glossymbols`, `glsnumbers`, A, ..., Z. If you want to redefine the group titles, you will need to redefine this command. Languages other than English may produce labels that are non-expandable, so we need to check for that otherwise it will create a “missing `\endcsname` inserted” error.

```

lsgetgrouptitle
6586 \newcommand*{\glsgrouptitle}[1]{%
6587   \gls@getgrouptitle{#1}{\gls@grptitle}%
6588   \gls@grptitle
6589 }

s@getgrouptitle Gets the group title specified by the label (first argument) and stores in the second argument, which must be a control sequence.
6590 \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.
6591 \dtl@ifsingle{#1}%
6592 {%
6593   \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
6594 }%
6595 {%
6596   \ifboolexpr{test{\ifstreq{\#1}{glssymbols}}%
6597               or test{\ifstreq{\#1}{glsnumbers}}}%
6598 }%
6599   \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
6600 }%
6601 {%
6602   \def#2{#1}%
6603 }%
6604 }%
6605 }

x@getgrouptitle Version for the no-indexing app option:
6606 \newcommand*{\gls@noidx@getgrouptitle}[2]{%
6607   \DTLifint{#1}%
6608   {\edef#2{\char#1\relax}}%
6609 }%
6610   \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
6611 }%
6612 }

```

`\glsgrouplabel{(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 `\glsgrouptitle`, you will also need to redefine `\glsgrouplabel`.

```

lsgetgrouplabel
6613 \newcommand*{\glsgrouplabel}[1]{%
6614 \ifthenelse{\equal{#1}{\glssymbolsgroupname}}{\glssymbols}{%
6615 \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`

```
6616 \newcommand*{\setentrycounter}[2] [] {%
6617   \def\@glo@counterprefix{#1}%
6618   \ifx\@glo@counterprefix\empty
6619     \def\@glo@counterprefix{.}%
6620   \else
6621     \def\@glo@counterprefix{.#1.}%
6622   \fi
6623   \def\glsentrycounter{#2}%
6624 }
```

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

#### `etglossarystyle`

```
6625 \newcommand*{\setglossarystyle}[1] {%
6626   \ifcsundef{@glsstyle@#1}%
6627   {%
6628     \PackageError{glossaries}{Glossary style '#1' undefined}{}
6629   }%
6630   {%
6631     \csname @glsstyle@#1\endcsname
6632   }%
```

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

```
6633 \ifx\@glossary@default@style\relax
6634   \protected@edef\@glossary@default@style{#1}%
6635 \fi
6636 }
```

#### `\glossarystyle`

```
6637 \newcommand*{\glossarystyle}[1] {%
6638   \ifcsundef{@glsstyle@#1}%
6639   {%
6640     \PackageError{glossaries}{Glossary style '#1' undefined}{}
6641   }%
6642   {%
6643     \GlossariesWarning
6644     {Deprecated command \string\glossarystyle.^^J
6645      I recommend you switch to \string\setglossarystyle\space unless
6646      you want to maintain backward compatibility}%
6647     \setglossentrycompatibility
6648     \csname @glsstyle@#1\endcsname
6649   \ifcsdef{@glscompstyle@#1}%
6650     {\setglossentrycompatibility\csuse{@glscompstyle@#1}}%
6651   {}%
6652 }
```

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

```
6653 \ifx\@glossary@default@style\relax
6654   \protected@edef\@glossary@default@style{\#1}%
6655 \fi
6656 }
```

`\newglossarystyle` New glossary styles can be defined using:

```
\newglossarystyle{<name>}{<definition>}
```

The *<definition>* argument should redefine the `\glossary`, `\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.

```
6657 \newcommand{\newglossarystyle}[2]{%
6658   \ifcsundef{@glsstyle@\#1}%
6659   {%
6660     \expandafter\def\csname @glsstyle@\#1\endcsname{#2}%
6661   }%
6662   {%
6663     \PackageError{glossaries}{Glossary style '#1' is already defined}{}%
6664   }%
6665 }
```

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

```
6666 \newcommand{\renewglossarystyle}[2]{%
6667   \ifcsundef{@glsstyle@\#1}%
6668   {%
6669     \PackageError{glossaries}{Glossary style '#1' isn't already defined}{}%
6670   }%
6671   {%
6672     \csdef{@glsstyle@\#1}{#2}%
6673   }%
6674 }
```

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`

```
6675 \newcommand*{\glsnamefont}[1]{#1}
```

Each glossary entry has an associated number list (usually page numbers) that indicate where in the document the entry has been used. The format for these number lists can be

changed using the format key in commands like `\glslink`. The default format is given by `\glshypernumber`. This takes a single argument which may be a single number, a number range or a number list. The number ranges are delimited with `\delimR`, the number lists are delimited with `\delimN`.

If the document doesn't have hyperlinks, the numbers can be displayed just as they are, but if the document supports hyperlinks, the numbers should link to the relevant location. This means extracting the individual numbers from the list or ranges. The package does this with the `\hyperpage` command, but this is encoded for comma and dash delimiters and only for the page counter, but this code needs to be more general. So I have adapted the code used in the package.

```
\glshypernumber
6676 \ifcsundef{hyperlink}%
6677 {%
6678   \def\glshypernumber#1{\#1}%
6679 }%
6680 {%
6681   \def\glshypernumber#1{\@glshypernumber#1\nohyperpage{}\@nil}%
6682 }
```

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

```
6683 \def\@glshypernumber#1\nohyperpage#2#3\@nil{%
6684   \ifx\\#1\\%
6685   \else
6686     \@delimR#1\delimR\delimR\\%
6687   \fi
6688   \ifx\\#2\\%
6689   \else
6690     #2%
6691   \fi
6692   \ifx\\#3\\%
6693   \else
6694     \@glshypernumber#3\@nil
6695   \fi
6696 }
```

`\@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
6697 \def\@delimR#1\delimR #2\delimR #3\\{%
6698 \ifx\\#2\\%
6699   \@delimN{\#1}%
6700 \else
6701   \gls@numberlink{\#1}\delimR\gls@numberlink{\#2}%
6702 \fi}
```

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

```

\@delimN
6703 \def\@delimN#1{\@@delimN#1\delimN \delimN\\}
6704 \def\@@delimN#1\delimN #2\delimN#3\\{%
6705 \ifx\\#3\\%
6706   \@gls@numberlink{#1}%
6707 \else
6708   \@gls@numberlink{#1}\delimN\@gls@numberlink{#2}%
6709 \fi
6710 }

```

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

```

6711 \def\@gls@numberlink#1{%
6712 \begingroup
6713   \toks@={}%
6714   \@gls@removespaces#1 \@nil
6715 \endgroup}

6716 \def\@gls@removespaces#1 #2\@nil{%
6717   \toks@=\expandafter{\the\toks@#1}%
6718   \ifx\\#2\\%
6719     \edef\x{\the\toks@}%
6720     \ifx\x\empty
6721     \else

6722       \hyperlink{\glsentrycounter@glo@counterprefix\the\toks@}%
6723           {\the\toks@}%
6724     \fi
6725   \else
6726     \@gls@ReturnAfterFi{%
6727       \@gls@removespaces#2\@nil
6728     }%
6729   \fi
6730 }
6731 \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
6732 \newcommand*{\hyperrm}[1]{\textrm{\glshypernumber{#1}}}

\hypersf
6733 \newcommand*{\hypersf}[1]{\textsf{\glshypernumber{#1}}}

\hypertt
6734 \newcommand*{\hypertt}[1]{\texttt{\glshypernumber{#1}}}

```

```

\hyperbf
 6735 \newcommand*{\hyperbf}[1]{\textbf{\glshypernumber{#1}}}

\hypermd
 6736 \newcommand*{\hypermd}[1]{\textmd{\glshypernumber{#1}}}

\hyperit
 6737 \newcommand*{\hyperit}[1]{\textit{\glshypernumber{#1}}}

\hypersl
 6738 \newcommand*{\hypersl}[1]{\textsl{\glshypernumber{#1}}}

\hyperup
 6739 \newcommand*{\hyperup}[1]{\textup{\glshypernumber{#1}}}

\hypersc
 6740 \newcommand*{\hypersc}[1]{\textsc{\glshypernumber{#1}}}

\hyperemph
 6741 \newcommand*{\hyperemph}[1]{\emph{\glshypernumber{#1}}}

```

## 1.17 Acronyms

```
\oldacronym[<label>]{<abbr>}[<long>][<key-val list>]
```

This emulates the way the old package defined acronyms. It is equivalent to `\newacronym [<key-val list>] [<label>]{<abbr>}[<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, `<abbr>` 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.

```

6742 \newcommand{\oldacronym}[4]{\gls@label}{%
6743   \def\gls@label{\#2}{%
6744     \newacronym[\#4]{\#1}{\#2}{\#3}{%
6745       \ifcsundef{xspace}{%
6746         {%
6747           \expandafter\edef\csname\#1\endcsname{%
6748             \noexpand\@ifstar{\noexpand\Gls{\#1}}{\noexpand\gls{\#1}}}}%

```

```

6749     }%
6750   }%
6751   {%
6752     \expandafter\edef\csname#1\endcsname{%
6753       \noexpand\@ifstar{\noexpand\Gls{#1}\noexpand\xspace}{%
6754         \noexpand\gls{#1}\noexpand\xspace}%
6755       }%
6756     }%
6757 }

```

`\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
6758   \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.

```
6759 \newcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}
```

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

```

\glstextup
6760 \newrobustcmd*{\glstextup}[1]{\ifdef\textulc{\textulc{#1}}{\textup{#1}}}

```

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

```

\glsshortkey
6761 \newcommand*{\glsshortkey}{short}

```

```

\shortpluralkey
6762 \newcommand*{\glsshortpluralkey}{shortplural}

```

```

\glslongkey
6763 \newcommand*{\glslongkey}{long}

```

```

lslongpluralkey
6764 \newcommand{\glslongpluralkey}{longplural}

\acrfull Full form of the acronym.
6765 \newrobustcmd{\acrfull}{\gls@hyp@opt\ns@acrfull}

6766 \newcommand*\ns@acrfull[2][]{%
6767   \new@ifnextchar[\{\gls@acrfull{\#1}{\#2}\}%
6768     {\gls@acrfull{\#1}{\#2}}[]\}%
6769 }

\@acrfull Low-level macro:
6770 \def\@acrfull#1#2[#3]{%
  Make it easier for acronym styles to change this:
6771   \acrfullfmt{\#1}{\#2}{\#3}%
6772 }

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.
6773 \newcommand{\acrfullfmt}[3]{%
6774   \acrlinkfullformat{\acrlong}{\acrshort}{\#1}{\#2}{\#3}%
6775 }

\linkfullformat Format for full links like \acrfull. Syntax: \acrlinkfullformat{\<long cs>}{\<short cs>}{\<options>}{\<label>}{\<insert>}
6776 \newcommand{\acrlinkfullformat}[5]{%
6777   \acrfullformat{\#1}{\#3}{\#4}{\#5}{\#2}{\#3}{\#4}[]\}%
6778 }

\acrfullformat Default full form is \<long> (\<short>).
6779 \newcommand{\acrfullformat}[2]{\#1\glsspace{\#2}\}

\glsspace Robust space to ensure it's written to the .glsdefs file.
6780 \newrobustcmd{\glsspace}{\space}

Default format for full acronym

\Acrfull
6781 \newrobustcmd{\Acrfull}{\gls@hyp@opt\ns@Acrfull}

6782 \newcommand*\ns@Acrfull[2][]{%
6783   \new@ifnextchar[\{\gls@Acrfull{\#1}{\#2}\}%
6784     {\gls@Acrfull{\#1}{\#2}}[]\}%
6785 }

```

Low-level macro:

```
6786 \def\@Acrfull#1#2[#3]{%
```

    Make it easier for acronym styles to change this:

```
6787   \Acrfullfmt{#1}{#2}{#3}%
6788 }
```

\Acrfullfmt First letter upper case full format.

```
6789 \newcommand*\Acrfullfmt[3]{%
6790   \acrlinkfullformat{\@Acrlong}{\acrshort}{#1}{#2}{#3}%
6791 }
```

\ACRfull

```
6792 \newrobustcmd*\ACRfull{\gls@hyp@opt\ns@ACRfull}
6793 \newcommand*\ns@ACRfull[2][]{%
6794   \new@ifnextchar[\{\@ACRfull{#1}{#2}\}%
6795     {\@ACRfull{#1}{#2}}[]\}%
6796 }
```

Low-level macro:

```
6797 \def\@ACRfull#1#2[#3]{%
```

    Make it easier for acronym styles to change this:

```
6798   \ACRfullfmt{#1}{#2}{#3}%
6799 }
```

\ACRfullfmt All upper case full format.

```
6800 \newcommand*\ACRfullfmt[3]{%
6801   \acrlinkfullformat{\@ACRlong}{\@ACRshort}{#1}{#2}{#3}%
6802 }
```

Plural:

\acrfullpl

```
6803 \newrobustcmd*\acrfullpl{\gls@hyp@opt\ns@acrfullpl}
6804 \newcommand*\ns@acrfullpl[2][]{%
6805   \new@ifnextchar[\{\@acrfullpl{#1}{#2}\}%
6806     {\@acrfullpl{#1}{#2}}[]\}%
6807 }
```

Low-level macro:

```
6808 \def\@acrfullpl#1#2[#3]{%
```

    Make it easier for acronym styles to change this:

```
6809   \acrfullplfmt{#1}{#2}{#3}%
6810 }
```

```

\acrfullplfmt No case change plural full format.
6811 \newcommand*\acrfullplfmt[3]{%
6812   \acrlinkfullformat{\acrlongpl}{\acrshortpl}{#1}{#2}{#3}%
6813 }

\Acrfullpl
6814 \newrobustcmd*\Acrfullpl{\gls@hyp@opt\ns@Acrfullpl}

6815 \newcommand*\ns@Acrfullpl[2][]{%
6816   \new@ifnextchar[\{@Acrfullpl{#1}{#2}\}%
6817     {\@Acrfullpl{#1}{#2}[]}%
6818 }

Low-level macro:
6819 \def\@Acrfullpl#1#2[#3]{%
  Make it easier for acronym styles to change this:
6820   \Acrfullplfmt{#1}{#2}{#3}%
6821 }

\Acrfullplfmt First letter upper case plural full format.
6822 \newcommand*\Acrfullplfmt[3]{%
6823   \acrlinkfullformat{\acrlongpl}{\acrshortpl}{#1}{#2}{#3}%
6824 }

\ACRfullpl
6825 \newrobustcmd*\ACRfullpl{\gls@hyp@opt\ns@ACRfullpl}

6826 \newcommand*\ns@ACRfullpl[2][]{%
6827   \new@ifnextchar[\{@ACRfullpl{#1}{#2}\}%
6828     {\@ACRfullpl{#1}{#2}[]}%
6829 }

Low-level macro:
6830 \def\@ACRfullpl#1#2[#3]{%
  Make it easier for acronym styles to change this:
6831   \ACRfullplfmt{#1}{#2}{#3}%
6832 }

\ACRfullplfmt All upper case plural full format.
6833 \newcommand*\ACRfullplfmt[3]{%
6834   \acrlinkfullformat{\ACRlongpl}{\ACRshortpl}{#1}{#2}{#3}%
6835 }

```

## 1.18 Predefined acronym styles

\acronymfont This is only used with the additional acronym styles:

6836 \newcommand{\acronymfont}[1]{#1}

\firstacronymfont This is only used with the additional acronym styles:

6837 \newcommand{\firstacronymfont}[1]{\acronymfont{#1}}

\acrnameformat The styles that allow an additional description use \acrnameformat{\short}{\long} to determine what information is displayed in the name.

6838 \newcommand\*{\acrnameformat}[2]{\acronymfont{#1}}

Define some tokens used by \newacronym:

\glskeylisttok

6839 \newtoks\glskeylisttok

\glslabeltok

6840 \newtoks\glslabeltok

\glsshorttok

6841 \newtoks\glsshorttok

\glslongtok

6842 \newtoks\glslongtok

\newacronymhook Provide a hook for \newacronym:

6843 \newcommand\*{\newacronymhook}{}%

\genericNewAcronym New improved version of setting the acronym style.

6844 \newcommand\*{\SetGenericNewAcronym}{}%

Change the behaviour of \Glsentryname to workaround expansion issues that cause a problem for \makefirstuc

6845 \let\@Gls@entryname\@Gls@acrentryname

Change the way acronyms are defined:

6846 \renewcommand{\newacronym}[4][]{%

6847 \ifdefempty{\@glsacronymlists}{%

6848 {%

6849 \def\@glo@type{\acronymtype}{%

6850 \setkeys{glossentry}{##1}{%

6851 \DeclareAcronymList{\@glo@type}{%

6852 }%

6853 {}{%

6854 \glskeylisttok{##1}{%

6855 \glslabeltok{##2}{%

6856 \glsshorttok{##3}{%

6857 \glslongtok{##4}{%

```

6858 \newacronymhook
6859 \protected@edef\@do@newglossaryentry{%
6860   \noexpand\newglossaryentry{\the\glslabeltok}%
6861   {%
6862     type=\acronymtype,%
6863     name={\expandonce{\acronymentry{##2}}},%
6864     sort={\acronymsort{\the\glsshorttok}{\the\glslongtok}},%
6865     text={\the\glsshorttok},%
6866     short={\the\glsshorttok},%
6867     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6868     long={\the\glslongtok},%
6869     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6870     \GenericAcronymFields,%
6871     \the\glskeylisttok
6872   }%
6873 }%
6874 \@do@newglossaryentry
6875 }%

```

Make sure that \acrfull etc reflects the new style:

```

6876 \renewcommand*{\acrfullfmt}[3]{%
6877   \glslink[##1]{##2}{\genacrfullformat{##2}{##3}}}%
6878 \renewcommand*{\Acrfullfmt}[3]{%
6879   \glslink[##1]{##2}{\Genacrfullformat{##2}{##3}}}%
6880 \renewcommand*{\ACRfullfmt}[3]{%
6881   \glslink[##1]{##2}{%
6882     \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}}%
6883 \renewcommand*{\acrfullplfmt}[3]{%
6884   \glslink[##1]{##2}{\genplacrfullformat{##2}{##3}}}%
6885 \renewcommand*{\Acrfullplfmt}[3]{%
6886   \glslink[##1]{##2}{\Genplacrfullformat{##2}{##3}}}%
6887 \renewcommand*{\ACRfullplfmt}[3]{%
6888   \glslink[##1]{##2}{%
6889     \mfirstucMakeUppercase{\genplacrfullformat{##2}{##3}}}}%

```

Make sure that \glsentryfull etc reflects the new style:

```

6890 \renewcommand*{\glsentryfull}[1]{\genacrfullformat{##1}{}}%
6891 \renewcommand*{\Glsentryfull}[1]{\Genacrfullformat{##1}{}}%
6892 \renewcommand*{\glsentryfullpl}[1]{\genplacrfullformat{##1}{}}%
6893 \renewcommand*{\Glsentryfullpl}[1]{\Genplacrfullformat{##1}{}}%
6894 }

```

**icAcronymFields** Fields used by \SetGenericNewAcronym that can be changed by the acronym style.

```
6895 \newcommand*{\GenericAcronymFields}{description={\the\glslongtok}}
```

\acronymentry{\label}
-----------------------

Display style for the name field in the list of acronyms.

```
6896 \newcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{#1}}}
```

```
\acronymsort \acronymsort{\short}{\long}
```

Default sort format for acronyms.

```
6897 \newcommand*\acronymsort[2]{#1}
```

```
\setacronymstyle{\style name}
```

```
6898 \newcommand*\setacronymstyle[1]{%
6899   \ifcsundef{@glsacr@dispstyle@#1}%
6900   {%
6901     \PackageError{glossaries}{Undefined acronym style '#1'}{}%
6902   }%
6903   {%
6904     \ifdefempty{\@glsacronymlists}{%
6905       \%
6906       \DeclareAcronymList{\acronymtype}{%
6907         \%
6908       }%
6909       \SetGenericNewAcronym
6910       \GlsUseAcrStyleDefs{#1}%
6911       \@for\@gls@type:=\@glsacronymlists\do{%
6912         \def\glsentryfmt[\@gls@type]{\GlsUseAcrEntryDisplayStyle{#1}}%
6913       }%
6914     }%
6915   }%
6916 }
```

```
\newacronymstyle{\style name}{\entry format definition}{\display definitions}
```

Defines a new acronym style called *<style name>*.

```
6916 \newcommand*\newacronymstyle[3]{%
6917   \ifcsdef{@glsacr@dispstyle@#1}%
6918   {%
6919     \PackageError{glossaries}{Acronym style '#1' already exists}{}%
6920   }%
6921   {%
6922     \csdef{@glsacr@dispstyle@#1}{#2}%
6923     \csdef{@glsacr@styledefs@#1}{#3}%
6924   }%
6925 }
```

**newacronymstyle** Redefines the given acronym style.

```
6926 \newcommand*\renewacronymstyle[3]{%
6927   \ifcsdef{@glsacr@dispstyle@#1}%
6928   {%
6929     \csdef{@glsacr@dispstyle@#1}{#2}%
6930   }%
```

```

6930   \csdef{@glsacr@styledefs@#1}{#3}%
6931   }%
6932   {%
6933     \PackageError{glossaries}{Acronym style ‘#1’ doesn’t exist}{}%
6934   }%
6935 }

```

#### rEntryDispStyle

```
6936 \newcommand*{\GlsUseAcrEntryDispStyle}[1]{\csuse{@glsacr@dispstyle@#1}}
```

#### UseAcrStyleDefs

```
6937 \newcommand*{\GlsUseAcrStyleDefs}[1]{\csuse{@glsacr@styledefs@#1}}
```

Predefined acronym styles:

**long-short** *<long>* (*<short>*) acronym style.

```

6938 \newacronymstyle{long-short}{%
6939 }%

```

Check for long form in case this is a mixed glossary.

```

6940 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6941 }%
6942 {%
6943 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6944 \renewcommand*{\genacrfullformat}[2]{%
6945   \glsentrylong{##1}##2\space
6946   (\protect\firstacronymfont{\glsentryshort{##1}})}%
6947 }%
6948 \renewcommand*{\Genacrfullformat}[2]{%
6949   \Glsentrylong{##1}##2\space
6950   (\protect\firstacronymfont{\glsentryshort{##1}})}%
6951 }%
6952 \renewcommand*{\genplacrfullformat}[2]{%
6953   \glsentrylongpl{##1}##2\space
6954   (\protect\firstacronymfont{\glsentryshortpl{##1}})}%
6955 }%
6956 \renewcommand*{\Genplacrfullformat}[2]{%
6957   \Glsentrylongpl{##1}##2\space
6958   (\protect\firstacronymfont{\glsentryshortpl{##1}})}%
6959 }%
6960 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}})}%
6961 \renewcommand*{\acronymsort}[2]{##1}%
6962 \renewcommand*{\acronymfont}[1]{##1}%
6963 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
6964 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6965 }

```

**long-sp-short** Similar to the previous style but allows the space between the long and short form to be customized.

```

6966 \newacronymstyle{long-sp-short}%
6967 {%
    Check for long form in case this is a mixed glossary.
6968   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6969 }%
6970 {%
6971   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6972   \renewcommand*{\genacrfullformat}[2]{%
6973     \glsentrylong{\##1}\##2\glsacspace{\##1}%
6974     (\protect\firstacronymfont{\glsentryshort{\##1}})%
6975   }%
6976   \renewcommand*{\Genacrfullformat}[2]{%
6977     \Glsentrylong{\##1}\##2\glsacspace{\##1}%
6978     (\protect\firstacronymfont{\glsentryshort{\##1}})%
6979   }%
6980   \renewcommand*{\genplacrfullformat}[2]{%
6981     \glsentrylongpl{\##1}\##2\glsacspace{\##1}%
6982     (\protect\firstacronymfont{\glsentryshortpl{\##1}})%
6983   }%
6984   \renewcommand*{\Genplacrfullformat}[2]{%
6985     \Glsentrylongpl{\##1}\##2\glsacspace{\##1}%
6986     (\protect\firstacronymfont{\glsentryshortpl{\##1}})%
6987   }%
6988   \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{\##1}}}%
6989   \renewcommand*{\acronymsort}[2]{\##1}%
6990   \renewcommand*{\acronymfont}[1]{\##1}%
6991   \renewcommand*{\firstacronymfont}[1]{\acronymfont{\##1}}%
6992   \renewcommand*{\acrluralsuffix}{\glspluralsuffix}%
6993 }

```

`\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.

```

6994 \newcommand*{\glsacspace}[1]{%
6995   \settowidth{\dimen@}{(\firstacronymfont{\glsentryshort{\##1}})}%
6996   \ifdim\dimen@<3em\else\space\fi
6997 }

```

`short-long` *<short>* (*<long>*) acronym style.

```

6998 \newacronymstyle{short-long}%
6999 {%

```

Check for long form in case this is a mixed glossary.

```

7000   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
7001 }%
7002 {%
7003   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
7004   \renewcommand*{\genacrfullformat}[2]{%
7005     \protect\firstacronymfont{\glsentryshort{\##1}}\##2\space
7006     (\glsentrylong{\##1})%

```

```

7007 }%
7008 \renewcommand*{\Genacrfullformat}[2]{%
7009   \protect\firstacronymfont{\Glsentryshort{\##1}}##2\space
7010   (\glsentrylong{\##1})%
7011 }%
7012 \renewcommand*{\genplacrfullformat}[2]{%
7013   \protect\firstacronymfont{\glsentryshortpl{\##1}}##2\space
7014   (\glsentrylongpl{\##1})%
7015 }%
7016 \renewcommand*{\Genplacrfullformat}[2]{%
7017   \protect\firstacronymfont{\Glsentryshortpl{\##1}}##2\space
7018   (\glsentrylongpl{\##1})%
7019 }%

7020 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{\##1}}}%
7021 \renewcommand*{\acronymsort}[2]{\##1}%
7022 \renewcommand*{\acronymfont}[1]{\##1}%
7023 \renewcommand*{\firstacronymfont}[1]{\acronymfont{\##1}}%
7024 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
7025 }

```

`long-sc-short` *<long>* (`\textsc{<short>}`) acronym style.

```

7026 \newacronymstyle{long-sc-short}%
7027 {%
7028   \GlsUseAcrEntryDispStyle{long-short}%
7029 }%
7030 {%
7031   \GlsUseAcrStyleDefs{long-short}%
7032   \renewcommand{\acronymfont}[1]{\textsc{\##1}}%
7033   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7034 }

```

`long-sm-short` *<long>* (`\textsmaller{<short>}`) acronym style.

```

7035 \newacronymstyle{long-sm-short}%
7036 {%
7037   \GlsUseAcrEntryDispStyle{long-short}%
7038 }%
7039 {%
7040   \GlsUseAcrStyleDefs{long-short}%
7041   \renewcommand{\acronymfont}[1]{\textsmaller{\##1}}%
7042   \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
7043 }

```

`sc-short-long` *<short>* (`\textsc{<long>}`) acronym style.

```

7044 \newacronymstyle{sc-short-long}%
7045 {%
7046   \GlsUseAcrEntryDispStyle{short-long}%
7047 }%
7048 {%

```

```
7049 \GlsUseAcrStyleDefs{short-long}%
7050 \renewcommand{\acronymfont}[1]{\textsc{##1}}%
7051 \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7052 }
```

sm-short-long *<short>* (\textsmaller{<long>}) acronym style.

```
7053 \newacronymstyle{sm-short-long}%
7054 {%
7055 \GlsUseAcrEntryDispStyle{short-long}%
7056 }%
7057 {%
7058 \GlsUseAcrStyleDefs{short-long}%
7059 \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
7060 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
7061 }
```

long-short-desc *<long>* ({<short>}) acronym style that has an accompanying description (which the user needs to supply).

```
7062 \newacronymstyle{long-short-desc}%
7063 {%
7064 \GlsUseAcrEntryDispStyle{long-short}%
7065 }%
7066 {%
7067 \GlsUseAcrStyleDefs{long-short}%
7068 \renewcommand*{\GenericAcronymFields}{}%
7069 \renewcommand*{\acronymsort}[2]{##2}%
7070 \renewcommand*{\acronymentry}[1]{%
7071 \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
7072 }
```

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.

```
7073 \newacronymstyle{long-sp-short-desc}%
7074 {%
7075 \GlsUseAcrEntryDispStyle{long-sp-short}%
7076 }%
7077 {%
7078 \GlsUseAcrStyleDefs{long-sp-short}%
7079 \renewcommand*{\GenericAcronymFields}{}%
7080 \renewcommand*{\acronymsort}[2]{##2}%
7081 \renewcommand*{\acronymentry}[1]{%
7082 \glsentrylong{##1}\glsacspace{##1}(\acronymfont{\glsentryshort{##1}})}%
7083 }
```

g-sc-short-desc *<long>* (\textsc{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```
7084 \newacronymstyle{long-sc-short-desc}%
7085 {%
```

```

7086  \GlsUseAcrEntryDispStyle{long-sc-short}%
7087 }%
7088 {%
7089  \GlsUseAcrStyleDefs{long-sc-short}%
7090  \renewcommand*\{\GenericAcronymFields\}{}%
7091  \renewcommand*\{\acronymsort\}[2]{##2}%
7092  \renewcommand*\{\acronymentry\}[1]{%
7093    \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
7094 }

g-sm-short-desc  <long> (\textsmaller{<short>}) acronym style that has an accompanying description (which
the user needs to supply).
7095 \newacronymstyle{long-sm-short-desc}%
7096 {%
7097  \GlsUseAcrEntryDispStyle{long-sm-short}%
7098 }%
7099 {%
7100  \GlsUseAcrStyleDefs{long-sm-short}%
7101  \renewcommand*\{\GenericAcronymFields\}{}%
7102  \renewcommand*\{\acronymsort\}[2]{##2}%
7103  \renewcommand*\{\acronymentry\}[1]{%
7104    \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
7105 }

short-long-desc  <short> ({<long>}) acronym style that has an accompanying description (which the user needs
to supply).
7106 \newacronymstyle{short-long-desc}%
7107 {%
7108  \GlsUseAcrEntryDispStyle{short-long}%
7109 }%
7110 {%
7111  \GlsUseAcrStyleDefs{short-long}%
7112  \renewcommand*\{\GenericAcronymFields\}{}%
7113  \renewcommand*\{\acronymsort\}[2]{##2}%
7114  \renewcommand*\{\acronymentry\}[1]{%
7115    \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
7116 }

short-long-desc  <long> (\textsc{<short>}) acronym style that has an accompanying description (which the
user needs to supply).
7117 \newacronymstyle{sc-short-long-desc}%
7118 {%
7119  \GlsUseAcrEntryDispStyle{sc-short-long}%
7120 }%
7121 {%
7122  \GlsUseAcrStyleDefs{sc-short-long}%
7123  \renewcommand*\{\GenericAcronymFields\}{}%
7124  \renewcommand*\{\acronymsort\}[2]{##2}%
7125  \renewcommand*\{\acronymentry\}[1]{%

```

```
7126     \glsentrylong{\#\#1}\space (\acronymfont{\glsentryshort{\#\#1}}))}%
7127 }
```

**short-long-desc** *<long>* (*\textsmaller{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```
7128 \newacronymstyle{sm-short-long-desc}%
7129 {%
7130   \GlsUseAcrEntryDispStyle{sm-short-long}%
7131 }%
7132 {%
7133   \GlsUseAcrStyleDefs{sm-short-long}%
7134   \renewcommand*{\GenericAcronymFields}{}
7135   \renewcommand*{\acronymsort}[2]{##2}%
7136   \renewcommand*{\acronymentry}[1]{%
7137     \glsentrylong{\#\#1}\space (\acronymfont{\glsentryshort{\#\#1}})}%
7138 }
```

**dua** *<long>* only acronym style.

```
7139 \newacronymstyle{dua}%
7140 {%
```

Check for long form in case this is a mixed glossary.

```
7141 \ifdefempty{\glscustomtext}%
7142 {%
7143   \ifglshaslong{\glslabel}%
7144   {%
7145     \glsifplural
7146   {%
```

Plural form:

```
7147   \glscapscase
7148 {%
```

Plural form, don't adjust case:

```
7149   \glsentrylongpl{\glslabel}\glsinsert
7150 {%
7151 {%
```

Plural form, make first letter upper case:

```
7152   \Glsentrylongpl{\glslabel}\glsinsert
7153 {%
7154 {%
```

Plural form, all caps:

```
7155   \mfirstucMakeUppercase
7156   {\glsentrylongpl{\glslabel}\glsinsert}%
7157 {%
7158 {%
7159 {%
```

## Singular form

```
7160      \glscapscase
7161      {%
```

Singular form, don't adjust case:

```
7162      \glsentrylong{\glslabel}\glsinsert
7163      }%
7164      {%
```

Subsequent singular form, make first letter upper case:

```
7165      \Glsentrylong{\glslabel}\glsinsert
7166      }%
7167      {%
```

Subsequent singular form, all caps:

```
7168      \mfirstucMakeUppercase
7169      {\glsentrylong{\glslabel}\glsinsert}%
7170      }%
7171      }%
7172      }%
7173      {%
```

Not an acronym:

```
7174      \glsgenentryfmt
7175      }%
7176      }%
7177      {\glscustomtext\glsinsert}%
7178 }%
7179 {%
7180 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
7181 \renewcommand*{\acrfullfmt}[3]{%
7182     \glslink[##1]{##2}{\glsentrylong{##2}##3\space
7183     (\acronymfont{\glsentryshort{##2}})}}%
7184 \renewcommand*{\Acrfullfmt}[3]{%
7185     \glslink[##1]{##2}{\Glsentrylong{##2}##3\space
7186     (\acronymfont{\glsentryshort{##2}})}}%
7187 \renewcommand*{\ACRfullfmt}[3]{%
7188     \glslink[##1]{##2}{%
7189         \mfirstucMakeUppercase{\glsentrylong{##2}##3\space
7190         (\acronymfont{\glsentryshort{##2}})}}}%
7191 \renewcommand*{\acrfullplfmt}[3]{%
7192     \glslink[##1]{##2}{\glsentrylongpl{##2}##3\space
7193     (\acronymfont{\glsentryshortpl{##2}})}}%
7194 \renewcommand*{\Acrfullplfmt}[3]{%
7195     \glslink[##1]{##2}{\Glsentrylongpl{##2}##3\space
7196     (\acronymfont{\glsentryshortpl{##2}})}}%
7197 \renewcommand*{\ACRfullplfmt}[3]{%
7198     \glslink[##1]{##2}{%
```

```

7199      \mfirstucMakeUppercase{\glsentrylongpl{##2}##3\space
7200          (\acronymfont{\glsentryshortpl{##2}})}%}
7201  \renewcommand*{\glsentryfull}[1]{%
7202      \glsentrylong{##1}\space(\acronymfont{\glsentryshort{##1}})%
7203  }%
7204  \renewcommand*{\Glsentryfull}[1]{%
7205      \Glsentrylong{##1}\space(\acronymfont{\glsentryshort{##1}})%
7206  }%
7207  \renewcommand*{\glsentryfullpl}[1]{%
7208      \glsentrylongpl{##1}\space(\acronymfont{\glsentryshortpl{##1}})%
7209  }%
7210  \renewcommand*{\Glsentryfullpl}[1]{%
7211      \Glsentrylongpl{##1}\space(\acronymfont{\glsentryshortpl{##1}})%
7212  }%
7213  \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
7214  \renewcommand*{\acronymsort}[2]{##1}%
7215  \renewcommand*{\acronymfont}[1]{##1}%
7216  \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
7217 }

```

**dua-desc** <*long*> only acronym style with user-supplied description.

```

7218 \newacronymstyle{dua-desc}%
7219 {%
7220   \GlsUseAcrEntryDispStyle{dua}%
7221 }%
7222 {%
7223   \GlsUseAcrStyleDefs{dua}%
7224   \renewcommand*{\GenericAcronymFields}{}%
7225   \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentrylong{##1}}}%
7226   \renewcommand*{\acronymsort}[2]{##2}%
7227 }%

```

**footnote** <*short*>\footnote{<*long*>} acronym style.

```

7228 \newacronymstyle{footnote}%
7229 {%
    Check for long form in case this is a mixed glossary.
7230   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
7231 }%
7232 {%
7233   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%

```

Need to ensure hyperlinks are switched off on first use:

```

7234   \glshyperfirstfalse
7235   \renewcommand*{\genacrfullformat}[2]{%
7236       \protect\firstacronymfont{\glsentryshort{##1}}##2%
7237       \protect\footnote{\glsentrylong{##1}}%
7238   }%
7239   \renewcommand*{\Genacrfullformat}[2]{%

```

```

7240 \firstacronymfont{\Glsentryshort{##1}}##2%
7241 \protect\footnote{\glsentrylong{##1}}%
7242 }%
7243 \renewcommand*\genplacrfullformat[2]{%
7244 \protect\firstacronymfont{\glsentryshortpl{##1}}##2%
7245 \protect\footnote{\glsentrylongpl{##1}}%
7246 }%
7247 \renewcommand*\Genplacrfullformat[2]{%
7248 \protect\firstacronymfont{\Glsentryshortpl{##1}}##2%
7249 \protect\footnote{\glsentrylongpl{##1}}%
7250 }%
7251 \renewcommand*\acronymentry[1]{\acronymfont{\glsentryshort{##1}}}}%
7252 \renewcommand*\acronymsort[2]{##1}%
7253 \renewcommand*\acronymfont[1]{##1}%
7254 \renewcommand*\acrpluralsuffix{\glsacrpluralsuffix}%

```

Don't use footnotes for \acrfull:

```

7255 \renewcommand*\acrfullfmt[3]{%
7256 \glslink[##1]{##2}{\acronymfont{\glsentryshort{##2}}##3\space
7257 (\glsentrylong{##2})}}%
7258 \renewcommand*\Acrfullfmt[3]{%
7259 \glslink[##1]{##2}{\acronymfont{\Glsentryshort{##2}}##3\space
7260 (\glsentrylong{##2})}}%
7261 \renewcommand*\ACRfullfmt[3]{%
7262 \glslink[##1]{##2}{%
7263 \mfirstucMakeUppercase{\acronymfont{\glsentryshort{##2}}##3\space
7264 (\glsentrylong{##2})}}%
7265 \renewcommand*\acrfullplfmt[3]{%
7266 \glslink[##1]{##2}{\acronymfont{\glsentryshortpl{##2}}##3\space
7267 (\glsentrylongpl{##2})}}%
7268 \renewcommand*\Acrfullplfmt[3]{%
7269 \glslink[##1]{##2}{\acronymfont{\Glsentryshortpl{##2}}##3\space
7270 (\glsentrylongpl{##2})}}%
7271 \renewcommand*\ACRfullplfmt[3]{%
7272 \glslink[##1]{##2}{%
7273 \mfirstucMakeUppercase{\acronymfont{\glsentryshortpl{##2}}##3\space
7274 (\glsentrylongpl{##2})}}%

```

Similarly for \glsentryfull etc:

```

7275 \renewcommand*\glsentryfull[1]{%
7276 \acronymfont{\glsentryshort{##1}}\space(\glsentrylong{##1})}}%
7277 \renewcommand*\Glsentryfull[1]{%
7278 \acronymfont{\Glsentryshort{##1}}\space(\glsentrylong{##1})}}%
7279 \renewcommand*\glsentryfullpl[1]{%
7280 \acronymfont{\glsentryshortpl{##1}}\space(\glsentrylongpl{##1})}}%
7281 \renewcommand*\Glsentryfullpl[1]{%
7282 \acronymfont{\Glsentryshortpl{##1}}\space(\glsentrylongpl{##1})}}%
7283 }

```

`footnote-sc \textsc{<short>}\footnote{<long>} acronym style.`

```

7284 \newacronymstyle{footnote-sc}%
7285 {%
7286   \GlsUseAcrEntryDispStyle{footnote}%
7287 }%
7288 {%
7289   \GlsUseAcrStyleDefs{footnote}%
7290   \renewcommand{\acronymentry}[1]{\acronymfont{\glsentryshort{\##1}}}
7291   \renewcommand{\acronymfont}[1]{\textsc{\##1}}%
7292   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7293 }%
footnote-sm  \textsmaller{\langle short \rangle}\footnote{\langle long \rangle} acronym style.
7294 \newacronymstyle{footnote-sm}%
7295 {%
7296   \GlsUseAcrEntryDispStyle{footnote}%
7297 }%
7298 {%
7299   \GlsUseAcrStyleDefs{footnote}%
7300   \renewcommand{\acronymentry}[1]{\acronymfont{\glsentryshort{\##1}}}
7301   \renewcommand{\acronymfont}[1]{\textsmaller{\##1}}%
7302   \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
7303 }%
footnote-desc  \langle short \rangle\footnote{\langle long \rangle} acronym style that has an accompanying description (which the user needs to supply).
7304 \newacronymstyle{footnote-desc}%
7305 {%
7306   \GlsUseAcrEntryDispStyle{footnote}%
7307 }%
7308 {%
7309   \GlsUseAcrStyleDefs{footnote}%
7310   \renewcommand*{\GenericAcronymFields}{}%
7311   \renewcommand*{\acronymsort}[2]{\##2}%
7312   \renewcommand*{\acronymentry}[1]{%
7313     \glsentrylong{\##1}\space (\acronymfont{\glsentryshort{\##1}})}%
7314 }%
footnote-sc-desc  \textsc{\langle short \rangle}\footnote{\langle long \rangle} acronym style that has an accompanying description (which the user needs to supply).
7315 \newacronymstyle{footnote-sc-desc}%
7316 {%
7317   \GlsUseAcrEntryDispStyle{footnote-sc}%
7318 }%
7319 {%
7320   \GlsUseAcrStyleDefs{footnote-sc}%
7321   \renewcommand*{\GenericAcronymFields}{}%
7322   \renewcommand*{\acronymsort}[2]{\##2}%
7323   \renewcommand*{\acronymentry}[1]{%
7324     \glsentrylong{\##1}\space (\acronymfont{\glsentryshort{\##1}})}%

```

```

7325 }

ootnote-sm-desc \textsmaller{\short}\footnote{\long} acronym style that has an accompanying de-
scription (which the user needs to supply).
7326 \newacronymstyle{footnote-sm-desc}%
7327 {%
7328   \GlsUseAcrEntryDispStyle{footnote-sm}%
7329 }%
7330 {%
7331   \GlsUseAcrStyleDefs{footnote-sm}%
7332   \renewcommand*\GenericAcronymFields{}%
7333   \renewcommand*\acronymsort}[2]{##2}%
7334   \renewcommand*\acronymentry}[1]{%
7335     \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
7336 }

```

#### AcronymSynonyms

```
7337 \newcommand*\DefineAcronymSynonyms}{%
```

Short form

```
\acs
7338 \let\acs\acrshort
```

First letter uppercase short form

```
\Acs
7339 \let\Acs\Acrshort
```

Plural short form

```
\acsp
7340 \let\acsp\acrshortpl
```

First letter uppercase plural short form

```
\Acsp
7341 \let\Acsp\Acrshortpl
```

Long form

```
\acl
7342 \let\acl\acrlong
```

Plural long form

```
\aclp
7343 \let\aclp\acrlongpl
```

First letter upper case long form

```

\Acl
7344 \let\Acl\Acrlong

First letter upper case plural long form

\Aclp
7345 \let\Aclp\Acrlongpl

Full form

\acf
7346 \let\acf\acrfull

Plural full form

\acfp
7347 \let\acfp\acrfullpl

First letter upper case full form

\Acf
7348 \let\Acf\Acrfull

First letter upper case plural full form

\Acfp
7349 \let\Acfp\Acrfullpl

Standard form

\ac
7350 \let\ac\gls

First upper case standard form

\Ac
7351 \let\Ac\Gls

Standard plural form

\ACP
7352 \let\ACP\Glspl

Standard first letter upper case plural form

\Acp
7353 \let\Acp\Glspl

7354 }

Define synonyms if required
7355 \ifglssacrshortcuts
7356 \DefineAcronymSynonyms
7357 \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.

```
7358 \newcommand*{\SetDefaultAcronymDisplayStyle}[1]{%
7359   \def\glsentryfmt[#1]{\glsentryfmt}%
7360 }
```

`ltNewAcronymDef` Sets up the acronym definition for the default style. The information is provided by the tokens `\glslabeltok`, `\glsshorttok`, `\glslongtok` and `\glskeylisttok`.

```
7361 \newcommand*{\DefaultNewAcronymDef}{%
7362   \edef\@do@newglossaryentry{%
7363     \noexpand\newglossaryentry{\the\glslabeltok}%
7364     {%
7365       type=\acronymtype,%
7366       name={\the\glsshorttok},%
7367       sort={\the\glsshorttok},%
7368       text={\the\glsshorttok},%
7369       first=\acrfullformat{\the\glslongtok}{\the\glsshorttok},%
7370       plural=\noexpand\expandonce\noexpand\@glo@shortpl},%
7371       firstplural=\acrfullformat{\noexpand\expandonce\noexpand\@glo@longpl}%
7372         {\noexpand\expandonce\noexpand\@glo@shortpl},%
7373       short={\the\glsshorttok},%
7374       shortplural=\the\glsshorttok\noexpand\acrpluralsuffix},%
7375       long={\the\glslongtok},%
7376       longplural=\the\glslongtok\noexpand\acrpluralsuffix},%
7377       description={\the\glslongtok},%
7378       descriptionplural=\noexpand\expandonce\noexpand\@glo@longpl},%
```

Remaining options specified by the user:

```
7379   \the\glskeylisttok
7380   }%
7381 }%
7382 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7383 \let\@org@gls@assign@plural\gls@assign@plural
7384 \let\@org@gls@assign@descplural\gls@assign@descplural
7385 \def\gls@assign@firstpl##1##2{%
7386   \@@gls@expand@field{##1}{firstpl}{##2}%
7387 }%
7388 \def\gls@assign@plural##1##2{%
7389   \@@gls@expand@field{##1}{plural}{##2}%
7390 }%
7391 \def\gls@assign@descplural##1##2{%
7392   \@@gls@expand@field{##1}{descplural}{##2}%
7393 }%
7394 \do@newglossaryentry
7395 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7396 \let\gls@assign@plural\@org@gls@assign@plural
7397 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7398 }
```

**ultAcronymStyle** Set up the default acronym style:

```
7399 \newcommand*{\SetDefaultAcronymStyle}{%
```

    Set the display style:

```
7400   \cfor@\gls@type:=\glsacronymlists\do{%
7401     \SetDefaultAcronymDisplayStyle{\gls@type}%
7402   }%
```

    Set up the definition of \newacronym:

```
7403   \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).

```
7404   \ifx\glsacronymlists\empty
7405     \def\glo@type{\acronymtype}%
7406     \setkeys{glossentry}{##1}%
7407     \DeclareAcronymList{\glo@type}%
7408     \SetDefaultAcronymDisplayStyle{\glo@type}%
7409   \fi
7410   \glskeylisttok{##1}%
7411   \glslabeltok{##2}%
7412   \glsshorttok{##3}%
7413   \glslongtok{##4}%
7414   \newacronymhook
7415   \DefaultNewAcronymDef
7416 }%
7417 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
7418 }
```

**\acrfootnote** Used by the footnote acronym styles.

```
7419 \newcommand*{\acrfootnote}[3]{\acrlinkfootnote{#1}{#2}{#3}}
```

**acrlinkfootnote**

```
7420 \newcommand*{\acrlinkfootnote}[3]{%
7421   \footnote{\glslink[#1]{#2}{#3}}%
7422 }
```

**rnolinkfootnote**

```
7423 \newcommand*{\acrnlkfootnote}[3]{%
7424   \footnote{#3}%
7425 }
```

**nymDisplayStyle** Sets the acronym display style for given glossary for the description and footnote combination.

```
7426 \newcommand*{\SetDescriptionFootnoteAcronymDisplayStyle}[1]{%
7427   \def\glsentryfmt[#1]{%
7428     \ifdefempty\glscustomtext
7429     {%
7430       \if\glsused{\glslabel}%

```

```

7431  {%
7432    \acronymfont{\glsgenentryfmt}%
7433  }%
7434  {%
7435    \firstacronymfont{\glsgenentryfmt}%
7436    \ifglshassymbol{\glslabel}%
7437    {%
7438      \expandafter\protect\expandafter\acrfootnote\expandafter
7439      {\@gls@link@opts}{\@gls@link@label}%
7440      {%
7441        \glsifplural
7442        {\glsentrysymbolplural{\glslabel}}%
7443        {\glsentrysymbol{\glslabel}}%
7444      }%
7445    }%
7446  }%
7447 }%
7448 {\glscustomtext\glsinsert}%
7449 }%
7450 }

```

#### teNewAcronymDef

```

7451 \newcommand*{\DescriptionFootnoteNewAcronymDef}{%
7452   \edef\@do@newglossaryentry{%
7453     \noexpand\newglossaryentry{\the\glslabeltok}%
7454   }%
7455   type=\acronymtype,%
7456   name={\noexpand\acronymfont{\the\glsshorttok}},%
7457   sort={\the\glsshorttok},%
7458   first={\the\glsshorttok},%
7459   firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7460   text={\the\glsshorttok},%
7461   plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7462   short={\the\glsshorttok},%
7463   shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7464   long={\the\glslongtok},%
7465   longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7466   symbol={\the\glslongtok},%
7467   symbolplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7468   \the\glskeylisttok
7469 }%
7470 }%
7471 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7472 \let\@org@gls@assign@plural\gls@assign@plural
7473 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7474 \def\gls@assign@firstpl##1##2{%
7475   \@@gls@expand@field{##1}{firstpl}{##2}%
7476 }%
7477 \def\gls@assign@plural##1##2{%

```

```

7478     \@@gls@expand@field{##1}{plural}{##2}%
7479   }%
7480   \def\gls@assign@symbolplural##1##2{%
7481     \@@gls@expand@field{##1}{symbolplural}{##2}%
7482   }%
7483   \do@newglossaryentry
7484   \let\gls@assign@plural\org@gls@assign@plural
7485   \let\gls@assign@firstpl\org@gls@assign@firstpl
7486   \let\gls@assign@symbolplural\org@gls@assign@symbolplural
7487 }

```

`noteAcronymStyle` 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.

```

7488 \newcommand*{\SetDescriptionFootnoteAcronymStyle}{%
7489   \renewcommand{\newacronym}[4][]{%
7490     \ifx\glsacronymlists\empty
7491       \def\@glo@type{\acronymtype}%
7492       \setkeys{glossentry}{##1}%
7493       \DeclareAcronymList{\@glo@type}%
7494       \SetDescriptionFootnoteAcronymDisplayStyle{\@glo@type}%
7495     \fi
7496     \glskeylisttok{##1}%
7497     \glslabeltok{##2}%
7498     \glsshorttok{##3}%
7499     \glslongtok{##4}%
7500     \newacronymhook
7501     \DescriptionFootnoteNewAcronymDef
7502   }%

```

If footnote package option is specified, set the first use to append the long form (stored in symbol) as a footnote.

```

7503   \for@\gls@type:=\glsacronymlists\do{%
7504     \SetDescriptionFootnoteAcronymDisplayStyle{\@gls@type}%
7505   }%

```

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.

```

7506   \ifglsacrsmalls
7507     \renewcommand*{\acronymfont}[1]{\textsc{##1}}%
7508     \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7509   \else
7510     \ifglsacrmaller
7511       \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
7512     \fi
7513   \fi

```

Check for package option clash

```

7514 \ifglsacrdua
7515   \PackageError{glossaries}{Option clash: 'footnote' and 'dua'
7516   can't both be set}{}
7517 \fi
7518 }%

```

nymDisplayStyle Sets the acronym display style for given glossary with description and dua combination.

```

7519 \newcommand*{\SetDescriptionDUAAcronymDisplayStyle}[1]{%
7520   \def\glsentryfmt[#1]{\glsgenentryfmt}%
7521 }

```

UANewAcronymDef

```

7522 \newcommand*{\DescriptionDUANewAcronymDef}{%
7523   \edef\@do@newglossaryentry{%
7524     \noexpand\newglossaryentry{\the\glslabeltok}%
7525     {%
7526       type=\acronymtype,%
7527       name={\the\glslongtok},%
7528       sort={\the\glslongtok},%
7529       text={\the\glslongtok},%
7530       first={\the\glslongtok},%
7531       plural={\noexpand\expandonce\noexpand\@glo@longpl},%
7532       firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7533       short={\the\glsshorttok},%
7534       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7535       long={\the\glslongtok},%
7536       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7537       symbol={\the\glsshorttok},%
7538       symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7539       \the\glskeylisttok
7540     }%
7541   }%
7542   \let\@org@gls@assign@firstpl\gls@assign@firstpl
7543   \let\@org@gls@assign@plural\gls@assign@plural
7544   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7545   \def\gls@assign@firstpl##1##2{%
7546     \@@gls@expand@field{##1}{firstpl}{##2}%
7547   }%
7548   \def\gls@assign@plural##1##2{%
7549     \@@gls@expand@field{##1}{plural}{##2}%
7550   }%
7551   \def\gls@assign@symbolplural##1##2{%
7552     \@@gls@expand@field{##1}{symbolplural}{##2}%
7553   }%
7554   \@do@newglossaryentry
7555   \let\gls@assign@firstpl\@org@gls@assign@firstpl
7556   \let\gls@assign@plural\@org@gls@assign@plural
7557   \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7558 }

```

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.

```
7559 \newcommand*{\SetDescriptionDUAAcronymStyle}{%
7560   \ifglsacrsmallicaps
7561     \PackageError{glossaries}{Option clash: 'smallcaps' and 'dua'
7562       can't both be set}{}%
7563   \else
7564     \ifglsacrsmailler
7565       \PackageError{glossaries}{Option clash: 'smaller' and 'dua'
7566         can't both be set}{}%
7567   \fi
7568 \fi
7569 \renewcommand{\newacronym}[4][]{%
7570   \ifx\@glsacronymlists\@empty
7571     \def\@glo@type{\acronymtype}%
7572     \setkeys{glossentry}{##1}%
7573     \DeclareAcronymList{\@glo@type}%
7574     \SetDescriptionDUAAcronymDisplayStyle{\@glo@type}%
7575   \fi
7576   \glskeylisttok{##1}%
7577   \glslabeltok{##2}%
7578   \glsshorttok{##3}%
7579   \glslongtok{##4}%
7580   \newacronymhook
7581   \DescriptionDUANewAcronymDef
7582 }%
```

Set display.

```
7583 \c@for\@gls@type:=\@glsacronymlists\do{%
7584   \SetDescriptionDUAAcronymDisplayStyle{\@gls@type}%
7585 }%
7586 }%
```

AcronymDisplayStyle Sets the acronym display style for given glossary using the description setting (but not footnote or dua).

```
7587 \newcommand*{\SetDescriptionAcronymDisplayStyle}[1]{%
7588   \def\glsentryfmt[#1]{%
7589     \ifdef\glscustomtext
7590       %
7591       \ifglsused{\glslabel}%
7592       %
```

Move the inserted text outside of \acronymfont

```
7593     \let\gls@org@insert\glsinsert
7594     \let\glsinsert\@empty
7595     \acronymfont{\glsgenentryfmt}\gls@org@insert
7596   }%
```

```

7597   {%
7598     \glsgenentryfmt
7599     \ifglshassymbol{\glslabel}{%
7600       {%
7601         \glsifplural
7602         {%
7603           \def\@glo@symbol{\glsentrysymbolplural{\glslabel}}%
7604         }%
7605         {%
7606           \def\@glo@symbol{\glsentrysymbol{\glslabel}}%
7607         }%
7608         \space(\protect\firstacronymfont
7609           {\glscapscase
7610             {\@glo@symbol}
7611             {\@glo@symbol}
7612             {\mfirstucMakeUppercase{\@glo@symbol}}})%
7613       }%
7614       {}%
7615     }%
7616   }%
7617   {\glscustomtext\glsinsert}%
7618 }%
7619 }

```

#### onNewAcronymDef

```

7620 \newcommand*{\DescriptionNewAcronymDef}{%
7621   \edef\@do@newglossaryentry{%
7622     \noexpand\newglossaryentry{\the\glslabeltok}{%
7623       {%
7624         type=\acronymtype,%
7625         name={\noexpand
7626           \acrnameformat{\the\glsshorttok}{\the\glslongtok}},%
7627         sort={\the\glsshorttok},%
7628         first={\the\glslongtok},%
7629         firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7630         text={\the\glsshorttok},%
7631         plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7632         short={\the\glsshorttok},%
7633         shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7634         long={\the\glslongtok},%
7635         longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7636         symbol={\noexpand\@glo@text},%
7637         symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7638         \the\glskeylisttok}%
7639   }%
7640   \let\@org@gls@assign@firstpl\gls@assign@firstpl
7641   \let\@org@gls@assign@plural\gls@assign@plural
7642   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7643   \def\gls@assign@firstpl##1##2{%

```

```

7644     \@@gls@expand@field{##1}{firstpl}{##2}%
7645   }%
7646   \def\gls@assign@plural##1##2{%
7647     \@@gls@expand@field{##1}{plural}{##2}%
7648   }%
7649   \def\gls@assign@symbolplural##1##2{%
7650     \@@gls@expand@field{##1}{symbolplural}{##2}%
7651   }%
7652   \do@newglossaryentry
7653   \let\gls@assign@firstpl\org@gls@assign@firstpl
7654   \let\gls@assign@plural\org@gls@assign@plural
7655   \let\gls@assign@symbolplural\org@gls@assign@symbolplural
7656 }

```

`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.

```

7657 \newcommand*\SetDescriptionAcronymStyle{%
7658   \renewcommand{\newacronym}[4][]{%
7659     \ifx\@glsacronymlists\empty
7660       \def\@glo@type{\acronymtype}%
7661       \setkeys{glossentry}{##1}%
7662       \DeclareAcronymList{\@glo@type}%
7663       \SetDescriptionAcronymDisplayStyle{\@glo@type}%
7664     \fi
7665     \glskeylisttok{##1}%
7666     \glslabeltok{##2}%
7667     \glsshorttok{##3}%
7668     \glslongtok{##4}%
7669     \newacronymhook
7670     \DescriptionNewAcronymDef
7671   }%

```

Set display.

```

7672   \foreach\gls@type:=\glsacronymlists\do{%
7673     \SetDescriptionAcronymDisplayStyle{\gls@type}%
7674   }%

```

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.

```

7675   \ifglsacrsmallcaps
7676     \renewcommand{\acronymfont}[1]{\textsc{##1}}
7677     \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7678   \else
7679     \ifglsacrsmaller
7680       \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
7681     \fi
7682   \fi
7683 }%

```

`nymDisplayStyle` Sets the acronym display style for given glossary with footnote setting (but not description or dua).

```
7684 \newcommand*{\SetFootnoteAcronymDisplayStyle}[1]{%
7685   \def\glsentryfmt[#1]{%
7686     \ifempty{\glscustomtext}{%
7687       \Move the inserted text outside of \acronymfont{%
7688         \let\gls@org@insert\glsinsert
7689         \let\glsinsert\empty
7690         \ifglsused{\glslabel}{%
7691           \%
7692             \acronymfont{\glsgenentryfmt}\gls@org@insert
7693           }%
7694           \%
7695             \firstacronymfont{\glsgenentryfmt}\gls@org@insert
7696             \ifglslong{\glslabel}{%
7697               \%
7698                 \expandafter\protect\expandafter\acrfootnote\expandafter
7699                   {\@gls@link@opts}{\@gls@link@label}%
7700               \%
7701                 \glsifplural
7702                   {\glsentrylongpl{\glslabel}}%
7703                   {\glsentrylong{\glslabel}}%
7704               }%
7705             }%
7706             \%
7707           }%
7708         }%
7709         {\glscustomtext\glsinsert}%
7710       }%
7711 }
```

`teNewAcronymDef`

```
7712 \newcommand*{\FootnoteNewAcronymDef}{%
7713   \edef\@do@newglossaryentry{%
7714     \noexpand\newglossaryentry{\the\glslabeltok}%
7715     \%
7716       type=\acronymtype,%
7717       name={\noexpand\acronymfont{\the\glsshorttok}},%
7718       sort={\the\glsshorttok},%
7719       text={\the\glsshorttok},%
7720       plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7721       first={\the\glsshorttok},%
7722       firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7723       short={\the\glsshorttok},%
7724       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7725       long={\the\glslongtok},%
```

```

7726     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7727     description={\the\glslongtok},%
7728     descriptionplural={\noexpand\expandonce\noexpand@glo@longpl},%
7729     \the\glskeylisttok
7730   }%
7731 }%
7732 \let\@org@gls@assign@plural\gls@assign@plural
7733 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7734 \let\@org@gls@assign@descplural\gls@assign@descplural
7735 \def\gls@assign@firstpl##1##2{%
7736   \@@gls@expand@field{##1}{firstpl}{##2}%
7737 }%
7738 \def\gls@assign@plural##1##2{%
7739   \@@gls@expand@field{##1}{plural}{##2}%
7740 }%
7741 \def\gls@assign@descplural##1##2{%
7742   \@@gls@expand@field{##1}{descplural}{##2}%
7743 }%
7744 \@do@newglossaryentry
7745 \let\gls@assign@plural\@org@gls@assign@plural
7746 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7747 \let\gls@assign@descplural\@org@gls@assign@descplural
7748 }

```

`noteAcronymStyle` 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.

```

7749 \newcommand*\SetFootnoteAcronymStyle{%
7750   \renewcommand{\newacronym}[4][]{}%
7751   \ifx\@glsacronymlists\empty
7752     \def\@glo@type{\acronymtype}%
7753     \setkeys{glossentry}{##1}%
7754     \DeclareAcronymList{\@glo@type}%
7755     \SetFootnoteAcronymDisplayStyle{\@glo@type}%
7756   \fi
7757   \glskeylisttok{##1}%
7758   \glslabeltok{##2}%
7759   \glsshorttok{##3}%
7760   \glslongtok{##4}%
7761   \newacronymhook
7762   \FootnoteNewAcronymDef
7763 }%

```

Set display

```

7764 \cfor{@gls@type:=\glsacronymlists}{\do}{%
7765   \SetFootnoteAcronymDisplayStyle{@gls@type}%
7766 }%

```

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.

```

7767 \ifglsacrsmallsCaps

```

```

7768     \renewcommand*{\acronymfont}[1]{\textsc{##1}}%
7769     \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7770 \else
7771     \ifglsacrsmlller
7772         \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
7773     \fi
7774 \fi

    Check for option clash

7775 \ifglsacrdua
7776     \PackageError{glossaries}{Option clash: 'footnote' and 'dua'
7777     can't both be set}{}%
7778 \fi
7779 }%

```

`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.

```

7780 \DeclareRobustCommand*{\glsdoparenifnotempty}[2]{%
7781     \protected@edef\gls@tmp{#1}%
7782     \ifdefempty\gls@tmp
7783     {}%
7784     {%
7785         \ifx\gls@tmp\@gls@default@value
7786         \else
7787             \space (#2{#1})%
7788         \fi
7789     }%
7790 }

```

`nymDisplayStyle` Sets the acronym display style for given glossary where neither footnote nor description is required, but `smallcaps` or `smaller` specified.

```

7791 \newcommand*{\SetSmallAcronymDisplayStyle}[1]{%
7792     \def\glsentryfmt[#1]{%
7793         \ifdefempty\glscustomtext
7794         {}%

```

Move the inserted text outside of `\acronymfont`

```

7795     \let\gls@org@insert\glsinsert
7796     \let\glsinsert\@empty
7797     \ifglsused{\glslabel}%
7798     {%
7799         \acronymfont{\glsgenentryfmt}\gls@org@insert
7800     }%
7801     {%
7802         \glsgenentryfmt
7803         \ifglsassymbol{\glslabel}%
7804     }%
7805         \glsifplural
7806         {}%

```

```

7807         \def\@glo@symbol{\glsentrysymbolplural{\glslabel}}%
7808     }%
7809     {%
7810         \def\@glo@symbol{\glsentrysymbol{\glslabel}}%
7811     }%
7812     \space
7813     (\glscapscase
7814     {\firstacronymfont{\@glo@symbol}}%
7815     {\firstacronymfont{\@glo@symbol}}%
7816     {\firstacronymfont{\mfirstucMakeUppercase{\@glo@symbol}}})%
7817   }%
7818   {}%
7819 }%
7820 }%
7821 {\glscustomtext\glsinsert}%
7822 }%
7823 }

```

### llNewAcronymDef

```

7824 \newcommand*{\SmallNewAcronymDef}{%
7825   \edef\@do@newglossaryentry{%
7826     \noexpand\newglossaryentry{\the\glslabeltok}%
7827   }%
7828   type=\acronymtype,%
7829   name={\noexpand\acronymfont{\the\glsshorttok}},%
7830   sort={\the\glsshorttok},%
7831   text={\the\glsshorttok},%

```

Default to the short plural.

```

7832   plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7833   first={\the\glslongtok},%

```

Default to the long plural.

```

7834   firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7835   short={\the\glsshorttok},%
7836   shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7837   long={\the\glslongtok},%
7838   longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7839   description={\noexpand\@glo@first},%
7840   descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7841   symbol={\the\glsshorttok},%

```

Default to the short plural.

```

7842   symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7843   \the\glskeylisttok
7844 }%
7845 }%
7846 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7847 \let\@org@gls@assign@plural\gls@assign@plural
7848 \let\@org@gls@assign@descplural\gls@assign@descplural

```

```

7849 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7850 \def\gls@assign@firstpl##1##2{%
7851   \@@gls@expand@field{##1}{firstpl}{##2}%
7852 }%
7853 \def\gls@assign@plural##1##2{%
7854   \@@gls@expand@field{##1}{plural}{##2}%
7855 }%
7856 \def\gls@assign@descplural##1##2{%
7857   \@@gls@expand@field{##1}{descplural}{##2}%
7858 }%
7859 \def\gls@assign@symbolplural##1##2{%
7860   \@@gls@expand@field{##1}{symbolplural}{##2}%
7861 }%
7862 \do@newglossaryentry
7863 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7864 \let\gls@assign@plural\@org@gls@assign@plural
7865 \let\gls@assign@descplural\@org@gls@assign@descplural
7866 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7867 }

```

`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.

```

7868 \newcommand*\SetSmallAcronymStyle{%
7869   \renewcommand{\newacronym}[4][]{%
7870     \ifx\@glsacronymlists\@empty
7871       \def\@glo@type{\acronymtype}%
7872       \setkeys{glossentry}{##1}%
7873       \DeclareAcronymList{\@glo@type}%
7874       \SetSmallAcronymDisplayStyle{\@glo@type}%
7875     \fi
7876     \glskeylisttok{##1}%
7877     \glslabeltok{##2}%
7878     \glsshorttok{##3}%
7879     \glslongtok{##4}%
7880     \newacronymhook
7881     \SmallNewAcronymDef
7882   }%

```

Change the display since `first` only contains long form.

```

7883 \cfor\@gls@type:=\glsacronymlists\do{%
7884   \SetSmallAcronymDisplayStyle{\@gls@type}%
7885 }%

```

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.

```

7886 \ifglsacrmallcaps
7887   \renewcommand*\acronymfont[1]{\textsc{##1}}
7888   \renewcommand*\acrpluralsuffix{\glsupacrpluralsuffix}%
7889 \else
7890   \renewcommand*\acronymfont[1]{\textsmaller{##1}}

```

```

7891 \fi
    check for option clash
7892 \ifglsacrdua
7893     \ifglsacrsmallcaps
7894         \PackageError{glossaries}{Option clash: `smallcaps' and `dua'
7895             can't both be set}{}
7896     \else
7897         \PackageError{glossaries}{Option clash: `smaller' and `dua'
7898             can't both be set}{}
7899     \fi
7900 \fi
7901 }%

```

**DUADisplayStyle** Sets the acronym display style for given glossary with dua setting.

```

7902 \newcommand*{\SetDUADisplayStyle}[1]{%
7903     \def\glsentryfmt[#1]{\glsgenentryfmt}%
7904 }

```

**UANewAcronymDef**

```

7905 \newcommand*{\DUANewAcronymDef}{%
7906     \edef\@do@newglossaryentry{%
7907         \noexpand\newglossaryentry{\the\glslabeltok}%
7908     }%
7909     type=\acronymtype,%
7910     name={\the\glsshorttok},%
7911     text={\the\glslongtok},%
7912     first={\the\glslongtok},%
7913     plural={\noexpand\expandonce\noexpand\@glo@longpl},%
7914     firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7915     short={\the\glsshorttok},%
7916     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7917     long={\the\glslongtok},%
7918     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7919     description={\the\glslongtok},%
7920     descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7921     symbol={\the\glsshorttok},%
7922     symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7923     \the\glskeylisttok
7924 }%
7925 }%
7926 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7927 \let\@org@gls@assign@plural\gls@assign@plural
7928 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7929 \let\@org@gls@assign@descplural\gls@assign@descplural
7930 \def\gls@assign@firstpl##1##2{%
7931     \@@gls@expand@field{##1}{firstpl}{##2}%
7932 }%
7933 \def\gls@assign@plural##1##2{%
7934     \@@gls@expand@field{##1}{plural}{##2}%

```

```

7935 }%
7936 \def\gls@assign@symbolplural##1##2{%
7937   \@@gls@expand@field{##1}{symbolplural}{##2}%
7938 }%
7939 \def\gls@assign@descplural##1##2{%
7940   \@@gls@expand@field{##1}{descplural}{##2}%
7941 }%
7942 \do@newglossaryentry
7943 \let\gls@assign@firstpl\org@gls@assign@firstpl
7944 \let\gls@assign@plural\org@gls@assign@plural
7945 \let\gls@assign@symbolplural\org@gls@assign@symbolplural
7946 \let\gls@assign@descplural\org@gls@assign@descplural
7947 }

```

\SetDUAStyle Always expand acronyms.

```

7948 \newcommand*{\SetDUAStyle}{%
7949   \renewcommand{\newacronym}[4][]{%
7950     \ifx\glsacronymlists\empty
7951       \def\glo@type{\acronymtype}%
7952       \setkeys{glossentry}{##1}%
7953       \DeclareAcronymList{\glo@type}%
7954       \SetDUADisplayStyle{\glo@type}%
7955     \fi
7956     \glskeylisttok{##1}%
7957     \glslabeltok{##2}%
7958     \glsshorttok{##3}%
7959     \glslongtok{##4}%
7960     \newacronymhook
7961     \DUANewAcronymDef
7962   }%

```

Set the display

```

7963   \for@\gls@type:=\glsacronymlists\do{%
7964     \SetDUADisplayStyle{\gls@type}%
7965   }%
7966 }

```

SetAcronymStyle

```

7967 \newcommand*{\SetAcronymStyle}{%
7968   \SetDefaultAcronymStyle
7969   \ifglsacrdescription
7970     \ifglsacrfootnote
7971       \SetDescriptionFootnoteAcronymStyle
7972     \else
7973       \ifglsacrdua
7974         \SetDescriptionDUAAcronymStyle
7975       \else
7976         \SetDescriptionAcronymStyle
7977       \fi
7978     \fi

```

```

7979 \else
7980   \ifglsacrfootnote
7981     \SetFootnoteAcronymStyle
7982   \else
7983     \ifthenelse{\boolean{glsacrsmallicaps}\OR
7984       \boolean{glsacrsmaller}}%
7985     {%
7986       \SetSmallAcronymStyle
7987     }%
7988     {%
7989       \ifglsacrdua
7990         \SetDUAStyle
7991       \fi
7992     }%
7993   \fi
7994 \fi
7995 }

```

Set the acronym style according to the package options

```
7996 \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.

`tomDisplayStyle` Sets the acronym display style.

