

Documented Code For glossaries v4.23

Nicola L.C. Talbot

Dickimaw Books

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

2016-04-30

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 v4.23: L^AT_EX2e Package to Assist Generating Glossaries”.

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

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

INSTALL Installation instructions.

CHANGES Change log.

README Package summary.

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

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

Contents

1	Main Package Code	4
1.1	Package Definition	4
1.2	Package Options	5
1.3	Predefined Text	29
1.4	Xindy	39
1.5	Loops and conditionals	48
1.6	Defining new glossaries	54
1.7	Defining new entries	59
1.8	Resetting and unsetting entry flags	83
1.9	Keeping Track of How Many Times an Entry Has Been Unset	86
1.10	Loading files containing glossary entries	91
1.11	Using glossary entries in the text	91
1.12	Adding an entry to the glossary without generating text	151
1.13	Creating associated files	153
1.14	Writing information to associated files	168
1.15	Glossary Entry Cross-References	174
1.16	Displaying the glossary	176
1.17	Acronyms	205
1.18	Predefined acronym styles	210
1.19	Predefined Glossary Styles	241
1.20	Debugging Commands	242
1.21	Compatibility with version 2.07 and below	247
2	Prefix Support (glossaries-prefix Code)	249
3	Glossary Styles	256
3.1	Glossary hyper-navigation definitions (glossary-hypernav package)	256
3.2	In-line Style (glossary-inline.sty)	258
3.3	List Style (glossary-list.sty)	260
3.4	Glossary Styles using longtable (the glossary-long package)	264
3.5	Glossary Styles using longtable and booktabs (the glossary-longbooktabs) package	270
3.6	Glossary Styles using longtable (the glossary-longragged package)	274
3.7	Glossary Styles using multicol (glossary-mcols.sty)	280
3.8	Glossary Styles using supertabular environment (glossary-super package)	286
3.9	Glossary Styles using supertabular environment (glossary-superragged package)	292
3.10	Tree Styles (glossary-tree.sty)	298

4 Backwards Compatibility	308
4.1 glossaries-compatible-207	308
4.2 glossaries-compatible-307	314
5 Accessibility Support (glossaries-accsupp Code)	328
5.1 Defining Replacement Text	329
5.2 Accessing Replacement Text	332
5.3 Displaying the Glossary	348
5.4 Acronyms	349
5.5 Debugging Commands	364
6 Multi-Lingual Support	366
6.1 Polyglossia Captions	366
Glossary	368
Change History	369
Index	391

1 Main Package Code

1.1 Package Definition

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

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries}[2016/04/30 v4.23 (NLCT)]
```

Required packages:

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

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

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

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

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

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

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

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

Check if doc has been loaded.

```
f@gls@docloaded
```

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

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

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

org@theglossary

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

@endtheglossary

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

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

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

End of doc stuff.

```
31 \fi
```

1.2 Package Options

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

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

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

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

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

```
34 \ifcsundef{chapter}%
35   {\newcommand*\@@glossarysec{section}}%
36   {\newcommand*\@@glossarysec{chapter}}
```

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

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

Determine whether or not to use numbered sections.

`\glossarysecstar`

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

`\glossaryseclabel`

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

`\glsautoprefix`

Prefix to add before label if automatically generated:

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

`\numberedsection`

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

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

`\@glossary@default@style`

```
62 \newcommand*\@glossary@default@style{list}
```

`\style`

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

```
63 \define@key{glossaries.sty}{style}{%
64   \renewcommand*\@glossary@default@style{#1}%
65 }
```

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

s@declareoption

```
66 \newcommand*{\@gls@declareoption}[2]{%
67   \DeclareOptionX{#1}{#2}%
68   \DeclareOption{#1}{#2}%
69 }
```

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

aryentrynumbers

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

nonumberlist

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

```
71 \@gls@declareoption{nonumberlist}{%
72   \renewcommand*{\glossaryentrynumbers}[1]{\gls@save@numberlist{#1}}%
73 }
```

savenumberlist

Provide means to store the number list for entries.

```
74 \define@boolkey{glossaries.sty}[gls]{savenumberlist}[true]{%
75   \glssavenumberlistfalse
```

eautionumberlist

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

eautionumberlist

Automatically activates number list for entries containing the see key.

```
77 \@gls@declareoption{seeautonumberlist}{%
78   \renewcommand*{\@glo@seeautonumberlist}{%
79     \def\@glo@prefix{\glsnextpages}%
80   }%
81 }
```

\@gls@loadlong

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

nolong

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

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

\@gls@loadsuper

The package isn't loaded if isn't installed.

```
84 \IfFileExists{supertabular.sty}{%
85   \newcommand*{\@gls@loadsuper}{\RequirePackage{glossary-super}}}%
86   \newcommand*{\@gls@loadsuper}{}}
```

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

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

\@gls@loadlist

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

nolist This option prevents from being loaded (to reduce overheads if required). Naturally, the styles defined in will not be available if this option is used.

```
89 \@gls@declareoption{nolist}{\renewcommand*{\@gls@loadlist}{}}
```

\@gls@loadtree

```
90 \newcommand*{\@gls@loadtree}{\RequirePackage{glossary-tree}}
```

notree This option prevents from being loaded (to reduce overheads if required). Naturally, the styles defined in will not be available if this option is used.

```
91 \@gls@declareoption{notree}{\renewcommand*{\@gls@loadtree}{}}
```

nostyles Provide an option to suppress all the predefined styles (in the event that the user has custom styles that are not dependent on the predefined styles).

```
92 \@gls@declareoption{nostyles}{%
93   \renewcommand*{\@gls@loadlong}{}%
94   \renewcommand*{\@gls@loadsuper}{}%
95   \renewcommand*{\@gls@loadlist}{}%
96   \renewcommand*{\@gls@loadtree}{}%
97   \let\@glossary@default@style\relax
98 }
```

postdescription The description terminator is given by `\glspostdescription` (except for the 3 and 4 column styles). This is a full stop by default. The spacefactor is adjusted in case the description ends with an upper case letter. (Patch provided by Michael Pock.)

```
99 \newcommand*{\glspostdescription}{%
100   \ifglsnopostdot\else.\spacefactor\sfcode'\. \fi
101 }
```

nopostdot Boolean option to suppress post description dot

```
102 \define@boolkey{glossaries.sty}[gls]{nopostdot}[true]{}
103 \glsnopostdotfalse
```

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

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

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

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

```



```

107 \ifclassloaded{memoir}
108 {%
109   \glsucmarktrue
110 }%
111 {%
112   \glsucmarkfalse
113 }

entrycounter  Defines a counter that can be used in the standard glossary styles to number each (main)
               entry. If true, this will define a counter called glossaryentry.
114 \define@boolkey{glossaries.sty}[gls]{entrycounter}[true]{}
115 \glsentrycounterfalse

rycounterwithin  This option can be used to set a parent counter for glossaryentry. This option automatically
                  sets entrycounter=true.
116 \define@key{glossaries.sty}{counterwithin}{%
117   \renewcommand*{\@gls@counterwithin}{#1}%
118   \glsentrycountertrue
119 }

s@counterwithin  The default value is no parent counter:
120 \newcommand*{\@gls@counterwithin}{}

subentrycounter  Define a counter that can be used in the standard glossary styles to number each level 1 entry.
                  If true, this will define a counter called glossarysubentry.
121 \define@boolkey{glossaries.sty}[gls]{subentrycounter}[true]{}
122 \glssubentrycounterfalse

efault@sorttype  Initialise default sort for \printnoidxglossary
123 \newcommand*{\@glo@default@sorttype}{standard}

sort  Define the sort method: sort=standard (default), sort=def (order of definition) or sort=use
      (order of use).
124 \define@choicekey{glossaries.sty}{sort}{standard,def,use}{%
125   \renewcommand*{\@glo@default@sorttype}{#1}%
126   \csname @gls@setupsort@#1\endcsname
127 }

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

                  Allow user to hook into sort mechanism. The first argument <sort cs> is the temporary control
                  sequence containing the sort value before it has been sanitized and had makeindex/xindy
                  special characters escaped.
128 \newcommand*{\glsprestandardsort}[3]{%
129   \glsdosanitizesort
130 }

```

upsort@standard Set up the macros for default sorting.

```

131 \newcommand*{\@gls@setupsort@standard}{%
    Store entry information when it's defined.
132   \def\do@glo@storeentry{\@glo@storeentry}%
    No count register required for standard sort.
133   \def\@gls@defsortcount##1{%
    Sort according to sort key (\@glo@sort) if provided otherwise sort according to the entry's
    name (\@glo@name). (First argument glossary type, second argument entry label.)
134   \def\@gls@defsort##1##2{%
135     \ifx\@glo@sort\@glsdefaultsort
136       \let\@glo@sort\@glo@name
137     \fi
138     \let\glsdosanitizesort\@gls@sanitizesort
139     \glsprestandardsort{\@glo@sort}{##1}{##2}%
140     \expandafter\protected@xdef\csname glo@##2@sort\endcsname{\@glo@sort}%
141   }%
    Don't need to do anything when the entry is used.
142   \def\@gls@setsort##1{%
143 }
    Set standard sort as the default:
144 \@gls@setupsort@standard

```

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

```

145 \newcommand*{\glssortnumberfmt[1]}{%
146   \ifnum#1<100000 0\fi
147   \ifnum#1<10000 0\fi
148   \ifnum#1<1000 0\fi
149   \ifnum#1<100 0\fi
150   \ifnum#1<10 0\fi
151   \number#1%
152 }

```

s@setupsort@def Set up the macros for order of definition sorting.

```

153 \newcommand*{\@gls@setupsort@def}{%
    Store entry information when it's defined.
154   \def\do@glo@storeentry{\@glo@storeentry}%
    Defined count register associated with the glossary.
155   \def\@gls@defsortcount##1{%
156     \expandafter\global
157     \expandafter\newcount\csname glossary@##1@sortcount\endcsname
158   }%

```

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

```
159 \def\@gls@defsort##1##2{%
160   \expandafter\global\expandafter
161   \advance\csname glossary@##1@sortcount\endcsname by 1\relax
162   \expandafter\protected@xdef\csname glo@##2@sort\endcsname{%
163     \expandafter\glssortnumberfmt
164     {\csname glossary@##1@sortcount\endcsname}}%
165 }%
```

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

```
166 \def\@gls@setsort##1{%
167 }
```

s@setupsort@use Set up the macros for order of use sorting.

```
168 \newcommand*{\@gls@setupsort@use}{%
```

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

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

Defined count register associated with the glossary.

```
170 \def\@gls@defsortcount##1{%
171   \expandafter\global
172   \expandafter\newcount\csname glossary@##1@sortcount\endcsname
173 }%
```

Initialise the sort key to empty.

```
174 \def\@gls@defsort##1##2{%
175   \expandafter\gdef\csname glo@##2@sort\endcsname{%
176 }%
```

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

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

Get the parent, if one exists

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

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

```
179 \ifx\@glo@parent\@empty
180 \else
181   \expandafter\@gls@setsort\expandafter{\@glo@parent}%
182 \fi
```

Set index information for this entry

```
183 \edef\@glo@type{\csname glo@##1@type\endcsname}%
184 \edef\@gls@tmp{\csname glo@##1@sort\endcsname}%
185 \ifx\@gls@tmp\@empty
186   \expandafter\global\expandafter
187   \advance\csname glossary@\@glo@type @sortcount\endcsname by 1\relax
188   \expandafter\protected@xdef\csname glo@##1@sort\endcsname{%
189     \expandafter\glssortnumberfmt
190     {\csname glossary@\@glo@type @sortcount\endcsname}}%
```

```

191      \@glo@storeentry{##1}%
192      \fi
193    }%
194 }

```

`\glsdefmain` Define the main glossary. This will be the first glossary to be displayed when using `\printglossaries`. The default extensions conflict if used with `doc`, so provide different extensions if `doc` loaded. (If these extensions are inappropriate, use `nomain` and manually define the main glossary with the desired extensions.)

```

195 \newcommand*{\glsdefmain}{%
196   \if@gls@docloaded
197     \newglossary[glg2]{main}{gls2}{glo2}{\glossaryname}%
198   \else
199     \newglossary{main}{gls}{glo}{\glossaryname}%
200   \fi

```

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

```

201   \newcommand*{\gls@tr@set@main@toctitle}{%
202     \translatelet{\glossarytoctitle}{Glossary}%
203   }%
204 }

```

Keep track of the default glossary. This is initialised to the main glossary, but can be changed if for some reason you want to make a secondary glossary the main glossary. This affects any commands that can optionally take a glossary name as an argument (or as the value of the type key in a key-value list). This was mainly done so that `\loadglsentries` can temporarily change `\glsdefaulttype` while it loads a file containing new glossary entries (see [section 1.10](#)).

`\glsdefaulttype`

```

205 \newcommand*{\glsdefaulttype}{main}

```

Keep track of which glossary the acronyms are in. This is initialised to `\glsdefaulttype`, but is changed by the acronym package option.

`\acronymtype`

```

206 \newcommand*{\acronymtype}{\glsdefaulttype}

```

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

```

207 \@gls@declareoption{nomain}{%
208   \let\glsdefaulttype\relax
209   \renewcommand*{\glsdefmain}{}%
210 }

```

`acronym` The `acronym` option sets an associated conditional which is used in [section 1.17](#) to determine whether or not to define a separate glossary for acronyms.

```

211 \define@boolkey{glossaries.sty}[gls]{acronym}[true]{%
212   \ifglsacronym

```

```

213 \renewcommand{\@gls@do@acronymsdef}{%
214 \DeclareAcronymList{acronym}%
215 \newglossary[alg]{acronym}{acr}{acn}{\acronymname}%
216 \renewcommand*{\acronymtype}{acronym}%

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

217 \newcommand*{\gls@tr@set@acronym@toctitle}{%
218 \translatelet{\glossarytoctitle}{Acronyms}%
219 }%
220 }%
221 \else
222 \let\@gls@do@acronymsdef\relax
223 \fi
224 }

```

`\printacronyms` Define `\printacronyms` at the start of the document if acronym is set and compatibility mode isn't on and `\printacronyms` hasn't already been defined.

```

225 \AtBeginDocument{%
226 \ifglsacronym
227 \ifbool{glscompatible-3.07}%
228 {}%
229 {%
230 \providecommand*{\printacronyms}[1][ ]{%
231 \printglossary[type=\acronymtype,#1]}%
232 }%
233 \fi
234 }

```

`@do@acronymsdef` Set default value

```

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

```

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

```

236 \@gls@declareoption{acronyms}{%
237 \glsacronymtrue
238 \renewcommand{\@gls@do@acronymsdef}{%
239 \DeclareAcronymList{acronym}%
240 \newglossary[alg]{acronym}{acr}{acn}{\acronymname}%
241 \renewcommand*{\acronymtype}{acronym}%

```

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

```

242 \newcommand*{\gls@tr@set@acronym@toctitle}{%
243 \translatelet{\glossarytoctitle}{Acronyms}%
244 }%
245 }%
246 }

```

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

```

247 \newcommand*{\@glsacronymlists}{}

```

dtoacronymlists

```

248 \newcommand*{\@addtoacronymlists}[1]{%
249   \ifx\@glsacronymlists\@empty
250     \protected@xdef\@glsacronymlists{#1}%
251   \else
252     \protected@xdef\@glsacronymlists{\@glsacronymlists,#1}%
253   \fi
254 }
```

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

```

255 \newcommand*{\DeclareAcronymList}[1]{%
256   \glsIfListOfAcronyms{#1}{\@addtoacronymlists{#1}}%
257 }
```

IfListOfAcronyms

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

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

```

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

Internal command requires label and list to be expanded:

```

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

Not found

```

268   #4%
269   \else
```

Found

```

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

lsisacronymlist Convenient boolean.

```

273 \newif\if@glsisacronymlist
```

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

```

274 \newcommand*{\gls@ckisacronymlist}[1]{%
275   \glsIfListOfAcronyms{#1}%
276   {\@glsisacronymlisttrue}{\@glsisacronymlistfalse}%
277 }
```

SetAcronymLists Sets the “list of acronyms” list. Argument must be a comma-separated list of glossary labels. (Doesn’t check at this point if the glossaries exists.)

```
278 \newcommand*{\SetAcronymLists}[1]{%
279   \renewcommand*{\@glsacronymlists}{#1}%
280 }
```

acronymlists

```
281 \define@key{glossaries.sty}{acronymlists}{%
282   \DeclareAcronymList{#1}%
283 }
```

The default counter associated with the numbers in the glossary is stored in `\glscounter`. This is initialised to the page counter. This is used as the default counter when a new glossary is defined, unless a different counter is specified in the optional argument to `\newglossary` (see [section 1.6](#)).

\glscounter

```
284 \newcommand{\glscounter}{page}
```

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

```
285 \define@key{glossaries.sty}{counter}{%
286   \renewcommand*{\glscounter}{#1}%
287 }
```

gls@nohyperlist

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

lareNoHyperList

```
289 \newcommand*{\GlsDeclareNoHyperList}[1]{%
290   \ifdefempty\@gls@nohyperlist
291     {%
292       \renewcommand*{\@gls@nohyperlist}{#1}%
293     }%
294     {%
295       \appto\@gls@nohyperlist{,#1}%
296     }%
297 }
```

nohypertypes

```
298 \define@key{glossaries.sty}{nohypertypes}{%
299   \GlsDeclareNoHyperList{#1}%
300 }
```

ossariesWarning Prints a warning message.

```
301 \newcommand*{\GlossariesWarning}[1]{%
302   \PackageWarning{glossaries}{#1}%
303 }
```

esWarningNoLine Prints a warning message without the line number.

```
304 \newcommand*\GlossariesWarningNoLine}[1]{%
305   \PackageWarningNoLine{glossaries}{#1}%
306 }
```

nowarn Define package option to suppress warnings

```
307 \@gls@declareoption{nowarn}{%
308   \renewcommand*\GlossariesWarning}[1]{}%
309   \renewcommand*\GlossariesWarningNoLine}[1]{}%
310 }
```

nonglossdefined Issue a warning if overriding \printglossary

```
311 \newcommand*\@gls@warnnonglossdefined{%
312   \GlossariesWarning{Overriding \string\printglossary}%
313 }
```

theglossdefined Issue a warning if overriding theglossary

```
314 \newcommand*\@gls@warnontheglossdefined{%
315   \GlossariesWarning{Overriding 'theglossary' environment}%
316 }
```

noredefwarn Suppress warning on redefinition of \printglossary

```
317 \@gls@declareoption{noredefwarn}{%
318   \renewcommand*\@gls@warnnonglossdefined}{}%
319   \renewcommand*\@gls@warnontheglossdefined}{}%
320 }
```

As from version 3.08a, the only information written to the external glossary files are the label and sort values. Therefore, now, the only sanitize option that makes sense is the one for the sort key. so the sanitize option is now deprecated and there is only a sanitizesort option.

ls@sanitizedesc

```
321 \newcommand*\@gls@sanitizedesc{%
322 }
```

lssetexpandfield `\glssetexpandfield{<field>}`

Sets field to always expand.

```
323 \newcommand*\glssetexpandfield}[1]{%
324   \csdef{gls@assign@#1@field}##1##2{%
325     \@gls@expand@field{##1}{#1}{##2}%
326   }%
327 }
```

setnoexpandfield `\glssetnoexpandfield{<field>}`

Sets field to never expand.

```

328 \newcommand*{\glsetnoexpandfield}[1]{%
329   \csdef{gls@assign@#1@field}##1##2{%
330     \@gls@noexpand@field{##1}{#1}{##2}%
331   }%
332 }

```

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

```

333 \glsetexpandfield{type}

```

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

```

334 \glsetnoexpandfield{desc}

```

escplural@field

```

335 \glsetnoexpandfield{descplural}

```

ls@sanitizename

```

336 \newcommand*{\@gls@sanitizename}{ }

```

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

```

337 \glsetnoexpandfield{name}

```

@sanitizesymbol

```

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

```

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

```

339 \glsetnoexpandfield{symbol}

```

bolplural@field

```

340 \glsetnoexpandfield{symbolplural}

```

Sanitizing stuff:

ls@sanitizesort

```

341 \newcommand*{\@gls@sanitizesort}{%
342   \ifglssanitizesort
343     \@gls@sanitizesort
344   \else
345     \@gls@nosanitizesort
346   \fi
347 }

```

ls@sanitizesort

```

348 \newcommand*\@gls@sanitizesort{%
349   \@onelevel@sanitize\@glo@sort
350 }

```

@nosanitizesort

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

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

```
352 \newcommand*\@gls@noidx@sanitizesort{%
353   \ifdefvoid\@glo@sort
354   {}%
355   {%
356     \expandafter\@@gls@noidx@sanitizesort\@glo@sort\gls@end@sanitizesort
357   }%
358 }
359 \def\@@gls@noidx@sanitizesort#1#2\gls@end@sanitizesort{%
360   \def\@glo@sort{#1#2}%
361   \@onelevel@sanitize\@glo@sort
362 }
```

@nosanitizesort

```
363 \newcommand*{\@@gls@noidx@nosanitizesort}{%
364   \ifdefvoid\@glo@sort
365   {}%
366   {%
367     \expandafter\@@gls@noidx@no@sanitizesort\@glo@sort\gls@end@sanitizesort
368   }%
369 }
370 \def\@@gls@noidx@no@sanitizesort#1#2\gls@end@sanitizesort{%
371   \bgroup
372   \glsnoidxstripaccents
373   \protected@xdef\@glo@sort{#1#2}%
374   \egroup
375   \let\@glo@sort\@glo@sort
376 }
```

idxstripaccents

```
377 \newcommand*\glsnoidxstripaccents{%
378   \let\IeC\@firstofone
379   \let\' \@firstofone
380   \let\' \@firstofone
381   \let\~ \@firstofone
382   \let\" \@firstofone
383   \let\u \@firstofone
384   \let\t \@firstofone
385   \let\d \@firstofone
386   \let\r \@firstofone
387   \let\= \@firstofone
388   \let\.\@firstofone
389   \let\~ \@firstofone
390   \let\v \@firstofone
391   \let\H \@firstofone
392   \let\c \@firstofone
```

```

393 \let\b\@firstofone
394 \def\AE{AE}%
395 \def\ae{ae}%
396 \def\OE{OE}%
397 \def\oe{oe}%
398 \def\AA{AA}%
399 \def\aa{aa}%
400 \def\L{L}%
401 \def\l{l}%
402 \def\O{O}%
403 \def\o{o}%
404 \def\SS{SS}%
405 \def\ss{ss}%
406 \def\th{th}%
407 }

```

Before defining the sanitize package option, The key-value list for the sanitize value needs to be defined. These are all boolean keys. If they are not given a value, assume true.

```

408 \define@boolkey[glS]{sanitize}{description}[true]{%
409   \GlossariesWarning{sanitize={description} package option deprecated}%
410   \ifglS@sanitize@description
411     \glSsetnoexpandfield{desc}%
412     \glSsetnoexpandfield{descplural}%
413   \else
414     \glSsetexpandfield{desc}%
415     \glSsetexpandfield{descplural}%
416   \fi
417 }

418 \define@boolkey[glS]{sanitize}{name}[true]{%
419   \GlossariesWarning{sanitize={name} package option deprecated}%
420   \ifglS@sanitize@name
421     \glSsetnoexpandfield{name}%
422   \else
423     \glSsetexpandfield{name}%
424   \fi
425 }

426 \define@boolkey[glS]{sanitize}{symbol}[true]{%
427   \GlossariesWarning{sanitize={symbol} package option deprecated}%
428   \ifglS@sanitize@symbol
429     \glSsetnoexpandfield{symbol}%
430     \glSsetnoexpandfield{symbolplural}%
431   \else
432     \glSsetexpandfield{symbol}%
433     \glSsetexpandfield{symbolplural}%
434   \fi
435 }

```

sanitizesort

```

436 \define@boolkey{glossaries.sty}[gls]{sanitizesort}[true]{%
437   \ifglssanitizesort
438     \glsssetnoexpandfield{sortvalue}%
439     \renewcommand*{\@gls@noidx@setsanitizesort}{%
440       \glssanitizesorttrue
441       \glsssetnoexpandfield{sortvalue}%
442     }%
443   \else
444     \glsssetexpandfield{sortvalue}%
445     \renewcommand*{\@gls@noidx@setsanitizesort}{%
446       \glssanitizesortfalse
447       \glsssetexpandfield{sortvalue}%
448     }%
449   \fi
450 }

```

Default setting:

```

451 \glssanitizesorttrue
452 \glsssetnoexpandfield{sortvalue}%

```

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

```

453 \newcommand*{\@gls@noidx@setsanitizesort}{%
454   \glssanitizesortfalse
455   \glsssetexpandfield{sortvalue}%
456 }

457 \define@choicekey[gls]{sanitize}{sort}{true,false}[true]{%
458   \setbool{glssanitizesort}{#1}%
459   \ifglssanitizesort
460     \glsssetnoexpandfield{sortvalue}%
461   \else
462     \glsssetexpandfield{sortvalue}%
463   \fi
464   \GlossariesWarning{sanitize={sort} package option
465     deprecated. Use sanitizesort instead}%
466 }

```

`sanitize`

```

467 \define@key{glossaries.sty}{sanitize}[description=true,symbol=true,name=true]{%
468   \ifthenelse{\equal{#1}{none}}{%
469     {%
470       \GlossariesWarning{sanitize package option deprecated}%
471       \glsssetexpandfield{name}%
472       \glsssetexpandfield{symbol}%
473       \glsssetexpandfield{symbolplural}%
474       \glsssetexpandfield{desc}%
475       \glsssetexpandfield{descplural}%
476     }%
477   }%
478   \setkeys[gls]{sanitize}{#1}%

```

```

479 }%
480 }

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

notranslatorhook \@gls@notranslatorhook has been removed.

s@usetranslator
482 \newcommand*\@gls@usetranslator{%
polyglossia tricks \@ifpackageloaded into thinking that babel has been loaded, so check for
polyglossia as well.
483 \@ifpackageloaded{polyglossia}%
484 {%
485 \let\glsifusetranslator\@secondoftwo
486 }%
487 {%
488 \@ifpackageloaded{babel}%
489 {%
490 \IfFileExists{translator.sty}%
491 {%
492 \RequirePackage{translator}%
493 \let\glsifusetranslator\@firstoftwo
494 }%
495 }%
496 }%
497 {}%
498 }%
499 }

dtranslatordict Checks if given translator dictionary has been loaded.
500 \newcommand{\glsifusedtranslatordict}[3]{%
501 \glsifusetranslator
502 {\ifcsdef{ver@glossaries-dictionary-#1.dict}{#2}{#3}}%
503 {#3}%
504 }

notranslate Provide a synonym for translate=false that can be passed via the document class.
505 \@gls@declareoption{notranslate}{%
506 \glstranslatefalse
507 \let\@gls@usetranslator\relax
508 \let\glsifusetranslator\@secondoftwo
509 }

translate Define translate option. If false don't set up multi-lingual support.
510 \define@choicekey{glossaries.sty}{translate}[\val\nr]%
511 {true,false,babel}[true]%

```

```

512 {%
513   \ifcase\nr\relax
514     \glstranslatetrue
515     \renewcommand*\@gls@usetranslator{%
516       \@ifpackageloaded{polyglossia}%
517       {%
518         \let\glsifusetranslator\@secondoftwo
519       }%
520       {%
521         \@ifpackageloaded{babel}%
522         {%
523           \IfFileExists{translator.sty}%
524           {%
525             \RequirePackage{translator}%
526             \let\glsifusetranslator\@firstoftwo
527           }%
528           {}%
529         }%
530         {}%
531       }%
532     }%
533   \or
534     \glstranslatefalse
535     \let\@gls@usetranslator\relax
536     \let\glsifusetranslator\@secondoftwo
537   \or
538     \glstranslatetrue
539     \let\@gls@usetranslator\relax
540     \let\glsifusetranslator\@secondoftwo
541   \fi
542 }

```

Set the default value:

```

543 \glstranslatefalse
544 \let\glsifusetranslator\@secondoftwo
545 \@ifpackageloaded{translator}%
546 {%
547   \glstranslatetrue
548   \let\glsifusetranslator\@firstoftwo
549 }%
550 {%
551   \@for\gls@thissty:=tracklang,babel,ngerman,polyglossia\do
552   {
553     \@ifpackageloaded{\gls@thissty}%
554     {%
555       \glstranslatetrue
556       \@endfortrue
557     }%
558     {}%

```

```

559   }
560 }

indexonlyfirst  Set whether to only index on first use.
561 \define@boolkey{glossaries.sty}[gls]{indexonlyfirst}[true]{}
562 \glsindexonlyfirstfalse

hyperfirst  Set whether or not terms should have a hyperlink on first use.
563 \define@boolkey{glossaries.sty}[gls]{hyperfirst}[true]{}
564 \glshyperfirsttrue

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

footnote  Set the long form of the acronym in footnote on first use.
566 \define@boolkey{glossaries.sty}[glsacr]{footnote}[true]{}
567 \ifbool{glsacrdescription}%
568 {}%
569 {%
570   \renewcommand*{\@gls@sanitizedesc}{}%
571 }%
572 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
573 }

description  Allow acronyms to have a description (needs to be set using the description key in the optional
              argument of \newacronym).
574 \define@boolkey{glossaries.sty}[glsacr]{description}[true]{}
575 \renewcommand*{\@gls@sanitizesymbol}{}%
576 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
577 }

smallcaps  Define \newacronym to set the short form in small capitals.
578 \define@boolkey{glossaries.sty}[glsacr]{smallcaps}[true]{}
579 \renewcommand*{\@gls@sanitizesymbol}{}%
580 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
581 }

smaller  Define \newacronym to set the short form using \smaller which obviously needs to be de-
          fined by loading the appropriate package.
582 \define@boolkey{glossaries.sty}[glsacr]{smaller}[true]{}
583 \renewcommand*{\@gls@sanitizesymbol}{}%
584 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
585 }

dua  Define \newacronym to always use the long forms (i.e. don't use acronyms)
586 \define@boolkey{glossaries.sty}[glsacr]{dua}[true]{}
587 \renewcommand*{\@gls@sanitizesymbol}{}%
588 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
589 }

```

shortcuts Define acronym shortcuts.

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

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

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

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

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

order
```

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

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

```
595 \newif\ifglxsindy

The default is makeindex:
```

```
596 \glxsindyfalse

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

```
597 \@gls@declareoption{makeindex}{\glxsindyfalse}

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

```
598 \define@boolkey[gls]{xindy}{glsnumbers}[true]{}
599 \gls@xindy@glsnumberstrue

y@main@language Define what language to use for each glossary type (if a language is not defined for a particular glossary type the language specified for the main glossary is used.)
```

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

Define key to set the language
```

```
601 \define@key[gls]{xindy}{language}{\def\@xdy@main@language{#1}}

\gls@codepage Define the code page. If \inputencodingname is defined use that, otherwise have initialise with no codepage.
```

```
602 \ifcsundef{inputencodingname}{%
603   \def\gls@codepage{}}{%
604   \def\gls@codepage{\inputencodingname}
605 }

Define a key to set the code page.
```

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


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

```
607 \define@key{glossaries.sty}{xindy}[] {%
608   \glsxindytrue
609   \setkeys{gls}{xindy}{#1}%
610 }
```

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

```
611 \@gls@declareoption{xindygloss}{%
612   \glsxindytrue
613 }
```

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

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

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

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

@gls@doautomake

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

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

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

Set default:

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

compatible-3.07

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

compatible-2.07

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

Also set 3.07 compatibility if this option is set.

```
647 \ifbool{glscompatible-2.07}%
648 {%
649   \booltrue{glscompatible-3.07}%
650 }%
651 {}%
652 }
653 \boolfalse{glscompatible-2.07}
```

symbols Create a “symbols” glossary type

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

Default is not to define the symbols glossary:

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

@gls@symbolsdef

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

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

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

numbers Create a “symbols” glossary type

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

Default is not to define the numbers glossary:

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

@gls@numbersdef

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

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

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

index Create an “index” glossary type

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

Default is not to define index glossary:

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

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

```
680 \newcommand*{\@gls@indexdef}{%
681   \newglossary[ilg]{index}{ind}{idx}{\indexname}%
682   \newcommand*{\printindex}[1][\@printglossary[type=index,##1]]{%
683     \newcommand*{\newterm}[2][\@newterm]{%
684       \newglossaryentry{##2}{%
685         {type={index},name={##2},description={\nopostdesc},##1}%
686       }%
687     }%
```

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

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

Now process options passed to the package:

```
698 \ProcessOptionsX
```

Load backward compatibility stuff:

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

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

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

Now define `\setupglossaries`:

```

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

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

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

```

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

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

```

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

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

```
738 \newcommand*{\@onlypremakeg}[1]{%
739   \ifx\@gls@onlypremakeg\@empty
740     \def\@gls@onlypremakeg{#1}%
741   \else
742     \expandafter\toks@\expandafter{\@gls@onlypremakeg}%
743     \edef\@gls@onlypremakeg{\the\toks@,\noexpand#1}%
744   \fi
745 }
```

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

```
746 \newcommand*{\@disable@onlypremakeg}{%
747 \@for\@thiscs:=\@gls@onlypremakeg\do{%
748   \expandafter\@disable@premakecs\@thiscs%
749 }}
```

`\sable@premakecs` Disables the given command.

```
750 \newcommand*{\@disable@premakecs}[1]{%
751   \def#1{\PackageError{glossaries}{\string#1\space may only be
752     used before \string\makeglossaries}{You can't use
753     \string#1\space after \string\makeglossaries}}%
754 }
```

1.3 Predefined Text

Set up default textual tags that are used by this package. Some of the names may already be defined (e.g. by) so `\providecommand` is used.

Main glossary title:

`\glossaryname`

```
755 \providecommand*\glossaryname{Glossary}
```

The title for the acronym glossary type (which is defined if acronym package option is used) is given by `\acronymname`. If the acronym package option is not used, `\acronymname` won't be used.

`\acronymname`

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

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

```
757 \newcommand*{\glssettocitle}[1]{%
758   \def\glossarytocitle{\csname @gls@#1@title\endcsname}}
```

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

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

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

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

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

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

`ymbolsgroupname`
763 `\providecommand*{\glssymbolsgroupname}{Symbols}`

`umbersgroupname`
764 `\providecommand*{\glsnumbersgroupname}{Numbers}`

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

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

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

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

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

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

`eGlossariesLang`
770 `\newcommand*{\RequireGlossariesLang}[1]{%`
771 `\@ifundefined{ver@glossaries-#1.ldf}{\input{glossaries-#1.ldf}}{}`
772 `}`

`sGlossariesLang`
773 `\newcommand*{\ProvidesGlossariesLang}[1]{%`
774 `\ProvidesFile{glossaries-#1.ldf}%`
775 `}`

ssarytocaptions Does nothing if translator hasn't been loaded.

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

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

```
777 \ifglstranslate
```

Load tracklang

```
778 \RequirePackage{tracklang}
```

Load translator if required.

```
779 \@gls@usetranslator
```

If using , \glossaryname should be defined in terms of \translate, but if babel is also loaded, it will redefine \glossaryname whenever the language is set, so override it. (Don't use \addto as doesn't define it.)

```
780 \@ifpackageloaded{translator}
```

```
781 {%
```

If the language options have been specified through the document class, then translator can pick them up. If not, translator will default to English and any language option passed to babel won't be detected, so if \trans@languages is just English and \bbl@loaded isn't simply english, then don't use the translator dictionaries.

```
782 \ifboolexpr
```

```
783 {
```

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

```
785 and not
```

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

```
787 }
```

```
788 {%
```

```
789 \let\glsifusetranslator\@secondoftwo
```

```
790 }%
```

```
791 {%
```

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

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

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

```
795 {%
```

```
796 \expandafter\let\expandafter\@gls@tmp\csname captions#1\endcsname
```

```
797 \expandafter\toks@\expandafter{\@gls@tmp
```

```
798 \renewcommand*{\glossaryname}{\translate{Glossary}}}%
```

```
799 }%
```

```
800 \expandafter\edef\csname captions#1\endcsname{\the\toks@}%
```

```
801 }%
```

```
802 }%
```

```
803 }%
```

```
804 }%
```

```
805 {}%
```

Check for tracked languages

```
806 \AnyTrackedLanguages
```

```

807 {%
808   \ForEachTrackedDialect{\this@dialect}{%
809     \IfTrackedLanguageFileExists{\this@dialect}%
810     {glossaries-}% prefix
811     {.ldf}%
812     {%
813       \RequireGlossariesLang{\CurrentTrackedTag}%
814     }%
815     {%
816       \PackageWarningNoLine{glossaries}%
817       {No language module detected for ‘\this@dialect’.\MessageBreak
818       Language modules need to be installed separately.\MessageBreak
819       Please check on CTAN for a bundle called\MessageBreak
820       ‘glossaries-\CurrentTrackedLanguage’ or similar}%
821     }%
822   }%
823 }%
824 {}%

```

if using translator use translator interface.

```

825 \glsifusetranslator
826 {%
827   \renewcommand*{\glssettoctitle}[1]{%
828     \ifcsdef{gls@tr@set@#1@toctitle}%
829     {%
830       \csuse{gls@tr@set@#1@toctitle}%
831     }%
832     {%
833       \def\glossarytoctitle{\csname @glotype@#1@title\endcsname}%
834     }%
835   }%
836   \renewcommand*{\glossaryname}{\translate{Glossary}}}%
837   \renewcommand*{\acronymname}{\translate{Acronyms}}}%
838   \renewcommand*{\entryname}{\translate{Notation (glossaries)}}}%
839   \renewcommand*{\descriptionname}{%
840     \translate{Description (glossaries)}}}%
841   \renewcommand*{\symbolname}{\translate{Symbol (glossaries)}}}%
842   \renewcommand*{\pagelistname}{%
843     \translate{Page List (glossaries)}}}%
844   \renewcommand*{\glssymbolsgroupname}{%
845     \translate{Symbols (glossaries)}}}%
846   \renewcommand*{\glsnumbersgroupname}{%
847     \translate{Numbers (glossaries)}}}%
848 }{}%
849 \fi

```

`\nopostdesc` Provide a means to suppress description terminator for a given entry. (Useful for entries with no description.) Has no effect outside the glossaries.

```

850 \DeclareRobustCommand*{\nopostdesc}{}

```


`\@nopostdesc` Suppress next description terminator.

```
851 \newcommand*{\@nopostdesc}{%
852   \let\org@glspostdescription\glspostdescription
853   \def\glspostdescription{%
854     \let\glspostdescription\org@glspostdescription}%
855 }
```

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

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

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

```
857 \newcommand{\glspar}{\par}
```

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

```
858 \newcommand{\setStyleFile}[1]{%
859   \renewcommand*{\gls@istfilebase}{#1}%
  Just in case \istfilename has been modified.
860   \ifglsxindy
861     \def\istfilename{\gls@istfilebase.xdy}
862   \else
863     \def\istfilename{\gls@istfilebase.ist}
864   \fi
865 }
```

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

```
866 \@onlypremakeg\setStyleFile
```

The name of the makeindex or xindy style file is given by `\istfilename`. This file is created by `\writeist` (which is used by `\makeglossaries`) so redefining this command will only have an effect if it is done *before* `\makeglossaries`. As from v1.17, use `\setStyleFile` instead of directly redefining `\istfilename`.

`\istfilename`

```
867 \ifglsxindy
868   \def\istfilename{\gls@istfilebase.xdy}
869 \else
870   \def\istfilename{\gls@istfilebase.ist}
871 \fi
```

`gls@istfilebase`

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

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

`\@istfilename`

```
873 \newcommand*{\@istfilename}[1]{}
```

This command is the value of the `page_compositor` `makeindex` key. Again, any redefinition of this command must take place *before* `\writeist` otherwise it will have no effect. As from 1.17, use `\glsSetCompositor` instead of directly redefining `\glscompositor`.

`\glscompositor`

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

`\glsSetCompositor` Sets the compositor.

```
875 \newcommand*\glsSetCompositor}[1]{%
876   \renewcommand*\glscompositor}{#1}}
```

Only use before `\makeglossaries`

```
877 \@onlypremakeg\glsSetCompositor
```

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

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

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

`\glsAlphaCompositor` Sets the alpha compositor.

```
879 \ifglsxindy
880   \newcommand*\glsSetAlphaCompositor[1]{%
881     \renewcommand*\@glsAlphaCompositor}{#1}}
882 \else
883   \newcommand*\glsSetAlphaCompositor[1]{%
884     \glsnxindywarning\glsSetAlphaCompositor}
885 \fi
```

Can only be used before `\makeglossaries`

```
886 \@onlypremakeg\glsSetAlphaCompositor
```

`\gls@suffixF` Suffix to use for a two page list. This overrides the separator and the closing page number if set to something other than an empty macro.

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

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

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

Only has an effect when used before `\makeglossaries`

```
890 \@onlypremakeg\glsSetSuffixF
```

`\gls@suffixFF` Suffix to use for a three page list. This overrides the separator and the closing page number if set to something other than an empty macro.

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

\glsSetSuffixFF Sets the suffix to use for a three page list.
892 \newcommand*{\glsSetSuffixFF}[1]{%
893   \renewcommand*{\gls@suffixFF}{#1}%
894 }

glsnumberformat The command \glsnumberformat indicates the default format for the page numbers in the glossary. (Note that this is not the same as \glossaryentrynumbers, but applies to individual numbers or groups of numbers within an entry's associated number list.) If hyperlinks are defined, it will use \glshypernumber, otherwise it will simply display its argument "as is".
895 \ifcsundef{hyperlink}%
896 {%
897   \newcommand*{\glsnumberformat}[1]{#1}%
898 }%
899 {%
900   \newcommand*{\glsnumberformat}[1]{\glshypernumber{#1}}%
901 }

Individual numbers in an entry's associated number list are delimited using \delimN (which corresponds to the delim_n makeindex keyword). The default value is a comma followed by a space.

\delimN
902 \newcommand{\delimN}{, }

A range of numbers within an entry's associated number list is delimited using \delimR (which corresponds to the delim_r makeindex keyword). The default is an en-dash.

\delimR
903 \newcommand{\delimR}{--}

The glossary preamble is given by \glossarypreamble. This will appear after the glossary sectioning command, and before the 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.

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

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

Code provided by Michael Pock.

```
907 \newcommand{\setglossarypreamble}[2][\glsdefaulttype]{%
908   \ifglossaryexists{#1}{%
909     \csgdef{@glossarypreamble@#1}{#2}%
910   }{%
911     \GlossariesWarning{%
912       Glossary ‘#1’ is not defined%
913     }%
914   }%
915 }
```

The glossary postamble is given by `\glossarypostamble`. This is provided to allow the user to add something after the end of the `\glossary` 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

```
916 \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`.

```
917 \newcommand*{\glossarysection}[2][\@gls@title]{%
918   \def\@gls@title{#2}%
919   \ifcsundef{phantomsection}%
920   {%
921     \@glossarysection{#1}{#2}%
922   }%
923   {%
924     \p@glossarysection{#1}{#2}%
925   }%

926   \glsglossarymark{\glossarytoctitle}%
927 }
```

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

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

```

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

```

`\glossarymark` Provided for backward compatibility:

```

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

```

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

`\glossarysection`

```

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

```

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

`\glossarysection`

```

962 \newcommand*{\@glossarysection}[2]{%
963   \ifdefempty\@glossarysecstar
964   {%
965     \csname\@glossarysec\endcsname[#1]{#2}%
966   }%
967   {%

```

```

968 \csname\@glossarysec\endcsname*{#2}%
969 \@gls@toc{#1}{\@glossarysec}%
970 }%

```

Do automatic labelling if required

```

971 \@glossaryseclabel
972 }

```

As `\@glossarysection`, but put in `\phantomsection`, and swap where `\@gls@toc` goes. If using chapters do a `\clearpage`. This ensures that the hyper link from the table of contents leads to the line above the heading, rather than the line below it.

`glossarysection`

```

973 \newcommand*\@p@glossarysection[2]{%
974 \glsclearpage
975 \phantomsection
976 \ifdefempty\@glossarysecstar
977 {%
978 \csname\@glossarysec\endcsname{#2}%
979 }%
980 {%
981 \@gls@toc{#1}{\@glossarysec}%
982 \csname\@glossarysec\endcsname*{#2}%
983 }%

```

Do automatic labelling if required

```

984 \@glossaryseclabel
985 }

```

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

```

986 \newcommand*\gls@docclearpage{%
987 \ifthenelse{\equal{\@glossarysec}{chapter}}{%
988 {%
989 \ifcsundef{cleardoublepage}%
990 {%
991 \clearpage
992 }%
993 {%
994 \ifcsdef{if@openright}%
995 {%
996 \if@openright
997 \cleardoublepage
998 \else
999 \clearpage
1000 \fi
1001 }%
1002 {%
1003 \cleardoublepage

```

```

1004     }%
1005     }%
1006     }%
1007     {}%
1008 }

```

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

```

1009 \newcommand*\glsclearpage{\gls@doclearpage}

```

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

`\@gls@toc`

```

1010 \newcommand*\@gls@toc[2]{%
1011   \ifglstoc
1012     \ifglsnumberline
1013       \addcontentsline{toc}{#2}{\protect\numberline{#1}}%
1014     \else
1015       \addcontentsline{toc}{#2}{#1}%
1016     \fi
1017   \fi
1018 }

```

1.4 Xindy

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

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

```

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

```

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

```

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

```

`\dyattributelist` Comma-separated list of attributes.

```

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

```

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

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

`\@gls@ifinlist`

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

`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.

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

Check if already in list before adding.

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

Only has an effect before `\writeist`:

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

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

```
1062 \newcommand*\@disabled@glssaddxdycounters{%
1063   \PackageError{glossaries}{\string\GlsAddXdyCounters\space
1064     can't be used after \string\GlsAddXdyAttribute}{Move all
```



```

1065 occurrences of \string\GlsAddXdyCounters\space before the first
1066 instance of \string\GlsAddXdyAttribute}%
1067 }

```

AddXdyAttribute Adds an attribute.

```

1068 \ifglxsindy

```

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

```

1069 \newcommand*\@glssaddxdyattribute[2]{%

```

Add to xindy attribute list

```

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

```

Add to xindy markup location.