```

7997 \newcommand*{\SetCustomDisplayStyle}[1]{%
7998   \def\glsentryfmt[#1]{\glsgenentryfmt}%
7999 }

```

`omAcronymFields`

```

8000 \newcommand*{\CustomAcronymFields}{%
8001   name={\the\glsshorttok},%
8002   description={\the\glslongtok},%
8003   first={\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
8004   firstplural={\acrfullformat
8005     {\noexpand\glsentrylongpl{\the\glslabeltok}}%
8006     {\noexpand\glsentryshortpl{\the\glslabeltok}}},%
8007   text={\the\glsshorttok},%
8008   plural={\the\glsshorttok\noexpand\acrpluralsuffix}%
8009 }

```

`omNewAcronymDef`

```

8010 \newcommand*{\CustomNewAcronymDef}{%
8011   \protected@edef\@do@newglossaryentry{%
8012     \noexpand\newglossaryentry{\the\glslabeltok}%
8013   }%

```

```

8014     type=\acronymtype,%
8015     short={\the\glshorttok},%
8016     shortplural={\the\glshorttok\noexpand\acrpluralsuffix},%
8017     long={\the\glslongtok},%
8018     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
8019     user1={\the\glshorttok},%
8020     user2={\the\glshorttok\noexpand\acrpluralsuffix},%
8021     user3={\the\glslongtok},%
8022     user4={\the\glslongtok\noexpand\acrpluralsuffix},%
8023     \CustomAcronymFields,%
8024     \the\glskeylisttok
8025   }%
8026 }%
8027 \do@newglossaryentry
8028 }

\SetCustomStyle
8029 \newcommand*\SetCustomStyle}{%
8030   \renewcommand{\newacronym}[4][]{%
8031     \ifx\@glsacronymlists\empty
8032       \def\@glo@type{\acronymtype}%
8033       \setkeys{glossentry}{##1}%
8034       \DeclareAcronymList{\@glo@type}%
8035       \SetCustomDisplayStyle{\@glo@type}%
8036     \fi
8037     \glskeylisttok{##1}%
8038     \glslabeltok{##2}%
8039     \glshorttok{##3}%
8040     \glslongtok{##4}%
8041     \newacronymhook
8042     \CustomNewAcronymDef
8043   }%
8044   Set the display
8045   \for@\gls@type:=\glsacronymlists\do{%
8046     \SetCustomDisplayStyle{\@gls@type}%
8047   }%

```

## 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.

```
8048 \RequirePackage{glossary-hypernav}
```

The styles that use list-like environments. These are not loaded if the nolist option is used:

```
8049 \@gls@loadlist
```

The styles that use the longtable environment. These are not loaded if the nolong package option is used.

```
8050 \@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.

```
8051 \@gls@loadsuper
```

The tree-like styles. These are not loaded if the notree package option is used.

```
8052 \@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.

```
8053 \ifx \@glossary@default@style \relax
```

```
8054 \else
```

```
8055 \setglossarystyle{\@glossary@default@style}
```

```
8056 \fi
```

## 1.20 Debugging Commands

```
\showgloparent \showgloparent{\label}
```

```
8057 \newcommand*{\showgloparent}[1]{%
8058   \expandafter\show\csname glo@\glsdetoklabel{#1}@parent\endcsname
8059 }
```

```
\showglolevel \showglolevel{\label}
```

```
8060 \newcommand*{\showglolevel}[1]{%
8061   \expandafter\show\csname glo@\glsdetoklabel{#1}@level\endcsname
8062 }
```

```
\showglotext \showglotext{\label}
```

```
8063 \newcommand*{\showglotext}[1]{%
8064   \expandafter\show\csname glo@\glsdetoklabel{#1}@text\endcsname
8065 }
```

```
\showgloplural \showgloplural{\label}
```

```
8066 \newcommand*{\showgloplural}[1]{%
8067   \expandafter\show\csname glo@\glsdetoklabel{#1}@plural\endcsname
8068 }
```

```
\showglofirst \showglofirst{\label}
```

```
8069 \newcommand*{\showglofirst}[1]{%
8070   \expandafter\show\csname glo@\glsdetoklabel{#1}@first\endcsname
8071 }
```

```
\showglofirstpl \showglofirstpl{\label}
```

```
8072 \newcommand*{\showglofirstpl}[1]{%
8073   \expandafter\show\csname glo@\glsdetoklabel{#1}@firstpl\endcsname
8074 }
```

```
\showglotype \showglotype{\label}
```

```
8075 \newcommand*{\showglotype}[1]{%
8076   \expandafter\show\csname glo@\glsdetoklabel{#1}@type\endcsname
8077 }
```

```
\showglocounter \showglocounter{\label}
```

```
8078 \newcommand*{\showglocounter}[1]{%
8079   \expandafter\show\csname glo@\glsdetoklabel{#1}@counter\endcsname
8080 }
```

```
\showglouserii \showglouserii{\label}
```

```
8081 \newcommand*{\showglouserii}[1]{%
8082   \expandafter\show\csname glo@\glsdetoklabel{#1}@userii\endcsname
8083 }
```

```
\showglouserii \showglouserii{\label}
```

```
8084 \newcommand*{\showglouserii}[1]{%
```

```
8085 \expandafter\show\csname glo@\glsdetoklabel{#1}@userii\endcsname  
8086 }
```

```
\showglouseriii {\label}
```

```
8087 \newcommand*\showglouseriii[1]{%  
8088 \expandafter\show\csname glo@\glsdetoklabel{#1}@useriii\endcsname  
8089 }
```

```
\showglouseriv {\label}
```

```
8090 \newcommand*\showglouseriv[1]{%  
8091 \expandafter\show\csname glo@\glsdetoklabel{#1}@useriv\endcsname  
8092 }
```

```
\showglouserv {\label}
```

```
8093 \newcommand*\showglouserv[1]{%  
8094 \expandafter\show\csname glo@\glsdetoklabel{#1}@userv\endcsname  
8095 }
```

```
\showglouservi {\label}
```

```
8096 \newcommand*\showglouservi[1]{%  
8097 \expandafter\show\csname glo@\glsdetoklabel{#1}@uservi\endcsname  
8098 }
```

```
\showgloname {\label}
```

```
8099 \newcommand*\showgloname[1]{%  
8100 \expandafter\show\csname glo@\glsdetoklabel{#1}@name\endcsname  
8101 }
```

```
\showglodesc {\label}
```

```
8102 \newcommand*\showglodesc[1]{%  
8103 \expandafter\show\csname glo@\glsdetoklabel{#1}@desc\endcsname  
8104 }
```

```
owglodescplural \showglodescplural{\label}
```

```
8105 \newcommand*{\showglodescplural}[1]{%
8106   \expandafter\show\csname glo@\glsdetoklabel{#1}@descplural\endcsname
8107 }
```

```
\showglosort \showglosort{\label}
```

```
8108 \newcommand*{\showglosort}[1]{%
8109   \expandafter\show\csname glo@\glsdetoklabel{#1}@sort\endcsname
8110 }
```

```
\showglosymbol \showglosymbol{\label}
```

```
8111 \newcommand*{\showglosymbol}[1]{%
8112   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbol\endcsname
8113 }
```

```
glosymbolplural \showglosymbolplural{\label}
```

```
8114 \newcommand*{\showglosymbolplural}[1]{%
8115   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolplural\endcsname
8116 }
```

```
\showgloshort \showgloshort{\label}
```

```
8117 \newcommand*{\showgloshort}[1]{%
8118   \expandafter\show\csname glo@\glsdetoklabel{#1}@short\endcsname
8119 }
```

```
\showglolong \showglolong{\label}
```

```
8120 \newcommand*{\showglolong}[1]{%
8121   \expandafter\show\csname glo@\glsdetoklabel{#1}@long\endcsname
8122 }
```

```
\showgloindex \showgloindex{\label}
```

```
8123 \newcommand*{\showgloindex}[1]{%
8124   \expandafter\show\csname glo@\glsdetoklabel{#1}@index\endcsname
8125 }
```

```
\showgloflag \showgloflag{\label}
```

```
8126 \newcommand*{\showgloflag}[1]{%
8127   \expandafter\show\csname ifglo@\glsdetoklabel{#1}@flag\endcsname
8128 }
```

```
\showgloloclist \showgloloclist{\label}
```

```
8129 \newcommand*{\showgloloclist}[1]{%
8130   \expandafter\show\csname glo@\glsdetoklabel{#1}@locist\endcsname
8131 }
```

```
\showglofield \showglofield{\label}{\field}
```

```
8132 \newcommand*{\showglofield}[2]{%
8133   \csshow{glo@\glsdetoklabel{#1}@#2}%
8134 }
```

```
howacronymlists \showacronymlists
```

Show list of glossaries that have been flagged as a list of acronyms.

```
8135 \newcommand*{\showacronymlists}{%
8136   \show@glssacronymlists
8137 }
```

```
\showglossaries \showglossaries
```

Show list of defined glossaries.

```
8138 \newcommand*{\showglossaries}{%
8139   \show@glo@types
8140 }
```

```
\showglossaryin \showglossaryin{\glossary-label}
```

Show the ‘in’ extension for the given glossary.

```
8141 \newcommand*{\showglossaryin}[1]{%
8142   \expandafter\show\csname @glotype@#1@in\endcsname
8143 }
```

showglossaryout **\showglossaryout{*glossary-label*}**

Show the ‘out’ extension for the given glossary.

```
8144 \newcommand*{\showglossaryout}[1]{%
8145   \expandafter\show\csname @glotype@#1@out\endcsname
8146 }
```

owglossarytitle **\showglossarytitle{*glossary-label*}**

Show the title for the given glossary.

```
8147 \newcommand*{\showglossarytitle}[1]{%
8148   \expandafter\show\csname @glotype@#1@title\endcsname
8149 }
```

glossarycounter **\showglossarycounter{*glossary-label*}**

Show the counter for the given glossary.

```
8150 \newcommand*{\showglossarycounter}[1]{%
8151   \expandafter\show\csname @glotype@#1@counter\endcsname
8152 }
```

glossaryentries **\showglossaryentries{*glossary-label*}**

Show the list of entry labels for the given glossary.

```
8153 \newcommand*{\showglossaryentries}[1]{%
8154   \expandafter\show\csname glolist@#1\endcsname
8155 }
```

## 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.

```
8156 \csname ifglscompatible-2.07\endcsname
8157   \RequirePackage{glossaries-compatible-207}
8158 \fi
```

## 2 Prefix Support (glossaries-prefix Code)

This package provides a means of adding prefixes to your glossary entries. For example, you may want to use “`a \gls{<label>}`” on first use but use “`an \gls{<label>}`” on subsequent use.

```
8159 \NeedsTeXFormat{LaTeX2e}
8160 \ProvidesPackage{glossaries-prefix}[2020/03/19 v4.46 (NLCT)]
```

Pass all options to glossaries:

```
8161 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
8162 \ProcessOptions
```

Load glossaries:

```
8163 \RequirePackage{glossaries}
```

Add the new keys:

```
8164 \define@key{glossentry}{prefixfirst}{\def\@glo@entryprefixfirst{\#1}}%
8165 \define@key{glossentry}{prefixfirstplural}{\def\@glo@entryprefixfirstplural{\#1}}%
8166 \define@key{glossentry}{prefix}{\def\@glo@entryprefix{\#1}}%
8167 \define@key{glossentry}{prefixplural}{\def\@glo@entryprefixplural{\#1}}%
```

Add them to `\@gls@keymap`:

```
8168 \appto\@gls@keymap{,
8169   {prefixfirst}{prefixfirst},%
8170   {prefixfirstplural}{prefixfirstplural},%
8171   {prefix}{prefix},%
8172   {prefixplural}{prefixplural}%
8173 }
```

Set the default values:

```
8174 \appto\@newglossaryentryprehook{%
8175   \def\@glo@entryprefix{}%
8176   \def\@glo@entryprefixplural{}%
8177   \let\@glo@entryprefixfirst\@gls@default@value
8178   \let\@glo@entryprefixfirstplural\@gls@default@value
8179 }
```

Set the assignment code:

```
8180 \appto\@newglossaryentryposthook{%
8181   \gls@assign@field{}{\@glo@label}{prefix}{\@glo@entryprefix}%
8182   \gls@assign@field{}{\@glo@label}{prefixplural}{\@glo@entryprefixplural}%

```

If `prefixfirst` has not been supplied, make it the same as `prefix`.

```
8183 \expandafter\gls@assign@field\expandafter
8184   {\csname glo@\@glo@label @prefix\endcsname}{\@glo@label}{prefixfirst}%
8185   {\@glo@entryprefixfirst}%

```

If `prefixfirstplural` has not been supplied, make it the same as `prefixplural`.

```
8186 \expandafter\gls@assign@field\expandafter
8187 {\csname glo@\glo@label \prefixplural\endcsname}{\glo@label}%
8188 {prefixfirstplural}{\glo@entryprefixfirstplural}%
8189 }
```

Define commands to access these fields:

```
ntryprefixfirst
8190 \newcommand*{\glsentryprefixfirst}[1]{\csuse{glo@\glsdetoklabel{#1}@prefixfirst}}}

efixfirstplural
8191 \newcommand*{\glsentryprefixfirstplural}[1]{%
8192 \csuse{glo@\glsdetoklabel{#1}@prefixfirstplural}%

\glsentryprefix
8193 \newcommand*{\glsentryprefix}[1]{\csuse{glo@\glsdetoklabel{#1}@prefix}}}

tryprefixplural
8194 \newcommand*{\glsentryprefixplural}[1]{\csuse{glo@\glsdetoklabel{#1}@prefixplural}}}

Now for the initial upper case variants:

ntryprefixfirst
8195 \newrobustcmd*{\Glsentryprefixfirst}[1]{%
8196 \protected@edef{\glo@text{\csname glo@\glsdetoklabel{#1}@prefixfirst\endcsname}}{%
8197 \xmakefirstuc{\glo@text
8198 }

efixfirstplural
8199 \newrobustcmd*{\Glsentryprefixfirstplural}[1]{%
8200 \protected@edef{\glo@text{\csname glo@\glsdetoklabel{#1}@prefixfirstplural\endcsname}}{%
8201 \xmakefirstuc{\glo@text
8202 }

\Glsentryprefix
8203 \newrobustcmd*{\Glsentryprefix}[1]{%
8204 \protected@edef{\glo@text{\csname glo@\glsdetoklabel{#1}@prefix\endcsname}}{%
8205 \xmakefirstuc{\glo@text
8206 }

tryprefixplural
8207 \newrobustcmd*{\Glsentryprefixplural}[1]{%
8208 \protected@edef{\glo@text{\csname glo@\glsdetoklabel{#1}@prefixplural\endcsname}}{%
8209 \xmakefirstuc{\glo@text
8210 }
```

Define commands to determine if the prefix keys have been set:

```

\ifglshasprefix
 8211 \newcommand*{\ifglshasprefix}[3]{%
 8212   \ifcsempy{glo@\glsdetoklabel{#1}@prefix}%
 8213   {#3}%
 8214   {#2}%
 8215 }

hasprefixplural
 8216 \newcommand*{\ifglshasprefixplural}[3]{%
 8217   \ifcsempy{glo@\glsdetoklabel{#1}@prefixplural}%
 8218   {#3}%
 8219   {#2}%
 8220 }

shasprefixfirst
 8221 \newcommand*{\ifglshasprefixfirst}[3]{%
 8222   \ifcsempy{glo@\glsdetoklabel{#1}@prefixfirst}%
 8223   {#3}%
 8224   {#2}%
 8225 }

efixfirstplural
 8226 \newcommand*{\ifglshasprefixfirstplural}[3]{%
 8227   \ifcsempy{glo@\glsdetoklabel{#1}@prefixfirstplural}%
 8228   {#3}%
 8229   {#2}%
 8230 }

fix@record@hook Need to take into account the possibility that glossaries-extra might be loaded with the record option.
 8231 \providecommand{\@glsprefix@record@hook}[2]{%
 8232   \ifdef\@glsxtr@record
 8233   {\@glsxtr@record{#1}{#2}{\glslink}}%
 8234   {}%
 8235 }

\glsprefixsep Separator between prefix and term. Does nothing by default.
 8236 \newcommand{\glsprefixsep}{}


```

Define commands that insert the prefix before commands like `\gls`:

```

\pgls
 8237 \newrobustcmd{\pgls}{\@gls@hyp@opt\@pgls}

\@pgls Unstarred version.
 8238 \newcommand*{\@pgls}[2][]{%
 8239   \new@ifnextchar[%
 8240     {\@pgls@{#1}{#2}}%

```

```
8241   {\@pglsc@{\#1}{\#2}[]}%  
8242 }
```

\@pglsc@ Read in the final optional argument:

```
8243 \def\@pglsc@#1#2[#3]{%  
8244   \@glsprefix@record@hook{\#1}{\#2}-%  
8245   \glsdoifexists{\#2}-%  
8246 {%-  
8247   \ifglsused{\#2}-%  
8248   {%-  
8249     \ifglshasprefix{\#2}{\glsentryprefix{\#2}\glsprefixsep}{}-%  
8250   }-%  
8251   {%-  
8252     \ifglshasprefixfirst{\#2}{\glsentryprefixfirst{\#2}\glsprefixsep}{}-%  
8253   }-%  
8254   \@gls@{\#1}{\#2}[#3]-%  
8255 }-%  
8256 }
```

Similarly for the plural version:

```
\pglsp  
8257 \newrobustcmd{\pglsp}{\gls@hyp@opt\pglsp}
```

\@pglsp Unstarred version.

```
8258 \newcommand*{\@pglsp}[2][]{%-  
8259   \new@ifnextchar[%  
8260   {\@pglsp@{\#1}{\#2}}%-  
8261   {\@pglsp@{\#1}{\#2}[]}%  
8262 }
```

\@pglsp@ Read in the final optional argument:

```
8263 \def\@pglsp@#1#2[#3]{%  
8264   \@glsprefix@record@hook{\#1}{\#2}-%  
8265   \glsdoifexists{\#2}-%  
8266 {%-  
8267   \ifglsused{\#2}-%  
8268   {%-  
8269     \ifglshasprefixplural{\#2}{\glsentryprefixplural{\#2}\glsprefixsep}{}-%  
8270   }-%  
8271   {%-  
8272     \ifglshasprefixfirstplural{\#2}-%  
8273       {\glsentryprefixfirstplural{\#2}\glsprefixsep}{}-%  
8274   }-%  
8275   \@glspl@{\#1}{\#2}[#3]-%  
8276 }-%  
8277 }
```

Now for the first letter upper case versions:

```
\Pgls
8278 \newrobustcmd{\Pgls}{\gls@hyp@opt\Pgls}
```

\@Pgls Unstarred version.

```
8279 \newcommand*{\@Pgls}[2] [] {%
8280   \new@ifnextchar[%
8281   {\@Pgls@{\#1}{\#2}}%
8282   {\@Pgls@{\#1}{\#2}[]}%
8283 }
```

\@Pgls@ Read in the final optional argument:

```
8284 \def{\@Pgls@#2[#3]}{%
8285   \glsprefix@record@hook{\#1}{\#2}%
8286   \glsdoifexists{\#2}%
8287   {%
8288     \ifglsused{\#2}%
8289     {%
8290       \ifglshasprefix{\#2}%
8291       {%
8292         \Glsentryprefix{\#2}%
8293         \glsprefixsep
8294         \gls{\#1}{\#2}[\#3]%
8295       }%
8296       {\@Gls{\#1}{\#2}[\#3]}%
8297     }%
8298     {%
8299       \ifglshasprefixfirst{\#2}%
8300       {%
8301         \Glsentryprefixfirst{\#2}%
8302         \glsprefixsep
8303         \gls{\#1}{\#2}[\#3]%
8304       }%
8305       {\@Gls{\#1}{\#2}[\#3]}%
8306     }%
8307   }%
8308 }
```

Similarly for the plural version:

```
\Pglspl
8309 \newrobustcmd{\Pglspl}{\gls@hyp@opt\Pglspl}
```

\@Pglspl Unstarred version.

```
8310 \newcommand*{\@Pglspl}[2] [] {%
8311   \new@ifnextchar[%
8312   {\@Pglspl@{\#1}{\#2}}%
8313   {\@Pglspl@{\#1}{\#2}[]}%
8314 }
```

\@Pglspl@ Read in the final optional argument:

```
8315 \def\@Pglspl@#1#2[#3]{%
8316   \glsprefix@record@hook{#1}{#2}%
8317   \glsdoifexists{#2}%
8318 {%
8319   \ifglsused{#2}%
8320   {%
8321     \ifglshasprefixplural{#2}%
8322     {%
8323       \Glsentryprefixplural{#2}%
8324       \glsprefixsep%
8325       \glspl@{#1}{#2}[#3]%
8326     }%
8327     {\@Glspl@{#1}{#2}[#3]}%
8328   }%
8329 {%
8330   \ifglshasprefixfirstplural{#2}%
8331   {%
8332     \Glsentryprefixfirstplural{#2}%
8333     \glsprefixsep%
8334     \glspl@{#1}{#2}[#3]%
8335   }%
8336   {\@Glspl@{#1}{#2}[#3]}%
8337 }%
8338 }%
8339 }
```

Finally the all upper case versions:

\PGLS

```
8340 \newrobustcmd{\PGLS}{\gls@hyp@opt\PGLS}
```

\@PGLS Unstarred version.

```
8341 \newcommand*\@PGLS[2][]{%
8342   \new@ifnextchar[%
8343   {\@PGLS@{#1}{#2}}%
8344   {\@PGLS@{#1}{#2}[]}}%
8345 }
```

\@PGLS@ Read in the final optional argument:

```
8346 \def\@PGLS@#1#2[#3]{%
8347   \glsprefix@record@hook{#1}{#2}%
8348   \glsdoifexists{#2}%
8349 {%
8350   \ifglsused{#2}%
8351   {%
8352     \ifglshasprefix{#2}%
8353       {\mfirstucMakeUppercase{\glsentryprefix{#2}\glsprefixsep}}{}%
```

```

8354    }%
8355    {%
8356        \ifglshasprefixfirst{#2}%
8357            {\mfirstucMakeUppercase{\glsentryprefixfirst{#2}\glsprefixsep}}{}%
8358        }%
8359        \c@GLS@{#1}{#2}[#3]%
8360    }%
8361 }

```

Plural version:

```
\PGLSp1
8362 \newrobustcmd{\PGLSp1}{\c@glshyp@opt\PGLSp1}
```

\c@PGLSp1 Unstarred version.

```

8363 \newcommand*{\c@PGLSp1}[2][]{%
8364     \new@ifnextchar[%
8365         {\c@PGLSp1@{#1}{#2}}%
8366         {\c@PGLSp1@{#1}{#2}[]}%
8367 }

```

\c@PGLSp1@ Read in the final optional argument:

```

8368 \def\c@PGLSp1@#1#2[#3]{%
8369     \c@glsprefix@record@hook{#1}{#2}%
8370     \glsdoifexists{#2}%
8371     {%
8372         \ifglsused{#2}%
8373         {%
8374             \ifglshasprefixplural{#2}%
8375                 {\mfirstucMakeUppercase{\glsentryprefixplural{#2}\glsprefixsep}}{}%
8376             }%
8377             {%
8378                 \ifglshasprefixfirstplural{#2}%
8379                     {\mfirstucMakeUppercase{\glsentryprefixfirstplural{#2}\glsprefixsep}}{}%
8380             }%
8381             \c@GLSp1@{#1}{#2}[#3]%
8382         }%
8383 }

```

# 3 Glossary Styles

## 3.1 Glossary hyper-navigation definitions (glossary-hypernav package)

Package Definition:

```
8384 \ProvidesPackage{glossary-hypernav}[2020/03/19 v4.46 (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`

```
8385 \newcommand*\glsnavhyperlink[3][\@glo@type]{%
8386   \edef\gls@grplabel{\#2}\protected@edef\gls@grptitle{\#3}%
8387   \glslink{\glsnavhyperlinkname{\#1}{\#2}}{\#3}}
```

`avhyperlinkname` Expands to the hypertarget name. The first argument is the glossary type. The second argument is the group label.

```
8388 \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.

```
8389 \newcommand*\glsnavhypertarget[3][\@glo@type]{%
8390   \glsnavhypertarget{\#1}{\#2}{\#3}%
8391 }
```

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.

`snavhypertarget`

```
8392 \newcommand*\glsnavhypertarget[3]{%
```

Add this group to the aux file for re-run check.

```
8393 \protected@write\@auxout{}{\string\gls@hypergroup{#1}{#2}}%
```

Add the target.

```
8394 \glstarget{\glsnavhyperlinkname{#1}{#2}}{#3}%
```

Check list of known groups to determine if a re-run is required.

```
8395 \expandafter\let
```

```
8396 \expandafter\@gls@list\csname \gls@hypergrouplist@#1\endcsname
```

Iterate through list and terminate loop if this group is found.

```
8397 \@for\@gls@elem:=\@gls@list\do{%
```

```
8398 \ifthenelse{\equal{\@gls@elem}{#2}}{\@endfortrue}{}%
```

Check if list terminated prematurely.

```
8399 \if@endfor
```

```
8400 \else
```

This group was not included in the list, so issue a warning.

```
8401 \GlossariesWarningNoLine{Navigation panel}
```

```
8402     for glossary type '#1'^^Jmissing group '#2'}%
```

```
8403 \gdef\gls@hypergrouprerun{%
```

```
8404 \GlossariesWarningNoLine{Navigation panel}
```

```
8405     has changed. Rerun LaTeX}}%
```

```
8406 \fi
```

```
8407 }
```

hypergrouprerun Give a warning at the end if re-run required

```
8408 \let\gls@hypergrouprerun\relax
```

```
8409 \AtEndDocument{\gls@hypergrouprerun}
```

@gls@hypergroup This adds to (or creates) the command \gls@hypergrouplist@(*glossary type*) which lists all groups for a given glossary, so that the navigation bar only contains those groups that are present. However it requires at least 2 runs to ensure the information is up-to-date.

```
8410 \newcommand*{\gls@hypergroup}[2]{%
```

```
8411 \ifundefined{\gls@hypergrouplist@#1}{%
```

```
8412 \expandafter\xdef\csname \gls@hypergrouplist@#1\endcsname{#2}{%
```

```
8413 }{%
```

```
8414 \expandafter\let\expandafter\gls@tmp
```

```
8415     \csname \gls@hypergrouplist@#1\endcsname
```

```
8416 \expandafter\xdef\csname \gls@hypergrouplist@#1\endcsname{%
```

```
8417     \gls@tmp, #2}{%
```

```
8418 }{%
```

```
8419 }
```

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
8420 \newcommand*{\glsnavigation}{%
8421   \def\@gls@between{}%
8422   \ifcsundef{@gls@hypergrouplist@\@glo@type}%
8423   {%
8424     \def\@gls@list{}%
8425   }%
8426   {%
8427     \expandafter\let\expandafter\@gls@list
8428       \csname @gls@hypergrouplist@\@glo@type\endcsname
8429   }%
8430   \cfor\@gls@tmp:=\@gls@list\do{%
8431     \@gls@between
8432     \gls@getgrouptitle{\@gls@tmp}\{\@gls@grptitle}\%
8433     \glsnavhyperlink{\@gls@tmp}\{\@gls@grptitle}\%
8434     \let\@gls@between\glshypernavsep
8435   }%
8436 }

```

\glshypernavsep Separator for the hyper navigation bar.

```
8437 \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
8438 \newcommand*{\glssymbolnav}{%
8439   \glsnavhyperlink{glssymbols}\{\glsgetgroup{glssymbols}\}\%
8440   \glshypernavsep
8441   \glsnavhyperlink{glsnumbers}\{\glsgetgroup{glsnumbers}\}\%
8442   \glshypernavsep
8443 }

```

## 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.

```
8444 \ProvidesPackage{glossary-inline}[2020/03/19 v4.46 (NLCT)]
```

`inline` Define the inline style.

```
8445 \newglossarystyle{inline}{%
```

Start of glossary sets up first empty separator between entries. (This is then changed by \glossentry)

```
8446 \renewenvironment{theglossary}%
8447   {}%
```

```

8448     \def\gls@inlinesep{}%
8449     \def\gls@inlinesubsep{}%
8450     \def\gls@inlinepostchild{}%
8451   }%
8452   {\glspostinline}%

```

No header:

```
8453 \renewcommand*{\glossaryheader}{}%
```

No group headings (if heading is required, add \glsinlinedopostchild to start definition in case heading follows a child entry):

```
8454 \renewcommand*{\glsgroupheading}[1]{}%
```

Just display separator followed by name and description:

```

8455 \renewcommand{\glossentry}[2]{%
8456   \glsinlinedopostchild
8457   \gls@inlinesep
8458   \glsentryitem{##1}%
8459   \glsinlinenameformat{##1}{%
8460     \glossentryname{##1}%
8461   }%
8462   \ifglsdescsuppressed{##1}%
8463   {%
8464     \glsinlineemptydescformat
8465   {%
8466     \glossentrysymbol{##1}%
8467   }%
8468   {%
8469     ##2%
8470   }%
8471 }%
8472 {%
8473   \ifglshasdesc{##1}%
8474   {\glsinlinedescformat{\glossentrydesc{##1}}{\glossentrysymbol{##1}}{##2}%
8475   {\glsinlineemptydescformat{\glossentrysymbol{##1}}{##2}}%
8476 }%
8477 \ifglshaschildren{##1}%
8478 {%
8479   \glsresetsubentrycounter
8480   \glsinlineparentchildseparator
8481   \def\gls@inlinesubsep{}%
8482   \def\gls@inlinepostchild{\glsinlinepostchild}%
8483 }%
8484 {%
8485   \def\gls@inlinesep{\glsinlineseparator}%
8486 }%

```

Sub-entries display description:

```

8487 \renewcommand{\subglossentry}[3]{%
8488   \gls@inlinesubsep%
8489   \glsinlinesubnameformat{##2}{%

```

```
8490     \glossentryname{##2}%
8491     \glssubentryitem{##2}%
8492     \glsinlinesubdescformat{\glossentrydesc{##2}}{\glossentrysymbol{##2}}{##3}%
8493     \def\gls@inlinesubsep{\glsinlinesubseparator}%
8494 }%
```

Nothing special between groups:

```
8495 \renewcommand*{\glsgroupskip}{}
8496 }
```

#### linedopostchild

```
8497 \newcommand*{\glsinlinedopostchild}{%
8498     \gls@inlinepostchild
8499     \def\gls@inlinepostchild{}%
8500 }
```

#### inlineseparator Separator to use between entries.

```
8501 \newcommand*{\glsinlineseparator}{; \space}
```

#### inesubseparator Separator to use between sub-entries.

```
8502 \newcommand*{\glsinlinesubseparator}{, \space}
```

#### tchildseparator Separator to use between parent and children.

```
8503 \newcommand*{\glsinlineparentchildseparator}{:\space}
```

#### inlinepostchild Hook to use between child and next entry

```
8504 \newcommand*{\glsinlinepostchild}{}
```

#### \glspostinline Terminator for inline glossary.

```
8505 \newcommand*{\glspostinline}{\glspostdescription\space}
```

#### linenameformat Formats the name of the entry (first argument label, second argument name):

```
8506 \newcommand*{\glsinlinenameformat}[2]{\glstarget{#1}{#2}}
```

#### linedescformat Formats the entry's description, symbol and location list:

```
8507 \newcommand*{\glsinlinedescformat}[3]{\space#1}
```

#### emptydescformat Formats the entry's symbol and location list when the description is empty:

```
8508 \newcommand*{\glsinlineemptydescformat}[2]{}{}
```

#### esubnameformat Formats the name of the subentry (first argument label, second argument name):

```
8509 \newcommand*{\glsinlinesubnameformat}[2]{\glstarget{#1}{}}
```

#### esubdescformat Formats the subentry's description, symbol and location list:

```
8510 \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.

```
8511 \ProvidesPackage{glossary-list}[2020/03/19 v4.46 (NLCT)]
```

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```
8512 \providecommand{\indexspace}{%
8513   \par \vskip 10\p@ \oplus 5\p@ \ominus 3\p@ \relax
8514 }
```

`tgroupheaderfmt` Provide a way of adjusting the format of the group headings.

```
8515 \newcommand*{\glslistgroupheaderfmt}[1]{#1}
```

`tnavigationitem` Provide a way of adjusting the format of the navigation header. This puts the navigation line inside the optional argument of `item` to prevent unwanted space occurring at the start, but this can cause a problem if the navigation line is too long. With this command, it makes it easier for the user to customise the style without having to remember to modify `\glossaryheader` after the style has been set.

```
8516 \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.

```
8517 \newglossarystyle{list}{%
```

Use `description` environment:

```
8518 \renewenvironment{theglossary}%
8519   {\begin{description}}{\end{description}}%
```

No header at the start of the environment:

```
8520 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8521 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries start a new item in the list:

```
8522 \renewcommand*{\glossentry}[2]{%
8523   \item[\glsentryitem{##1}%
8524     \glisttarget{##1}{\glossentryname{##1}}]
8525     \glossentrydesc{##1}\glspostdescription\space ##2}%

```

Sub-entries continue on the same line:

```
8526 \renewcommand*{\subglossentry}[3]{%
8527   \glssubentryitem{##2}%

```

```

8528   \glstarget{##2}{\strut}\space
8529   \glossentrydesc{##2}\glspostdescription\space ##3.}%
      Add vertical space between groups:
8530   \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
8531 }

listgroup The listgroup style is like the list style, but the glossary groups have headings.
8532 \newglossarystyle{listgroup}{%
      Base it on the list style:
8533   \setglossarystyle{list}%
      Each group has a heading:
8534   \renewcommand*{\glsgroupheading}[1]{%
8535     \item[\glslistgroupheaderfmt{\glsgetgrouptitle{##1}}]}}

listhypergroup The listhypergroup style is like the listgroup style, but has a set of links to the groups at the
start of the glossary.
8536 \newglossarystyle{listhypergroup}{%
      Base it on the list style:
8537   \setglossarystyle{list}%
      Add navigation links at the start of the environment.
8538   \renewcommand*{\glossaryheader}{%
8539     \glslistnavigationitem{\glsnavigation}}%
      Each group has a heading with a hypertarget:
8540   \renewcommand*{\glsgroupheading}[1]{%
8541     \item[\glslistgroupheaderfmt
           {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}]}}
```

altlist The altlist glossary style is like the list style, but places the description on a new line. Sub-entries follow in separate paragraphs without the sub-entry name. This style does not use the entry's symbol.

```

8543 \newglossarystyle{altlist}{%
      Base it on the list style:
8544   \setglossarystyle{list}%
      Main (level 0) entries start a new item in the list with a line break after the entry name:
8545   \renewcommand*{\glossentry}[2]{%
8546     \item[\glsentryitem{##1}%
8547       \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.

```

8548   \mbox{} \par\nobreak\@afterheading
8549   \glossentrydesc{##1}\glspostdescription\space ##2}%
```

Sub-entries start a new paragraph:

```
8550 \renewcommand{\subglossentry}[3]{%
8551   \par
8552   \glssubentryitem{##2}%
8553   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space ##3}%
8554 }
```

**altlistgroup** The `altlistgroup` glossary style is like the `altlist` style, but the glossary groups have headings.

```
8555 \newglossarystyle{altlistgroup}{%
```

Base it on the `altlist` style:

```
8556 \setglossarystyle{altlist}{%
```

Each group has a heading:

```
8557 \renewcommand*{\glsgroupheading}[1]{%
8558   \item[\glslistgroupheaderfmt{\glsgetgrouptitle{##1}}]}
```

**tlisthypergroup** The `tlisthypergroup` glossary style is like the `altlistgroup` style, but has a set of links to the groups at the start of the glossary.

```
8559 \newglossarystyle{tlisthypergroup}{%
```

Base it on the `altlist` style:

```
8560 \setglossarystyle{altlist}{%
```

Add navigation links at the start of the environment.

```
8561 \renewcommand*{\glossaryheader}{%
8562   \glslistnavigationitem{\glsnavigation}}%
```

Each group has a heading with a hypertarget:

```
8563 \renewcommand*{\glsgroupheading}[1]{%
8564   \item[\glslistgroupheaderfmt
8565     {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}]}
```

**listdotted** The `listdotted` glossary style was supplied by Axel Menzel. I've modified it slightly so that the distance from the start of the name to the end of the dotted line is specified by `\glslistdottedwidth`. Note that this style ignores the page numbers as well as the symbol. Sub-entries are displayed in the same way as top-level entries.

```
8566 \newglossarystyle{listdotted}{%
```

Base it on the `list` style:

```
8567 \setglossarystyle{list}{%
```

Each main (level 0) entry starts a new item:

```
8568 \renewcommand*{\glossentry}[2]{%
8569   \item[]\makebox[\glslistdottedwidth][1]{%
8570     \glsentryitem{##1}%
8571     \glstarget{##1}{\glossentryname{##1}}%
8572     \unskip\leaders\hbox to 2.9mm{\hss.\hfill\strut}\glossentrydesc{##1}}%
```

Sub entries have the same format as main entries:

```
8573 \renewcommand*{\subglossentry}[3]{%
8574   \item[]\makebox[\glslistdottedwidth][1]{%
8575     \glssubentryitem{##2}%
8576     \glstarget{##2}{\glossentryname{##2}}%
8577     \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##2}}%
8578 }
```

listdottedwidth

```
8579 \newlength\glslistdottedwidth
8580 \setlength{\glslistdottedwidth}{.5\hsize}
```

sublistdotted This style is similar to the glostylelistdotted style, except that the main entries just have the name displayed.

```
8581 \newglossarystyle{sublistdotted}{%
```

Base it on the listdotted style:

```
8582 \setglossarystyle{listdotted}{%
```

Main (level 0) entries just display the name:

```
8583 \renewcommand*{\glossentry}[2]{%
8584   \item[\glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}}]}%
8585 }
```

### 3.4 Glossary Styles using longtable (the glossary-long package)

The glossary styles defined in the package used the longtable environment in the glossary.

```
8586 \ProvidesPackage{glossary-long}[2020/03/19 v4.46 (NLCT)]
```

Requires the package:

```
8587 \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.)

```
8588 \@ifundefined{glsdescwidth}{%
8589   \newlength\glsdescwidth
8590   \setlength{\glsdescwidth}{0.6\hsize}
8591 }{}
```

\glspagelistwidth This is a length that governs the width of the page list column.

```
8592 \@ifundefined{glspagelistwidth}{%
8593   \newlength\glspagelistwidth
8594   \setlength{\glspagelistwidth}{0.1\hsize}
8595 }{}
```

**long** The long glossary style command which uses the longtable environment:

```
8596 \newglossarystyle{long}{%
```

    Use longtable with two columns:

```
8597 \renewenvironment{theglossary}{%
8598     \begin{longtable}{lp{\glsdescwidth}}}{%
8599     \end{longtable}}%
```

    Do nothing at the start of the environment:

```
8600 \renewcommand*\glossaryheader{}%
```

    No heading between groups:

```
8601 \renewcommand*\glsgroupheading[1]{}%
```

    Main (level 0) entries displayed in a row:

```
8602 \renewcommand{\glossentry}[2]{%
8603     \glsentryitem{\#\#1}\glstarget{\#\#1}{\glossentryname{\#\#1}} &
8604     \glossentrydesc{\#\#1}\glspostdescription\space ##2\tabularnewline
8605 }%
```

    Sub entries displayed on the following row without the name:

```
8606 \renewcommand{\subglossentry}[3]{%
8607     &
8608     \glssubentryitem{\#\#2}%
8609     \glstarget{\#\#2}{\strut}\glossentrydesc{\#\#2}\glspostdescription\space
8610     ##3\tabularnewline
8611 }%
```

    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>)

```
8612 \ifglsnogroupskip
8613     \renewcommand*\glsgroupskip{}%
8614 \else
8615     \renewcommand*\glsgroupskip{ \& \tabularnewline}%
8616 \fi
8617 }
```

**longborder** The longborder style is like the above, but with horizontal and vertical lines:

```
8618 \newglossarystyle{longborder}{%
```

    Base it on the glosstylelong style:

```
8619 \setglossarystyle{long}{%
```

    Use longtable with two columns with vertical lines between each column:

```
8620 \renewenvironment{theglossary}{%
8621     \begin{longtable}{|l|p{\glsdescwidth}|}}{\end{longtable}}%
```

    Place horizontal lines at the head and foot of the table:

```
8622 \renewcommand*\glossaryheader{\hline\endhead\hline\endfoot}%
8623 }
```

**longheader** The longheader style is like the long style but with a header:

```
8624 \newglossarystyle{longheader}{%
```

Base it on the `glostylelong` style:

```
8625 \setglossarystyle{long}%
```

Set the table's header:

```
8626 \renewcommand*\glossaryheader{%
8627   \bfseries \entryname & \bfseries \descriptionname\tabularnewline\endhead}%
8628 }
```

`ongheaderborder` The `longheaderborder` style is like the `long` style but with a header and border:

```
8629 \newglossarystyle{longheaderborder}{%
```

Base it on the `glostylelongborder` style:

```
8630 \setglossarystyle{longborder}%
```

Set the table's header and add horizontal line to table's foot:

```
8631 \renewcommand*\glossaryheader{%
8632   \hline\bfseries \entryname & \bfseries \descriptionname\tabularnewline\hline
8633   \endhead
8634   \hline\endfoot}%
8635 \hline\endfoot}%
8636 }
```

`long3col` The `long3col` style is like `long` but with 3 columns

```
8637 \newglossarystyle{long3col}{%
```

Use a `longtable` with 3 columns:

```
8638 \renewenvironment{theglossary}%
8639 { \begin{longtable}{lp{\glscdescwidth}p{\glspagelistwidth}} }%
8640 { \end{longtable} }%
```

No table header:

```
8641 \renewcommand*\glossaryheader{}%
```

No headings between groups:

```
8642 \renewcommand*\glsgroupheading[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
8643 \renewcommand{\glossentry}[2]{%
8644   \glstarget{\#1}{\glsentryname{\#1}} &
8645   \glossentrydesc{\#1} & \#2\tabularnewline
8646 }%
```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```
8647 \renewcommand{\subglossentry}[3]{%
8648   &
8649   \glssubentryitem{\#2}%
8650   \glstarget{\#2}{\strut}\glossentrydesc{\#2} &
8651   \#3\tabularnewline
8652 }%
```

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>)

```
8653 \ifglsnogroupskip
8654   \renewcommand*\glsgroupskip{}%
8655 \else
8656   \renewcommand*\glsgroupskip{\&\& \tabularnewline}%
8657 \fi
8658 }
```

**long3colborder** The long3colborder style is like the long3col style but with a border:

```
8659 \newglossarystyle{long3colborder}{%
```

Base it on the glostylelong3col style:

```
8660 \setglossarystyle{long3col}{%
```

Use a longtable with 3 columns with vertical lines around them:

```
8661 \renewenvironment{theglossary}{%
8662   {\begin{longtable}{|l|p{\glsdescwidth}|p{\glspagelistwidth}|}}%
8663   {\end{longtable}}}%
```

Place horizontal lines at the head and foot of the table:

```
8664 \renewcommand*\glossaryheader{\hline\endhead\hline\endfoot}%
8665 }
```

**long3colheader** The long3colheader style is like long3col but with a header row:

```
8666 \newglossarystyle{long3colheader}{%
```

Base it on the glostylelong3col style:

```
8667 \setglossarystyle{long3col}{%
```

Set the table's header:

```
8668 \renewcommand*\glossaryheader{%
8669   \bfseries\entryname\&\bfseries\descriptionname\&
8670   \bfseries\pagelistname\tabularnewline\endhead}%
8671 }
```

**colheaderborder** The long3colheaderborder style is like the above but with a border

```
8672 \newglossarystyle{long3colheaderborder}{%
```

Base it on the glostylelong3colborder style:

```
8673 \setglossarystyle{long3colborder}{%
```

Set the table's header and add horizontal line at table's foot:

```
8674 \renewcommand*\glossaryheader{%
8675   \hline
8676   \bfseries\entryname\&\bfseries\descriptionname\&
8677   \bfseries\pagelistname\tabularnewline\hline\endhead
8678   \hline\endfoot}%
8679 }
```

`long4col` The `long4col` style has four columns where the third column contains the value of the associated symbol key.

```
8680 \newglossarystyle{long4col}{%
```

Use a `longtable` with 4 columns:

```
8681 \renewenvironment{theglossary}{%
8682 {\begin{longtable}{l l l l}}%
8683 {\end{longtable}}%
```

No table header:

```
8684 \renewcommand*\glossaryheader{}%
```

No group headings:

```
8685 \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):

```
8686 \renewcommand{\glossentry}[2]{%
8687 \glstarget{\glossentryname} &
8688 \glossentrydesc &
8689 \glossentrysymbol &
8690 ##2\tabularnewline
8691 }%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
8692 \renewcommand{\subglossentry}[3]{%
8693 &
8694 \glssubentryitem{##2}%
8695 \glstarget{##2}{\strut}\glossentrydesc{##2} &
8696 \glossentrysymbol{##2} & ##3\tabularnewline
8697 }%
```

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>)

```
8698 \ifglsnogroupskip
8699 \renewcommand*\glsgroupskip{}%
8700 \else
8701 \renewcommand*\glsgroupskip{ & & & \tabularnewline}%
8702 \fi
8703 }
```

`long4colheader` The `long4colheader` style is like `long4col` but with a header row.

```
8704 \newglossarystyle{long4colheader}{%
```

Base it on the `glostylelong4col` style:

```
8705 \setglossarystyle{long4col}{}
```

Table has a header:

```
8706 \renewcommand*\glossaryheader{}%
8707 \bfseries\entryname\bfseries\descriptionname&
8708 \bfseries\symbolname\bfseries
```

```
8709     \bfseries\pagelistname\tabularnewline\endhead}%
8710 }
```

**long4colborder** The **long4colborder** style is like **long4col** but with a border.

```
8711 \newglossarystyle{long4colborder}{%
```

Base it on the **glostylelong4col** style:

```
8712 \setglossarystyle{long4col}{%
```

Use a **longtable** with 4 columns surrounded by vertical lines:

```
8713 \renewenvironment{theglossary}%
8714 {\begin{longtable}{|l|l|l|l|}}%
8715 {\end{longtable}}%
```

Add horizontal lines to the head and foot of the table:

```
8716 \renewcommand*\glossaryheader{\hline\endhead\hline\endfoot}%
8717 }
```

**colheaderborder** The **long4colheaderborder** style is like the above but with a border.

```
8718 \newglossarystyle{long4colheaderborder}{%
```

Base it on the **glostylelong4col** style:

```
8719 \setglossarystyle{long4col}{%
```

Use a **longtable** with 4 columns surrounded by vertical lines:

```
8720 \renewenvironment{theglossary}%
8721 {\begin{longtable}{|l|l|l|l|}}%
8722 {\end{longtable}}%
```

Add table header and horizontal line at the table's foot:

```
8723 \renewcommand*\glossaryheader{%
8724   \hline\bfseries\entryname\&\bfseries\descriptionname\&
8725   \bfseries \symbolname\&
8726   \bfseries\pagelistname\tabularnewline\hline\endhead
8727   \hline\endfoot}%
8728 }
```

**altnlong4col** The **altnlong4col** style is like the **long4col** style but can have multiline descriptions and page lists.

```
8729 \newglossarystyle{altnlong4col}{%
```

Base it on the **glostylelong4col** style:

```
8730 \setglossarystyle{long4col}{%
```

Use a **longtable** with 4 columns where the second and last columns may have multiple lines in each row:

```
8731 \renewenvironment{theglossary}%
8732 {\begin{longtable}{lp{\glscdescwidth}lp{\glspagelistwidth}}}%
8733 {\end{longtable}}%
8734 }
```

tlong4colheader The `altnlong4colheader` style is like `altnlong4col` but with a header row.

```
8735 \newglossarystyle{altnlong4colheader}{%
```

Base it on the `glostylelong4colheader` style:

```
8736 \setglossarystyle{long4colheader}{%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8737 \renewenvironment{theglossary}{%
```

```
8738 {\begin{longtable}{lp{\glscdescwidth}lp{\glspagelistwidth}}}{%
```

```
8739 {\end{longtable}}{}}
```

```
8740 }
```

tlong4colborder The `altnlong4colborder` style is like `altnlong4col` but with a border.

```
8741 \newglossarystyle{altnlong4colborder}{%
```

Base it on the `glostylelong4colborder` style:

```
8742 \setglossarystyle{long4colborder}{%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8743 \renewenvironment{theglossary}{%
```

```
8744 {\begin{longtable}{|l|p{\glscdescwidth}|l|p{\glspagelistwidth}|}}{}}
```

```
8745 {\end{longtable}}{}}
```

```
8746 }
```

colheaderborder The `altnlong4colheaderborder` style is like the above but with a header as well as a border.

```
8747 \newglossarystyle{altnlong4colheaderborder}{%
```

Base it on the `glostylelong4colheaderborder` style:

```
8748 \setglossarystyle{long4colheaderborder}{%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8749 \renewenvironment{theglossary}{%
```

```
8750 {\begin{longtable}{|l|p{\glscdescwidth}|l|p{\glspagelistwidth}|}}{}}
```

```
8751 {\end{longtable}}{}}
```

```
8752 }
```

### 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>

```
8753 \ProvidesPackage{glossary-longbooktabs}[2020/03/19 v4.46 (NLCT)]
```

Requires booktabs package:

```
8754 \RequirePackage{booktabs}
```

and the base packages for long styles:

```
8755 \RequirePackage{glossary-long}
8756 \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.

```
8757 \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.

```
8758 \glspatchLToutput
```

As with the `longheader` style, use the `long` style as a base.

```
8759 \setglossarystyle{long}%
```

Add a header with rules.

```
8760 \renewcommand*{\glossaryheader}{%
8761   \toprule \bfseries \entryname & \bfseries
8762   \descriptionname\tabularnewline\midrule\endhead
8763   \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.

```
8764 \ifglsnogroupskip
8765   \renewcommand*{\glsgroupskip}{}%
8766 \else
8767   \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
8768 \fi
8769 }
```

`ng3col-booktabs` The `long3col-booktabs` style is similar to the `long3colheader` style but uses the `booktabs` rules and patches `longtable` to check for group skip occurring at a page break.

```
8770 \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.

```
8771 \glspatchLToutput
```

Use the `long3col` style as a base.

```
8772 \setglossarystyle{long3col}%
```

Add a header with rules.

```
8773 \renewcommand*{\glossaryheader}{%
8774   \toprule \bfseries \entryname &
8775   \bfseries \descriptionname &
8776   \bfseries \pagelistname
8777   \tabularnewline\midrule\endhead
8778   \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.

```
8779 \ifglsnogroupskip
8780   \renewcommand*\glsgroupskip{}%
8781 \else
8782   \renewcommand*\glsgroupskip{\glspenaltygroupskip}%
8783 \fi
8784 }
```

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.

```
8785 \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.

```
8786 \glspatchLToutput
```

Use the long4col style as a base.

```
8787 \setglossarystyle{long4col}{%
```

Add a header with rules.

```
8788 \renewcommand*\glossaryheader{}%
8789   \toprule \bfseries \entryname &
8790   \bfseries \descriptionname &
8791   \bfseries \symbolname &
8792   \bfseries \pagelistname
8793   \tabularnewline\midrule\endhead
8794 \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.

```
8795 \ifglsnogroupskip
8796   \renewcommand*\glsgroupskip{}%
8797 \else
8798   \renewcommand*\glsgroupskip{\glspenaltygroupskip}%
8799 \fi
8800 }
```

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.

```
8801 \newglossarystyle{altnormal4col-booktabs}{%
```

The patch \glspatchLToutput is already applied in long4col-booktabs and so doesn't need to be here.

```
8802 \glspatchLToutput
```

Use the long4col-booktabs style as a base.

```
8803 \setglossarystyle{long4col-booktabs}{%
```

Change the column specifications:

```
8804 \renewenvironment{theglossary}%
8805   {\begin{longtable}{lp{\glsdescwidth}lp{\glspagelistwidth}}}%
8806   {\end{longtable}}%
8807 }
```

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.

```
8808 \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.

```
8809 \glspatchLToutput
```

Use the `long-booktabs` style as a base.

```
8810 \setglossarystyle{long-booktabs}{%
```

Adjust the column specification.

```
8811 \renewenvironment{theglossary}%
8812   {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}}}%
8813   {\end{longtable}}%
8814 }
```

`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.

```
8815 \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.

```
8816 \glspatchLToutput
```

Use the `long3col-booktabs` style as a base.

```
8817 \setglossarystyle{long3col-booktabs}{%
```

Adjust the column specification.

```
8818 \renewenvironment{theglossary}%
8819   {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}}%
8820     >{\raggedright}p{\glspagelistwidth}}}%
8821   {\end{longtable}}%
8822 }
```

`ed4col-booktabs` The `altrongagged4col-booktabs` style is similar to the `altrongagged4col` style but uses the `booktabs` rules and patches `longtable` to check for group skip occurring at a page break.

```
8823 \newglossarystyle{altrongagged4col-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.

```
8824 \glspatchLToutput
```

Use the `altnlong4col-booktabs` style as a base.

```
8825 \setglossarystyle{altnlong4col-booktabs}%
```

Adjust the column specification.

```
8826 \renewenvironment{theglossary}%
8827 { \begin{longtable}{l>{\raggedright\arraybackslash}p{\glsdescwidth}l}%
8828 >{\raggedright\arraybackslash}p{\glspagelistwidth}} }%
8829 \end{longtable} }%
8830 }
```

`sLTpenaltycheck`

```
8831 \newcommand*{\glsLTpenaltycheck}{%
8832 \ifnum\outputpenalty=-50\vskip-\normalbaselineskip\relax\fi
8833 }
```

`enaltygroupskip`

```
8834 \newcommand{\glspenaltygroupskip}{%
8835 \noalign{\penalty-50\vskip\normalbaselineskip}}
```

`restoreLToutput` Provide a way of restoring `\LT@output` for the user.

```
8836 \let\@gls@org@LT@output\LT@output
8837 \newcommand*{\glsrestoreLToutput}{\let\LT@output\@gls@org@LT@output}
```

This is David's patch, but I've replaced the hard-coded values with `\glsLTpenaltycheck` to make it easier to adjust.

`lspatchLToutput`

```
8838 \newcommand*{\glspatchLToutput}{%
8839 \renewcommand*{\LT@output}{%
8840 \ifnum\outputpenalty <-@Mi
8841 \ifnum\outputpenalty > -\LT@end@open
8842 \LT@err{floats and marginpars not allowed in a longtable}@ehc
8843 \else
8844 \setbox\z@\vbox{\unvbox\@cclv}%
8845 \ifdim\ht\LT@lastfoot>\ht\LT@foot
8846 \dimen@\pagegoal
8847 \advance\dimen@-\ht\LT@lastfoot
8848 \ifdim\dimen@<\ht\z@
8849 \setbox\@cclv\vbox{\unvbox\z@\copy\LT@foot\vss}%
8850 \makecol
8851 \outputpage
8852 \setbox\z@\vbox{\box\LT@head\glsLTpenaltycheck}%
8853 \fi
8854 \fi
8855 \global\@colroom\@colht
8856 \global\vsiz@\colht
8857 {\unvbox\z@\box\ifvoid\LT@lastfoot\LT@foot\else\LT@lastfoot\fi}%
8858 \fi
8859 \else}
```

```

8860      \setbox\@cclv\vbox{\unvbox\@cclv\copy\LT@foot\vss}%
8861      \@makecol
8862      \@outputpage
8863      \global\vsize\@colroom
8864      \copy\LT@head
8865      \glsLTpenaltycheck
8866      \nobreak
8867      \fi
8868  }%
8869 }

```

## 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.

```
8870 \ProvidesPackage{glossary-longragged}[2020/03/19 v4.46 (NLCT)]
```

Requires the package:

```
8871 \RequirePackage{array}
```

Requires the package:

```
8872 \RequirePackage{longtable}
```

`\glsdescwidth` This is a length that governs the width of the description column. This may have already been defined.

```

8873 \@ifundefined{glsdescwidth}{%
8874   \newlength\glsdescwidth
8875   \setlength{\glsdescwidth}{0.6\hsize}
8876 }{}
```

`lspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```

8877 \@ifundefined{glspagelistwidth}{%
8878   \newlength\glspagelistwidth
8879   \setlength{\glspagelistwidth}{0.1\hsize}
8880 }{}
```

`longragged` The longragged glossary style is like the long but uses ragged right formatting for the description column.