```

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

```

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

```

1080 \expandafter\gdef\csname glsX#2X#1\endcsname##1##2{%
1081 \setentrycounter{##1}{#2}\csname #1\endcsname{##2}%
1082 }%
1083 }

```

High-level command:

```

1084 \newcommand*\GlsAddXdyAttribute[1]{%

```

Add to comma-separated attribute list

```

1085 \ifx\@xdyattributelist\@empty
1086 \edef\@xdyattributelist{#1}%
1087 \else
1088 \edef\@xdyattributelist{\@xdyattributelist,#1}%
1089 \fi

```

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

```

1090 \@for\@this@counter:=\@xdycounters\do{%
1091 \protected@edef\gls@do@addxdyattribute{%
1092 \noexpand\@glssaddxdyattribute{#1}{\@this@counter}%
1093 }
1094 \gls@do@addxdyattribute
1095 }%

```

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

```

1096 \let\GlsAddXdyCounters\@disabled@glssaddxdycounters
1097 }

```

Only has an effect before `\writeist`:

```

1098 \onlypremakeg\GlsAddXdyAttribute
1099 \else
1100 \newcommand*\GlsAddXdyAttribute[1]{%
1101 \glsnoxindywarning\GlsAddXdyAttribute}
1102 \fi

```

`\definedattributes` Add known attributes for all defined counters

```

1103 \ifglxindy
1104 \newcommand*\@gls@addpredefinedattributes{%
1105 \GlsAddXdyAttribute{glsnumberformat}
1106 \GlsAddXdyAttribute{textrm}
1107 \GlsAddXdyAttribute{textsf}
1108 \GlsAddXdyAttribute{texttt}
1109 \GlsAddXdyAttribute{textbf}
1110 \GlsAddXdyAttribute{textmd}
1111 \GlsAddXdyAttribute{textit}
1112 \GlsAddXdyAttribute{textup}
1113 \GlsAddXdyAttribute{textsl}
1114 \GlsAddXdyAttribute{textsc}
1115 \GlsAddXdyAttribute{emph}
1116 \GlsAddXdyAttribute{glshypernumber}
1117 \GlsAddXdyAttribute{hyper rm}
1118 \GlsAddXdyAttribute{hypersf}
1119 \GlsAddXdyAttribute{hypertt}
1120 \GlsAddXdyAttribute{hyperbf}
1121 \GlsAddXdyAttribute{hypermd}
1122 \GlsAddXdyAttribute{hyperit}
1123 \GlsAddXdyAttribute{hyperup}
1124 \GlsAddXdyAttribute{hypersl}
1125 \GlsAddXdyAttribute{hypersc}
1126 \GlsAddXdyAttribute{hyperemph}

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

```

`\xdyuseralphabets` List of additional alphabets

```

1132 \def\xdyuseralphabets{}

```

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

```

1133 \ifglxindy
1134 \newcommand*\GlsAddXdyAlphabet[2]{%
1135 \edef\xdyuseralphabets{%
1136 \xdyuseralphabets ^^J
1137 (define-alphabet "#1" (#2))}}

```

```

1138 \else
1139   \newcommand*{\GlsAddXdyAlphabet}[2]{%
1140     \glsnoxywarning\GlsAddXdyAlphabet}
1141 \fi

```

This code is only required for xindy:

```

1142 \ifglsxindy

```

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

```

1143   \newcommand*{\@gls@xdy@locationlist}{%
1144     roman-page-numbers,%
1145     Roman-page-numbers,%
1146     arabic-page-numbers,%
1147     alpha-page-numbers,%
1148     Alpha-page-numbers,%
1149     Appendix-page-numbers,%
1150     arabic-section-numbers%
1151   }

```

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

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

```

1152   \protected@edef\@gls@roman{\@roman{0}\string"
1153     \string"roman-numbers-lowercase\string" :sep \string"}}%
1154   \@onelevel@sanitize\@gls@roman
1155   \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
1156     :sep \string"}%
1157   \@onelevel@sanitize\@tmp
1158   \ifx\@tmp\@gls@roman
1159     \expandafter
1160     \edef\csname @gls@xdy@Lclass@roman-page-numbers\endcsname{%
1161       \string"roman-numbers-lowercase\string"%
1162     }%
1163   \else
1164     \expandafter
1165     \edef\csname @gls@xdy@Lclass@roman-page-numbers\endcsname{
1166       :sep \string"\@gls@roman\string"%
1167     }%
1168   \fi

```

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

```

1169   \expandafter\def\csname @gls@xdy@Lclass@Roman-page-numbers\endcsname{%
1170     \string"roman-numbers-uppercase\string"%
1171   }%

```

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

```

1172 \expandafter\def\csname @gls@xdy@Lclass@arabic-page-numbers\endcsname{%
1173     \string"arabic-numbers\string"%
1174 }%
```

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

```

1175 \expandafter\def\csname @gls@xdy@Lclass@alpha-page-numbers\endcsname{%
1176     \string"alpha\string"%
1177 }%
```

`\Alpha-page-numbers` Upper case alphabetical (A, B, ...).

```

1178 \expandafter\def\csname @gls@xdy@Lclass@Alpha-page-numbers\endcsname{%
1179     \string"ALPHA\string"%
1180 }%
```

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

```

1181 \expandafter\def\csname @gls@xdy@Lclass@Appendix-page-numbers\endcsname{%
1182     \string"ALPHA\string"
1183     :sep \string"\@glsAlphacompositor\string"
1184     \string"arabic-numbers\string"%
1185 }
```

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

```

1186 \expandafter\def\csname @gls@xdy@Lclass@arabic-section-numbers\endcsname{%
1187     \string"arabic-numbers\string"
1188     :sep \string"\glscompositor\string"
1189     \string"arabic-numbers\string"%
1190 }%
```

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

```

1191 \def\@xdyuserlocationdefs{}
```

`\userlocationnames` List of additional user location names

```

1192 \def\@xdyuserlocationnames{}
```

End of xindy-only block:

```

1193 \fi
```

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

```

1194 \ifglsxindy
1195 \newcommand*\GlsAddXdyLocation[3][{}]{%
1196     \def\@gls@tmp{#1}%
1197     \ifx\@gls@tmp\@empty
1198         \edef\@xdyuserlocationdefs{%

```

```

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

```

Only has an effect before \writeist:

```

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

```

ationclassorder Define location class order

```

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

```

Change the location order.

ationClassOrder

```

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

```

```

1241     \glsnoxywarning\GlsSetXdyLocationClassOrder}
1242 \fi

\@xdysortrules   Define sort rules
1243 \ifglxindy
1244   \def\@xdysortrules{}
1245 \fi

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

xyrequiredstyles  Define list of required styles (this should be a comma-separated list of xindy styles)
1256 \ifglxindy
1257   \def\@xdyrequiredstyles{tex}
1258 \fi

\GlsAddXdyStyle  Add a xindy style to the list of required styles
1259 \ifglxindy
1260   \newcommand*\GlsAddXdyStyle[1]{%
1261     \edef\@xdyrequiredstyles{\@xdyrequiredstyles,#1}}%
1262 \else
1263   \newcommand*\GlsAddXdyStyle[1]{%
1264     \glsnoxywarning\GlsAddXdyStyle}
1265 \fi

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

findrootlanguage  This used to determine the root language, using a bit of trickery since babel doesn't supply the
                  information, but now that babel is once again actively maintained, we can't do this any more,
                  so \findrootlanguage is no longer available. Now provide a command that does nothing
                  (in case it's been patched), but this may be removed completely in the future.
1273 \newcommand*\findrootlanguage{}

```

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

```

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

```

`\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.

```

1275 \ifglxindy
1276   \newcommand*\GlsSetXdyLanguage[2][\glsdefaulttype]{%
1277     \ifglossaryexists{#1}{%
1278       \expandafter\def\csname @xdy@#1@language\endcsname{#2}%
1279     }{%
1280       \PackageError{glossaries}{Can't set language type for
1281         glossary type '#1' --- no such glossary}{%
1282         You have specified a glossary type that doesn't exist}}
1283 \else
1284   \newcommand*\GlsSetXdyLanguage[2][]{%
1285     \glsnoxywarning\GlsSetXdyLanguage}
1286 \fi

```

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

```

1287 \def\@gls@codepage#1#2{}

```

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

```

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

```

Suggested by egreg:

```

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

```

`\xdylettergroups` Store letter group definitions.

```

1301 \ifglxindy
1302   \ifglxindy@glsnumbers
1303     \def\@xdylettergroups{(define-letter-group
1304       \string"glnumbers\string"^^J\space\space\space
1305       :prefixes (\string"0\string" \string"1\string"

```

```

1306      \string"2\string" \string"3\string" \string"4\string"
1307      \string"5\string" \string"6\string" \string"7\string"
1308      \string"8\string" \string"9\string")^^J\space\space\space
1309      :before \string"\@glsfirstletter\string"))}
1310 \else
1311   \def\@xdylettergroups{}
1312 \fi
1313 \fi

```

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.

```

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

```

1.5 Loops and conditionals

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

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

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

```

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

```

\forallacronyms

```

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

```

\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.

```

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

```


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

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

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

```
1332 \newcommand*{\forall glossary entries}[3][\@glo@types]{%
1333   \expandafter\forall glossaries\expandafter[#1]{\@@this@glo@}%
1334   {%
1335     \forall glossary entries[\@@this@glo@]{#2}{#3}%
1336   }%
1337 }
```

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

```
\if glossary exists {⟨type⟩} {⟨true-text⟩} {⟨false-text⟩}
```

where `⟨type⟩` is the glossary's label.

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

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

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

(Note, don't use `\detokenize` or it will cause commands like `\glsaddall` to fail.) Since re-defining `\glsdetoklabel` can cause things to go badly wrong, I'm not going to mention it in the main user guide. Only advanced users who know what they're doing ought to attempt it.

`\glsdetoklabel`

```
1341 \newcommand*{\glsdetoklabel}[1]{#1}
```

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

```
\if glossentry exists {⟨label⟩} {⟨true text⟩} {⟨false text⟩}
```

where `⟨label⟩` is the entry's label.

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

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

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

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

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

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

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

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

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

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

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

```
1354 \newcommand{\glsdoifnoexists}[2]{%
1355   \ifglsentryexists{#1}{%
1356     \PackageError{glossaries}{Glossary entry '\glsdetoklabel{#1}' has already
1357       been defined}{}}{#2}%
1358 }
```

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

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

```
1359 \newcommand{\glsdoifexistsorwarn}[2]{%
1360   \ifglsentryexists{#1}{#2}{%
1361     \GlossariesWarning{Glossary entry '\glsdetoklabel{#1}'
1362       has not been defined}%
1363   }%
1364 }
```

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

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

```

1365 \newcommand{\glsdoifexistsordo}[3]{%
1366   \ifglentryexists{#1}{#2}{%
1367     \PackageError{glossaries}{Glossary entry ‘\glsdetoklabel{#1}’
1368       has not been defined}{You need to define a glossary entry before you
1369       can use it.}%
1370     #3%
1371   }%
1372 }

```

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

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

```

1373 \newcommand{\doifglossarynoexistsordo}[3]{%
1374   \ifglossaryexists{#1}%
1375   {%
1376     \PackageError{glossaries}{Glossary type ‘#1’ already exists}{}%
1377     #3%
1378   }%
1379   {#2}%
1380 }

```

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

```

1381 \newcommand{\ifglshaschildren}[3]{%
1382   \glsdoifexists{#1}%
1383   {%
1384     \def\do@glshaschildren{#3}%
1385     \edef\@gls@thislabel{\glsdetoklabel{#1}}%
1386     \expandafter\forglentries\expandafter
1387       [\csname glo@\@gls@thislabel @type\endcsname]
1388     {\glo@label}%
1389     {%
1390       \letcs\glo@parent{glo@\glo@label @parent}%
1391       \ifdefequal\@gls@thislabel\glo@parent
1392       {%
1393         \def\do@glshaschildren{#2}%
1394         \@endfortrue
1395       }%
1396     }%
1397   }%
1398   \do@glshaschildren
1399 }%
1400 }

```

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

```

1401 \newcommand{\ifglshasparent}[3]{%
1402   \glsdoifexists{#1}%
1403   {%
1404     \ifcseempty{glo@\glsdetoklabel{#1}@parent}{#3}{#2}%
1405   }%
1406 }

```

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

```

1407 \newcommand*{\ifglshasdesc}[3]{%
1408   \ifcseempty{glo@\glsdetoklabel{#1}@desc}%
1409   {#3}%
1410   {#2}%
1411 }

```

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

```

1412 \newcommand*{\ifglstdescsuppressed}[3]{%
1413   \ifcsequal{glo@\glsdetoklabel{#1}@desc}{@no@post@desc}%
1414   {#2}%
1415   {#3}%
1416 }

```

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

```

1417 \newcommand*{\ifglshassymbol}[3]{%
1418   \letcs{\@glo@symbol}{glo@\glsdetoklabel{#1}@symbol}%
1419   \ifdefempty\@glo@symbol
1420   {#3}%
1421   {%
1422     \ifdefequal\@glo@symbol\@gls@default@value
1423     {#3}%
1424     {#2}%
1425   }%
1426 }

```

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

```

1427 \newcommand*{\ifglshaslong}[3]{%
1428   \letcs{\@glo@long}{glo@\glsdetoklabel{#1}@long}%
1429   \ifdefempty\@glo@long
1430   {#3}%
1431   {%
1432     \ifdefequal\@glo@long\@gls@default@value
1433     {#3}%
1434     {#2}%
1435   }%
1436 }

```

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

```

1437 \newcommand*{\ifglshasshort}[3]{%

```

```

1438 \letcs{\@glo@short}{glo@\glsdetoklabel{#1}@short}%
1439 \ifdefempty\@glo@short
1440 {#3}%
1441 {%
1442   \ifdefequal\@glo@short\@gls@default@value
1443   {#3}%
1444   {#2}%
1445 }%
1446 }

```

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

```

1447 \newcommand*{\ifglshasfield}[4]{%
1448   \glsdoifexists{#2}%
1449   {%
1450     \letcs{\@glo@thisvalue}{glo@\glsdetoklabel{#2}@#1}%

```

First check supplied field label is defined.

```

1451   \ifdef\@glo@thisvalue
1452   {%

```

Is defined, so now check if empty.

```

1453     \ifdefempty\@glo@thisvalue
1454     {%

```

Is empty, so doesn't have field set.

```

1455         #4%
1456     }%
1457   {%

```

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

```

1458     \ifdefequal\@glo@thisvalue\@gls@default@value
1459     {%

```

Value is set to the default value.

```

1460         #4%
1461     }%
1462   {%

```

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

```

1463       \let\glscurrentfieldvalue\@glo@thisvalue
1464       #3%
1465     }%
1466   }%
1467 }%
1468 {%

```

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

```

1469   \@gls@fetchfield{\@gls@thisfield}{#1}%

```

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

```
1470 \ifdef\@gls@thisfield
1471 {%
```

Is defined, so now check if empty.

```
1472 \letcs{\@glo@thisvalue}{glo@glsetoklabel{#2}\@gls@thisfield}%
1473 \ifdefempty\@glo@thisvalue
1474 {%
```

Is empty so field hasn't been set.

```
1475 #4%
1476 }%
1477 {%
```

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

```
1478 \ifdequal\@glo@thisvalue\@gls@default@value
1479 {%
```

Value is set to the default value.

```
1480 #4%
1481 }%
1482 {%
```

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

```
1483 \let\glscurrentfieldvalue\@glo@thisvalue
1484 #3%
1485 }%
1486 }%
1487 }%
1488 {%
```

Not defined.

```
1489 \GlossariesWarning{Unknown entry field '#1'}%
1490 #4%
1491 }%
1492 }%
1493 }%
1494 }
```

`\glscurrentfieldvalue`

```
1495 \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`

```
1496 \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.

```
1497 \newcommand*\@gls@provide@newglossary{%
1498   \protected@write\@auxout{}\string\providecommand\string\@newglossary[4]{}%

```

Only need to do this once.

```
1499   \let\@gls@provide@newglossary\relax
1500 }
```

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

```
1501 \newcommand*\defglsentryfmt}[2][\glsdefaulttype]{%
1502   \csgdef{gls@#1@entryfmt}{#2}%
1503 }
```

`\gls@doentryfmt`

```
1504 \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.

```
1505 \newcommand*\@gls@forbidtexext}[1]{%
1506   \ifboolexpr{test {\ifdefstring{#1}{tex}}
1507               or test {\ifdefstring{#1}{TEX}}}
1508   {%
1509     \def#1{nottex}%
1510     \PackageError{glossaries}%
1511       {Forbidden '.tex' extension replaced with '.nottex'}%
1512       {I'm sorry, I can't allow you to do something so reckless.\MessageBreak
1513         Don't use '.tex' as an extension for a temporary file.}%
1514   }%
1515   {%
1516   }%
1517 }
```

`\gls@gobbleopt` Discard optional argument.

```
1518 \newcommand*\gls@gobbleopt{}\new@ifnextchar[{\@gls@gobbleopt}{}
1519 \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`

```
1520 \newcommand*{\newglossary}{\@ifstar\s@newglossary\ns@newglossary}
```

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

```
1521 \newcommand*{\s@newglossary}[2]{%
1522 \ns@newglossary[#1-glg]{#1}{#1-gls}{#1-glo}{#2}%
1523 }
```

`\ns@newglossary` Define the unstarred version.

```
1524 \newcommand*{\ns@newglossary}[5][glg]{%
1525 \doifglossarynoexistsordo{#2}%
1526 {%
```

Check if default has been set

```
1527 \ifundef\glsdefaulttype
1528 {%
1529 \gdef\glsdefaulttype{#2}%
1530 }{}}%
```

Add this to the list of glossary types:

```
1531 \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.

```
1532 \expandafter\gdef\csname glolist@#2\endcsname{,}%
```

Store the file extensions:

```
1533 \expandafter\edef\csname @glotype@#2@log\endcsname{#1}%
1534 \expandafter\edef\csname @glotype@#2@in\endcsname{#3}%
1535 \expandafter\edef\csname @glotype@#2@out\endcsname{#4}%
1536 \expandafter\@gls@forbidtextext\csname @glotype@#2@log\endcsname
1537 \expandafter\@gls@forbidtextext\csname @glotype@#2@in\endcsname
1538 \expandafter\@gls@forbidtextext\csname @glotype@#2@out\endcsname
```

Store the title:

```
1539 \expandafter\def\csname @glotype@#2@title\endcsname{#5}%
```

```
1540 \@gls@provide@newglossary
```

```
1541 \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.

```
1542 \ifcsundef{gls@#2@entryfmt}%
1543 {%
1544 \defglsentryfmt[#2]{\glsentryfmt}%
1545 }%
1546 {}%
```


Define sort counter if required:

```
1547 \@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.)

```
1548 \@ifnextchar[{\@gls@setcounter{#2}}{%
1549   {\@gls@setcounter{#2}[\glscounter]}}%
1550 }%
1551 {%
1552   \gls@gobbleopt
1553 }%
1554 }
```

`\altnewglossary`

```
1555 \newcommand*{\altnewglossary}[3]{%
1556   \newglossary[#2-glg]{#1}{#2-gls}{#2-glo}{#3}%
1557 }
```

Only define new glossaries in the preamble:

```
1558 \@onlypreamble{\newglossary}
```

Only define new glossaries before `\makeglossaries`

```
1559 \@onlypremakeg\newglossary
```

`\@newglossary` is used to specify the file extensions for the `makeindex` input, output and transcript files. It is written to the auxiliary file by `\newglossary`. Since it is not used by \LaTeX , `\@newglossary` simply ignores its arguments.

`\@newglossary`

```
1560 \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`

```
1561 \def\@gls@setcounter#1[#2]{%
1562   \expandafter\def\csname @glotype@#1@counter\endcsname{#2}%
1563   \ifglsxindy
1564     \GlsAddXdyCounters{#2}%
1565   \fi
1566 }
```

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

`@gls@getcounter`

```
1567 \newcommand*{\@gls@getcounter}[1]{%
1568   \csname @glotype@#1@counter\endcsname
1569 }
```

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

```
1570 \glsdefmain
```

Define the “acronym” glossaries if required.

```
1571 \@gls@do@acronymsdef
```

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

```
1572 \@gls@do@symbolsdef
```

```
1573 \@gls@do@numbersdef
```

```
1574 \@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.

```
1575 \newcommand*{\newignoredglossary}[1]{%
1576   \ifdefempty\@ignored@glossaries
1577   {%
1578     \edef\@ignored@glossaries{#1}%
1579   }%
1580   {%
1581     \eappto\@ignored@glossaries{, #1}%
1582   }%
1583   \csgdef{glolist@#1}{,}%
1584   \ifcsundef{gls@#1@entryfmt}%
1585   {%
1586     \defglsentryfmt[#1]{\glsentryfmt}%
1587   }%
1588   {}%
1589   \ifdefempty\@gls@nohyperlist
1590   {%
1591     \renewcommand*{\@gls@nohyperlist}{#1}%
1592   }%
1593   {%
1594     \eappto\@gls@nohyperlist{, #1}%
1595   }%
1596 }
```

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

```
1597 \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.

```
1598 \newcommand*{\ifignoredglossary}[3]{%
1599   \edef\@gls@igtype{#1}%
1600   \expandafter\DTLifinlist\expandafter
1601   {\@gls@igtype}{\@ignored@glossaries}{#2}{#3}%
1602 }
```

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.

```
1603 \define@key{glossentry}{name}{%
1604 \def\@glo@name{#1}%
1605 }
```

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`.

```
1606 \define@key{glossentry}{description}{%
1607 \def\@glo@desc{#1}%
1608 }
```

descriptionplural

```
1609 \define@key{glossentry}{descriptionplural}{%
1610 \def\@glo@descplural{#1}%
1611 }
```

sort The sort key needs to be sanitized here (the sort key is provided for `makeindex`'s benefit, not for use in the document). The sort key is optional when defining a new glossary entry. If omitted, the value is given by `\langle name \rangle \langle description \rangle`.

```
1612 \define@key{glossentry}{sort}{%
1613 \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.

```
1614 \define@key{glossentry}{text}{%
1615 \def\@glo@text{#1}%
1616 }
```

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.

```
1617 \define@key{glossentry}{plural}{%
1618 \def\@glo@plural{#1}%
1619 }
```

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.

```
1620 \define@key{glossentry}{first}{%
1621 \def\@glo@first{#1}%
1622 }
```

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.

```
1623 \define@key{glossentry}{firstplural}{%
1624 \def\@glo@firstplural{#1}%
1625 }
```

s@default@value

```
1626 \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).

```
1627 \define@key{glossentry}{symbol}{%
1628 \def\@glo@symbol{#1}%
1629 }
```

symbolplural

```
1630 \define@key{glossentry}{symbolplural}{%
1631 \def\@glo@symbolplural{#1}%
1632 }
```

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

```
1633 \define@key{glossentry}{type}{%
1634 \def\@glo@type{#1}}
```

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

```
1635 \define@key{glossentry}{counter}{%
1636 \ifcsundef{c@#1}%
1637 {%
1638 \PackageError{glossaries}%
1639 {There is no counter called ‘#1’}%
1640 {%
1641 The counter key should have the name of a valid counter
1642 as its value%
1643 }%
1644 }%
```

```

1645  {%
1646    \def\@glo@counter{#1}%
1647  }%
1648 }

```

see The see key specifies a list of cross-references

```

1649 \define@key{glossentry}{see}{%
1650   \gls@checkseeallowed
1651   \def\@glo@see{#1}%
1652   \@glo@seeautonumberlist
1653 }

```

checkseeallowed

```

1654 \newcommand*{\gls@checkseeallowed}{%
1655   \PackageError{glossaries}%
1656   {'see' key may only be used after \string\makeglossaries\space
1657   or \string\makenoidxglossaries}%
1658   {You must use \string\makeglossaries\space
1659   or \string\makenoidxglossaries\space before defining
1660   any entries that have a 'see' key}%
1661 }

```

ed@preambleonly

```

1662 \newcommand*{\gls@checkseeallowed@preambleonly}{%
1663   \GlossariesWarning{glossaries}%
1664   {'see' key doesn't have any effect when used in the document
1665   environment. Move the definition to the preamble
1666   after \string\makeglossaries\space
1667   or \string\makenoidxglossaries}%
1668 }

```

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

```

1669 \define@key{glossentry}{parent}{%
1670 \def\@glo@parent{#1}}

```

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

```

1671 \define@choicekey{glossentry}{nonumberlist}[\val\nr]{true,false}[true]{%
1672   \ifcase\nr\relax
1673     \def\@glo@prefix{\glsnonextpages}%
1674   \else
1675     \def\@glo@prefix{\glsnextpages}%
1676   \fi
1677 }

```

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

user1

```

1678 \define@key{glossentry}{user1}{%
1679   \def\@glo@useri{#1}%
1680 }

```

user2

```
1681 \define@key{glossentry}{user2}{%  
1682   \def\@glo@userii{#1}%  
1683 }
```

user3

```
1684 \define@key{glossentry}{user3}{%  
1685   \def\@glo@useriii{#1}%  
1686 }
```

user4

```
1687 \define@key{glossentry}{user4}{%  
1688   \def\@glo@useriv{#1}%  
1689 }
```

user5

```
1690 \define@key{glossentry}{user5}{%  
1691   \def\@glo@userv{#1}%  
1692 }
```

user6

```
1693 \define@key{glossentry}{user6}{%  
1694   \def\@glo@uservi{#1}%  
1695 }
```

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.

```
1696 \define@key{glossentry}{short}{%  
1697   \def\@glo@short{#1}%  
1698 }
```

shortplural This key is provided for use by \newacronym.

```
1699 \define@key{glossentry}{shortplural}{%  
1700   \def\@glo@shortpl{#1}%  
1701 }
```

long This key is provided for use by \newacronym.

```
1702 \define@key{glossentry}{long}{%  
1703   \def\@glo@long{#1}%  
1704 }
```

longplural This key is provided for use by \newacronym.

```
1705 \define@key{glossentry}{longplural}{%  
1706   \def\@glo@longpl{#1}%  
1707 }
```

```

\@glsnname Define command to generate error if name key is missing.
1708 \newcommand*{\@glsnname}{%
1709   \PackageError{glossaries}{name key required in
1710   \string\newglossaryentry\space for entry '\@glo@label'}{You
1711   haven't specified the entry name}}

\@glsnodelsc Define command to generate error if description key is missing.
1712 \newcommand*\@glsnodelsc{%
1713   \PackageError{glossaries}
1714   {%
1715     description key required in \string\newglossaryentry\space
1716     for entry '\@glo@label'%
1717   }%
1718   {%
1719     You haven't specified the entry description%
1720   }%
1721 }%

lsdefaultplural Now obsolete. Don't use.
1722 \newcommand*{\@glsdefaultplural}{}

missingnumberlist Define a command to generate warning when numberlist not set.
1723 \newcommand*{\@gls@missingnumberlist}[1]{%
1724   ??%
1725   \ifglssavenumberlist
1726     \GlossariesWarning{Missing number list for entry '#1'.
1727     Maybe makeglossaries + rerun required.}%
1728   \else
1729     \PackageError{glossaries}%
1730     {Package option 'savenumberlist=true' required.}%
1731     {%
1732       You must use the 'savenumberlist' package option
1733       to reference location lists.%
1734     }%
1735   \fi
1736 }

@glsdefaultsort Define command to set default sort.
1737 \newcommand*{\@glsdefaultsort}{\@glo@name}

\gls@level Register to increment entry levels.
1738 \newcount\gls@level

@noexpand@field
1739 \newcommand{\@@gls@noexpand@field}[3]{%
1740   \expandafter\global\expandafter
1741   \let\csname glo@#1@#2\endcsname#3%
1742 }

```

noexpand@fields

```
1743 \newcommand{\@gls@noexpand@fields}[4]{%
1744   \ifcsdef{gls@assign@#3@field}
1745   {%
1746     \ifdefequal{#4}{\@gls@default@value}%
1747     {%
1748       \edef\@gls@value{\expandonce{#1}}%
1749       \csuse{gls@assign@#3@field}{#2}{\@gls@value}%
1750     }%
1751     {%
1752       \csuse{gls@assign@#3@field}{#2}{#4}%
1753     }%
1754   }%
1755   {%
1756     \ifdefequal{#4}{\@gls@default@value}%
1757     {%
1758       \edef\@gls@value{\expandonce{#1}}%
1759       \@gls@noexpand@field{#2}{#3}{\@gls@value}%
1760     }%
1761     {%
1762       \@gls@noexpand@field{#2}{#3}{#4}%
1763     }%
1764   }%
1765 }
```

ls@expand@field

```
1766 \newcommand{\@gls@expand@field}[3]{%
1767   \expandafter
1768   \protected@xdef\csname glo@#1@#2\endcsname{#3}%
1769 }
```

s@expand@fields

```
1770 \newcommand{\@gls@expand@fields}[4]{%
1771   \ifcsdef{gls@assign@#3@field}
1772   {%
1773     \ifdefequal{#4}{\@gls@default@value}%
1774     {%
1775       \edef\@gls@value{\expandonce{#1}}%
1776       \csuse{gls@assign@#3@field}{#2}{\@gls@value}%
1777     }%
1778     {%
1779       \expandafter\@gls@startswithexpandonce#4\relax\relax\@gls@endcheck
1780       {%
1781         \@gls@expand@field{#2}{#3}{#4}%
1782       }%
1783     }%
1784     \csuse{gls@assign@#3@field}{#2}{#4}%
1785   }%
```



```

1786     }%
1787 }%
1788 {%
1789     \ifdefequal{#4}{\@gls@default@value}%
1790     {%
1791         \@gls@expand@field{#2}{#3}{#1}%
1792     }%
1793     {%
1794         \@gls@expand@field{#2}{#3}{#4}%
1795     }%
1796 }%
1797 }

```

switexpandonce

```

1798 \def\@gls@expandonce{\expandonce}
1799 \def\@gls@startswithexpandonce#1#2\gls@endcheck#3#4{%
1800     \def\@gls@tmp{#1}%
1801     \ifdefequal{\@gls@expandonce}{\@gls@tmp}{#3}{#4}%
1802 }

```

gls@assign@field

```
\gls@assign@field{\<def value>}{\<label>}{\<field>}{\<tmp cs>}
```

Assigns an entry field. Expansion performed by default (except for name, symbol and description where backward compatibility required). If *<tmp cs>* is *<@gls@default@value>*, *<def value>* is used instead.

```
1803 \let\gls@assign@field\@gls@expand@fields
```

glsexpandfields

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

```

1804 \newcommand*{\glsexpandfields}{%
1805     \let\gls@assign@field\@gls@expand@fields
1806 }

```

snoexpandfields

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

```

1807 \newcommand*{\glsnoexpandfields}{%
1808     \let\gls@assign@field\@gls@noexpand@fields
1809 }

```

ewglossaryentry

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

```
1810 \newrobustcmd{\newglossaryentry}[2]{%
```

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

```

1811     \glsdoifnoexists{#1}%
1812     {%
1813         \gls@defglossaryentry{#1}{#2}%

```

```

1814 }%
1815 }

```

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

```

1816 \newcommand*{\gls@defdocnewglossaryentry}{%
1817   \let\gls@checkseeallowed\gls@checkseeallowed@preambleonly
1818   \let\newglossaryentry\new@glossaryentry
1819 }

```

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

```

1820 \newrobustcmd{\provideglossaryentry}[2]{%
1821   \ifglstryexists{#1}%
1822   }{%
1823   }%
1824   \gls@defglossaryentry{#1}{#2}%
1825 }%
1826 }
1827 \@onlypreamble{\provideglossaryentry}

```

`\w@glossaryentry` For use in document environment.

```

1828 \newrobustcmd{\new@glossaryentry}[2]{%
1829   \ifundef\@gls@deffile
1830   {%
1831     \global\newwrite\@gls@deffile
1832     \immediate\openout\@gls@deffile=\jobname.glsdefs
1833   }%
1834   }{%
1835     \ifglstryexists{#1}{}%
1836     {%
1837       \gls@defglossaryentry{#1}{#2}%
1838     }%
1839     \@gls@writedef{#1}%
1840   }
1841 \AtBeginDocument
1842 {
1843   \makeatletter
1844   \InputIfFileExists{\jobname.glsdefs}{\}{}%
1845   \makeatother
1846   \gls@defdocnewglossaryentry
1847 }
1848 \AtEndDocument{\ifdef\@gls@deffile{\closeout\@gls@deffile}{}}

```

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

```

1849 \newcommand*{\@gls@writedef}[1]{%
1850   \immediate\write\@gls@deffile
1851   {%
1852     \string\ifglstryexists{#1}{}\glspercentchar^^J%
1853     \expandafter\@gobble\string\{\glspercentchar^^J%

```

```

1854 \string\gls@defglossaryentry{\glsdetoklabel{#1}}\glspercentchar^^J%
1855 \expandafter\@gobble\string\{\glspercentchar%
1856 }%

Write key value information:

1857 \@for\@gls@map:=\@gls@keymap\do
1858 {%
1859 \edef\glo@value{\expandafter\expandonce
1860 \csname glo@\glsdetoklabel{#1}@\expandafter
1861 \@secondoftwo\@gls@map\endcsname}%
1862 \@onelevel@sanitize\glo@value
1863 \immediate\write\@gls@deffile
1864 {%
1865 \expandafter\@firstoftwo\@gls@map
1866 =\expandafter\@gobble\string\{\glo@value\expandafter\@gobble\string\},%
1867 \glspercentchar%
1868 }%
1869 }%

Provide hook:

1870 \gls.writedefhook
1871 \immediate\write\@gls@deffile
1872 {%
1873 \glspercentchar^^J%
1874 \expandafter\@gobble\string\}\glspercentchar^^J%
1875 \expandafter\@gobble\string\}\glspercentchar%
1876 }%
1877 }

```

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

```

1878 \newcommand*{\@gls@keymap}{%
1879 {name}{name},%
1880 {sort}{sortvalue},% unescaped sort value
1881 {type}{type},%
1882 {first}{first},%
1883 {firstplural}{firstpl},%
1884 {text}{text},%
1885 {plural}{plural},%
1886 {description}{desc},%
1887 {descriptionplural}{descplural},%
1888 {symbol}{symbol},%
1889 {symbolplural}{symbolplural},%
1890 {user1}{useri},%
1891 {user2}{userii},%
1892 {user3}{useriii},%
1893 {user4}{useriv},%
1894 {user5}{userv},%
1895 {user6}{uservi},%
1896 {long}{long},%

```

```

1897 {longplural}{longpl},%
1898 {short}{short},%
1899 {shortplural}{shortpl},%
1900 {counter}{counter},%
1901 {parent}{parent}%
1902 }

```

\@gls@fetchfield

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

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

```
1903 \newcommand*{\@gls@fetchfield}[2]{%
```

Ensure user field name is fully expanded

```
1904 \edef\@gls@thisval{#2}%
```

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

```
1905 \@for\@gls@map:=\@gls@keymap\do{%
```

```
1906 \edef\@this@key{\expandafter\@firstoftwo\@gls@map}%
```

```
1907 \ifdefequal{\@this@key}{\@gls@thisval}%
```

```
1908 {%
```

Found it.

```
1909 \edef#1{\expandafter\@secondoftwo\@gls@map}%
```

Break out of loop.

```
1910 \@endfortrue
```

```
1911 }%
```

```
1912 {}%
```

```
1913 }%
```

```
1914 }
```

glsaddstoragekey

```
\glsaddstoragekey{<key>}{<default value>}{<no link cs>}
```

Similar to \glsaddkey but intended for keys whose values aren't explicitly used in the document, but might be required behind the scenes by other commands.

```
1915 \newcommand*{\glsaddstoragekey}{\@ifstar\@sglsaddstoragekey\@glsaddstoragekey}
```

Starred version switches on expansion for this key.

```
1916 \newcommand*{\@sglsaddstoragekey}[1]{%
```

```
1917 \key@ifundefined{glossentry}{#1}%
```

```
1918 {%
```

```
1919 \expandafter\newcommand\expandafter*\expandafter
```

```
1920 {\csname gls@assign@#1@field\endcsname}[2]{%
```

```
1921 \@@gls@expand@field{##1}{#1}{##2}%
```

```
1922 }%
```

```
1923 }%
```

```
1924 {}%
```

```
1925 \@glsaddstoragekey{#1}%
```

```
1926 }
```

Unstarred version doesn't override default expansion.

```
1927 \newcommand*{\@glsaddstoragekey}[3]{%
```

Check the specified key doesn't already exist.

```
1928 \key@ifundefined{glossentry}{#1}%
1929 {%
```

Set up the key.

```
1930 \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
1931 \appto\@gls@keymap{, {#1}{#1}}%
```

Set the default value.

```
1932 \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
```

Assignment code.

```
1933 \appto\@newglossaryentryposthook{%
1934 \letcs{\@glo@tmp}{@glo@#1}%
1935 \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
1936 }%
```

Define the no-link commands.

```
1937 \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
1938 }%
1939 {%
1940 \PackageError{glossaries}{Key ‘#1’ already exists}{}%
1941 }%
1942 }
```

<code>\glsaddkey</code>	<code>\glsaddkey{<key>}{<default value>}{<no link cs>}{<no link ucfirst cs>}{<link cs>}{<link ucfirst cs>}{<link allcaps cs>}</code>
-------------------------	--

Allow user to add their own custom keys.

```
1943 \newcommand*{\glsaddkey}{\@ifstar\@sglsaddkey\@glsaddkey}
```

Starred version switches on expansion for this key.

```
1944 \newcommand*{\@sglsaddkey}[1]{%
1945 \key@ifundefined{glossentry}{#1}%
1946 {%
1947 \expandafter\newcommand\expandafter*\expandafter
1948 {\csname gls@assign@#1@field\endcsname}[2]{%
1949 \@gls@expand@field{##1}{#1}{##2}%
1950 }%
1951 }%
1952 }%
1953 \@glsaddkey{#1}%
1954 }
```

Unstarred version doesn't override default expansion.

```
1955 \newcommand*{\@glsaddkey}[7]{%
```

Check the specified key doesn't already exist.

```
1956 \key@ifundefined{glossentry}{#1}%
1957 {%
```

Set up the key.

```
1958 \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
1959 \appto\@gls@keymap{,{#1}{#1}}%
```

Set the default value.

```
1960 \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
```

Assignment code.

```
1961 \appto\@newglossaryentryposthook{%
1962 \letcs{\@glo@tmp}{@glo@#1}%
1963 \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
1964 }%
```

Define the no-link commands.

```
1965 \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
1966 \newcommand*{#4}[1]{\@Gls@entry@field{##1}{#1}}%
```

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

```
1967 \ifcsdef{@gls@user@#1@}%
1968 {%
1969 \PackageError{glossaries}%
1970 {Can't define '\string#5' as helper command
1971 '\expandafter\string\csname @gls@user@#1@\endcsname' already exists}%
1972 }%
1973 }%
1974 {%
1975 \expandafter\newcommand\expandafter*\expandafter
1976 {\csname @gls@user@#1@\endcsname}[2][ ]{%
1977 \new@ifnextchar[%
1978 {\csuse{@gls@user@#1@}{##1}{##2}}%
1979 {\csuse{@gls@user@#1@}{##1}{##2}[ ]}%
1980 \csdef{@gls@user@#1@}##1##2[##3]{%
1981 \@gls@field@link{##1}{##2}{#3{##2}##3}%
1982 }%
1983 \newrobustcmd*{#5}{%
1984 \expandafter\@gls@hyp@opt\csname @gls@user@#1@\endcsname}%
1985 }%
```

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

```
1986 \ifcsdef{@Gls@user@#1@}%
1987 {%
1988 \PackageError{glossaries}%
1989 {Can't define '\string#6' as helper command
1990 '\expandafter\string\csname @Gls@user@#1@\endcsname' already exists}%
1991 }%
1992 }%
1993 {%
```

```

1994 \expandafter\newcommand\expandafter*\expandafter
1995 {\csname @Gls@user@#1\endcsname}[2][\%
1996 \new@ifnextchar[\%
1997 {\csuse{@Gls@user@#1@}{##1}{##2}}}%
1998 {\csuse{@Gls@user@#1@}{##1}{##2}[]}}}%
1999 \csdef{@Gls@user@#1@}##1##2[##3]{\%
2000 \@gls@field@link{##1}{##2}{#4{##2}##3}%
2001 }%
2002 \newrobustcmd*{#6}{\%
2003 \expandafter\@gls@hyp@opt\csname @Gls@user@#1\endcsname}%
2004 }%

```

Finally the all caps version:

```

2005 \ifcsdef{@GLS@user@#1@}%
2006 {\%
2007 \PackageError{glossaries}%
2008 {Can't define '\string#7' as helper command
2009 '\expandafter\string\csname @GLS@user@#1\endcsname' already exists}%
2010 }%
2011 }%
2012 {\%
2013 \expandafter\newcommand\expandafter*\expandafter
2014 {\csname @GLS@user@#1\endcsname}[2][\%
2015 \new@ifnextchar[\%
2016 {\csuse{@GLS@user@#1@}{##1}{##2}}}%
2017 {\csuse{@GLS@user@#1@}{##1}{##2}[]}}}%
2018 \csdef{@GLS@user@#1@}##1##2[##3]{\%
2019 \@gls@field@link{##1}{##2}{\mfirstucMakeUppercase{#3{##2}##3}}}%
2020 }%
2021 \newrobustcmd*{#7}{\%
2022 \expandafter\@gls@hyp@opt\csname @GLS@user@#1\endcsname}%
2023 }%
2024 }%
2025 {\%
2026 \PackageError{glossaries}{Key '#1' already exists}{}%
2027 }%
2028 }

```

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

```

2029 \newcommand{\glsfieldxdef}[3]{\%
2030 \glsdoifexists{#1}%
2031 {\%
2032 \edef\@glo@label{\glsdetoklabel{#1}}}%
2033 \ifcsdef{glo@\@glo@label @#2}%
2034 {\%
2035 \expandafter\xdef\csname glo@\@glo@label @#2\endcsname{#3}%

```

```

2036 }%
2037 {%
2038     \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2039 }%
2040 }%
2041 }

```

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

```

2042 \newcommand{\glsfielddedef}[3]{%
2043     \glsdoifexists{#1}%
2044     {%
2045         \edef\@glo@label{\glsdetoklabel{#1}}%
2046         \ifcsdef{glo@\@glo@label @#2}%
2047         {%
2048             \expandafter\edef\csname glo@\@glo@label @#2\endcsname{#3}%
2049         }%
2050         {%
2051             \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2052         }%
2053     }%
2054 }

```

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

```

2055 \newcommand{\glsfielddgdef}[3]{%
2056     \glsdoifexists{#1}%
2057     {%
2058         \edef\@glo@label{\glsdetoklabel{#1}}%
2059         \ifcsdef{glo@\@glo@label @#2}%
2060         {%
2061             \expandafter\gdef\csname glo@\@glo@label @#2\endcsname{#3}%
2062         }%
2063         {%
2064             \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2065         }%
2066     }%
2067 }

```

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

```

2068 \newcommand{\glsfieldddef}[3]{%
2069     \glsdoifexists{#1}%

```



```

2070 {%
2071   \edef\@glo@label{\glsdetoklabel{#1}}%
2072   \ifcsdef{glo@\@glo@label @#2}%
2073   {%
2074     \expandafter\def\csname glo@\@glo@label @#2\endcsname{#3}%
2075   }%
2076   {%
2077     \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2078   }%
2079 }%
2080 }

```

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

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

```

2081 \newcommand{\glsfieldfetch}[3]{%
2082   \glsdoifexists{#1}%
2083   {%
2084     \edef\@glo@label{\glsdetoklabel{#1}}%
2085     \ifcsdef{glo@\@glo@label @#2}%
2086     {%
2087       \letcs#3{glo@\@glo@label @#2}%
2088     }%
2089     {%
2090       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2091     }%
2092   }%
2093 }

```

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

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

```

2094 \newcommand{\ifglsfieldeq}[5]{%
2095   \glsdoifexists{#1}%
2096   {%
2097     \edef\@glo@label{\glsdetoklabel{#1}}%
2098     \ifcsdef{glo@\@glo@label @#2}%
2099     {%
2100       \ifcsstring{glo@\@glo@label @#2}{#3}{#4}{#5}%
2101     }%
2102     {%
2103       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2104     }%
2105   }%
2106 }

```

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

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

```
2107 \newcommand{\ifglsfielddefeq}[5]{%
2108   \glsdoifexists{#1}%
2109   {%
2110     \edef\@glo@label{\glsdetoklabel{#1}}%
2111     \ifcsdef{glo@\@glo@label @#2}%
2112     {%
2113       \expandafter\ifdefstrequal
2114       \csname glo@\@glo@label @#2\endcsname{#3}{#4}{#5}%
2115     }%
2116     {%
2117       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2118     }%
2119   }%
2120 }
```

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

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

```
2121 \newcommand{\ifglsfieldcseq}[5]{%
2122   \glsdoifexists{#1}%
2123   {%
2124     \edef\@glo@label{\glsdetoklabel{#1}}%
2125     \ifcsdef{glo@\@glo@label @#2}%
2126     {%
2127       \ifcsstrequal{glo@\@glo@label @#2}{#3}{#4}{#5}%
2128     }%
2129     {%
2130       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2131     }%
2132   }%
2133 }
```

`glswritedefhook`

```
2134 \newcommand*{\glswritedefhook}{}%
```

`gls@assign@desc`

```
2135 \newcommand*{\gls@assign@desc}[1]{%
2136   \gls@assign@field{#1}{desc}{\@glo@desc}%
2137   \gls@assign@field{\@glo@desc}{#1}{descplural}{\@glo@descplural}%
2138 }
```

`ewglossaryentry`

```
2139 \newcommand{\longnewglossaryentry}[3]{%
```

```

2140 \glsdoifnoexists{#1}%
2141 {%
2142   \bgroup
2143   \let\@org@newglossaryentryprehook\@newglossaryentryprehook
2144   \long\def\@newglossaryentryprehook{%
2145     \long\def\@glo@desc{#3\leavevmode\unskip\nopostdesc}%
2146     \@org@newglossaryentryprehook
2147   }%
2148   \renewcommand*\@gls@assign@desc}[1]{%
2149     \global\cslet{glo@\glsdetoklabel{#1}@desc}{\@glo@desc}%
2150     \global\cslet{glo@\glsdetoklabel{#1}@descplural}{\@glo@desc}%
2151   }
2152   \gls@defglossaryentry{#1}{#2}%
2153 \egroup
2154 }
2155 }

```

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

```
2156 \@onlypreamble{\longnewglossaryentry}
```

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

```

2157 \newcommand{\longprovideglossaryentry}[3]{%
2158   \ifglsentryexists{#1}{}%
2159   {\longnewglossaryentry{#1}{#2}{#3}}%
2160 }
2161 \@onlypreamble{\longprovideglossaryentry}

```

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

Defines a new entry without checking if it already exists.

```
2162 \newcommand{\gls@defglossaryentry}[2]{%
```

Store label

```
2163   \edef\@glo@label{\glsdetoklabel{#1}}%
```

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

```
2164   \let\glslabel\@glo@label
```

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

```
2165   \let\@glo@name\@glsnname
```

```
2166   \let\@glo@desc\@glsndesc
```

```
2167   \let\@glo@descplural\@gls@default@value
```

```
2168   \let\@glo@type\@gls@default@value
```

```
2169   \let\@glo@symbol\@gls@default@value
```

```
2170   \let\@glo@symbolplural\@gls@default@value
```

```

2171 \let\@glo@text\@gls@default@value
2172 \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.)

```

2173 \let\@glo@first\@gls@default@value
2174 \let\@glo@firstplural\@gls@default@value

```

Set the default sort:

```

2175 \let\@glo@sort\@gls@default@value

```

Set the default counter:

```

2176 \let\@glo@counter\@gls@default@value
2177 \def\@glo@see{}%
2178 \def\@glo@parent{}%
2179 \def\@glo@prefix{}%
2180 \def\@glo@useri{}%
2181 \def\@glo@userii{}%
2182 \def\@glo@useriii{}%
2183 \def\@glo@useriv{}%
2184 \def\@glo@userv{}%
2185 \def\@glo@uservi{}%
2186 \def\@glo@short{}%
2187 \def\@glo@shortpl{}%
2188 \def\@glo@long{}%
2189 \def\@glo@longpl{}%

```

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

```

2190 \@newglossaryentryprehook

```

Extract key-val information from third parameter:

```

2191 \setkeys{glossentry}{#2}%

```

Check there is a default glossary.

```

2192 \ifundef\glsdefaultttype
2193 {%
2194   \PackageError{glossaries}%
2195   {No default glossary type (have you used ‘nomain’ by mistake?)}%
2196   {If you use package option ‘nomain’ you must define
2197    a new glossary before you can define entries}%
2198 }%
2199 {}%

```

Assign type. This must be fully expandable

```

2200 \gls@assign@field{\glsdefaultttype}{\@glo@label}{type}{\@glo@type}%
2201 \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.

```

2202 \ifcsundef{glolist@%glo@type}%
2203 {%
2204     \PackageError{glossaries}%
2205     {Glossary type ‘\@glo@type’ has not been defined}%
2206     {You need to define a new glossary type, before making entries
2207      in it}%
2208 }%
2209 {%

```

Check if it's an ignored glossary

```

2210 \ifignoredglossary%glo@type
2211 {%

```

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

```

2212 \ifx%glo@desc%glsnodesc
2213 \let%glo@desc%empty
2214 \fi
2215 }%
2216 {%
2217 }%
2218 \protected@edef%glolist@{\csname glolist@%glo@type\endcsname}%
2219 \expandafter\xdef\csname glolist@%glo@type\endcsname{%
2220 \glolist@{\@glo@label},}%
2221 }%

```

Initialise level to 0.

```

2222 \gls@level=0\relax

```

Has this entry been assigned a parent?

```

2223 \ifx%glo@parent%empty

```

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

```

2224 \expandafter\gdef\csname glo@%glo@label @parent\endcsname{}%
2225 \else

```

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

```

2226 \ifdefequal%glo@label%glo@parent%
2227 {%
2228     \PackageError{glossaries}{Entry ‘\@glo@label’ can't be its own parent}{}%
2229     \def%glo@parent{}%
2230     \expandafter\gdef\csname glo@%glo@label @parent\endcsname{}%
2231 }%
2232 {%

```

Check the parent exists:

```

2233 \ifglstryexists{\@glo@parent}%
2234 {%

```

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

```

2235 \expandafter\xdef\csname glo@%glo@label @parent\endcsname{%
2236 \glo@parent}%

```

Determine level.