```
8881 \newglossarystyle{longragged}{%
```

Use longtable with two columns:

```

8882  \renewenvironment{theglossary}{%
8883    {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}}}{%
8884      {\end{longtable}}}{%
```

Do nothing at the start of the environment:

```
8885  \renewcommand*\glossaryheader{}%
```

No heading between groups:

```
8886 \renewcommand*\glsgroupheading}[1]{}
```

Main (level 0) entries displayed in a row:

```
8887 \renewcommand{\glossentry}[2]{%
8888   \glstarget{\#1}\glsentryname{\#1} &
8889   \glossentrydesc{\#1}\glspostdescription\space ##2%
8890   \tabularnewline
8891 }
```

Sub entries displayed on the following row without the name:

```
8892 \renewcommand{\subglossentry}[3]{%
8893   &
8894   \glssubentryitem{\#2}%
8895   \glstarget{\#2}{\strut}\glossentrydesc{\#2}%
8896   \glspostdescription\space ##3%
8897   \tabularnewline
8898 }
```

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>)

```
8899 \ifglsnogroupskip
8900   \renewcommand*\glsgroupskip}{}%
8901 \else
8902   \renewcommand*\glsgroupskip}{\&\tabularnewline}%
8903 \fi
8904 }
```

ongraggedborder The longraggedborder style is like the above, but with horizontal and vertical lines:

```
8905 \newglossarystyle{longraggedborder}{%
```

Base it on the glostylelongragged style:

```
8906 \setglossarystyle{longragged}{%
```

Use longtable with two columns with vertical lines between each column:

```
8907 \renewenvironment{theglossary}{%
8908   \begin{longtable}{|l|>{\raggedright}p{\glsdescwidth}|}}%
8909   \end{longtable}}
```

Place horizontal lines at the head and foot of the table:

```
8910 \renewcommand*\glossaryheader}{\hline\endhead\hline\endfoot}%
8911 }
```

ongraggedheader The longraggedheader style is like the longragged style but with a header:

```
8912 \newglossarystyle{longraggedheader}{%
```

Base it on the glostylelongragged style:

```
8913 \setglossarystyle{longragged}{%
```

Set the table's header:

```
8914 \renewcommand*\glossaryheader}{%
8915   \bfseries \entryname \& \bfseries \descriptionname
```

```
8916     \tabularnewline\endhead}%
8917 }
```

gedheaderborder The `longraggedheaderborder` style is like the `longragged` style but with a header and border:

```
8918 \newglossarystyle{longraggedheaderborder}{%
```

Base it on the `glostylelongraggedborder` style:

```
8919 \setglossarystyle{longraggedborder}{%
```

Set the table's header and add horizontal line to table's foot:

```
8920 \renewcommand*\glossaryheader}{%
8921   \hline\bfseries \entryname & \bfseries \descriptionname
8922   \tabularnewline\hline
8923   \endhead
8924   \hline\endfoot}%
8925 }
```

`longragged3col` The `longragged3col` style is like `longragged` but with 3 columns

```
8926 \newglossarystyle{longragged3col}{%
```

Use a `longtable` with 3 columns:

```
8927 \renewenvironment{theglossary}{%
8928   {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}>{\raggedright}p{\glspagelistwidth}}}}%
8929   {\end{longtable}}%
```

No table header:

```
8931 \renewcommand*\glossaryheader{}{%
```

No headings between groups:

```
8932 \renewcommand*\glsgroupheading[1]{}{%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
8933 \renewcommand{\glossentry}[2]{%
8934   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8935   \glossentrydesc{##1} & ##2\tabularnewline
8936 }{%
```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```
8937 \renewcommand{\subglossentry}[3]{%
8938   &
8939   \glssubentryitem{##2}%
8940   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8941   ##3\tabularnewline
8942 }{%
```

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>)

```
8943 \ifglsnogroupskip
8944   \renewcommand*\glsgroupskip{}{%
```

```

8945 \else
8946   \renewcommand*{\glsgroupskip}{\& \tabularnewline}%
8947 \fi
8948 }

```

`agged3colborder` The `longragged3colborder` style is like the `longragged3col` style but with a border:

```
8949 \newglossarystyle{longragged3colborder}{%
```

Base it on the `glostylelongragged3col` style:

```
8950 \setglossarystyle{longragged3col}{%
```

Use a `longtable` with 3 columns with vertical lines around them:

```

8951 \renewenvironment{theglossary}%
8952   {\begin{longtable}{|l|>{\raggedright}p{\glsdescwidth}|%}
8953     >{\raggedright}p{\glspagelistwidth}|}}%
8954 {\end{longtable}}%

```

Place horizontal lines at the head and foot of the table:

```

8955 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8956 }%

```

`agged3colheader` The `longragged3colheader` style is like `longragged3col` but with a header row:

```
8957 \newglossarystyle{longragged3colheader}{%
```

Base it on the `glostylelongragged3col` style:

```
8958 \setglossarystyle{longragged3col}{%
```

Set the table's header:

```

8959 \renewcommand*{\glossaryheader}{%
8960   \bfseries\entryname\&\bfseries\descriptionname\&
8961   \bfseries\pagelistname\tabularnewline\endhead}%
8962 }%

```

`colheaderborder` The `longragged3colheaderborder` style is like the above but with a border

```
8963 \newglossarystyle{longragged3colheaderborder}{%
```

Base it on the `glostylelongragged3colborder` style:

```
8964 \setglossarystyle{longragged3colborder}{%
```

Set the table's header and add horizontal line at table's foot:

```

8965 \renewcommand*{\glossaryheader}{%
8966   \hline
8967   \bfseries\entryname\&\bfseries\descriptionname\&
8968   \bfseries\pagelistname\tabularnewline\hline\endhead
8969   \hline\endfoot}%
8970 }%

```

`tlongragged4col` The `altnlongragged4col` style is like the `altnlong4col` style defined in the package, except that ragged right formatting is used for the description and page list columns.

```
8971 \newglossarystyle{altnlongragged4col}{%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8972 \renewenvironment{theglossary}%
8973   {\begin{longtable}{l>{\raggedright\hspace*{\glsdescwidth}l}
8974     >{\raggedright\hspace*{\glspagelistwidth}}l}%
8975   {\end{longtable}}%
```

No table header:

```
8976 \renewcommand*\glossaryheader{}%
```

No group headings:

```
8977 \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):

```
8978 \renewcommand{\glossentry}[2]{%
8979   \glsentryitem{\#1}\glstarget{\#1}{\glossentryname{\#1}} &
8980   \glossentrydesc{\#1} & \glossentrysymbol{\#1} &
8981   ##2\tabularnewline
8982 }%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
8983 \renewcommand{\subglossentry}[3]{%
8984   &
8985   \glssubentryitem{\#2}%
8986   \glstarget{\#2}{\strut}\glossentrydesc{\#2} &
8987   \glossentrysymbol{\#2} & ##3\tabularnewline
8988 }%
```

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>)

```
8989 \ifglsnogroupskip
8990   \renewcommand*\glsgroupskip{}%
8991 \else
8992   \renewcommand*\glsgroupskip{ & & & \tabularnewline}%
8993 \fi
8994 }
```

agged4colheader The altlongragged4colheader style is like altlongragged4col but with a header row.

```
8995 \newglossarystyle{altlongragged4colheader}{%
```

Base it on the glostylealtlongragged4col style:

```
8996 \setglossarystyle{altlongragged4col}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8997 \renewenvironment{theglossary}%
8998   {\begin{longtable}{l>{\raggedright\hspace*{\glsdescwidth}l}
8999     >{\raggedright\hspace*{\glspagelistwidth}}l}%
9000   {\end{longtable}}%
```

Table has a header:

```
9001 \renewcommand*{\glossaryheader}{%
9002   \bfseries\entryname&\bfseries\descriptionname&
9003   \bfseries \symbolname&
9004   \bfseries\pagelistname\tabularnewline\endhead}%
9005 }
```

agged4colborder The `altlongragged4colborder` style is like `altlongragged4col` but with a border.

```
9006 \newglossarystyle{altlongragged4colborder}{%
```

Base it on the `glostylealtlongragged4col` style:

```
9007 \setglossarystyle{altlongragged4col}{%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
9008 \renewenvironment{theglossary}{%
9009   {\begin{longtable}{|l|>{\raggedright}p{\glsdescwidth}|l|%
9010     >{\raggedright}p{\glspagelistwidth}|}}%
9011   {\end{longtable}}{}}
```

Add horizontal lines to the head and foot of the table:

```
9012 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
9013 }
```

colheaderborder The `altlongragged4colheaderborder` style is like the above but with a header as well as a border.

```
9014 \newglossarystyle{altlongragged4colheaderborder}{%
```

Base it on the `glostylealtlongragged4col` style:

```
9015 \setglossarystyle{altlongragged4col}{%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
9016 \renewenvironment{theglossary}{%
9017   {\begin{longtable}{|l|>{\raggedright}p{\glsdescwidth}|l|%
9018     >{\raggedright}p{\glspagelistwidth}|}}%
9019   {\end{longtable}}{}}
```

Add table header and horizontal line at the table's foot:

```
9020 \renewcommand*{\glossaryheader}{%
9021   \hline\bfseries\entryname&\bfseries\descriptionname&
9022   \bfseries \symbolname&
9023   \bfseries\pagelistname\tabularnewline\hline\endhead
9024   \hline\endfoot}%
9025 }
```

## 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.

```
9026 \ProvidesPackage{glossary-mcols}[2020/03/19 v4.46 (NLCT)]
```

Required packages:

```
9027 \RequirePackage{multicol}
9028 \RequirePackage{glossary-tree}
```

\indexspace The are a few classes that don't define \indexspace, so provide a definition if it hasn't been defined.

```
9029 \providecommand{\indexspace}{%
9030   \par \vskip 10\p@ \oplus 5\p@ \ominus 3\p@ \relax
9031 }
```

\glsmcols Define macro in which to store the number of columns. (Defaults to 2.)

```
9032 \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 multcols, but the title isn't part of the glossary style.)

```
9033 \newglossarystyle{mcolindex}{%
9034   \setglossarystyle{index}%
9035   \renewenvironment{theglossary}%
9036     {%
9037       \begin{multicols}{\glsmcols}
9038         \setlength{\parindent}{0pt}%
9039         \setlength{\parskip}{0pt plus 0.3pt}%
9040         \let\item\glstreeitem
9041         \let\subitem\glstreesubitem
9042         \let\subsubitem\glstreesubsubitem
9043     }%
9044   \end{multicols}%
9045 }
```

mcolindexgroup As mcolindex but has headings:

```
9046 \newglossarystyle{mcolindexgroup}{%
9047   \setglossarystyle{mcolindex}%
9048   \renewcommand*\{glsgroupheading}[1]{%
9049     \item\glstreegroupheaderfmt{\glsgetgroup{##1}\indexspace}%
9050 }
```

indexhypergroup The mcolindexhypergroup style is like the mcolindexgroup style but has hyper navigation.

```
9051 \newglossarystyle{mcolindexhypergroup}{%
```

Base it on the glostestylemcolindex style:

```
9052 \setglossarystyle{mcolindex}{%
```

Put navigation links to the groups at the start of the glossary:

```
9053 \renewcommand*\{glossaryheader}{%
9054   \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.

```
9055 \renewcommand*{\glsgroupheading}[1]{%
9056   \item\glstreegroupheaderfmt
9057   {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}{%
9058   \indexspace}%
9059 }
```

`colindexspannav` Similar to `mcolindexhypergroup`, but puts the navigation line in the optional argument of `multicol`.

```
9060 \newglossarystyle{mcolindexspannav}{%
9061   \setglossarystyle{index}%
9062   \renewenvironment{theglossary}%
9063   {%
9064     \begin{multicols}{\glsmcols}[\noindent\glstreenavigationfmt{\glsnavigation}]
9065     \setlength{\parindent}{0pt}%
9066     \setlength{\parskip}{0pt plus 0.3pt}%
9067     \let\item\glstreeitem}%
9068   {\end{multicols}}}
```

Add a heading for each group (with a target). The group's title is in bold followed by a vertical gap.

```
9069 \renewcommand*{\glsgroupheading}[1]{%
9070   \item\glstreegroupheaderfmt
9071   {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}{%
9072   \indexspace}%
9073 }
```

`mcoltree` Multi-column index style. Same as the tree, but puts the glossary in multiple columns.

```
9074 \newglossarystyle{mcoltree}{%
9075   \setglossarystyle{tree}%
9076   \renewenvironment{theglossary}%
9077   {%
9078     \begin{multicols}{\glsmcols}
9079     \setlength{\parindent}{0pt}%
9080     \setlength{\parskip}{0pt plus 0.3pt}%
9081   }%
9082   {\end{multicols}}%
9083 }
```

`mcoltreegroup` Like the `mcoltree` style but the glossary groups have headings.

```
9084 \newglossarystyle{mcoltreegroup}{%
9085   \setglossarystyle{mcoltree}%
9086 }
```

Base it on the `glostylemcoltree` style:

```
9085 \setglossarystyle{mcoltree}%
```

Each group has a heading (in bold) followed by a vertical gap):

```
9086 \renewcommand{\glsgroupheading}[1]{\par
9087   \noindent\glstreegroupheaderfmt{\glsgroupname}\par\indexspace}%
9088 }
```

**mcoltreehypergroup** The mcoltreehypergroup style is like the treegroup style, but has a set of links to the groups at the start of the glossary.

```
9089 \newglossarystyle{mcoltreehypergroup}{%
```

Base it on the glostylemcoltree style:

```
9090 \setglossarystyle{mcoltree}{%
```

Put navigation links to the groups at the start of the theglossary environment:

```
9091 \renewcommand*{\glossaryheader}{%
9092   \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
9093 \renewcommand*{\glsgroupheading}[1]{%
9094   \par\noindent
9095   \glstreegroupheaderfmt{\glsnavhypertarget{\glsgroupname}{\glsgroupname}}\par
9096   \indexspace}%
9097 }
```

**mcoltreespannav** Similar to the mcoltreehypergroup style but the navigation line is put in the optional argument of the multicols environment.

```
9098 \newglossarystyle{mcoltreespannav}{%
9099   \setglossarystyle{tree}{%
9100     \renewenvironment{theglossary}{%
9101       \begin{multicols}{\glsmcols}[\noindent\glstreenavigationfmt{\glsnavigation}]
9102         \setlength{\parindent}{0pt}%
9103         \setlength{\parskip}{0pt plus 0.3pt}%
9104       }%
9105     }%
9106   \end{multicols}}}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
9107 \renewcommand*{\glsgroupheading}[1]{%
9108   \par\noindent
9109   \glstreegroupheaderfmt{\glsnavhypertarget{\glsgroupname}{\glsgroupname}}\par
9110   \indexspace}%
9111 }
```

**mcoltreenoname** Multi-column index style. Same as the treenoname, but puts the glossary in multiple columns.

```
9112 \newglossarystyle{mcoltreenoname}{%
9113   \setglossarystyle{treenoname}{%
9114     \renewenvironment{theglossary}{%
9115       \%
```

```
9116     \begin{multicols}{\glsmcols}
9117     \setlength{\parindent}{0pt}%
9118     \setlength{\parskip}{0pt plus 0.3pt}%
9119   }%
9120   {\end{multicols}}%
9121 }
```

`treenonamegroup` Like the `mcoltreenoname` style but the glossary groups have headings.

9122 \newglossarystyle{mcoltreeonenamegroup}{%

Base it on the `glostylemcoltreeonename` style:

```
9123 \setglossarystyle{mcoltreeonename}%
```

Give each group a heading:

```
9124 \renewcommand{\glsgroupheading}[1]{\par
9125   \noindent\glstreegroupheaderfmt{\glsgetgroupname{##1}}\par\indexspace}%
9126 }
```

`onamehypergroup` The `mcoltreeonenamehypergroup` style is like the `mcoltreeonenamegroup` style, but has a set of links to the groups at the start of the glossary.

```
9127 \newglossarystyle{mcoltreeonenamehypergroup}{%
```

Base it on the `glostylemcoltreeonename` style:

```
9128 \setglossarystyle{mcoltreeonename}%
```

Put navigation links to the groups at the start of the glossary environment:

```
9129 \renewcommand*\glossaryheader{%
9130   \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
9131 \renewcommand*{\glsgroupheading}[1]{%
9132   \par\noindent
9133   \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgroupname{##1}}}\par
9134   \indexspace}%
9135 }
```

`eenonamestitle` Similar to the `mcol` style but the navigation line is put in the optional argument of the `multicols` environment.

```
9136 \newglossarystyle{mcoltreenonamespannav}{%
9137   \setglossarystyle{treenoname}%
9138   \renewenvironment{theglossary}%
9139 {%
9140   \begin{multicols}{\glsmcols}[\noindent
9141     \setlength{\parindent}{0pt}%
9142     \setlength{\parskip}{0pt plus 0.3pt}%
9143 }%
9144 \end{multicols}}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
9145 \renewcommand*{\glsgroupheading}[1]{  
9146   \par\noindent
```

```

9147     \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
9148     \indexspace}%
9149 }

```

`mcolalttree` Multi-column index style. Same as the `alttree`, but puts the glossary in multiple columns.

```

9150 \newglossarystyle{mcolalttree}{%
9151   \setglossarystyle{alttree}{%
9152   \renewenvironment{theglossary}{%
9153     {%
9154       \begin{multicols}{\glsmcols}
9155         \def\@gls@prevlevel{-1}%
9156         \mbox{}\par
9157     }%
9158   {\par\end{multicols}}%
9159 }

```

`colalttreegroup` Like the `mcolalttree` style but the glossary groups have headings.

```
9160 \newglossarystyle{mcolalttreegroup}{%
```

Base it on the `glostylemcolalttree` style:

```
9161   \setglossarystyle{mcolalttree}{%
```

Give each group a heading.

```

9162   \renewcommand{\glsgroupheading}[1]{\par
9163     \def\@gls@prevlevel{-1}%
9164     \hangindent0pt\relax
9165     \parindent0pt\relax
9166     \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}\par\indexspace}%
9167 }

```

`ttrreehypergroup` The `mcolalttreehypergroup` style is like the `mcolalttreegroup` style, but has a set of links to the groups at the start of the glossary.

```
9168 \newglossarystyle{mcolalttreehypergroup}{%
```

Base it on the `glostylemcolalttree` style:

```
9169   \setglossarystyle{mcolalttree}{%
```

Put the navigation links in the header

```

9170   \renewcommand*{\glossaryheader}{%
9171     \par
9172     \def\@gls@prevlevel{-1}%
9173     \hangindent0pt\relax
9174     \parindent0pt\relax
9175     \glstreenavigationfmt{\glsnavigation}\par\indexspace}%

```

Put a hypertarget at the start of each group

```

9176   \renewcommand*{\glsgroupheading}[1]{%
9177     \par
9178     \def\@gls@prevlevel{-1}%
9179     \hangindent0pt\relax

```

```

9180   \parindent0pt\relax
9181   \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
9182   \indexspace}%
9183 }

```

`lalttreespannav` Similar to the `mcolalttreehypergroup` style but the navigation line is put in the optional argument of the `multicols` environment.

```

9184 \newglossarystyle{mcolalttreespannav}{%
9185   \setglossarystyle{alttree}%
9186   \renewenvironment{theglossary}{%
9187     {%
9188       \begin{multicols}{\glsmcols}[\noindent\glstreenavigationfmt{\glsnavigation}]
9189       \def\@gls@prevlevel{-1}%
9190       \mbox{}\par
9191     }%
9192     {\par\end{multicols}}%

```

Put a hypertarget at the start of each group

```

9193 \renewcommand*{\glsgroupheading}[1]{%
9194   \par
9195   \def\@gls@prevlevel{-1}%
9196   \hangindent0pt\relax
9197   \parindent0pt\relax
9198   \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
9199   \indexspace}%
9200 }

```

## 3.8 Glossary Styles using supertabular environment (`glossary-super` package)

The glossary styles defined in the package use the `supertabular` environment.

```
9201 \ProvidesPackage{glossary-super}[2020/03/19 v4.46 (NLCT)]
```

Requires the package:

```
9202 \RequirePackage{supertabular}
```

`\glsdescwidth` This is a length that governs the width of the description column. This may already have been defined if has been loaded.

```

9203 \@ifundefined{\glsdescwidth}{%
9204   \newlength\glsdescwidth
9205   \setlength{\glsdescwidth}{0.6\hsize}
9206 }{ }

```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined if has been loaded.

```

9207 \@ifundefined{\glspagelistwidth}{%
9208   \newlength\glspagelistwidth
9209   \setlength{\glspagelistwidth}{0.1\hsize}

```

```
9210 }{}
```

super The super glossary style uses the supertabular environment (it uses lengths defined in the package.)

```
9211 \newglossarystyle{super}{%
```

Put the glossary in a supertabular environment with two columns and no head or tail:

```
9212 \renewenvironment{theglossary}{%
9213   {\tablehead{}\tabletail{}%
9214   \begin{supertabular}{lp{\glsdescwidth}}}}%
9215   {\end{supertabular}}%
```

Do nothing at the start of the table:

```
9216 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9217 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```
9218 \renewcommand{\glossentry}[2]{%
9219   \glstarget{\glossentryname}{\glossentrydesc} &
9220   \glossentrydesc{\glspostdescription\space##2\tabularnewline
9221 }%
```

Sub entries put in a row (no name, description and page list in second column):

```
9222 \renewcommand{\subglossentry}[3]{%
9223   &
9224   \glssubentryitem{\glossentrydesc{\glspostdescription\space
9225   \glstarget{\strut}\glossentrydesc{\glspostdescription\space
9226   ##3\tabularnewline
9227 }%
```

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>)

```
9228 \ifglsnogroupskip
9229   \renewcommand*{\glsgroupskip}{}%
9230 \else
9231   \renewcommand*{\glsgroupskip}{\& \tabularnewline}%
9232 \fi
9233 }
```

superborder The superborder style is like the above, but with horizontal and vertical lines:

```
9234 \newglossarystyle{superborder}{%
```

Base it on the glostypesuper style:

```
9235 \setglossarystyle{super}{%
```

Put the glossary in a supertabular environment with two columns and a horizontal line in the head and tail:

```
9236 \renewenvironment{theglossary}{%
9237   {\tablehead{\hline}\tabletail{\hline}}%
```

```
9238     \begin{supertabular}{|l|p{\glsdescwidth}|}{}%
9239     {\end{supertabular}}%
9240 }
```

**superheader** The superheader style is like the super style, but with a header:

```
9241 \newglossarystyle{superheader}{%
```

Base it on the `glostypesuper` style:

```
9242 \setglossarystyle{super}{%
```

Put the glossary in a supertabular environment with two columns, a header and no tail:

```
9243 \renewenvironment{theglossary}{%
9244   {\tablehead{\bfseries \entryname &
9245     \bfseries\descriptionname\tabularnewline}%
9246   \tabletail{}}%
9247   \begin{supertabular}{lp{\glsdescwidth}}{}%
9248   {\end{supertabular}}%
9249 }
```

**perheaderborder** The superheaderborder style is like the super style but with a header and border:

```
9250 \newglossarystyle{superheaderborder}{%
```

Base it on the `glostypesuper` style:

```
9251 \setglossarystyle{super}{%
```

Put the glossary in a supertabular environment with two columns, a header and horizontal lines above and below the table:

```
9252 \renewenvironment{theglossary}{%
9253   {\tablehead{\hline\bfseries \entryname &
9254     \bfseries\descriptionname\tabularnewline\hline}%
9255   \tabletail{\hline}%
9256   \begin{supertabular}{|l|p{\glsdescwidth}|}{}%
9257   {\end{supertabular}}%
9258 }
```

**super3col** The super3col style is like the super style, but with 3 columns:

```
9259 \newglossarystyle{super3col}{%
```

Put the glossary in a supertabular environment with three columns and no head or tail:

```
9260 \renewenvironment{theglossary}{%
9261   {\tablehead{}\tabletail{}}%
9262   \begin{supertabular}{lp{\glsdescwidth}p{\glspagelistwidth}}{}%
9263   {\end{supertabular}}%
```

Do nothing at the start of the table:

```
9264 \renewcommand*\glossaryheader{}%
```

No group headings:

```
9265 \renewcommand*\glsgroupheading[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
9266 \renewcommand{\glossentry}[2]{%
9267   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
9268   \glossentrydesc{##1} & ##2\tabularnewline
9269 }%
```

Sub entries on a row (no name, description in second column, page list in last column):

```
9270 \renewcommand{\subglossentry}[3]{%
9271   &
9272   \glssubentryitem{##2}%
9273   \glstarget{##2}{\strut}\glossentrydesc{##2} &
9274   ##3\tabularnewline
9275 }%
```

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>)

```
9276 \ifglsnogroupskip
9277   \renewcommand*{\glsgroupskip}{}%
9278 \else
9279   \renewcommand*{\glsgroupskip}{\& \& \tabularnewline}%
9280 \fi
9281 }
```

super3colborder The super3colborder style is like the super3col style, but with a border:

```
9282 \newglossarystyle{super3colborder}{%
```

Base it on the glostypesuper3col style:

```
9283 \setglossarystyle{super3col}{}
```

Put the glossary in a supertabular environment with three columns and a horizontal line in the head and tail:

```
9284 \renewenvironment{theglossary}{%
9285   {\tablehead{\hline}\tabletail{\hline}%
9286   \begin{supertabular}{|l|p{\glsdescwidth}|p{\glspagelistwidth}|}{}%
9287   \end{supertabular}}%
9288 }
```

super3colheader The super3colheader style is like the super3col style but with a header row:

```
9289 \newglossarystyle{super3colheader}{%
```

Base it on the glostypesuper3col style:

```
9290 \setglossarystyle{super3col}{}
```

Put the glossary in a supertabular environment with three columns, a header and no tail:

```
9291 \renewenvironment{theglossary}{%
9292   {\bfseries\tablehead{\entryname&\bfseries\descriptionname&
9293     \bfseries\pagelistname\tabularnewline}\tabletail{}%}
9294   \begin{supertabular}{lp{\glsdescwidth}p{\glspagelistwidth}}{}%
9295   \end{supertabular}}%
9296 }
```

`colheaderborder` The `super3colheaderborder` style is like the `super3col` style but with a header and border:

```
9297 \newglossarystyle{super3colheaderborder}{%
```

Base it on the `glostypesuper3colborder` style:

```
9298 \setglossarystyle{super3colborder}{%
```

Put the glossary in a `supertabular` environment with three columns, a header with horizontal lines and a horizontal line in the tail:

```
9299 \renewenvironment{theglossary}{%
9300   {\tablehead{\hline
9301     \bfseries\entryname&\bfseries\descriptionname&
9302     \bfseries\pagelistname\tabularnewline\hline}%
9303   \tabletail{\hline}%
9304   \begin{supertabular}{|l|p{\glscdescwidth}|p{\glspagelistwidth}|}{}%
9305   \end{supertabular}}%
9306 }
```

`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.

```
9307 \newglossarystyle{super4col}{%
```

Put the glossary in a `supertabular` environment with four columns and no head or tail:

```
9308 \renewenvironment{theglossary}{%
9309   {\tablehead{}\tabletail{}%
9310   \begin{supertabular}{llll}{}%
9311   \end{supertabular}}%
```

Do nothing at the start of the table:

```
9312 \renewcommand*\glossaryheader{}%
```

No group headings:

```
9313 \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:

```
9314 \renewcommand*\glossentry[2]{%
9315   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
9316   \glossentrydesc{##1} &
9317   \glossentrysymbol{##1} & ##2\tabularnewline
9318 }%
```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```
9319 \renewcommand*\subglossentry[3]{%
9320   &
9321   \glssubentryitem{##2}%
9322   \glstarget{##2}{\strut}\glossentrydesc{##2} &
9323   \glossentrysymbol{##2} & ##3\tabularnewline
9324 }%
```

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>)

```
9325 \ifglsnogroupskip
9326   \renewcommand*\glsgroupskip{}%
9327 \else
9328   \renewcommand*\glsgroupskip{\& & \tabularnewline}%
9329 \fi
9330 }
```

super4colheader The super4colheader style is like the super4col but with a header row.

```
9331 \newglossarystyle{super4colheader}{%
```

Base it on the glostypesuper4col style:

```
9332 \setglossarystyle{super4col}{%
```

Put the glossary in a supertabular environment with four columns, a header and no tail:

```
9333 \renewenvironment{theglossary}{%
9334   {\tablehead{\bfseries\entryname\&\bfseries\descriptionname\&
9335     \bfseries\symbolname \&
9336     \bfseries\pagelistname\tabularnewline}%
9337   \tabletail{}%
9338   \begin{supertabular}{llll}%
9339   \end{supertabular}}%
9340 }
```

super4colborder The super4colborder style is like the super4col but with a border.

```
9341 \newglossarystyle{super4colborder}{%
```

Base it on the glostypesuper4col style:

```
9342 \setglossarystyle{super4col}{%
```

Put the glossary in a supertabular environment with four columns and a horizontal line in the head and tail:

```
9343 \renewenvironment{theglossary}{%
9344   {\tablehead{\hline}\tabletail{\hline}%
9345   \begin{supertabular}{|l|l|l|l|}%
9346   \end{supertabular}}%
9347 }
```

colheaderborder The super4colheaderborder style is like the super4col but with a header and border.

```
9348 \newglossarystyle{super4colheaderborder}{%
```

Base it on the glostypesuper4col style:

```
9349 \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:

```
9350 \renewenvironment{theglossary}{%
9351   {\tablehead{\hline\bfseries\entryname\&\bfseries\descriptionname\&
9352     \bfseries\symbolname \&
```

```

9353     \bfseries\pagelistname\tabularnewline\hline}%
9354     \tabletail{\hline}%
9355     \begin{supertabular}{|l|l|l|l|}%
9356     \end{supertabular}}%
9357 }

```

**altsuper4col** The altsuper4col glossary style is like super4col but has provision for multiline descriptions.

```
9358 \newglossarystyle{altsuper4col}{%
```

Base it on the glostypesuper4col style:

```
9359 \setglossarystyle{super4col}{%
```

Put the glossary in a supertabular environment with four columns and no head or tail:

```

9360 \renewenvironment{theglossary}%
9361   {\tablehead{}\tabletail{}%
9362   \begin{supertabular}{lp{\glsdescwidth}lp{\glspagelistwidth}}}%
9363   \end{supertabular}}%
9364 }

```

**super4colheader** The altsuper4colheader style is like the altsuper4col but with a header row.

```
9365 \newglossarystyle{altsuper4colheader}{%
```

Base it on the glostypesuper4colheader style:

```
9366 \setglossarystyle{super4colheader}{%
```

Put the glossary in a supertabular environment with four columns, a header and no tail:

```

9367 \renewenvironment{theglossary}%
9368   {\tablehead{\bfseries\entryname\&\bfseries\descriptionname\&
9369   \bfseries\symbolname\&
9370   \bfseries\pagelistname\tabularnewline}\tabletail{}%
9371   \begin{supertabular}{lp{\glsdescwidth}lp{\glspagelistwidth}}}%
9372   \end{supertabular}}%
9373 }

```

**super4colborder** The altsuper4colborder style is like the altsuper4col but with a border.

```
9374 \newglossarystyle{altsuper4colborder}{%
```

Base it on the glostypesuper4colborder style:

```
9375 \setglossarystyle{super4colborder}{%
```

Put the glossary in a supertabular environment with four columns and a horizontal line in the head and tail:

```

9376 \renewenvironment{theglossary}%
9377   {\tablehead{\hline}\tabletail{\hline}%
9378   \begin{supertabular}%
9379   {|l|p{\glsdescwidth}|l|p{\glspagelistwidth}|}%
9380   \end{supertabular}}%
9381 }

```

**colheaderborder** The altsuper4colheaderborder style is like the altsuper4col but with a header and border.

```
9382 \newglossarystyle{altsuper4colheaderborder}{%
```

Base it on the `glostylesuper4colheaderborder` style:

```
9383 \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:

```
9384 \renewenvironment{theglossary}%
9385   {\tablehead{\hline
9386     \bfseries\entryname &
9387     \bfseries\descriptionname &
9388     \bfseries\symbolname &
9389     \bfseries\pagelistname\tabularnewline\hline}%
9390   \tabletail{\hline}%
9391   \begin{supertabular}%
9392     {||l|p{\glsdescwidth}|l|p{\glspagelistwidth}|}{}%
9393   \end{supertabular}%
9394 }
```

## 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.

```
9395 \ProvidesPackage{glossary-superragged}[2020/03/19 v4.46 (NLCT)]
```

Requires the package:

```
9396 \RequirePackage{array}
```

Requires the package:

```
9397 \RequirePackage{supertabular}
```

`\glsdescwidth` This is a length that governs the width of the description column. This may already have been defined.

```
9398 \@ifundefined{\glsdescwidth}{%
9399   \newlength\glsdescwidth
9400   \setlength{\glsdescwidth}{0.6\hsize}
9401 }{}
```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```
9402 \@ifundefined{\glspagelistwidth}{%
9403   \newlength\glspagelistwidth
9404   \setlength{\glspagelistwidth}{0.1\hsize}
9405 }{}
```

`superragged` The superragged glossary style uses the `supertabular` environment.

```
9406 \newglossarystyle{superragged}{%
```

Put the glossary in a supertabular environment with two columns and no head or tail:

```
9407 \renewenvironment{theglossary}%
9408   {\tablehead{}\tabletail{}%
9409   \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}}{}%
9410   \end{supertabular}}%
```

Do nothing at the start of the table:

```
9411 \renewcommand*\glossaryheader{}%
```

No group headings:

```
9412 \renewcommand*\glsgroupheading[1]{}%
```

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```
9413 \renewcommand{\glossentry}[2]{%
9414   \glsentryitem{\#\#1}\glstarget{\#\#1}{\glossentryname{\#\#1}} &
9415   \glossentrydesc{\#\#1}\glspostdescription\space ##2%
9416   \tabularnewline
9417 }%
```

Sub entries put in a row (no name, description and page list in second column):

```
9418 \renewcommand{\subglossentry}[3]{%
9419   &
9420   \glssubentryitem{\#\#2}%
9421   \glstarget{\#\#2}{\strut}\glossentrydesc{\#\#2}\glspostdescription\space
9422   ##3%
9423   \tabularnewline
9424 }%
```

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>)

```
9425 \ifglsnogroupskip
9426   \renewcommand*\glsgroupskip{}%
9427 \else
9428   \renewcommand*\glsgroupskip{\& \tabularnewline}%
9429 \fi
9430 }
```

perraggedborder The superraggedborder style is like the above, but with horizontal and vertical lines:

```
9431 \newglossarystyle{superraggedborder}{%
```

Base it on the glostypesuperragged style:

```
9432 \setglossarystyle{superragged}{%
```

Put the glossary in a supertabular environment with two columns and a horizontal line in the head and tail:

```
9433 \renewenvironment{theglossary}%
9434   {\tablehead{\hline}\tabletail{\hline}%
9435   \begin{supertabular}{|l|>{\raggedright}p{\glsdescwidth}|}{}%
9436   \end{supertabular}}%
9437 }
```

perraggedheader The superraggedheader style is like the super style, but with a header:

```
9438 \newglossarystyle{superraggedheader}{%
```

Base it on the glostypesuperragged style:

```
9439  \setglossarystyle{superragged}{%
```

Put the glossary in a supertabular environment with two columns, a header and no tail:

```
9440 \renewenvironment{theglossary}{%
9441   {\tablehead{\bfseries \entryname & \bfseries \descriptionname
9442     \tabularnewline}%
9443   \tabletail{}%
9444   \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}}}}%
9445   {\end{supertabular}}%
9446 }
```

gedheaderborder The superraggedheaderborder style is like the superragged style but with a header and border:

```
9447 \newglossarystyle{superraggedheaderborder}{%
```

Base it on the glostypesuper style:

```
9448  \setglossarystyle{superragged}{%
```

Put the glossary in a supertabular environment with two columns, a header and horizontal lines above and below the table:

```
9449 \renewenvironment{theglossary}{%
9450   {\tablehead{\hline\bfseries \entryname &
9451     \bfseries \descriptionname\tabularnewline\hline}%
9452   \tabletail{\hline}%
9453   \begin{supertabular}{|l|>{\raggedright}p{\glsdescwidth}|}}%
9454   {\end{supertabular}}%
9455 }
```

superragged3col The superragged3col style is like the superragged style, but with 3 columns:

```
9456 \newglossarystyle{superragged3col}{%
```

Put the glossary in a supertabular environment with three columns and no head or tail:

```
9457 \renewenvironment{theglossary}{%
9458   {\tablehead{}\tabletail{}%
9459   \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}%
9460     >{\raggedright}p{\glspagelistwidth}}}}%
9461 {\end{supertabular}}%
```

Do nothing at the start of the table:

```
9462 \renewcommand*\glossaryheader{}%
```

No group headings:

```
9463 \renewcommand*\glsgroupheading[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
9464 \renewcommand{\glossentry}[2]{%
9465   \glsentryitem{\#\#1}\glisttarget{\#\#1}{\glossentryname{\#\#1}} &
9466   \glossentrydesc{\#\#1} &
```

```

9467     ##\tabularnewline
9468 }%

```

Sub entries on a row (no name, description in second column, page list in last column):

```

9469 \renewcommand{\subglossentry}[3]{%
9470   &
9471   \glssubentryitem{##2}%
9472   \glstarget{##2}{\strut}\glossentrydesc{##2} &
9473   ##3\tabularnewline
9474 }%

```

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>)

```

9475 \ifglsnogroupskip
9476   \renewcommand*{\glsgroupskip}{}%
9477 \else
9478   \renewcommand*{\glsgroupskip}{\& \& \tabularnewline}%
9479 \fi
9480 }%

```

agged3colborder The superragged3colborder style is like the superragged3col style, but with a border:

```
9481 \newglossarystyle{superragged3colborder}{%
```

Base it on the glostypesuperragged3col style:

```
9482 \setglossarystyle{superragged3col}{%
```

Put the glossary in a supertabular environment with three columns and a horizontal line in the head and tail:

```

9483 \renewenvironment{theglossary}%
9484   {\tablehead{\hline}\tabletail{\hline}%
9485   \begin{supertabular}{|l|>{\raggedright}p{\glsdescwidth}|%
9486   >{\raggedright}p{\glspagelistwidth}|}}%
9487 \end{supertabular}%
9488 }%

```

agged3colheader The superragged3colheader style is like the superragged3col style but with a header row:

```
9489 \newglossarystyle{superragged3colheader}{%
```

Base it on the glostypesuperragged3col style:

```
9490 \setglossarystyle{superragged3col}{%
```

Put the glossary in a supertabular environment with three columns, a header and no tail:

```

9491 \renewenvironment{theglossary}%
9492   {\tablehead{\bfseries\entryname\&\bfseries\descriptionname\&%
9493   \bfseries\pagelistname\tabularnewline}\tabletail{}%
9494   \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}%
9495   >{\raggedright}p{\glspagelistwidth}}}}%
9496 \end{supertabular}%
9497 }%

```

`colheaderborder` The `superragged3colheaderborder` style is like the `superragged3col` style but with a header and border:

```
9498 \newglossarystyle{superragged3colheaderborder}{%
```

Base it on the `glostypesuperragged3colborder` style:

```
9499 \setglossarystyle{superragged3colborder}{%
```

Put the glossary in a `supertabular` environment with three columns, a header with horizontal lines and a horizontal line in the tail:

```
9500 \renewenvironment{theglossary}{%
9501   {\tablehead{\hline
9502     \bfseries\entryname&\bfseries\descriptionname&
9503     \bfseries\pagelistname\tabularnewline\hline}%
9504   \tabletail{\hline}%
9505   \begin{supertabular}{|l|>{\raggedright}p{\glscdescwidth}|%
9506     >{\raggedright}p{\glspagelistwidth}|}%
9507   \end{supertabular}}%
9508 }
```

`superragged4col` The `altsuperragged4col` glossary style is like `altsuper4col` style in the package but uses ragged right formatting in the description and page list columns.

```
9509 \newglossarystyle{altsuperragged4col}{%
```

Put the glossary in a `supertabular` environment with four columns and no head or tail:

```
9510 \renewenvironment{theglossary}{%
9511   {\tablehead{}\tabletail{}%
9512   \begin{supertabular}{l>{\raggedright}p{\glscdescwidth}l%
9513     >{\raggedright}p{\glspagelistwidth}}%
9514   \end{supertabular}}%
```

Do nothing at the start of the table:

```
9515 \renewcommand*\glossaryheader{}%
```

No group headings:

```
9516 \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:

```
9517 \renewcommand{\glossentry}[2]{%
9518   \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
9519   \glossentrydesc{##1} &
9520   \glossentrysymbol{##1} & ##2\tabularnewline
9521 }%
```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```
9522 \renewcommand{\subglossentry}[3]{%
9523   &
9524   \glssubentryitem{##2}%
9525   \glstarget{##2}{\strut}\glossentrydesc{##2} &
9526   \glossentrysymbol{##2} & ##3\tabularnewline
9527 }%
```

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>)

```
9528 \ifglsnogroupskip
9529   \renewcommand*\glsgroupskip{}%
9530 \else
9531   \renewcommand*\glsgroupskip{\& & \tabularnewline}%
9532 \fi
9533 }
```

agged4colheader The altsuperragged4colheader style is like the altsuperragged4col style but with a header row.

```
9534 \newglossarystyle{altsuperragged4colheader}{%
```

Base it on the glostylealtsuperragged4col style:

```
9535 \setglossarystyle{altsuperragged4col}{%
```

Put the glossary in a supertabular environment with four columns, a header and no tail:

```
9536 \renewenvironment{theglossary}{%
9537   {\tablehead{\bfseries\entryname\&\bfseries\descriptionname\&
9538     \bfseries\symbolname \&
9539     \bfseries\pagelistname\tabularnewline}\tabletail{}%
9540   \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}l%
9541     >{\raggedright}p{\glspagelistwidth}}}}%
9542   \end{supertabular}%
9543 }
```

agged4colborder The altsuperragged4colborder style is like the altsuperragged4col style but with a border.

```
9544 \newglossarystyle{altsuperragged4colborder}{%
```

Base it on the glostylealtsuperragged4col style:

```
9545 \setglossarystyle{altsuperragged4col}{%
```

Put the glossary in a supertabular environment with four columns and a horizontal line in the head and tail:

```
9546 \renewenvironment{theglossary}{%
9547   {\tablehead{\hline}\tabletail{\hline}%
9548   \begin{supertabular}%
9549     {|l|>{\raggedright}p{\glsdescwidth}|l|%
9550       >{\raggedright}p{\glspagelistwidth}|}{}%
9551   \end{supertabular}%
9552 }
```

colheaderborder The altsuperragged4colheaderborder style is like the altsuperragged4col style but with a header and border.

```
9553 \newglossarystyle{altsuperragged4colheaderborder}{%
```

Base it on the glostylealtsuperragged4col style:

```
9554 \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:

```

9555 \renewenvironment{theglossary}%
9556   {\tablehead{\hline
9557     \bfseries\entryname &
9558     \bfseries\descriptionname &
9559     \bfseries\symbolname &
9560     \bfseries\pagelistname\tabularnewline\hline}%
9561   \tabletail{\hline}%
9562   \begin{supertabular}%
9563     {|l|>{\raggedright}p{\glsdescwidth}|l|%
9564       >{\raggedright}p{\glspagelistwidth}|}%
9565   \end{supertabular}%
9566 }

```

### 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.

```
9567 \ProvidesPackage{glossary-tree}[2020/03/19 v4.46 (NLCT)]
```

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```

9568 \providecommand{\indexspace}{%
9569   \par \vskip 10\p@ \plus 5\p@ \minus 3\p@ \relax
9570 }

```

`\glstreenamefmt` Format used to display the name in the tree styles. (This may be counteracted by `\glsnamefont`.) This command was previously also used to format the group headings.

```
9571 \newcommand*{\glstreenamefmt}[1]{\textbf{#1}}
```

`\groupheaderfmt` 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.

```
9572 \newcommand*{\glstreegroupheaderfmt}[1]{\glstreenamefmt{#1}}
```

`\navigationfmt` Format used to display the navigation header in the tree styles.

```
9573 \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.

```

9574 \ifdef{\idxitem}{%
9575   \newcommand{\glstreeitem}{\@idxitem}%
9576   \newcommand{\glstreeitem}{\par\hangindent40\p@}%
}

```

```

\glstreesubitem Level 1 item used in index style.
9577 \ifdef\subitem
9578 {\let\glstreesubitem\subitem}
9579 {\newcommand\glstreesubitem{\glstreeitem\hspace*{20\p0}}}

streessubsubitem Level 1 item used in index style.
9580 \ifdef\subsubitem
9581 {\let\glstreesubsubitem\subsubitem}
9582 {\newcommand\glstreesubsubitem{\glstreeitem\hspace*{30\p0}}}

\glstreepredesc Allow the user to adjust the space before the description (except for the alttree style).
9583 \newcommand{\glstreepredesc}{\space}

reechildpredesc Allow the user to adjust the space before the description for sub-entries (except for the treenoname and alttree style).
9584 \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.
9585 \newglossarystyle{index}{%
    Set the paragraph indentation and skip and define \item to be the same as that used by theindex:
    9586 \renewenvironment{theglossary}{%
        9587 {\setlength{\parindent}{0pt}%
        9588 \setlength{\parskip}{0pt plus 0.3pt}%
        9589 \let\item\glstreeitem
        9590 \let\subitem\glstreesubitem
        9591 \let\subsubitem\glstreesubsubitem
        9592 }%
        9593 {\par}%
    Do nothing at the start of the environment:
    9594 \renewcommand*{\glossaryheader}{}%
    No group headers:
    9595 \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.
    9596 \renewcommand*{\glossentry}[2]{%
        9597 \item\glsentryitem{\#1}\glstreenamefmt{\glstarget{\#1}{\glossentryname{\#1}}}%
        9598 \ifglshassymbol{\#1}{\space(\glossentrysymbol{\#1})}{}%
        9599 \glstreepredesc \glossentrydesc{\#1}\glspostdescription\space ##2%
        9600 }%

```

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.

```

9601 \renewcommand{\subglossentry}[3]{%
9602   \ifcase##1\relax
9603     % level 0
9604     \item
9605   \or
9606     % level 1
9607     \subitem
9608     \glssubentryitem{##2}%
9609   \else
9610     % all other levels
9611     \subsubitem
9612   \fi
9613   \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
9614   \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{ }%
9615   \glstreechildpredesc\glossentrydesc{##2}\glspostdescription\space ##3%
9616 }%

```

Vertical gap between groups is the same as that used by indices:

```
9617 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}}
```

`indexgroup` The `indexgroup` style is like the `index` style but has headings.

```
9618 \newglossarystyle{indexgroup}{%
```

Base it on the `glostyleindex` style:

```
9619 \setglossarystyle{index}{%
```

Add a heading for each group. This puts the group's title in bold followed by a vertical gap.

```

9620 \renewcommand*{\glsgroupheading}[1]{%
9621   \item\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}%
9622   \indexspace
9623 }%
9624 }
```

`indexhypergroup` The `indexhypergroup` style is like the `indexgroup` style but has hyper navigation.

```
9625 \newglossarystyle{indexhypergroup}{%
```

Base it on the `glostyleindex` style:

```
9626 \setglossarystyle{index}{%
```

Put navigation links to the groups at the start of the glossary:

```

9627 \renewcommand*{\glossaryheader}{%
9628   \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.

```

9629 \renewcommand*{\glsgroupheading}[1]{%
9630   \item\glstreegroupheaderfmt
```

```

9631      {\glsnavhypertarget{##1}{\glsgetgroupitle{##1}}}%
9632      \indexspace}%
9633 }

```

**tree** The tree glossary style is similar in style to the index style, but can have arbitrary levels.

```
9634 \newglossarystyle{tree}{%
```

Set the paragraph indentation and skip:

```

9635 \renewenvironment{theglossary}%
9636   {\setlength{\parindent}{0pt}%
9637   \setlength{\parskip}{0pt plus 0.3pt}}%
9638 {}%

```

Do nothing at the start of the theglossary environment:

```
9639 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9640 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries: name in bold, followed by symbol in brackets (if it exists), the description and the page list:

```

9641 \renewcommand{\glossentry}[2]{%
9642   \hangindent0pt\relax
9643   \parindent0pt\relax
9644   \glsentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
9645   \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
9646   \glstreepredesc\glossentrydesc{##1}\glspostdescription\space##2\par
9647 }%

```

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.

```

9648 \renewcommand{\subglossentry}[3]{%
9649   \hangindent##1\glstreeindent\relax
9650   \parindent##1\glstreeindent\relax
9651   \ifnum##1=1\relax
9652     \glssubentryitem{##2}%
9653   \fi
9654   \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
9655   \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
9656   \glstreechildpredesc\glossentrydesc{##2}\glspostdescription\space ##3\par
9657 }%

```

Vertical gap between groups is the same as that used by indices:

```
9658 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}}
```

**treegroup** Like the tree style but the glossary groups have headings.

```
9659 \newglossarystyle{treegroup}{%
```

Base it on the glostyletree style:

```
9660 \setglossarystyle{tree}%
```

Each group has a heading (in bold) followed by a vertical gap):

```
9661 \renewcommand{\glsgroupheading}[1]{\par
9662   \noindent\glstreegroupheaderfmt{\glsgetgroupname{##1}}\par
9663   \indexspace}%
9664 }
```

`treehypergroup` The `treehypergroup` style is like the `treegroup` style, but has a set of links to the groups at the start of the glossary.

```
9665 \newglossarystyle{treehypergroup}{%
```

Base it on the `glostyletree` style:

```
9666 \setglossarystyle{tree}{%
```

Put navigation links to the groups at the start of the `theglossary` environment:

```
9667 \renewcommand*{\glossaryheader}{%
9668   \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
9669 \renewcommand*{\glsgroupheading}[1]{%
9670   \par\noindent
9671   \glstreegroupheaderfmt
9672   {\glsnavhypertarget{##1}{\glsgetgroupname{##1}}}\par
9673   \indexspace}%
9674 }
```

`\glstreeindent` Length governing left indent for each level of the tree style.

```
9675 \newlength\glstreeindent
9676 \setlength{\glstreeindent}{10pt}
```

`treenoname` The `treenoname` glossary style is like the `tree` style, but doesn't print the name or symbol for sub-levels.

```
9677 \newglossarystyle{treenoname}{%
```

Set the paragraph indentation and skip:

```
9678 \renewenvironment{theglossary}{%
9679   {\setlength{\parindent}{0pt}%
9680     \setlength{\parskip}{0pt plus 0.3pt}}%
9681 }
```

No header:

```
9682 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9683 \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.

```
9684 \renewcommand{\glossentry}[2]{%
9685   \hangindent0pt\relax
9686   \parindent0pt\relax
9687   \glsentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
```

```

9688     \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
9689     \glstreepredesc\glossentrydesc{##1}\glspostdescription\space##2\par
9690 }%

```

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.

```

9691 \renewcommand{\subglossentry}[3]{%
9692   \hangindent##1\glstreeindent\relax
9693   \parindent##1\glstreeindent\relax
9694   \ifnum##1=1\relax
9695     \glssubentryitem{##2}%
9696   \fi
9697   \glstarget{##2}{\strut}%
9698   \glossentrydesc{##2}\glspostdescription\space##3\par
9699 }%

```

Vertical gap between groups is the same as that used by indices:

```

9700 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
9701 }%

```

`treenonamegroup` Like the `treenoname` style but the glossary groups have headings.

```
9702 \newglossarystyle{treenonamegroup}{%
```

Base it on the `glostyletreenoname` style:

```
9703 \setglossarystyle{treenoname}{%
```

Give each group a heading:

```

9704 \renewcommand{\glsgroupheading}[1]{\par
9705   \noindent\glstreegroupheaderfmt
9706   {\glsgetgroupname{##1}}\par\indexspace}%
9707 }%

```

`onamehypergroup` The `treenonamehypergroup` style is like the `treenonamegroup` style, but has a set of links to the groups at the start of the glossary.

```
9708 \newglossarystyle{treenonamehypergroup}{%
```

Base it on the `glostyletreenoname` style:

```
9709 \setglossarystyle{treenoname}{%
```

Put navigation links to the groups at the start of the `glossary` environment:

```

9710 \renewcommand*{\glossaryheader}{%
9711   \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%

```

Each group has a heading (in bold with a target) followed by a vertical gap):

```

9712 \renewcommand*{\glsgroupheading}[1]{%
9713   \par\noindent
9714   \glstreegroupheaderfmt
9715   {\glsnavhypertarget{##1}{\glsgetgroupname{##1}}}\par
9716   \indexspace}%
9717 }%

```

`\esttoplevelname` Find the widest name over all parentless entries in the given glossary or glossaries.

```

9718 \newrobustcmd*{\glsfindwidesttoplevelname}[1][\@glo@types]{%
9719   \dimen@=0pt\relax
9720   \gls@tmplen=0pt\relax
9721   \forallglossaries[#1]{\gls@type}%
9722   {%
9723     \forglsentries[\gls@type]{\glo@label}%
9724     {%
9725       \ifglshasparent{\glo@label}%
9726       {}%
9727       {%
9728         \settowidth{\dimen@}%
9729         {\glstreenamefmt{\glsentryname{\glo@label}}}%
9730         \ifdim\dimen@>\gls@tmplen
9731           \gls@tmplen=\dimen@
9732           \letcs{\glswidestname}{\glo@\glsdetoklabel{\glo@label}@name}%
9733           \fi
9734       }%
9735     }%
9736   }%
9737 }
```

`\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.

```

9738 \newcommand*{\glssetwidest}[2][0]{%
9739   \expandafter\def\csname@glswidestname\romannumeral#1\endcsname{%
9740     #2}%
9741 }
```

`\@glswidestname` Initialise `\@glswidestname`.

```

9742 \newcommand*{\@glswidestname}{}%
```

`\glstreenamebox` Used by the alttree style to create the box for the name and associated information.

```

9743 \newcommand*{\glstreenamebox}[2]{%
9744   \makebox[#1][1]{#2}%
9745 }
```

`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`.

```

9746 \newglossarystyle{alttree}{%
  Redefine theglossary environment.
  9747   \renewenvironment{theglossary}{%
  9748     {\def\@gls@prevlevel{-1}%
  9749      \mbox{}\par}%
  9750      \par}%
  Set the header and group headers to nothing.
  9751   \renewcommand*{\glossaryheader}{}%
  9752   \renewcommand*{\glsgroupheading}[1]{}%
```

Redefine the way that the level 0 entries are displayed.

```
9753 \renewcommand{\glossentry}[2]{%
9754   \ifnum\@gls@prevlevel=0\relax
9755   \else
```

Find out how big the indentation should be by measuring the widest entry.

```
9756   \settowidth{\glstreeindent}{\glstreenamefmt{\@glswidestname\space}}%
9757 \fi
```

Set the hangindent and paragraph indent.

```
9758 \hangindent\glstreeindent
9759 \parindent\glstreeindent
```

Put the name to the left of the paragraph block.

```
9760 \makebox[0pt][r]{\glstreenamebox{\glstreeindent}{%
9761   \glsentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}}}%
```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
9762 \ifglshassymbol{##1}{(\glossentrysymbol{##1})\space}{}%
```

Do the description followed by the description terminator and location list.

```
9763 \glossentrydesc{##1}\glspostdescription \space ##2\par
```

Set the previous level to 0.

```
9764 \def\@gls@prevlevel{0}%
9765 }%
```

Redefine the way sub-entries are displayed.

```
9766 \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.

```
9767 \ifnum##1=1\relax
9768   \glssubentryitem{##2}%
9769 \fi
```

If the level hasn't changed, keep the same settings, otherwise adjust \glstreeindent accordingly.

```
9770 \ifnum\@gls@prevlevel=##1\relax
9771 \else
```

Compute the widest entry for this level, or for level 0 if not defined for this level. Store in \gls@tmp

```
9772   \@ifundefined{@glswidestname\romannumeral##1}{%
9773     \settowidth{\gls@tmp}{\glstreenamefmt{\@glswidestname\space}}}%
9774   \settowidth{\gls@tmp}{\glstreenamefmt{%
9775     \csname @glswidestname\romannumeral##1\endcsname\space}}}%
```

Determine if going up or down a level

```
9776 \ifnum\@gls@prevlevel<##1\relax
```

Depth has increased, so add the width of the widest entry to \glstreeindent.

```
9777      \setlength\glstreeindent\gls@tmp{len}
9778      \addtolength\glstreeindent\parindent
9779      \parindent\glstreeindent
9780  \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.

```
9781      @ifundefined{@glswidestname\romannumeral@gls@prevlevel}{%
9782          \settowidth{\glstreeindent}{\glstreenamefmt{%
9783              @glswidestname\space}}}{%
9784          \settowidth{\glstreeindent}{\glstreenamefmt{%
9785              \csname @glswidestname\romannumeral@gls@prevlevel
9786                  \endcsname\space}}}{%
```

Subtract this length from the previous level's paragraph indent and set to \glstreeindent.

```
9787      \addtolength\parindent{-\glstreeindent}%
9788      \setlength\glstreeindent\parindent
9789  \fi
9790  \fi
```

Set the hanging indentation.

```
9791  \hangindent\glstreeindent
```

Put the name to the left of the paragraph block

```
9792  \makebox[0pt][r]{\glstreenamebox{\gls@tmp}{%
9793      \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}}}
```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
9794  \ifglshassymbol{##2}{(\glossentrysymbol{##2})\space}{}
```

Do the description followed by the description terminator and location list.

```
9795  \glossentrydesc{##2}\glspostdescription\space ##3\par
```

Set the previous level macro to the current level.

```
9796  \def@gls@prevlevel{##1}%
9797  }%
```

Vertical gap between groups is the same as that used by indices:

```
9798  \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
9799 }
```

**alttreegroup** Like the alttree style but the glossary groups have headings.

```
9800 \newglossarystyle{alttreegroup}{%
```

Base it on the glostylealttree style:

```
9801  \setglossarystyle{alttree}{%
```

Give each group a heading.