```
2237      \gls@level=\csname glo@\glo@parent @level\endcsname\relax
2238      \advance\gls@level by 1\relax
```

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

```
2239      \ifx\@glo@name\@gls@name
2240      \expandafter\let\expandafter\@glo@name
2241      \csname glo@\glo@parent @name\endcsname
```

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

```
2242      \ifx\@glo@plural\@gls@default@value
2243      \expandafter\let\expandafter\@glo@plural
2244      \csname glo@\glo@parent @plural\endcsname
2245      \fi
2246      \fi
2247      }%
2248      {%
```

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

```
2249      \PackageError{glossaries}%
2250      {%
2251      Invalid parent '@glo@parent'
2252      for entry '@glo@label' - parent doesn't exist%
2253      }%
2254      {%
2255      Parent entries must be defined before their children%
2256      }%
2257      \def\@glo@parent{%
2258      \expandafter\gdef\csname glo@\glo@label @parent\endcsname{%
2259      }%
2260      }%
2261      \fi
```

Set the level for this entry

```
2262      \expandafter\xdef\csname glo@\glo@label @level\endcsname{\number\gls@level}%
```

Define commands associated with this entry:

```
2263      \gls@assign@field{\@glo@name}{\@glo@label}{sortvalue}{\@glo@sort}%
2264      \letcs\@glo@sort{glo@\glo@label @sortvalue}%
2265      \gls@assign@field{\@glo@name}{\@glo@label}{text}{\@glo@text}%
2266      \expandafter\gls@assign@field\expandafter
2267      {\csname glo@\glo@label @text\endcsname\glspluralsuffix}%
2268      {\@glo@label}{plural}{\@glo@plural}%
2269      \expandafter\gls@assign@field\expandafter
2270      {\csname glo@\glo@label @text\endcsname}%
2271      {\@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.

```
2272      \ifx\@glo@first\@gls@default@value
2273      \expandafter\gls@assign@field\expandafter
```

```

2274         {\csname glo@\@glo@label @plural\endcsname}%
2275         {\@glo@label}{firstpl}{\@glo@firstplural}%
2276     \else
2277         \expandafter\gls@assign@field\expandafter
2278         {\csname glo@\@glo@label @first\endcsname\glspluralsuffix}%
2279         {\@glo@label}{firstpl}{\@glo@firstplural}%
2280     \fi

2281     \ifcsundef{@glo@type@\@glo@type @counter}%
2282     {%
2283         \def\@glo@defaultcounter{\glscounter}%
2284     }%
2285     {%
2286         \letcs\@glo@defaultcounter{@glo@type@\@glo@type @counter}%
2287     }%
2288     \gls@assign@field{\@glo@defaultcounter}{\@glo@label}{counter}{\@glo@counter}%
2289     \gls@assign@field{}{\@glo@label}{useri}{\@glo@useri}%
2290     \gls@assign@field{}{\@glo@label}{userii}{\@glo@userii}%
2291     \gls@assign@field{}{\@glo@label}{useriii}{\@glo@useriii}%
2292     \gls@assign@field{}{\@glo@label}{useriv}{\@glo@useriv}%
2293     \gls@assign@field{}{\@glo@label}{userv}{\@glo@userv}%
2294     \gls@assign@field{}{\@glo@label}{uservi}{\@glo@uservi}%
2295     \gls@assign@field{}{\@glo@label}{short}{\@glo@short}%
2296     \gls@assign@field{}{\@glo@label}{shortpl}{\@glo@shortpl}%
2297     \gls@assign@field{}{\@glo@label}{long}{\@glo@long}%
2298     \gls@assign@field{}{\@glo@label}{longpl}{\@glo@longpl}%
2299     \ifx\@glo@name\@glsnoname
2300         \@glsnoname
2301         \let\@glo@name\@gls@default@value
2302     \fi
2303     \gls@assign@field{}{\@glo@label}{name}{\@glo@name}%

```

Set default numberlist if not defined:

```

2304     \ifcsundef{glo@\@glo@label @numberlist}%
2305     {%
2306         \csxdef{glo@\@glo@label @numberlist}{%
2307             \noexpand\@gls@missingnumberlist{\@glo@label}}%
2308     }%
2309     {}%

```

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.

```

2310     \def\@glo@@desc{\@glo@first}%
2311     \ifx\@glo@desc\@glo@@desc
2312         \let\@glo@desc\@glo@first
2313     \fi
2314     \ifx\@glo@desc\@glsnodeldesc
2315         \@glsnodeldesc
2316         \let\@glo@desc\@gls@default@value
2317     \fi

```

2318 \gls@assign@desc{\@glo@label}%

Set the sort key for this entry:

2319 \@gls@defsort{\@glo@type}{\@glo@label}%

2320 \def\@glo@symbol{\@glo@text}%

2321 \ifx\@glo@symbol\@glo@symbol

2322 \let\@glo@symbol\@glo@text

2323 \fi

2324 \gls@assign@field{\relax}{\@glo@label}{symbol}{\@glo@symbol}%

2325 \expandafter

2326 \gls@assign@field\expandafter

2327 {\csname glo@\@glo@label @symbol\endcsname}

2328 {\@glo@label}{symbolplural}{\@glo@symbolplural}%

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

2329 \expandafter\xdef\csname glo@\@glo@label @flagfalse\endcsname{%

2330 \noexpand\global

2331 \noexpand\let\expandafter\noexpand

2332 \csname ifglo@\@glo@label @flag\endcsname\noexpand\iffalse

2333 }%

2334 \expandafter\xdef\csname glo@\@glo@label @flagtrue\endcsname{%

2335 \noexpand\global

2336 \noexpand\let\expandafter\noexpand

2337 \csname ifglo@\@glo@label @flag\endcsname\noexpand\iftrue

2338 }%

2339 \csname glo@\@glo@label @flagfalse\endcsname

Sort out any cross-referencing if required.

2340 \ifdefined\@glo@see

2341 {}%

2342 {%

2343 \protected@edef\@do@glsee{%

2344 \noexpand\@gls@fixbraces\noexpand\@glo@list\@glo@see

2345 \noexpand\@nil

2346 \noexpand\expandafter\noexpand\@glsee\noexpand\@glo@list{\@glo@label}}%

2347 \@do@glsee

2348 }%

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

2349 \ifignoredglossary\@glo@type

2350 {%

2351 \csdef{glo@\@glo@label @index}{}%

2352 }

2353 {%

2354 \do@glo@storeentry{\@glo@label}%

2355 }%

Define entry counters if enabled:

2356 \@newglossaryentry@defcounters

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

```
2357 \newglossaryentryposthook
2358 }
```

aryentryprehook Allow extra information to be added to glossary entries:

```
2359 \newcommand*{\@newglossaryentryprehook}{}%
```

ryentryposthook Allow extra information to be added to glossary entries:

```
2360 \newcommand*{\@newglossaryentryposthook}{}%
```

try@defcounters

```
2361 \newcommand*{\@newglossaryentry@defcounters}{}%
```

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

```
2362 \newcommand*{\glsmoveentry}[2]{%
2363   \edef\@glo@thislabel{\glstoklabel{#1}}%
2364   \edef\glo@type{\csname glo@\@glo@thislabel @type\endcsname}%
2365   \def\glo@list{,%}%
2366   \for\glsentries[\glo@type]{\glo@label}%
2367   {%
2368     \ifdefequal\@glo@thislabel\glo@label
2369     {\eappto\glo@list{\glo@label,}}%
2370     }%
2371   \cslet\glolist@\glo@type{\glo@list}%
2372   \csdef{glo@\@glo@thislabel @type}{#2}%
2373 }
```

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

```
2374 \ifglxindy
2375   \newcommand*{\@glossaryentryfield}{\string\glossentry}
2376 \else
2377   \newcommand*{\@glossaryentryfield}{\string\glossentry}
2378 \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.)

```
2379 \ifglxindy
2380   \newcommand*{\@glossarysubentryfield}{%
2381     \string\subglossentry}
2382 \else
2383   \newcommand*{\@glossarysubentryfield}{%
2384     \string\subglossentry}
2385 \fi
```

\@glo@storeentry

\@glo@storeentry{\<label>}

Determine the format to write the entry in the glossary output (.glo) file. The argument is the entry's label (should already have been de-tok'ed if required). The result is stored in \@glo@<label>@index, where <label> is the entry's label. (This doesn't include any formatting or location information.)

```
2386 \newcommand{\@glo@storeentry}[1]{%
```

Escape makeindex/xindy special characters in the label:

```
2387 \edef\@glo@esclabel{#1}%
```

```
2388 \gls@checkmkidxchars\@glo@esclabel
```

Get the sort string and escape any special characters

```
2389 \protected@edef\@glo@sort{\csname glo@#1@sort\endcsname}%
```

```
2390 \gls@checkmkidxchars\@glo@sort
```

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

```
2391 \gls@checkmkidxchars\@glo@prefix
```

Get the parent, if one exists

```
2392 \edef\@glo@parent{\csname glo@#1@parent\endcsname}%
```

Write the information to the glossary file.

```
2393 \ifglxindy
```

Store using xindy syntax.

```
2394 \ifx\@glo@parent\@empty
```

Entry doesn't have a parent

```
2395 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
```

```
2396 (\string"\@glo@sort\string" %
```

```
2397 \string"\@glo@prefix\@glossaryentryfield{\@glo@esclabel}\string") %
```

```
2398 }%
```

```
2399 \else
```

Entry has a parent

```
2400 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
```

```
2401 \csname glo@\@glo@parent @index\endcsname
```

```
2402 (\string"\@glo@sort\string" %
```

```
2403 \string"\@glo@prefix\@glossarysubentryfield
```

```
2404 {\csname glo@#1@level\endcsname}{\@glo@esclabel}\string") %
```

```
2405 }%
```

```
2406 \fi
```

```
2407 \else
```

Store using makeindex syntax.

```
2408 \ifx\@glo@parent\@empty
```

Sanitize \@glo@prefix

```
2409 \@onelevel@sanitize\@glo@prefix
```

Entry doesn't have a parent

```

2410 \expandafter\protected\xdef\csname glo@#1@index\endcsname{%
2411 \glo@sort\@gls@actualchar\@glo@prefix
2412 \@glossaryentryfield{\@glo@esclabel}%
2413 }%
2414 \else

```

Entry has a parent

```

2415 \expandafter\protected\xdef\csname glo@#1@index\endcsname{%
2416 \csname glo@\@glo@parent @index\endcsname\@gls@levelchar
2417 \glo@sort\@gls@actualchar\@glo@prefix
2418 \@glossarysubentryfield
2419 {\csname glo@#1@level\endcsname}\@glo@esclabel}%
2420 }%
2421 \fi
2422 \fi
2423 }

```

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`

```

2424 \AtBeginDocument{%
2425 \ifpackageloaded{amsmath}%
2426 {\let\gls@ifnotmeasuring\@gls@ifnotmeasuring}%
2427 }%
2428 }
2429 \newcommand*\@gls@ifnotmeasuring[1]{%
2430 \ifmeasuring@
2431 \else
2432 #1%
2433 \fi
2434 }
2435 \newcommand*\gls@ifnotmeasuring[1]{#1}

```

`\glsreset` The command `\glsreset{<label>}` can be used to set the entry flag to indicate that it hasn't been used yet. The required argument is the entry label.

```

2436 \newcommand*\glsreset[1]{%
2437 \gls@ifnotmeasuring
2438 {%
2439 \glsdoifexists{#1}%
2440 {%
2441 \@glsreset{#1}%
2442 }%

```

```

2443 }%
2444 }

```

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

```

2445 \newcommand*{\glslocalreset}[1]{%
2446   \gls@ifnotmeasuring
2447   {%
2448     \glsdoifexists{#1}%
2449     {%
2450       \@glslocalreset{#1}%
2451     }%
2452   }%
2453 }

```

`\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.

```

2454 \newcommand*{\glsunset}[1]{%
2455   \gls@ifnotmeasuring
2456   {%
2457     \glsdoifexists{#1}%
2458     {%
2459       \@glsunset{#1}%
2460     }%
2461   }%
2462 }

```

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

```

2463 \newcommand*{\glslocalunset}[1]{%
2464   \gls@ifnotmeasuring
2465   {%
2466     \glsdoifexists{#1}%
2467     {%
2468       \@glslocalunset{#1}%
2469     }%
2470   }%
2471 }

```

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

```

2472 \newcommand*{\@glslocalunset}{\@glslocalunset}

```

`@@glslocalunset` Local unset without checks.

```

2473 \newcommand*{\@@glslocalunset}[1]{%
2474   \expandafter\let\csname ifglo@glsdetoklabel{#1}@flag\endcsname\iftrue
2475 }

```

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

```

2476 \newcommand*{\@glsunset}{\@glsunset}

```

`\@@glsunset` Global unset without checks.

```
2477 \newcommand*{\@@glsunset}[1]{%
2478   \expandafter\global\csname glo@glsdetoklabel{#1}@flagtrue\endcsname
2479 }
```

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

```
2480 \newcommand*{\@glslocalreset}{\@@glslocalreset}
```

`@@glslocalreset` Local reset without checks.

```
2481 \newcommand*{\@@glslocalreset}[1]{%
2482   \expandafter\let\csname ifglo@glsdetoklabel{#1}@flag\endcsname\iffalse
2483 }
```

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

```
2484 \newcommand*{\@glsreset}{\@@glsreset}
```

`\@@glsreset` Global reset without checks.

```
2485 \newcommand*{\@@glsreset}[1]{%
2486   \expandafter\global\csname glo@glsdetoklabel{#1}@flagfalse\endcsname
2487 }
```

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

`\glsresetall`

```
2488 \newcommand*{\glsresetall}[1][\@glo@types]{%
2489   \forallglsentries[#1]{\@glsentry}%
2490   {%
2491     \glsreset{\@glsentry}%
2492   }%
2493 }
```

As above, but with only a local effect:

`\glslocalresetall`

```
2494 \newcommand*{\glslocalresetall}[1][\@glo@types]{%
2495   \forallglsentries[#1]{\@glsentry}%
2496   {%
2497     \glslocalreset{\@glsentry}%
2498   }%
2499 }
```

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

`\glsunsetall`

```
2500 \newcommand*{\glsunsetall}[1][\@glo@types]{%
2501   \forallglsentries[#1]{\@glsentry}%

```

```

2502  {%
2503    \glsunset{\@glsentry}%
2504  }%
2505 }

```

As above, but with only a local effect:

`\glslocalunsetall`

```

2506 \newcommand*{\glslocalunsetall}[1][\@glo@types]{%
2507   \forallglsentries[#1]{\@glsentry}%
2508   {%
2509     \glslocalunset{\@glsentry}%
2510   }%
2511 }

```

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

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

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

```

2512 \newcommand*{\@newglossaryentry@defcounters}{%
2513   \csdef{glo@\@glo@label @currcount}{0}%
2514   \csdef{glo@\@glo@label @prevcount}{0}%
2515 }

```

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

```

2516 \newcommand*{\glsenableentrycount}{%
  Enable new entry fields.
2517   \let\@newglossaryentry@defcounters\@newglossaryentry@defcounters
  Disable \newglossaryentry in the document environment.
2518   \renewcommand*{\gls@defdocnewglossaryentry}{%
2519     \renewcommand*{\newglossaryentry}[2]{%
2520       \PackageError{glossaries}{\string\newglossaryentry\space
2521         may only be used in the preamble when entry counting has
2522         been activated}{If you use \string\glsenableentrycount\space
2523         you must place all entry definitions in the preamble not in
2524         the document environment}%
2525     }%
2526   }%

```

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

```

2527 \newcommand*\glsentrycurrcount}[1]{%
2528   \ifcsundef{glo@glsdetoklabel{##1}@currcount}%
2529   {0}{\@gls@entry@field{##1}{currcount}}%
2530 }%
2531 \newcommand*\glsentryprevcount}[1]{%
2532   \ifcsundef{glo@glsdetoklabel{##1}@prevcount}%
2533   {0}{\@gls@entry@field{##1}{prevcount}}%
2534 }%

```

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

```

2535 \renewcommand*\@glsunset}[1]{%
2536   \@glsunset{##1}%
2537   \@gls@increment@currcount{##1}%
2538 }%
2539 \renewcommand*\@glslocalunset}[1]{%
2540   \@glslocalunset{##1}%
2541   \@gls@local@increment@currcount{##1}%
2542 }%
2543 \renewcommand*\@glsreset}[1]{%
2544   \@glsreset{##1}%
2545   \csgdef{glo@glsdetoklabel{##1}@currcount}{0}%
2546 }%
2547 \renewcommand*\@glslocalreset}[1]{%
2548   \@glslocalreset{##1}%
2549   \csdef{glo@glsdetoklabel{##1}@currcount}{0}%
2550 }%

```

Alter behaviour of `\cgl`s. (Only global unset is used if previous count was one as it doesn't make sense to have a local unset here given that the previous count was global.)

```

2551 \def\@cgl@##1##2[##3]{%
2552   \ifnum\glsentryprevcount{##2}=1\relax
2553     \cglformat{##2}{##3}%
2554     \glsunset{##2}%
2555   \else
2556     \@gls@{##1}{##2}[##3]%
2557   \fi
2558 }%

```

Similarly for the analogous commands. No case change plural:

```

2559 \def\@cglsp@##1##2[##3]{%
2560   \ifnum\glsentryprevcount{##2}=1\relax
2561     \cglspformat{##2}{##3}%
2562     \glsunset{##2}%
2563   \else
2564     \@cglsp@{##1}{##2}[##3]%
2565   \fi
2566 }%

```

First letter uppercase singular:

```

2567 \def\@cGls@##1##2[##3]{%
2568   \ifnum\glsentryprevcount{##2}=1\relax
2569     \cGlsformat{##2}{##3}%
2570     \glsunset{##2}%
2571   \else
2572     \@Gls@{##1}{##2}[##3]%
2573   \fi
2574 }%

```

First letter uppercase plural:

```

2575 \def\@cGlspl@##1##2[##3]{%
2576   \ifnum\glsentryprevcount{##2}=1\relax
2577     \cGlsplformat{##2}{##3}%
2578     \glsunset{##2}%
2579   \else
2580     \@Glspl@{##1}{##2}[##3]%
2581   \fi
2582 }%

```

Write information to aux file at the end of the document

```

2583 \AtEndDocument{\@gls@write@entrycounts}%

```

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

```

2584 \renewcommand*{\@gls@entry@count}[2]{%
2585   \csxdef{glo@\glsdetoklabel{##1}@prevcount}{##2}%
2586 }%

```

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

```

2587 \let\glsenableentrycount\relax
2588 }
2589 \@onlypreamble\glsenableentrycount

```

ement@currcount

```

2590 \newcommand*{\@gls@increment@currcount}[1]{%
2591   \csxdef{glo@\glsdetoklabel{##1}@currcount}{%
2592     \number\numexpr\glsentrycurrcount{##1}+1}%
2593 }

```

ement@currcount

```

2594 \newcommand*{\@gls@local@increment@currcount}[1]{%
2595   \csedef{glo@\glsdetoklabel{##1}@currcount}{%
2596     \number\numexpr\glsentrycurrcount{##1}+1}%
2597 }

```

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.)

```

2598 \newcommand*{\@gls@write@entrycounts}{%

```



```

2599 \immediate\write\@auxout
2600   {\string\providecommand*\@string\@gls@entry@count}[2]{}%
2601 \forallglsentries{\@glsentry}{%
2602   \ifglsused{\@glsentry}%
2603   {\immediate\write\@auxout
2604     {\string\@gls@entry@count{\@glsentry}{\glsentrycurrcount{\@glsentry}}}%
2605   }%
2606 }%
2607 }

```

`\gls@entry@count` Default behaviour is to ignore arguments. Activated by `\glsenableentrycount`.

```
2608 \newcommand*\@gls@entry@count}[2]{}
```

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

```
2609 \newrobustcmd*\cgl{\@gls@hyp@opt\@cgl}
```

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

```

2610 \newcommand*\@cgl}[2][{}]{%
2611   \new@ifnextchar[{\@cgl@{#1}{#2}}{\@cgl@{#1}{#2}[]}%
2612 }

```

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

```

2613 \def\@cgl@#1#2[#3]{%
2614   \GlossariesWarning{\string\cgl\space is defaulting to
2615     \string\gls\space since you haven't enabled entry counting}%
2616   \@gls@{#1}{#2}[#3]%
2617 }

```

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

```

2618 \newcommand*\cglformat}[2]{%
2619   \ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}#2%
2620 }

```

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

```
2621 \newrobustcmd*\cGl{\@gls@hyp@opt\@cGl}
```

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

```

2622 \newcommand*\@cGl}[2][{}]{%
2623   \new@ifnextchar[{\@cGl@{#1}{#2}}{\@cGl@{#1}{#2}[]}%
2624 }

```

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

```

2625 \def\cGls@#1#2[#3]{%
2626   \GlossariesWarning{\string\cGls\space is defaulting to
2627     \string\Gls\space since you haven't enabled entry counting}%
2628   \cGls@{#1}{#2}[#3]%
2629 }

```

`\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.

```

2630 \newcommand*\cGlsformat}[2]{%
2631   \ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}#2%
2632 }

```

`\cglsp1` Define command that works like `\glsp1` but behaves differently if the entry count function is enabled. (If not enabled, it behaves the same as `\glsp1` but issues a warning.)

```

2633 \newrobustcmd*\cglsp1{\cGls@hyp@opt\cglsp1}

```

`\cglsp1` Defined the un-starred form. Need to determine if there is a final optional argument

```

2634 \newcommand*\cglsp1}[2][ ]{%
2635   \new@ifnextchar[\cglsp1@{#1}{#2}}{\cglsp1@{#1}{#2}[ ]}%
2636 }

```

`\cglsp1@` Read in the final optional argument. This defaults to same behaviour as `\glsp1` but issues a warning.

```

2637 \def\cglsp1@#1#2[#3]{%
2638   \GlossariesWarning{\string\cglsp1\space is defaulting to
2639     \string\glsp1\space since you haven't enabled entry counting}%
2640   \cglsp1@{#1}{#2}[#3]%
2641 }

```

`\cglsp1format` Format used by `\cglsp1` if entry only used once on previous run. The first argument is the label, the second argument is the insert text.

```

2642 \newcommand*\cglsp1format}[2]{%
2643   \ifglshaslong{#1}{\glsp1entrylongpl{#1}}{\glsp1entryfirstplural{#1}}#2%
2644 }

```

`\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.)

```

2645 \newrobustcmd*\cGlspl{\cGls@hyp@opt\cGlspl}

```

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

```

2646 \newcommand*\cGlspl}[2][ ]{%
2647   \new@ifnextchar[\cGlspl@{#1}{#2}}{\cGlspl@{#1}{#2}[ ]}%
2648 }

```

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

```

2649 \def\cGlspl@#1#2[#3]{%

```

```

2650 \GlossariesWarning{\string\cGlspl\space is defaulting to
2651 \string\Glspl\space since you haven't enabled entry counting}%
2652 \@Glspl@{#1}{#2}[#3]%
2653 }

```

`\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.

```

2654 \newcommand*{\cGlsplformat}[2]{%
2655 \ifglshaslong{#1}{\Glsentrylongpl{#1}}{\Glsentryfirstplural{#1}}#2%
2656 }

```

1.10 Loading files containing glossary entries

Glossary entries can be defined in an external file. These external files can contain `\newglossaryentry` and `\newacronym` commands.¹

`\loadglsentries[⟨type⟩]{⟨filename⟩}`

This command will input the file using `\input`. The optional argument specifies to which glossary the entries should be assigned if they haven't used the type key. If the optional argument is not specified, the default glossary is used. Only those entries used in the document (via `\glslink`, `\gls`, `\glspl` and uppercase variants or `\glsadd` and `\glsaddall` will appear in the glossary). The mandatory argument is the filename (with or without `.tex` extension).

`\loadglsentries`

```

2657 \newcommand*{\loadglsentries}[2][\@gls@default]{%
2658 \let\@gls@default\glsdefaulttype
2659 \def\glsdefaulttype{#1}\input{#2}%
2660 \let\glsdefaulttype\@gls@default
2661 }

```

`\loadglsentries` can only be used in the preamble:

```

2662 \@onlypreamble{\loadglsentries}

```

1.11 Using glossary entries in the text

Any term that has been defined using `\newglossaryentry` (or `\newacronym`) can be displayed in the text (i.e. outside of the glossary) using one of the commands defined in this section. Unless you use `\glslink`, the way the term appears in the text is determined by `\glsdisplayfirst` (if it is the first time the term has been used) or `\glsdisplay` (for subsequent use). Any formatting commands (such as `\textbf` is governed by `\glstextformat`. By default this just displays the link text “as is”.

`\glstextformat`

```

2663 \newcommand*{\glstextformat}[1]{#1}

```

¹and any other valid \TeX code that can be used in the preamble.

`\glentryfmt` As from version 3.11a, the way in which an entry is displayed is now governed by `\glentryfmt`. This doesn't take any arguments. The required information is set by commands like `\gls`. To ensure backward compatibility, the default use the old `\glsdisplay` and `\glsdisplayfirst` style of commands

```
2664 \newcommand*{\glentryfmt}{%
2665   \@gls@default@entryfmt\glsdisplayfirst\glsdisplay
2666 }
```

Format that provides backwards compatibility:

```
2667 \newcommand*{\@gls@default@entryfmt}[2]{%
2668   \ifdefempty\glscustomtext
2669     {%
2670       \glsifplural
2671       {%
```

Plural form

```
2672       \glscapscase
2673       {%
```

Don't adjust case

```
2674       \ifglsused\glslabel
2675       {%
```

Subsequent use

```
2676         #2{\glentryplural{\glslabel}}%
2677         {\glentrydescplural{\glslabel}}%
2678         {\glentrysymbolplural{\glslabel}}{\glsinsert}%
2679       }%
2680     {%
```

First use

```
2681         #1{\glentryfirstplural{\glslabel}}%
2682         {\glentrydescplural{\glslabel}}%
2683         {\glentrysymbolplural{\glslabel}}{\glsinsert}%
2684       }%
2685     }%
2686     {%
```

Make first letter upper case

```
2687       \ifglsused\glslabel
2688       {%
```

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

```
2689       \ifbool{glscompatible-3.07}%
2690       {%
2691         \protected@edef\@glo@etext{%
2692           #2{\glentryplural{\glslabel}}%
2693           {\glentrydescplural{\glslabel}}%
2694           {\glentrysymbolplural{\glslabel}}{\glsinsert}}%
```

```

2695         \xmakefirstuc \@glo@etext
2696     }%
2697 {%
2698     #2{\Glsentryplural{\glslabel}}%
2699     {\Glsentrydescplural{\glslabel}}%
2700     {\Glsentrysymbolplural{\glslabel}}{\glsinsert}%
2701 }%
2702 }%
2703 {%

```

First use

```

2704     \ifbool{glscompatible-3.07}%
2705     {%
2706         \protected@edef \@glo@etext{%
2707             #1{\Glsentryfirstplural{\glslabel}}%
2708             {\Glsentrydescplural{\glslabel}}%
2709             {\Glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2710         \xmakefirstuc \@glo@etext
2711     }%
2712 {%
2713     #1{\Glsentryfirstplural{\glslabel}}%
2714     {\Glsentrydescplural{\glslabel}}%
2715     {\Glsentrysymbolplural{\glslabel}}{\glsinsert}%
2716 }%
2717 }%
2718 }%
2719 {%

```

Make all upper case

```

2720     \ifglsused\glslabel
2721     {%

```

Subsequent use

```

2722         \mfirstucMakeUppercase{#2{\Glsentryplural{\glslabel}}%
2723         {\Glsentrydescplural{\glslabel}}%
2724         {\Glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2725     }%
2726     {%

```

First use

```

2727         \mfirstucMakeUppercase{#1{\Glsentryfirstplural{\glslabel}}%
2728         {\Glsentrydescplural{\glslabel}}%
2729         {\Glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2730     }%
2731 }%
2732 }%
2733 {%

```

Singular form

```

2734     \glscapscase
2735     {%

```

Don't adjust case

```
2736      \ifglsused\glslabel
2737      {%
```

Subsequent use

```
2738      #2{\glsentrytext{\glslabel}}%
2739      {\glsentrydesc{\glslabel}}%
2740      {\glsentrysymbol{\glslabel}}{\glsinsert}%
2741      }%
2742      {%
```

First use

```
2743      #1{\glsentryfirst{\glslabel}}%
2744      {\glsentrydesc{\glslabel}}%
2745      {\glsentrysymbol{\glslabel}}{\glsinsert}%
2746      }%
2747      }%
2748      {%
```

Make first letter upper case

```
2749      \ifglsused\glslabel
2750      {%
```

Subsequent use

```
2751      \ifbool{glscompatible-3.07}%
2752      {%
2753      \protected@edef\@glo@etext{%
2754      #2{\glsentrytext{\glslabel}}%
2755      {\glsentrydesc{\glslabel}}%
2756      {\glsentrysymbol{\glslabel}}{\glsinsert}}%
2757      \xmakefirstuc\@glo@etext
2758      }%
2759      {%
2760      #2{\Glsentrytext{\glslabel}}%
2761      {\glsentrydesc{\glslabel}}%
2762      {\glsentrysymbol{\glslabel}}{\glsinsert}%
2763      }%
2764      }%
2765      {%
```

First use

```
2766      \ifbool{glscompatible-3.07}%
2767      {%
2768      \protected@edef\@glo@etext{%
2769      #1{\glsentryfirst{\glslabel}}%
2770      {\glsentrydesc{\glslabel}}%
2771      {\glsentrysymbol{\glslabel}}{\glsinsert}}%
2772      \xmakefirstuc\@glo@etext
2773      }%
2774      {%
2775      #1{\Glsentryfirst{\glslabel}}%
```

```

2776         {\glsentrydesc{\glslabel}}}%
2777         {\glsentrysymbol{\glslabel}}{\glsinsert}}%
2778     }%
2779 }%
2780 }%
2781 {%

```

Make all upper case

```

2782     \ifglsused\glslabel
2783     {%

```

Subsequent use

```

2784         \mfirstucMakeUppercase{#2{\glsentrytext{\glslabel}}}%
2785         {\glsentrydesc{\glslabel}}}%
2786         {\glsentrysymbol{\glslabel}}{\glsinsert}}}%
2787     }%
2788     {%

```

First use

```

2789         \mfirstucMakeUppercase{#1{\glsentryfirst{\glslabel}}}%
2790         {\glsentrydesc{\glslabel}}}%
2791         {\glsentrysymbol{\glslabel}}{\glsinsert}}}%
2792     }%
2793 }%
2794 }%
2795 }%
2796 {%

```

Custom text provided in \glsdisp

```

2797     \ifglsused{\glslabel}%
2798     {%

```

Subsequent use

```

2799         #2{\glscustomtext}%
2800         {\glsentrydesc{\glslabel}}}%
2801         {\glsentrysymbol{\glslabel}}{}%
2802     }%
2803     {%

```

First use

```

2804         #1{\glscustomtext}%
2805         {\glsentrydesc{\glslabel}}}%
2806         {\glsentrysymbol{\glslabel}}{}%
2807     }%
2808 }%
2809 }

```

`\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.

```

2810 \newcommand*{\glsgenentryfmt}{%
2811     \ifdefempty\glscustomtext

```

```

2812  {%
2813    \glsifplural
2814    {%

```

Plural form

```

2815    \glscapscase
2816    {%

```

Don't adjust case

```

2817    \ifglsused\glslabel
2818    {%

```

Subsequent use

```

2819    \glsentryplural{\glslabel}\glsinsert
2820    }%
2821    {%

```

First use

```

2822    \glsentryfirstplural{\glslabel}\glsinsert
2823    }%
2824    }%
2825    {%

```

Make first letter upper case

```

2826    \ifglsused\glslabel
2827    {%

```

Subsequent use.

```

2828    \Glsentryplural{\glslabel}\glsinsert
2829    }%
2830    {%

```

First use

```

2831    \Glsentryfirstplural{\glslabel}\glsinsert
2832    }%
2833    }%
2834    {%

```

Make all upper case

```

2835    \ifglsused\glslabel
2836    {%

```

Subsequent use

```

2837    \mfirstucMakeUppercase
2838    {\glsentryplural{\glslabel}\glsinsert}%
2839    }%
2840    {%

```

First use

```

2841    \mfirstucMakeUppercase
2842    {\glsentryfirstplural{\glslabel}\glsinsert}%
2843    }%
2844    }%

```


2845 }%

2846 {%

Singular form

2847 \glscapscase

2848 {%

Don't adjust case

2849 \ifglused\glslabel

2850 {%

Subsequent use

2851 \glentrytext{\glslabel}\glinsert

2852 }%

2853 {%

First use

2854 \glentryfirst{\glslabel}\glinsert

2855 }%

2856 }%

2857 {%

Make first letter upper case

2858 \ifglused\glslabel

2859 {%

Subsequent use

2860 \Glentrytext{\glslabel}\glinsert

2861 }%

2862 {%

First use

2863 \Glentryfirst{\glslabel}\glinsert

2864 }%

2865 }%

2866 {%

Make all upper case

2867 \ifglused\glslabel

2868 {%

Subsequent use

2869 \mfirstucMakeUppercase{\glentrytext{\glslabel}\glinsert}%

2870 }%

2871 {%

First use

2872 \mfirstucMakeUppercase{\glentryfirst{\glslabel}\glinsert}%

2873 }%

2874 }%

2875 }%

2876 }%

2877 {%

Custom text provided in `\glsdisp`. (The insert is most likely to be empty at this point.)

```
2878 \glscustomtext\glsinsert
2879 }%
2880 }
```

`\glsgenacfmt` Define a generic acronym format that uses the long and short keys (or their plurals) and `\acrfullformat`, `\firstacronymfont` and `\acronymfont`.

```
2881 \newcommand*{\glsgenacfmt}{%
2882 \ifdefempty\glscustomtext
2883 {%
2884 \ifglsused\glslabel
2885 {%
```

Subsequent use:

```
2886 \glsifplural
2887 {%
```

Subsequent plural form:

```
2888 \glscapscase
2889 {%
```

Subsequent plural form, don't adjust case:

```
2890 \acronymfont{\glsentryshortpl{\glslabel}}\glsinsert
2891 }%
2892 {%
```

Subsequent plural form, make first letter upper case:

```
2893 \acronymfont{\Glsentryshortpl{\glslabel}}\glsinsert
2894 }%
2895 {%
```

Subsequent plural form, all caps:

```
2896 \mfirstucMakeUppercase
2897 {\acronymfont{\glsentryshortpl{\glslabel}}\glsinsert}%
2898 }%
2899 }%
2900 {%
```

Subsequent singular form

```
2901 \glscapscase
2902 {%
```

Subsequent singular form, don't adjust case:

```
2903 \acronymfont{\glsentryshort{\glslabel}}\glsinsert
2904 }%
2905 {%
```

Subsequent singular form, make first letter upper case:

```
2906 \acronymfont{\Glsentryshort{\glslabel}}\glsinsert
2907 }%
2908 {%
```

Subsequent singular form, all caps:

```
2909      \mfirstucMakeUppercase
2910      {\acronymfont{\glentryshort{\glslabel}}\glsinsert}%
2911      }%
2912      }%
2913      }%
2914      {%
```

First use:

```
2915      \glsifplural
2916      {%
```

First use plural form:

```
2917      \glscapscase
2918      {%
```

First use plural form, don't adjust case:

```
2919      \genplacrfullformat{\glslabel}{\glsinsert}%
2920      }%
2921      {%
```

First use plural form, make first letter upper case:

```
2922      \Genplacrfullformat{\glslabel}{\glsinsert}%
2923      }%
2924      {%
```

First use plural form, all caps:

```
2925      \mfirstucMakeUppercase
2926      {\genplacrfullformat{\glslabel}{\glsinsert}}}%
2927      }%
2928      }%
2929      {%
```

First use singular form

```
2930      \glscapscase
2931      {%
```

First use singular form, don't adjust case:

```
2932      \genacrfullformat{\glslabel}{\glsinsert}%
2933      }%
2934      {%
```

First use singular form, make first letter upper case:

```
2935      \Genacrfullformat{\glslabel}{\glsinsert}%
2936      }%
2937      {%
```

First use singular form, all caps:

```
2938      \mfirstucMakeUppercase
2939      {\genacrfullformat{\glslabel}{\glsinsert}}}%
2940      }%
2941      }%
2942      }%
```

```
2943 }%
2944 {%
```

User supplied text.

```
2945 \glscustomtext
2946 }%
2947 }
```

genacrfullformat $\backslash\text{genacrfullformat}\{\langle label \rangle\}\{\langle insert \rangle\}$

The full format used by $\backslash\text{glsgenacfmt}$ (singular).

```
2948 \newcommand*\genacrfullformat}[2]{%
2949 \glentrylong{#1}#2\space
2950 (\protect\firstacronymfont{\glentryshort{#1}})%
2951 }
```

Genacrfullformat $\backslash\text{Genacrfullformat}\{\langle label \rangle\}\{\langle insert \rangle\}$

As above but makes the first letter upper case.

```
2952 \newcommand*\Genacrfullformat}[2]{%
2953 \protected@edef\gls@text{\genacrfullformat{#1}{#2}}%
2954 \xmakefirstuc\gls@text
2955 }
```

nplacrfullformat $\backslash\text{genplacrfullformat}\{\langle label \rangle\}\{\langle insert \rangle\}$

The full format used by $\backslash\text{glsgenacfmt}$ (plural).

```
2956 \newcommand*\genplacrfullformat}[2]{%
2957 \glentrylongpl{#1}#2\space
2958 (\protect\firstacronymfont{\glentryshortpl{#1}})%
2959 }
```

nplacrfullformat $\backslash\text{Genplacrfullformat}\{\langle label \rangle\}\{\langle insert \rangle\}$

As above but makes the first letter upper case.

```
2960 \newcommand*\Genplacrfullformat}[2]{%
2961 \protected@edef\gls@text{\genplacrfullformat{#1}{#2}}%
2962 \xmakefirstuc\gls@text
2963 }
```

glsdisplayfirst Deprecated. Kept for backward compatibility.

```
2964 \newcommand*\glsdisplayfirst}[4]{#1#4}
```

`\glsdisplay` Deprecated. Kept for backward compatibility.

```
2965 \newcommand*{\glsdisplay}[4]{#1#4}
```

`\defglsdisplay` Deprecated. Kept for backward compatibility.

```
2966 \newcommand*{\defglsdisplay}[2][\glsdefaulttype]{%
2967   \GlossariesWarning{\string\defglsdisplay\space is now obsolete.^^J
2968   Use \string\defglsentryfmt\space instead}%
2969   \expandafter\def\csname gls@#1@display\endcsname##1##2##3##4{#2}%
2970   \edef\@gls@doentrydef{%
2971     \noexpand\defglsentryfmt[#1]{%
2972       \noexpand\ifcsdef{gls@#1@displayfirst}%
2973       {%
2974         \noexpand\@gls@default@entryfmt
2975         {\noexpand\csuse{gls@#1@displayfirst}}%
2976         {\noexpand\csuse{gls@#1@display}}%
2977       }%
2978       {%
2979         \noexpand\@gls@default@entryfmt
2980         {\noexpand\glsdisplayfirst}%
2981         {\noexpand\csuse{gls@#1@display}}%
2982       }%
2983     }%
2984   }%
2985   \@gls@doentrydef
2986 }
```

`glsdisplayfirst` Deprecated. Kept for backward compatibility.

```
2987 \newcommand*{\defglsdisplayfirst}[2][\glsdefaulttype]{%
2988   \GlossariesWarning{\string\defglsdisplayfirst\space is now obsolete.^^J
2989   Use \string\defglsentryfmt\space instead}%
2990   \expandafter\def\csname gls@#1@displayfirst\endcsname##1##2##3##4{#2}%
2991   \edef\@gls@doentrydef{%
2992     \noexpand\defglsentryfmt[#1]{%
2993       \noexpand\ifcsdef{gls@#1@display}%
2994       {%
2995         \noexpand\@gls@default@entryfmt
2996         {\noexpand\csuse{gls@#1@displayfirst}}%
2997         {\noexpand\csuse{gls@#1@display}}%
2998       }%
2999       {%
3000         \noexpand\@gls@default@entryfmt
3001         {\noexpand\csuse{gls@#1@displayfirst}}%
3002         {\noexpand\glsdisplay}%
3003       }%
3004     }%
3005   }%
3006   \@gls@doentrydef
3007 }
```

Links to glossary entries

The links to glossary entries all have a first optional argument that can be used to change the format and counter of the associated entry number. Except for `\glslink` and `\glsdisp`, the commands like `\gls` have a final optional argument that can be used to insert additional text in the link (this will usually be appended, but can be redefined using `\defentryfmt`). It goes against the \TeX norm to have an optional argument after the mandatory arguments, but it makes more sense to write, say, `\gls{label}[‘s]` rather than, say, `\gls[append=‘s]{label}`. Since these control sequences are defined to include the final square bracket, spaces will be ignored after them. This is likely to lead to confusion as most users would not expect, say, `\gls{<label>}` to ignore following spaces, so `\new@ifnextchar` from the package is required.

The following keys can be used in the first optional argument. The counter key checks that the value is the name of a valid counter.

```
3008 \define@key{glslink}{counter}{%
3009   \ifcsundef{c@#1}%
3010   {%
3011     \PackageError{glossaries}%
3012     {There is no counter called ‘#1’}%
3013     {%
3014       The counter key should have the name of a valid counter
3015       as its value%
3016     }%
3017   }%
3018   {%
3019     \def\@gls@counter{#1}%
3020   }%
3021 }
```

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.

```
3022 \define@key{glslink}{format}{%
3023   \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.

```
3024 \define@boolkey{glslink}{hyper}[true]{}
```

Initialise hyper key.

```
3025 \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.

```
3026 \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.

```
3027 \newcommand*{\glslinkvar}[3]{#1}
```

`\glsifhyper` Now deprecated.

```
3028 \newcommand*{\glsifhyper}[2]{%
3029 \glslinkvar{#1}{#2}{#1}%
3030 \GlossariesWarning{\string\glsifhyper\space is deprecated. Did
3031 you mean \string\glsifhyperon\space or \string\glslinkvar?}%
3032 }
```

`\@gls@hyp@opt` Used by the commands such as `\glslink` to determine whether to modify the hyper option.

```
3033 \newcommand*{\@gls@hyp@opt}[1]{%
3034 \let\glslinkvar\@firstofthree
3035 \let\@gls@hyp@opt@cs#1\relax
3036 \@ifstar{\s@gls@hyp@opt}%
3037 {\@ifnextchar+{\@firstoftwo{\p@gls@hyp@opt}}{#1}}%
3038 }
```

`\s@gls@hyp@opt` Starred version

```
3039 \newcommand*{\s@gls@hyp@opt}[1] []{%
3040 \let\glslinkvar\@secondofthree
3041 \@gls@hyp@opt@cs[hyper=false,#1]}
```

`\p@gls@hyp@opt` Plus version

```
3042 \newcommand*{\p@gls@hyp@opt}[1] []{%
3043 \let\glslinkvar\@thirdofthree
3044 \@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`

```
3045 \newrobustcmd*{\glslink}{%
3046 \@gls@hyp@opt\@gls@@link
3047 }
```

`\@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.

```
3048 \newcommand*{\@gls@@link}[3] [] {%
3049 \glsdoifexistsordo{#2}%
3050 {%
3051 \let\do@gls@link@checkfirsthyper\relax
3052 \@gls@link[#1]{#2}{#3}%
3053 }%
```

Display the specified text. (The entry doesn't exist so there's nothing to link it to.)

```
3054 \glstextformat{#3}%
3055 }%

3056 \glspostlinkhook
3057 }
```

`glspostlinkhook`

```
3058 \newcommand*{\glspostlinkhook}{}
3059 % \end{macrocode}
3060 %\end{macro}
3061 %
3062 %
3063 %\begin{macro}{\@gls@link@checkfirsthyper}
3064 % Check for first use and switch off \gloskey[glslink]{hyper} key
3065 % if hyperlink not wanted. (Should be off if first use and
3066 % hyper=false is on or if first use and both the entry is in an acronym
3067 % list and the acrfootnote setting is on.)
3068 % This assumes the glossary type is stored in \cs{glstype} and the
3069 % label is stored in \cs{glslabel}.
3070 %\changes{4.08}{2014-07-30}{new}
3071 % \begin{macrocode}
3072 \newcommand*{\@gls@link@checkfirsthyper}{%
3073 \ifglsused{\glslabel}%
3074 {%
3075 }%
3076 {%
3077 \gls@checkisacronymlist\glstype
3078 \ifglshyperfirst
3079 \if@glsisacronymlist
3080 \ifglsacrfootnote
3081 \KV@glslink@hyperfalse
3082 \fi
```



```

3083     \fi
3084     \else
3085         \KV@glslink@hyperfalse
3086     \fi
3087 }%

    Allow user to hook into this
3088 \glslinkcheckfirsthyperhook
3089 }

\firsthyperhook    Allow used to hook into the \@gls@link@checkfirsthyper macro
3090 \newcommand*{\glslinkcheckfirsthyperhook}{}

linkpostsetkeys
3091 \newcommand*{\glslinkpostsetkeys}{}

\glsifhyperon    Check the value of the hyper key:
3092 \newcommand{\glsifhyperon}[2]{\ifKV@glslink@hyper#1\else#2\fi}

ablehyperinlist    Disable hyperlink if in the “nohyper” list.
3093 \newcommand*{\do@glstdisablehyperinlist}{%
3094     \expandafter\DTLifinlist\expandafter{\glstype}{\@gls@nohyperlist}%
3095     {\KV@glslink@hyperfalse}}%
3096 }

\let@glslink@opts    Hook to set default options for \@glslink.
3097 \newcommand*{\@gls@setdefault@glslink@opts}{}

\@gls@link
3098 \def\@gls@link[#1]#2#3{%
    Inserting \leavevmode suggested by Donald Arseneau (avoids problem with tabularx).
3099     \leavevmode
3100     \edef\glslabel{\glstdetoklabel{#2}}%

    Save options in \@gls@link@opts and label in \@gls@link@label
3101     \def\@gls@link@opts{#1}%
3102     \let\@gls@link@label\glslabel

3103     \def\@glsnumberformat{glsnumberformat}%
3104     \edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%

    If this is in one of the “nohypertypes” glossaries, suppress the hyperlink by default
3105     \edef\glstype{\csname glo@\glslabel @type\endcsname}%

    Save original setting
3106     \let\org@ifKV@glslink@hyper\ifKV@glslink@hyper

    Set defaults:
3107     \@gls@setdefault@glslink@opts

```

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

```
3108 \do@gl:disablehyperinlist
```

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

```
3109 \do@gls@link@checkfirsthyper
```

```
3110 \setkeys{glslink}{#1}%
```

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

```
3111 \glslinkpostsetkeys
```

Store the entry's counter in `\theglsentrycounter`

```
3112 \@gls@saveentrycounter
```

Define sort key if necessary:

```
3113 \@gls@setsort{\glslabel}%
```

(De-tok'ing done by `\@do@wrglossary`)

```
3114 \@do@wrglossary{#2}%
```

```
3115 \ifKV@glslink@hyper
```

```
3116 \glslink{\glolinkprefix\glslabel}{\glstextformat{#3}}%
```

```
3117 \else
```

```
3118 \glsdonohyperlink{\glolinkprefix\glslabel}{\glstextformat{#3}}%
```

```
3119 \fi
```

Restore original setting

```
3120 \let\ifKV@glslink@hyper\org@ifKV@glslink@hyper
```

```
3121 }
```

`\glolinkprefix`

```
3122 \newcommand*{\glolinkprefix}{glo:}
```

`glsentrycounter` Set default value of entry counter

```
3123 \def\glsentrycounter{\glscounter}%
```

`saveentrycounter` Need to check if using equation counter in align environment:

```
3124 \newcommand*{\@gls@saveentrycounter}{%
```

```
3125 \def\@gls@Hcounter{}}%
```

Are we using equation counter?

```
3126 \ifthenelse{\equal{\@gls@counter}{equation}}{%
```

```
3127 {
```

If we're in align environment, `\xatlevel@` will be defined. (Can't test for `\@currentenv` as may be inside an inner environment.)