```
9802  \renewcommand{\glsgroupheading}[1]{\par
9803      \def@gls@prevlevel{-1}%
9804      \hangindent0pt\relax
```

```

9805   \parindent0pt\relax
9806   \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}%
9807   \par\indexspace}%
9808 }

ttreehypergroup The alttreehypergroup style is like the alttreegroup style, but has a set of links to the groups at the start of the glossary.
9809 \newglossarystyle{alttreehypergroup}{%
  Base it on the glostylealttree style:
9810   \setglossarystyle{alttree}{%
    Put the navigation links in the header
9811   \renewcommand*{\glossaryheader}{%
9812     \par
9813     \def\@gls@prevlevel{-1}%
9814     \hangindent0pt\relax
9815     \parindent0pt\relax
9816     \glstreenavigationfmt{\glsnavigation}\par\indexspace}%
    Put a hypertarget at the start of each group
9817   \renewcommand*{\glsgroupheading}[1]{%
9818     \par
9819     \def\@gls@prevlevel{-1}%
9820     \hangindent0pt\relax
9821     \parindent0pt\relax
9822     \glstreegroupheaderfmt
       {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
9824     \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.

```
9825 \NeedsTeXFormat{LaTeX2e}
9826 \ProvidesPackage{glossaries-compatible-207}[2020/03/19 v4.46 (NLCT)]
```

`AddXdyAttribute` Adds an attribute in old format.

```
9827 \ifglsxindy
9828   \renewcommand*\GlsAddXdyAttribute[1]{%
9829     \edef\@xdyattributes{\@xdyattributes ^~J \string"#1\string"}%
9830     \expandafter\toks@\expandafter{\@xdylocref}%
9831     \edef\@xdylocref{\the\toks@ ^~J%
9832       (markup-locref
9833         :open \string"\string~n\string\setentrycounter
9834           {\noexpand\glscounter}%
9835           \expandafter\string\csname#1\endcsname
9836           \expandafter@gobble\string\{\string" ^~J
9837         :close \string"\expandafter@gobble\string\}\string" ^~J
9838         :attr \string"#1\string")}}
```

Only has an effect before `\writeis`:

```
9839 \fi
```

`sAddXdyCounters`

```
9840 \renewcommand*\GlsAddXdyCounters[1]{%
9841   \GlossariesWarning{\string\GlsAddXdyCounters\space not available
9842     in compatibility mode.}%
9843 }
```

Add predefined attributes

```
9844 \GlsAddXdyAttribute{glsnumberformat}
9845 \GlsAddXdyAttribute{textrm}
9846 \GlsAddXdyAttribute{textsf}
9847 \GlsAddXdyAttribute{texttt}
9848 \GlsAddXdyAttribute{textbf}
9849 \GlsAddXdyAttribute{textmd}
9850 \GlsAddXdyAttribute{textit}
9851 \GlsAddXdyAttribute{textup}
9852 \GlsAddXdyAttribute{textsl}
```

```

9853 \GlsAddXdyAttribute{textsc}
9854 \GlsAddXdyAttribute{emph}
9855 \GlsAddXdyAttribute{glshypernumber}
9856 \GlsAddXdyAttribute{hyperrm}
9857 \GlsAddXdyAttribute{hypersf}
9858 \GlsAddXdyAttribute{hypertt}
9859 \GlsAddXdyAttribute{hyperbf}
9860 \GlsAddXdyAttribute{hypermd}
9861 \GlsAddXdyAttribute{hyperit}
9862 \GlsAddXdyAttribute{hyperup}
9863 \GlsAddXdyAttribute{hypersl}
9864 \GlsAddXdyAttribute{hypersc}
9865 \GlsAddXdyAttribute{hyperemph}

```

sAddXdyLocation Restore v2.07 definition:

```

9866 \ifglsxindy
9867   \renewcommand*\{\GlsAddXdyLocation}[2]{%
9868     \edef\xdyuserlocationdefs{%
9869       \xdyuserlocationdefs ^~J%
9870       (define-location-class \string"#1\string"~J\space\space
9871         \space(#2))
9872     }%
9873     \edef\xdyuserlocationnames{%
9874       \xdyuserlocationnames~J\space\space\space
9875       \string"#1\string"}%
9876   }
9877 \fi

```

\@do@wrglossary

```
9878 \renewcommand{\@do@wrglossary}[1]{%
```

Determine whether to use xindy or makeindex syntax

```
9879 \ifglsxindy
```

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

```

9880 \expandafter\@glo@check@mkidxrangechar\@glsnumberformat\@nil
9881 \def\@glo@range{}%
9882 \expandafter\if\@glo@prefix(\relax
9883   \def\@glo@range{:open-range}%
9884 \else
9885   \expandafter\if\@glo@prefix)\relax
9886   \def\@glo@range{:close-range}%
9887 \fi
9888 \fi

```

Get the location and escape any special characters

```
9889 \protected@edef\@glslocref{\theglsentrycounter}%
9890 \gls@checkmkidxchars\@glslocref
```

Write to the glossary file using xindy syntax.

```
9891 \glossary[\csname glo@\#1@type\endcsname]{%
```

```

9892 (indexentry :tkey (\csname glo@#1@index\endcsname)
9893   :locref \string"\@glslocref\string" %
9894   :attr \string"\@glo@suffix\string" \@glo@range
9895 )
9896 }%
9897 \else
Convert the format information into the format required for makeindex
9898 \cset@glo@numformat\glo@numfmt\gls@counter\glsnumberformat
Write to the glossary file using makeindex syntax.
9899 \glossary[\csname glo@#1@type\endcsname]{%
9900 \string\glossaryentry{\csname glo@#1@index\endcsname
9901   \gls@encapchar\glo@numfmt}{\theglsentrycounter}}%
9902 \fi
9903 }

t@glo@numformat Only had 3 arguments in v2.07
9904 \def\cset@glo@numformat#1#2#3{%
9905 \expandafter\glo@check@mkidxrangechar#3\@nil
9906 \protected@edef#1{%
9907   \glo@prefix setentrycounter[] {#2}%
9908   \expandafter\string\csname\glo@suffix\endcsname
9909 }%
9910 \gls@checkmkidxchars#1%
9911 }

\writeist Redefine \writeist back to the way it was in v2.07, but change \istfile to \glswrite.
9912 \ifglsxindy
9913 \def\writeist{%
9914   \openout\glswrite=\istfilename
9915   \write\glswrite{;; xindy style file created by the glossaries
9916     package in compatible-2.07 mode}%
9917   \write\glswrite{;; for document '\jobname' on
9918     \the\year-\the\month-\the\day}%
9919   \write\glswrite{^^J; required styles^^J}
9920   \cfor\@xdystyle:=\xdyrequiredstyles\do{%
9921     \ifx\@xdystyle\@empty
9922     \else
9923       \protected@write\glswrite{}{(require
9924         \string"\@xdystyle.xdy\string")}%
9925     \fi
9926   }%
9927   \write\glswrite{^^J%
9928     ; list of allowed attributes (number formats)^^J}%
9929   \write\glswrite{(define-attributes ((\@xdyattributes)))}%
9930   \write\glswrite{^^J; user defined alphabets^^J}%
9931   \write\glswrite{@xdyuseralphabets}%
9932   \write\glswrite{^^J; location class definitions^^J}%
9933   \protected@edef\gls@roman{\roman{0}\string"

```

```

9934     \string"roman-numbers-lowercase\string" :sep \string"}}%
9935     \@onelvel@sanitize\@gls@roman
9936     \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
9937         :sep \string"}%
9938     \@onelvel@sanitize\@tmp
9939     \ifx\@tmp\@gls@roman
9940         \write\glswrite{(define-location-class
9941             \string"roman-page-numbers\string"^^J\space\space\space
9942             (\string"roman-numbers-lowercase\string")
9943             :min-range-length \@glsminrange)}%
9944     \else
9945         \write\glswrite{(define-location-class
9946             \string"roman-page-numbers\string"^^J\space\space\space
9947             (:sep "\@gls@roman")
9948             :min-range-length \@glsminrange)}%
9949     \fi
9950     \write\glswrite{(define-location-class
9951         \string"Roman-page-numbers\string"^^J\space\space\space
9952         (\string"roman-numbers-uppercase\string")
9953         :min-range-length \@glsminrange)}%
9954     \write\glswrite{(define-location-class
9955         \string"arabic-page-numbers\string"^^J\space\space\space
9956         (\string"arabic-numbers\string")
9957         :min-range-length \@glsminrange)}%
9958     \write\glswrite{(define-location-class
9959         \string"alpha-page-numbers\string"^^J\space\space\space
9960         (\string"alpha\string")
9961         :min-range-length \@glsminrange)}%
9962     \write\glswrite{(define-location-class
9963         \string"Alpha-page-numbers\string"^^J\space\space\space
9964         (\string"ALPHA\string")
9965         :min-range-length \@glsminrange)}%
9966     \write\glswrite{(define-location-class
9967         \string"Appendix-page-numbers\string"^^J\space\space\space
9968         (\string"ALPHA\string"
9969         :sep \string"\@glsAlphacompositor\string"
9970         \string"arabic-numbers\string")
9971         :min-range-length \@glsminrange)}%
9972     \write\glswrite{(define-location-class
9973         \string"arabic-section-numbers\string"^^J\space\space\space
9974         (\string"arabic-numbers\string"
9975         :sep \string"\@glscompositor\string"
9976         \string"arabic-numbers\string")
9977         :min-range-length \@glsminrange)}%
9978     \write\glswrite{^^J; user defined location classes}%
9979     \write\glswrite{\@xdyuserlocationdefs}%
9980     \write\glswrite{^^J; define cross-reference class}%
9981     \write\glswrite{(define-crossref-class \string"see\string"
9982         :unverified )}%

```

```

9983 \write\glswrite{(\markup-crossref-list
9984   :class \string"see\string"^^J\space\space\space
9985   :open \string"\string\glsseeformat\string"
9986   :close \string"{}\string")}%
9987 \write\glswrite{^^J; define the order of the location classes}%
9988 \write\glswrite{(\define-location-class-order
9989   (\@xdylocationclassorder))}%
9990 \write\glswrite{^^J; define the glossary markup}%
9991 \write\glswrite{(\markup-index}%
9992   :open \string"\string
9993   \glossarysection[\string\glossarytoctitle]{\string
9994   \glossarytitle}\string\glossarypreamble\string~n\string\begin
9995   {theglossary}\string\glossaryheader\string~n\string" ^^J\space
9996   \space\space:close \string"\expandafter\@gobble
9997   \string\%\string~n\string
9998   \end{theglossary}\string\glossarypostamble
9999   \string~n\string" ^^J\space\space\space
10000   :tree)}%
10001 \write\glswrite{(\markup-letter-group-list
10002   :sep \string"\string\glsgroupskip\string~n\string")}%
10003 \write\glswrite{(\markup-indexentry
10004   :open \string"\string\relax \string\glsresetentrylist
10005   \string~n\string")}%
10006 \write\glswrite{(\markup-locclass-list :open
10007   \string"\glsopenbrace\string\glossaryentrynumbers
10008   \glsopenbrace\string\relax\space \string"^^J\space\space\space
10009   :sep \string", \string"
10010   :close \string"\glsclosebrace\glsclosebrace\string")}%
10011 \write\glswrite{(\markup-locref-list
10012   :sep \string"\string\delimN\space\string")}%
10013 \write\glswrite{(\markup-range
10014   :sep \string"\string\delimR\space\string")}%
10015 \@onelvel@sanitize\gls@suffixF
10016 \@onelvel@sanitize\gls@suffixFF
10017 \ifx\gls@suffixF\@empty
10018 \else
10019   \write\glswrite{(\markup-range
10020   :close "\gls@suffixF" :length 1 :ignore-end)}%
10021 \fi
10022 \ifx\gls@suffixFF\@empty
10023 \else
10024   \write\glswrite{(\markup-range
10025   :close "\gls@suffixFF" :length 2 :ignore-end)}%
10026 \fi
10027 \write\glswrite{^^J; define format to use for locations}%
10028 \write\glswrite{(\@xdylocref)}%
10029 \write\glswrite{^^J; define letter group list format}%
10030 \write\glswrite{(\markup-letter-group-list
10031   :sep \string"\string\glsgroupskip\string~n\string")}%

```

```

10032 \write\glswrite{^^J; letter group headings^^J}%
10033 \write\glswrite{(markup-letter-group
10034   :open-head \string"\string\glsgroupheading
10035   \glsopenbrace\string"^^J\space\space\space
10036   :close-head \string"\glsclosebrace\string")}%
10037 \write\glswrite{^^J; additional letter groups^^J}%
10038 \write\glswrite{@xdylettergroups}%
10039 \write\glswrite{^^J; additional sort rules^^J}%
10040 \write\glswrite{@xdysortrules}%
10041 \noist}
10042 \else
10043 \edef\@gls@actualchar{\string?}
10044 \edef\@gls@encapchar{\string!}
10045 \edef\@gls@levelchar{\string!}
10046 \edef\@gls@quotechar{\string"}
10047 \def\writeist{\relax
10048   \openout\glswrite=\listfilename
10049   \write\glswrite{\expandafter\@gobble\string\% makeindex style file
10050     created by the glossaries package}
10051   \write\glswrite{\expandafter\@gobble\string\% for document
10052     '\jobname' on \the\year-\the\month-\the\day}
10053   \write\glswrite{actual '\@gls@actualchar'}
10054   \write\glswrite{encap '\@gls@encapchar'}
10055   \write\glswrite{level '\@gls@levelchar'}
10056   \write\glswrite{quote '\@gls@quotechar'}
10057   \write\glswrite{keyword \string"\string"\glossaryentry\string"}
10058   \write\glswrite{preamble \string"\string"\glossarysection[\string
10059     \glossarytoctitle]\{\string"\string"\glossarytitle}\string
10060     \glossarypreamble\string\n\string\\begin{theglossary}\string
10061       \glossaryheader\string\n\string"}
10062   \write\glswrite{postamble \string"\string"\% \string\n\string
10063     \end{theglossary}\string\n\glossarypostamble\string\n
10064     \string"}
10065   \write\glswrite{group_skip \string"\string"\glsgroupskip\string\n
10066     \string"}
10067   \write\glswrite{item_0 \string"\string"\% \string\n\string"}
10068   \write\glswrite{item_1 \string"\string"\% \string\n\string"}
10069   \write\glswrite{item_2 \string"\string"\% \string\n\string"}
10070   \write\glswrite{item_01 \string"\string"\% \string\n\string"}
10071   \write\glswrite{item_x1
10072     \string"\string"\relax \string\\glsresetentrylist\string\n
10073     \string"}
10074   \write\glswrite{item_12 \string"\string"\% \string\n\string"}
10075   \write\glswrite{item_x2
10076     \string"\string"\relax \string\\glsresetentrylist\string\n
10077     \string"}
10078   \write\glswrite{delim_0 \string"\string"\{\string
10079     \glossaryentrynumbers\string\{\string\relax \string"\string"
10080   \write\glswrite{delim_1 \string"\string"\{\string

```

```

10081      \\glossaryentrynumbers\string{\string\\relax \string"}
10082      \write\glswrite{delim_2 \string"\string\{\string"
10083          \\glossaryentrynumbers\string{\string\\relax \string"}
10084      \write\glswrite{delim_t \string"\string\}\string{}\string"}}
10085      \write\glswrite{delim_n \string"\string\string\\delimN \string"}
10086      \write\glswrite{delim_r \string"\string\string\\delimR \string"}
10087      \write\glswrite{headings_flag 1}
10088      \write\glswrite{heading_prefix
10089          \string"\string\glsgroupheading\string\{\string"
10090      \write\glswrite{heading_suffix
10091          \string"\string\string\}\string\\relax
10092          \string"\string\glsresetentrylist \string"
10093      \write\glswrite{symhead_positive \string"\string"glssymbols\string"}
10094      \write\glswrite{numhead_positive \string"\string"glsnrnumbers\string"}
10095      \write\glswrite{page_compositor \string"\string"\glscompositor\string"}
10096      \gls@escbsdq\gls@suffixF
10097      \gls@escbsdq\gls@suffixFF
10098      \ifx\gls@suffixF\empty
10099      \else
10100          \write\glswrite{suffix_2p \string"\string"\gls@suffixF\string"}
10101      \fi
10102      \ifx\gls@suffixFF\empty
10103      \else
10104          \write\glswrite{suffix_3p \string"\string"\gls@suffixFF\string"}
10105      \fi
10106      \noist
10107  }
10108 \fi

\noist
10109 \renewcommand*\noist{\let\writeist\relax}

```

## 4.2 glossaries-compatible-307

```

10110 \NeedsTeXFormat{LaTeX2e}
10111 \ProvidesPackage{glossaries-compatible-307}[2020/03/19 v4.46 (NLCT)]

```

Compatibility macros for predefined glossary styles:

`atglossarystyle` Defines a compatibility glossary style.

```

10112 \newcommand{\compatglossarystyle}[2]{%
10113     \ifcsundef{@glscompstyle@#1}%
10114     {%
10115         \csdef{@glscompstyle@#1}{#2}%
10116     }%
10117     {%
10118         \PackageError{glossaries}{Glossary compatibility style '#1' is already defined}{}%
10119     }%
10120 }

```

Backward compatible inline style.

```
10121 \compatglossarystyle{inline}{%
10122   \renewcommand{\glossaryentryfield}[5]{%
10123     \glsinlinedopostchild
10124     \gls@inlinesep
10125     \def\glo@desc{##3}%
10126     \def\@no@post@desc{\nopo@desc}%
10127     \glsentryitem{##1}\glsinlinenameformat{##1}{##2}%
10128     \ifx\glo@desc\@no@post@desc
10129       \glsinlineemptydescformat{##4}{##5}%
10130     \else
10131       \ifstrempty{##3}%
10132         {\glsinlineemptydescformat{##4}{##5}}%
10133         {\glsinlinedescformat{##3}{##4}{##5}}%
10134     \fi
10135     \ifglshaschildren{##1}%
10136     {%
10137       \glsresetsubentrycounter
10138       \glsinlineparentchildseparator
10139       \def\gls@inlinesubsep{}%
10140       \def\gls@inlinepostchild{\glsinlinepostchild}%
10141     }%
10142     {}%
10143     \def\gls@inlinesep{\glsinlineseparator}%
10144   }%
```

Sub-entries display description:

```
10145 \renewcommand{\glossarysubentryfield}[6]{%
10146   \gls@inlinesubsep%
10147   \glsinlinesubnameformat{##2}{##3}%
10148   \glssubentryitem{##2}\glsinlinesubdescformat{##4}{##5}{##6}%
10149   \def\gls@inlinesubsep{\glsinlinesubseparator}%
10150 }%
10151 }
```

Backward compatible list style.

```
10152 \compatglossarystyle{list}{%
10153   \renewcommand*\glossaryentryfield[5]{%
10154     \item[\glsentryitem{##1}\glstarget{##1}{##2}]
10155     ##3\glspostdescription\space ##5}%
10156 }
```

Sub-entries continue on the same line:

```
10156 \renewcommand*\glossarysubentryfield[6]{%
10157   \glssubentryitem{##2}%
10158   \glstarget{##2}{\strut}##4\glspostdescription\space ##6.}%
10159 }
```

Backward compatible listgroup style.

```
10160 \compatglossarystyle{listgroup}{%
10161   \csuse{@glscompstyle@list}%
10162 }%
```

Backward compatible listhypergroup style.

```
10163 \compatglossarystyle{listhypergroup}{%
10164   \csuse{@glscompstyle@list}%
10165 }%
```

Backward compatible altlist style.

```
10166 \compatglossarystyle{altlist}{%
10167   \renewcommand*\glossaryentryfield}[5]{%
10168     \item[\glsentryitem{##1}\glstarget{##1}{##2}]%
10169       \mbox{}\par\nobreak\@afterheading
10170         ##3\glspostdescription\space ##5}%
10171   \renewcommand*\glossarysubentryfield}[6]{%
10172     \par
10173     \glssubentryitem{##2}%
10174     \glstarget{##2}{\strut}##4\glspostdescription\space ##6}%
10175 }%
```

Backward compatible altlistgroup style.

```
10176 \compatglossarystyle{altlistgroup}{%
10177   \csuse{@glscompstyle@altlist}%
10178 }%
```

Backward compatible altlisthypergroup style.

```
10179 \compatglossarystyle{altlisthypergroup}{%
10180   \csuse{@glscompstyle@altlist}%
10181 }%
```

Backward compatible listdotted style.

```
10182 \compatglossarystyle{listdotted}{%
10183   \renewcommand*\glossaryentryfield}[5]{%
10184     \item[]\makebox[\glslistdottedwidth][1]{%
10185       \glsentryitem{##1}\glstarget{##1}{##2}%
10186       \unskip\leaders\hbox to 2.9mm{\hss.\hfill\strut}##3}%
10187   \renewcommand*\glossarysubentryfield}[6]{%
10188     \item[]\makebox[\glslistdottedwidth][1]{%
10189       \glssubentryitem{##2}%
10190       \glstarget{##2}{##3}%
10191       \unskip\leaders\hbox to 2.9mm{\hss.\hfill\strut}##4}%
10192 }%
```

Backward compatible sublistdotted style.

```
10193 \compatglossarystyle{sublistdotted}{%
10194   \csuse{@glscompstyle@listdotted}%
10195   \renewcommand*\glossaryentryfield}[5]{%
10196     \item[\glsentryitem{##1}\glstarget{##1}{##2}]}%
10197 }%
```

Backward compatible long style.

```
10198 \compatglossarystyle{long}{%
10199   \renewcommand*\glossaryentryfield}[5]{%
10200     \glsentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
10201   \renewcommand*\glossarysubentryfield}[6]{%
```

```

10202     &
10203     \glssubentryitem{##2}%
10204     \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
10205 }%}

    Backward compatible longborder style.

10206 \compatglossarystyle{longborder}{%
10207   \csuse{@glscompstyle@long}%
10208 }%}

    Backward compatible longheader style.

10209 \compatglossarystyle{longheader}{%
10210   \csuse{@glscompstyle@long}%
10211 }%}

    Backward compatible longheaderborder style.

10212 \compatglossarystyle{longheaderborder}{%
10213   \csuse{@glscompstyle@long}%
10214 }%}

    Backward compatible long3col style.

10215 \compatglossarystyle{long3col}{%
10216   \renewcommand*\glossaryentryfield}[5]{%
10217     \glsentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\\}%
10218   \renewcommand*\glossarysubentryfield}[6]{%
10219     &
10220     \glssubentryitem{##2}%
10221     \glstarget{##2}{\strut}##4 & ##6\\}%
10222 }%}

    Backward compatible long3colborder style.

10223 \compatglossarystyle{long3colborder}{%
10224   \csuse{@glscompstyle@long3col}%
10225 }%}

    Backward compatible long3colheader style.

10226 \compatglossarystyle{long3colheader}{%
10227   \csuse{@glscompstyle@long3col}%
10228 }%}

    Backward compatible long3colheaderborder style.

10229 \compatglossarystyle{long3colheaderborder}{%
10230   \csuse{@glscompstyle@long3col}%
10231 }%}

    Backward compatible long4col style.

10232 \compatglossarystyle{long4col}{%
10233   \renewcommand*\glossaryentryfield}[5]{%
10234     \glsentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%
10235   \renewcommand*\glossarysubentryfield}[6]{%
10236     &
10237     \glssubentryitem{##2}%

```

```

10238      \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%
10239 }%
    Backward compatible long4colheader style.
10240 \compatglossarystyle{long4colheader}{%
10241   \csuse{@glscompstyle@long4col}%
10242 }%
    Backward compatible long4colborder style.
10243 \compatglossarystyle{long4colborder}{%
10244   \csuse{@glscompstyle@long4col}%
10245 }%
    Backward compatible long4colheaderborder style.
10246 \compatglossarystyle{long4colheaderborder}{%
10247   \csuse{@glscompstyle@long4col}%
10248 }%
    Backward compatible altlong4col style.
10249 \compatglossarystyle{altlong4col}{%
10250   \csuse{@glscompstyle@long4col}%
10251 }%
    Backward compatible altlong4colheader style.
10252 \compatglossarystyle{altlong4colheader}{%
10253   \csuse{@glscompstyle@long4col}%
10254 }%
    Backward compatible altlong4colborder style.
10255 \compatglossarystyle{altlong4colborder}{%
10256   \csuse{@glscompstyle@long4col}%
10257 }%
    Backward compatible altlong4colheaderborder style.
10258 \compatglossarystyle{altlong4colheaderborder}{%
10259   \csuse{@glscompstyle@long4col}%
10260 }%
    Backward compatible long style.
10261 \compatglossarystyle{longragged}{%
10262   \renewcommand*\glossaryentryfield}[5]{%
10263     \glsentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
10264     \tabularnewline}%
10265 \renewcommand*\glossarysubentryfield}[6]{%
10266   &
10267   \glssubentryitem{##2}%
10268   \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
10269   \tabularnewline}%
10270 }%
    Backward compatible longraggedborder style.
10271 \compatglossarystyle{longraggedborder}{%
10272   \csuse{@glscompstyle@longragged}%
10273 }%

```

Backward compatible longraggedheader style.

```
10274 \compatglossarystyle{longraggedheader}{%
10275  \csuse{@glscompstyle@longragged}%
10276 }%
```

Backward compatible longraggedheaderborder style.

```
10277 \compatglossarystyle{longraggedheaderborder}{%
10278  \csuse{@glscompstyle@longragged}%
10279 }%
```

Backward compatible longragged3col style.

```
10280 \compatglossarystyle{longragged3col}{%
10281  \renewcommand*\glossaryentryfield}[5]{%
10282    \glstarget{##1}{\glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%
10283  \renewcommand*\glossarysubentryfield}[6]{%
10284    &
10285    \glssubentryitem{##2}%
10286    \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%
10287 }%
```

Backward compatible longragged3colborder style.

```
10288 \compatglossarystyle{longragged3colborder}{%
10289  \csuse{@glscompstyle@longragged3col}%
10290 }%
```

Backward compatible longragged3colheader style.

```
10291 \compatglossarystyle{longragged3colheader}{%
10292  \csuse{@glscompstyle@longragged3col}%
10293 }%
```

Backward compatible longragged3colheaderborder style.

```
10294 \compatglossarystyle{longragged3colheaderborder}{%
10295  \csuse{@glscompstyle@longragged3col}%
10296 }%
```

Backward compatible altlongragged4col style.

```
10297 \compatglossarystyle{altnlongragged4col}{%
10298  \renewcommand*\glossaryentryfield}[5]{%
10299    \glstarget{##1}{\glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%
10300  \renewcommand*\glossarysubentryfield}[6]{%
10301    &
10302    \glssubentryitem{##2}%
10303    \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%
10304 }%
```

Backward compatible altnlongragged4colheader style.

```
10305 \compatglossarystyle{altnlongragged4colheader}{%
10306  \csuse{@glscompstyle@altnlong4col}%
10307 }%
```

Backward compatible altnlongragged4colborder style.

```
10308 \compatglossarystyle{altnlongragged4colborder}{%
```

```

10309 \csuse{@glscompstyle@altlong4col}%
10310 }%
    Backward compatible altlongragged4colheaderborder style.
10311 \compatglossarystyle{altlongragged4colheaderborder}{%
10312 \csuse{@glscompstyle@altlong4col}%
10313 }%
    Backward compatible index style.
10314 \compatglossarystyle{index}{%
10315 \renewcommand*\glossaryentryfield}[5]{%
10316 \item\glsentryitem{##1}\textbf{\glstarget{##1}{##2}}%
10317 \ifx\relax##4\relax
10318 \else
10319 \space##4}%
10320 \fi
10321 \space##3\glspostdescription \space##5}%
10322 \renewcommand*\glossarysubentryfield}[6]{%
10323 \ifcase##1\relax
10324 % level 0
10325 \item
10326 \or
10327 % level 1
10328 \subitem
10329 \glssubentryitem{##2}}%
10330 \else
10331 % all other levels
10332 \subsubitem
10333 \fi
10334 \textbf{\glstarget{##2}{##3}}%
10335 \ifx\relax##5\relax
10336 \else
10337 \space##5}%
10338 \fi
10339 \space##4\glspostdescription\space##6}%
10340 }%
    Backward compatible indexgroup style.
10341 \compatglossarystyle{indexgroup}{%
10342 \csuse{@glscompstyle@index}%
10343 }%
    Backward compatible indexhypergroup style.
10344 \compatglossarystyle{indexhypergroup}{%
10345 \csuse{@glscompstyle@index}%
10346 }%
    Backward compatible tree style.
10347 \compatglossarystyle{tree}{%
10348 \renewcommand*\glossaryentryfield}[5]{%
10349 \hangindent0pt\relax

```

```

10350 \parindent0pt\relax
10351 \glsentryitem{##1}\textbf{\glstarget{##1}{##2}}%
10352 \ifx\relax##4\relax
10353 \else
10354 \space(##4)%
10355 \fi
10356 \space ##3\glspostdescription \space ##5\par}%
10357 \renewcommand{\glossarysubentryfield}[6]{%
10358 \hangindent##1\glstreeindent\relax
10359 \parindent##1\glstreeindent\relax
10360 \ifnum##1=1\relax
10361 \glssubentryitem{##2}%
10362 \fi
10363 \textbf{\glstarget{##2}{##3}}%
10364 \ifx\relax##5\relax
10365 \else
10366 \space(##5)%
10367 \fi
10368 \space##4\glspostdescription\space ##6\par}%
10369 }%

```

Backward compatible treegroup style.

```

10370 \compatglossarystyle{treegroup}{%
10371 \csuse{@glscompstyle@tree}%
10372 }%

```

Backward compatible treehypergroup style.

```

10373 \compatglossarystyle{treehypergroup}{%
10374 \csuse{@glscompstyle@tree}%
10375 }%

```

Backward compatible treenoname style.

```

10376 \compatglossarystyle{treenoname}{%
10377 \renewcommand{\glossaryentryfield}[5]{%
10378 \hangindent0pt\relax
10379 \parindent0pt\relax
10380 \glsentryitem{##1}\textbf{\glstarget{##1}{##2}}%
10381 \ifx\relax##4\relax
10382 \else
10383 \space(##4)%
10384 \fi
10385 \space ##3\glspostdescription \space ##5\par}%
10386 \renewcommand{\glossarysubentryfield}[6]{%
10387 \hangindent##1\glstreeindent\relax
10388 \parindent##1\glstreeindent\relax
10389 \ifnum##1=1\relax
10390 \glssubentryitem{##2}%
10391 \fi
10392 \glstarget{##2}{\strut}%
10393 ##4\glspostdescription\space ##6\par}%
10394 }%

```

Backward compatible treenonamegroup style.

```
10395 \compatglossarystyle{treenonamegroup}{%
10396   \csuse{@glscompstyle@treenoname}%
10397 }%
```

Backward compatible treenonamehypergroup style.

```
10398 \compatglossarystyle{treenonamehypergroup}{%
10399   \csuse{@glscompstyle@treenoname}%
10400 }%
```

Backward compatible alttree style.

```
10401 \compatglossarystyle{alttree}{%
10402   \renewcommand{\glossaryentryfield}[5]{%
10403     \ifnum@gls@prevlevel=0\relax
10404       \else
10405         \settowidth{\glstreeindent}{\textbf{@glswidestname\space}}%
10406         \hangindent\glstreeindent
10407         \parindent\glstreeindent
10408       \fi
10409       \makebox[0pt][r]{\makebox[\glstreeindent][1]{%
10410         \glsentryitem{##1}\textbf{\glstarget{##1}{##2}}}}%
10411       \ifx\relax##4\relax
10412         \else
10413           (##4)\space
10414         \fi
10415       ##3\glspostdescription \space ##5\par
10416       \def@gls@prevlevel{0}%
10417   }%
10418   \renewcommand{\glossarysubentryfield}[6]{%
10419     \ifnum##1=1\relax
10420       \glssubentryitem{##2}%
10421     \fi
10422     \ifnum@gls@prevlevel=##1\relax
10423     \else
10424       \@ifundefined{@glswidestname\romannumeral##1}{%
10425         \settowidth{\gls@tmp[1]}{\textbf{@glswidestname\space}}%
10426         \settowidth{\gls@tmp[1]}{\textbf{%
10427           \csname @glswidestname\romannumeral##1\endcsname\space}}%
10428       \ifnum@gls@prevlevel<##1\relax
10429         \setlength\glstreeindent{\gls@tmp[1]}
10430         \addtolength\glstreeindent\parindent
10431         \parindent\glstreeindent
10432       \else
10433         \ifundefined{@glswidestname\romannumeral\gls@prevlevel}{%
10434           \settowidth{\glstreeindent}{\textbf{%
10435             @glswidestname\space}}%
10436           \settowidth{\glstreeindent}{\textbf{%
10437             \csname @glswidestname\romannumeral\gls@prevlevel\endcsname\space}}%
10438         \addtolength\parindent{-\glstreeindent}%
10439       }
```

```

10440      \setlength\glstreeindent\parindent
10441      \fi
10442      \fi
10443      \hangindent\glstreeindent
10444      \makebox[0pt][r]{\makebox[\gls@tmpplen][1]{%
10445          \textbf{\glstarget{##2}{##3}}}}%
10446      \ifx##5\relax\relax
10447      \else
10448          (##5)\space
10449      \fi
10450      ##4\glspostdescription\space ##6\par
10451      \def\@gls@prevlevel{##1}%
10452 }%
10453 }%

```

Backward compatible alttreegroup style.

```

10454 \compatglossarystyle{alttreegroup}{%
10455 \csuse{@glscompstyle@alttree}%
10456 }%

```

Backward compatible alttreehypergroup style.

```

10457 \compatglossarystyle{alttreehypergroup}{%
10458 \csuse{@glscompstyle@alttree}%
10459 }%

```

Backward compatible mcolindex style.

```

10460 \compatglossarystyle{mcolindex}{%
10461 \csuse{@glscompstyle@index}%
10462 }%

```

Backward compatible mcolindexgroup style.

```

10463 \compatglossarystyle{mcolindexgroup}{%
10464 \csuse{@glscompstyle@index}%
10465 }%

```

Backward compatible mcolindexhypergroup style.

```

10466 \compatglossarystyle{mcolindexhypergroup}{%
10467 \csuse{@glscompstyle@index}%
10468 }%

```

Backward compatible mcoltree style.

```

10469 \compatglossarystyle{mcoltree}{%
10470 \csuse{@glscompstyle@tree}%
10471 }%

```

Backward compatible mcoltreegroup style.

```

10472 \compatglossarystyle{mcolindextreegroup}{%
10473 \csuse{@glscompstyle@tree}%
10474 }%

```

Backward compatible mcoltreehypergroup style.

```

10475 \compatglossarystyle{mcolindextreehypergroup}{%

```

```

10476 \csuse{@glscompstyle@tree}%
10477 }%
    Backward compatible mcoltreeonename style.
10478 \compatglossarystyle{mcoltreeonename}{%
10479 \csuse{@glscompstyle@tree}%
10480 }%
    Backward compatible mcoltreeonenamegroup style.
10481 \compatglossarystyle{mcoltreeonenamegroup}{%
10482 \csuse{@glscompstyle@tree}%
10483 }%
    Backward compatible mcoltreeonenamehypergroup style.
10484 \compatglossarystyle{mcoltreeonenamehypergroup}{%
10485 \csuse{@glscompstyle@tree}%
10486 }%
    Backward compatible mcolalttree style.
10487 \compatglossarystyle{mcolalttree}{%
10488 \csuse{@glscompstyle@alttree}%
10489 }%
    Backward compatible mcolalttreegroup style.
10490 \compatglossarystyle{mcolalttreegroup}{%
10491 \csuse{@glscompstyle@alttree}%
10492 }%
    Backward compatible mcolalttreehypergroup style.
10493 \compatglossarystyle{mcolalttreehypergroup}{%
10494 \csuse{@glscompstyle@alttree}%
10495 }%
    Backward compatible superragged style.
10496 \compatglossarystyle{superragged}{%
10497 \renewcommand*\glossaryentryfield}[5]{%
10498 \glsentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
10499 \tabularnewline}%
10500 \renewcommand*\glossarysubentryfield}[6]{%
10501 &
10502 \glssubentryitem{##2}%
10503 \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
10504 \tabularnewline}%
10505 }%
    Backward compatible superraggedborder style.
10506 \compatglossarystyle{superraggedborder}{%
10507 \csuse{@glscompstyle@superragged}%
10508 }%
    Backward compatible superraggedheader style.
10509 \compatglossarystyle{superraggedheader}{%
10510 \csuse{@glscompstyle@superragged}%
10511 }%

```

Backward compatible superraggedheaderborder style.

```
10512 \compatglossarystyle{superraggedheaderborder}{%
10513   \csuse{@glscompstyle@superragged}%
10514 }%
```

Backward compatible superragged3col style.

```
10515 \compatglossarystyle{superragged3col}{%
10516   \renewcommand*\glossaryentryfield}[5]{%
10517     \glstarget{\glsentryitem[\#1]}{\glstarget{\#1\#2} & \#3 & \#5\tabularnewline}%
10518   \renewcommand*\glossarysubentryfield}[6]{%
10519     &
10520     \glssubentryitem[\#2]%
10521     \glstarget{\#2\{\strut\}\#4 & \#6\tabularnewline}%
10522 }%
```

Backward compatible superragged3colborder style.

```
10523 \compatglossarystyle{superragged3colborder}{%
10524   \csuse{@glscompstyle@superragged3col}%
10525 }%
```

Backward compatible superragged3colheader style.

```
10526 \compatglossarystyle{superragged3colheader}{%
10527   \csuse{@glscompstyle@superragged3col}%
10528 }%
```

Backward compatible superragged3colheaderborder style.

```
10529 \compatglossarystyle{superragged3colheaderborder}{%
10530   \csuse{@glscompstyle@superragged3col}%
10531 }%
```

Backward compatible altsuperragged4col style.

```
10532 \compatglossarystyle{altsuperragged4col}{%
10533   \renewcommand*\glossaryentryfield}[5]{%
10534     \glstarget{\glsentryitem[\#1]}{\glstarget{\#1\#2} & \#3 & \#4 & \#5\tabularnewline}%
10535   \renewcommand*\glossarysubentryfield}[6]{%
10536     &
10537     \glssubentryitem[\#2]%
10538     \glstarget{\#2\{\strut\}\#4 & \#5 & \#6\tabularnewline}%
10539 }%
```

Backward compatible altsuperragged4colheader style.

```
10540 \compatglossarystyle{altsuperragged4colheader}{%
10541   \csuse{@glscompstyle@altsuperragged4col}%
10542 }%
```

Backward compatible altsuperragged4colborder style.

```
10543 \compatglossarystyle{altsuperragged4colborder}{%
10544   \csuse{@glscompstyle@altsuperragged4col}%
10545 }%
```

Backward compatible altsuperragged4colheaderborder style.

```
10546 \compatglossarystyle{altsuperragged4colheaderborder}{%
```

```

10547 \csuse{@glscompstyle@altsuperragged4col}%
10548 }%
    Backward compatible super style.

10549 \compatglossarystyle{super}{%
10550 \renewcommand*\glossaryentryfield}[5]{%
10551 \glsentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
10552 \renewcommand*\glossarysubentryfield}[6]{%
10553 &
10554 \glssubentryitem{##2}%
10555 \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
10556 }%
    Backward compatible superborder style.

10557 \compatglossarystyle{superborder}{%
10558 \csuse{@glscompstyle@super}%
10559 }%
    Backward compatible superheader style.

10560 \compatglossarystyle{superheader}{%
10561 \csuse{@glscompstyle@super}%
10562 }%
    Backward compatible superheaderborder style.

10563 \compatglossarystyle{superheaderborder}{%
10564 \csuse{@glscompstyle@super}%
10565 }%
    Backward compatible super3col style.

10566 \compatglossarystyle{super3col}{%
10567 \renewcommand*\glossaryentryfield}[5]{%
10568 \glsentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\\}%
10569 \renewcommand*\glossarysubentryfield}[6]{%
10570 &
10571 \glssubentryitem{##2}%
10572 \glstarget{##2}{\strut}##4 & ##6\\}%
10573 }%
    Backward compatible super3colborder style.

10574 \compatglossarystyle{super3colborder}{%
10575 \csuse{@glscompstyle@super3col}%
10576 }%
    Backward compatible super3colheader style.

10577 \compatglossarystyle{super3colheader}{%
10578 \csuse{@glscompstyle@super3col}%
10579 }%
    Backward compatible super3colheaderborder style.

10580 \compatglossarystyle{super3colheaderborder}{%
10581 \csuse{@glscompstyle@super3col}%
10582 }%

```

Backward compatible super4col style.

```
10583 \compatglossarystyle{super4col}{%
10584   \renewcommand*{\glossaryentryfield}[5]{%
10585     \glsetentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\}%
10586   \renewcommand*{\glossarysubentryfield}[6]{%
10587     &
10588     \glssubentryitem{##2}%
10589     \glstarget{##2}{\strut}##4 & ##5 & ##6\}%
10590 }%
```

Backward compatible super4colheader style.

```
10591 \compatglossarystyle{super4colheader}{%
10592   \csuse{@glscompstyle@super4col}%
10593 }%
```

Backward compatible super4colborder style.

```
10594 \compatglossarystyle{super4colborder}{%
10595   \csuse{@glscompstyle@super4col}%
10596 }%
```

Backward compatible super4colheaderborder style.

```
10597 \compatglossarystyle{super4colheaderborder}{%
10598   \csuse{@glscompstyle@super4col}%
10599 }%
```

Backward compatible altsuper4col style.

```
10600 \compatglossarystyle{altsuper4col}{%
10601   \csuse{@glscompstyle@super4col}%
10602 }%
```

Backward compatible altsuper4colheader style.

```
10603 \compatglossarystyle{altsuper4colheader}{%
10604   \csuse{@glscompstyle@super4col}%
10605 }%
```

Backward compatible altsuper4colborder style.

```
10606 \compatglossarystyle{altsuper4colborder}{%
10607   \csuse{@glscompstyle@super4col}%
10608 }%
```

Backward compatible altsuper4colheaderborder style.

```
10609 \compatglossarystyle{altsuper4colheaderborder}{%
10610   \csuse{@glscompstyle@super4col}%
10611 }%
```

## 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.

```
10612 \NeedsTeXFormat{LaTeX2e}
```

Package version number now in line with main glossaries package number.

```
10613 \ProvidesPackage{glossaries-accsupp}[2020/03/19 v4.46 (NLCT)
```

```
10614 Experimental glossaries accessibility]
```

Pass all options to glossaries:

```
10615 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
10616 \ProcessOptions
```

This package should be loaded before glossaries-extra, so complain if that has already been loaded.

```
10617 \@ifpackageloaded{glossaries-extra}
```

```
10618 {%
```

If the accsupp option was used, \@glsxtr@doaccsupp will have been set, otherwise it will be empty.

```
10619 \ifx\@glsxtr@doaccsupp\empty
10620 \GlossariesWarning{The ‘glossaries-accsupp’
10621 package has been loaded\MessageBreak
10622 after the ‘glossaries-extra’ package. This\MessageBreak
10623 can cause a failure to integrate both packages.\MessageBreak
10624 Either use the ‘accsupp’ option when you load\MessageBreak
10625 ‘glossaries-extra’ or load ‘glossaries-accsupp’\MessageBreak
10626 before loading ‘glossaries-extra’}%
10627 \fi
10628 }
10629 {}
```

tibleglossentry Override style compatibility macros:

```
10630 \def\compatibileglossentry#1#2{%
10631 \toks@{\#2}%
10632 \protected\edef\@do@glossentry{%
10633 \noexpand\accsuppglossaryentryfield{#1}%
10634 {\noexpand\glsnamefont
10635 \expandafter\expandonce\csname glo@\glsdetoklabel{#1}@name\endcsname}%
10636 }
```

```

10636     {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@desc\endcsname}%
10637     {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@symbol\endcsname}%
10638     {\the\toks@}%
10639 }%
10640 \do@glossentry
10641 }

```

#### lesubglossentry

```

10642 \def\compatiblesubglossentry#1#2#3{%
10643   \toks@{#3}%
10644   \protected@edef\do@subglossentry{%
10645     \noexpand\acccsuppglossarysubentryfield{\number#1}%
10646     {#2}%
10647     {\noexpand\glsnamefont
10648       {\expandafter\expandonce\csname glo@\glsdetoklabel{#2}@name\endcsname}%
10649       {\expandafter\expandonce\csname glo@\glsdetoklabel{#2}@desc\endcsname}%
10650       {\expandafter\expandonce\csname glo@\glsdetoklabel{#2}@symbol\endcsname}%
10651       {\the\toks@}%
10652     }%
10653     \do@subglossentry
10654 }

```

Required packages:

```
10655 \RequirePackage{glossaries}
```

@acccsupp@engine There's currently only support for acccsupp, but if you define \gls@acccsupp@engine before loading glossaries-acccsupp, you can prevent acccsupp from being loaded. Redefining this command after glossaries-acccsupp has loaded obviously won't do anything (and so is an internal command to deter casual use). If it is defined to something other than acccsupp then \gls@accessibility will need to be defined to something appropriate.

```
10656 \providecommand{\gls@acccsupp@engine}{acccsupp}
```

s@accessibility \gls@accessibility{<options>}{{PDF element}}{<value>}{<content>}

```

10657 \providecommand{\gls@accessibility}[4]{#4}
10658 \ifdefstring\gls@acccsupp@engine{acccsupp}{%
10659 {
10660   \RequirePackage{acccsupp}
10661   \renewcommand{\gls@accessibility}[4]{%
10662     \BeginAccSupp{#1,#2={#3}}#4\EndAccSupp{}%
10663   }
10664 }
10665 {}}

```

lsaccessibility \glsaccessibility[<options>]{{PDF element}}{<value>}{<content>}

User-level command that includes debug info if required.

```

10666 \newcommand{\glsaccessibility}[4] []{%
10667  \@glsshowaccsupp{#1}{#2}{#3}{#4}%
10668  \gls@accessibility{#1}{#2}{#3}{#4}%
10669 }

```

## 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:

```

10670 \define@key{glossentry}{access}{%
10671  \def\@glo@access{#1}%
10672 }

```

**textaccess** The replacement text corresponding to the text key:

```

10673 \define@key{glossentry}{textaccess}{%
10674  \def\@glo@textaccess{#1}%
10675 }

```

**firstaccess** The replacement text corresponding to the first key:

```

10676 \define@key{glossentry}{firstaccess}{%
10677  \def\@glo@firstaccess{#1}%
10678 }

```

**pluralaccess** The replacement text corresponding to the plural key:

```

10679 \define@key{glossentry}{pluralaccess}{%
10680  \def\@glo@pluralaccess{#1}%
10681 }

```

**rstpluralaccess** The replacement text corresponding to the firstplural key:

```

10682 \define@key{glossentry}{firstpluralaccess}{%
10683  \def\@glo@firstpluralaccess{#1}%
10684 }

```

**symbolaccess** The replacement text corresponding to the symbol key:

```

10685 \define@key{glossentry}{symbolaccess}{%
10686  \def\@glo@symbolaccess{#1}%
10687 }

```

**bolpluralaccess** The replacement text corresponding to the symbolplural key:

```

10688 \define@key{glossentry}{symbolpluralaccess}{%
10689  \def\@glo@symbolpluralaccess{#1}%
10690 }

```

**scriptionaccess** The replacement text corresponding to the description key:

```
10691 \define@key{glossentry}{descriptionaccess}{%
10692   \def\@glo@descaccess{\#1}%
10693 }
```

**ionpluralaccess** The replacement text corresponding to the descriptionplural key:

```
10694 \define@key{glossentry}{descriptionpluralaccess}{%
10695   \def\@glo@descpluralaccess{\#1}%
10696 }
```

**shortaccess** The replacement text corresponding to the short key:

```
10697 \define@key{glossentry}{shortaccess}{%
10698   \def\@glo@shortaccess{\#1}%
10699 }
```

**ortpluralaccess** The replacement text corresponding to the shortplural key:

```
10700 \define@key{glossentry}{shortpluralaccess}{%
10701   \def\@glo@shortpluralaccess{\#1}%
10702 }
```

**longaccess** The replacement text corresponding to the long key:

```
10703 \define@key{glossentry}{longaccess}{%
10704   \def\@glo@longaccess{\#1}%
10705 }
```

**ongpluralaccess** The replacement text corresponding to the longplural key:

```
10706 \define@key{glossentry}{longpluralaccess}{%
10707   \def\@glo@longpluralaccess{\#1}%
10708 }
```

There are now also keys that correspond to the user keys:

**user1access** The replacement text corresponding to the user1 key:

```
10709 \define@key{glossentry}{user1access}{%
10710   \def\@glo@useriaccess{\#1}%
10711 }
```

**user2access** The replacement text corresponding to the user2 key:

```
10712 \define@key{glossentry}{user2access}{%
10713   \def\@glo@useriiaccess{\#1}%
10714 }
```

**user3access** The replacement text corresponding to the user3 key:

```
10715 \define@key{glossentry}{user3access}{%
10716   \def\@glo@useriiiaccess{\#1}%
10717 }
```

`user4access` The replacement text corresponding to the `user4` key:

```
10718 \define@key{glossentry}{user4access}{%
10719   \def\@glo@userivaccess{\#1}%
10720 }
```

`user5access` The replacement text corresponding to the `user5` key:

```
10721 \define@key{glossentry}{user5access}{%
10722   \def\@glo@uservaccess{\#1}%
10723 }
```

`user6access` The replacement text corresponding to the `user6` key:

```
10724 \define@key{glossentry}{user6access}{%
10725   \def\@glo@userviaccess{\#1}%
10726 }
```

For any custom keys, the replacement text would have to be explicitly put in the value, e.g.,  
`user1={\glsshortaccesupp{inches}{in}}`.

Append these new keys to `\@gls@keymap`:

```
10727 \appto{\@gls@keymap}{%
10728   {access}{access},%
10729   {textaccess}{textaccess},%
10730   {firstaccess}{firstaccess},%
10731   {pluralaccess}{pluralaccess},%
10732   {firstpluralaccess}{firstpluralaccess},%
10733   {symbolaccess}{symbolaccess},%
10734   {symbolpluralaccess}{symbolpluralaccess},%
10735   {descaccess}{descaccess},%
10736   {descpluralaccess}{descpluralaccess},%
10737   {shortaccess}{shortaccess},%
10738   {shortpluralaccess}{shortpluralaccess},%
10739   {longaccess}{longaccess},%
10740   {longpluralaccess}{longpluralaccess},%
10741   {user1access}{useriaccess},%
10742   {user2access}{useriiaccess},%
10743   {user3access}{useriiiaccess},%
10744   {user4access}{userivaccess},%
10745   {user5access}{uservaccess},%
10746   {user6access}{userviaccess}%
10747 }
```

`\@gls@noaccess` Indicates that no replacement text has been provided.

```
10748 \def\@gls@noaccess{\relax}
```

Previously, the access key was initialised to the value of the symbol key at the start for backwards compatibility. This causes a problem for situations where the replacement text is provided for symbol but not for name so this behaviour has been removed.

```
10749 \let\@gls@oldnewglossaryentryprehook\@newglossaryentryprehook
10750 \renewcommand*{\@newglossaryentryprehook}{%
```

```
10751  \@gls@oldnewglossaryentryprehook  
10752  \def\@glo@access{\relax}%
```

Initialise the other keys:

```
10753  \def\@glo@textaccess{\@glo@access}%
10754  \def\@glo@firstaccess{\@glo@access}%
10755  \def\@glo@pluralaccess{\@glo@textaccess}%
10756  \def\@glo@firstpluralaccess{\@glo@pluralaccess}%
10757  \def\@glo@symbolaccess{\relax}%
10758  \def\@glo@symbolpluralaccess{\@glo@symbolaccess}%
10759  \def\@glo@descaccess{\relax}%
10760  \def\@glo@descpluralaccess{\@glo@descaccess}%
10761  \def\@glo@shortaccess{\relax}%
10762  \def\@glo@shortpluralaccess{\@glo@shortaccess}%
10763  \def\@glo@longaccess{\relax}%
10764  \def\@glo@longpluralaccess{\@glo@longaccess}%
10765  \def\@glo@useriaccess{\relax}%
10766  \def\@glo@useriiaccess{\relax}%
10767  \def\@glo@useriiiaccess{\relax}%
10768  \def\@glo@userivaccess{\relax}%
10769  \def\@glo@uservaccess{\relax}%
10770  \def\@glo@userviaccess{\relax}%
10771 }
```

Add to the end hook:

```
10772 \let\@gls@oldnewglossaryentryposthook\@newglossaryentryposthook  
10773 \renewcommand*{\@newglossaryentryposthook}{%  
10774  \@gls@oldnewglossaryentryposthook}
```

Store the access information:

```
10775  \expandafter  
10776  \protected\xdef\csname glo@\@glo@label @access\endcsname{  
10777  \@glo@access}%
10778  \expandafter  
10779  \protected\xdef\csname glo@\@glo@label @textaccess\endcsname{  
10780  \@glo@textaccess}%
10781  \expandafter  
10782  \protected\xdef\csname glo@\@glo@label @firstaccess\endcsname{  
10783  \@glo@firstaccess}%
10784  \expandafter  
10785  \protected\xdef\csname glo@\@glo@label @pluralaccess\endcsname{  
10786  \@glo@pluralaccess}%
10787  \expandafter  
10788  \protected\xdef\csname glo@\@glo@label @firstpluralaccess\endcsname{  
10789  \@glo@firstpluralaccess}%
10790  \expandafter  
10791  \protected\xdef\csname glo@\@glo@label @symbolaccess\endcsname{  
10792  \@glo@symbolaccess}%
10793  \expandafter  
10794  \protected\xdef\csname glo@\@glo@label @symbolpluralaccess\endcsname{  
10795  \@glo@symbolpluralaccess}%
```

```

10796 \expandafter
10797   \protected@xdef\csname glo@\glo@label @descaccess\endcsname{%
10798     \@glo@descaccess}%
10799 \expandafter
10800   \protected@xdef\csname glo@\glo@label @descpluralaccess\endcsname{%
10801     \@glo@descpluralaccess}%
10802 \expandafter
10803   \protected@xdef\csname glo@\glo@label @shortaccess\endcsname{%
10804     \@glo@shortaccess}%
10805 \expandafter
10806   \protected@xdef\csname glo@\glo@label @shortpluralaccess\endcsname{%
10807     \@glo@shortpluralaccess}%
10808 \expandafter
10809   \protected@xdef\csname glo@\glo@label @longaccess\endcsname{%
10810     \@glo@longaccess}%
10811 \expandafter
10812   \protected@xdef\csname glo@\glo@label @longpluralaccess\endcsname{%
10813     \@glo@longpluralaccess}%
10814 \expandafter
10815   \protected@xdef\csname glo@\glo@label @useriaccess\endcsname{%
10816     \@glo@useriaccess}%
10817 \expandafter
10818   \protected@xdef\csname glo@\glo@label @useriiaccess\endcsname{%
10819     \@glo@useriiaccess}%
10820 \expandafter
10821   \protected@xdef\csname glo@\glo@label @userivaccess\endcsname{%
10822     \@glo@userivaccess}%
10823 \expandafter
10824   \protected@xdef\csname glo@\glo@label @uservaccess\endcsname{%
10825     \@glo@uservaccess}%
10826 \expandafter
10827   \protected@xdef\csname glo@\glo@label @userviaccess\endcsname{%
10828     \@glo@userviaccess}%
10829 \expandafter
10830   \protected@xdef\csname glo@\glo@label @userviiaccess\endcsname{%
10831     \@glo@userviiaccess}%
10832 }

```

## 5.2 Accessing Replacement Text

\glsentryaccess Get the value of the access key for the entry with the given label:

```

10833 \newcommand*{\glsentryaccess}[1]{%
10834   \@gls@entry@field{#1}{access}}%
10835 }

```

entrytextaccess Get the value of the textaccess key for the entry with the given label:

```

10836 \newcommand*{\glsentrytextaccess}[1]{%
10837   \@gls@entry@field{#1}{textaccess}}

```

```

10838 }

entryfirstaccess Get the value of the firstaccess key for the entry with the given label:
10839 \newcommand*{\glsentryfirstaccess}[1]{%
10840   \@gls@entry@field{#1}{firstaccess}%
10841 }

trypluralaccess Get the value of the pluralaccess key for the entry with the given label:
10842 \newcommand*{\glsentrypluralaccess}[1]{%
10843   \@gls@entry@field{#1}{pluralaccess}%
10844 }

rstpluralaccess Get the value of the firstpluralaccess key for the entry with the given label:
10845 \newcommand*{\glsentryfirstpluralaccess}[1]{%
10846   \@gls@entry@field{#1}{firstpluralaccess}%
10847 }

trysymbolaccess Get the value of the symbolaccess key for the entry with the given label:
10848 \newcommand*{\glsentrysymbolaccess}[1]{%
10849   \@gls@entry@field{#1}{symbolaccess}%
10850 }

bolpluralaccess Get the value of the symbolpluralaccess key for the entry with the given label:
10851 \newcommand*{\glsentrysymbolpluralaccess}[1]{%
10852   \@gls@entry@field{#1}{symbolpluralaccess}%
10853 }

entrydescaccess Get the value of the descriptionaccess key for the entry with the given label:
10854 \newcommand*{\glsentrydescaccess}[1]{%
10855   \@gls@entry@field{#1}{descaccess}%
10856 }

escpluralaccess Get the value of the descriptionpluralaccess key for the entry with the given label:
10857 \newcommand*{\glsentrydescpluralaccess}[1]{%
10858   \@gls@entry@field{#1}{descpluralaccess}%
10859 }

entryshortaccess Get the value of the shortaccess key for the entry with the given label:
10860 \newcommand*{\glsentryshortaccess}[1]{%
10861   \@gls@entry@field{#1}{shortaccess}%
10862 }

ortpluralaccess Get the value of the shortpluralaccess key for the entry with the given label:
10863 \newcommand*{\glsentryshortpluralaccess}[1]{%
10864   \@gls@entry@field{#1}{shortpluralaccess}%
10865 }

```

entrylongaccess Get the value of the longaccess key for the entry with the given label:

```
10866 \newcommand*{\glsentrylongaccess}[1]{%
10867   \@gls@entry@field{#1}{longaccess}%
10868 }
```

ongpluralaccess Get the value of the longpluralaccess key for the entry with the given label:

```
10869 \newcommand*{\glsentrylongpluralaccess}[1]{%
10870   \@gls@entry@field{#1}{longpluralaccess}%
10871 }
```

ntryuseriaccess Get the value of the user1access key for the entry with the given label:

```
10872 \newcommand*{\glsentryuseriaccess}[1]{%
10873   \@gls@entry@field{#1}{useriaccess}%
10874 }
```

tryuseriaccess Get the value of the user2access key for the entry with the given label:

```
10875 \newcommand*{\glsentryuseriaccess}[1]{%
10876   \@gls@entry@field{#1}{useriaccess}%
10877 }
```

ryuseriiiaccess Get the value of the user3access key for the entry with the given label:

```
10878 \newcommand*{\glsentryuseriiiaccess}[1]{%
10879   \@gls@entry@field{#1}{useriiiaccess}%
10880 }
```

tryuserivaccess Get the value of the user4access key for the entry with the given label:

```
10881 \newcommand*{\glsentryuserivaccess}[1]{%
10882   \@gls@entry@field{#1}{userivaccess}%
10883 }
```

ntryuservaccess Get the value of the user5access key for the entry with the given label:

```
10884 \newcommand*{\glsentryuservaccess}[1]{%
10885   \@gls@entry@field{#1}{uservaccess}%
10886 }
```

tryuserviaccess Get the value of the user6access key for the entry with the given label:

```
10887 \newcommand*{\glsentryuserviaccess}[1]{%
10888   \@gls@entry@field{#1}{userviaccess}%
10889 }
```

There are three types of replacement text:

**Alt** Description of some content that's non-textual (for example, an image). A word break is assumed after the content.

**ActualText** A character or sequence of characters that replaces textual content (for example, a dropped capital, a ligature or a symbol). No word break is assumed after the content.

**E** Expansion of an abbreviation to avoid ambiguity (for example, “St” could be short for “saint” or “street”).

Therefore, rather than having one command for all fields, it’s better to have a command dependent on the field type. For example, the short and shortpl keys would require E, the symbol key would require ActualText, and a field that contains an image would require Alt.

```
glsfieldaccsupp \glsfieldaccsupp{<replacement>}{<content>}{{<field>}}{<label>}
```

Test if there’s a command called `\gls<field>accsupp`. If there is then use that otherwise use `\glsaccsupp`. The first argument should be the internal field label (not the key). The final argument is the entry label. If glossaries-extra has been loaded, this first checks for `\glsxtr<category><field>accsupp` and `\glsxtr<category>accsupp`.

```
10890 \newcommand{\glsfieldaccsupp}[4]{%
10891   \ifdef\glscategory
10892   {%
10893     \ifcsdef{\glsxtr\glscategory}{#4}{#3accsupp}%
10894     {\csname glsxtr\glscategory{#4}{#3accsupp}\endcsname{#1}{#2}}%
10895     {%
10896       \ifcsdef{\glsxtr\glscategory}{#4}{accsupp}%
10897         {\csname glsxtr\glscategory{#4}{accsupp}\endcsname{#1}{#2}}%
10898         {%
10899           \ifcsdef{gls#3accsupp}{%
10900             {\csname gls#3accsupp\endcsname{#1}{#2}}%
10901             {\glsaccsupp{#1}{#2}}%
10902           }%
10903         }%
10904       }%
10905     {%
10906       \ifcsdef{gls#3accsupp}{%
10907         {\csname gls#3accsupp\endcsname{#1}{#2}}%
10908         {\glsaccsupp{#1}{#2}}%
10909       }%
10910     }%
}
```

```
glsfieldaccsupp \xglsfieldaccsupp{<replacement>}{<content>}{{<field>}}{<label>}
```

As `\glsfieldaccsupp` but fully expand replacement text.

```
10911 \newcommand{\xglsfieldaccsupp}[1]{%
10912   \protected@edef{\gls@replacementtext}{#1}%
10913   \expandafter\glsfieldaccsupp\expandafter{\gls@replacementtext}%
10914 }
```

```
glsshortaccsupp \glsshortaccsupp{<replacement text>}{<text>}
```

```
10915 \newcommand*{\glsshortaccsupp}[2]{\glsaccessibility{E}{#1}{#2}}
```

```
\glsshortplaccsupp{\<replacement text>}{\<text>}
```

```
10916 \newcommand*\glsshortplaccsupp{\glsshortplaccsupp}{\glsshortplaccsupp}
```

```
\glsaccsupp{\<replacement text>}{\<text>}
```

```
10917 \newcommand*\glsaccsupp[2]{\glsaccessibility{ActualText}{#1}{#2}}
```

\xglsaccsupp Fully expands replacement text before calling \glsaccsupp

```
10918 \newcommand*\xglsaccsupp[2]{%
10919   \protected@edef\gls@replacementtext{#1}%
10920   \expandafter\glsaccsupp\expandafter{\gls@replacementtext}{#2}%
10921 }
```

@access@display Deprecated. Use \@gls@fieldaccess@display instead.

```
10922 \newcommand*\@gls@access@display[2]{%
10923   \protected@edef\glo@access{#2}%
10924   \ifx\glo@access\gls@noaccess
10925     #1%
10926   \else
10927     \xglsaccsupp{\glo@access}{#1}%
10928   \fi
10929 }
```

```
daccess@display \@gls@fieldaccess@display{\<label>}{\<field>}{\<content>}{\<replacement>}
```

```
10930 \newcommand*\@gls@fieldaccess@display[4]{%
10931   \protected@edef\glo@access{#4}%
10932   \ifdefequal\glo@access\gls@noaccess
10933   {#3}%
10934   {\expandafter\glsfieldaccsupp\expandafter{\glo@access}{#3}{#2}{#1}}%
10935 }
```

meaccessdisplay Displays the first argument with the accessibility text for the entry with the label given by the second argument (if set).

```
10936 \newrobustcmd*\glsnameaccessdisplay[2]{%
10937   \ifcsundef{glo@\glsdetoklabel{#2}@access}%
10938   {#1}%
10939   {\@gls@fieldaccess@display{#2}{name}{#1}{\glsentryaccess{#2}}}%
10940 }
```

xtaccessdisplay As above but for the textaccess replacement text.

```
10941 \newrobustcmd*\glstextaccessdisplay[2]{%
10942   \ifcsundef{glo@\glsdetoklabel{#2}@textaccess}%
10943   {#1}%
10944 }
```

```
10944 {\@gls@fieldaccess@display{#2}{text}{#1}{\glsentrytextaccess{#2}}}}%  
10945 }
```

alaccessdisplay As above but for the pluralaccess replacement text.

```
10946 \newrobustcmd*\{\glspluralaccessdisplay}[2]{%  
10947 \ifcsundef{glo@\glsdetoklabel{#2}@pluralaccess}{%  
10948 {#1}}%  
10949 {\@gls@fieldaccess@display{#2}{plural}{#1}{\glsentrypluralaccess{#2}}}}%  
10950 }
```

staccessdisplay As above but for the firstaccess replacement text.

```
10951 \newrobustcmd*\{\glsfirstaccessdisplay}[2]{%  
10952 \ifcsundef{glo@\glsdetoklabel{#2}@firstaccess}{%  
10953 {#1}}%  
10954 {\@gls@fieldaccess@display{#2}{first}{#1}{\glsentryfirstaccess{#2}}}}%  
10955 }
```

alaccessdisplay As above but for the firstpluralaccess replacement text.

```
10956 \newrobustcmd*\{\glsfirstpluralaccessdisplay}[2]{%  
10957 \ifcsundef{glo@\glsdetoklabel{#2}@firstpluralaccess}{%  
10958 {#1}}%  
10959 {\@gls@fieldaccess@display{#2}{firstpl}{#1}{\glsentryfirstpluralaccess{#2}}}}%  
10960 }
```

olaccessdisplay As above but for the symbolaccess replacement text.

```
10961 \newrobustcmd*\{\glssymbolaccessdisplay}[2]{%  
10962 \ifcsundef{glo@\glsdetoklabel{#2}@symbolaccess}{%  
10963 {#1}}%  
10964 {\@gls@fieldaccess@display{#2}{symbol}{#1}{\glsentrysymbolaccess{#2}}}}%  
10965 }
```

alaccessdisplay As above but for the symbolpluralaccess replacement text.

```
10966 \newrobustcmd*\{\glssymbolpluralaccessdisplay}[2]{%  
10967 \ifcsundef{glo@\glsdetoklabel{#2}@symbolpluralaccess}{%  
10968 {#1}}%  
10969 {\@gls@fieldaccess@display{#2}{symbolplural}{#1}{\glsentrysymbolpluralaccess{#2}}}}%  
10970 }
```

onaccessdisplay As above but for the descriptionaccess replacement text.

```
10971 \newrobustcmd*\{\glsdescriptionaccessdisplay}[2]{%  
10972 \ifcsundef{glo@\glsdetoklabel{#2}@descaccess}{%  
10973 {#1}}%  
10974 {\@gls@fieldaccess@display{#2}{desc}{#1}{\glsentrydescaccess{#2}}}}%  
10975 }
```

alaccessdisplay As above but for the descriptionpluralaccess replacement text.

```
10976 \newrobustcmd*\{\glsdescriptionpluralaccessdisplay}[2]{%  
10977 \ifcsundef{glo@\glsdetoklabel{#2}@descpluralaccess}{%
```

```
10978  {#1}%
10979  {\@gls@fieldaccess@display{#2}{descplural}{#1}{\glsentrydescpluralaccess{#2}}}}%
10980 }
```

**rtaccessdisplay** As above but for the shortaccess replacement text.

```
10981 \newrobustcmd*{\glsshortaccessdisplay}[2]{%
10982  \ifcsundef{glo@\glsdetoklabel{#2}@shortaccess}%
10983  {#1}%
10984  {\@gls@fieldaccess@display{#2}{short}{#1}{\glsentryshortaccess{#2}}}}%
10985 }
```

**alaccessdisplay** As above but for the shortpluralaccess replacement text.

```
10986 \newrobustcmd*{\glsshortpluralaccessdisplay}[2]{%
10987  \ifcsundef{glo@\glsdetoklabel{#2}@shortpluralaccess}%
10988  {#1}%
10989  {\@gls@fieldaccess@display{#2}{shortpl}{#1}{\glsentryshortpluralaccess{#2}}}}%
10990 }
```

**ngaccessdisplay** As above but for the longaccess replacement text.

```
10991 \newrobustcmd*{\glslongaccessdisplay}[2]{%
10992  \ifcsundef{glo@\glsdetoklabel{#2}@longaccess}%
10993  {#1}%
10994  {\@gls@fieldaccess@display{#2}{long}{#1}{\glsentrylongaccess{#2}}}}%
10995 }
```

**alaccessdisplay** As above but for the longpluralaccess replacement text.

```
10996 \newrobustcmd*{\glslongpluralaccessdisplay}[2]{%
10997  \ifcsundef{glo@\glsdetoklabel{#2}@longpluralaccess}%
10998  {#1}%
10999  {\@gls@fieldaccess@display{#2}{longpl}{#1}{\glsentrylongpluralaccess{#2}}}}%
11000 }
```

**riaccessdisplay** As above but for the user1access replacement text.

```
11001 \newrobustcmd*{\glsuseriaccessdisplay}[2]{%
11002  \ifcsundef{glo@\glsdetoklabel{#2}@useriaccess}%
11003  {#1}%
11004  {\@gls@fieldaccess@display{#2}{useri}{#1}{\glsentryuseriaccess{#2}}}}%
11005 }
```

**iiaccessdisplay** As above but for the user2access replacement text.

```
11006 \newrobustcmd*{\glsuseriiaccessdisplay}[2]{%
11007  \ifcsundef{glo@\glsdetoklabel{#2}@useriiaccess}%
11008  {#1}%
11009  {\@gls@fieldaccess@display{#2}{userii}{#1}{\glsentryuseriiaccess{#2}}}}%
11010 }
```

**iiaccessdisplay** As above but for the user3access replacement text.

```
11011 \newrobustcmd*{\glsuseriiiaccessdisplay}[2]{%
```

```
11012 \ifcsundef{glo@\glsdetoklabel{#2}@useriiiaccess}{%
11013 {#1}%
11014 {\@gls@fieldaccess@display{#2}{useriii}{#1}{\glsentryuseriiiaccess{#2}}}}%
11015 }
```

ivaccessdisplay As above but for the user4access replacement text.

```
11016 \newrobustcmd*\{\glsuserivaccessdisplay}[2]{%
11017 \ifcsundef{glo@\glsdetoklabel{#2}@userivaccess}{%
11018 {#1}%
11019 {\@gls@fieldaccess@display{#2}{useriv}{#1}{\glsentryuserivaccess{#2}}}}%
11020 }
```

rvaccessdisplay As above but for the user5access replacement text.

```
11021 \newrobustcmd*\{\glsuservaccessdisplay}[2]{%
11022 \ifcsundef{glo@\glsdetoklabel{#2}@uservaccess}{%
11023 {#1}%
11024 {\@gls@fieldaccess@display{#2}{userv}{#1}{\glsentryuservaccess{#2}}}}%
11025 }
```

viaccessdisplay As above but for the user6access replacement text.

```
11026 \newrobustcmd*\{\glsuserviaccessdisplay}[2]{%
11027 \ifcsundef{glo@\glsdetoklabel{#2}@userviaccess}{%
11028 {#1}%
11029 {\@gls@fieldaccess@display{#2}{uservi}{#1}{\glsentryuserviaccess{#2}}}}%
11030 }
```

lsaccessdisplay Gets the replacement text corresponding to the named key given by the first argument and calls the appropriate command defined above.

```
11031 \newrobustcmd*\{\glsaccessdisplay}[3]{%
11032 \ifcsundef{gls#1accessdisplay}{%
11033 {%
11034 \PackageError{glossaries-accsupp}{No accessibility support
11035 for key '#1'}{}%
11036 }%
11037 {%
11038 \csname gls#1accessdisplay\endcsname{#2}{#3}%
11039 }%
11040 }
```

efault@entryfmt Redefine the default entry format to use accessibility information

```
11041 \renewcommand*\{@@gls@default@entryfmt}[2]{%
11042 \ifdefempty\glscustomtext{%
11043 {%
11044 \glsifplural{%
11045 {%
```

Plural form

```
11046 \glscapscase{%
11047 {%
```

### Don't adjust case

```
11048      \ifglsused\glslabel  
11049      {%
```

### Subsequent use

```
11050      #2{\glspluralaccessdisplay  
11051          {\glsentryplural{\glslabel}}{\glslabel}}%  
11052          {\glsdescriptionpluralaccessdisplay  
11053              {\glsentrydescplural{\glslabel}}{\glslabel}}%  
11054              {\glssymbolpluralaccessdisplay  
11055                  {\glsentrysymbolplural{\glslabel}}{\glslabel}}%  
11056                  {\glsinsert}}%  
11057      }%  
11058      {%
```

### First use

```
11059      #1{\glsfirstpluralaccessdisplay  
11060          {\glsentryfirstplural{\glslabel}}{\glslabel}}%  
11061          {\glsdescriptionpluralaccessdisplay  
11062              {\glsentrydescplural{\glslabel}}{\glslabel}}%  
11063              {\glssymbolpluralaccessdisplay  
11064                  {\glsentrysymbolplural{\glslabel}}{\glslabel}}%  
11065                  {\glsinsert}}%  
11066      }%  
11067      }%  
11068      {%
```

### Make first letter upper case

```
11069      \ifglsused\glslabel  
11070      {%
```

### Subsequent use.

```
11071      #2{\glspluralaccessdisplay  
11072          {\Glsentryplural{\glslabel}}{\glslabel}}%  
11073          {\glsdescriptionpluralaccessdisplay  
11074              {\glsentrydescplural{\glslabel}}{\glslabel}}%  
11075              {\glssymbolpluralaccessdisplay  
11076                  {\glsentrysymbolplural{\glslabel}}{\glslabel}}%  
11077                  {\glsinsert}}%  
11078      }%  
11079      {%
```

### First use

```
11080      #1{\glsfirstpluralaccessdisplay  
11081          {\Glsentryfirstplural{\glslabel}}{\glslabel}}%  
11082          {\glsdescriptionpluralaccessdisplay  
11083              {\glsentrydescplural{\glslabel}}{\glslabel}}%  
11084              {\glssymbolpluralaccessdisplay  
11085                  {\glsentrysymbolplural{\glslabel}}{\glslabel}}%  
11086                  {\glsinsert}}%  
11087      }%
```

```

11088      }%
11089      {%
Make all upper case
11090      \ifglsused\glslabel
11091      {%
Subsequent use
11092      \MakeUppercase{%
11093          #2{\glspluralaccessdisplay
11094              {\glsentryplural{\glslabel}}{\glslabel}}%
11095          {\glsdescriptionpluralaccessdisplay
11096              {\glsentrydescplural{\glslabel}}{\glslabel}}%
11097          {\glssymbolpluralaccessdisplay
11098              {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
11099          {\glsinsert}}%
11100      }%
11101      {%

```

#### First use

```

11102      \MakeUppercase{%
11103          #1{\glsfirstpluralaccessdisplay
11104              {\glsentryfirstplural{\glslabel}}{\glslabel}}%
11105          {\glsdescriptionpluralaccessdisplay
11106              {\glsentrydescplural{\glslabel}}{\glslabel}}%
11107          {\glssymbolpluralaccessdisplay
11108              {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
11109          {\glsinsert}}%
11110      }%
11111      }%
11112      }%
11113      {%

```

#### Singular form

```

11114      \glscapscase
11115      {%

```

#### Don't adjust case

```

11116      \ifglsused\glslabel
11117      {%

```

#### Subsequent use

```

11118      #2{\glstextaccessdisplay
11119          {\glsentrytext{\glslabel}}{\glslabel}}%
11120          {\glsdescriptionaccessdisplay
11121              {\glsentrydesc{\glslabel}}{\glslabel}}%
11122              {\glssymbolaccessdisplay
11123                  {\glsentrysymbol{\glslabel}}{\glslabel}}%
11124                  {\glsinsert}}%
11125      }%
11126      {%

```

First use

```
11127      #1{\glsfirstaccessdisplay
11128          {\glsentryfirst{\glslabel}}{\glslabel}}%
11129          {\glsdescriptionaccessdisplay
11130              {\glsentrydesc{\glslabel}}{\glslabel}}%
11131          {\glssymbolaccessdisplay
11132              {\glsentrysymbol{\glslabel}}{\glslabel}}%
11133          {\glsinsert}%
11134      }%
11135  }%
11136  {%
```

Make first letter upper case

```
11137      \ifglsused\glslabel
11138  {%
```

Subsequent use

```
11139      #2{\glstextaccessdisplay
11140          {\Glsentrytext{\glslabel}}{\glslabel}}%
11141          {\glsdescriptionaccessdisplay
11142              {\glsentrydesc{\glslabel}}{\glslabel}}%
11143          {\glssymbolaccessdisplay
11144              {\glsentrysymbol{\glslabel}}{\glslabel}}%
11145          {\glsinsert}%
11146      }%
11147  {%
```

First use

```
11148      #1{\glsfirstaccessdisplay
11149          {\Glsentryfirst{\glslabel}}{\glslabel}}%
11150          {\glsdescriptionaccessdisplay
11151              {\glsentrydesc{\glslabel}}{\glslabel}}%
11152          {\glssymbolaccessdisplay
11153              {\glsentrysymbol{\glslabel}}{\glslabel}}%
11154          {\glsinsert}%
11155      }%
11156  }%
11157  {%
```

Make all upper case

```
11158      \ifglsused\glslabel
11159  {%
```

Subsequent use

```
11160      \MakeUppercase{%
11161          #2{\glstextaccessdisplay
11162              {\glsentrytext{\glslabel}}{\glslabel}}%
11163              {\glsdescriptionaccessdisplay
11164                  {\glsentrydesc{\glslabel}}{\glslabel}}%
11165                  {\glssymbolaccessdisplay
11166                      {\glsentrysymbol{\glslabel}}{\glslabel}}%
```

```

11167          {\glsinsert}}}%
11168      }%
11169  {%
First use
11170      \MakeUppercase{%
11171          #1{\glsfirstaccessdisplay
11172              {\glsentryfirst{\glslabel}}{\glslabel}}%
11173          {\glsdescriptionaccessdisplay
11174              {\glsentrydesc{\glslabel}}{\glslabel}}%
11175          {\glssymbolaccessdisplay
11176              {\glsentrysymbol{\glslabel}}{\glslabel}}%
11177          {\glsinsert}}}%
11178      }%
11179  }%
11180  }%
11181 }%
11182 {%

```

Custom text provided in \glsdisp

```

11183 \ifglsused{\glslabel}{%
11184 {%

```

Subsequent use

```

11185 #2{\glscustomtext}{%
11186     {\glsdescriptionaccessdisplay
11187         {\glsentrydesc{\glslabel}}{\glslabel}}%
11188         {\glssymbolaccessdisplay
11189             {\glsentrysymbol{\glslabel}}{\glslabel}}%
11190             {\glsinsert}}}%
11191     }%
11192 {%

```

First use

```

11193 #1{\glscustomtext}{%
11194     {\glsdescriptionaccessdisplay
11195         {\glsentrydesc{\glslabel}}{\glslabel}}%
11196         {\glssymbolaccessdisplay
11197             {\glsentrysymbol{\glslabel}}{\glslabel}}%
11198             {\glsinsert}}}%
11199     }%
11200 }%
11201 }%

```

\glsgenentryfmt Redefine to use accessibility information.

```

11202 \renewcommand*{\glsgenentryfmt}{%
11203 \ifdefempty\glscustomtext
11204 {%
11205 \glsifplural
11206 {%

```

Plural form

```
11207      \glscaps{case}
11208      {%
```

Don't adjust case

```
11209      \ifglsused{\glslabel}
11210      {%
```

Subsequent use

```
11211      \glspluralaccess{display}
11212          {\glsentryplural{\glslabel}}{\glslabel}%
11213          \glsinsert
11214      }%
11215      {%
```

First use

```
11216      \glsfirstpluralaccess{display}
11217          {\glsentryfirstplural{\glslabel}}{\glslabel}%
11218          \glsinsert
11219      }%
11220      {%
11221      {%
```

Make first letter upper case

```
11222      \ifglsused{\glslabel}
11223      {%
```

Subsequent use.

```
11224      \glspluralaccess{display}
11225          {\Glsentryplural{\glslabel}}{\glslabel}%
11226          \glsinsert
11227      }%
11228      {%
```

First use

```
11229      \glsfirstpluralaccess{display}
11230          {\Glsentryfirstplural{\glslabel}}{\glslabel}%
11231          \glsinsert
11232      }%
11233      {%
11234      {%
```

Make all upper case

```
11235      \ifglsused{\glslabel}
11236      {%
```

Subsequent use

```
11237      \glspluralaccess{display}
11238          {\mfirstuc{MakeUppercase}{\glsentryplural{\glslabel}}}%
11239          {\glslabel}%
11240          \mfirstuc{MakeUppercase}{\glsinsert}%
11241      }%
11242      {%
```

First use

```
11243      \glsfirstpluralacessdisplay
11244          {\mfirstucMakeUppercase{\glsentryfirstplural{\glslabel}}}{%
11245              {\glslabel}%
11246          \mfirstucMakeUppercase{\glsinsert}%
11247      }%
11248  }%
11249 }%
11250 {%
```

Singular form

```
11251      \glscapscase
11252  {%
```

Don't adjust case

```
11253      \ifglsused\glslabel
11254  {%
```

Subsequent use

```
11255      \glstextaccessdisplay{\glsentrytext{\glslabel}}{\glslabel}%
11256          \glsinsert
11257      }%
11258  {%
```

First use

```
11259      \glsfirstaccessdisplay{\glsentryfirst{\glslabel}}{\glslabel}%
11260          \glsinsert
11261      }%
11262  }%
11263  {%
```

Make first letter upper case

```
11264      \ifglsused\glslabel
11265  {%
```

Subsequent use

```
11266      \glstextaccessdisplay{\Glsentrytext{\glslabel}}{\glslabel}%
11267          \glsinsert
11268      }%
11269  {%
```

First use

```
11270      \glsfirstaccessdisplay{\Glsentryfirst{\glslabel}}{\glslabel}%
11271          \glsinsert
11272      }%
11273  }%
11274  {%
```

Make all upper case

```
11275      \ifglsused\glslabel
11276  {%
```

### Subsequent use

```
11277      \glstextaccessdisplay
11278          {\mfirstucMakeUppercase{\glsentrytext{\glslabel}}}{\glslabel}%
11279          \mfirstucMakeUppercase{\glsinsert}%
11280      }%
11281  {%
```

### First use

```
11282      \glsfirstaccessdisplay
11283          {\mfirstucMakeUppercase{\glsentryfirst{\glslabel}}}{\glslabel}%
11284          \mfirstucMakeUppercase{\glsinsert}%
11285      }%
11286      }%
11287      }%
11288  }%
11289  {%
```

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.

```
11290      \glscustomtext\glsinsert
11291  }%
11292 }
```

`\glsgenacfmt` Redefine to include accessibility information.

```
11293 \renewcommand*\glsgenacfmt}{%
11294     \ifdefempty\glscustomtext
11295     {%
11296         \ifglsused\glslabel
11297     {%
```

### Subsequent use:

```
11298      \glsifplural
11299  {%
```

### Subsequent plural form:

```
11300      \glscapscase
11301  {%
```

### Subsequent plural form, don't adjust case:

```
11302      \acronymfont
11303          {\glsshortpluralaccessdisplay
11304              {\glsentryshortpl{\glslabel}}{\glslabel}%
11305              \glsinsert
11306          }%
11307  {%
```

### Subsequent plural form, make first letter upper case:

```
11308      \acronymfont
11309          {\glsshortpluralaccessdisplay
11310              {\Glsentryshortpl{\glslabel}}{\glslabel}%
11311              \glsinsert
```

```
11312      }%
11313      {%
```

Subsequent plural form, all caps:

```
11314          \mfirstucMakeUppercase
11315          {\acronymfont
11316          {\glsshortpluralaccessdisplay
11317          {\glsentryshortpl{\glslabel}{\glslabel}}%
11318          \glsinsert}%
11319          }%
11320      }%
11321      {%
```

Subsequent singular form

```
11322      \glscapscase
11323      {%
```

Subsequent singular form, don't adjust case:

```
11324          \acronymfont
11325          {\glsshortaccessdisplay{\glsentryshort{\glslabel}{\glslabel}}%
11326          \glsinsert
11327          }%
11328          {%
```

Subsequent singular form, make first letter upper case:

```
11329          \acronymfont
11330          {\glsshortaccessdisplay{\Glsentryshort{\glslabel}{\glslabel}}%
11331          \glsinsert
11332          }%
11333          {%
```

Subsequent singular form, all caps:

```
11334          \mfirstucMakeUppercase
11335          {\acronymfont{%
11336          {\glsshortaccessdisplay{\glsentryshort{\glslabel}{\glslabel}}%
11337          \glsinsert}%
11338          }%
11339          }%
11340      }%
11341      {%
```

First use:

```
11342      \glsifplural
11343      {%
```

First use plural form:

```
11344      \glscapscase
11345      {%
```

First use plural form, don't adjust case:

```
11346          \genplacrfullformat{\glslabel}{\glsinsert}%
11347          }%
11348          {%
```

First use plural form, make first letter upper case:

```
11349      \Genplacrfullformat{\glslabel}{\glsinsert}%
11350      }%
11351      {%
```

First use plural form, all caps:

```
11352      \mfirstucMakeUppercase
11353      {\genplacrfullformat{\glslabel}{\glsinsert}}%
11354      }%
11355      }%
11356      {%
```

First use singular form

```
11357      \glscapscase
11358      {%
```

First use singular form, don't adjust case:

```
11359      \genacrfullformat{\glslabel}{\glsinsert}%
11360      }%
11361      {%
```

First use singular form, make first letter upper case:

```
11362      \Genacrfullformat{\glslabel}{\glsinsert}%
11363      }%
11364      {%
```

First use singular form, all caps:

```
11365      \mfirstucMakeUppercase
11366      {\genacrfullformat{\glslabel}{\glsinsert}}%
11367      }%
11368      }%
11369      }%
11370      }%
11371      {%
```

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.

```
11372      \glscustomtext
11373      }%
11374 }
```

`enacrfullformat` Redefine to include accessibility information.

```
11375 \renewcommand*{\genacrfullformat}[2]{%
11376   \glslongaccessdisplay{\glsentrylong{#1}}{#1}#2\space
11377   (\glsshortaccessdisplay{\protect\firstracronymfont{\glsentryshort{#1}}}{#1})%
11378 }
```

`enacrfullformat` Redefine to include accessibility information.

```
11379 \renewcommand*{\Genacrfullformat}[2]{%
11380   \glslongaccessdisplay{\Glsentrylong{#1}}{#1}#2\space
11381   (\glsshortaccessdisplay{\protect\firstracronymfont{\Glsentryshort{#1}}}{#1})%
11382 }
```

```

placrfullformat Redefine to include accessibility information.
11383 \renewcommand*{\genplacrfullformat}[2]{%
11384   \glslongpluralaccessdisplay{\glsentrylongpl{#1}}{#1}#2\space
11385   (\glsshortpluralaccessdisplay
11386     {\protect\firstacronymfont{\glsentryshortpl{#1}}}{#1})%
11387 }

placrfullformat Redefine to include accessibility information.
11388 \renewcommand*{\Genplacrfullformat}[2]{%
11389   \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}#2\space
11390   (\glsshortpluralaccessdisplay
11391     {\protect\firstacronymfont{\glsentryshortpl{#1}}}{#1})%
11392 }

\@acrshort
11393 \def\@acrshort#1#2[#3]{%
11394   \glsdoifexists{#2}%
11395   {%
11396     \let\do@gls@link@checkfirsthyper\relax
11397     \let\glsifplural@\secondoftwo
11398     \let\glscapscase@\firstofthree
11399     \let\glsinsert@\empty
11400     \def\glscustomtext{%
11401       \acronymfont{\glsshortaccessdisplay{\glsentryshort{#2}}}{#2}}#3%
11402   }%
11403   Call \gls@link
11404   \gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
11405 \glspostlinkhook
11406 }

\@Acrshort
11407 \def\@Acrshort#1#2[#3]{%
11408   \glsdoifexists{#2}%
11409   {%
11410     \let\do@gls@link@checkfirsthyper\relax
11411     \let\glsifplural@\secondoftwo
11412     \let\glscapscase@\secondofthree
11413     \let\glsinsert@\empty
11414     \def\glscustomtext{%
11415       \acronymfont{\glsshortaccessdisplay{\Glsentryshort{#2}}}{#2}}#3%
11416   }%
11417   Call \gls@link
11418   \gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
11419 }

```

```

11419 \glspostlinkhook
11420 }

\@ACRshort
11421 \def\@ACRshort#1#2[#3]{%
11422   \glsdoifexists{#2}{%
11423     {%
11424       \let\do@gls@link@checkfirsthyper\relax
11425       \let\glsifplural@\secondoftwo
11426       \let\glscapscase@\thirdofthree
11427       \let\glsinsert@\empty
11428       \def\glscustomtext{%
11429         \acronymfont{\glsshortaccessdisplay
11430           {\MakeUppercase{\glsentryshort{#2}}}{#2}}#3%
11431       }%
11432     Call \gls@link
11433     \gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
11434   }%
11435   \glspostlinkhook
11436 }
11437 }

\@acrlong
11438 \def\@acrlong#1#2[#3]{%
11439   \glsdoifexists{#2}{%
11440     {%
11441       \let\glsifplural@\secondoftwo
11442       \let\glscapscase@\firstofthree
11443       \let\glsinsert@\empty
11444       \def\glscustomtext{%
11445         \acronymfont{\glslongaccessdisplay{\glsentrylong{#2}}}{#2}}#3%
11446       }%
11447     Call \gls@link
11448     \gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
11449   }%
11450   \glspostlinkhook
11451 }
11452 }
11453   \let\do@gls@link@checkfirsthyper\relax

```

```

11454     \let\glsifplural\@secondoftwo
11455     \let\glscapscase\@firstofthree
11456     \let\glsinsert\@empty
11457     \def\glscustomtext{%
11458         \acronymfont{\glslongaccessdisplay{\Glsentrylong{#2}}{#2}}{#3}%
11459     }%
11460     Call \gls@link
11461     \gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
11462     \glspostlinkhook
11463 }

\@ACRlong
11464 \def\@ACRlong#1#2[#3]{%
11465   \glsdoifexists{#2}{%
11466     {%
11467       \let\do@gls@link@checkfirthyper\relax
11468       \let\glsifplural\@secondoftwo
11469       \let\glscapscase\@firstofthree
11470       \let\glsinsert\@empty
11471       \def\glscustomtext{%
11472         \acronymfont{\glslongaccessdisplay{%
11473             \MakeUppercase{\glsentrylong{#2}}}}{#2}}{#3}%
11474     }%
11475     Call \gls@link
11476     \gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
11477     \glspostlinkhook
11478 }

\@glstext@
11479 \def\@glstext@#1#2[#3]{%
11480   \gls@field@link{#1}{#2}{\glstextaccessdisplay{\glsentrytext{#2}}{#2}}{#3}%
11481 }

\@GLStext@
11482 \def\@GLStext@#1#2[#3]{%
11483   \gls@field@link{#1}{#2}{\glstextaccessdisplay{\Glsentrytext{#2}}{#2}}{#3}%
11484 }

\@GLStext@
11485 \def\@GLStext@#1#2[#3]{%
11486   \gls@field@link{#1}{#2}%
11487   {\glstextaccessdisplay{\mfirstucMakeUppercase{\glsentrytext{#2}}}}{#2}%
11488   \mfirstucMakeUppercase{#3}%
11489 }

```

```

\@glsfirst@

11490 \def\@glsfirst@#1#2[#3]{%
11491   \@gls@field@link{#1}{#2}{\glsfirstaccessdisplay{\glsentryfirst{#2}}{#2}#3}%
11492 }

\@Glsfirst@

11493 \def\@Glsfirst@#1#2[#3]{%
11494   \@gls@field@link{#1}{#2}{\glsfirstaccessdisplay{\Glsentryfirst{#2}}{#2}#3}%
11495 }

\@GLSfirst@

11496 \def\@GLSfirst@#1#2[#3]{%
11497   \@gls@field@link{#1}{#2}{%
11498     {\glsfirstaccessdisplay{\mfirstucMakeUppercase{\glsentryfirst{#2}}}{#2}}%
11499     \mfirstucMakeUppercase{#3}}%
11500 }

\@glsplural@

11501 \def\@glsplural@#1#2[#3]{%
11502   \@gls@field@link{#1}{#2}{\glspluralaccessdisplay{\glsentryplural{#2}}{#2}#3}%
11503 }

\@Glsplural@

11504 \def\@Glsplural@#1#2[#3]{%
11505   \@gls@field@link{#1}{#2}{\glspluralaccessdisplay{\Glsentryplural{#2}}{#2}#3}%
11506 }

\@GLSplural@

11507 \def\@GLSplural@#1#2[#3]{%
11508   \@gls@field@link{#1}{#2}{%
11509     {\glspluralaccessdisplay{\mfirstucMakeUppercase{\glsentryplural{#2}}}{#2}}%
11510     \mfirstucMakeUppercase{#3}}%
11511 }

glsfirstplural@

11512 \def\@glsfirstplural@#1#2[#3]{%
11513   \@gls@field@link{#1}{#2}{\glsfirstpluralaccessdisplay{\glsentryfirstplural{#2}}{#2}#3}%
11514 }

Glsfirstplural@

11515 \def\@glsfirstplural@#1#2[#3]{%
11516   \@gls@field@link{#1}{#2}{\glsfirstpluralaccessdisplay{\Glsentryfirstplural{#2}}{#2}#3}%
11517 }

GLSfirstplural@

11518 \def\@GLSfirstplural@#1#2[#3]{%
11519   \@gls@field@link{#1}{#2}{%
11520     {\glsfirstpluralaccessdisplay{\mfirstucMakeUppercase{\glsentryfirstplural{#2}}}{#2}}%

```

```

11521     \mfirstucMakeUppercase{#3}%
11522 }

\@glsname@

11523 \def\@glsname@#1#2[#3]{%
11524   \@gls@field@link{#1}{#2}{\glsnameaccessdisplay{\glsentryname{#2}}{#2}#3}%
11525 }

\@Glsname@

11526 \def\@Glsname@#1#2[#3]{%
11527   \@gls@field@link{#1}{#2}{\glsnameaccessdisplay{\Glsentryname{#2}}{#2}#3}%
11528 }

\@GLSname@

11529 \def\@GLSname@#1#2[#3]{%
11530   \@gls@field@link{#1}{#2}%
11531   {\glsnameaccessdisplay{\mfirstucMakeUppercase{\glsentryname{#2}}}{#2}%
11532   \mfirstucMakeUppercase{#3}}%
11533 }

\@glsdesc@

11534 \def\@glsdesc@#1#2[#3]{%
11535   \@gls@field@link{#1}{#2}{\glsdescriptionaccessdisplay{\glsentrydesc{#2}}{#2}#3}%
11536 }

\@Glsdesc@

11537 \def\@Glsdesc@#1#2[#3]{%
11538   \@gls@field@link{#1}{#2}{\glsdescriptionaccessdisplay{\Glsentrydesc{#2}}{#2}#3}%
11539 }

\@GLSdesc@

11540 \def\@GLSdesc@#1#2[#3]{%
11541   \@gls@field@link{#1}{#2}%
11542   {\glsdescriptionaccessdisplay{\mfirstucMakeUppercase{\glsentrydesc{#2}}}{#2}%
11543   \mfirstucMakeUppercase{#3}}%
11544 }

@glsdescplural@

11545 \def\@glsdescplural@#1#2[#3]{%
11546   \@gls@field@link{#1}{#2}{\glsdescriptionpluralaccessdisplay{\glsentrydescplural{#2}}{#2}#3}%
11547 }

@Glsdescplural@

11548 \def\@Glsdescplural@#1#2[#3]{%
11549   \@gls@field@link{#1}{#2}{\glsdescriptionpluralaccessdisplay{\Glsentrydescplural{#2}}{#2}#3}%
11550 }

```

```

@GLSdescplural@

11551 \def\@GLSdescplural@#1#2[#3]{%
11552   \gls@field@link{#1}{#2}%
11553   {\glsdescriptionpluralaccessdisplay{\mfirstucMakeUppercase{\glsentrydescplural{#2}}}{#2}%
11554   \mfirstucMakeUppercase{#3}}%
11555 }

\@glssymbol@

11556 \def\@glssymbol@#1#2[#3]{%
11557   \gls@field@link{#1}{#2}{\glssymbolaccessdisplay{\glsentrysymbol{#2}}{#2}#3}%
11558 }

\@Glssymbol@

11559 \def\@Glssymbol@#1#2[#3]{%
11560   \gls@field@link{#1}{#2}{\glssymbolaccessdisplay{\Glsentrysymbol{#2}}{#2}#3}%
11561 }

\@GLSsymbol@

11562 \def\@GLSsymbol@#1#2[#3]{%
11563   \gls@field@link{#1}{#2}%
11564   {\glssymbolaccessdisplay{\mfirstucMakeUppercase{\glsentrysymbol{#2}}}{#2}%
11565   \mfirstucMakeUppercase{#3}}%
11566 }

lssymbolplural@

11567 \def\@glssymbolplural@#1#2[#3]{%
11568   \gls@field@link{#1}{#2}{\glssymbolpluralaccessdisplay{\glsentrysymbolplural{#2}}{#2}#3}%
11569 }

lssymbolplural@

11570 \def\@Glssymbolplural@#1#2[#3]{%
11571   \gls@field@link{#1}{#2}{\glssymbolpluralaccessdisplay{\Glsentrysymbolplural{#2}}{#2}#3}%
11572 }

LSSymbolplural@

11573 \def\@GLSsymbolplural@#1#2[#3]{%
11574   \gls@field@link{#1}{#2}%
11575   {\glssymbolpluralaccessdisplay{\mfirstucMakeUppercase{\glsentrysymbolplural{#2}}}{#2}%
11576   \mfirstucMakeUppercase{#3}}%
11577 }

\@glsuseri@

11578 \def\@glsuseri@#1#2[#3]{%
11579   \gls@field@link{#1}{#2}{\glsuseriaccessdisplay{\glsentryuseri{#2}}{#2}#3}%
11580 }

```

```

\@Glsuseri@

11581 \def\@Glsuseri#1#2[#3]{%
11582   \gls@field@link{#1}{#2}{\glsuseriaccessdisplay{\Glsentryuseri{#2}}{#2}#3}%
11583 }

\@GLSuseri@

11584 \def\@GLSuseri#1#2[#3]{%
11585   \gls@field@link{#1}{#2}{%
11586     {\glsuseriaccessdisplay{\mfirstucMakeUppercase{\glsentryuseri{#2}}}{#2}%
11587       \mfirstucMakeUppercase{#3}}%
11588 }

\@glsuserii@

11589 \def\@glsuserii#1#2[#3]{%
11590   \gls@field@link{#1}{#2}{\glsuseriiaccessdisplay{\glsentryuserii{#2}}{#2}#3}%
11591 }

\@Glsuserii@

11592 \def\@Glsuserii#1#2[#3]{%
11593   \gls@field@link{#1}{#2}{\glsuseriiaccessdisplay{\Glsentryuserii{#2}}{#2}#3}%
11594 }

\@GLSuserii@

11595 \def\@GLSuserii#1#2[#3]{%
11596   \gls@field@link{#1}{#2}{%
11597     {\glsuseriiaccessdisplay{\mfirstucMakeUppercase{\glsentryuserii{#2}}}{#2}%
11598       \mfirstucMakeUppercase{#3}}%
11599 }

\@glsuseriii@

11600 \def\@glsuseriii#1#2[#3]{%
11601   \gls@field@link{#1}{#2}{\glsuseriiiaccessdisplay{\glsentryuseriii{#2}}{#2}#3}%
11602 }

\@Glsuseriii@

11603 \def\@Glsuseriii#1#2[#3]{%
11604   \gls@field@link{#1}{#2}{\glsuseriiiaccessdisplay{\Glsentryuseriii{#2}}{#2}#3}%
11605 }

\@GLSuseriii@

11606 \def\@GLSuseriii#1#2[#3]{%
11607   \gls@field@link{#1}{#2}{%
11608     {\glsuseriiiaccessdisplay{\mfirstucMakeUppercase{\glsentryuseriii{#2}}}{#2}%
11609       \mfirstucMakeUppercase{#3}}%
11610 }

```

```

\@glsuseriv@
11611 \def\@glsuseriv@#1#2[#3]{%
11612   \gls@field@link{#1}{#2}{\glsuserivaccessdisplay{\glsentryuseriv{#2}}{#2}#3}%
11613 }

\@Glsuseriv@
11614 \def\@Glsuser@i#1#2[#3]{%
11615   \gls@field@link{#1}{#2}{\glsuserivaccessdisplay{\Glsentryuseriv{#2}}{#2}#3}%
11616 }

\@GLSuseriv@
11617 \def\@GLSuseriv@#1#2[#3]{%
11618   \gls@field@link{#1}{#2}{%
11619     {\glsuserivaccessdisplay{\mfirstucMakeUppercase{\glsentryuseriv{#2}}}{#2}%
11620       \mfirstucMakeUppercase{#3}}%
11621 }

\@glsuserv@
11622 \def\@glsuserv@#1#2[#3]{%
11623   \gls@field@link{#1}{#2}{\glsuservaccessdisplay{\glsentryuserv{#2}}{#2}#3}%
11624 }

\@Glsuserv@
11625 \def\@Glsuser@i#1#2[#3]{%
11626   \gls@field@link{#1}{#2}{\glsuservaccessdisplay{\Glsentryuserv{#2}}{#2}#3}%
11627 }

\@GLSuserv@
11628 \def\@GLSuserv@#1#2[#3]{%
11629   \gls@field@link{#1}{#2}{%
11630     {\glsuservaccessdisplay{\mfirstucMakeUppercase{\glsentryuserv{#2}}}{#2}%
11631       \mfirstucMakeUppercase{#3}}%
11632 }

\@glsuservi@
11633 \def\@glsuservi@#1#2[#3]{%
11634   \gls@field@link{#1}{#2}{\glsuserviaccessdisplay{\glsentryuservi{#2}}{#2}#3}%
11635 }

\@Glsuservi@
11636 \def\@Glsuser@i#1#2[#3]{%
11637   \gls@field@link{#1}{#2}{\glsuserviaccessdisplay{\Glsentryuservi{#2}}{#2}#3}%
11638 }

\@GLSuservi@
11639 \def\@GLSuservi@#1#2[#3]{%
11640   \gls@field@link{#1}{#2}{%
11641     {\glsuserviaccessdisplay{\mfirstucMakeUppercase{\glsentryuservi{#2}}}{#2}%
11642       \mfirstucMakeUppercase{#3}}%
11643 }

```

## 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 \acccsuppglossaryentryfield and \acccsuppglossarysubentryfield.

Now redefine \glossentryname, \glossentrydesc and \glossentrysymbol etc so they use the accessibility stuff.

```
11644 \renewcommand*{\glossentryname}[1]{%
11645   \glsdoifexists{#1}%
11646   {%
11647     \glsnamefont{\glsnameaccessdisplay{\glsentryname{#1}}{#1}}%
11648   }%
11649 }

11650 \renewcommand*{\glossentryname}[1]{%
11651   \glsdoifexists{#1}%
11652   {%
11653     \glsnamefont{\glsnameaccessdisplay{\Glsentryname{#1}}{#1}}%
11654   }%
11655 }

11656 \renewcommand*{\glossentrydesc}[1]{%
11657   \glsdoifexists{#1}%
11658   {%
11659     \glsdescriptionaccessdisplay{\glsentrydesc{#1}}{#1}%
11660   }%
11661 }

11662 \renewcommand*{\Glossentrydesc}[1]{%
11663   \glsdoifexists{#1}%
11664   {%
11665     \glsdescriptionaccessdisplay{\Glsentrydesc{#1}}{#1}}%
11666   }%
11667 }

11668 \renewcommand*{\glossentrysymbol}[1]{%
11669   \glsdoifexists{#1}%
11670   {%
11671     \glssymbolaccessdisplay{\glsentrysymbol{#1}}{#1}}%
11672   }%
11673 }

11674 \renewcommand*{\Glossentrysymbol}[1]{%
11675   \glsdoifexists{#1}%
11676   {%
11677     \glssymbolaccessdisplay{\Glsentrysymbol{#1}}{#1}}%
11678   }%
11679 }
```

ssaryentryfield

```

11680 \newcommand*{\accsuppglossaryentryfield}[5]{%
11681   \glossaryentryfield{#1}%
11682   {\glsnameaccessdisplay{#2}{#1}}%
11683   {\glsdescriptionaccessdisplay{#3}{#1}}%
11684   {\glssymbolaccessdisplay{#4}{#1}}{#5}%
11685 }

\rysubentryfield
11686 \newcommand*{\accsuppglossarysubentryfield}[6]{%
11687   \glossarysubentryfield{#1}{#2}%
11688   {\glsnameaccessdisplay{#3}{#2}}%
11689   {\glsdescriptionaccessdisplay{#4}{#2}}%
11690   {\glssymbolaccessdisplay{#5}{#2}}{#6}%
11691 }

```

## 5.4 Acronyms

Redefine acronym styles provided by glossaries:

`long-short` *<long>* (*<short>*) acronym style.

```

11692 \renewacronymstyle{long-short}%
11693 {%

```

Check for long form in case this is a mixed glossary.

```

11694 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
11695 }%
11696 {%
11697 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
11698 \renewcommand*{\genacrfullformat}[2]{%
11699   \glslongaccessdisplay{\glsentrylong{##1}}{##1}##2\space
11700   (\glsshortaccessdisplay
11701     {\protect\firstacronymfont{\glsentryshort{##1}}}{##1})%
11702 }%
11703 \renewcommand*{\Genacrfullformat}[2]{%
11704   \glslongaccessdisplay{\Glsentrylong{##1}}{##1}##2\space
11705   (\glsshortaccessdisplay
11706     {\protect\firstacronymfont{\glsentryshort{##1}}}{##1})%
11707 }%
11708 \renewcommand*{\genplacrfullformat}[2]{%
11709   \glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}##2\space
11710   (\glsshortpluralaccessdisplay
11711     {\protect\firstacronymfont{\glsentryshortpl{##1}}}{##1})%
11712 }%
11713 \renewcommand*{\Genplacrfullformat}[2]{%
11714   \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}##2\space
11715   (\glsshortpluralaccessdisplay
11716     {\protect\firstacronymfont{\glsentryshortpl{##1}}}{##1})%
11717 }%
11718 \renewcommand*{\acronymentry}[1]{%

```

```

11719     \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}%
11720     \renewcommand*{\acronymsort}[2]{##1}%
11721     \renewcommand*{\acronymfont}[1]{##1}%
11722     \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
11723     \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11724 }

```

`short-long` *<short>* (*<long>*) acronym style.

```

11725 \renewacronymstyle{short-long}%
11726 {%

```

Check for long form in case this is a mixed glossary.

```

11727 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
11728 }%
11729 {%
11730     \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
11731     \renewcommand*{\genacrfullformat}[2]{%
11732         \glsshortaccessdisplay
11733             {\protect\firstacronymfont{\glsentryshort{##1}}}{##1}##2\space
11734             (\glslongaccessdisplay{\glsentrylong{##1}}{##1})%
11735     }%
11736     \renewcommand*{\Genacrfullformat}[2]{%
11737         \glsshortaccessdisplay
11738             {\protect\firstacronymfont{\Glsentryshort{##1}}}{##1}##2\space
11739             (\glslongaccessdisplay{\glsentrylong{##1}}{##1})%
11740     }%
11741     \renewcommand*{\genplacrfullformat}[2]{%
11742         \glsshortpluralaccessdisplay
11743             {\protect\firstacronymfont{\glsentryshortpl{##1}}}{##1}##2\space
11744             (\glslongpluralaccessdisplay
11745                 {\glsentrylongpl{##1}}{##1})%
11746     }%
11747     \renewcommand*{\Genplacrfullformat}[2]{%
11748         \glsshortpluralaccessdisplay
11749             {\protect\firstacronymfont{\Glsentryshortpl{##1}}}{##1}##2\space
11750             (\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1})%
11751     }%
11752     \renewcommand*{\acronymentry}[1]{%
11753         \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}%
11754     \renewcommand*{\acronymsort}[2]{##1}%
11755     \renewcommand*{\acronymfont}[1]{##1}%
11756     \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
11757     \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11758 }

```

`long-short-desc` *<long>* ({*<short>*}) acronym style that has an accompanying description (which the user needs to supply).

```

11759 \renewacronymstyle{long-short-desc}%
11760 {%

```

```

11761 \GlsUseAcrEntryDispStyle{long-short}%
11762 }%
11763 {%
11764 \GlsUseAcrStyleDefs{long-short}%
11765 \renewcommand*\{\GenericAcronymFields\}{}%
11766 \renewcommand*\{\acronymsort}[2]{##2}%
11767 \renewcommand*\{\acronymentry}[1]{%
11768 \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11769 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11770 }

```

**g-sc-short-desc** *<long>* (*\textsc{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

11771 \renewacronymstyle{long-sc-short-desc}%
11772 {%
11773 \GlsUseAcrEntryDispStyle{long-sc-short}%
11774 }%
11775 {%
11776 \GlsUseAcrStyleDefs{long-sc-short}%
11777 \renewcommand*\{\GenericAcronymFields\}{}%
11778 \renewcommand*\{\acronymsort}[2]{##2}%
11779 \renewcommand*\{\acronymentry}[1]{%
11780 \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11781 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11782 }

```

**g-sm-short-desc** *<long>* (*\textsmaller{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

11783 \renewacronymstyle{long-sm-short-desc}%
11784 {%
11785 \GlsUseAcrEntryDispStyle{long-sm-short}%
11786 }%
11787 {%
11788 \GlsUseAcrStyleDefs{long-sm-short}%
11789 \renewcommand*\{\GenericAcronymFields\}{}%
11790 \renewcommand*\{\acronymsort}[2]{##2}%
11791 \renewcommand*\{\acronymentry}[1]{%
11792 \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11793 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11794 }

```

**short-long-desc** *<short>* ({*<long>*}) acronym style that has an accompanying description (which the user needs to supply).

```

11795 \renewacronymstyle{short-long-desc}%
11796 {%
11797 \GlsUseAcrEntryDispStyle{short-long}%
11798 }%
11799 {%
11800 \GlsUseAcrStyleDefs{short-long}%

```

```

11801 \renewcommand*\{\GenericAcronymFields\}%
11802 \renewcommand*\{\acronymsort\}[2]{##2}%
11803 \renewcommand*\{\acronymentry\}[1]{%
11804   \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11805   (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11806 }

```

**short-long-desc** *<long>* (`\textsc{<short>}`) acronym style that has an accompanying description (which the user needs to supply).

```

11807 \renewacronymstyle{sc-short-long-desc}%
11808 {%
11809   \GlsUseAcrEntryDispStyle{sc-short-long}%
11810 }%
11811 {%
11812   \GlsUseAcrStyleDefs{sc-short-long}%
11813   \renewcommand*\{\GenericAcronymFields\}%
11814   \renewcommand*\{\acronymsort\}[2]{##2}%
11815   \renewcommand*\{\acronymentry\}[1]{%
11816     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11817     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11818 }

```

**short-long-desc** *<long>* (`\textsmaller{<short>}`) acronym style that has an accompanying description (which the user needs to supply).

```

11819 \renewacronymstyle{sm-short-long-desc}%
11820 {%
11821   \GlsUseAcrEntryDispStyle{sm-short-long}%
11822 }%
11823 {%
11824   \GlsUseAcrStyleDefs{sm-short-long}%
11825   \renewcommand*\{\GenericAcronymFields\}%
11826   \renewcommand*\{\acronymsort\}[2]{##2}%
11827   \renewcommand*\{\acronymentry\}[1]{%
11828     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11829     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11830 }

```

**dua** *<long>* only acronym style.

```

11831 \renewacronymstyle{dua}%
11832 {%

```

Check for long form in case this is a mixed glossary.

```

11833 \ifdefempty\glscustomtext
11834 {%
11835   \ifglshaslong{\glslabel}%
11836   {%
11837     \glsifplural
11838   {%

```

Plural form:

```
11839      \glscapscase
11840      {%
```

Plural form, don't adjust case:

```
11841      \glslongpluralaccessdisplay{\glsentrylongpl{\glslabel}}{\glslabel}%
11842          \glsinsert
11843      }%
11844      {%
```

Plural form, make first letter upper case:

```
11845      \glslongpluralaccessdisplay{\Glsentrylongpl{\glslabel}}{\glslabel}%
11846          \glsinsert
11847      }%
11848      {%
```

Plural form, all caps:

```
11849      \glslongpluralaccessdisplay
11850          {\mfirstucMakeUppercase{\glsentrylongpl{\glslabel}}}{\glslabel}%
11851          \mfirstucMakeUppercase{\glsinsert}%
11852      }%
11853      }%
11854      {%
```

Singular form

```
11855      \glscapscase
11856      {%
```

Singular form, don't adjust case:

```
11857      \glslongaccessdisplay{\glsentrylong{\glslabel}}{\glslabel}\glsinsert
11858      }%
11859      {%
```

Subsequent singular form, make first letter upper case:

```
11860      \glslongaccessdisplay{\Glsentrylong{\glslabel}}{\glslabel}\glsinsert
11861      }%
11862      {%
```

Subsequent singular form, all caps:

```
11863      \glslongaccessdisplay
11864          {\mfirstucMakeUppercase
11865              {\glsentrylong{\glslabel}\glsinsert}}{\glslabel}%
11866          \mfirstucMakeUppercase{\glsinsert}%
11867      }%
11868      }%
11869      }%
11870      {%
```

Not an acronym:

```
11871      \glsgenentryfmt
11872      }%
11873      }%
```

```

11874 {\glscustomtext\glsinsert}%
11875 }%
11876 {%
11877 \renewcommand*\GenericAcronymFields{description={\the\glslongtok}}%
11878 \renewcommand*\acrfullfmt}[3]{%
11879 \glslink[##1]{##2}{%
11880 \glslongaccessdisplay{\glsentrylong{##2}}{##2}##3\space
11881 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2})}%
11882 \renewcommand*\Acrfullfmt}[3]{%
11883 \glslink[##1]{##2}{%
11884 \glslongaccessdisplay{\Glsentrylong{##2}}{##2}##3\space
11885 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2})}%
11886 \renewcommand*\ACRfullfmt}[3]{%
11887 \glslink[##1]{##2}{%
11888 \glslongaccessdisplay
11889 {\mfirstucMakeUppercase{\glsentrylong{##2}}}{##2}##3\space
11890 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2})}%
11891 \renewcommand*\acrfullplfmt}[3]{%
11892 \glslink[##1]{##2}{%
11893 \glslongpluralaccessdisplay
11894 {\glsentrylongpl{##2}}{##2}##3\space
11895 (\glsshortpluralaccessdisplay
11896 {\acronymfont{\glsentryshortpl{##2}}}{##2})}%
11897 \renewcommand*\Acrfullplfmt}[3]{%
11898 \glslink[##1]{##2}{%
11899 \glslongpluralaccessdisplay
11900 {\Glsentrylongpl{##2}}{##2}##3\space
11901 (\glsshortpluralaccessdisplay
11902 {\acronymfont{\glsentryshortpl{##2}}}{##2})}%
11903 \renewcommand*\ACRfullplfmt}[3]{%
11904 \glslink[##1]{##2}{%
11905 \glslongpluralaccessdisplay
11906 {\mfirstucMakeUppercase{\glsentrylongpl{##2}}}{##2}##3\space
11907 (\glsshortpluralaccessdisplay
11908 {\acronymfont{\glsentryshortpl{##2}}}{##2})}%
11909 \renewcommand*\glsentryfull}[1]{%
11910 \glslongaccessdisplay{\glsentrylong{##1}}\space
11911 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})%
11912 }%
11913 \renewcommand*\Glsentryfull}[1]{%
11914 \glslongaccessdisplay{\Glsentrylong{##1}}{##1}\space
11915 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})%
11916 }%
11917 \renewcommand*\glsentryfullpl}[1]{%
11918 \glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}\space
11919 (\glsshortpluralaccessdisplay{\acronymfont{\glsentryshortpl{##1}}}{##1})%
11920 }%
11921 \renewcommand*\Glsentryfullpl}[1]{%
11922 \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}\space

```

```

11923   (\glsshortpluralaccessdisplay{\acronymfont{\glsentryshortpl{##1}}}{##1})%
11924 }%
11925 \renewcommand*{\acronymentry}[1]{%
11926   \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}%
11927 \renewcommand*{\acronymsort}[2]{##1}%
11928 \renewcommand*{\acronymfont}[1]{##1}%
11929 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11930 }

```

**dua-desc** <*long*> only acronym style with user-supplied description.

```

11931 \renewacronymstyle{dua-desc}%
11932 {%
11933   \GlsUseAcrEntryDispStyle{dua}%
11934 }%
11935 {%
11936   \GlsUseAcrStyleDefs{dua}%
11937 \renewcommand*{\GenericAcronymFields}{}%
11938 \renewcommand*{\acronymentry}[1]{%
11939   \glslongaccessdisplay{\acronymfont{\glsentrylong{##1}}}{##1}%
11940 \renewcommand*{\acronymsort}[2]{##2}%
11941 }%

```

**footnote** <*short*>\footnote{<*long*>} acronym style.

```

11942 \renewacronymstyle{footnote}%
11943 {%
  Check for long form in case this is a mixed glossary.
11944 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
11945 }%
11946 {%
11947 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%

```

Need to ensure hyperlinks are switched off on first use:

```

11948 \glshyperfirstfalse
11949 \renewcommand*{\genacrfullformat}[2]{%
11950   \glsshortaccessdisplay
11951     {\protect\firstacronymfont{\glsentryshort{##1}}}{##1}##2%
11952   \protect\footnote{\glslongaccessdisplay{\glsentrylong{##1}}}{##1}%
11953 }%
11954 \renewcommand*{\Genacrfullformat}[2]{%
11955   \glsshortaccessdisplay
11956     {\firstacronymfont{\Glsentryshort{##1}}}{##1}##2%
11957   \protect\footnote{\glslongaccessdisplay{\glsentrylong{##1}}}{##1}%
11958 }%
11959 \renewcommand*{\genplacrfullformat}[2]{%
11960   \glsshortpluralaccessdisplay
11961     {\protect\firstacronymfont{\glsentryshortpl{##1}}}{##1}##2%
11962   \protect\footnote{\glslongpluralaccessdisplay{\glsentrylongpl{##1}}}{##1}%
11963 }%
11964 \renewcommand*{\Genplacrfullformat}[2]{%

```

```

11965 \glsshortpluralaccessdisplay
11966 {\protect\firstacronymfont{\Glsentryshortpl{##1}}}{##1}##2%
11967 \protect\footnote{\glslongpluralaccessdisplay{\glsentrylongpl{##1}}}{##1}%
11968 }%
11969 \renewcommand*{\acronymentry}[1]{%
11970   \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}%
11971 \renewcommand*{\acronymsort}[2]{##1}%
11972 \renewcommand*{\acronymfont}[1]{##1}%
11973 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%

```

Don't use footnotes for \acrfull:

```

11974 \renewcommand*{\acrfullfmt}[3]{%
11975   \glslink[##1]{##2}{%
11976     \glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}##3\space
11977     (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}%
11978 \renewcommand*{\Acrfullfmt}[3]{%
11979   \glslink[##1]{##2}{%
11980     \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##2}}}{##2}##3\space
11981     (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}%
11982 \renewcommand*{\ACRfullfmt}[3]{%
11983   \glslink[##1]{##2}{%
11984     \glsshortaccessdisplay
11985       {\mfirstucMakeUppercase
11986         {\acronymfont{\glsentryshort{##2}}}{##2}##3\space
11987         (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}%
11988 \renewcommand*{\acrfullplfmt}[3]{%
11989   \glslink[##1]{##2}{%
11990     \glsshortpluralaccessdisplay
11991       {\acronymfont{\glsentryshortpl{##2}}}{##2}##3\space
11992       (\glslongpluralaccessdisplay{\glsentrylongpl{##2}}{##2})}%
11993 \renewcommand*{\Acrfullplfmt}[3]{%
11994   \glslink[##1]{##2}{%
11995     \glsshortpluralaccessdisplay
11996       {\acronymfont{\Glsentryshortpl{##2}}}{##2}##3\space
11997       (\glslongpluralaccessdisplay{\glsentrylongpl{##2}})}%
11998 \renewcommand*{\ACRfullplfmt}[3]{%
11999   \glslink[##1]{##2}{%
12000     \glsshortpluralaccessdisplay
12001       {\mfirstucMakeUppercase
12002         {\acronymfont{\glsentryshortpl{##2}}}{##2}##3\space
12003         (\glslongpluralaccessdisplay{\glsentrylongpl{##2}})}%

```

Similarly for \glsentryfull etc:

```

12004 \renewcommand*{\glsentryfull}[1]{%
12005   \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}\space
12006   (\glslongaccessdisplay{\glsentrylong{##1}}{##1})}%
12007 \renewcommand*{\Glsentryfull}[1]{%
12008   \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##1}}}{##1}\space
12009   (\glslongaccessdisplay{\glsentrylong{##1}}{##1})}%
12010 \renewcommand*{\glsentryfullpl}[1]{%

```

```

12011     \glsshortpluralaccessdisplay
12012         {\acronymfont{\glsentryshortpl{##1}}}{##1}\space
12013         (\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1})}%
12014 \renewcommand*{\Glsentryfullpl}[1]{%
12015     \glsshortpluralaccessdisplay
12016         {\acronymfont{\Glsentryshortpl{##1}}}{##1}\space
12017         (\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1})}%
12018 }

footnote-sc \textsc{\langle short \rangle}\footnote{\langle long \rangle} acronym style.
12019 \renewacronymstyle{footnote-sc}%
12020 {%
12021     \GlsUseAcrEntryDispStyle{footnote}%
12022 }%
12023 {%
12024     \GlsUseAcrStyleDefs{footnote}%
12025     \renewcommand{\acronymentry}[1]{%
12026         \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}%
12027     \renewcommand{\acronymfont}[1]{\textsc{##1}}%
12028     \renewcommand*{\acrpluralsuffix}{\glstextup{\glspluralsuffix}}}%
12029 }%

footnote-sm \textsmaller{\langle short \rangle}\footnote{\langle long \rangle} acronym style.
12030 \renewacronymstyle{footnote-sm}%
12031 {%
12032     \GlsUseAcrEntryDispStyle{footnote}%
12033 }%
12034 {%
12035     \GlsUseAcrStyleDefs{footnote}%
12036     \renewcommand{\acronymentry}[1]{%
12037         \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}%
12038     \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
12039     \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}}%
12040 }%

footnote-desc \langle short \rangle\footnote{\langle long \rangle} acronym style that has an accompanying description (which the user needs to supply).
12041 \renewacronymstyle{footnote-desc}%
12042 {%
12043     \GlsUseAcrEntryDispStyle{footnote}%
12044 }%
12045 {%
12046     \GlsUseAcrStyleDefs{footnote}%
12047     \renewcommand*{\GenericAcronymFields}{}%
12048     \renewcommand*{\acronymsort}[2]{##2}%
12049     \renewcommand*{\acronymentry}[1]{%
12050         \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
12051         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
12052 }

```

ootnote-sc-desc \textsc{\{short\}}\footnote{\{long\}} acronym style that has an accompanying description (which the user needs to supply).

```
12053 \renewacronymstyle{footnote-sc-desc}%
12054 {%
12055   \GlsUseAcrEntryDispStyle{footnote-sc}%
12056 }%
12057 {%
12058   \GlsUseAcrStyleDefs{footnote-sc}%
12059   \renewcommand*\{GenericAcronymFields\}{}%
12060   \renewcommand*\{acronymsort\}[2]{##2}%
12061   \renewcommand*\{acronymentry\}[1]{%
12062     \glslongaccessdisplay{\glsentrylong{##1}{##1}\space
12063     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}{##1}})}%
12064 }
```

ootnote-sm-desc \textsmaller{\{short\}}\footnote{\{long\}} acronym style that has an accompanying description (which the user needs to supply).

```
12065 \renewacronymstyle{footnote-sm-desc}%
12066 {%
12067   \GlsUseAcrEntryDispStyle{footnote-sm}%
12068 }%
12069 {%
12070   \GlsUseAcrStyleDefs{footnote-sm}%
12071   \renewcommand*\{GenericAcronymFields\}{}%
12072   \renewcommand*\{acronymsort\}[2]{##2}%
12073   \renewcommand*\{acronymentry\}[1]{%
12074     \glslongaccessdisplay{\glsentrylong{##1}{##1}\space
12075     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}{##1}})}%
12076 }
```

**\glsdefaultshortaccess{\{long\}}{\{short\}}**

Default shortaccess value.

```
12077 \newcommand*\{glsdefaultshortaccess\}[2]{#1}
```

Use \newacronymhook to modify the key list to set the access text to the long version by default.

```
12078 \renewcommand*\{newacronymhook\}%
12079   \edef\@gls@keylist{%
12080     shortaccess=\glsdefaultshortaccess{\the\glslongtok}{\the\glsshorttok},%
12081     shortpluralaccess=\glsdefaultshortaccess{\the\glslongtok}{\the\glsshorttok},%
12082     \the\glskeylisttok}%
12083   \expandafter\glskeylisttok\expandafter\{@gls@keylist}%
12084 }
```

**ltNewAcronymDef** Modify default style to use access text:

```
12085 \renewcommand*\{DefaultNewAcronymDef\}%
```

```

12086 \edef\@do@newglossaryentry{%
12087   \noexpand\newglossaryentry{\the\glslabeltok}%
12088   {%
12089     type=\acronymtype,%
12090     name={\the\glsshorttok},%
12091     description={\the\glslongtok},%
12092     descriptionaccess=\relax,
12093     text={\the\glsshorttok},%
12094     access={\noexpand\@glo@textaccess},%
12095     sort={\the\glsshorttok},%
12096     short={\the\glsshorttok},%
12097     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
12098     shortaccess={\glsdefaultshortaccess{\glslongtok}{\the\glsshorttok}},%
12099     long={\the\glslongtok},%
12100     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
12101     descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
12102     first={\noexpand\glslongaccessdisplay
12103       {\the\glslongtok}{\the\glslabeltok}\space
12104       (\noexpand\glsshortaccessdisplay
12105         {\the\glsshorttok}{\the\glslabeltok})},%
12106     plural={\the\glsshorttok\acrpluralsuffix},%
12107     firstplural={\noexpand\glslongpluralaccessdisplay
12108       {\noexpand\@glo@longpl}{\the\glslabeltok}\space
12109       (\noexpand\glsshortpluralaccessdisplay
12110         {\noexpand\@glo@shortpl}{\the\glslabeltok})},%
12111     firstaccess=\relax,
12112     firstpluralaccess=\relax,
12113     textaccess={\noexpand\@glo@shortaccess},%
12114     \the\glskeylisttok
12115   }%
12116 }%
12117 \let\@org@gls@assign@firstpl\gls@assign@firstpl
12118 \let\@org@gls@assign@plural\gls@assign@plural
12119 \let\@org@gls@assign@descplural\gls@assign@descplural
12120 \def\gls@assign@firstpl##1##2{%
12121   \@@gls@expand@field{##1}{firstpl}{##2}%
12122 }%
12123 \def\gls@assign@plural##1##2{%
12124   \@@gls@expand@field{##1}{plural}{##2}%
12125 }%
12126 \def\gls@assign@descplural##1##2{%
12127   \@@gls@expand@field{##1}{descplural}{##2}%
12128 }%
12129 \do@newglossaryentry
12130 \let\gls@assign@firstpl\@org@gls@assign@firstpl
12131 \let\gls@assign@plural\@org@gls@assign@plural
12132 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
12133 }

```

```

teNewAcronymDef
12134 \renewcommand*{\DescriptionFootnoteNewAcronymDef}{%
12135   \edef\@do@newglossaryentry{%
12136     \noexpand\newglossaryentry{\the\glslabeltok}%
12137     {%
12138       type=\acronymtype,%
12139       name={\noexpand\acronymfont{\the\glsshorttok}},%
12140       sort={\the\glsshorttok},%
12141       text={\the\glsshorttok},%
12142       short={\the\glsshorttok},%
12143       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
12144       shortaccess={\glsdefaultshortaccess{\the\glslongtok}{\the\glsshorttok}},%
12145       long={\the\glslongtok},%
12146       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
12147       access={\noexpand\@glo@textaccess},%
12148       plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
12149       symbol={\the\glslongtok},%
12150       symbolplural={\the\glslongtok\noexpand\acrpluralsuffix},%
12151       firstpluralaccess=\relax,
12152       textaccess={\noexpand\@glo@shortaccess},%
12153       \the\glskeylisttok
12154     }%
12155   }%
12156   \let\@org@gls@assign@firstpl\gls@assign@firstpl
12157   \let\@org@gls@assign@plural\gls@assign@plural
12158   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
12159   \def\gls@assign@firstpl##1##2{%
12160     \@@gls@expand@field{##1}{firstpl}{##2}%
12161   }%
12162   \def\gls@assign@plural##1##2{%
12163     \@@gls@expand@field{##1}{plural}{##2}%
12164   }%
12165   \def\gls@assign@symbolplural##1##2{%
12166     \@@gls@expand@field{##1}{symbolplural}{##2}%
12167   }%
12168   \do@newglossaryentry
12169   \let\gls@assign@plural\@org@gls@assign@plural
12170   \let\gls@assign@firstpl\@org@gls@assign@firstpl
12171   \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
12172 }

```

onNewAcronymDef

```

12173 \renewcommand*{\DescriptionNewAcronymDef}{%
12174   \edef\@do@newglossaryentry{%
12175     \noexpand\newglossaryentry{\the\glslabeltok}%
12176     {%
12177       type=\acronymtype,%
12178       name={\noexpand
12179         \acrnameformat{\the\glsshorttok}{\the\glslongtok}},%

```

```

12180     access={\noexpand\@glo@textaccess},%
12181     sort={\the\glsshorttok},%
12182     short={\the\glsshorttok},%
12183     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
12184     shortaccess={\glsdefaultshortaccess{\the\glslongtok}{\the\glsshorttok}},%
12185     long={\the\glslongtok},%
12186     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
12187     first={\the\glslongtok},%
12188     firstaccess=\relax,
12189     firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
12190     text={\the\glsshorttok},%
12191     textaccess={\the\glslongtok},%
12192     plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
12193     symbol={\noexpand\@glo@text},%
12194     symbolaccess={\noexpand\@glo@textaccess},%
12195     symbolplural={\noexpand\@glo@plural},%
12196     firstpluralaccess=\relax,
12197     textaccess={\noexpand\@glo@shortaccess},%
12198     \the\glskeylisttok}%
12199 }%
12200 \let\@org@gls@assign@firstpl\gls@assign@firstpl
12201 \let\@org@gls@assign@plural\gls@assign@plural
12202 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
12203 \def\gls@assign@firstpl##1##2{%
12204   \@@gls@expand@field{##1}{firstpl}{##2}%
12205 }%
12206 \def\gls@assign@plural##1##2{%
12207   \@@gls@expand@field{##1}{plural}{##2}%
12208 }%
12209 \def\gls@assign@symbolplural##1##2{%
12210   \@@gls@expand@field{##1}{symbolplural}{##2}%
12211 }%
12212 \do@newglossaryentry
12213 \let\gls@assign@firstpl\@org@gls@assign@firstpl
12214 \let\gls@assign@plural\@org@gls@assign@plural
12215 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
12216 }

```

#### teNewAcronymDef

```

12217 \renewcommand*\FootnoteNewAcronymDef{%
12218   \edef\do@newglossaryentry{%
12219     \noexpand\newglossaryentry{\the\glslabeltok}%
12220     {%
12221       type=acronymtype,%
12222       name={\noexpand\acronymfont{\the\glsshorttok}},%
12223       sort={\the\glsshorttok},%
12224       text={\the\glsshorttok},%
12225       textaccess={\the\glslongtok},%
12226       access={\noexpand\@glo@textaccess},%

```

```

12227     plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
12228     short={\the\glsshorttok},%
12229     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
12230     long={\the\glslongtok},%
12231     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
12232     description={\the\glslongtok},%
12233     descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
12234     \the\glskeylisttok
12235   }%
12236 }%
12237 \let\@org@gls@assign@plural\gls@assign@plural
12238 \let\@org@gls@assign@firstpl\gls@assign@firstpl
12239 \let\@org@gls@assign@descplural\gls@assign@descplural
12240 \def\gls@assign@firstpl##1##2{%
12241   \@@gls@expand@field{##1}{firstpl}{##2}%
12242 }%
12243 \def\gls@assign@plural##1##2{%
12244   \@@gls@expand@field{##1}{plural}{##2}%
12245 }%
12246 \def\gls@assign@descplural##1##2{%
12247   \@@gls@expand@field{##1}{descplural}{##2}%
12248 }%
12249 \do@newglossaryentry
12250 \let\gls@assign@plural\@org@gls@assign@plural
12251 \let\gls@assign@firstpl\@org@gls@assign@firstpl
12252 \let\gls@assign@descplural\@org@gls@assign@descplural
12253 }

```

## 11NewAcronymDef

```

12254 \renewcommand*\SmallNewAcronymDef{%
12255   \edef\do@newglossaryentry{%
12256     \noexpand\newglossaryentry{\the\glslabeltok}%
12257   }%
12258   type=\acronymtype,%
12259   name={\noexpand\acronymfont{\the\glsshorttok}},%
12260   access={\noexpand\@glo@symbolaccess},%
12261   sort={\the\glsshorttok},%
12262   short={\the\glsshorttok},%
12263   shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
12264   shortaccess={\glsdefaultshortaccess{\the\glslongtok}{\the\glsshorttok}},%
12265   long={\the\glslongtok},%
12266   longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
12267   text={\noexpand\@glo@short},%
12268   textaccess={\noexpand\@glo@shortaccess},%
12269   plural={\noexpand\@glo@shortpl},%
12270   first={\the\glslongtok},%
12271   firstaccess=\relax,
12272   firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
12273   description={\noexpand\@glo@first},%

```

```

12274     descriptionplural={\noexpand\@glo@firstplural},%
12275     symbol={\the\glsshorttok},%
12276     symbolaccess={\the\glslongtok},%
12277     symbolplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
12278     \the\glskeylisttok
12279   }%
12280 }%
12281 \let\@org@gls@assign@firstpl\gls@assign@firstpl
12282 \let\@org@gls@assign@plural\gls@assign@plural
12283 \let\@org@gls@assign@descplural\gls@assign@descplural
12284 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
12285 \def\gls@assign@firstpl##1##2{%
12286   \@@gls@expand@field{##1}{firstpl}{##2}%
12287 }%
12288 \def\gls@assign@plural##1##2{%
12289   \@@gls@expand@field{##1}{plural}{##2}%
12290 }%
12291 \def\gls@assign@descplural##1##2{%
12292   \@@gls@expand@field{##1}{descplural}{##2}%
12293 }%
12294 \def\gls@assign@symbolplural##1##2{%
12295   \@@gls@expand@field{##1}{symbolplural}{##2}%
12296 }%
12297 \do@newglossaryentry
12298 \let\gls@assign@firstpl\@org@gls@assign@firstpl
12299 \let\gls@assign@plural\@org@gls@assign@plural
12300 \let\gls@assign@descplural\@org@gls@assign@descplural
12301 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
12302 }