```
3128 \ifcsundef{xatlevel@}%
```

```
3129 {%
```

```
3130 \edef\theglsentrycounter{\expandafter\noexpand
```

```
3131 \csname the\@gls@counter\endcsname}%
```

```
3132 }%
```

```

3133   {%
3134       \ifx\xatlevel@\@empty
3135           \edef\theglentrycounter{\expandafter\noexpand
3136               \csname the\@gls@counter\endcsname}%
3137       \else
3138           \savecounters@
3139           \advance\c@equation by 1\relax
3140           \edef\theglentrycounter{\csname the\@gls@counter\endcsname}%

```

Check if hyperref version of this counter

```

3141       \ifcsundef{theH\@gls@counter}%
3142       {%
3143           \def\@gls@Hcounter{\theglentrycounter}%
3144       }%
3145       {%
3146           \def\@gls@Hcounter{\csname theH\@gls@counter\endcsname}%
3147       }%
3148       \protected@edef\theHglentrycounter{\@gls@Hcounter}%
3149       \restorecounters@
3150   \fi
3151 }%
3152 }%
3153 {%

```

Not using equation counter so no special measures:

```

3154   \edef\theglentrycounter{\expandafter\noexpand
3155       \csname the\@gls@counter\endcsname}%
3156 }%

```

Check if hyperref version of this counter

```

3157   \ifx\@gls@Hcounter\@empty
3158       \ifcsundef{theH\@gls@counter}%
3159       {%
3160           \def\theHglentrycounter{\theglentrycounter}%
3161       }%
3162       {%
3163           \protected@edef\theHglentrycounter{\expandafter\noexpand
3164               \csname theH\@gls@counter\endcsname}%
3165       }%
3166   \fi
3167 }

```

`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.

```

3168 \def\@set@glo@numformat#1#2#3#4{%
3169     \expandafter\@glo@check@mkidxrangechar#3\@nil
3170     \protected@edef#1{%

```

```

3171 \@glo@prefix setentrycounter[#4]{#2}%
3172 \expandafter\string\csname\@glo@suffix\endcsname
3173 }%
3174 \@gls@checkmkidxchars#1%
3175 }

```

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.

```

3176 \def\@glo@check@mkidxrangechar#1#2\@nil{%
3177 \if#1(\relax
3178 \def\@glo@prefix{(%
3179 \if\relax#2\relax
3180 \def\@glo@suffix{glsnumberformat}%
3181 \else
3182 \def\@glo@suffix{#2}%
3183 \fi
3184 \else
3185 \if#1)\relax
3186 \def\@glo@prefix{)%
3187 \if\relax#2\relax
3188 \def\@glo@suffix{glsnumberformat}%
3189 \else
3190 \def\@glo@suffix{#2}%
3191 \fi
3192 \else
3193 \def\@glo@prefix{}\def\@glo@suffix{#1#2}%
3194 \fi
3195 \fi}

```

\@gls@escbsdq Escape backslashes and double quote marks. The argument must be a control sequence.

```

3196 \newcommand*{\@gls@escbsdq}[1]{%
3197 \def\@gls@checkedmkidx{%
3198 \let\gls@xdystring=#1\relax
3199 \@onelevel@sanitize\gls@xdystring
3200 \edef\do@gls@xdycheckbackslash{%
3201 \noexpand\@gls@xdycheckbackslash\gls@xdystring\noexpand\@nil
3202 \@backslashchar\@backslashchar\noexpand\null}%
3203 \do@gls@xdycheckbackslash
3204 \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdystring}%
3205 \def\@gls@checkedmkidx{%
3206 \expandafter\@gls@xdycheckquote\gls@xdystring\@nil""\null
3207 \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdystring}%

```

Unsanitize \gls@numberpage, \gls@alphpage, \gls@Alphpage and \gls@romanpage (thanks to David Carlisle for the suggestion.)

```

3208 \@for\@gls@tmp:=\gls@protected@pagefmts\do
3209 {%
3210 \edef\@gls@sanitized@tmp{\expandafter\@gobble\string\\ \expandonce\@gls@tmp}%

```

```

3211 \@onelevel@sanitize\@gls@sanitized@tmp
3212 \edef\gls@dosubst{%
3213 \noexpand\DTLsubstituteall\noexpand\gls@xdystring
3214 {\@gls@sanitized@tmp}{\expandonce\@gls@tmp}%
3215 }%
3216 \gls@dosubst
3217 }%

```

Assign to required control sequence

```

3218 \let#1=\gls@xdystring
3219 }

```

Catch special characters (argument must be a control sequence):

checkmkidxchars

```

3220 \newcommand{\@gls@checkmkidxchars}[1]{%
3221 \ifglxsindy
3222 \@gls@escbsdq{#1}%
3223 \else
3224 \def\@gls@checkedmkidx{%
3225 \expandafter\@gls@checkquote#1\@nil""\null
3226 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3227 \def\@gls@checkedmkidx{%
3228 \expandafter\@gls@checkescquote#1\@nil\""\null
3229 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3230 \def\@gls@checkedmkidx{%
3231 \expandafter\@gls@checkescactual#1\@nil\?\?\null
3232 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3233 \def\@gls@checkedmkidx{%
3234 \expandafter\@gls@checkactual#1\@nil??\null
3235 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3236 \def\@gls@checkedmkidx{%
3237 \expandafter\@gls@checkbar#1\@nil||\null
3238 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3239 \def\@gls@checkedmkidx{%
3240 \expandafter\@gls@checkescbar#1\@nil\\|\null
3241 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3242 \def\@gls@checkedmkidx{%
3243 \expandafter\@gls@checklevel#1\@nil!!\null
3244 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3245 \fi
3246 }

```

Update the control sequence and strip trailing \@nil:

s@updatechecked

```

3247 \def\@gls@updatechecked#1\@nil#2{\def#2{#1}}

```

\@gls@tmpb Define temporary token

```

3248 \newtoks\@gls@tmpb

```

@gls@checkquote Replace " with "" since " is a makeindex special character.

```
3249 \def\@gls@checkquote#1"#2"#3\null{%
3250   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3251   \toks@={#1}%
3252   \ifx\null#2\null
3253   \ifx\null#3\null
3254   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3255   \def\@gls@checkquote{\relax}%
3256   \else
3257   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3258     \@gls@quotechar\@gls@quotechar\@gls@quotechar\@gls@quotechar}%
3259   \def\@gls@checkquote{\@gls@checkquote#3\null}%
3260   \fi
3261   \else
3262   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3263     \@gls@quotechar\@gls@quotechar}%
3264   \ifx\null#3\null
3265     \def\@gls@checkquote{\@gls@checkquote#2""\null}%
3266   \else
3267     \def\@gls@checkquote{\@gls@checkquote#2"#3\null}%
3268   \fi
3269   \fi
3270   \@gls@checkquote
3271 }
```

s@checkescquote Do the same for \":

```
3272 \def\@gls@checkescquote#1\"#2\"#3\null{%
3273   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3274   \toks@={#1}%
3275   \ifx\null#2\null
3276   \ifx\null#3\null
3277   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3278   \def\@gls@checkescquote{\relax}%
3279   \else
3280   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3281     \@gls@quotechar\string\"@gls@quotechar
3282     \@gls@quotechar\string\"@gls@quotechar}%
3283   \def\@gls@checkescquote{\@gls@checkescquote#3\null}%
3284   \fi
3285   \else
3286   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3287     \@gls@quotechar\string\"@gls@quotechar}%
3288   \ifx\null#3\null
3289     \def\@gls@checkescquote{\@gls@checkescquote#2\"\" \null}%
3290   \else
3291     \def\@gls@checkescquote{\@gls@checkescquote#2\"#3\null}%
3292   \fi
3293   \fi
3294   \@gls@checkescquote
```

3295 }

@checkescactual Similarly for \? (which is replaces @ as makeindex's special character):

```
3296 \def\@gls@checkescactual#1\?#2\?#3\null{%
3297 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3298 \toks@={#1}%
3299 \ifx\null#2\null
3300 \ifx\null#3\null
3301 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3302 \def\@gls@checkescactual{\relax}%
3303 \else
3304 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3305 \@gls@quotechar\string\" \@gls@actualchar
3306 \@gls@quotechar\string\" \@gls@actualchar}%
3307 \def\@gls@checkescactual{\@gls@checkescactual#3\null}%
3308 \fi
3309 \else
3310 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3311 \@gls@quotechar\string\" \@gls@actualchar}%
3312 \ifx\null#3\null
3313 \def\@gls@checkescactual{\@gls@checkescactual#2\?\?\null}%
3314 \else
3315 \def\@gls@checkescactual{\@gls@checkescactual#2\?#3\null}%
3316 \fi
3317 \fi
3318 \@gls@checkescactual
3319 }
```

gls@checkeschar Similarly for \||:

```
3320 \def\@gls@checkeschar#1\||#2\||#3\null{%
3321 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3322 \toks@={#1}%
3323 \ifx\null#2\null
3324 \ifx\null#3\null
3325 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3326 \def\@gls@checkeschar{\relax}%
3327 \else
3328 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3329 \@gls@quotechar\string\" \@gls@encapchar
3330 \@gls@quotechar\string\" \@gls@encapchar}%
3331 \def\@gls@checkeschar{\@gls@checkeschar#3\null}%
3332 \fi
3333 \else
3334 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3335 \@gls@quotechar\string\" \@gls@encapchar}%
3336 \ifx\null#3\null
3337 \def\@gls@checkeschar{\@gls@checkeschar#2\||\|\null}%
3338 \else
3339 \def\@gls@checkeschar{\@gls@checkeschar#2\||#3\null}%

```

```

3340 \fi
3341 \fi
3342 \@gls@checkescbar
3343 }

```

s@checkesclevel Similarly for \!:

```

3344 \def\@gls@checkesclevel#1\!#2\!#3\null{%
3345 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3346 \toks@={#1}%
3347 \ifx\null#2\null
3348 \ifx\null#3\null
3349 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3350 \def\@gls@checkesclevel{\relax}%
3351 \else
3352 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3353 \@gls@quotechar\string"\@gls@levelchar
3354 \@gls@quotechar\string"\@gls@levelchar}%
3355 \def\@gls@checkesclevel{\@gls@checkesclevel#3\null}%
3356 \fi
3357 \else
3358 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3359 \@gls@quotechar\string"\@gls@levelchar}%
3360 \ifx\null#3\null
3361 \def\@gls@checkesclevel{\@gls@checkesclevel#2\!\!\null}%
3362 \else
3363 \def\@gls@checkesclevel{\@gls@checkesclevel#2\!#3\null}%
3364 \fi
3365 \fi
3366 \@gls@checkesclevel
3367 }

```

\@gls@checkbar and for |:

```

3368 \def\@gls@checkbar#1|#2|#3\null{%
3369 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3370 \toks@={#1}%
3371 \ifx\null#2\null
3372 \ifx\null#3\null
3373 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3374 \def\@gls@checkbar{\relax}%
3375 \else
3376 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3377 \@gls@quotechar\@gls@encapchar\@gls@quotechar\@gls@encapchar}%
3378 \def\@gls@checkbar{\@gls@checkbar#3\null}%
3379 \fi
3380 \else
3381 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3382 \@gls@quotechar\@gls@encapchar}%
3383 \ifx\null#3\null
3384 \def\@gls@checkbar{\@gls@checkbar#2||\null}%

```



```

3385 \else
3386 \def\@gls@checkbar{\@gls@checkbar#2|#3\null}%
3387 \fi
3388 \fi
3389 \@gls@checkbar
3390 }

```

@gls@checklevel and for !:

```

3391 \def\@gls@checklevel#1!#2!#3\null{%
3392 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3393 \toks@={#1}%
3394 \ifx\null#2\null
3395 \ifx\null#3\null
3396 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3397 \def\@gls@checklevel{\relax}%
3398 \else
3399 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3400 \@gls@quotechar\@gls@levelchar\@gls@quotechar\@gls@levelchar}%
3401 \def\@gls@checklevel{\@gls@checklevel#3\null}%
3402 \fi
3403 \else
3404 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3405 \@gls@quotechar\@gls@levelchar}%
3406 \ifx\null#3\null
3407 \def\@gls@checklevel{\@gls@checklevel#2!!\null}%
3408 \else
3409 \def\@gls@checklevel{\@gls@checklevel#2!#3\null}%
3410 \fi
3411 \fi
3412 \@gls@checklevel
3413 }

```

gls@checkactual and for ?:

```

3414 \def\@gls@checkactual#1?#2?#3\null{%
3415 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3416 \toks@={#1}%
3417 \ifx\null#2\null
3418 \ifx\null#3\null
3419 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3420 \def\@gls@checkactual{\relax}%
3421 \else
3422 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3423 \@gls@quotechar\@gls@actualchar\@gls@quotechar\@gls@actualchar}%
3424 \def\@gls@checkactual{\@gls@checkactual#3\null}%
3425 \fi
3426 \else
3427 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3428 \@gls@quotechar\@gls@actualchar}%
3429 \ifx\null#3\null

```

```

3430     \def\@gls@checkactual{\@gls@checkactual#2??\null}%
3431   \else
3432     \def\@gls@checkactual{\@gls@checkactual#2?#3\null}%
3433   \fi
3434 \fi
3435 \@gls@checkactual
3436 }

```

s@xdycheckquote As before but for use with xindy

```

3437 \def\@gls@xdycheckquote#1"#2"#3\null{%
3438   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3439   \toks@={#1}%
3440   \ifx\null#2\null
3441     \ifx\null#3\null
3442       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3443       \def\@gls@xdycheckquote{\relax}%
3444     \else
3445       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3446         \string"\string"}%
3447       \def\@gls@xdycheckquote{\@gls@xdycheckquote#3\null}%
3448     \fi
3449   \else
3450     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3451       \string"}%
3452     \ifx\null#3\null
3453       \def\@gls@xdycheckquote{\@gls@xdycheckquote#2""\null}%
3454     \else
3455       \def\@gls@xdycheckquote{\@gls@xdycheckquote#2"#3\null}%
3456     \fi
3457   \fi
3458   \@gls@xdycheckquote
3459 }

```

ycheckbackslash Need to escape all backslashes for xindy. Define command that will define \@gls@xdycheckbackslash

```

3460 \edef\def@gls@xdycheckbackslash{%
3461   \noexpand\def\noexpand\@gls@xdycheckbackslash##1\@backslashchar
3462     ##2\@backslashchar##3\noexpand\null{%
3463     \noexpand\@gls@tmpb=\noexpand\expandafter
3464       {\noexpand\@gls@checkedmkidx}%
3465     \noexpand\toks@={##1}%
3466     \noexpand\ifx\noexpand\null##2\noexpand\null
3467       \noexpand\ifx\noexpand\null##3\noexpand\null
3468         \noexpand\edef\noexpand\@gls@checkedmkidx{%
3469           \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
3470         \noexpand\def\noexpand\@gls@xdycheckbackslash{\relax}%
3471       \noexpand\else
3472         \noexpand\edef\noexpand\@gls@checkedmkidx{%
3473           \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
3474           \@backslashchar\@backslashchar\@backslashchar\@backslashchar}%

```

```

3475 \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3476   \noexpand\@gls@xdycheckbackslash##3\noexpand\null}%
3477   \noexpand\fi
3478 \noexpand\else
3479   \noexpand\edef\noexpand\@gls@checkedmkidx{%
3480     \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
3481     \@backslashchar\@backslashchar}%
3482 \noexpand\ifx\noexpand\null##3\noexpand\null
3483   \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3484     \noexpand\@gls@xdycheckbackslash##2\@backslashchar
3485     \@backslashchar\noexpand\null}%
3486   \noexpand\else
3487     \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3488       \noexpand\@gls@xdycheckbackslash##2\@backslashchar
3489       ##3\noexpand\null}%
3490   \noexpand\fi
3491 \noexpand\fi
3492 \noexpand\@gls@xdycheckbackslash
3493 }%
3494 }

```

Now go ahead and define \@gls@xdycheckbackslash

```

3495 \def@gls@xdycheckbackslash

```

lsdohypertarget

```

3496 \newlength\gls@tmplen
3497 \newcommand*{\glsdohypertarget}[2]{%
3498   \settoheight{\gls@tmplen}{#2}%
3499   \raisebox{\gls@tmplen}{\hypertarget{#1}{}}#2%
3500 }

```

\glsdohyperlink

```

3501 \newcommand*{\glsdohyperlink}[2]{\hyperlink{#1}{#2}}

```

lsdonohyperlink

```

3502 \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.

```

3503 \ifcsundef{hyperlink}%
3504 {%
3505   \let\@glslink\glsdonohyperlink
3506 }%
3507 {%
3508   \let\@glslink\glsdohyperlink
3509 }

```

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

```
3510 \ifcsundef{hypertarget}%
3511 {%
3512   \let\@glstarget\@secondoftwo
3513 }%
3514 {%
3515   \let\@glstarget\glsdohypertarget
3516 }
```

Glossary hyperlinks can be disabled using `\glsdisablehyper` (effect can be localised):

`\glsdisablehyper`

```
3517 \newcommand{\glsdisablehyper}{%
3518   \KV@glslink@hyperfalse
3519   \let\@glslink\glsdonohyperlink
3520   \let\@glstarget\@secondoftwo
3521 }
```

Glossary hyperlinks can be enabled using `\glsenablehyper` (effect can be localised):

`\glsenablehyper`

```
3522 \newcommand{\glsenablehyper}{%
3523   \KV@glslink@hypertrue
3524   \let\@glslink\glsdohyperlink
3525   \let\@glstarget\glsdohypertarget
3526 }
```

Provide some convenience commands if not already defined:

```
3527 \providecommand{\@firstofthree}[3]{#1}
3528 \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`

```
3529 \newrobustcmd*{\gls}{\@gls@hyp@opt\@gls}
```

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

`\@gls`

```
3530 \newcommand*{\@gls}[2] [] {%
3531   \new@ifnextchar[{\@gls@{#1}{#2}}{\@gls@{#1}{#2} []}]%
3532 }
```

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

```
3533 \def\@gls@#1#2[#3]{%
3534   \glsdoifexists{#2}%
3535   {%
3536     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3537     \let\glsifplural\@secondoftwo
3538     \let\glsupcase\@firstofthree
3539     \let\glscustomtext\@empty
3540     \def\glsinsert{#3}%
```

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

```
3541   \def\@glo@text{\csname gls@\gls@type @entryfmt\endcsname}%
```

Call `\@gls@link`. If footnote package option has been used and the glossary type is `\acronymtype`, suppress hyperlink for first use. Likewise if the `hyperfirst=false` package option is used.

```
3542   \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3543   \ifKV@glslink@local
3544     \glslocalunset{#2}%
3545   \else
3546     \glsunset{#2}%
3547   \fi
3548   }%

3549   \glspostlinkhook
3550 }
```

`\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`

```
3551 \newrobustcmd*{\Gls}{\@gls@hyp@opt\@Gls}
```

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

```
3552 \newcommand*{\@Gls}[2] [] {%
3553   \new@ifnextchar[{\@Gls@{#1}{#2}}{\@Gls@{#1}{#2} []}]%
3554 }
```

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

```
3555 \def\@Gls@#1#2[#3]{%
3556   \glsdoifexists{#2}%
3557   {%
3558     \let\do@glsl@link@checkfirsthyper\@glsl@link@checkfirsthyper

3559     \let\glsl@plural\@secondoftwo
3560     \let\glscapscase\@secondofthree
3561     \let\glscustomtext\@empty
3562     \def\glslinsert{#3}%
```

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

```
3563   \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%
```

Call `\@glsl@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.

```
3564   \@glsl@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3565   \ifKV@glsl@link@local
3566     \glsl@localunset{#2}%
3567   \else
3568     \glsl@unset{#2}%
3569   \fi
3570 }%
```

```
3571 \glspostlinkhook
3572 }
```

`\GLS` behaves like `\gls`, but the link text is converted to uppercase:

`\GLS`

```
3573 \newrobustcmd*{\GLS}{\@glsl@hyp@opt\@GLS}
```

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

```
3574 \newcommand*{\@GLS}[2][\@GLS@#1]{%
3575   \new@ifnextchar[\@GLS@{#1}{#2}}{\@GLS@{#1}{#2}}}%
3576 }
```

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

```
3577 \def\@GLS@#1#2[#3]{%
3578   \glsdoifexists{#2}%
3579   {%
3580     \let\do@glsl@link@checkfirsthyper\@glsl@link@checkfirsthyper

3581     \let\glsl@plural\@secondoftwo
3582     \let\glscapscase\@thirdofthree
3583     \let\glscustomtext\@empty
3584     \def\glslinsert{#3}%
```

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

```
3585 \def\@glo@text{\csname gls@\gls@type @entryfmt\endcsname}%
```

Call `\@gls@link`. If footnote package option has been used and the glossary type is `\acronym@type`, suppress hyperlink for first use. Likewise if the `hyperfirst=false` package option is used.

```
3586 \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3587 \ifKV@gls@link@local
3588 \glslocalunset{#2}%
3589 \else
3590 \glsunset{#2}%
3591 \fi
3592 }%
```

```
3593 \gls@postlinkhook
3594 }
```

`\glspl` behaves in the same way as `\gls` except it uses the plural form.

`\glspl`

```
3595 \newrobustcmd*{\glspl}{\@gls@hyp@opt\@glspl}
```

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

```
3596 \newcommand*{\@glspl}[2][{}]{%
3597 \new@ifnextchar[\@glspl@{#1}{#2}]{\@glspl@{#1}{#2}[{}]}%
3598 }
```

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

```
3599 \def\@glspl@#1#2[#3]{%
3600 \glsdoifexists{#2}%
3601 {%
3602 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3603 \let\glsifplural\@firstoftwo
3604 \let\gls@scapscase\@firstofthree
3605 \let\gls@customtext\@empty
3606 \def\glsinsert{#3}%
```

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

```
3607 \def\@glo@text{\csname gls@\gls@type @entryfmt\endcsname}%
```

Call `\@gls@link`. If footnote package option has been used and the glossary type is `\acronym@type`, suppress hyperlink for first use. Likewise if the `hyperfirst=false` package option is used.

```
3608 \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3609 \ifKV@glslink@local
3610 \glsllocalunset{#2}%
3611 \else
3612 \glsunset{#2}%
3613 \fi
3614 }%

3615 \glspostlinkhook
3616 }
```

`\Glspl` behaves in the same way as `\glsp1`, except that the first letter of the link text is converted to uppercase (as with `\Gls`, if the first letter has an accent, it will need to be grouped).

`\Glspl`

```
3617 \newrobustcmd*{\Glspl}{\@gls@hyp@opt\@Glspl}
```

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

```
3618 \newcommand*{\@Glspl}[2][{}]{%
3619 \new@ifnextchar[{\@Glspl@{#1}{#2}}{\@Glspl@{#1}{#2}[]}%
3620 }
```

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

```
3621 \def\@Glspl@#1#2[#3]{%
3622 \glstoifexists{#2}%
3623 {%
3624 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3625 \let\glsifplural\@firstoftwo
3626 \let\glscapscase\@secondofthree
3627 \let\glscustomtext\@empty
3628 \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`.

```
3629 \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.

```
3630 \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3631 \ifKV@glslink@local
3632 \glsllocalunset{#2}%
3633 \else
3634 \glsunset{#2}%
3635 \fi
3636 }%
```



```

3637 \glspostlinkhook
3638 }

```

\GLSp1 behaves like \glsp1 except that all the link text is converted to uppercase.

\GLSp1

```

3639 \newrobustcmd*{\GLSp1}{\@gls@hyp@opt\@GLSp1}

```

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

```

3640 \newcommand*{\@GLSp1}[2][\@GLSp1@{#1}{#2}]{\@GLSp1@{#1}{#2}[]}
3641 \new@ifnextchar[\@GLSp1@{#1}{#2}]{\@GLSp1@{#1}{#2}[]}
3642 }

```

\@GLSp1 Read in the final optional argument:

```

3643 \def\@GLSp1@#1#2[#3]{%
3644   \glsdoifexists{#2}%
3645   {%
3646     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
3647     \let\glsifplural\@firstoftwo
3648     \let\glsapscase\@thirdofthree
3649     \let\glscustomtext\@empty
3650     \def\glsinsert{#3}%

```

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

```

3651 \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.

```

3652 \@gls@link[#1]{#2}{\@glo@text}%

```

Indicate that this entry has now been used

```

3653 \ifKV@glslink@local
3654   \glslocalunset{#2}%
3655 \else
3656   \glsunset{#2}%
3657 \fi
3658 }%

```

```

3659 \glspostlinkhook
3660 }

```

\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:

```

3661 \newrobustcmd*{\glsdisp}{\@gls@hyp@opt\@glsdisp}

```

Defined the un-starred form.