```

The following are kept for compatibility with versions before 3.0:

```

\shortaccesskey
12303 \newcommand*{\glsshortaccesskey}{\glsshortkey access}%

\pluralaccesskey
12304 \newcommand*{\glsshortpluralaccesskey}{\glsshortpluralkey access}%

\longaccesskey
12305 \newcommand*{\glslongaccesskey}{\glslongkey access}%

\pluralaccesskey
12306 \newcommand*{\glslongpluralaccesskey}{\glslongpluralkey access}%

```

## 5.5 Debugging Commands

```

\showgлонameaccess
12307 \newcommand*{\showgлонameaccess}[1]{%

```

```

12308 \expandafter\show\csname glo@\glsdetoklabel{#1}@access\endcsname
12309 }

owglotextaccess
12310 \newcommand*{\showglotextaccess}[1]{%
12311   \expandafter\show\csname glo@\glsdetoklabel{#1}@textaccess\endcsname
12312 }

glopluralaccess
12313 \newcommand*{\showglopluralaccess}[1]{%
12314   \expandafter\show\csname glo@\glsdetoklabel{#1}@pluralaccess\endcsname
12315 }

wglofirstaccess
12316 \newcommand*{\showglofirstaccess}[1]{%
12317   \expandafter\show\csname glo@\glsdetoklabel{#1}@firstaccess\endcsname
12318 }

rstpluralaccess
12319 \newcommand*{\showglofirstpluralaccess}[1]{%
12320   \expandafter\show\csname glo@\glsdetoklabel{#1}@firstpluralaccess\endcsname
12321 }

glosymbolaccess
12322 \newcommand*{\showglosymbolaccess}[1]{%
12323   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolaccess\endcsname
12324 }

bolpluralaccess
12325 \newcommand*{\showglosymbolpluralaccess}[1]{%
12326   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolpluralaccess\endcsname
12327 }

owglodescaccess
12328 \newcommand*{\showglodescaccess}[1]{%
12329   \expandafter\show\csname glo@\glsdetoklabel{#1}@descaccess\endcsname
12330 }

escpluralaccess
12331 \newcommand*{\showglodescpluralaccess}[1]{%
12332   \expandafter\show\csname glo@\glsdetoklabel{#1}@descpluralaccess\endcsname
12333 }

wgloshortaccess
12334 \newcommand*{\showgloshortaccess}[1]{%
12335   \expandafter\show\csname glo@\glsdetoklabel{#1}@shortaccess\endcsname
12336 }

```

```
ortpluralaccess
12337 \newcommand*{\showgloshortpluralaccess}[1]{%
12338   \expandafter\show\csname glo@\glsdetoklabel{#1}@shortpluralaccess\endcsname
12339 }

owglolongaccess
12340 \newcommand*{\showglolongaccess}[1]{%
12341   \expandafter\show\csname glo@\glsdetoklabel{#1}@longaccess\endcsname
12342 }

ongpluralaccess
12343 \newcommand*{\showglolongpluralaccess}[1]{%
12344   \expandafter\show\csname glo@\glsdetoklabel{#1}@longpluralaccess\endcsname
12345 }
```

# 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.

```
12346 \NeedsTeXFormat{LaTeX2e}
12347 \ProvidesPackage{glossaries-babel}[2020/03/19 v4.46 (NLCT)]
```

Load `tracklang` to obtain language settings.

```
12348 \RequirePackage{tracklang}
12349 \let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
12350 \AnyTrackedLanguages
12351 {%
12352     \ForEachTrackedDialect{\this@dialect}{%
12353         \IfTrackedLanguageFileExists{\this@dialect}{%
12354             {glossaries-}\% prefix
12355             {.ldf}\%
12356             {%
12357                 \RequireGlossariesLang{\CurrentTrackedTag}\%
12358             }%
12359             {%
12360                 \PackageWarningNoLine{glossaries}%
12361                 {No language module detected for '\this@dialect'. \MessageBreak
12362                  Language modules need to be installed separately. \MessageBreak
12363                  Please check on CTAN for a bundle called \MessageBreak
12364                  'glossaries-\CurrentTrackedLanguage' or similar}\%
12365             }%
12366             {%
12367             }%
12368             {}%
12369         }%
12370     }%
```

## 6.1 Polyglossia Captions

Language support has now been split off into independent language modules.

```
12369 \NeedsTeXFormat{LaTeX2e}
12370 \ProvidesPackage{glossaries-polyglossia}[2020/03/19 v4.46 (NLCT)]
```

Load `tracklang` to obtain language settings.

```
12371 \RequirePackage{tracklang}
12372 \let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
12373 \AnyTrackedLanguages
```

```
12374  {%
12375    \ForEachTrackedDialect{\this@dialect}{%
12376      \IfTrackedLanguageFileExists{\this@dialect}{%
12377        {glossaries-}\% prefix
12378        {.ldf}\%
12379        {%
12380          \RequireGlossariesLang{\CurrentTrackedTag}\%
12381        }%
12382        {%
12383          \PackageWarningNoLine{glossaries}\%
12384          {No language module detected for '\this@dialect'.\MessageBreak
12385            Language modules need to be installed separately.\MessageBreak
12386            Please check on CTAN for a bundle called\MessageBreak
12387            'glossaries-\CurrentTrackedLanguage' or similar}\%
12388        }%
12389      }%
12390    }%
12391  {}%
```

# Glossary

`makeindex` An indexing application [10](#), [14](#), [30](#), [33](#), [187](#), [197](#)

`xindy` An flexible indexing application with multilingual support written in Perl [10](#), [14](#), [30](#), [33](#), [187](#), [197](#)

# Change History

1.01 (2007-05-17)	numberline: numberline option added . . . 8
General: Added range facility in format key ..... 119	
\writeist: Added spaces after \delimN and \delimR in ist file ..... 167	
1.04 (2007-08-03)	
General: Added \glstextformat ..... 104	
1.05 (2007-08-10)	
\glossarysection: added \@mkboth to \glossarysection ..... 46	
\gls@defglossaryentry: Changed the default value of the sort key to just the value of the name key ..... 87	
1.07 (2007-09-13)	
\@gls@link: fixed bug caused by \the\glsentrycounter setting the page number too soon ..... 117	
\glsadd: fixed bug caused by \the\glsentrycounter setting the page number too soon ..... 164	
1.08 (2007-10-13)	
General: Added babel support ..... 40	
listgroup: changed listgroup style to use \glsgetgroupstyle ..... 282	
altlistgroup: changed altlistgroup style to use \glsgetgroupstyle ..... 283	
1.1 (2008-02-22)	
\@glossarysection: numbered sections and auto label added ..... 47	
\@gls@tmpb: changed \toksdef to \newtoks ..... 122	
\@gls@toc: numberline added ..... 49	
\@p@glossarysection: numbered sections and auto label added ..... 48	
General: amsgen now loaded (\new@ifnextchar needed) ..... 4	
translate: translate option added ..... 28	
\setglossarysection: new ..... 47	
numberedsection: numberedsection package option added ..... 9	
1.12 (2008-03-08)	
\@GLSpl: now uses \glsentrydescplural and \glsentrysymbolplural instead of \glsentrydesc and \glsentrysymbol ..... 133	
\@GLSpl@: now uses \glsentrydescplural and \glsentrysymbolplural instead of \glsentrydesc and \glsentrysymbol ..... 132	
\@glspl@: now uses \glsentrydescplural and \glsentrysymbolplural instead of \glsentrydesc and \glsentrysymbol ..... 132	
General: added check for \hypertarget separate to \hyperlink (memoir defines \hyperlink but not \hypertarget) ..... 128	
descriptionplural: new ..... 70	
\gls@defglossaryentry: Changed default first plural to be first key with s appended (was text key with s appended) ..... 87	
descriptionplural support added ..... 87	
symbolplural support added ..... 87	
\Glsentrydescplural: New ..... 157	
\glsentrydescplural: New ..... 157	
\Glsentrysymbolplural: New ..... 158	
\glsentrysymbolplural: New ..... 158	
\SetDescriptionFootnoteAcronymStyle: Added \protect before \footnote and \glslink ..... 248	
\SetFootnoteAcronymStyle: Added \protect before \footnote and \glslink ..... 254	
symbolplural: new ..... 71	

1.13 (2008-05-10)	
General: fixed bug that ignored 3rd parameter .....	135–142
\ACRfullpl: new .....	229
\Acrfullpl: new .....	229
\acrfullpl: new .....	228
\acrpluralsuffix: New .....	226
\gls@defglossaryentry: Changed default first value .....	87
Changed default firstplural value .....	87
Removed restriction on only using \newglossaryentry in the preamble .....	92
\newacronym: Removed restriction on only using \newacronym in the preamble .....	226
1.14 (2008-06-17)	
\@gls@hypergroup: new .....	277
General: added nonumberlist key to \printglossary .....	213
added numberedsection key to \printglossary .....	212
\firstracronymfont: new .....	230
\glsautoprefix: new .....	8
\glsnavhyperlink: changed \edef to \protected@edef .....	276
\glsnavhypertarget: added write to aux file .....	276
\glsnavigation: changed to only use labels for groups that are present ..	278
1.15 (2008-08-15)	
\@gls@link: added \glslabel .....	117
\gls@defglossaryentry: check for \glo@first in description .....	91
check for \glo@text in symbol .....	91
\gls@hypergrouprerun: new .....	277
\glsnavhypertarget: added check if rerun required .....	276
\glssettoctitle: new .....	39
\printglossary: changed the way the TOC title is set .....	197
1.16 (2008-08-27)	
\@GLS@: Test glossary type is \acronymtype in addition to checking if footnote option has been used .....	131
\@GLSp1: Test glossary type is \acronymtype in addition to checking if footnote option has been used .....	133
\@Gls@: Test glossary type is \acronymtype in addition to checking if footnote option has been used .....	130
\@Glspl@: Test glossary type is \acronymtype in addition to checking if footnote option has been used .....	132
\@gls@: Test glossary type is \acronymtype in addition to checking if footnote option has been used .....	129
\@glsdisp: Test glossary type is \acronymtype in addition to checking if footnote option has been used .....	134
\@glspl@: Test glossary type is \acronymtype in addition to checking if footnote option has been used .....	132
\@glstarget: raised the hypertarget so the target text doesn't scroll off the top of the page .....	128
\gls@defglossaryentry: Changed def to let .....	87
1.17 (2008-12-26)	
\@odo@esc@wrglossary: new .....	191
\@do@seeglossary: new .....	195
\@glo@storeentry: new .....	94
\@gls@glossary: changed definition to use \index instead of \index .....	187
\@glsdefaultplural: new .....	74
\@glsdefaultsort: new .....	75
\@glshypernumber: new .....	223
\@glsnoname: new .....	74
\@glsnonextpages: new .....	213
General: added xindy support .....	30
parent: new .....	72
see: new .....	71
\gls@defglossaryentry: added nonumberlist key .....	88
added parent key .....	88
added see key .....	88
Stored main part of entry format when entry is defined .....	92
\gls@suffixF: new .....	44
\gls@suffixFF: new .....	44
\gls@wrglossary: modified to allow for xindy support .....	188

\glshyperlink: new	163	\SetDescriptionFootnoteAcronymStyle:	
\glshypernumber: modified to allow material to be attached to location	223	changed \acronymfont to use \textsmaller instead of \smaller	248
\glsnavhyperlink: replaced \hyperlink to \@glslink	276	\SetFootnoteAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	254
\glsnavhypertarget: replaced \hypertarget to \@glstarget	276	\SetSmallAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	257
\glssee: new	195	2.01 (2009 May 30)	
\glsseeformat: new	196	\@gls@link: moved \@do@wrglossary before term is displayed to prevent unwanted whatsit	118
\glsSetSuffixF: new	44	\forallglossaries: replaced \ifthenelse with \ifix	58
\glsSetSuffixFF: new	44	\forglsentries: replaced \ifthenelse with \ifix	58
\ifglsxindy: new	30	\glsdefmain: new	17
\listfilename: added xindy support	43	\glsdescwidth: changed \linewidth to \hsize	284, 306
\newglossarystyle: made \newglossarystyle long	222	\glslistdottedwidth: changed \linewidth to \hsize	284
\nopostdesc: new	42	\gspagelistwidth: changed \linewidth to \hsize	284, 306
\nonumberlist: new	72	nomain: added nomain package option	17
\printglossary: added check to determine if \printglossary is already defined	197	\writeist: removed item_02 - no such makeindex key	171
added print language to aux file	197	2.02 (2007-07-13)	
order: order package option added	30	\@printglossary: suppressed warning globally rather than locally	200
\writeist: added xindy support	167	2.02 (2009-07-13)	
1.18 (2009-01-14)		\glossarysection: changed \mkboth to \glossarymark	46
\@gls@loadlist: new	11	\glsGLOSSARYMARK: New	46
\@gls@loadlong: new	10	2.03 (2009-09-23)	
\@gls@loadsuper: new	11	\@GLS@: Added check for hyperfirst	131
\@gls@loadtree: new	11	\@GLSp1: Added check for hyperfirst	133
\gls@defglossaryentry: Changed default value of sort to \glsdefaultsort	87	\@G1s@: Added check for hyperfirst	130
moved sort sanitization to \newglossaryentry	91	\@GLSp1@: Added check for hyperfirst	132
\glstarget: new	216	\@gls@: Added check for hyperfirst	129
\oldacronym: new	225	\@gls@link: new	116
\nolist: new	11	\@gls@link: added \leavevmode	117
\nolong: new	10	Moved entry existence check to avoid duplicate code	117
sort: moved sanitization to \newglossaryentry	70	\@glsdisp: Added check for hyperfirst	134
\nostyles: new	11	\@glspl@: Added check for hyperfirst	132
\nosuper: new	11	\glsGLOSSARYMARK: Added check to see if it's already defined	46
\notree: new	11	hyperfirst: new	29
1.19 (2009-03-02)			
\glsclearpage: new	48		
\glsdisp: new	134		
\SetDescriptionAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	252		

2.04 (2009-11-10)		
\@GLS@: Changed test to check if glossary type has been identified as a list of acronyms .....	131	
\@GLSp1: Changed test to check if glossary type has been identified as a list of acronyms .....	133	
\@Gls@: Changed test to check if glossary type has been identified as a list of acronyms .....	130	
\@Glspl@: Changed test to check if glossary type has been identified as a list of acronyms .....	132	
\@glossaryentryfield: new .....	93	
\@glossarysubentryfield: new .....	93	
\@gls@: Changed test to check if glossary type has been identified as a list of acronyms .....	129	
\@glsacronymlists: new .....	18	
\@glsdisp: Changed test to check if glossary type has been identified as a list of acronyms .....	134	
\@glspl@: Changed test to check if glossary type has been identified as a list of acronyms .....	132	
\@newglossaryentryposthook: new ..	93	
\@newglossaryentryprehook: new ..	93	
acronymlists: new .....	20	
\DeclareAcronymList: new .....	19	
\DefineAcronymSynonyms: new .....	243	
\gls@defglossaryentry: added user1-6 keys .....	88	
\glsadd: fixed bug that ignored counter	164	
\Glsentryuseri: new .....	160	
\glsentryuseri: new .....	160	
\Glsentryuserii: new .....	160	
\glsentryuserii: new .....	160	
\Glsentryuseriii: new .....	160	
\glsentryuseriii: new .....	160	
\Glsentryuseriv: new .....	160	
\glsentryuseriv: new .....	160	
\Glsentryuserserv: new .....	161	
\glsentryuserserv: new .....	160	
\Glsentryuserservi: new .....	161	
\glsentryuserservi: new .....	161	
\ns@newglossary: added check to determine if \gls@<type>@display and \gls@<type>@displayfirst have been defined. ....	67	
	\SetAcronymLists: new .....	20
	\SetDefaultAcronymDisplayStyle: new .....	245
	\SetDefaultAcronymStyle: new .....	246
	\SetDescriptionAcronymDisplayStyle: new .....	250
	\SetDescriptionDUAAcronymDisplayStyle: new .....	249
	\SetDescriptionFootnoteAcronymDisplayStyle: new .....	246
	\SetDUADisplayStyle: new .....	258
	\SetFootnoteAcronymDisplayStyle: new .....	253
	\SetSmallAcronymDisplayStyle: new .....	255
2.05 (2010-02-06)		
	\@glsdisp: Added closing brace. Patch provided by Sergiu Dotenco .....	134
	Removed spurious brace. Patch provided by Sergiu Dotenco .....	134
	\writeist: Added \string before opening and closing braces. Patch provided by Segiu Dotenco .....	172
2.06 (2010-06-14)		
	\altnewglossary: new .....	67
	\CustomAcronymFields: new .....	260
	\CustomNewAcronymDef: new .....	260
	\SetCustomDisplayStyle: new .....	260
	\SetCustomStyle: new .....	261
2.07 (2010-07-10)		
	General: glsadd format key stored in \@glsnumberformat (was mistakenly stored in \@glo@format) .....	164
3.0 (2010-07-12)		
	\@makeglossary: Added check for savewrites .....	177
	\gls@wrglossary: modified to take into account savewrites .....	188
3.0 (2010/03/31)		
	\@set@glo@numformat: added 4th argument .....	120
3.0 (2011-04-02)		
	\@odo@esc@wrglossary: added check for hyper location prefix .....	193
	modified to use new format .....	191
	\@cglossarysec: replaced \@ifundefined with \ifcsundef ...	8
	\@do@seeglossary: Sanitize and escape cross-referencing information .....	195
	\@gls@counterwithin: new .....	13

\@gls@ifinlist: new .....	50
\@gls@link: added	
\@gls@saveentrycounter .....	118
added \@gls@setsort .....	118
\@gls@saveentrycounter: new .....	118
\@gls@setupsort@def: new .....	15
\@gls@setupsort@standard: new .....	14
\@gls@setupsort@use: new .....	15
\@gls@xdy@locationlist: new .....	53
\@glslink: replaced \@ifundefined	
with \ifcsundef .....	128
\@glsnextpages: new .....	214
\@print@glossary: replaced	
\@ifundefined with \ifcsundef .	201
\@printglossary: added	
\currentglossary .....	199
added \glsnextpages .....	200
make toctitle default to title .....	199
\@xdyattributelist: new .....	49
General: added prefix to hyperlink .....	224
etoolbox now loaded .....	4
replaced \@ifundefined with	
\ifcsundef .....	38, 41, 114, 211
\acrfootnote: new .....	246
\ACRfull: added starred version .....	228
\Acrfull: added starred version .....	227
\acrfull: added starred version .....	227
\ACRfullpl: added starred version ...	229
\Acrfullpl: added starred version ...	229
\acrfullpl: added starred version ...	228
\acrlinkfootnote: new .....	246
\acrnolinkfootnote: new .....	246
\savewrites: new .....	33
see: added \glo@seeautonumberlist	71
seeautonumberlist: new .....	10
\glossarysection: replaced	
\@ifundefined with \ifcsundef ..	46
\glossarystyle: replaced	
\@ifundefined with \ifcsundef .	221
\gls@codepage: replaced	
\@ifundefined with \ifcsundef ..	31
\gls@defglossaryentry: added	
\@gls@defsort .....	91
added short and long keys .....	88
replaced \@ifundefined with	
\ifcsundef .....	88
\gls@doclearpage: replaced	
\@ifundefined with \ifcsundef ..	48
\glsadd: added	
\@gls@saveentrycounter .....	164
\GlsAddXdyCounters: new .....	50
\glsentrycounterlabel: new .....	215
\glsentryitem: new .....	215
\Glsentrylong: new .....	161
\glsentrylong: new .....	161
\Glsentrylongpl: new .....	162
\glsentrylongpl: new .....	161
\Glsentryshort: new .....	161
\glsentryshort: new .....	161
\Glsentryshortpl: new .....	161
\glsentryshortpl: new .....	161
\glsgetgrouptitle: replaced	
\@ifundefined with \ifcsundef .	220
\glsGLOSSARYmark: replaced	
\@ifundefined with \ifcsundef ..	46
\glshyperlink: changed default from	
\glsentryname to \glsentrytext	163
\glshypernumber: replaced	
\@ifundefined with \ifcsundef ..	223
\glsnumberformat: replaced	
\@ifundefined with \ifcsundef ..	45
\glsrefentry: new .....	215
\glsresetsubentrycounter: new ...	214
\glsseeitem: hyperlink uses	
\glsseeitemformat instead of	
\glsentryname .....	196
\glsseeitemformat: new .....	196
\glsortnumberfmt: new .....	14
\glsstepentry: new .....	214
\glsstepsubentry: new .....	215
\glssubentrycounterlabel: new ...	215
\glssubentryitem: new .....	215
\theglossary: replaced \@ifundefined	
with \ifcsundef .....	216
short: new .....	73
shortplural: new .....	74
\ifglossaryexists: replaced	
\@ifundefined with \ifcsundef ..	59
\ifglsentryexists: replaced	
\@ifundefined with \ifcsundef ..	60
\istfile: deprecated .....	186
\glossaryentry: new .....	12
\glossarysubentry: new .....	13
\newglossaryentry: replaced	
\DeclareRobustCommand with	
\newrobustcmd .....	77

\newglossarystyle: replaced	224
\@ifundefined with \ifcsundef .	222
\ns@newglossary: added	
\@gls@defsortcount .....	67
replaced \@ifundefined with	
\ifcsundef .....	67
entrycounter: new .....	12
\oldacronym: replaced \@ifundefined	
with \ifcsundef .....	225
compatible-2.07: compatible-2.07	
option added .....	34
long: new .....	74
longplural: new .....	74
nonumberlist: now boolean .....	72
sort: new .....	13
counter: replaced \@ifundefined with	
\ifcsundef .....	71
counterwithin: new .....	12
\printglossary: replaced	
\@ifundefined with \ifcsundef .	197
\SetDescriptionFootnoteAcronymDisplayStyle	
expanded options link options .....	246
\setentrycounter: added optional	
argument .....	221
\showacronymlists: new .....	266
\showglocounter: new .....	263
\showglodesc: new .....	264
\showglodesplural: new .....	265
\showglofirst: new .....	263
\showglofirstpl: new .....	263
\showgloflag: new .....	266
\showgloindex: new .....	266
\showglevel: new .....	262
\showgloname: new .....	264
\showgloparent: new .....	262
\showgloplural: new .....	262
\showglosort: new .....	265
\showglossaries: new .....	266
\showglossarycounter: new .....	267
\showglossaryentries: new .....	267
\showglossaryin: new .....	266
\showglossaryout: new .....	267
\showglossarytitle: new .....	267
\showglosymbol: new .....	265
\showglosymbolplural: new .....	265
\showglotext: new .....	262
\showglotype: new .....	263
\showglouserii: new .....	263
\showglouseriii: new .....	263
\showglouseriv: new .....	264
\showglouserv: new .....	264
\showglouservi: new .....	264
subentrycounter: new .....	13
\writeist: added xindy-only macro	
definitions to glossary open tag .....	169
modified to support new format .....	167
3.01 (2011-04-12)	
\@glswritefiles: added check for	
empty glossaries .....	186
General: made robust .....	130
\ACRfull: made robust .....	228
\Acrfull: made robust .....	227
\acrfull: made robust .....	227
\acrfullformat: removed	
\acronymfont as it should already be	
set in the second argument. ....	227
\ACRfullpl: made robust .....	229
\Acrfullpl: made robust .....	229
\acrfullpl: made robust .....	228
\ACRlong: made robust .....	152
\Acrlong: made robust .....	151
\acrlong: made robust .....	150
\ACRlongpl: made robust .....	154
\Acrlongpl: made robust .....	153
\acrlongpl: made robust .....	152
\ACRshort: made robust .....	148
\Acrshort: made robust .....	147
\acrshort: made robust .....	147
\ACRshortpl: made robust .....	150
\Acrshortpl: made robust .....	149
\acrshortpl: made robust .....	149
\Gls: made robust .....	130
\glsadd: made robust .....	164
\glsaddall: made robust .....	165
\GLSdesc: made robust .....	139
\Glsdesc: made robust .....	139
\glsdesc: made robust .....	139
\GLSdescplural: made robust .....	140
\Glsdescplural: made robust .....	140
\glsdescplural: made robust .....	140
\glsfirst: made robust .....	135
\GLSfirstplural: made robust .....	138
\Glsfirstplural: made robust .....	137
\glsfirstplural: made robust .....	137
\glslink: made robust .....	116
\GLSname: made robust .....	138
\Glsname: made robust .....	138

\glsname: made robust .....	138
\GLSp1: made robust .....	133
\Glsp1: made robust .....	132
\glspl: made robust .....	131
\GLSplural: made robust .....	137
\GLSsymbol: made robust .....	141
\Glssymbol: made robust .....	141
\glssymbol: made robust .....	140
\GLSsymbolplural: made robust .....	142
\Glssymbolplural: made robust .....	141
\glssymbolplural: made robust .....	141
\Glstext: made robust .....	135
\glstext: made robust .....	135
\GLSuseri: made robust .....	143
\Glsuseri: made robust .....	142
\glsuseri: made robust .....	142
\GLSuserii: made robust .....	143
\Glsuserii: made robust .....	143
\glsuserii: made robust .....	143
\GLSuseriii: made robust .....	144
\Glsuseriii: made robust .....	144
\glsuseriii: made robust .....	144
\GLSuseriv: made robust .....	145
\Glsuseriv: made robust .....	145
\glsuseriv: made robust .....	145
\GLSuserv: made robust .....	146
\Glsuserv: made robust .....	146
\glsuserv: made robust .....	145
\GLSservi: made robust .....	147
\Glsservi: made robust .....	146
\glservi: made robust .....	146
3.02 (2012-05-19)	
\glsnumlistlastsep: new .....	163
\glsnumlistsep: new .....	163
3.02 (2012-05-21)	
\@do@wrglossary: changed	
\glslocref to	
\theglsentrycounter .....	194
\@do@wrglossary: changed	
\do@wr@glossary to test for	
indexonlyfirst option; put old	
\do@wr@glossary code into	
\@do@wrglossary .....	188
\@gls@missingnumberlist: new .....	74
\@glswritefiles: added check for	
existence of token in case	
\makeglossaries has been	
omitted .....	186
\@printglossary: add a way to fetch	
current entry label .....	200
\savenunderlist: new .....	10
\ucmark: new .....	12
\gls@defglossaryentry: added	
numberlist element .....	91
\gls@save@numberlist: new .....	197
\gls@wrglossary: added check for	
glossary file defined .....	188
\glsdisplaynumberlist: new .....	162
\glsentrycounter: set default value ..	118
\Glsentryfull: fixed bug (replaced)	
\glsentryshortpl with	
\glsentryshort) .....	162
\glsentryfullpl: fixed bug (replaced)	
\glsentryshort with	
\glsentryshortpl) .....	162
\glsentrynumberlist: new .....	162
\glsmoveentry: new .....	93
\glsresetsubentrycounter: new ..	214
\ifglshaschildren: new .....	61
\ifglshasparent: new .....	62
\makeglossaries: added list parser ..	181
\indexonlyfirst: new .....	29
\renewglossarystyle: new .....	222
\showglossaryentries: fixed misspelt	
command .....	267
\SmallNewAcronymDef: fixed broken	
short and long plural .....	256
3.03 (2012/09/21)	
\@gls@sanitizesort: new .....	23
\@gls@setupsort@standard: used	
\@gls@sanitizesort .....	14
\@printglossary: allow title to override	
default toctitle .....	199
General: allow title to set toctitle .....	211
\glsinlinedescformat: new .....	280
\glsinlineemptydescformat: new ..	280
\glsinlineformat: new .....	280
\glsinlinepostchild: new .....	280
\glsinlinesubdescformat: new .....	280
\glsinlinesubnameformat: new .....	280
\glspostinline: replaced “.” with	
\glspostdescription .....	280
list: added check for glsnogroupskip ..	282
altsuperragged4col: added check for	
glsnogroupskip .....	299
altsuperragged4col: added check for	
glsnogroupskip .....	318

alttree: added check for	
glsnogroupskip .....	327
index: added check for glsnogroupskip	321
nogroupskip: new .....	12
long: added check for glsnogroupskip .	285
long3col: added check for	
glsnogroupskip .....	287
long4col: added check for	
glsnogroupskip .....	288
longragged: added check for	
glsnogroupskip .....	296
longragged3col: added check for	
glsnogroupskip .....	297
nopostdot: new .....	12
tree: added check for glsnogroupskip .	322
treenoname: added check for	
glsnogroupskip .....	324
super: added check for glsnogroupskip	307
super3col: added check for	
glsnogroupskip .....	309
super4col: added check for	
glsnogroupskip .....	311
superragged: added check for	
glsnogroupskip .....	314
superragged3col: added check for	
glsnogroupskip .....	316
3.04 (2012-11-11)	
altlist: replaced \newline with	
paragraph break .....	282
3.04 (2012-11-18)	
\@do@wrglossary: changed	
\the\glstentrycounter back to	
\@glslocref .....	194
\@do@esc@wrglossary: modified to	
compensate for possible incorrect	
page number .....	192
\@gls@escbsdq: unsanitize	
\gls@numberpage, \gls@alphpage,	
\gls@Alphpage and	
\gls@romanpage .....	121
\@print@glossary: Moved aux write to	
end of document to prevent	
unwanted whatsit occurring here. .	201
General: Added check for doc package .	4
added datatool-base as a required	
package .....	4
added local key .....	115
\gls@Alphpage: new .....	189
\gls@alphpage: new .....	189
\gls@disablepagerefexpansion: new	189
\gls@numberpage: new .....	189
\gls@protected@pagefmts: new .....	189
\gls@romanpage: new .....	189
\glsdefmain: added check for doc	
package .....	17
\glsorg@endtheglossary: new .....	5
\glsorg@theglossary: new .....	5
\PrintChanges: new .....	5
3.05 (2013-04-21)	
\@do@esc@wrglossary: add Roman	
case. Fixed bugs in the else	
statements .....	192
\@gls@link: added check for	
“nohypertypes” .....	117
\mcolalttree: replaced ‘2’ with	
\glsmcols .....	305
\mcolindex: replaced ‘2’ with \glsmcols	301
\mcolindexspannav: replaced ‘2’ with	
\glsmcols .....	302
\mcoltree: replaced ‘2’ with \glsmcols	302
\mcoltreenoname: replaced ‘2’ with	
\glsmcols .....	304
\mcoltreespannav: replaced ‘2’ with	
\glsmcols .....	303
\gls@protected@pagefmts: added	
Roman to list .....	189
\gls@Romanpage: new .....	189
\glsgetgrouplabel: fixed bug (typo in	
\equal) .....	220
\nopostdesc: made robust .....	42
3.05 (2013/04/21)	
\@gls@nohyperlist: new .....	20
\GlsDeclareNoHyperList: new .....	20
nohypertypes: new .....	20
3.06 (2013/06/17)	
\@xdy@main@language: Changed back to	
using \languagename .....	30
\findrootlanguage: Obsoleted .....	56
3.07 (2013-07-05)	
\@gls@link: fixed bug that failed to find	
entry in list .....	117
\glossarypreamble: modified to work	
with \setglossarypreamble .....	45
\gls@docclearpage: added check for	
openright .....	48
\glspostdescription: Added	
spacefactor code .....	11

\GlsSetXdyCodePage: Added check for fontspec .....	57	\glsseelist: made robust .....	196
\SetDescriptionAcronymDisplayStyle: now using \glsdoparenifnotempty	250	\ifglsdescsuppressed: new .....	62
\setglossarypreamble: new .....	45	\ifglshasdesc: new .....	62
3.08a (2013-08-30)		\ifglshassymbol: new .....	62
list: updated list style to use \glossentry and \subglossentry	281	altnragged4col: updated to use \glossentry and \subglossentry	299
listdotted: updated listdotted style to use \glossentry and \subglossentry .....	283	alttree: updated to use \glossentry and \subglossentry .....	326
altlist: updated altlist style to use \glossentry and \subglossentry	282	index: added paragraph break at end of environment .....	320
inline: updated inline style to use \glossentry and \subglossentry	279	updated to use \glossentry and \subglossentry .....	320
3.08a (2013-09-28)		long: updated to use \glossentry and \subglossentry .....	285
{@glo@storeentry: no longer need to check for special characters in any of the fields other than sort .....	94	longragged: updated to use \glossentry and \subglossentry	296
updated for \glossentry .....	94	longragged3col: updated to use \glossentry and \subglossentry	297
{@glossaryentryfield: switched to \glossentry .....	93	tree: updated to use \glossentry and \subglossentry .....	322
{@glossarysubentryfield: switched to \subglossentry .....	93	\setglossarystyle: new .....	221
General: added nogroupskip key to \printglossary .....	212	\setglossentrycompatibility: new	218
removed definition of {@glossaryentryfield .....	380	superragged: updated to use \glossentry and \subglossentry	314
removed definition of {@glossarysubentryfield .....	380	3.09a (2013-10-09)	
\compatibleglossentry: new .....	216	{@gls@assign@symbolplural@field: new .....	23
\compatiblesubglossentry: new .....	218	{@gls@default@value: new .....	70
\glossaryentryfield: deprecated .....	218	\Glsentrydesc: made robust .....	157
\Glossentrydesc: new .....	217	\Glsentrydescplural: made robust ..	157
\glossentrydesc: new .....	217	\Glsentryfirst: made robust .....	158
\Glossentryname: new .....	217	\Glsentryfirstplural: made robust ..	159
\glossentryname: new .....	217	\Glsentryfull: made robust .....	162
\Glossentrysymbol: new .....	217	\Glsentryfullpl: made robust .....	162
\glossentrysymbol: new .....	217	\Glsentrylong: made robust .....	161
\gls@assign@desc@field: new .....	22	\Glsentrylongpl: made robust .....	162
\gls@assign@descplural@field: new	22	\Glsentryname: made robust .....	156
\gls@assign@field: new .....	76	\Glsentryplural: made robust .....	158
\gls@ifnotmeasuring: new .....	95	\Glsentryshort: made robust .....	161
\glsaddallunused: new .....	165	\Glsentryshortpl: made robust .....	161
\glsexpandfields: new .....	76	\Glsentrysymbol: made robust .....	158
\glsnoexpandfields: new .....	77	\Glsentrysymbolplural: made robust	158
\glssee: made robust .....	195	\Glsentrytext: made robust .....	157
\glsseeformat: made robust .....	196	\Glsentryuseri: made robust .....	160
\glsseeitem: made robust .....	196	\Glsentryuserii: made robust .....	160
		\Glsentryuseriii: made robust .....	160
		\Glsentryuseriv: made robust .....	160
		\Glsentryuserv: made robust .....	161
		\Glsentryuserserv: made robust .....	161

\glstextup: new .....	226
\ifglsassymbol: changed test to check for \@gls@default@symbol .....	62
3.10a (2013-09-28)	
\gls@assign@type@field: new .....	22
3.10a (2013-10-13)	
\@gls@keymap: new .....	79
\@gls@provide@newglossary: new ...	65
\@gls@writedef: new .....	78
\@glsdefaultplural: Obsolete .....	74
\@glsnodec: new .....	74
\@print@glossary: Added providecommand code to aux file ..	201
\gls@defglossaryentry: Changed to using \@gls@default@value .....	87
new .....	87
\glswritelnhook: new .....	86
\makeglossaries: Added providecommand code to aux file ..	180
\new@glossaryentry: new .....	77
\ns@newglossary: added \@gls@provide@newglossary ....	67
3.11a (2013-10-15)	
\@ACRlong: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	374
\@ACRshort: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	373
\@Acrlong: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	374
\@Acrshort: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	372
\@GLS@: add \glslabel, \glsifplural, \glscapscase, \glscustomtext and \glsinsert .....	131
change to using \glsentryfmt style commands .....	131
removed \MakeUppercase (now moved to \glsentryfmt) .....	131
\@GLSpl: add \glslabel, \glsifplural, \glscapscase, \glscustomtext and \glsinsert	133
change to using \glsentryfmt style commands .....	133
removed \MakeUppercase as now dealt with in \glsentryfmt .....	133
\@Gls@: add \glsifplural, \glscapscase, \glscustomtext and \glsinsert .....	130
change to using \glsentryfmt style commands .....	130
removed \makefirstuc (now dealt with in \glsentryfmt) .....	130
\@Glspl@: add \glsifplural, \glscapscase, \glscustomtext and \glsinsert .....	132
change to using \glsentryfmt style commands .....	132
removed \makefirstuc (now dealt with in \glsentryfmt) .....	132
\@acrlong: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	373
\@acrshort: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	372
\@gls@: add \glslabel, \glsifplural, \glscapscase, \glscustomtext and \glsinsert .....	129
change to using \glsentryfmt style commands .....	129
\@gls@noexpand@fields: Fixed bug expand replaced with noexpand ....	75
\@glsdisp: add \glslabel, \glsifplural, \glscapscase, \glscustomtext and \glsinsert	134
change to using \glsentryfmt style commands .....	134
\@glspl@: add \glslabel, \glsifplural, \glscapscase, \glscustomtext and \glsinsert	131
change to using \glsentryfmt style commands .....	132
General: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext .....	147–154
changed to just use \Glsentrydescplural .....	140
changed to just use \glsentrydescplural .....	140
changed to just use \Glsentrydesc ..	139
changed to just use \glsentrydesc ..	139
changed to just use \Glsentryfirstplural .....	138

changed to just use	
\glsentryfirstplural .....	137, 138
changed to just use \Glsentryfirst	136
changed to just use \glsentryfirst	136
changed to just use \Glsentryname .	138
changed to just use	
\glsentryname .....	138, 139
changed to just use \Glsentryplural	137
changed to just use	
\glsentryplural .....	136, 137
changed to just use	
\Glsentrysymbolplural .....	142
changed to just use	
\glsentrysymbolplural .....	141, 142
changed to just use \Glsentrysymbol	141
changed to just use \glsentrysymbol	141
Changed to just use \Glsentrytext .	135
changed to just use \glsentrytext .	135
changed to just use	
\Glsentryuseriii .....	144
changed to just use	
\glsentryuseriii .....	144
changed to just use \Glsentryuserii	143
changed to just use	
\glsentryuserii .....	143, 144
changed to just use \Glsentryuseriv	145
changed to just use \glsentryuseriv	145
changed to just use \Glsentryuseri	142
changed to just use	
\glsentryuseri .....	142, 143
changed to just use \Glsentryuserservi	147
changed to just use	
\glsentryuserservi .....	146, 147
changed to just use \Glsentryuserserv	146
changed to just use	
\glsentryuserserv .....	145, 146
Now requires textcase .....	4
acronymlists: replaced	
@addtoacronymlists with	
\DeclareAcronymList .....	20
\defglsdisplay: obsoleted .....	113
\defglsdisplayfirst: obsoleted .....	113
\defglsentryfmt: new .....	65
\forglsentries: replaced \ifx with	
\ifdefempty .....	58
\gls@assign@desc: new .....	86
\gls@defglossaryentry: Fixed default	
counter if none supplied .....	90
\gls@doentryfmt: new .....	65
\glsdisplay: obsoleted .....	113
\glsdisplayfirst: obsoleted .....	113
\glsentryfmt: new .....	108
\glsgetgroupitle: Added check in	
case non-Latin alphabet in use .....	220
\glsGLOSSARYmark: replaced	
\MakeUppercase with	
\mfirstrucMakeUppercase .....	46
\glsnavigation: switched to using	
\@gls@getgroupitle .....	278
\ifglshasdesc: replaced \ifdefempty	
with \ifcsempyty .....	62
\ifglshaslong: new .....	63
\ifglshasshort: new .....	63
\ifglshassymbol: replaced	
\ifdefempty with \ifcsempyty ....	62
\ifglsused: replaced \ifthenelse with	
\ifbool .....	60
\longnewglossaryentry: new .....	86
\ns@newglossary: replaced	
\glsdisplay and	
\glsdisplayfirst with	
\glsentryfmt .....	67
compatible-3.07:cnew .....	33
\SetCustomDisplayStyle: updated to	
use \defglsentryfmt .....	260
\SetDefaultAcronymDisplayStyle:	
changed to use \defglsentryfmt .	245
\SetDescriptionAcronymDisplayStyle:	
updated to use \defglsentryfmt .	250
\SetDescriptionDUAACronymDisplayStyle:	
updated to use \defglsentryfmt .	249
\SetDescriptionFootnoteAcronymDisplayStyle:	
updated to use \defglsentryfmt .	246
\SetDUADisplayStyle: updated to use	
\defglsentryfmt .....	258
\SetFootnoteAcronymDisplayStyle:	
updated to use \defglsentryfmt .	253
\SetSmallAcronymDisplayStyle:	
updated to use \defglsentryfmt .	255
\setupglossaries: new .....	37
\showglolong: new .....	265
\showgloshort: new .....	265
numbers: new .....	36
symbols: new .....	35
3.12a (2013-10-16)	
\gls@defglossaryentry: added	
\glslabel .....	87
\glsaddkey: new .....	81

3.13a (2013-11-05)	
\@gls@assign@symbol@field: changed	
to use \glssetnoexpandfield .....	23
\@gls@assign@symbolplural@field:	
changed to use	
\glssetnoexpandfield .....	23
\@gls@link: removed \relax .....	118
\@gls@notranslatorhook: new .....	27
\@gls@setupsort@standard: moved	
\@gls@santizesort to	
\glsprestandardsort .....	14
ucmark: added check for memoir .....	12
see: added \gls@checkseallowed ...	71
\glossarysection: changed	
\glossarymark to	
\glsglossarymark .....	46
\glossarystyle: fixed bug caused by	
using \ifdef instead of \ifcsdef .	221
\gls@assign@desc@field: changed to	
use \glssetnoexpandfield .....	22
\gls@assign@descplural@field:	
changed to use	
\glssetnoexpandfield .....	22
\gls@assign@name@field: changed to	
use \glssetnoexpandfield .....	22
\gls@assign@type@field: changed to	
use \glssetexpandfield .....	22
\gls@checkseallowed: new .....	72
\glsaddallunused: set default to	
\@glo@types .....	165
\Glsentryfull: changed to use	
\acrfullformat .....	162
\glsentryfull: changed to use	
\acrfullformat .....	162
\Glsentryfullpl: changed to use	
\acrfullformat .....	162
\glsentryfullpl: changed to use	
\acrfullformat .....	162
\glsglossarymark: renamed	
\glossarymark to	
\glsglossarymark to avoid conflict	
with memoir .....	46
\glsprestandardsort: new .....	14
\glssetexpandfield: new .....	22
\glssetnoexpandfield: new .....	22
altsuper4colheader: switched to	
\tabularnewline .....	312
altsuper4colheaderborder: switched	
to \tabularnewline .....	313
long: switched to \tabularnewline ..	285
long3col: switched to	
\tabularnewline .....	286
long3colheader: switched to	
\tabularnewline .....	287
long3colheaderborder: switched to	
\tabularnewline .....	287
long4col: switched to	
\tabularnewline .....	288
long4colheader: switched to	
\tabularnewline .....	288
longheader: switched to	
\tabularnewline .....	286
longheaderborder: switched to	
\tabularnewline .....	286
\SetFootnoteAcronymDisplayStyle:	
fixed missing argument bug .....	253
super: switched to \tabularnewline .	307
super3col: switched to	
\tabularnewline .....	309
super3colheader: switched to	
\tabularnewline .....	309
super4col: switched to	
\tabularnewline .....	310
super4colheader: switched to	
\tabularnewline .....	311
super4colheaderborder: switched to	
\tabularnewline .....	311
superheader: switched to	
\tabularnewline .....	308
superheaderborder: switched to	
\tabularnewline .....	308
3.14a (2013-11-12)	
\@glswritefiles: renamed	
\glswritefiles to	
\@glswritefiles and used	
“savewrites” option to set	
\glswritefiles .....	186
General: new .....	269
acronyms: new .....	18
\gls@defglossaryentry: added check	
for existence of default glossary .....	88
set the default for firstplural to be the	
value of plural .....	90
xindygloss: new .....	31
\longprovideglossaryentry: new ...	87
compatible-2.07: added check for 2.07	
before setting 3.07 compatibility .....	34
nottranslate: new .....	27

\provideglossaryentry: new	77	index: new	36
4.0 (2013-11-14)		\newacronymstyle: new	232
\gls@defglossaryentry: added check		long-sc-short: new	235
for first key	90	long-sc-short-desc: new	236
super: fixed typo in \subglossentry		long-short: new	233
(\glossentrydesc)	307	long-short-desc: new	236
4.01 (2013-11-16)		long-sm-short: new	235
General: fixed non-value options so that		long-sm-short-desc: new	237
they can be passed to document class	9	long-sp-short-desc: new	236
\CustomAcronymFields: inserted		footnote: new	240
missing comma	260	footnote-desc: new	242
4.02 (2013-12-05)		footnote-sc: new	241
@\acrfull: now using \acrfullfmt	227	footnote-sc-desc: new	242
@\gls@indexdef: new	37	footnote-sm: new	242
@\gls@numbersdef: new	36	footnote-sm-desc: new	243
@\gls@symbolsdef: new	36	\setacronymstyle: new	232
General: Removed \acronymfont	151–154	\SetDescriptionAcronymDisplayStyle:	
\ACRfullfmt: new	228	Moved check for empty custom text to	
\Acrfullfmt: new	228	prevent unwanted parenthetical	
\acrfullfmt: new	227	material	250
\ACRfullplfmt: new	229	\SetDescriptionFootnoteAcronymDisplayStyle:	
\Acrfullplfmt: new	229	Moved check for empty custom text to	
\acrfullplfmt: new	229	prevent unwanted parenthetical	
\acronymentry: new	231	material	246
sanitize: fixed bug that caused an error		\SetFootnoteAcronymDisplayStyle:	
here	26	Moved check for empty custom text to	
sc-short-long: new	235	prevent unwanted parenthetical	
sc-short-long-desc: new	237	material	253
\Genacrfullformat: new	112	\SetGenericNewAcronym: new	230
\genacrfullformat: new	112	\SetSmallAcronymDisplayStyle:	
\GenericAcronymFields: new	231	Moved check for empty custom text to	
\Genplacrfullformat: new	113	prevent unwanted parenthetical	
\genplacrfullformat: new	112	material	255
\Glsentryfull: bug fix: added missing		dua: new	238
\acronymfont	162	dua-desc: new	240
\glsentryfull: bug fix: added missing		numberedsection: added nameref	
\acronymfont	162	option	9
\Glsentryfullpl: bug fix: added		4.02 (2013-13-05)	
missing \acronymfont	162	\makeglossaries: made preamble only	182
\glsentryfullpl: bug fix: added		4.03 (2014-01-17)	
missing \acronymfont	162	General: changed default to \empty	
\glsgenacfmt: new	110	instead of \relax	33
\GlsUseAcrEntryDispStyle: new	233	4.03 (2014-01-20)	
\GlsUseAcrStyleDefs: new	233	\@odo@esc@wrglossary: added	
short-long: new	234	\glsdetoklabel	193
short-long-desc: new	237	\@odo@noesc@wrglossary: added	
xindynoglsnumbers: new	31	\glsdetoklabel	191
sm-short-long: new	236	\@ACRlong: removed \glslabel	
sm-short-long-desc: new	238	(defined in \gls@link)	374

\@ACRshort: removed \glslabel (define in \@gls@link) .....	373
\@Acrlong: removed \glslabel (define in \@gls@link) .....	374
\@Acrshort: removed \glslabel (define in \@gls@link) .....	372
\@GLS@: removed \glslabel (defined in \@gls@link) .....	131
\@GLSpl: removed \glslabel (defined in \@gls@link) .....	133
\@Gls@: removed \glslabel (defined in \@gls@link) .....	130
\@Gls@entry@field: new .....	155
\@Gspl@: removed \glslabel (defined in \@gls@link) .....	132
\@acrlong: removed \glslabel (define in \@gls@link) .....	373
\@acrshort: removed \glslabel (define in \@gls@link) .....	372
\@gls@: removed \glslabel (defined in \@gls@link) .....	129
\@gls@access@display: new .....	359
\@gls@entry@field: new .....	155
\@gls@fetchfield: new .....	79
\@gls@field@link: new .....	134
\@gls@link: added \glsdetoklabel . moved \@gls@link@opts and \@gls@link@label to \@gls@link	117
\@gls@writedef: added \glsdetoklabel .....	78
\@glsdisp: removed \glslabel (define in \@gls@link) .....	134
\@gspcl@: removed \glslabel (defined in \@gls@link) .....	131
\@printglossary: added \glsdetoklabel .....	200
General: removed \glslabel (defined in \@gls@link) .....	147
sc-short-long-desc: redefined to use accessibility information .....	384
\compatibleglossentry: added \glsdetoklabel .....	349
\compatiblesubglossentry: added \glsdetoklabel .....	350
\Genacrfullformat: redefined to use accessibility information .....	371
\genacrfullformat: redefined to use accessibility information .....	371
\Genplacrfullformat: redefined to use accessibility information .....	372
\genplacrfullformat: redefined to use accessibility information .....	372
\glossentryname: added \glsdetoklabel .....	217
\gls@defglossaryentry: added \glsdetoklabel .....	87
replaced #1 with \@glo@label .....	88
replaced \ifthenelse with \ifdefequal .....	89
\glsadd: added \glsdetoklabel ....	164
\glsaddkey: switched to using \@gls@field@link .....	82
\glsdetoklabel: new .....	59
\glsdisplaynumberlist: added \glsdetoklabel .....	163
\glsdoifexistsorwarn: new .....	60
\glsentryaccess: switched to using \@gls@entry@field .....	355
\glsentrydescaccess: switched to using \@gls@entry@field .....	356
\glsentrydescpluralaccess: switched to using \@gls@entry@field .....	356
\glsentryfirstaccess: switched to using \@gls@entry@field .....	356
\glsentryfirstplural: added \glsdetoklabel .....	159
\glsentrylongaccess: switched to using \@gls@entry@field .....	357
\glsentrylongpluralaccess: switched to using \@gls@entry@field .....	357
\glsentrypluralaccess: switched to using \@gls@entry@field .....	356
\glsentryshortaccess: switched to using \@gls@entry@field .....	356
\glsentryshortpluralaccess: switched to using \@gls@entry@field .....	356
\glsentrysymbolaccess: switched to using \@gls@entry@field .....	356
\glsentrysymbolpluralaccess: switched to using \@gls@entry@field .....	356
\glsentrytextaccess: switched to using \@gls@entry@field .....	355
\glsgenacfmt: redefined to use accessibility information .....	369

\glsgenentryfmt: redefined to use accessibility information .....	366
\glshyperlink: added	
\glsdetoklabel .....	163
\glslocalreset: added	
\glsdetoklabel .....	96
\glslocalunset: added	
\glsdetoklabel .....	96
\glsmoveentry: added	
\glsdetoklabel .....	93
replaced \ifthenelse with	
\ifdefequal .....	93
\glsrefentry: added	\glsdetoklabel 215
\glsreset: added	\glsdetoklabel ... 96
\glsseelist: added	\expandafter commands .....
196	
\glsstepentry: added	
\glsdetoklabel .....	214
\glsstepsubentry: added	
\glsdetoklabel .....	215
\glsunset: added	\glsdetoklabel ... 96
short-long: commented spurious EOL	235
redefined to use accessibility information .....	382
short-long-desc: redefined to use accessibility information .....	383
\ifglsdescsuppressed: added	
\glsdetoklabel .....	62
fixed typo .....	62
\ifglsentryexists: added	
\glsdetoklabel .....	60
\ifglschildren: added	
\glsdetoklabel .....	61
\ifglsdesc: added	
\glsdetoklabel .....	62
\ifglsfield: new	63
\ifglslong: added	
\glsdetoklabel .....	63
\ifglsparent: added	
\glsdetoklabel .....	62
\ifglsshort: added	
\glsdetoklabel .....	63
\ifglssymbol: added	
\glsdetoklabel .....	62
replaced \ifcempty with	
\ifdefempty and replaced \ifx with	
\ifdefequal .....	62
\ifglsused: added	\glsdetoklabel .. 60
sm-short-long-desc: redefined to use accessibility information .....	384
long-sc-short-desc: redefined to use accessibility information .....	383
long-short: redefined to use accessibility information .....	381
long-short-desc: redefined to use accessibility information .....	382
long-sm-short-desc: redefined to use accessibility information .....	383
footnote: redefined to use accessibility information .....	387
footnote-desc: redefined to use accessibility information .....	389
footnote-sc: redefined to use accessibility information .....	389
footnote-sc-desc: redefined to use accessibility information .....	390
footnote-sm: redefined to use accessibility information .....	389
footnote-sm-desc: redefined to use accessibility information .....	390
\renewacronymstyle: new	232
\showglocounter: added	
\glsdetoklabel .....	263
\showglodesc: added	\glsdetoklabel 264
\showglodescaccess: added	
\glsdetoklabel .....	396
\showglodescpplural: added	
\glsdetoklabel .....	265
\showglodescppluralaccess: added	
\glsdetoklabel .....	396
\showglofirst: added	
\glsdetoklabel .....	263
\showglofirstaccess: added	
\glsdetoklabel .....	396
\showglofirstpl: added	
\glsdetoklabel .....	263
\showglofirstplpluralaccess: added	
\glsdetoklabel .....	396
\showgloflag: added	\glsdetoklabel 266
\showgloindex: added	
\glsdetoklabel .....	266
\showglolevel: added	
\glsdetoklabel .....	262
\showglolong: added	\glsdetoklabel 265
\showglolongaccess: added	
\glsdetoklabel .....	397

\showglolongpluralaccess: added	redefined to use accessibility information .....	387
\glsdetoklabel .....	397	
\showgloname: added \glsdetoklabel	264	
\showglonameaccess: added		
\glsdetoklabel .....	395	
\showgloparent: added		
\glsdetoklabel .....	262	
\showgloplural: added		
\glsdetoklabel .....	262	
\showglopluralaccess: added		
\glsdetoklabel .....	396	
\showgloshort: added		
\glsdetoklabel .....	265	
\showgloshortaccess: added		
\glsdetoklabel .....	396	
\showgloshortpluralaccess: added		
\glsdetoklabel .....	397	
\showglosort: added \glsdetoklabel	265	
\showglosymbol: added		
\glsdetoklabel .....	265	
\showglosymbolaccess: added		
\glsdetoklabel .....	396	
\showglosymbolplural: added		
\glsdetoklabel .....	265	
\showglosymbolpluralaccess: added		
\glsdetoklabel .....	396	
\showglotext: added \glsdetoklabel	262	
\showglotextaccess: added		
\glsdetoklabel .....	396	
\showglotype: added \glsdetoklabel	263	
\showglouserii: added		
\glsdetoklabel .....	263	
\showglouserii: added		
\glsdetoklabel .....	263	
\showglouseriii: added		
\glsdetoklabel .....	264	
\showglouseriv: added		
\glsdetoklabel .....	264	
\showglouserv: added		
\glsdetoklabel .....	264	
\showglouservi: added		
\glsdetoklabel .....	264	
dua: fixed bug in \acrfullfmt .....	239	
fixed bug in \Acrfullplfmt .....	239	
fixed bug in \acrfullplfmt .....	239	
redefined to use accessibility information .....	384	
dua-desc: commented spurious EOL ..	240	
4.04 (2014-03-04)		
\@gls@getcounterprefix: added warning if no prefix can be formed ..	194	
4.04 (2014-03-06)		
\@gls@noidx@nosanitizesort: new ..	23	
\@gls@noidx@sanitizesort: new ..	23	
\@gls@nosanitizesort: new .....	23	
\@gls@sanitizesort: new .....	23	
\@glo@addchildren: new .....	202	
\@glo@do@sortentries: new .....	203	
\@glo@grabfirst: new .....	208	
\@glo@sortedinsert: new .....	204	
\@glo@sortentries: new .....	202	
\@glo@sorthandler@case: new .....	204	
\@glo@sorthandler@letter: new ..	204	
\@glo@sorthandler@nocase: new ..	204	
\@glo@sorthandler@word: new .....	204	
\@glo@sortmacro@case: new .....	206	
\@glo@sortmacro@def: new .....	206	
\@glo@sortmacro@def@do: new .....	207	
\@glo@sortmacro@letter: new .....	205	
\@glo@sortmacro@nocase: new .....	206	
\@glo@sortmacro@standard: new ..	205	
\@glo@sortmacro@use: new .....	207	
\@glo@sortmacro@word: new .....	205	
\@gls@noidx@do: new .....	209	
\@gls@noidx@getgrouptitle: new ..	220	
\@gls@noref@warn: new .....	186	
\@gls@reference: new .....	211	
\@gls@warnonglossdefined: new ..	21	
\@gls@warnonthe glossdefined: new ..	21	
\@no@makeglossaries: new .....	185	
\@print@glossary: new .....	200	
\@print@noidx@glossary: new .....	207	
\@printgloss@setsort: new .....	198	
\@printglossary: new .....	199	
General: added sort key to printgloss group .....	213	
\compatibleglossentry: changed		
\newcommand to \def as is may or may not be defined .....	349	
\compatiblesubglossentry: changed		
\newcommand to \def as is may or may not be defined .....	350	
\defglsdisplayfirst: fixed unwanted space .....	114	
\glo@grabfirst: new .....	208	

\gls@defglossaryentry: replaced \ifx	4.08 (2014-07-30)
with \ifdefvoid .....	92
\glsnoidxdisplayloc: new .....	211
\glsnoidxdisplayloclisthandler:	
new .....	210
\glsnoidxloclist: new .....	210
\glsnoidxloclisthandler: new .....	210
\glsnoidxstripaccents: new .....	24
alttree: moved hangindent and	
parindent assignments outside level	
test .....	326
\makeglossaries: Moved definition of	
\glswrite to \makeglossaries ..	180
\makenoidxglossaries: new .....	182
\printglossary: changed to use new	
\@printglossary .....	198
\printnoidxglossaries: new .....	198
\printnoidxglossary: new .....	198
\showgloclist: new .....	266
\warn@noprintglossary: Activate	
warning in \makeglossaries ....	197
\writeist: checked for definition of	
\glswrite .....	167, 171
4.06 (2014-03-12)	
\@GLS@: added \glsifhyper .....	131
\@GLSpl: added \glsifhyper .....	133
\@Gls@: added \glsifhyper .....	130
\@Glspl@: added \glsifhyper .....	132
\@gls@: added \glsifhyper .....	129
\@gls@numbersdef: added hook to set	
toc title .....	36
\@gls@symbolsdef: added hook to set	
toc title .....	36
\@glsdisp: added \glsifhyper .....	134
\@glspl@: added \glsifhyper .....	132
General: added \glsifhyper ....	147–154
acronym: added hook to set toc title ....	18
acronyms: added hook to set toc title ...	18
\glsdefmain: added hook to set toc title	17
4.07 (2014-04-04)	
\@glossarysection: added optional	
argument when using unstarred	
version .....	47
\@gls@noidx@do: added \global in case	
it's used in a tabular-like style ....	209
\Acrfullplfmt: fixed no case change	
bug .....	229
\glsletentryfield: new .....	155
\@ACRlong: added	
\do@gls@link@checkfirsthyper .....	374
\@ACRshort: added	
\do@gls@link@checkfirsthyper .....	373
\@Acrlong: added	
\do@gls@link@checkfirsthyper .....	373
\@Acrshort: added	
\do@gls@link@checkfirsthyper .....	372
\@GLS@: moved \glsifhyper .....	131
moved check for first use to	
\@gls@link .....	131
\@GLSpl: moved \glsifhyper .....	133
moved check for first use to	
\@gls@link .....	133
\@Gls@: moved \glsifhyper .....	130
moved check for first use to	
\@gls@link .....	130
\@Glspl@: moved \glsifhyper .....	132
moved check for first use to	
\@gls@link .....	132
\@acrlong: added	
\do@gls@link@checkfirsthyper .....	373
\@acrshort: added	
\do@gls@link@checkfirsthyper .....	372
\@closegls: new .....	177
\@gls@: moved \glsifhyper .....	129
moved check for first use to	
\@gls@link .....	129
\@gls@automake: new .....	177
\@gls@doautomake: new .....	33
\@gls@field@link: added assignment	
of	
\do@gls@link@checkfirsthyper .....	134
\@gls@forbidtexext: new .....	65
\@gls@hyp@opt: new .....	115
\@gls@link: removed redundancy ....	118
renamed \gls@type to \glstype ...	117
\@gls@link@checkfirsthyper: new ..	116
\@glsdisp: moved \glsifhyper .....	134
moved check for first use to	
\@gls@link .....	134
\@glspl@: moved \glsifhyper .....	132
moved check for first use to	
\@gls@link .....	132
\@ignored@glossaries: new .....	69
General: added entrycounter option to	
printgloss family .....	213

added nopostdot option to	
printgloss family .....	212
added subentrycounter option to	
printgloss family .....	213
explicitly initialise hyper key .....	115
moved \glsifhyper .....	147–154
removed \@sACRlongpl .....	154
removed \@sAcrlongpl .....	153
removed \@sacrlongpl .....	153
removed \@sACRlong .....	152
removed \@sAcrlong .....	151
removed \@sacrlong .....	151
removed \@sACRshortpl .....	150
removed \@sAcrshortpl .....	149
removed \@sacrshortpl .....	149
removed \@sACRshort .....	148
removed \@sAcrshort .....	148
removed \@sacrshort .....	147
removed \@sgls@link .....	116
removed \@sGLSdescplural .....	140
removed \@sGlsdescplural .....	140
removed \@sGLSdesc .....	139
removed \@sGlsdesc .....	139
removed \@sglsdesc .....	139
removed \@sglsdisp .....	134
removed \@sGLSfirstplural .....	138
removed \@sGlsfirstplural .....	137
removed \@sGlsfirstplural .....	137
removed \@sGLSfirst .....	136
removed \@sGlsfirst .....	136
removed \@sglsfirst .....	136
removed \@sGLSname .....	139
removed \@sGlsname .....	138
removed \@sglsname .....	138
removed \@sGLSplural .....	137
removed \@sGlsplural .....	137
removed \@sglsplural .....	136
removed \@sGLSpl .....	133
removed \@sGlspl .....	132
removed \@sglspl .....	131
removed \@sGLSsymbolplural .....	142
removed \@sGlssymbolplural .....	142
removed \@sglssymbolplural .....	141
removed \@sGLSsymbol .....	141
removed \@sGlssymbol .....	141
removed \@sglssymbol .....	140
removed \@sGLStext .....	135
removed \@sGlstext .....	135
removed \@sglstext .....	135
removed \@sGLSuseriii .....	144
removed \@sGlsuseriii .....	144
removed \@sglsuseriii .....	144
removed \@sGLSuserii .....	143
removed \@sGlsuserii .....	143
removed \@sglsuserii .....	143
removed \@sGLSuseriv .....	145
removed \@sGlsuseriv .....	145
removed \@sglsuseriv .....	145
removed \@sGLSuseri .....	143
removed \@sGlsuseri .....	142
removed \@sglsuseri .....	142
removed \@sGLSuservi .....	147
removed \@sGlsuservi .....	146
removed \@sglsuservi .....	146
removed \@sGLSuserv .....	146
removed \@sglsuserv .....	145
removed \@sGLS .....	130
removed \@sGls .....	130
removed \@sgls .....	129
removed \@thirdofthree (defined in kernel) .....	128
removed sPGLS .....	274
removed sPgl .....	273
removed spgl .....	271
removed sPGLSpl .....	275
removed sPglspl .....	273
removed spglspl .....	272
\ACRfull: removed \s@ACRfull .....	228
switched to using \@gls@hyp@opt ..	228
\Acrfull: removed \@sAcrfull .....	227
switched to using \@gls@hyp@opt ..	227
\acrfull: removed \@sacrfull .....	227
switched to using \@gls@hyp@opt ..	227
\ACRfullpl: removed \s@ACRfullpl .....	229
switched to using \@gls@hyp@opt ..	229
\Acrfullpl: removed \s@Acrfullpl .....	229
switched to using \@gls@hyp@opt ..	229
\acrfullpl: removed \s@acrfullpl .....	228
switched to using \@gls@hyp@opt ..	228
\ACRlong: switched to using \@gls@hyp@opt .....	152
\Acrlong: switched to using \@gls@hyp@opt .....	151
\acrlong: switched to using \@gls@hyp@opt .....	150

\ACRlongpl: switched to using	
\@gls@hyp@opt .....	154
\Acrlongpl: switched to using	
\@gls@hyp@opt .....	153
\acrlongpl: switched to using	
\@gls@hyp@opt .....	152
\ACRshort: switched to using	
\@gls@hyp@opt .....	148
\Acrshort: switched to using	
\@gls@hyp@opt .....	147
\acrshort: switched to using	
\@gls@hyp@opt .....	147
\ACRshortpl: switched to using	
\@gls@hyp@opt .....	150
\Acrshortpl: switched to using	
\@gls@hyp@opt .....	149
\acrshortpl: switched to using	
\@gls@hyp@opt .....	149
\forallacronyms: new	58
\GLS: switched to using \@gls@hyp@opt	130
\Gls: switched to using \@gls@hyp@opt	130
\gls: switched to using \@gls@hyp@opt	129
\gls@defglossaryentry: added check	
for ignored glossary .....	89
\gls@istfilebase: new	43
\glsaddkey: removed	
\@sGLS@user@{ <i>key</i> } .....	83
removed \@sGls@user@{ <i>key</i> } .....	82
removed \@sgls@user@{ <i>key</i> } .....	82
switched to using \@gls@hyp@opt	82, 83
\GLSdesc: switched to using	
\@gls@hyp@opt .....	139
\Glsdesc: switched to using	
\@gls@hyp@opt .....	139
\glsdesc: switched to using	
\@gls@hyp@opt .....	139
\GLSdescplural: switched to using	
\@gls@hyp@opt .....	140
\Glsdescplural: switched to using	
\@gls@hyp@opt .....	140
\glsdescplural: switched to using	
\@gls@hyp@opt .....	140
\glsdisablehyper: added	
\KV@glslink@hyperfalse to	
definition .....	128
\glsdisp: switched to using	
\@gls@hyp@opt .....	134
\glsdohyperlink: new	127
\glsdohypertarget: new	127
\glsenablehyper: added	
\KV@glslink@hypertrue to	
definition .....	128
\GLSfirst: switched to using	
\@gls@hyp@opt .....	136
\Glsfirst: switched to using	
\@gls@hyp@opt .....	136
\glsfirst: switched to using	
\@gls@hyp@opt .....	135
\GLSfirstplural: switched to using	
\@gls@hyp@opt .....	138
\Glsfirstplural: switched to using	
\@gls@hyp@opt .....	137
\glsfirstplural: switched to using	
\@gls@hyp@opt .....	137
\glsifhyper: deprecated	115
\glslink: switched to using	
\@gls@hyp@opt .....	116
\glslinkcheckfirsthyperhook: new	117
\glslinkvar: new	115
\GLSname: switched to using	
\@gls@hyp@opt .....	138
\Glsname: switched to using	
\@gls@hyp@opt .....	138
\glsname: switched to using	
\@gls@hyp@opt .....	138
\GLSpl: switched to using	
\@gls@hyp@opt .....	133
\Glspl: switched to using	
\@gls@hyp@opt .....	132
\glspl: switched to using	
\@gls@hyp@opt .....	131
\GLSplural: switched to using	
\@gls@hyp@opt .....	137
\Gsplural: switched to using	
\@gls@hyp@opt .....	137
\glsplural: switched to using	
\@gls@hyp@opt .....	136
\glsspace: new	227
\GLSsymbol: switched to using	
\@gls@hyp@opt .....	141
\Gssymbol: switched to using	
\@gls@hyp@opt .....	141
\gssymbol: switched to using	
\@gls@hyp@opt .....	140
\GLSsymbolplural: switched to using	
\@gls@hyp@opt .....	142
\Gssymbolplural: switched to using	
\@gls@hyp@opt .....	141

\glssymbolplural: switched to using	
\@gls@hyp@opt .....	141
\GLStext: switched to using	
\@gls@hyp@opt .....	135
\Glstext: switched to using	
\@gls@hyp@opt .....	135
\glstext: switched to using	
\@gls@hyp@opt .....	135
\glstreenamefmt: new	319
\GLSuseri: switched to using	
\@gls@hyp@opt .....	143
\Glsuseri: switched to using	
\@gls@hyp@opt .....	142
\glsuseri: switched to using	
\@gls@hyp@opt .....	142
\GLSuserii: switched to using	
\@gls@hyp@opt .....	143
\Glsuserii: switched to using	
\@gls@hyp@opt .....	143
\Glsuserii: switched to using	
\@gls@hyp@opt .....	143
\GLSuseriii: switched to using	
\@gls@hyp@opt .....	144
\Glsuseriii: switched to using	
\@gls@hyp@opt .....	144
\GLSuseriv: switched to using	
\@gls@hyp@opt .....	145
\Glsuseriv: switched to using	
\@gls@hyp@opt .....	145
\glsuseriv: switched to using	
\@gls@hyp@opt .....	145
\GLSuserv: switched to using	
\@gls@hyp@opt .....	146
\Glsuserv: switched to using	
\@gls@hyp@opt .....	146
\glsuserv: switched to using	
\@gls@hyp@opt .....	145
\GLSuservi: switched to using	
\@gls@hyp@opt .....	147
\Glsuservi: switched to using	
\@gls@hyp@opt .....	146
\glsuservi: switched to using	
\@gls@hyp@opt .....	146
\ifignoredglossary: new	69
altnongragged4col: fixed bug that	
displayed description instead of	
symbol .....	299
\newglossary: added starred version	.. 66
\newignoredglossary: new	.. 68
\ns@newglossary: added	
\@gloctype@ <i>(name)</i> @log .....	67
new .....	66
\p@gls@hyp@opt: new	115
\PGLS: changed to use \@gls@hyp@opt	274
\Pgls: changed to use \@gls@hyp@opt	273
\pgls: changed to use \@gls@hyp@opt	271
\PGLSpl: changed to use	
\@gls@hyp@opt .....	275
\PglSpl: changed to use	
\@gls@hyp@opt .....	273
\pglSpl: changed to use	
\@gls@hyp@opt .....	272
\s@gls@hyp@opt: new	115
\s@newglossary: new	.. 66
automake: new	.. 33
4.09 (2014-08-12)	
\glsaddkey: fixed bug in user commands	82
4.10 (2014-08-27)	
\@Gls@acrentryname: new	.. 156
\@Gls@entryname: new	.. 156
\@gls@glossary: Renamed \glossary	
to \@gls@glossary .....	187
\glspercentchar: new	.. 166
\glistildechar: new	.. 166
alttree: moved space after symbol	326, 327
4.11 (2014-09-01)	
\@odo@esc@wrglossary: added hook	.. 193
sanitize: none option	.. 26
\gls@wrglossary: renamed from	
\@wrglossary to \gls@wrglossary	188
\glsaddprotectedpagefmt: new	.. 189
\glsbackslash: new	.. 166
4.12 (2014-11-22)	
\@gls@addpredefinedattributes:	
Added gsignore attribute .....	52
\@gls@adjustmode: new	.. 164
\@gls@notranslatorhook: removed	.. 27
\@gls@toc: added \protect to	
\numberline .....	49
\@gls@usetranslator: new	.. 27
\glsacrpluralsuffix: new	.. 40
\glsadd: added check for vertical mode	164
\glsaddallunused: replaced @gobble	
with gsignore .....	165
\glsifusedtranslatordict: new	.. 27
\glsignore: new	.. 165

\glsupacrpluralsuffix: new .....	40	4.16 (2015-06-18)	
\ProvidesGlossariesLang: new .....	40	\glsaddstoragekey: new .....	80
\RequireGlossariesLang: new .....	40	4.16 (2015-07-08)	
4.13 (2015-02-03)		\@ACRlong: added \glspostlinkhook	374
\indexspace: new .....	281, 301, 319	\@ACRshort: added \glspostlinkhook	373
4.14 (2015-02-28)		\@Acrlong: added \glspostlinkhook	374
\@glslocalreset: new .....	97	\@Acrshort: added \glspostlinkhook	373
\@glslocalunset: new .....	97	\@GLS@: added \glspostlinkhook ...	131
\@glsreset: new .....	97	\@GLSpl: added \glspostlinkhook ...	133
\@glsunset: new .....	97	\@Gls@: added \glspostlinkhook ...	130
\@newglossaryentry@defcounters:		\@Glspl@: added \glspostlinkhook .	133
new .....	98	\@acrlong: added \glspostlinkhook	373
\cGls: new .....	102	\@acrshort: added \glspostlinkhook	372
\cGls@: new .....	102	\@gls@: added \glspostlinkhook ...	129
\cGlspl@: new .....	103	\@gls@link: added	
\cgls: new .....	101	\glspostlinkhook .....	116
\cgls@: new .....	101	\@gls@field@link: added	
\cgmentspl: new .....	102, 103	\glspostlinkhook .....	135
\cgmentspl@: new .....	102	\@gls@link: moved definition of	
\@gls@entry@count: new .....	101	\glsifhyperon outside of this	
\@gls@increment@currcount: new ..	100	macro .....	118
\@gls@local@increment@currcount:		\@glsdisp: added \glspostlinkhook	134
new .....	101	\@glspl@: added \glspostlinkhook .	132
\@gls@write@entrycounts: new ..	101	General: added \glspostlinkhook	147-154
\@glslocalreset: new .....	97	\glsacspace: new .....	234
\@glslocalunset: new .....	97	\glsadd: changed \@do@wrglossary to	
\@glsreset: new .....	97	\@do@wrglossary .....	164
\@glsunset: new .....	97	\glsfielddef: new .....	84
\@newglossaryentry@defcounters:		\glsfieldedef: new .....	83
new .....	93	\glsfieldfetch: new .....	85
\cGls: new .....	102	\glsfieldgdef: new .....	84
\cgls: new .....	101	\glsfieldxdef: new .....	83
\cGlsformat: new .....	102	\glsifhyperon: moved definition of	
\cgmentsformat: new .....	101	\glsifhyperon .....	117
\cGlspl: new .....	103	\glslinkpostsetkeys: new .....	117
\cgmentspl: new .....	102	\glspostlinkhook: new .....	116
\cgmentsplformat: new .....	103	\glswriteentry: new .....	189
\cgmentsplformat: new .....	102	\ifglsfieldcseq: new .....	86
\gls@defdocnewglossaryentry: new ..	77	\ifglsfielddefeq: new .....	85
\glsenableentrycount: new .....	99	\ifglsfieldeq: new .....	85
\glslocalreset: switched to		long-sp-short: new .....	233
\glslocalreset .....	96	\showglofield: new .....	266
\glslocalunset: switched to		4.18 (2015-09-09)	
\glslocalunset .....	96	General: split mfistruc into separate	
\glsreset: switched to \@glsreset ..	96	bundle .....	4
\glsunset: switched to \@glsunset ..	96	4.19 (2015-10-31)	
4.15 (2015-03-16)		\glstreenamebox: new .....	325
General: bug fix replaced \@glo@type		4.19 (2015-11-22)	
with \glstype .....	154	\@gls@link@nocheckfirsthyper: new	134

\@gls@preglossaryhook: new .....	198	\glsfindwidesttoplevelname: new ..	325
\@printglossary: added		\glslistgroupheaderfmt: new .....	281
\@gls@preglossaryhook .....	200	\glslistnavigationitem: new .....	281
\do@glsdisablehyperinlist: new ..	117	\glistreegroupheaderfmt: new .....	319
\doifglossarynoexistsordo: new ..	61	\glistreenavigationfmt: new .....	319
\gls@gobbleopt: new .....	66	\ifglswallowprimitivemods: new ..	191
\glsdoifexistsordo: new .....	61	list: fixed missing space before	
4.20 (2015-11-30)		description .....	281
\@gls@link: added		long: fixed typo in \glossentrydesc ..	285
\@gls@setdefault@glslink@opts	118	super4col: fixed bug in \glossentry ..	310
added \glsdonohyperlink when			
hyperlink is suppressed .....	118		
\@gls@setdefault@glslink@opts:			
new .....	117		
\gls@checkseeallowed@preambleonly:			
new .....	72		
\glsdonohyperlink: new .....	127		
4.21 (2016-01-24)			
\@printglossary: warn if no style has			
been set .....	199		
General: changed checkfirsthyper			
assignment .....	147–154		
\glossarystyle: set default style if not			
already set .....	222		
\glsLTpenaltycheck: new .....	294		
\glspatchLToutput: new .....	294		
\glspenaltygroupskip: new .....	294		
altnlong4col-booktabs: new .....	292		
altnlongragged4col-booktabs: new ..	293		
long-booktabs: new .....	291		
long3col-booktabs: new .....	291		
long4col-booktabs: new .....	292		
longragged-booktabs: new .....	293		
longragged3col-booktabs: new ..	293		
\setglossarystyle: set default style if			
not already set .....	221		
4.22 (2016-04-19)			
\@@do@esc@wrglossary: added check			
for \@arabic .....	192		
added test to allow temporary primitive			
modifications and added arabic case	192		
mcolalttreespannav: new .....	306		
mcolindexspannav: new .....	302		
mcoltreenamespannav: new .....	304		
mcoltreespannav: new .....	303		
\gls@arabicpage: new .....	189		
\gls@protected@pagefmcts: added			
arabic to list .....	189		
\glsentrytitlecase: new .....	159		
4.23 (2016-04-30)			
\glscurrentfieldvalue: new .....	65		
\ifglshasfield: added			
\glscurrentfieldvalue .....	64		
altnlongragged4col: check for			
nogroupskip changed .....	299		
altsuperragged4col: check for			
nogroupskip changed .....	318		
long: check for nogroupskip changed ..	285		
long-booktabs: check for nogroupskip			
changed .....	291		
long3col: check for nogroupskip			
changed .....	287		
long3col-booktabs: check for			
nogroupskip changed .....	292		
long4col: check for nogroupskip			
changed .....	288		
long4col-booktabs: check for			
nogroupskip changed .....	292		
longragged: check for nogroupskip			
changed .....	296		
longragged3col: check for nogroupskip			
changed .....	297		
super: check for nogroupskip changed ..	307		
super3col: check for nogroupskip			
changed .....	309		
super4col: check for nogroupskip			
changed .....	311		
superragged: check for nogroupskip			
changed .....	314		
superragged3col: check for			
nogroupskip changed .....	316		
4.24 (2016-05-27)			
\@gls@extramakeindexopts: new ...	176		
\@gls@glossary: added check for debug			
mode .....	187		
\@gls@see@noindex: new .....	7		
debug: new .....	5		
seenoindex: new .....	7		

\glsnomakeindexwarning: new .....	49	\gls@set@xr@key: new .....	71
\GlsSetQuote: new .....	173	\gls@xr@key: new .....	71
\GlsSetWriteIstHook: new .....	173	\GlsAddXdyLocation: bug fix: changed #1 to #2 .....	55
4.25 (2016-06-09)		\glsnoidxstripaccents: added \a ... added \TH, \dh and \DH .....	24 25
\@gls@enablesavenonumberlist: new	72		
\@gls@initnonumberlist: new .....	72		
\@gls@savenonumberlist: new .....	72		
4.25 (???)		4.31 (2017-08-10)	
General: changed		nolist: added check for “list” style .....	11
\DeclareRobustCommand to			
\newrobustcmd and changed		4.31 (2017-09-10)	
\@ifundefined to \ifcsundef ...	362	style: changed \renewcommand to \def .	9
4.26 (2016-10-12)		4.32 (2017-08-24)	
\@glossary@default@style: added		\@glsnavhypertarget: new .....	276
check for classictthesis .....	9	\@glsshowtarget: new .....	7
mcolindex: replaced \idxitem with		\glsshowtarget: new .....	6
\glstreeitem .....	301		
mcolindexspannav: replaced \idxitem		4.33 (2017-09-20)	
with \glstreeitem .....	302	\@odo@esc@wrglossary: added	
\glstreechildpredesc: new .....	320	\gls@the and \gls@number .....	192
\glstreeitem: new .....	319	renamed from	
\glstreepredesc: new .....	320	\@odo@esc@wrglossary .....	191
\glstreesubitem: new .....	320	\@odo@noesc@wrglossary: new .....	190
\glstreesubsubitem: new .....	320	\@odo@wrglossary: changed to check	
4.28 (2017-01-07)		for esclocations .....	190
\glspatchtabularx: new .....	95	\@gls@missinglang@warn: new .....	21
4.29 (2017-01-19)		\GlsSetXdyFirstLetterAfterDigits:	
\@gls@noidx@do: current letter group		added starred version .....	166
assignment made global .....	209	\GlsSetXdyNumberGroupOrder: new ..	166
\@print@noidx@glossary: moved		esclocations: new .....	10
definition of			
\@gls@currentlettergroup outside		4.34 (2017-11-03)	
of the glossary environment .....	207	mcolalttreespannav: removed spurious	
General: added check for		space .....	306
\@glsxtr@doacccsupp .....	349	\glsshowtarget: modified to check for	
\glsnavhyperlinkname: new .....	276	math mode and inner .....	6
4.30 (2017-06-11)		4.35 (2017-11-14)	
\@glo@autosee: new .....	92	\glsadd: added \gls@setsort (in case	
\@glo@autoseehook: new .....	93	of sort=use) .....	164
\@glo@check@sortallowed: new .....	14	4.36 (2018-03-07)	
\@gls@noidx@do: letter group		\@gls@glossary: removed \index ...	188
assignment made global .....	209	4.37 (2018-04-07)	
\@gls@setupsort@def: added check for		\gls@begindocdefs: new .....	78
register .....	15	4.38 (2018-05-10)	
\@gls@setupsort@none: new .....	16	\@gls@define@glossaryentrycounter:	
\@xdycrossrefhook: new .....	54	added check for existence of	
\@xdylocationclassorder: bug fix:		glossaryentry counter .....	12
changed \edef to \def .....	55	new .....	12
\glosortentrieswarning: new .....	21	\@gls@define@glossarysubentrycounter:	
		new .....	13
		prepended \currentglossary. to	
		\theHglossarysubentry and	
		removed spurious eol space .....	13

\glsaccsupp: added braces around actual text argument .....	359	numberedsection: changed \val and \nr to \gls@numberedsection@val and \gls@numberedsection@nr .....	9
\glsentrycounterlabel: bug fix: move conditional inside command .....	215	4.42 (2019-01-06)	
\GlsEntryCounterLabelPrefix: new .....	212	\@gls@automake@immediate: new ..	179
\glsentryitem: bug fix: move conditional inside command .....	215	\@gls@automake@immediate: new ..	178
\glsrefentry: bug fix: move conditional inside command .....	215	\gls@automake@nr: new .....	33
\glsresetsubentrycounter: bug fix: move conditional inside command .....	214	\glsfielddef: changed from \edef to \protected@csedef .....	83
\glsstepentry: bug fix: move conditional inside command .....	214	\glsfieldxdef: changed from \edef to \protected@csxdef .....	83
\glsstepsubentry: bug fix: move conditional inside command .....	215	\ifglsautomake: now defined explicitly instead of through boolean key .....	32
\glssubentrycounterlabel: bug fix: move conditional inside command .....	215	noglossaryindex: new .....	36
\glssubentryitem: bug fix: move conditional inside command .....	215	automake: switch from boolean to choice	33
\showglonameaccess: bug fix: corrected field (was showing text access field) .....	395	4.42 (??)	
4.40 (2018-06-01)		altnlong4col-booktabs: removed superfluous \glspatchLToutput ..	292
\istfile: changed \def to \providecommand .....	186	4.43 (2019-09-28)	
\makenoidxglossaries: false .....	182	\glsnoidxstripaccents: add check for LaTeX version 2019/10/01 .....	25
4.41 (2018-07-23)		4.44 (2019-12-06)	
@\gls@Override@glossary: new .....	34	\@glsprefix@record@hook: new .....	271
General: changed \val and \nr to \gls@numberedsection@val and \gls@numberedsection@nr .....	212	4.45 (2020-02-13)	
debug: changed \val and \nr to \gls@debug@val and \gls@debug@nr .....	5	\@odo@write@glslabels: new .....	32
seenoindex: changed \val and \nr to \gls@seenoindex@val and \gls@seenoindex@nr .....	7	\@glsshowtarget: new .....	7
kernelglossredefs: new .....	35	\@GLSdesc@: added accessibility support	376
\glossary: added warning .....	35	\@GLSdescplural@: added accessibility support .....	377
\gls@original@glossary: new .....	34	\@GLSfirst@: added accessibility support .....	375
\gls@original@makeglossary: new .....	34	\@GLSfirstplural@: added accessibility support .....	375
\makeglossaries: removed redefinition of \makeglossary .....	181	\@GLSname@: added accessibility support	376
\makeglossary: added warning .....	34	\@GLSplural@: added accessibility support .....	375
nonumberlist: changed \val and \nr to \gls@nonumberlist@val and \gls@nonumberlist@nr .....	72	\@GLSsymbol@: added accessibility support .....	377
translate: changed \val and \nr to \gls@translate@val and \gls@translate@nr .....	28	\@GLSsymbolplural@: added accessibility support .....	377
		\@GLStext@: added accessibility support	374
		\@GLSuseri@: added accessibility support .....	378
		\@GLSuserii@: added accessibility support .....	378
		\@GLSuseriii@: added accessibility support .....	378
		\@GLSuseriv@: added accessibility support .....	379

\@GLSuser@: added accessibility support	379
\@GLSuseri@: added accessibility support	379
\@Gls@acronymname: added check for \glsshortaccessdisplay	156
\@Glsdesc@: added accessibility support	376
\@Glsdescplural@: added accessibility support	376
\@Glsfirst@: added accessibility support	375
\@Glsfirstplural@: added accessibility support	375
\@Glsname@: added accessibility support	376
\@Glsplural@: added accessibility support	375
\@Glssymbol@: added accessibility support	377
\@Glssymbolplural@: added accessibility support	377
\@Glstext@: added accessibility support	374
\@Glsuseri@: added accessibility support	378
\@Glsuserii@: added accessibility support	378
\@Glsuseriii@: added accessibility support	378
\@Glsuseriv@: added accessibility support	379
\@Glsuserv@: added accessibility support	379
\@Glsuservi@: added accessibility support	379
General: removed backward compatibility use of symbol key	353
acronyms: changed \renewcommand to \def	18
debug: showaccsupp	5
restoremakergloss: new	32
\gls@accessibility: new	350
\gls@accsupp@engine: new	350
\glsaccessibility: new	350
\glsdefaultshortaccess: new	390
\glsdescriptionaccessdisplay: added check for existence	360
\glsdescriptionpluralaccessdisplay: added check for existence	360
\glsentrydescpluralaccess: corrected field reference	356
\glsentryfirstpluralaccess: switched to using \gls@entry@field	356
\glsentryparent: new	159
\Glsentryprefix: added \glsdetoklabel	270
\glsentryprefix: added \glsdetoklabel	270
\Glsentryprefixfirst: added \glsdetoklabel	270
\glsentryprefixfirst: added \glsdetoklabel	270
\Glsentryprefixfirstplural: added \glsdetoklabel	270
\glsentryprefixfirstplural: added \glsdetoklabel	270
\Glsentryprefixplural: added \glsdetoklabel	270

\glsentryprefixplural: added	
\glsdetoklabel .....	270
\glsentrytitlecase: added existence	
check .....	159
\glsentryuseriaccess: new	357
\glsentryuseriiaccess: new	357
\glsentryuseriiiaccess: new	357
\glsentryuserivaccess: new	357
\glsentryuserservaccess: new	357
\glsentryuserviaccess: new	357
\glsfieldaccsupp: new	358
\glsfirstaccessdisplay: added check	
for existence .....	360
\glsfirstpluralaccessdisplay:	
added check for existence .....	360
\glslongaccessdisplay: added check	
for existence .....	361
\glslongpluralaccessdisplay: added	
check for existence .....	361
\glsnameaccessdisplay: added check	
for existence .....	359
\glspluralaccessdisplay: added	
check for existence .....	360
\glsprefixsep: new	271
\glssee: switched to \newrobustcmd	195
\glsseeformat: switched to	
\newrobustcmd .....	196
\glsseeitem: switched to	
\newrobustcmd .....	196
\glsseelist: switched to	
\newrobustcmd .....	196
\glsshortaccessdisplay: added check	
for existence .....	361
\glsshortaccsupp: new	358
\glsshortplaccsupp: new	359
\glsshortpluralaccessdisplay:	
added check for existence .....	361
\glsshowaccsupp: new	7
\glsshowtargetfont: new	6
\glsshowtargetouter: new	6
\glsshowtargetsymbol: new	6
\glssymbolaccessdisplay: added	
check for existence .....	360
\glssymbolpluralaccessdisplay:	
added check for existence .....	360
\gstextaccessdisplay: added check	
for existence .....	359
\glsuseriaccessdisplay: new	361
\glsuseriiaccessdisplay: new	361
\glsuseriiiaccessdisplay: new	361
\glsuserivaccessdisplay: new	362
\glsuservaccessdisplay: new	362
\glsuserviaccessdisplay: new	362
\ifglshaschildren: made robust	61
\ifglshasfield: made robust	63
\ifglshaslong: made robust	63
\ifglshasprefix: added	
\glsdetoklabel .....	271
\ifglshasprefixfirst: added	
\glsdetoklabel .....	271
\ifglshasprefixfirstplural: added	
\glsdetoklabel .....	271
\ifglshasprefixplural: added	
\glsdetoklabel .....	271
\ifglshasshort: made robust	63
\ifglshassymbol: made robust	62
\disablemakegloss: new	31
\makeglossaries: let \makeglossary	
to \@gobble instead of \relax	181
\writeglslabels: new	32
\user1access: new	352
\user2access: new	352
\user3access: new	352
\user4access: new	353
\user5access: new	353
\user6access: new	353
\xglsfieldaccsupp: new	358
4.46 (2020-03-19)	
\@ifglossaryexists: new	59
\@printgloss@checkexists: new	198
\@printgloss@checkexists@allowignored:	
new .....	198
\@printgloss@checkexists@noignored:	
new .....	197
\@printglossary: replaced	
\ifglossaryexists with	
\@printgloss@checkexists	199
\doifglossarynoexistsordo: switched	
to starred form of	
\ifglossaryexists .....	61
\ifglossaryexists: added starred form	59
\s@ifglossaryexists: new	59
\setglossarypreamble: switched to	
starred form of \ifglossaryexists	45

# 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	
\!	124
\"	24, 121–124, 126
\#	170
\%	166, 172, 333, 334
\&	40, 163
\'	24
\.	11, 24
\=	24
\?	121, 123, 174
\@	78
\@delimN	224
\@do@wrglossary	182, 191, 193
\@do@esc@wrglossary	190
\@do@noesc@wrglossary	190
\@do@wrglossary	164, 188
\@do@write@glslabels	32
\@glo@assign@sortkey	183
\@glo@list	58
\@glo@sort	24
\@glo@type	198
\@glossarysec	8, 47, 48
\@glossaryseclabel	9, 47, 48, 212
\@glossarysecstar	9, 47, 48, 212
\@gls@checkactual	125, 126
\@gls@checkbar	124, 125
\@gls@checkescactual	123
\@gls@checkescbar	123, 124
\@gls@checkesclevel	124
\@gls@checkescquote	122, 123, 175, 176
\@gls@checklevel	125
\@gls@checkquote	122, 174
\@gls@default@entryfmt	104, 113, 114
\@gls@expand@field	22, 76, 80, 81, 245, 247–249, 252, 254, 257–259, 391–395
\@gls@extramakeindexopts	173, 180
\@gls@fixbraces	195
\@gls@noexpand@field	22, 75
\@gls@noidx@no@sanitizesort	23, 24
\@gls@noidx@nosanitizesort	185
\@gls@nosanitizesort	23, 185
\@gls@sanitizesort	23, 185
\@gls@xdycheckbackslash	127
\@gls@xdycheckquote	126
\@glslocalreset	97, 99
\@glslocalunset	97, 99
\@glsreset	97, 99
\@glsshowtarget	6, 7
\@glsunset	97, 99
\@newglossaryentry@defcounters	99
\@this@glo@	59
\@ACRfull	228
\@ACRfullpl	229
\@ACRlong	152, 228
\@ACRlongpl	154, 229
\@ACRshort	148, 228
\@ACRshortpl	150, 229
\@Acrfull	227, 228
\@Acrfullpl	229
\@Acrlong	151, 228
\@Acrlongpl	153, 229
\@Acrshort	148
\@Acrshortpl	149
\@Alph	189, 192, 193
\@GLS	130
\@GLS@	130, 275
\@GLSdesc	139
\@GLSdesc@	139
\@GLSdescplural	140
\@GLSdescplural@	140
\@GLSfirst	136
\@GLSfirst@	136
\@GLSfirstplural	138
\@GLSfirstplural@	138

\@GLSname .....	138, 139	\@Glstext .....	135
\@GLSname@ .....	139	\@Glstext@ .....	135
\@GLSpl .....	133	\@Glsuser@i .....	378, 379
\@GLSpl@ .....	133, 275	\@Glsuseri .....	142
\@GLSplural .....	137	\@Glsuseri@ .....	142
\@GLSplural@ .....	137	\@Glsuserii .....	143
\@GLSsymbol .....	141	\@Glsuserii@ .....	143
\@GLSsymbol@ .....	141	\@Glsuseriii .....	144
\@GLSsymbolplural .....	142	\@Glsuseriii@ .....	144
\@GLSsymbolplural@ .....	142	\@Glsuseriv .....	145
\@GLStext .....	135	\@Glsuseriv@ .....	145
\@GLStext@ .....	135	\@Glsuserv .....	146
\@GLSuseri .....	143	\@Glsuserv@ .....	146
\@GLSuseri@ .....	143	\@Glsuservi .....	146
\@GLSuserii .....	143	\@Glsuservi@ .....	146, 147
\@GLSuserii@ .....	143, 144	\@Mi .....	294
\@GLSuseriii .....	144	\@PGLS .....	274
\@GLSuseriii@ .....	144	\@PGLS@ .....	274
\@GLSuseriv .....	145	\@PGLSpl .....	275
\@GLSuseriv@ .....	145	\@PGLSpl@ .....	275
\@GLSuserv .....	146	\@Pgls .....	273
\@GLSuserv@ .....	146	\@Pgls@ .....	273
\@GLSuservi .....	147	\@Pglspl .....	273
\@GLSuservi@ .....	147	\@Pglspl@ .....	273
\@Gls .....	130	\@Roman .....	189, 192, 193
\@Gls@ .....	100, 102, 130, 273	\@acrfull .....	227
\@Gls@crentryname .....	230	\@acrfullpl .....	228
\@Gls@entry@field .....	82, 156–162	\@acrlong .....	151, 227
\@Gls@entryname .....	156, 230	\@acrlongpl .....	153, 229
\@GlsSetXdyFirstLetterAfterDigits ..	166	\@acrshort .....	147, 227, 228
\@GlsSetXdyNumberGroupOrder ...	166, 167	\@acrshortpl .....	149, 229
\@Glsdesc .....	139	\@addtoacronymlists .....	18, 19
\@Glsdesc@ .....	139	\@afterheading .....	282, 337
\@Glsdescplural .....	140	\@alph .....	189, 192, 193
\@Glsdescplural@ .....	140	\@arabic .....	189, 192
\@Glsfirst .....	136	\@auxout .....	65, 67, 101, 180, 183, 186, 197, 201, 277
\@Glsfirst@ .....	136	\@backslashchar .....	120, 126, 127
\@Glsfirstplural .....	137	\@bsphack .....	188
\@Glsfirstplural@ .....	137, 138	\@cGls .....	102
\@Glsname .....	138	\@cGls@ .....	100, 102
\@Glsname@ .....	138	\@cGlspl .....	103
\@Glspl .....	132	\@cGlspl@ .....	100, 103
\@Glspl@ .....	100, 103, 132, 274	\@cclv .....	294, 295
\@Glsplural .....	137	\@cgls .....	101
\@Glsplural@ .....	137	\@cgls@ .....	99, 101
\@Glssymbol .....	141	\@cglspl .....	102
\@Glssymbol@ .....	141	\@cglspl@ .....	100, 102
\@Glssymbolplural .....	141, 142	\@chapter .....	38
\@Glssymbolplural@ .....	142	\@classoptionslist .....	37

\@closegls ..... 177, 178  
 \@colht ..... 294  
 \@colroom ..... 294, 295  
 \@currentlabelname ..... 9, 212  
 \@curroptions ..... 37  
 \@declaredoptions ..... 37  
 \@delimN ..... 223  
 \@delimR ..... 223  
 \@disable@onlypremakeg ..... 181  
 \@disable@premakecs ..... 39  
 \@disabled@glsaddxdycounters ..... 51  
 \@do@addcounter ..... 50  
 \@do@auxoutstuff ..... 201, 202  
 \@do@glossentry ..... 216, 217, 349, 350  
 \@do@gls@getcounterprefix ..... 191, 193  
 \@do@gls@islistofacronyms ..... 19  
 \@do@glssee ..... 92  
 \@do@ifinlist ..... 50  
 \@do@newglossaryentry ..... 231,  
     245, 247–249, 251–254, 256–261, 391–395  
 \@do@seeglossary ..... 183, 196  
 \@do@subglossentry ..... 218, 350  
 \@do@wrglossary ..... 118  
 \@do@write@glslabels ..... 32  
 \@do@writeaux@info ..... 197  
 \@domakeglossaries ..... 31, 32, 180, 182  
 \@ehc ..... 294  
 \@empty ..... 12, 16, 19, 33, 36, 37,  
     39, 50, 51, 54, 57, 58, 89, 94, 119, 129–  
     133, 147–154, 168, 170–173, 177–179,  
     186, 188, 194, 221, 246, 248, 250, 252–  
     255, 257, 259, 261, 331, 333, 335, 372–374  
 \@end@fixbraces ..... 195  
 \@endfortrue ..... 29, 62, 80, 277  
 \@esphack ..... 188  
 \@expandtwoargs ..... 37  
 \@firstofone ..... 24, 32  
 \@firstofthree ..... 115, 128,  
     129, 131, 134, 147, 149, 151, 153, 372–374  
 \@firstoftwo ..... 27,  
     28, 78, 80, 115, 131–133, 149, 150, 153, 154  
 \@for ..... 28, 37,  
     39, 50, 51, 58, 78, 80, 121, 168, 169, 180,  
     182, 189, 196, 202, 203, 232, 246, 248,  
     250, 252, 254, 257, 259, 261, 277, 278, 331  
 \@glo@desc ..... 91  
 \@glo@symbol ..... 91  
 \@glo@access ..... 351, 354, 359  
 \@glo@addchildren ..... 202, 207  
 \@glo@assign@sortkey ..... 181, 183, 213  
 \@glo@autosee ..... 92  
 \@glo@autoseehook ..... 92  
 \@glo@check@mkidxrangechar .....  
     120, 193, 330, 331  
 \@glo@check@sortallowed ..... 14–16, 182, 185  
 \@glo@childlist ..... 202, 203  
 \@glo@counter ..... 71, 88, 91  
 \@glo@counterprefix .....  
     186, 190, 193, 194, 221, 224  
 \@glo@default@sorttype ..... 13, 183, 205, 206  
 \@glo@defaultcounter ..... 91  
 \@glo@desc ..... 69, 86, 87, 89, 91  
 \@glo@descaccess ..... 352, 354, 355  
 \@glo@descplural ..... 70, 86, 87  
 \@glo@descpluralaccess ..... 352, 354, 355  
 \@glo@do@sortentries ..... 202  
 \@glo@entry ..... 165  
 \@glo@entryprefix ..... 269  
 \@glo@entryprefixfirst ..... 269  
 \@glo@entryprefixfirstplural ..... 269, 270  
 \@glo@entryprefixplural ..... 269  
 \@glo@esclabel ..... 94, 95  
 \@glo@etext ..... 105–107  
 \@glo@first ..... 70, 87, 90, 91, 256, 394  
 \@glo@firstaccess ..... 351, 354  
 \@glo@firstplural ..... 70, 87, 90, 395  
 \@glo@firstpluralaccess ..... 351, 354  
 \@glo@grabfirst ..... 208  
 \@glo@label .....  
     74, 81–92, 98, 163, 269, 270, 325, 354, 355  
 \@glo@list ..... 92  
 \@glo@long ..... 63, 74, 88, 91  
 \@glo@longaccess ..... 352, 354, 355  
 \@glo@longpl ..... 74,  
     88, 91, 245, 247, 249, 251, 254, 256, 258, 391  
 \@glo@longpluralaccess ..... 352, 354, 355  
 \@glo@name ..... 14, 69, 75, 87, 89–91  
 \@glo@no@assign@sortkey ..... 181  
 \@glo@nonumberlist ..... 72, 73  
 \@glo@numfmt ..... 194, 331  
 \@glo@parent ..... 16, 72, 88–90, 94, 95, 203  
 \@glo@plural ..... 70, 87, 90, 393  
 \@glo@pluralaccess ..... 351, 354  
 \@glo@prefix .....  
     10, 72, 88, 94, 95, 120, 193, 194, 330, 331  
 \@glo@orange ..... 193, 194, 330, 331  
 \@glo@see ..... 71, 88, 92  
 \@glo@seeautonumberlist ..... 10, 71

\glo@short ..... 63, 74, 88, 91, 394  
 \glo@shortaccess ... 352, 354, 355, 391–394  
 \glo@shortpl ..... 74, 88, 91,  
     245, 247, 249, 251, 253, 256, 258, 391, 394  
 \glo@shortpluralaccess .... 352, 354, 355  
 \glo@sort ... 14, 16, 23, 24, 70, 87, 90, 94, 95  
 \glo@sortedinsert ..... 203  
 \glo@sortentries ..... 205, 206  
 \glo@sorthandler@case ..... 206  
 \glo@sorthandler@letter ..... 205  
 \glo@sorthandler@nocase ..... 206  
 \glo@sorthandler@word ..... 205  
 \glo@sortinghandler ..... 202, 204  
 \glo@sortinglist ..... 202, 203, 206  
 \glo@sorttype ..... 183, 207, 208, 213  
 \glo@storeentry ..... 14–16  
 \glo@suffix ..... 120, 194, 331  
 \glo@symbol ..... 62, 71, 87, 91, 92, 251, 256  
 \glo@symbolaccess ..... 351, 354, 394  
 \glo@symbolplural ..... 71, 87, 92  
 \glo@symbolpluralaccess .... 351, 354  
 \glo@text ..... 70,  
     87, 90–92, 129–134, 155, 156, 251, 270, 393  
 \glo@textaccess ..... 351, 354, 391–393  
 \glo@thislabel ..... 93  
 \glo@thislettergrp ..... 208, 209  
 \glo@thisvalue ..... 63, 64  
 \glo@tmp ..... 81, 82, 194  
 \glo@type ..... 9, 16,  
     71, 87–92, 164, 165, 180, 186, 187, 199,  
     201–203, 207, 208, 211, 212, 230, 246,  
     248, 250, 252, 254, 257, 259, 261, 276, 278  
 \glo@types ..... 32,  
     58, 59, 66, 97, 98, 165, 180–182, 266, 325  
 \glo@useri ..... 73, 88, 91  
 \glo@useriaccess ..... 352, 354, 355  
 \glo@userii ..... 73, 88, 91  
 \glo@useriiaccess ..... 352, 354, 355  
 \glo@useriii ..... 73, 88, 91  
 \glo@useriiiaccess ..... 352, 354, 355  
 \glo@useriv ..... 73, 88, 91  
 \glo@userivaccess ..... 353–355  
 \glo@userserv ..... 73, 88, 91  
 \glo@usersvaccess ..... 353–355  
 \glo@usersvi ..... 73, 88, 91  
 \glo@usersviaccess ..... 353–355  
 \glo@glodesc ..... 91  
 \glo@glolist@ ..... 89  
 \glo@gloname ..... 91  
 \glossary@default@style .....  
     9, 11, 199, 221, 222, 262  
 \glossaryentryfield ..... 94, 95  
 \glossarysection ..... 46  
 \glossarystyle ..... 199, 200, 212  
 \glossarysubentryfield ..... 94, 95  
 \gls ..... 129  
 \gls@ ..... 100, 101, 129, 272, 273  
 \gls@@automake@immediate ..... 180  
 \gls@link ..... 116  
 \gls@Hcounter ..... 118, 119  
 \gls@returnAfterFi ..... 224  
 \gls@actualchar . 95, 123, 126, 171, 172, 334  
 \gls@addpredefinedattributes . 167, 176  
 \gls@adjustmode ..... 164  
 \gls@after ..... 19  
 \gls@automake ..... 182  
 \gls@automake@immediate ..... 180  
 \gls@before ..... 19  
 \gls@between ..... 278  
 \gls@body ..... 156  
 \gls@checkactual ..... 121, 174  
 \gls@checkbar ..... 121, 175  
 \gls@checkedmkidx ..... 120–127, 173–175  
 \gls@checkescactual ..... 121, 174  
 \gls@checkescbar ..... 121, 175  
 \gls@checkescquote ..... 121, 174–176  
 \gls@checklevel ..... 121, 175  
 \gls@checkmkidxchars .....  
     94, 120, 174, 182, 193, 195, 330, 331  
 \gls@checkquote ..... 121, 173, 174  
 \gls@classI ..... 168  
 \gls@classII ..... 168  
 \gls@codepage ..... 201  
 \gls@counter .....  
     114, 117–119, 164, 186, 194, 195, 331  
 \gls@counterwithin ..... 12, 13  
 \gls@ctr ..... 50  
 \gls@currentlettergroup ..... 207, 209  
 \gls@debugfalse ..... 5  
 \gls@debugtrue ..... 5, 6  
 \gls@declareoption .....  
     10, 11, 17, 18, 21, 22, 27, 30–32, 35, 36  
 \gls@default ..... 103  
 \gls@default@value .....  
     62–64, 75, 76, 87, 88, 90, 91, 255, 269  
 \gls@deffile ..... 77–79  
 \gls@define@glossaryentrycounter .  
     13, 38, 213, 214

\@gls@define@glossarysubentrycounter	50
.....	38, 213, 214
\@gls@defsort	95
.....	14–16, 91
\@gls@defsortcount	99
.....	14–16, 67
\@gls@do@acronymsdef	69
.....	17, 18, 38, 68
\@gls@do@indexdef	36
.....	36–38, 68
\@gls@do@numbersdef	19
.....	36, 38, 68
\@gls@do@symbolsdef	390
.....	35, 36, 68
\@gls@do@symbolssdef	333
.....	38
\@gls@doautomake	184
.....	33, 180, 182
\@gls@dochekquotedef	32
.....	173–176
\@gls@docloadedfalse	177–179
.....	4
\@gls@docloadedtrue	34
.....	4
\@gls@dodeflistparser	172
.....	181
\@gls@doentrycounterdef	129–134
.....	38
\@gls@doentrydef	134
.....	113, 114
\@gls@dolast	147–154
.....	196
\@gls@donext	147–154
.....	196
\@gls@donext@def	278
.....	163
\@gls@dosubentrycounterdef	50
.....	38
\@gls@dothiswrite	277
.....	177–179
\@gls@elem	116
.....	277
\@gls@enablesavenonumberlist	129–134
.....	78
\@gls@encapchar	134
....	123–125, 171, 172, 194, 195, 331, 334
\@gls@entry@count	100, 101
\@gls@entry@field	109
.....	81, 82, 99, 156–162, 355–357
\@gls@escbsdq	91
.....	121, 172, 335
\@gls@expand@fields	359
.....	76, 77
\@gls@expandonce	77
.....	76
\@gls@extramakeindexopts	208
.....	180
\@gls@fetchfield	208
.....	64
\@gls@field@link	208
..	82, 83, 135–147, 374–379
\@gls@fieldaccess@display	208
.....	359–362
\@gls@firstttok	208
.....	182
\@gls@fixbraces	208
.....	92
\@gls@forbidtexext	208
.....	67
\@gls@get@counterprefix	208
.....	194
\@gls@getbody	208
.....	156
\@gls@getcounterprefix	223, 224
.....	191, 193
\@gls@getgrouptitle	223, 224
.....	182, 220, 278
\@gls@glossary	223, 224
.....	187, 188
\@gls@gobbleopt	223, 224
.....	66
\@gls@grptitle	223, 224
.....	220, 276, 278
\@gls@hyp@opt	223, 224
.....	82, 83,
.....	101–103, 116, 129–154, 227–229, 271–275
\@gls@hyp@opt@cs	354
.....	115
\@gls@hypergroup	354
.....	277

\@gls@onlypremakeg .....	39	\@gls@updatechecked ....	120, 121, 174, 175
\@gls@order .....	177–179	\@gls@usetranslator .....	27, 28, 41
\@gls@org@LT@output .....	294	\@gls@value .....	75, 76, 159
\@gls@org@glsnoidxdisplayloc ..	184, 185	\@gls@warnonglossdefined .....	22, 197
\@gls@org@glsseeformat .....	184, 185	\@gls@warnonthe glossdefined .....	22, 216
\@gls@override@glossary .....	35	\@gls@write@entrycounts .....	100
\@gls@patchtabularx .....	95	\@gls@writedef .....	78
\@gls@preglossaryhook .....	200	\@gls@writeis hook .....	171, 173
\@gls@prevlevel .	305, 306, 325–328, 343, 344	\@gls@xdy@locationlist .....	168
\@gls@provide@newglossary .....	67	\@gls@xdycheckbackslash .....	120
\@gls@quotechar ....	122–126, 171–175, 334	\@gls@xdycheckquote .....	121
\@gls@reference .....	183, 186	\@gls@xref .....	195
\@gls@removespaces .....	224	\@glsAlphacompositor .....	44, 54, 332
\@gls@renewglossary .....	177	\@glsHlocref .....	190, 191, 193
\@gls@replacementtext .....	358, 359	\@glsacronymlists ...	19, 20, 58, 230, 232,
\@gls@rest .....	156	246, 248, 250, 252, 254, 257, 259, 261, 266	
\@gls@restoreat .....	78	\@glsaddkey .....	81
\@gls@roman .....	53, 331, 332	\@glsaddstoragekey .....	80
\@gls@sanitized@tmp .....	121	\@glsaddxdyattribute .....	51
\@gls@sanitizeddesc .....	29	\@glsdefaultsort .....	14
\@gls@sanitizesort .....	14	\@glsdesc .....	139
\@gls@sanitizesymbol .....	29, 30	\@glsdesc@ .....	139
\@gls@saveentrycounter .....	118, 164	\@glsdescplural .....	140
\@gls@savenonumberlist .....	72, 73	\@glsdescplural@ .....	140
\@gls@see@noindex .....	8, 72	\@glsdisp .....	134
\@gls@setacrstyle .....	29, 30, 37, 38	\@glsentry .....	32, 97, 98, 101
\@gls@setcounter .....	67	\@glsentrytitlecase .....	159
\@gls@setdefault@glslink@opts .....	118	\@glsfirst .....	135, 136
\@gls@setsort .....	14–16, 118, 164	\@glsfirst@ .....	136
\@gls@setupshortcuts .....	37, 38	\@glsfirstletter .....	166
\@gls@sort .....	209	\@glsfirstplural .....	137
\@gls@sort@A .....	204, 205	\@glsfirstplural@ .....	137, 375
\@gls@sort@B .....	204, 205	\@glshypernumber .....	223
\@gls@startswithxponce .....	76	\@glsisacronymlistfalse .....	19
\@gls@storenonumberlist .....	72, 73, 91	\@glsisacronymlisttrue .....	19
\@gls@symbolsdef .....	35	\@glslink .....	118, 128, 163, 276
\@gls@this .....	189	\@glslocalreset .....	96, 99
\@gls@thisHloc .....	194	\@glslocalunset .....	96, 99
\@gls@thisfield .....	64	\@glslocref .	186, 190, 191, 193, 194, 330, 331
\@gls@thislabel .....	61, 62, 196, 206	\@glsminrange .....	167, 168, 332
\@gls@thislist .....	163	\@glsname .....	138
\@gls@thisloc .....	194	\@glsname@ .....	138
\@gls@thisval .....	80	\@glsnavhypertarget .....	276
\@gls@title .....	46	\@glsnextpages .....	200
\@gls@tmp ..	7, 16, 41, 54, 76, 121, 188, 277, 278	\@glsnodec .....	87, 89, 91
\@gls@tmpb .....	122–127, 173–175	\@glsnoname .....	87, 89, 91
\@gls@toc .....	47, 48	\@glsnonextpages .....	200
\@gls@type .....	180, 182, 232,	\@glsnumberformat .....	
	246, 248, 250, 252, 254, 257, 259, 261, 325	114, 117, 164, 186, 193, 194, 330, 331	

\@glsopenfile .....	177, 187	\@input@ .....	201
\@glsorder .....	180	\@istfilename .....	180
\@glsp{ .....	131	\@makecol .....	294, 295
\@glsp{@ .....	100, 102, 131, 272, 274	\@makeglossary .....	31, 32, 180, 181
\@glsplural .....	136	\@minus .....	281, 301, 319
\@glsplural@ .....	136	\@mkboth .....	47
\@glsprefix@record@hook .....	272–275	\@newglossary .....	65, 67
\@glsreset .....	96, 99	\@newglossaryentry@defcounters ..	92, 99
\@glssee .....	92, 196	\@newglossaryentryposthook ..	81, 92, 269, 354
\@glsshowaccsupp .....	5, 6, 351	\@newglossaryentryprehook .....	
\@glsshowtarget .....	5, 6, 127	.....	80, 81, 86, 88, 269, 353
\@glssymbol .....	140	\@nil .....	19, 92, 120–122, 156, 174,
\@glssymbol@ .....	140, 141	175, 193, 195, 208, 209, 223, 224, 330, 331	
\@glssymbolplural .....	141	\@nnil .....	19, 196
\@glssymbolplural@ .....	141	\@no@makeglossaries .....	31, 32, 181, 183
\@glstarget .....	128, 216, 277	\@no@post@desc .....	336
\@glstext .....	135	\@nopostdesc .....	200
\@glstext@ .....	135	\@onelevel@sanitize .....	7, 23, 53, 78, 94, 120,
\@glsunset .....	96, 99	121, 166, 167, 170, 195, 197, 208, 332, 333	
\@glsuseri .....	142	\@onlypreamble ..	67, 77, 87, 100, 103, 182, 185
\@glsuseri@ .....	142	\@onlypremakeg .....	43, 44, 50, 51, 55, 67, 173
\@glsuserii .....	143	\@org@glossaryentrynumbers .....	199, 200
\@glsuserii@ .....	143	\@org@gls@assign@descplural .....	
\@glsuseriii .....	144	.....	245, 254, 256–259, 391, 394, 395
\@glsuseriii@ .....	144	\@org@gls@assign@firstpl .....	245,
\@glsuseriv .....	145	247–249, 251, 252, 254, 256–259, 391–395	
\@glsuseriv@ .....	145	\@org@gls@assign@plural .....	245,
\@glsuserv .....	145	247–249, 251, 252, 254, 256–259, 391–395	
\@glsuserv@ .....	145	\@org@gls@assign@symbolplural ..	245,
\@glsuservi .....	146	247–249, 251, 252, 257–259, 391–393, 395	
\@glsuservi@ .....	146	\@org@glsnnumberformat .....	163
\@glswidestname .....	325–327, 343	\@org@newglossaryentryprehook .....	86
\@glswritefiles .....	33	\@outputpage .....	294, 295
\@glsxtr@doaccsupp .....	349	\@p@glossarysection .....	46
\@glsxtr@record .....	271	\@pgls .....	271
\@gobble .....	5, 14–16, 31, 32, 78, 79, 95,	\@pgls@ .....	271, 272
121, 165, 166, 170, 181, 182, 329, 333, 334		\@pglsp{ .....	272
\@gobblethree .....	5	\@pglsp{@ .....	272
\@idxitem .....	319	\@plus .....	281, 301, 319
\@ifclassloaded .....	4, 12, 46	\@print@glossary .....	198
\@ifglossaryexists .....	59, 197	\@print@noidx@glossary .....	198
\@ifl@t@r .....	25	\@printgloss@checkexists .....	198, 199
\@ifnextchar .....	67, 115	\@printgloss@checkexists@noignored ..	198
\@ifpackageloaded .....		\@printgloss@setsort .....	181, 183, 199
.....	4, 9, 27, 28, 41, 57, 95, 162, 173, 349	\@printglossary .....	198
\@ifstar .....	59, 66, 80, 81, 115, 166, 225, 226	\@roman .....	53, 331
\@undefined .....		\@secondofthree .....	
.....	40, 277, 284, 295, 306, 313, 326, 327, 343	115, 128, 130, 132, 148, 150, 151, 153, 372	
\@ignored@glossaries .....	32, 68, 69		

\@secondoftwo .	24, 27, 28, 41, 78, 80, 128– 131, 134, 147, 148, 151, 152, 372–374, 398		
\@set@glo@numformat .	194, 331	\AA .....	24
\@sglsaddkey .	81	\aa .....	24
\@sglsaddstoragekey .	80	accsupp package .....	<u>349, 350</u>
\@tabacckludge .	24	\accsuppglossaryentryfield .....	349
\@text@composite@x .	24	\accsuppglossarysubentryfield .....	350
\@thirdofthree .		\acrfootnote .....	247, 253
....	115, 131, 133, 148, 150, 152, 154, 373	\Acrfull .....	244
\@this@attr .	169, 170	\Acrfull .....	244
\@this@childlabel .	203	\ACRfullfmt .....	228, 231, 239, 241, 386, 388
\@this@counter .	51	\Acrfullfmt .....	228, 231, 239, 241, 386, 388
\@this@ctr .	169, 170	\acrfullfmt .....	227, 231, 239, 241, 386, 388
\@this@key .	80	\acrfullformat .....	162, 227, 245, 260
\@this@label .	202	\Acrfullpl .....	244
\@thiscs .	39	\acrfullpl .....	244
\@tmp .	53, 332	\ACRfullplfmt .....	229, 231, 239, 241, 386, 388
\@use@ption .	37	\Acrfullplfmt .....	229, 231, 239, 241, 386, 388
\@warn@nomakeglossaries .	179, 202	\Acrfullplfmt .....	228, 231, 239, 241, 386, 388
\@wrglossary@pageformat .	190	\acrfullplfmt .....	228, 231, 239, 241, 386, 388
\@wrglossarynumberhook .	190, 193	\acrlinkfootnote .....	246
\@xdy@main@language .	30, 177, 178, 201	\acrlinkfullformat .....	227–229
\@xdyattributelist .	51, 169	\Acrlong .....	244
\@xdyattributes .	51, 168, 329, 331	\acrlong .....	243
\@xdycounters .	50, 51, 169	\Acrlongpl .....	244
\@xdycrossrefhook .	169	\acrlongpl .....	243
\@xdylanguage .	201	\acrnameformat .....	251, 392
\@xdylettergroups .	58, 171, 334	\acronymentry .....	
\@xdylocationclassorder .	55, 169, 333	231, 233–238, 240–243, 381–384, 387–390	
\@xdylocref .	51, 171, 329, 333	\acronymfont .....	110, 111,
\@xdynumbergrouporder .	57, 167	147–150, 156, 162, 230, 231, 233–243,	
\@xdyrequiredstyles .	56, 168, 331	247, 248, 250, 252, 253, 255–257, 369,	
\@xdysortrules .	56, 171, 334	370, 372–374, 382–384, 386–390, 392–394	
\@xdystyle .	168, 331	\acronymname .....	17, 18, 42
\@xdyuseralphabets .	52, 168, 331	\acronymsort .....	
\@xdyuserlocationdefs .	54, 55, 169, 330, 332	231, 233–238, 240–243, 382–384, 387–390	
\@xdyuserlocationnames .	55, 330	\acronymtype .....	17, 18,
\@xfor@nextelement .	196	230–232, 245–254, 256–259, 261, 391–394	
\` .	93, 121, 166, 172, 223, 224, 334, 335, 337–339, 347, 348	\acrpluralsuffix .....	
\{ .	78, 79, 165, 172, 329, 334, 335	231, 233–236, 240–242, 245–249,	
\} .	79, 166, 172, 329, 335	251–258, 260, 261, 382, 387–389, 391–395	
\^ .	24	\Acrshort .....	243
\` .	24	\acrshort .....	243
\  .	121, 123, 124, 175	\Acrshortpl .....	243
\~ .	24	\acrshortpl .....	243
\a .	24	\add@accent@ .....	24
		\addcontentsline .....	49
		\addglossarytocaptions .....	41
		\addtolen...	327, 343
		\advance .....	15, 16, 89, 119, 294
		\AE .....	24
		\ae .....	24

## A

\amsn package . . . . .	4, 114
\amsmath package . . . . .	95
\andname . . . . .	196
\AnyTrackedLanguages . . . . .	41, 398
\appto . . . . .	20, 25, 73, 80, 81, 269, 353
array package . . . . .	291, 295, 313
article class . . . . .	194
\AtBeginDocument . . . . .	18, 57, 78, 95, 164, 183
\AtEndDocument . . . . .	32, 33, 78, 100, 182, 186, 201, 277
<b>B</b>	
\b . . . . .	24
babel package . . . . .	27, 39, 41, 56
\begin . . . . .	170, 207, 281, 285–290, 293–319, 333
\BeginAccSupp . . . . .	350
\begingroup . . . . .	5, 7, 188, 192, 224
\bfseries . . . . .	286–289, 291, 292, 296–298, 300, 308–313, 315–319
\bgroup . . . . .	24, 86, 163, 199, 202
bib2gls . . . . .	24, 61, 191
booktabs package . . . . .	290–293
\boolean . . . . .	260
\boolfalse . . . . .	33, 34
\booltrue . . . . .	34
\bottomrule . . . . .	291, 292
\box . . . . .	294
<b>C</b>	
\c . . . . .	24
\c@equation . . . . .	119
\c@glossaryentry . . . . .	12
\c@glossarysubentry . . . . .	13
\c@page . . . . .	189, 190, 192, 193
\catcode . . . . .	78
\cGls . . . . .	102
\cgls . . . . .	101
\cGlsformat . . . . .	100
\cglstformat . . . . .	99
\cGlspl . . . . .	103
\cglsp . . . . .	102
\cGlsplformat . . . . .	100
\cglsp . . . . .	100
\char . . . . .	220
classicthesis package . . . . .	9
\cleardoublepage . . . . .	48
\clearpage . . . . .	48
\closeout . . . . .	32, 78, 171, 173, 177, 187
\compatglossarystyle . . . . .	336–348
\compatibleglossentry . . . . .	218
\compatiblesubglossentry . . . . .	218
\copy . . . . .	294, 295
\count@ . . . . .	208
\csdef . . . . .	22, 80–83, 92, 93, 98, 99, 202, 203, 222, 232, 233, 335
\csedef . . . . .	101, 190
\csgdef . . . . .	45, 65, 68, 99, 100, 197, 211
\cslet . . . . .	73, 86, 93, 206
\csname . . . . .	13–16, 24, 37, 39, 41, 42, 47, 48, 51, 53, 54, 57, 58, 61, 66–68, 75, 76, 80–85, 89, 90, 92–95, 97, 113, 117, 119, 120, 129–134, 147–155, 163, 164, 168–170, 174–177, 183, 186– 188, 190, 194, 195, 199, 201, 203, 212, 216, 218, 221, 222, 225, 226, 262–270, 277, 278, 325–327, 329–331, 343, 349, 350, 354, 355, 358, 362, 372–374, 396, 397
\csshow . . . . .	266
\csuse . . . . .	42, 45, 65, 75, 76, 82, 83, 113, 114, 178, 179, 203, 205, 207, 209, 211–213, 221, 233, 270, 336–348
\csxdef . . . . .	91, 101
\currentglossary . . . . .	12, 13, 45, 199
\currentglssubentry . . . . .	13, 215
\CurrentOption . . . . .	37, 269, 349
\CurrentTrackedLanguage . . . . .	42, 398, 399
\CurrentTrackedTag . . . . .	42, 398, 399
\CustomAcronymFields . . . . .	261
\CustomNewAcronymDef . . . . .	261
<b>D</b>	
\d . . . . .	24
datatool package . . . . .	204
datatool-base package . . . . .	4
\day . . . . .	167, 172, 331, 334
\DeclareAcronymList . . . . .	17, 18, 20, 230, 232, 246, 248, 250, 252, 254, 257, 259, 261
\DeclareListParser . . . . .	181
\DeclareOption . . . . .	9, 269, 349
\DeclareOptionX . . . . .	9
\DeclareRobustCommand . . . . .	42, 255
\def . . . . .	9, 10, 12–16, 18, 19, 23–25, 30, 31, 34, 35, 37–39, 42, 43, 46, 49, 50, 52–58, 61, 62, 65–74, 76, 84, 86, 88–91, 93, 95, 99– 103, 113, 114, 117–127, 129–154, 163, 164, 167, 171, 173–179, 181, 183, 184, 186, 190, 192–195, 199, 202, 206–208, 210–213, 220–225, 227–230, 245–252,

254, 256–259, 261, 269, 272–275, 278–  
 280, 305, 306, 325–328, 330, 331, 334,  
 336, 343, 344, 349–354, 372–379, 391–395  
`\def@gls@xdycheckbackslash` .... 126, 127  
`\DefaultNewAcronymDef` ..... 246  
`\defglsentryfmt` ..... 67, 69, 113,  
 232, 245, 246, 249, 250, 253, 255, 258, 260  
`\define@boolkey` ..... 8,  
 10, 12, 13, 17, 25, 26, 29, 30, 33, 34, 115, 213  
`\define@choicekey` ..... 5,  
 7–9, 13, 26, 28, 30, 33, 35, 72, 212, 213  
`\define@key` ..... 9, 13, 20, 26, 30, 31, 69–  
 74, 80, 81, 114, 164, 211, 213, 269, 351–353  
`\DefineAcronymSynonyms` ..... 37, 244  
`\delimN` ..... 170, 181, 210, 224, 333  
`\delimR` ..... 170, 223, 333  
`\DescriptionDUANewAcronymDef` ..... 250  
`\DescriptionFootnoteNewAcronymDef` . 248  
`\descriptionname` ..... 42, 286–289,  
 291, 292, 296–298, 300, 308–313, 315–319  
`\DescriptionNewAcronymDef` ..... 252  
`\DH` ..... 25  
`\dh` ..... 25  
`\dimen@` ..... 234, 294, 325  
`\disable@keys` ..... 37  
`\do` ..... 28, 37, 39, 50,  
 51, 58, 78, 80, 121, 163, 168, 169, 180,  
 182, 189, 196, 202, 203, 232, 246, 248,  
 250, 252, 254, 257, 259, 261, 277, 278, 331  
`\do@glo@storeentry` ..... 14–16, 92  
`\do@gls@link@checkfirsthyper` .....  
 .... 116, 118, 129–134, 147–154, 372–374  
`\do@gls@xdycheckbackslash` ..... 120  
`\do@glsdisablehyperinlist` ..... 118  
`\do@glshaschildren` ..... 61, 62  
`doc package` ..... 4, 5, 17  
`\doifglossarynoexistsordo` ..... 66  
`\dtl@ifsingle` ..... 220  
`\dtl@insertinto` ..... 204  
`\dtl@sortresult` ..... 204, 205  
`\dtlcompare` ..... 204  
`\dtlicompare` ..... 205  
`\DTLifinlist` ..... 69, 117  
`\DTLifint` ..... 220  
`\dtlletterindexcompare` ..... 204  
`\DTLsubstituteall` ..... 121  
`\dtlwordindexcompare` ..... 204  
`\DUANewAcronymDef` ..... 259

**E**

`\eappto` ..... 68, 69, 93, 190  
`\edef` ..... 16, 19, 39, 41,  
 49–56, 58, 61, 66–69, 75, 76, 78, 80, 83–  
 88, 93, 94, 113, 117, 119–127, 163–166,  
 171, 173–175, 177–179, 181, 183, 186,  
 191, 193, 194, 197, 201–205, 208, 215,  
 220, 224–226, 245, 247, 249, 251, 253,  
 256, 258, 276, 329, 330, 332, 334, 390–394  
`\egroup` ..... 24, 86, 163, 200, 203  
`\else` ..... 6, 11–13, 16–  
 19, 21, 23, 25, 26, 33, 35, 37–39, 43, 44,  
 46–53, 55–58, 72, 74, 89, 90, 93–95, 100,  
 117–127, 129–134, 156, 166–168, 170,  
 171, 173–179, 187–190, 192–196, 199,  
 208, 213, 215, 221, 223, 224, 234, 248,  
 250, 252, 255, 257–260, 262, 277, 282,  
 285, 287, 288, 291, 292, 294, 296, 298,  
 299, 307, 309, 311, 314, 316, 318, 321,  
 322, 324, 326, 327, 330–336, 341–344, 359  
`\emph` ..... 196, 225  
`\empty` ..... 224, 349  
`\encodingdefault` ..... 24  
`\end` ..... 170, 208, 281, 285–290, 293–319, 333  
`\end@doifinlist` ..... 50  
`\end@getprefix` ..... 194  
`\end@gls@islistofacronyms` ..... 19  
`\EndAccSupp` ..... 350  
`\endcsname` ..... 13–16, 24, 37,  
 39, 41, 42, 47, 48, 51, 53, 54, 57, 58, 61,  
 66–68, 75, 76, 80–85, 89, 90, 92–95, 97,  
 113, 117, 119, 120, 129–134, 147–155,  
 163, 164, 168–170, 174–177, 183, 186–  
 188, 190, 194, 195, 199, 201, 203, 212,  
 216, 218, 221, 222, 225, 226, 262–270,  
 277, 278, 325–327, 329–331, 343, 349,  
 350, 354, 355, 358, 362, 372–374, 396, 397  
`\endfoot` . 285–287, 289, 291, 292, 296–298, 300  
`\endgroup` ..... 5, 7, 188, 193, 224  
`\endhead` . 285–287, 289, 291, 292, 296–298, 300  
`\endtheglossary` ..... 5  
`\entryname` ..... 42, 286–289,  
 291, 292, 296–298, 300, 308–313, 315–319  
`\equal` ..... 26, 38, 48, 118, 180, 220, 277  
`equation (counter)` ..... 118, 119  
`etoolbox package` ..... 4  
`\expandafter` ..... 7, 14–16, 23,  
 24, 37, 39, 41, 51, 53, 54, 56–59, 61, 66–  
 69, 75, 76, 78–85, 89, 90, 92, 94, 95, 97,

113, 117, 119–126, 156, 163, 165, 166, 169, 170, 173–175, 177, 186–190, 192–194, 196, 199, 203, 208, 209, 216–218, 222, 224–226, 247, 253, 262–267, 269, 270, 277, 278, 325, 329–331, 333, 334, 349, 350, 354, 355, 358, 359, 390, 396, 397	\GenericAcronymFields 231, 233, 234, 236–240, 242, 243, 381–384, 386, 387, 389, 390
\expandoncde ..... 75, 76, 121, 174–176, 190, 204, 205, 216, 218, 231, 245, 247, 249, 251, 253, 254, 256, 258, 349, 350	\Genplacrfullformat ..... 111, 231, 233–235, 241, 371, 381, 382, 387
<b>F</b>	
\fi ..... 5, 6, 8, 9, 11–19, 21, 23, 25, 26, 28, 33, 35–39, 42–44, 46–58, 68, 72, 75, 89–95, 100, 117–127, 129–134, 156, 157, 164, 166–168, 171, 173–182, 187–197, 199, 202, 208, 212–216, 221–224, 234, 244, 246, 248–250, 252, 254, 255, 257–262, 268, 277, 282, 285, 287, 288, 291, 292, 294–296, 298, 299, 307, 309, 311, 314, 316, 318, 321, 322, 324–327, 329–333, 335, 336, 341–344, 349, 359	\glo@desc ..... 336
file types	\glo@do@compare ..... 204, 205
.aux ..... 201	\glo@grabfirst ..... 209
.glo ..... 94	\glo@label ..... 61, 93
.ist ..... 165, 176	\glo@list ..... 93
.toc ..... 49	\glo@name ..... 217
.xdy ..... 43	\glo@parent ..... 61, 62
glo ..... 267	\glo@type ..... 93
\firstacronymfont ..... 112, 233–235, 240, 241, 247, 251, 253, 256, 371, 372, 381, 382, 387, 388	\glo@value ..... 78, 79
\fmtversion ..... 25	\global ..... 15, 16, 75, 77, 86, 92, 97, 188, 200, 209, 214, 294, 295
\footnote ..... 240, 241, 246, 387, 388	\glolinkprefix ..... 118, 163, 216
\FootnoteNewAcronymDef ..... 254	\glosortentrieswarning ..... 21, 202
\footnotesize ..... 6	glossaries package ..... 35, 37, 56, 57, 167, 261, 269, 281, 329, 349
\forallglossaries ..... 59, 186, 198, 325	glossaries-accsupp package ..... 93, 349, 350
\forallglsentries ..... 32, 97, 98, 101, 165	glossaries-extra package ..... 78, 169, 176, 199, 271, 349, 358
\ForEachTrackedDialect ..... 41, 398, 399	\GlossariesWarning ..... 5, 6, 8, 21, 25, 26, 31, 32, 34, 35, 46, 49, 60, 64, 72, 74, 101–103, 113, 115, 162, 178, 179, 182, 184–186, 188, 194, 197–199, 218, 219, 221, 329, 349
\forglsentries ..... 59, 61, 93, 206, 325	\GlossariesWarningNoLine ..... 5, 6, 21, 181, 183, 187, 202, 277
\forlistcsloop ..... 202, 208	\glossary ..... 34, 35, 330, 331
\forlistloop ..... 184, 185, 210	glossary package ..... 1, 35, 225
<b>G</b>	
garamondx package ..... 226	glossary styles:
\gdef .. 15, 51, 66, 84, 89, 90, 188, 213, 214, 277	altlist ..... 282, 283, 337
\Genacrfullformat ..... 112, 231, 233–235, 240, 371, 381, 382, 387	altlistgroup ..... 283, 337
\genacrfullformat ..... 111, 112, 231, 233, 234, 240, 371, 381, 382, 387	altlisthypergroup ..... 283, 337
	altlong4col ..... 289, 290, 298, 339
	altlong4col-booktabs ..... 292, 294
	altlong4colborder ..... 290, 339
	altlong4colheader ..... 290, 292, 339
	altlong4colheaderborder ..... 290, 339
	altlongragged4col ... 293, 298–300, 340
	altlongragged4col-booktabs ..... 293
	altlongragged4colborder .... 300, 340
	altlongragged4colheader .... 299, 340
	altlongragged4colheaderborder 300, 341
	altsuper4col ..... 312, 317, 348
	altsuper4colborder ..... 312, 348

altsuper4colheader .....	312, 348	mcolindexhypergroup .....	301, 302, 344
altsuper4colheaderborder ...	312, 348	mcoltree .....	302, 344
altsuperragged4col .....	317, 318, 346	mcoltreegroup .....	344
altsuperragged4colborder ...	318, 346	mcoltreehypergroup .....	303, 344
altsuperragged4colheader ...	318, 346	mcoltreename .....	304, 345
altsuperragged4colheaderborder .	318, 346	mcoltreenamegroup .....	304, 345
alttree .....	305, 320, 325, 327, 343	mcoltreenamehypergroup ...	304, 345
alttreegroup .....	328, 344	sublistdotted .....	337
alttreehypergroup .....	328, 344	super .....	307, 308, 315, 347
index .....	9, 301, 319–322, 341	super3col .....	308–310, 347
indexgroup .....	321, 341	super3colborder .....	309, 347
indexhypergroup .....	321, 341	super3colheader .....	309, 347
inline .....	336	super3colheaderborder .....	310, 347
list .....	9, 11, 281–283, 336	super4col .....	310–312, 348
listdotted .....	283, 284, 337	super4colborder .....	311, 348
listgroup .....	282, 336	super4colheader .....	311, 348
listhypergroup .....	282, 337	super4colheaderborder .....	311, 348
long .....	285, 286, 291, 295, 337, 339	superborder .....	307, 347
long-booktabs .....	291, 293	superheader .....	308, 347
long3col .....	286, 287, 291, 338	superheaderborder .....	308, 347
long3col-booktabs .....	291, 293	superragged .....	313, 315, 345
long3colborder .....	287, 338	superragged3col .....	315–317, 346
long3colheader .....	287, 291, 338	superragged3colborder .....	316, 346
long3colheaderborder .....	287, 338	superragged3colheader .....	316, 346
long4col .....	288, 289, 292, 338	superragged3colheaderborder .....	317, 346
long4col-booktabs .....	292	superraggedborder .....	314, 345
long4colborder .....	289, 339	superraggedheader .....	315, 345
long4colheader .....	288, 292, 339	superraggedheaderborder .....	315, 346
long4colheaderborder .....	289, 339	tree .....	302, 322, 323, 325, 341
longborder .....	285, 338	treegroup .....	303, 323, 342
longheader .....	285, 291, 338	treehypergroup .....	323, 342
longheaderborder .....	286, 338	treenoname .....	303, 320, 323, 324, 342
longragged .....	293, 295–297	treenonamegroup .....	324, 343
longragged-booktabs .....	293	treenonamehypergroup .....	324, 343
longragged3col .....	293, 297, 298, 340	glossary-hypernav package .....	165
longragged3col-booktabs .....	293	glossary-list package .....	9, 11, 281
longragged3colborder .....	298, 340	glossary-long package ...	10, 284, 298, 306, 307
longragged3colheader .....	298, 340	glossary-longragged package .....	295
longragged3colheaderborder .....	298, 340	glossary-mcols package .....	300
longraggedborder .....	296, 339	glossary-super package ...	11, 284, 306, 313, 317
longraggedheader .....	296, 340	glossary-superragged package .....	313
longraggedheaderborder .....	297, 340	glossary-tree package .....	11, 319
mcolalttree .....	305, 345	\glossaryentry .....	194, 195, 331
mcolalttreegroup .....	305, 345	\glossaryentry (counter) .....	12, 214, 215
mcolalttreehypergroup ...	305, 306, 345	\glossaryentryfield .....	216, 336–343, 345–348, 381
mcolindex .....	301, 344	\glossaryentrynumbers .....	. 10, 170, 199, 200, 209, 210, 213, 214, 333
mcolindexgroup .....	301, 344		

\glossaryheader .....	170, 207, 279, 281–283, 285–289, 291, 292, 295–301, 303–305, 307, 308, 310, 314, 315, 317, 320–325, 328, 333	\gls@checkisacronymlist .....	117
\glossarymark .....	46	\gls@checkseeallowed .....	71, 77, 181, 183
\glossaryname .....	17, 41, 42	\gls@checkseeallowed@preambleonly ..	77
\glossarypostamble .....	170, 208, 333	\gls@codepage .....	57, 178, 179, 201
\glossarypreamble .....	169, 207, 333	\gls@debug@nr .....	5, 35
\glossarysection .....	169, 207, 333	\gls@debug@val .....	5, 35
glossarysubentry (counter) .....	13, 214, 215	\gls@defdocnewglossaryentry .....	78, 99
\glossarysubentryfield .....	218, 336–343, 345–348, 381	\gls@defglossaryentry .....	77, 78, 86
\glossarytitle .....	169, 199, 207, 211, 333	\gls@disablepagerefexpansion ..	188, 193
\glossarytoctitle .....	9, 17, 18, 36, 39, 42, 46, 169, 199, 207, 211, 212, 333	\gls@do@addxdyattribute .....	51
\glossentry .....	93, 200, 209, 210, 218, 279, 281–286, 288, 296, 297, 299, 307, 309, 310, 314, 315, 317, 318, 320, 322, 323, 326	\gls@doclearpage .....	48
\Glossentrydesc .....	380	\gls@dosubst .....	121
\glossentrydesc .....	. 279–286, 288, 296, 297, 299, 307, 309, 310, 314–317, 320–322, 324, 326, 327, 380	\gls@dotocitle .....	199, 200, 211
\glossentryname .....	. 279–286, 288, 296, 297, 299, 307, 309, 310, 314, 315, 317, 320–323, 326, 327, 380	\gls@end@sanitizesort .....	23, 24
\Glossentrysymbol .....	380	\gls@endcheck .....	76
\glossentrysymbol .....	279, 280, 288, 299, 310, 317, 320–322, 324, 326, 327, 380	\gls@glossary .....	34, 35, 194, 195
\Gls .....	102, 225, 226, 244	\gls@gobbleopt .....	67
\gls .....	34, 101, 182, 215, 225, 226, 244	\gls@grplabel .....	276
\gls@accessibility .....	351	\gls@hypergrouprerun .....	277
\gls@accsupp@engine .....	350	\gls@ifnotmeasuring .....	96
\gls@Alphpage .....	189, 193	\gls@inlinenpostchild .....	279, 280, 336
\gls@alphpage .....	189, 193	\gls@inlinesep .....	279, 336
\gls@arabicpage .....	189, 192	\gls@inlinesubsep .....	279, 280, 336
\gls@assign@desc .....	86, 91	\gls@islistofacronyms .....	19
\gls@assign@descplural .....	245, 254, 256–259, 391, 394, 395	\gls@istfilebase .....	43, 177, 179
\gls@assign@field .....	77, 81, 82, 86, 88, 90–92, 269, 270	\gls@label .....	225
\gls@assign@firsttpl .....	245, 247–249, 251, 252, 254, 256–259, 391–395	\gls@level .....	89, 90, 209
\gls@assign@plural .....	245, 247–249, 251, 252, 254, 256–259, 391–395	\gls@noidxglossary .....	182
\gls@assign@symbolplural .....	245, 247–249, 251, 252, 257–259, 391–393, 395	\gls@nonumberlist@nr .....	72
\gls@automake@nr .....	33, 180	\gls@nonumberlist@val .....	72
\gls@automake@val .....	33	\gls@nosetquote .....	87, 171, 173, 176
\gls@begindocdefs .....	78	\gls@number .....	192
		\gls@numberedsection@nr .....	9, 212
		\gls@numberedsection@val .....	9, 212
		\gls@numberpage .....	189, 192
		\gls@org@glossaryentryfield .....	200
		\gls@org@glossarysubentryfield .....	200
		\gls@org@insert .....	250, 253, 255
		\gls@orgAlph .....	192, 193
		\gls@orgalph .....	192, 193
		\gls@orgarabic .....	192
		\gls@orgnumber .....	192
		\gls@orgRoman .....	192, 193
		\gls@orgromannumeral .....	192, 193
		\gls@orgthe .....	192
		\gls@original@glossary .....	35
		\gls@original@makeglossary .....	35
		\gls@protected@pagefmts .....	121, 189, 190
		\gls@Romanpage .....	189, 193

\gls@romanpage ..... 189, 193  
\gls@save@numberlist ..... 10  
\gls@seenoindex@nr ..... 7  
\gls@seenoindex@val ..... 7  
\gls@set@xr@key ..... 71  
\gls@suffixF ..... 44, 170–173, 333, 335  
\gls@suffixFF ..... 44, 170–173, 333, 335  
\gls@text ..... 112, 113  
\gls@the ..... 192  
\gls@thissty ..... 28  
\gls@tmp ..... 186, 255  
\gls@tmpolen ..... 127, 325–327, 343, 344  
\gls@tr@set@acronym@toctitle ..... 18  
\gls@tr@set@main@toctitle ..... 17  
\gls@tr@set@numbers@toctitle ..... 36  
\gls@tr@set@symbols@toctitle ..... 36  
\gls@translate@nr ..... 28  
\gls@translate@val ..... 28  
\gls@wrgglossary ..... 188  
\gls@xdystring ..... 120, 121  
\gls@xindy@glsnumbersfalse ..... 31  
\gls@xindy@glsnumberstrue ..... 30  
\gls@xr@key ..... 7, 8, 71, 72  
\glsaccessibility ..... 358, 359  
\glsaccsupp ..... 358, 359  
\glsacronymtrue ..... 18  
\glsacrpluralsuffix ..... 40, 226, 235, 236, 240–242, 246  
\glsacrshortcutsfalse ..... 37  
\glsacrshortcutstrue ..... 37  
\glsacspace ..... 234, 236  
\glsadd ..... 34, 165  
\glsadd options  
  counter ..... 164  
  format ..... 164, 223  
\glsaddall options  
  types ..... 164  
\GlsAddXdyAttribute ..... 50, 52, 329, 330  
\GlsAddXdyCounters ..... 50, 51, 68  
\glsautomakefalse ..... 33, 180  
\glsautomaketru ..... 33  
\glsautoprefix ..... 9, 212  
\glscapscase ..... . 104, 106, 108–111, 129–134, 147–154,  
                      238, 239, 251, 256, 362, 364, 367–374, 385  
\glscategory ..... 358  
\glsclearpage ..... 48  
\glsclosebrace ..... 54, 55, 170, 171, 333, 334  
\glscompositor ..... 44, 54, 172, 332, 335  
\glscounter ..... 20, 38, 50, 67, 91, 118, 329  
\glscurrententrylabel ..... 197, 200  
\glscurrentfieldvalue ..... 64  
\glscustomtext ..... 104, 107, 108, 110, 112, 129–134, 147–  
                      154, 238, 239, 246, 247, 250, 251, 253,  
                      255, 256, 362, 366, 369, 371–374, 384, 386  
\GlsDeclareNoHyperList ..... 20  
\glsdefaultshortaccess ..... 390–394  
\glsdefaulttype ..... 17,  
                      45, 57, 58, 65, 66, 88, 103, 113, 186, 198, 199  
\glsdefmain ..... 17, 68  
\glsdescriptionaccessdisplay ..... 364–366, 376, 380, 381  
\glsdescriptionpluralaccessdisplay ..... 363, 364, 376, 377  
\glsdescwidth ..... 285–  
                      287, 289, 290, 293–300, 307–310, 312–319  
\glsdetoklabel ..... 60–64, 73,  
                      78, 83–87, 93, 97, 99–101, 117, 155, 156,  
                      163, 164, 183, 184, 191, 193, 200, 203–  
                      205, 209, 211, 214, 215, 217, 262–266,  
                      270, 271, 325, 349, 350, 359–362, 396, 397  
\glsdisplay ..... 104, 114  
\glsdisplayfirst ..... 104, 113  
\glsdisplaynumberlist ..... 184  
\glsdohyperlink ..... 128  
\glsdochypertarget ..... 128  
\glsdoifexists ..... 61–  
                      63, 83–86, 96, 129–134, 147–154, 159,  
                      162–164, 184, 185, 272–275, 372–374, 380  
\glsdoifexistsordo ..... 116, 155  
\glsdoifexistsorwarn ..... 211, 217  
\glsdoifnoexists ..... 77, 86  
\glsdonohyperlink ..... 118, 128  
\glsdosanitizesort ..... 14  
\glsentryaccess ..... 359  
\glsentrycounter ..... 221, 224  
\glsentrycounterfalse ..... 12  
\glsentrycounterlabel ..... 215  
\GlsEntryCounterLabelPrefix ... 214, 215  
\glsentrycountertrue ..... 13  
\glsentrycurrcount ..... 99, 101  
\Glsentrydesc ..... 139, 217, 376, 380  
\glsentrydesc ..... 106–108, 139, 217, 364–366, 376, 380  
\glsentrydescaccess ..... 360  
\Glsentrydescplural ..... 140, 376

\glsentrydescplural .....  
     ..... 104–106, 140, 363, 364, 376, 377  
 \glsentrydescpluralaccess ..... 361  
 \Glsentryfirst 102, 107, 109, 136, 365, 368, 375  
 \glsentryfirst ..... 102, 106,  
     107, 109, 110, 136, 365, 366, 368, 369, 375  
 \glsentryfirstaccess ..... 360  
 \Glsentryfirstplural .....  
     ..... 103, 105, 108, 138, 363, 367, 375  
 \glsentryfirstplural ... 102, 104–106,  
     108, 109, 137, 138, 363, 364, 367, 368, 375  
 \glsentryfirstpluralaccess ..... 360  
 \glsentryfmt ..... 67, 69  
 \Glsentryfull ..... 231, 240, 241, 386, 388  
 \glsentryfull ..... 231, 240, 241, 386, 388  
 \Glsentryfullpl ..... 231, 240, 241, 386, 389  
 \glsentryfullpl ..... 231, 240, 241, 386, 388  
 \glsentryitem ..... 279, 281–286, 288,  
     296, 297, 299, 307, 309, 310, 314, 315,  
     317, 320, 322, 323, 326, 336–343, 345–348  
 \Glsentrylong ..... 102, 152, 156, 162,  
     233, 234, 239, 240, 371, 374, 381, 385, 386  
 \glsentrylong ... 102, 112, 151, 152, 156,  
     162, 233–243, 253, 371, 373, 374, 381–390  
 \glsentrylongaccess ..... 361  
 \Glsentrylongpl ..... 103, 154,  
     162, 233, 234, 238–240, 372, 381, 385, 386  
 \glsentrylongpl .....  
     .... 102, 112, 153, 154, 162, 233–235,  
     238–241, 253, 260, 372, 381, 382, 385–389  
 \glsentrylongpluralaccess ..... 361  
 \Glsentryname ..... 138, 217, 376, 380  
 \glsentryname ..... 138, 139, 325, 376, 380  
 \glsentrynumberlist ..... 162, 184  
 \Glsentryplural . 105, 108, 137, 363, 367, 375  
 \glsentryplural ..... 104,  
     105, 108, 109, 136, 137, 363, 364, 367, 375  
 \glsentrypluralaccess ..... 360  
 \Glsentryprefix ..... 273  
 \glsentryprefix ..... 272, 274  
 \Glsentryprefixfirst ..... 273  
 \glsentryprefixfirst ..... 272, 275  
 \Glsentryprefixfirstplural ..... 274  
 \glsentryprefixfirstplural ..... 272, 275  
 \Glsentryprefixplural ..... 274  
 \glsentryprefixplural ..... 272, 275  
 \glsentryprevcount ..... 99, 100  
 \Glsentryshort ..... 111,  
     148, 156, 235, 241, 370–372, 382, 387, 388  
 \glsentryshort 111, 112, 147, 148, 156, 162,  
     231, 233–243, 370–373, 381–384, 386–390  
 \glsentryshortaccess ..... 361  
 \Glsentryshortpl .....  
     .... 110, 150, 235, 241, 369, 382, 388, 389  
 \glsentryshortpl .....  
     110, 112, 149, 150, 162, 233–235, 239–  
     241, 260, 369, 370, 372, 381, 382, 386–389  
 \glsentryshortpluralaccess ..... 361  
 \Glsentrysymbol ..... 141, 218, 377, 380  
 \glsentrysymbol ..... 106–108,  
     141, 217, 247, 251, 256, 364–366, 377, 380  
 \glsentrysymbolaccess ..... 360  
 \Glsentrysymbolplural ..... 142, 377  
 \glsentrysymbolplural ..... 104–  
     106, 141, 142, 247, 251, 256, 363, 364, 377  
 \glsentrysymbolpluralaccess ..... 360  
 \Glsentrytext ... 106, 109, 135, 365, 368, 374  
 \glsentrytext ..... 106, 107, 109,  
     110, 135, 163, 196, 364, 365, 368, 369, 374  
 \glsentrytextaccess ..... 360  
 \glsentrytype ..... 88  
 \Glsentryuseri ..... 142, 378  
 \glsentryuseri ..... 142, 143, 377, 378  
 \glsentryuseriaccess ..... 361  
 \Glsentryuserii ..... 143, 378  
 \glsentryuserii ..... 143, 144, 378  
 \glsentryuseriiaccess ..... 361  
 \Glsentryuseriiv ..... 144, 378  
 \glsentryuseriiv ..... 144, 378  
 \glsentryuseriiaccess ..... 362  
 \Glsentryuseriiv ..... 145, 379  
 \glsentryuseriiv ..... 145, 379  
 \glsentryuserivaccess ..... 362  
 \Glsentryusersv ..... 146, 379  
 \glsentryusersv ..... 145, 146, 379  
 \glsentryusersvaccess ..... 362  
 \Glsentryusersvi ..... 147, 379  
 \glsentryusersvi ..... 146, 147, 379  
 \glsentryusersviaccess ..... 362  
 \glsesclocationsfalse ..... 182  
 \glsesclocationstrue ..... 10  
 \glsfieldaccsupp ..... 358, 359  
 \glsfieldfetch ..... 159  
 \glsfirstaccessdisplay .....  
     ..... 365, 366, 368, 369, 375  
 \glsfirstpluralaccessdisplay .....  
     ..... 363, 364, 367, 375  
 \glsfirstpluralaccessdisplay ..... 368

\glsgenacfmt ....	233, 234, 240, 381, 382, 387	format .....	114, 129, 223
\glsgenentryfmt .....	....	hyper .....	114, 116, 117, 129
....	233, 234, 239, 240, 245, 247, 249–	local .....	115
251, 253, 255, 258, 260, 381, 382, 385, 387		\glslinkcheckfirsthyperhook .....	117
\glsgetgroupitle .....	....	\glslinkpostsetkeys .....	118
....	278, 282, 283, 301–306, 321–324, 328	\glslinkvar .....	115
\glsGLOSSARYmark .....	46	\glslistdottedwidth .....	283, 284, 337
\glsgroupheading .....	171, 209, 279, 281–283,	\glslistgroupheaderfmt .....	282, 283
285, 286, 288, 296, 297, 299, 301–308,		\glslistnavigationitem .....	282, 283
310, 314, 315, 317, 320–325, 327, 328, 334		\glslocalreset .....	98
\glsgroupskip .....	170, 171, 209, 280, 282, 285,	\glslocalunset .....	98, 129–134
287, 288, 291, 292, 296–299, 307, 309,		\glslongaccessdisplay .....	371, 373, 374, 381–391
311, 314, 316, 318, 321, 322, 324, 327, 333		\glslongkey .....	395
\glshyperfirstfalse .....	240, 387	\glslongpluralaccessdisplay .....	
\glshyperfirsttrue .....	29	....	372, 381, 382, 385–389, 391
\glshyperlink .....	196	\glslongpluralkey .....	395
\glshypernavsep .....	278	\glslongtok .....	
\glshypernumber .....	45, 224, 225	....	230, 231, 233, 234, 239, 240, 245–
\glsifhyperon .....	115	254, 256–261, 381, 382, 386, 387, 390–395	
\glsIfListOfAcronyms .....	19	\glsLTpenaltycheck .....	294, 295
\glsifplural .....	104, 108, 110,	\glsmcols .....	301–306
111, 129–134, 147–154, 238, 247, 251,		\glsnameaccessdisplay .....	376, 380, 381
253, 255, 362, 366, 369, 370, 372–374, 384		\glsnamefont .....	216–218, 349, 350, 380
\glsifusetranslator .....	27, 28, 41, 42, 398	\glsnavhyperlink .....	278
\glsindexonlyfirstfalse .....	29	\glsnavhyperlinkname .....	276, 277
\glsinlinedescformat .....	279, 336	\glsnavhypertarget .....	
\glsinlinedopostchild .....	279, 336	....	282, 283, 302–306, 322–324, 328
\glsinlineemptydescformat .....	279, 336	\glsnavigation .....	
\glsinlineenameformat .....	279, 336	....	282, 283, 301–306, 321, 323, 324, 328
\glsinlineparentchildseparator .....	279, 336	\glsnextpages .....	10, 72, 200
\glsinlinepostchild .....	279, 336	\glsnogroupskipfalse .....	12
\glsinlineseparator .....	279, 336	\glsnoidxdisplayloc .....	184–186
\glsinlinesubdescformat .....	280, 336	\glsnoidxdisplayloclisthandler .....	184
\glsinlinesubnameformat .....	279, 336	\glsnoidxloclist .....	184, 209, 210
\glsinlinesubseparator .....	280, 336	\glsnoidxloclisthandler .....	210
\glsinsert .....	104–	\glsnoidxnumberlistloophandler .....	185
112, 129–134, 147–154, 238, 239, 247,		\glsnoidxstripaccents .....	24
250, 251, 253, 255, 256, 363–374, 385, 386		\glsnomakeindexwarning .....	173
\glskeylisttok .....	230,	\glsnonextpages .....	72, 200
231, 245–252, 254, 256–259, 261, 390–395		\glsnopostdotfalse .....	12
\glslabel .....	.	\glsnoxindywarning .....	44, 50–52, 55–57, 166, 167
. 87, 104–112, 116–118, 148–154, 233,		\glsnumberformat .....	163
234, 238–240, 246, 247, 250, 251, 253,		\glsnumberlistloop .....	184
255, 256, 363–371, 381, 382, 384, 385, 387		\glsnumbersgroupname .....	36, 42, 220
\glslabeltok .....	....	\glsnumlistlastsep .....	163, 184
230, 231, 245–254, 256–261, 391–394		\glsnumlistparser .....	163, 181
\glslink .....	231, 239, 241, 246, 386, 388	\glsnumlistsep .....	163, 184
\glslink options .....		\glsopenbrace .....	54, 55, 170, 171, 333, 334
counter .....	114, 129, 267	\glsorder .....	30, 177–180, 205

\glsorg@endtheglossary .....	5	\glsshowtarget .....	7
\glsorg@PrintChanges .....	5	\glsshowtargetfont .....	6
\glsorg@theglossary .....	5	\glsshowtargetouter .....	6
\glspagelistwidth 286, 287, 289, 290, 293, 294, 297–300, 308–310, 312, 313, 315–319		\glsshowtargetsymbol .....	6
\glspatchLToutput .....	291–293	\glssortnumberfmt .....	15, 16
\glspenaltygroupskip .....	291, 292	\glsspace .....	227
\glspcentchar .....	78, 79, 170–172	\glsstepentry .....	215
\Glspl .....	103, 244	\glsstepsubentry .....	216
\glspl .....	102, 244	\glssubentrycounterfalse .....	13
\glspluralaccessdisplay 363, 364, 367, 375		\glssubentrycounterlabel .....	216
\glspluralsuffix .....	40, 90, 233–235, 382, 387–389	\glssubentryitem 280, 281, 283–286, 288, 296, 297, 299, 307, 309, 310, 314, 316, 317, 321, 322, 324, 326, 336–343, 345–348	
\glspostdescription .....	. 42, 280–283, 285, 296, 307, 314, 320– 322, 324, 326, 327, 336–339, 341–345, 347	\glssymbolaccessdisplay .....	364–366, 377, 380, 381
\glspostinline .....	279	\glssymbolpluralaccessdisplay .....	363, 364, 377
\glspostlinkhook .....	116, 129–135, 147–154, 372–374	\glssymbolsgroupname .....	36, 42, 220
\glsprefixsep .....	272–275	\glstarget .....	218, 219, 280–286, 288, 296, 297, 299, 307, 309, 310, 314–317, 320–324, 326, 327, 336–348
\glsprestandardsort .....	14	\glstextaccessdisplay 364, 365, 368, 369, 374	
\glsreset .....	97	\glstextformat .....	116, 118
\glsresetentrycounter .....	214	\glstextup .....	40, 389
\glsresetentrylist . 170, 207, 213, 214, 333		\glstildechar .....	51, 170, 171
\glsresetsubentrycounter ... 215, 279, 336		\glstranslatefalse .....	27, 28
\glssanitizesortfalse .....	26	\glstranslatetrue .....	28, 29
\glssanitizesorttrue .....	26	\glstreechildpredesc .....	321, 322
\glssavenumberlistfalse .....	10	\glstreegroupheaderfmt .....	301–306, 321, 323, 324, 328
\glssavewritesfalse .....	33	\glstreeindent .. 322, 324, 326, 327, 342–344	
\glsseeformat .....	169, 183–185, 333	\glstreeitem .....	301, 302, 320
\glsseeitem .....	196	\glstreenamebox .....	326, 327
\glsseeitemformat .....	196	\glstreenamefmt .....	319–323, 325–327
\glsseelastsep .....	196	\glstreenavigationfmt .....	301–306, 321, 323, 324, 328
\glsseelist .....	196	\glstreepredesc .....	320, 322, 324
\glsseesep .....	196	\glstreesubitem .....	301, 320
\glssetexpandfield .....	22, 25, 26	\glstreesubsubitem .....	301, 320
\glssetnoexpandfield .....	22, 23, 25, 26	\glstype .... 117, 129–134, 147–154, 372–374	
\GlsSetQuote .....	87, 171	\glsucmarkfalse .....	12
\glssettoctitle .....	42, 199	\glsucmarktrue .....	12
\glsshortaccessdisplay .....	156, 370–373, 381–384, 386–391	\glsunset .....	95, 98, 100, 129–134
\glsshortaccsupp .....	359	\glsupacrpluralsuffix .....	235, 236, 242, 248, 252, 255, 257
\glsshortkey .....	395	\GlsUseAcrEntryDispStyle ... 232, 235– 238, 240, 242, 243, 383, 384, 387, 389, 390	
\glsshortpluralaccessdisplay .....	369, 370, 372, 381, 382, 386–389, 391	\GlsUseAcrStyleDefs .....	232, 235– 238, 240, 242, 243, 383, 384, 387, 389, 390
\glsshortpluralkey .....	395		
\glsshorttok .....	230, 231, 245–254, 256–261, 390–395		
\glsshowaccsupp .....	6		

\glseriaccessdisplay .....	377, 378	177, 178, 186, 197, 201, 203, 211, 212,
\glseriiaccessdisplay .....	378	216, 220–223, 225, 232, 278, 335, 359–362
\glseriiiaccessdisplay .....	378	\ifdef ..... 63, 64, 73, 78, 95, 115,
\glserivaccessdisplay .....	379	155, 159, 184, 185, 226, 271, 319, 320, 358
\glservaccessdisplay .....	379	\ifdefempty ..... .. 20, 32, 47, 48, 58, 62–64, 68, 69, 104,
\glswallowprimitivemode true .....	191	108, 110, 180, 182, 208, 209, 230, 232,
\glswrite ... 167–173, 180, 186, 187, 331–335		238, 246, 250, 253, 255, 362, 366, 369, 384
\glswritedefhook .....	79	\ifdefequal ..... . 31, 32, 62–64, 75, 76, 80, 89, 93, 209, 359
\glswriteentry .....	188	\ifdefstrequal ..... 85
\glswritefiles .....	33, 186	\ifdefstring ..... 11,
\glsxindyfalse .....	30	41, 65, 177–179, 181, 205, 206, 208, 210, 350
\glsxindytrue .....	31	\ifdefvoid ..... 23, 92, 209
<b>H</b>		
\H .....	24	\ifdim ..... 234, 294, 325
\hangindent .....		\iffalse ..... 92, 97
305, 306, 319, 322–324, 326–328, 341–344		\IfFileExists ..... 11, 27, 28, 178, 179, 201
\hbox .....	95, 283, 284, 337	\ifglossaryexists ..... 45, 57, 61, 176–179
\hfill .....	283, 284, 337	\ifgls@sanitize@description ..... 25
\hline ... 285–287, 289, 296–298, 300, 307–319		\ifgls@sanitize@name ..... 25
\hsize .....	284, 295, 306, 313	\ifgls@sanitize@symbol ..... 25
\hspace .....	320	\ifgls@xindy@glsnumbers ..... 57
\hss .....	283, 284, 337	\ifglsacrdescription ..... 259
\ht .....	294	\ifglsacrdua ..... 249, 255, 258–260
\hyperdef .....	38	\ifglsacrfootnote ..... 117, 259, 260
\hyperlink .....	115, 127, 224	\ifglsacronym ..... 17, 18
hyperref package .....	194, 197, 223, 268	\ifglsacrshortcuts ..... 37, 244
\hypertarget .....	127	\ifglsacrsmalls caps ..... ..... 248, 250, 252, 254, 257, 258
<b>I</b>		
\IeC .....	24	\ifglsacrsmaller ..... 248, 250, 252, 255
\if .....	120, 193, 194, 330	\ifglsautomake ..... 33, 182
\if@endfor .....	277	\ifglsdescsuppressed ..... 279
\if@gls@debug .....	5, 21, 188	\ifglsentrycounter ..... 12, 13, 38, 214, 215
\if@gls@docloaded .....	4, 17, 35	\ifglsentryexists ..... 60, 61, 77, 78, 87, 89
\if@glsisacronymlist .....	117	\ifglsesclocations ..... 190
\if@openright .....	48	\ifglshaschildren ..... 279, 336
\ifbool .....	18, 29, 34, 60, 105–107	\ifglshasdesc ..... 279
\ifboolexpr .....	41, 65, 220	\ifglshaslong ..... 102, 103, 156, 233, 234, 238, 240, 253, 381, 382, 384, 387
\ifcase .....	5, 7, 9, 28, 35, 72, 212, 321, 341	\ifglshasparent ..... 203, 209, 325
\ifcsdef .....	27, 42, 48, 75, 76, 82–86, 113, 188, 202, 203, 205, 207, 221, 232, 358	\ifglshasprefix ..... 272–274
\ifcsempty .....	62, 271	\ifglshasprefixfirst ..... 272, 273, 275
\ifcsequal .....	62	\ifglshasprefixfirstplural ..... 272, 274, 275
\ifcsstrequal .....	86	\ifglshasprefixplural ..... 272, 274, 275
\ifcsstring .....	85	\ifglshassymbol ..... .... 247, 251, 255, 320–322, 324, 326, 327
\ifcsundef 8, 15, 31, 38, 41, 45, 46, 48, 59, 60, 67, 68, 71, 88, 90, 91, 99, 114, 119, 128,		\ifglshyperfirst ..... 117
		\ifglsindexonlyfirst ..... 189

\ifglsnogroupskip .....	282, 285, 287, 288, 291, 292, 296, 297, 299, 307, 309, 311, 314, 316, 318, 321, 322, 324, 327
\ifglsnonumberlist .....	213
\ifglsnopostdot .....	11
\ifglsnumberline .....	49
\ifglssanitizesort .....	23, 26
\ifglssavenumberlist .....	74, 181, 197
\ifglssavewrites .....	33, 177, 188
\ifglssubentrycounter ....	13, 38, 214–216
\ifglstoc .....	49
\ifglstranslate .....	41
\ifglscmark .....	46, 47
\ifglsused .....	101, 104–110, 116, 165, 189, 246, 250, 253, 255, 272–275, 363–369
\ifglswallowprimitivemods .....	192
\ifglsxindy .....	43, 44, 49, 50, 52, 54–57, 68, 93, 94, 121, 166, 167, 173, 177, 178, 193, 195, 201, 329–331
\ifignoredglossary .....	89, 92, 188, 197
\ifin@ .....	37
\ifinlistcs .....	207, 211
\ifinner .....	6
\ifKV@glslink@hyper .....	117, 118
\ifKV@glslink@local .....	129–134
\ifmeasuring@ .....	95
\ifmmode .....	6
\ifnum .....	14, 15, 33, 99, 100, 180, 208, 294, 322, 324, 326, 342, 343
\ifstrempty .....	7, 336
\ifstrequal .....	220
\ifthenelse .....	26, 38, 48, 118, 180, 220, 260, 277
\IfTrackedLanguage .....	173
\IfTrackedLanguageFileExists .....	41, 398, 399
\iftrue .....	92, 97
\ifundef .....	12, 13, 66, 77, 88, 167, 171, 180
\ifvmode .....	164
\ifvoid .....	294
\ifx .....	12, 14, 16, 19, 36, 37, 39, 50, 51, 53, 54, 57, 58, 89–91, 94, 119, 122–127, 156, 168, 170–175, 186, 190, 192–196, 199, 221–224, 246, 248, 250, 252, 254, 255, 257, 259, 261, 262, 331–333, 335, 336, 341–344, 349, 359
\immediate .....	32, 77–79, 101, 177–179, 187, 201
\in@ .....	37
\indexname .....	37
\indexspace .....	282, 301–306, 321–324, 327, 328
\input .....	40, 103
\inputencodingname .....	31
\InputIfFileExists .....	78
\istfilename .....	43, 167, 171, 178–180, 331, 334
\item .....	281–284, 301, 302, 320, 321, 336, 337, 341
<b>J</b>	
\jobname .....	32, 43, 77, 78, 167, 172, 177–179, 201, 331, 334
<b>K</b>	
\key@ifundefined .....	80, 81
\KV@glslink@hyperfalse .....	115, 117, 128
\KV@glslink@hypertrue .....	115, 128
<b>L</b>	
\L .....	24
\l .....	24
\label .....	9, 212, 214, 215
\language .....	30
\leaders .....	283, 284, 337
\leavevmode .....	86, 117
\let .....	5, 11, 14–18, 24, 25, 27, 28, 32–38, 41, 42, 51, 52, 64, 65, 75–77, 86–92, 95–97, 99, 100, 103, 115–118, 120, 121, 128–134, 147–154, 156, 163, 171, 173, 176–179, 181–185, 188–190, 192, 196, 198–200, 209, 211, 214, 218, 230, 243–245, 247–259, 269, 277, 278, 294, 301, 302, 320, 335, 353, 354, 372–374, 391–395, 398
\letcs .....	61–64, 78, 81, 85, 90, 91, 155, 156, 177, 178, 184, 202, 204, 205, 209, 217, 220, 325
link text .....	104
\listcsadd .....	207
\listcsgadd .....	211
\listcsxadd .....	202, 203
\listead .....	206
\loadglsentries .....	103
\long .....	86, 224
\longnewglossaryentry .....	87
longtable package .....	284, 291, 295
\LT@end@pen .....	294
\LT@err .....	294
\LT@foot .....	294, 295
\LT@head .....	294, 295
\LT@lastfoot .....	294
\LT@output .....	294
<b>M</b>	
\makeatletter .....	78, 200
\makebox .....	283, 284, 325–327, 337, 343, 344

makeglossaries	30, 43, 56, 57, 66, 173, 180, 201
\makeglossaries	..... 7, 8, 31–35, 39, 72, 177, 183, 185, 202
\makeglossary	..... 34, 35
makeindex	..... 400
makeindex	.... 10, 14, 30, 33, 40, 43–45, 49, 66, 67, 70, 94, 120, 123, 165, 169, 171, 173, 176, 187, 191–194, 197, 219, 330, 331 delim_n ..... 45 delim_r ..... 45 page_compositor ..... 43 special characters ..... 121, 122, 165
\makenoidxglossaries	..... ..... 7, 8, 31, 32, 72, 181, 185, 186
\MakeTextUppercase	..... 4
\MakeUppercase	..... 364–366, 373, 374
\marginpar	..... 6
\markboth	..... 46
\mbox	..... 164, 282, 305, 306, 325, 337
memoir class	..... 187
\memUhead	..... 46
\MessageBreak	.. 21, 34, 65, 199, 349, 398, 399
mfirstuc package	..... 1, 156
\mfistucMakeUppercase	.... 4, 47, 83, 105–107, 109–112, 135–148, 150, 152, 154, 231, 238–241, 251, 256, 274, 275, 367–371, 374–379, 385, 386, 388
\midrule	..... 291, 292
\month	..... 167, 172, 331, 334
multicol package	..... 300
 <b>N</b>	
\n	..... 172, 334
\NeedsTeXFormat	... 4, 269, 329, 335, 349, 398
\new@glossaryentry	..... 77, 183
\new@ifnextchar	..... 66, 82, 83, 101– 103, 129–133, 135–154, 227–229, 271–275
\newacronym	..... 225, 230, 246, 248, 250, 252, 254, 257, 259, 261
\newacronymhook	..... 231, 246, 248, 250, 252, 254, 257, 259, 261, 390
\newacronymstyle	... 233–238, 240, 242, 243
\newcommand	..... 6–24, 26, 27, 29, 30, 32–34, 36–40, 42–62, 65–72, 74–87, 92–99, 101–104, 108, 110, 112, 113, 115–118, 120, 121, 127–167, 173, 176–180, 182, 185–190, 192–200, 202– 207, 209–222, 224–234, 243, 245–267, 270–278, 280, 281, 294, 301, 319, 320, 325, 335, 351, 355–359, 381, 390, 395–397
\newcount	..... 15, 75
\newcounter	..... 12, 13
\newenvironment	..... 216
\newglossary	..... 17, 18, 36, 37, 67, 181
\newglossaryentry	..... ..... 7, 37, 74, 77, 99, 231, 245, 247, 249, 251, 253, 256, 258, 260, 391–394
\newglossaryentry options	
access	..... 353, 355
counter	..... 71
description	..... ..... 29, 69, 74, 77, 87, 139, 157, 226, 254, 352
descriptionaccess	..... 356, 360
descriptionplural	..... 139, 352
descriptionpluralaccess	..... 356, 360
entrycounter	..... 213
first	.... 70, 90, 128, 135, 158, 252, 257, 351
firstaccess	..... 356, 360
firstplural	.... 70, 137, 158, 351
firstpluralaccess	..... 356, 360
format	..... 168
long	.... 110, 161, 352
longaccess	..... 357, 361
longplural	.... 161, 352
longpluralaccess	..... 357, 361
name	69, 70, 74, 77, 87, 138, 155, 196, 351, 353
nonumberlist	..... 72
parent	..... 72, 77
plural	.... 70, 90, 136, 351
pluralaccess	..... 356, 360
prefix	..... 269
prefixfirst	..... 269
prefixfirstplural	.... 270
prefixplural	.... 270
see	.... 7, 10, 71, 77, 181, 183
short	.... 110, 161, 352, 358
shortaccess	..... 356, 361, 390
shortpl	.... 358
shortplural	.... 161, 352
shortpluralaccess	..... 356, 361
sort	.... 70, 159, 187, 219
symbol	.... 69, 70, 140, 248, 250, 252, 257, 288, 310, 351, 353, 358
symbolaccess	..... 356, 360
symbolplural	.... 141, 351
symbolpluralaccess	..... 356, 360
text	.... 70, 128, 135, 157, 248, 252, 351
textaccess	..... 355, 359
type	.... 17, 71, 103, 159

user1 .....	142, 160, 352	\ns@ACRshort .....	148
user1access .....	357, 361	\ns@Acrshort .....	147, 148
user2 .....	143, 160, 352	\ns@acrshort .....	147
user2access .....	357, 361	\ns@ACRshortpl .....	150
user3 .....	144, 160, 352	\ns@Acrshortpl .....	149
user3access .....	357, 361	\ns@acrshortpl .....	149
user4 .....	144, 160, 353	\ns@newglossary .....	66
user4access .....	357, 362	\null .....	120–127, 173–176, 201
user5 .....	145, 160, 353	\number .....	15, 78, 90, 101, 189, 192, 218, 350
user5access .....	357, 362	\numberline .....	49
user6 .....	146, 161, 353	\numexpr .....	101
user6access .....	357, 362		
\newglossarystyle .....	278, 281–293, 295–318, 320–325, 327, 328	<b>O</b>	
\newif .....	4, 5, 19, 27, 30, 32, 191	\o .....	24
\newlength .....	127, 284, 295, 306, 313, 323	\OE .....	24
\newrobustcmd .....	7, 61–63, 77, 82, 83, 101–103, 116, 129–154, 156–162, 164, 165, 196, 226–229, 270–275, 325, 359–362	\oe .....	24
\newterm .....	37	\openout .....	32, 77, 167, 171, 177, 331, 334
\newtoks .....	122, 177, 230	\OR .....	260
\newwrite .....	32, 77, 167, 171, 177, 180	\or .....	5, 6, 8, 9, 28, 35, 212, 321, 341
\nfss@text .....	6	\org@glossaryentrynumbers .....	200, 214
ngerman package .....	173	\org@glossarytitle .....	199
\noalign .....	294	\org@glspostdescription .....	42
\nobreak .....	282, 295, 337	\org@ifKV@glslink@hyper .....	118
\noexpand .....	19, 39, 50, 51, 78, 91, 92, 113, 114, 119–121, 126, 127, 163, 173–179, 181, 190, 191, 193, 197, 201, 204, 205, 216, 218, 225, 226, 231, 245, 247, 249, 251, 253, 254, 256, 258, 260, 261, 329, 349, 350, 391–395	\outputpenalty .....	294
\nohyperpage .....	223		
\noindent .....	218, 302–304, 306, 323, 324	<b>P</b>	
\noist .....	334, 335	\p@ .....	281, 301, 319, 320
\nopostdesc .....	37, 43, 86, 200, 336	\p@gls@hyp@opt .....	115
\normalbaselineskip .....	294	package options:	
\ns@ACRfull .....	228	acronym .....	17, 18, 39, 198, 226
\ns@Acrfull .....	227	true .....	18
\ns@acrfull .....	227	counter .....	20
\ns@ACRfullpl .....	229	debug .....	
\ns@Acrlfullpl .....	229	showaccsupp .....	7
\ns@acrlfullpl .....	228	showtargets .....	6, 7
\ns@ACRlong .....	152	description .....	252, 253
\ns@Acrlong .....	151	disablemakegloss .....	32
\ns@acrlong .....	150, 151	dua .....	250, 252, 253
\ns@ACRlongpl .....	154	entrycounter .....	12, 213, 214
\ns@Acrlongpl .....	153	true .....	12
\ns@acrlongpl .....	152	esclocations .....	423
		false .....	10
		footnote .....	129–134, 248, 250, 252, 254
		hyperfirst .....	
		false .....	129–134
		index .....	36
		indexonlyfirst .....	407

kernelglossredefs		
nowarn	35	
makeindex	169, 268	
nogroupskip	285, 287, 288, 291, 292, 296, 297, 299, 307, 309, 311, 314, 316, 318	
nolist	261	
nolong	262, 284	
nomain	17	
nonumberlist	10	
nosuper	262	
notree	262	
nowarn	5	
numberline	8	
record	271	
sanitize	25, 69, 155, 157	
sanitizesort	22	
savewrites	33, 404 false	177
true	180, 186	
section	8, 47	
sort		
def	13, 14	
none	13	
standard	13	
use	13, 14, 423	
style	9, 261, 262	
subentrycounter	13, 213, 214	
toc	8 true	8
translate	28 false	27
translator	27	
xindy	30, 31, 169, 267, 268	
\PackageError	7, 8, 16, 33, 39, 50, 57, 60, 61, 65, 71, 74, 81–86, 88–90, 99, 114, 155, 176, 177, 181, 183, 185, 205–207, 212, 213, 221, 222, 232, 233, 249, 250, 255, 258, 335, 362	
\PackageInfo	5, 6, 32, 177, 188	
\PackageWarning	5, 6, 20	
\PackageWarningNoLine	5, 6, 21, 398, 399	
\pagegoal	294	
\pagelistname	42, 287, 289, 291, 292, 298, 300, 309–313, 316–319	
\par	43, 218, 219, 281–283, 301, 303–306, 319, 320, 322–328, 337, 342–344	
\parindent	301–306, 320, 322–324, 326–328, 342–344	
\parskip	301–304, 320, 322, 323	
\PassOptionsToPackage	269, 349	
\penalty	294	
\phantomsection	48	
polyglossia package	27, 41	
\printglossaries	181	
\printglossary	18, 21, 36, 37, 181, 198, 213	
\printglossary options		
entrycounter	213	
nogroupskip	212	
nonumberlist	213	
nopostdot	212	
numberedsection	212	
style	211	
subentrycounter	213	
title	211	
toctitle	211	
type	17, 197, 211	
\printindex	37	
\printnoidxglossaries	183	
\printnoidxglossary		
182, 183, 186, 198, 205, 206, 213		
\printnoidxglossary options		
sort	213	
\printnumbers	36	
\printsymbols	36	
\ProcessOptions	269, 349	
\ProcessOptionsX	37	
\protect	49, 112, 233–235, 240, 241, 247, 251, 253, 371, 372, 381, 382, 387, 388	
\protected@csedef	84	
\protected@csxdef	83	
\protected@edef		
... 7, 9, 51, 53, 56, 58, 89, 92, 94, 105– 107, 112, 113, 119, 120, 163, 188, 191, 193, 212, 216, 218, 221, 222, 231, 255, 260, 270, 276, 330, 331, 349, 350, 358, 359		
\protected@write	65, 67, 168, 169, 180, 183, 186, 188, 197, 277, 331	
\protected@xdef		
... 14–16, 19, 24, 76, 94, 95, 193, 354, 355		
\providecommand	18, 31, 39, 40, 47, 65, 101, 128, 169, 180, 183, 186, 201, 216, 218, 271, 281, 301, 319, 350	
\ProvidesFile	40	
\ProvidesPackage		
... 4, 269, 276, 278, 281, 284, 290, 295, 300, 306, 313, 319, 329, 335, 349, 398		
\r	24	

```

\raggedright ..... 293–300, 314–319
\raisebox ..... 127
\ref ..... 215
\refstepcounter ..... 214, 215
\relax ..... 5, 9, 11, 15–18, 27, 28, 32,
            33, 35–38, 52, 65, 70, 72, 76, 78, 89, 92,
            96, 99, 100, 115, 116, 119, 120, 122–127,
            156, 170, 171, 173–176, 180, 181, 183–
            185, 189, 193–196, 199, 200, 208, 209,
            211, 212, 220–222, 262, 277, 281, 294,
            301, 305, 306, 319, 321–328, 330, 333–
            335, 341–344, 353, 354, 372–374, 391–394
\renewacronymstyle . 381–384, 387, 389, 390
\renewcommand ..... 4–6, 8–11, 13, 16–
            18, 20–22, 26, 28–31, 33, 35, 37, 41–44,
            57, 69, 71–73, 86, 99, 100, 163, 164, 166,
            167, 173, 174, 180–184, 200, 202, 212,
            230, 231, 233–243, 246, 248, 250, 252,
            254, 255, 257, 259, 261, 279–289, 291,
            292, 294–311, 314–318, 320–330, 335–
            343, 345–348, 350, 353, 354, 362, 366,
            369, 371, 372, 380–384, 386–390, 392–394
\renewenvironment ..... 216,
            278, 281, 285–290, 293–320, 322, 323, 325
\RequireGlossariesLang ..... 42, 398, 399
\RequirePackage ..... 4,
            10, 11, 27, 28, 37, 41, 261, 268, 269,
            284, 290, 291, 295, 301, 306, 313, 350, 398
\restorecounters@ ..... 119
\romannumeral ..... 189, 192, 325–327, 343

S
\s@gls@hyp@opt ..... 115
\s@GlsSetXdyFirstLetterAfterDigits 166
\s@GlsSetXdyNumberGroupOrder .. 166, 167
\s@ifglossaryexists ..... 59, 198
\s@newglossary ..... 66
\savecounters@ ..... 119
\seename ..... 196
\SetAcronymStyle ..... 29, 30
\setbool ..... 26
\setbox ..... 294, 295
\setcounter ..... 214
\SetCustomDisplayStyle ..... 261
\SetDefaultAcronymDisplayStyle .... 246
\SetDefaultAcronymStyle ..... 259
\SetDescriptionAcronymDisplayStyle 252
\SetDescriptionAcronymStyle ..... 259
\SetDescriptionDUAAcronymDisplayStyle
            ..... 250

\SetDescriptionDUAAcronymStyle ..... 259
\SetDescriptionFootnoteAcronymDisplayStyle
            ..... 248
\SetDescriptionFootnoteAcronymStyle 259
\SetDUADisplayStyle ..... 259
\SetDUAStyle ..... 260
\setentrycounter ..... 51, 170, 211, 329
\SetFootnoteAcronymDisplayStyle ... 254
\SetFootnoteAcronymStyle ..... 260
\SetGenericNewAcronym ..... 232
\setglossarystyle ..... 199, 221,
            262, 282–294, 296–318, 321–324, 327, 328
\setglossentrycompatibility ... 212, 221
\setkeys 27, 31, 38, 47, 88, 118, 164, 165, 199,
            230, 246, 248, 250, 252, 254, 257, 259, 261
\setlength ..... 284, 295, 301–
            304, 306, 313, 320, 322, 323, 327, 343, 344
\SetSmallAcronymDisplayStyle ..... 257
\SetSmallAcronymStyle ..... 260
\settoheight ..... 127
\settowidth ..... 234, 325–327, 343
\sfcode ..... 11
\show ..... 262–267, 396, 397
\SmallNewAcronymDef ..... 257
\space ..... 7, 8, 31–35, 39, 50, 54,
            55, 57, 58, 72, 74, 99, 101–103, 112, 113,
            115, 162, 168–172, 176–179, 181, 183,
            185, 186, 196, 199, 202, 215, 221, 227,
            233–243, 251, 255, 256, 278, 280–283,
            285, 296, 307, 314, 320–322, 324, 326,
            327, 329, 330, 332–334, 336–339, 341–
            345, 347, 371, 372, 381–384, 386, 388–391
\spacefactor ..... 11
\SS ..... 25
\ss ..... 25
\string ..... 7, 8, 16, 21, 31–35, 39, 49–51,
            53–58, 65, 67, 72, 74, 78, 79, 82, 83, 93,
            94, 99, 101–103, 113, 115, 120–124, 126,
            162, 165, 166, 168–173, 175–177, 180–
            183, 185, 186, 194, 195, 199, 201, 202,
            205, 206, 213, 218, 219, 221, 277, 329–335
\strut ..... 219, 282–
            286, 288, 296, 297, 299, 307, 309, 310,
            314, 316, 317, 324, 336–340, 342, 345–348
\subglossentry 93, 200, 209, 218, 219, 279,
            281, 283–286, 288, 296, 297, 299, 307,
            309, 310, 314, 316, 317, 321, 322, 324, 326
\subitem ..... 301, 320, 321, 341
\subsubitem ..... 301, 320, 321, 341

```

supertabular package .....	<a href="#">11</a> , <a href="#">262</a> , <a href="#">306</a> , <a href="#">313</a>
\symbolname .....	<a href="#">42</a> , <a href="#">288</a> , <a href="#">289</a> , <a href="#">292</a> , <a href="#">300</a> , <a href="#">311–313</a> , <a href="#">318</a> , <a href="#">319</a>
T	
\t .....	<a href="#">24</a>
\tablehead .....	<a href="#">307–319</a>
\tabletail .....	<a href="#">307–319</a>
\tabularnewline .....	<a href="#">285–289</a> , <a href="#">291</a> , <a href="#">292</a> , <a href="#">296–300</a> , <a href="#">307–319</a> , <a href="#">339</a> , <a href="#">340</a> , <a href="#">345</a> , <a href="#">346</a>
\texorpdfstring .....	<a href="#">159</a>
\textbar .....	<a href="#">278</a>
\textbf .....	<a href="#">218</a> , <a href="#">225</a> , <a href="#">319</a> , <a href="#">341–344</a>
textcase package .....	<a href="#">4</a>
\textit .....	<a href="#">225</a>
\textmd .....	<a href="#">225</a>
\textrm .....	<a href="#">224</a>
\textsc .....	<a href="#">225</a> , <a href="#">235</a> , <a href="#">236</a> , <a href="#">242</a> , <a href="#">248</a> , <a href="#">252</a> , <a href="#">255</a> , <a href="#">257</a> , <a href="#">389</a>
\textsf .....	<a href="#">224</a>
\textsl .....	<a href="#">225</a>
\textsmaller .....	<a href="#">235</a> , <a href="#">236</a> , <a href="#">242</a> , <a href="#">248</a> , <a href="#">252</a> , <a href="#">255</a> , <a href="#">257</a> , <a href="#">389</a>
\texttt .....	<a href="#">224</a>
\textulc .....	<a href="#">226</a>
\textup .....	<a href="#">225</a> , <a href="#">226</a>
\TH .....	<a href="#">25</a>
\th .....	<a href="#">25</a>
\the .....	<a href="#">39</a> , <a href="#">41</a> , <a href="#">51</a> , <a href="#">56</a> , <a href="#">58</a> , <a href="#">66</a> , <a href="#">122–127</a> , <a href="#">167</a> , <a href="#">172</a> , <a href="#">174</a> , <a href="#">175</a> , <a href="#">186–188</a> , <a href="#">192</a> , <a href="#">197</a> , <a href="#">208</a> , <a href="#">216</a> , <a href="#">218</a> , <a href="#">224</a> , <a href="#">231</a> , <a href="#">233</a> , <a href="#">234</a> , <a href="#">239</a> , <a href="#">240</a> , <a href="#">245</a> , <a href="#">247</a> , <a href="#">249</a> , <a href="#">251</a> , <a href="#">253</a> , <a href="#">254</a> , <a href="#">256</a> , <a href="#">258</a> , <a href="#">260</a> , <a href="#">261</a> , <a href="#">329</a> , <a href="#">331</a> , <a href="#">334</a> , <a href="#">350</a> , <a href="#">381</a> , <a href="#">382</a> , <a href="#">386</a> , <a href="#">387</a> , <a href="#">390–395</a>
\the@numberlist .....	<a href="#">163</a>
\theglossary .....	<a href="#">5</a>
\theglossaryentry .....	<a href="#">12</a> , <a href="#">215</a>
\theglossarysubentry .....	<a href="#">13</a> , <a href="#">215</a>
\theglentrycounter .....	<a href="#">119</a> , <a href="#">190</a> , <a href="#">193</a> , <a href="#">330</a> , <a href="#">331</a>
\theH .....	<a href="#">195</a>
\theHglossaryentry .....	<a href="#">12</a>
\theHglossarysubentry .....	<a href="#">13</a>
\theHglsentrycounter .....	<a href="#">119</a> , <a href="#">190</a> , <a href="#">193</a>
\thesection .....	<a href="#">38</a>
>this@dialect .....	<a href="#">41</a> , <a href="#">42</a> , <a href="#">398</a> , <a href="#">399</a>
\tiny .....	<a href="#">6</a>
\toks@ .....	<a href="#">39</a> , <a href="#">41</a> , <a href="#">51</a> , <a href="#">56</a> , <a href="#">58</a> , <a href="#">66</a> , <a href="#">122–127</a> , <a href="#">173–175</a> , <a href="#">197</a> , <a href="#">216</a> , <a href="#">218</a> , <a href="#">224</a> , <a href="#">329</a> , <a href="#">349</a> , <a href="#">350</a>
\toprule .....	<a href="#">291</a> , <a href="#">292</a>
tracklang package .....	<a href="#">41</a> , <a href="#">398</a>
\trans@languages .....	<a href="#">41</a>
\translate .....	<a href="#">41</a> , <a href="#">42</a>
\translatelet .....	<a href="#">17</a> , <a href="#">18</a> , <a href="#">36</a>
translator package .....	<a href="#">17</a> , <a href="#">18</a> , <a href="#">27</a> , <a href="#">36</a> , <a href="#">40–42</a> , <a href="#">197</a>
\triangleright .....	<a href="#">6</a>
\ttfamily .....	<a href="#">6</a>
\TX@trial .....	<a href="#">95</a>
\typeout .....	<a href="#">21</a>
U	
\u .....	<a href="#">24</a>
\uccode .....	<a href="#">208</a>
\undef .....	<a href="#">72</a> , <a href="#">78</a> , <a href="#">197</a>
\unskip .....	<a href="#">86</a> , <a href="#">283</a> , <a href="#">284</a> , <a href="#">337</a>
\unvbox .....	<a href="#">294</a> , <a href="#">295</a>
\usedictionary .....	<a href="#">41</a>
\usepackage .....	<a href="#">205</a> , <a href="#">206</a>
\UTFviii@two@octets .....	<a href="#">25</a>
\UTFviii@two@octets@combine .....	<a href="#">25</a>
V	
\v .....	<a href="#">24</a>
\vbox .....	<a href="#">294</a> , <a href="#">295</a>
\vsizer .....	<a href="#">294</a> , <a href="#">295</a>
\vskip .....	<a href="#">281</a> , <a href="#">294</a> , <a href="#">301</a> , <a href="#">319</a>
\vss .....	<a href="#">294</a> , <a href="#">295</a>
W	
\warn@nomakeglossaries .....	<a href="#">181–183</a>
\warn@noprintglossary .....	<a href="#">181–183</a> , <a href="#">200</a>
\write .....	<a href="#">32</a> , <a href="#">78</a> , <a href="#">79</a> , <a href="#">101</a> , <a href="#">167–173</a> , <a href="#">177–179</a> , <a href="#">183</a> , <a href="#">187</a> , <a href="#">201</a> , <a href="#">331–335</a>
\writeist .....	<a href="#">176</a> , <a href="#">177</a> , <a href="#">335</a>
X	
\x .....	<a href="#">224</a>
\xatlevel@ .....	<a href="#">119</a>
\xcapitaliswords .....	<a href="#">159</a>
\xdef .....	<a href="#">89</a> , <a href="#">90</a> , <a href="#">92</a> , <a href="#">200</a> , <a href="#">277</a>
\xglsaccsupp .....	<a href="#">359</a>
\xifinlistcs .....	<a href="#">202</a> , <a href="#">203</a> , <a href="#">206</a>
\xindy .....	<a href="#">400</a>
\xindy .....	<a href="#">10</a> , <a href="#">14</a> , <a href="#">24</a> , <a href="#">30</a> , <a href="#">31</a> , <a href="#">33</a> , <a href="#">43</a> , <a href="#">44</a> , <a href="#">49</a> , <a href="#">52</a> , <a href="#">54</a> , <a href="#">56–58</a> , <a href="#">94</a> , <a href="#">126</a> , <a href="#">166</a> , <a href="#">167</a> , <a href="#">169</a> , <a href="#">187</a> , <a href="#">191</a> , <a href="#">193</a> , <a href="#">194</a> , <a href="#">197</a> , <a href="#">201</a> , <a href="#">219</a> , <a href="#">267</a> , <a href="#">330</a>
\xmakefirstuc .....	<a href="#">105–107</a> , <a href="#">112</a> , <a href="#">113</a> , <a href="#">155</a> , <a href="#">156</a> , <a href="#">270</a>
\xspace .....	<a href="#">226</a>
\xspace package .....	<a href="#">4</a> , <a href="#">225</a>

Y  
\year ..... 167, 172, 331, 334 Z  
\z@ ..... 294