`\@glsdisp`

```
3662 \newcommand*{\@glsdisp}[3] [] {%
3663   \glsdoifexists{#2}{%

3664     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3665     \let\glsifplural\@secondoftwo
3666     \let\glscapscase\@firstofthree
3667     \def\glscustomtext{#3}%
3668     \def\glsinsert{}}%
```

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

```
3669   \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.

```
3670   \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3671   \ifKV@glslink@local
3672     \glslocalunset{#2}%
3673   \else
3674     \glsunset{#2}%
3675   \fi
3676 }%
```

```
3677 \glspostlinkhook
3678 }
```

`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.

```
3679 \newcommand*{\@gls@link@nocheckfirsthyper}{}
```

`@gls@field@link`

```
3680 \newcommand{\@gls@field@link}[3] {%
3681   \glsdoifexists{#2}%
3682   {%
3683     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
3684     \@gls@link[#1]{#2}{#3}%
3685   }%

3686   \glspostlinkhook
3687 }
```

`\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`

```
3688 \newrobustcmd*{\glstext}{\@gls@hyp@opt\@glstext}
```

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

```
3689 \newcommand*{\@glstext}[2] [] {%
```

```
3690   \new@ifnextchar[{\@glstext@{#1}{#2}}{\@glstext@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3691 \def\@glstext@#1#2[#3] {%
```

```
3692   \@gls@field@link{#1}{#2}{\glstentrytext{#2}#3}%
```

```
3693 }
```

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

`\GLStext`

```
3694 \newrobustcmd*{\GLStext}{\@gls@hyp@opt\@GLStext}
```

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

```
3695 \newcommand*{\@GLStext}[2] [] {%
```

```
3696   \new@ifnextchar[{\@GLStext@{#1}{#2}}{\@GLStext@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3697 \def\@GLStext@#1#2[#3] {%
```

```
3698   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glstentrytext{#2}#3}}%
```

```
3699 }
```

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

`\Glstext`

```
3700 \newrobustcmd*{\Glstext}{\@gls@hyp@opt\@Glstext}
```

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

```
3701 \newcommand*{\@Glstext}[2] [] {%
```

```
3702   \new@ifnextchar[{\@Glstext@{#1}{#2}}{\@Glstext@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3703 \def\@Glstext@#1#2[#3] {%
```

```
3704   \@gls@field@link{#1}{#2}{\Glstentrytext{#2}#3}%
```

```
3705 }
```

`\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`

```
3706 \newrobustcmd*{\glsfirst}{\@gls@hyp@opt\@glsfirst}
```

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

```
3707 \newcommand*{\@glsfirst}[2] [] {%
```

```
3708   \new@ifnextchar[{\@glsfirst@{#1}{#2}}{\@glsfirst@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3709 \def\@glsfirst@#1#2[#3]{%
3710   \gls@field@link{#1}{#2}{\glsentryfirst{#2}#3}%
3711 }
```

\Glsfirst behaves like \glsfirst except it displays the first letter in uppercase.

\Glsfirst

```
3712 \newrobustcmd*{\Glsfirst}{\@gls@hyp@opt\@Glsfirst}
```

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

```
3713 \newcommand*{\@Glsfirst}[2] [] {%
3714   \new@ifnextchar[{\@Glsfirst@{#1}{#2}}{\@Glsfirst@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3715 \def\@Glsfirst@#1#2[#3]{%
3716   \gls@field@link{#1}{#2}{\glsentryfirst{#2}#3}%
3717 }
```

\GLSfirst behaves like \Glsfirst except it displays the text in uppercase.

\GLSfirst

```
3718 \newrobustcmd*{\GLSfirst}{\@gls@hyp@opt\@GLSfirst}
```

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

```
3719 \newcommand*{\@GLSfirst}[2] [] {%
3720   \new@ifnextchar[{\@GLSfirst@{#1}{#2}}{\@GLSfirst@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3721 \def\@GLSfirst@#1#2[#3]{%
3722   \gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryfirst{#2}#3}}%
3723 }
```

\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

```
3724 \newrobustcmd*{\glsplural}{\@gls@hyp@opt\@glsplural}
```

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

```
3725 \newcommand*{\@glsplural}[2] [] {%
3726   \new@ifnextchar[{\@glsplural@{#1}{#2}}{\@glsplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3727 \def\@glsplural@#1#2[#3]{%
3728   \gls@field@link{#1}{#2}{\glsentryplural{#2}#3}%
3729 }
```

\Glsplural behaves like \glsplural except that the first letter is converted to uppercase.

\Glsplural

```
3730 \newrobustcmd*{\Glsplural}{\@gls@hyp@opt\@Glsplural}
```

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

```
3731 \newcommand*{\@Glsplural}[2] [] {%  
3732   \new@ifnextchar [{\@Glsplural@{#1}{#2}}{\@Glsplural@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3733 \def\@Glsplural@#1#2[#3] {%  
3734   \@gls@field@link{#1}{#2}{\Glsentryplural{#2}#3}%  
3735 }
```

\Glsplural behaves like \glsplural except that the text is converted to uppercase.

\Glsplural

```
3736 \newrobustcmd*{\Glsplural}{\@gls@hyp@opt\@Glsplural}
```

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

```
3737 \newcommand*{\@Glsplural}[2] [] {%  
3738   \new@ifnextchar [{\@Glsplural@{#1}{#2}}{\@Glsplural@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3739 \def\@Glsplural@#1#2[#3] {%  
3740   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryplural{#2}#3}}%  
3741 }
```

\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

```
3742 \newrobustcmd*{\glsfirstplural}{\@gls@hyp@opt\@glsfirstplural}
```

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

```
3743 \newcommand*{\@glsfirstplural}[2] [] {%  
3744   \new@ifnextchar [{\@glsfirstplural@{#1}{#2}}{\@glsfirstplural@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3745 \def\@glsfirstplural@#1#2[#3] {%  
3746   \@gls@field@link{#1}{#2}{\glsentryfirstplural{#2}#3}%  
3747 }
```

\Glsfirstplural behaves like \glsfirstplural except that the first letter is converted to uppercase.

\Glsfirstplural

```
3748 \newrobustcmd*{\Glsfirstplural}{\@gls@hyp@opt\@Glsfirstplural}
```

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

```
3749 \newcommand*{\@Glsfirstplural}[2] [] {%  
3750   \new@ifnextchar [{\@Glsfirstplural@{#1}{#2}}{\@Glsfirstplural@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3751 \def\@Glsfirstplural@#1#2[#3] {%  
3752   \@gls@field@link{#1}{#2}{\Glsentryfirstplural{#2}#3}%  
3753 }
```

`\GLSfirstplural` behaves like `\glsfirstplural` except that the link text is converted to uppercase.

`\GLSfirstplural`

```
3754 \newrobustcmd*{\GLSfirstplural}{\@gls@hyp@opt\@GLSfirstplural}
```

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

```
3755 \newcommand*{\@GLSfirstplural}[2] [] {%
```

```
3756   \new@ifnextchar[{\@GLSfirstplural@{#1}{#2}}{\@GLSfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3757 \def\@GLSfirstplural@#1#2[#3] {%
```

```
3758   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryfirstplural{#2}{#3}}}%
```

```
3759 }
```

`\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`

```
3760 \newrobustcmd*{\glsname}{\@gls@hyp@opt\@glsname}
```

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

```
3761 \newcommand*{\@glsname}[2] [] {%
```

```
3762   \new@ifnextchar[{\@glsname@{#1}{#2}}{\@glsname@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3763 \def\@glsname@#1#2[#3] {%
```

```
3764   \@gls@field@link{#1}{#2}{\glsentryname{#2}{#3}}%
```

```
3765 }
```

`\Glsname` behaves like `\glsname` except that the first letter is converted to uppercase.

`\Glsname`

```
3766 \newrobustcmd*{\Glsname}{\@gls@hyp@opt\@Glsname}
```

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

```
3767 \newcommand*{\@Glsname}[2] [] {%
```

```
3768   \new@ifnextchar[{\@Glsname@{#1}{#2}}{\@Glsname@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3769 \def\@Glsname@#1#2[#3] {%
```

```
3770   \@gls@field@link{#1}{#2}{\Glsentryname{#2}{#3}}%
```

```
3771 }
```

`\GLSname` behaves like `\glsname` except that the link text is converted to uppercase.

`\GLSname`

```
3772 \newrobustcmd*{\GLSname}{\@gls@hyp@opt\@GLSname}
```

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

```
3773 \newcommand*{\@GLSname}[2] [] {%
```

```
3774   \new@ifnextchar[{\@GLSname@{#1}{#2}}{\@GLSname@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3775 \def\@GLSname@#1#2[#3]{%
3776   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryname{#2}#3}}%
3777 }
```

\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

```
3778 \newrobustcmd*{\glsdesc}{\@gls@hyp@opt\@glsdesc}
```

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

```
3779 \newcommand*{\@glsdesc}[2][{}]{%
3780   \new@ifnextchar[{\@glsdesc@{#1}{#2}}{\@glsdesc@{#1}{#2}[]}}
```

Read in the final optional argument:

```
3781 \def\@glsdesc@#1#2[#3]{%
3782   \@gls@field@link{#1}{#2}{\glsentrydesc{#2}#3}%
3783 }
```

\Glsdesc behaves like \glsdesc except that the first letter is converted to uppercase.

\Glsdesc

```
3784 \newrobustcmd*{\Glsdesc}{\@gls@hyp@opt\@Glsdesc}
```

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

```
3785 \newcommand*{\@Glsdesc}[2][{}]{%
3786   \new@ifnextchar[{\@Glsdesc@{#1}{#2}}{\@Glsdesc@{#1}{#2}[]}}
```

Read in the final optional argument:

```
3787 \def\@Glsdesc@#1#2[#3]{%
3788   \@gls@field@link{#1}{#2}{\Glsentrydesc{#2}#3}%
3789 }
```

\GLSdesc behaves like \glsdesc except that the link text is converted to uppercase.

\GLSdesc

```
3790 \newrobustcmd*{\GLSdesc}{\@gls@hyp@opt\@GLSdesc}
```

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

```
3791 \newcommand*{\@GLSdesc}[2][{}]{%
3792   \new@ifnextchar[{\@GLSdesc@{#1}{#2}}{\@GLSdesc@{#1}{#2}[]}}
```

Read in the final optional argument:

```
3793 \def\@GLSdesc@#1#2[#3]{%
3794   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrydesc{#2}#3}}%
3795 }
```

\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

```
3796 \newrobustcmd*{\glsdescplural}{\@gls@hyp@opt\@glsdescplural}
```

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

```
3797 \newcommand*{\@glsdescplural}[2] [] {%  
3798   \new@ifnextchar [{\@glsdescplural@{#1}{#2}}{\@glsdescplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3799 \def\@glsdescplural@#1#2[#3] {%  
3800   \@gls@field@link{#1}{#2}{\glsentrydescplural{#2}#3}%  
3801 }
```

\Glsdescplural behaves like \glsdescplural except that the first letter is converted to uppercase.

\Glsdescplural

```
3802 \newrobustcmd*{\Glsdescplural}{\@gls@hyp@opt\@Glsdescplural}
```

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

```
3803 \newcommand*{\@GLSdescplural}[2] [] {%  
3804   \new@ifnextchar [{\@GLSdescplural@{#1}{#2}}{\@GLSdescplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3805 \def\@GLSdescplural@#1#2[#3] {%  
3806   \@gls@field@link{#1}{#2}{\Glsentrydescplural{#2}#3}%  
3807 }
```

\GLSdescplural behaves like \glsdescplural except that the link text is converted to uppercase.

\GLSdescplural

```
3808 \newrobustcmd*{\GLSdescplural}{\@gls@hyp@opt\@GLSdescplural}
```

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

```
3809 \newcommand*{\@GLSdescplural}[2] [] {%  
3810   \new@ifnextchar [{\@GLSdescplural@{#1}{#2}}{\@GLSdescplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3811 \def\@GLSdescplural@#1#2[#3] {%  
3812   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrydescplural{#2}#3}}%  
3813 }
```

\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

```
3814 \newrobustcmd*{\glssymbol}{\@gls@hyp@opt\@glssymbol}
```

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

```
3815 \newcommand*{\@glssymbol}[2] [] {%  
3816   \new@ifnextchar [{\@glssymbol@{#1}{#2}}{\@glssymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3817 \def\@glssymbol@#1#2[#3] {%  
3818   \@gls@field@link{#1}{#2}{\glsentrysymbol{#2}#3}%  
3819 }
```


`\Glssymbol` behaves like `\glssymbol` except that the first letter is converted to uppercase.

`\Glssymbol`

```
3820 \newrobustcmd*{\Glssymbol}{\@gls@hyp@opt\@Glssymbol}
```

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

```
3821 \newcommand*{\@Glssymbol}[2] [] {%
```

```
3822   \new@ifnextchar[{\@Glssymbol@{#1}{#2}}{\@Glssymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3823 \def\@Glssymbol@#1#2[#3] {%
```

```
3824   \@gls@field@link{#1}{#2}{\glstentrysymbol{#2}#3}%
```

```
3825 }
```

`\GLSsymbol` behaves like `\glssymbol` except that the link text is converted to uppercase.

`\GLSsymbol`

```
3826 \newrobustcmd*{\GLSsymbol}{\@gls@hyp@opt\@GLSsymbol}
```

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

```
3827 \newcommand*{\@GLSsymbol}[2] [] {%
```

```
3828   \new@ifnextchar[{\@GLSsymbol@{#1}{#2}}{\@GLSsymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3829 \def\@GLSsymbol@#1#2[#3] {%
```

```
3830   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glstentrysymbol{#2}#3}%
```

```
3831 }
```

`\glssymbolplural` behaves like `\gls` except it always uses the value given by the symbol-plural key and it doesn't mark the entry as used.

`glssymbolplural`

```
3832 \newrobustcmd*{\glssymbolplural}{\@gls@hyp@opt\@glssymbolplural}
```

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

```
3833 \newcommand*{\@glssymbolplural}[2] [] {%
```

```
3834   \new@ifnextchar[{\@glssymbolplural@{#1}{#2}}{\@glssymbolplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3835 \def\@glssymbolplural@#1#2[#3] {%
```

```
3836   \@gls@field@link{#1}{#2}{\glstentrysymbolplural{#2}#3}%
```

```
3837 }
```

`\Glssymbolplural` behaves like `\glssymbolplural` except that the first letter is converted to uppercase.

`Glssymbolplural`

```
3838 \newrobustcmd*{\Glssymbolplural}{\@gls@hyp@opt\@Glssymbolplural}
```

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

```
3839 \newcommand*{\@Glssymbolplural}[2] [] {%
```

```
3840   \new@ifnextchar[{\@Glssymbolplural@{#1}{#2}}{\@Glssymbolplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3841 \def\@Glsymbolplural@#1#2[#3]{%
3842   \@gls@field@link{#1}{#2}{\Glsentrysymbolplural{#2}#3}%
3843 }
```

\Glsymbolplural behaves like \glsymbolplural except that the link text is converted to uppercase.

GLSsymbolplural

```
3844 \newrobustcmd*{\Glsymbolplural}{\@gls@hyp@opt\@Glsymbolplural}
```

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

```
3845 \newcommand*{\@Glsymbolplural}[2] [] {%
3846   \new@ifnextchar[{\@Glsymbolplural@{#1}{#2}}{\@Glsymbolplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3847 \def\@Glsymbolplural@#1#2[#3]{%
3848   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentrysymbolplural{#2}#3}}%
3849 }
```

\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

```
3850 \newrobustcmd*{\glsuseri}{\@gls@hyp@opt\@glsuseri}
```

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

```
3851 \newcommand*{\@glsuseri}[2] [] {%
3852   \new@ifnextchar[{\@glsuseri@{#1}{#2}}{\@glsuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3853 \def\@glsuseri@#1#2[#3]{%
3854   \@gls@field@link{#1}{#2}{\Glsentryuseri{#2}#3}%
3855 }
```

\Glsuseri behaves like \glsuseri except that the first letter is converted to uppercase.

\Glsuseri

```
3856 \newrobustcmd*{\Glsuseri}{\@gls@hyp@opt\@Glsuseri}
```

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

```
3857 \newcommand*{\@Glsuseri}[2] [] {%
3858   \new@ifnextchar[{\@Glsuseri@{#1}{#2}}{\@Glsuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3859 \def\@Glsuseri@#1#2[#3]{%
3860   \@gls@field@link{#1}{#2}{\Glsentryuseri{#2}#3}%
3861 }
```

\GLSuseri behaves like \glsuseri except that the link text is converted to uppercase.

\GLSuseri

```
3862 \newrobustcmd*{\GLSuseri}{\@gls@hyp@opt\@GLSuseri}
```

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

```
3863 \newcommand*{\@GLSuseri}[2] [] {%
3864   \new@ifnextchar[{\@GLSuseri@{#1}{#2}}{\@GLSuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3865 \def\@GLSuseri@#1#2[#3] {%
3866   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseri{#2}#3}}%
3867 }
```

\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

```
3868 \newrobustcmd*{\glsuserii}{\@gls@hyp@opt\@glsuserii}
```

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

```
3869 \newcommand*{\@glsuserii}[2] [] {%
3870   \new@ifnextchar[{\@glsuserii@{#1}{#2}}{\@glsuserii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3871 \def\@glsuserii@#1#2[#3] {%
3872   \@gls@field@link{#1}{#2}{\glsentryuserii{#2}#3}}%
3873 }
```

\Glsuserii behaves like \glsuserii except that the first letter is converted to uppercase.

\Glsuserii

```
3874 \newrobustcmd*{\Glsuserii}{\@gls@hyp@opt\@Glsuserii}
```

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

```
3875 \newcommand*{\@Glsuserii}[2] [] {%
3876   \new@ifnextchar[{\@Glsuserii@{#1}{#2}}{\@Glsuserii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3877 \def\@Glsuserii@#1#2[#3] {%
3878   \@gls@field@link{#1}{#2}{\Glsentryuserii{#2}#3}}%
3879 }
```

\GLSuserii behaves like \glsuserii except that the link text is converted to uppercase.

\GLSuserii

```
3880 \newrobustcmd*{\GLSuserii}{\@gls@hyp@opt\@GLSuserii}
```

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

```
3881 \newcommand*{\@GLSuserii}[2] [] {%
3882   \new@ifnextchar[{\@GLSuserii@{#1}{#2}}{\@GLSuserii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3883 \def\@GLSuserii@#1#2[#3] {%
3884   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuserii{#2}#3}}%
3885 }
```

\glsuseriii behaves like \gls except it always uses the value given by the user3 key and it doesn't mark the entry as used.

`\glsuseriii`

```
3886 \newrobustcmd*{\glsuseriii}{\@gls@hyp@opt\@glsuseriii}
```

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

```
3887 \newcommand*{\@glsuseriii}[2] [] {%
```

```
3888   \new@ifnextchar[{\@glsuseriii@{#1}{#2}}{\@glsuseriii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3889 \def\@glsuseriii@#1#2[#3] {%
```

```
3890   \@gls@field@link{#1}{#2}{\glsentryuseriii{#2}#3}%
```

```
3891 }
```

`\Glsuseriii` behaves like `\glsuseriii` except that the first letter is converted to upper-case.

`\Glsuseriii`

```
3892 \newrobustcmd*{\Glsuseriii}{\@gls@hyp@opt\@Glsuseriii}
```

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

```
3893 \newcommand*{\@Glsuseriii}[2] [] {%
```

```
3894   \new@ifnextchar[{\@Glsuseriii@{#1}{#2}}{\@Glsuseriii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3895 \def\@Glsuseriii@#1#2[#3] {%
```

```
3896   \@gls@field@link{#1}{#2}{\Glsentryuseriii{#2}#3}%
```

```
3897 }
```

`\GLSuseriii` behaves like `\glsuseriii` except that the link text is converted to uppercase.

`\GLSuseriii`

```
3898 \newrobustcmd*{\GLSuseriii}{\@gls@hyp@opt\@GLSuseriii}
```

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

```
3899 \newcommand*{\@GLSuseriii}[2] [] {%
```

```
3900   \new@ifnextchar[{\@GLSuseriii@{#1}{#2}}{\@GLSuseriii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3901 \def\@GLSuseriii@#1#2[#3] {%
```

```
3902   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseriii{#2}#3}}%
```

```
3903 }
```

`\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`

```
3904 \newrobustcmd*{\glsuseriv}{\@gls@hyp@opt\@glsuseriv}
```

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

```
3905 \newcommand*{\@glsuseriv}[2] [] {%
```

```
3906   \new@ifnextchar[{\@glsuseriv@{#1}{#2}}{\@glsuseriv@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3907 \def\@glsuseriv@#1#2[#3]{%
3908   \@gls@field@link{#1}{#2}{\glsentryuseriv{#2}#3}%
3909 }
```

\Glsuseriv behaves like \glsuseriv except that the first letter is converted to uppercase.

\Glsuseriv

```
3910 \newrobustcmd*{\Glsuseriv}{\@gls@hyp@opt\@Glsuseriv}
```

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

```
3911 \newcommand*{\@Glsuseriv}[2][\@Glsuseriv@{#1}{#2}]{\@Glsuseriv@{#1}{#2}[]}}
3912   \new@ifnextchar[{\@Glsuseriv@{#1}{#2}}{\@Glsuseriv@{#1}{#2}[]}]}
```

Read in the final optional argument:

```
3913 \def\@Glsuseriv@#1#2[#3]{%
3914   \@gls@field@link{#1}{#2}{\glsentryuseriv{#2}#3}%
3915 }
```

\GLSuseriv behaves like \glsuseriv except that the link text is converted to uppercase.

\GLSuseriv

```
3916 \newrobustcmd*{\GLSuseriv}{\@gls@hyp@opt\@GLSuseriv}
```

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

```
3917 \newcommand*{\@GLSuseriv}[2][\@GLSuseriv@{#1}{#2}]{\@GLSuseriv@{#1}{#2}[]}}
3918   \new@ifnextchar[{\@GLSuseriv@{#1}{#2}}{\@GLSuseriv@{#1}{#2}[]}]}
```

Read in the final optional argument:

```
3919 \def\@GLSuseriv@#1#2[#3]{%
3920   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseriv{#2}#3}}%
3921 }
```

\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

```
3922 \newrobustcmd*{\glsuserv}{\@gls@hyp@opt\@glsuserv}
```

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

```
3923 \newcommand*{\@glsuserv}[2][\@glsuserv@{#1}{#2}]{\@glsuserv@{#1}{#2}[]}}
3924   \new@ifnextchar[{\@glsuserv@{#1}{#2}}{\@glsuserv@{#1}{#2}[]}]}
```

Read in the final optional argument:

```
3925 \def\@glsuserv@#1#2[#3]{%
3926   \@gls@field@link{#1}{#2}{\glsentryuserv{#2}#3}%
3927 }
```

\Glsuserv behaves like \glsuserv except that the first letter is converted to uppercase.

\Glsuserv

```
3928 \newrobustcmd*{\Glsuserv}{\@gls@hyp@opt\@Glsuserv}
```

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

```
3929 \newcommand*{\@Glsuserv}[2] [] {%
3930 \new@ifnextchar [{\@Glsuserv@{#1}{#2}}{\@Glsuserv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3931 \def\@Glsuserv@#1#2[#3] {%
3932   \@gls@field@link{#1}{#2}{\Glsentryuserv{#2}#3}%
3933 }
```

\Glsuserv behaves like \glsuserv except that the link text is converted to uppercase.

\Glsuserv

```
3934 \newrobustcmd*{\Glsuserv}{\@gls@hyp@opt\@Glsuserv}
```

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

```
3935 \newcommand*{\@GLSuserv}[2] [] {%
3936 \new@ifnextchar [{\@GLSuserv@{#1}{#2}}{\@GLSuserv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3937 \def\@GLSuserv@#1#2[#3] {%
3938   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuserv{#2}#3}}%
3939 }
```

\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

```
3940 \newrobustcmd*{\glsuservi}{\@gls@hyp@opt\@glsuservi}
```

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

```
3941 \newcommand*{\@glsuservi}[2] [] {%
3942   \new@ifnextchar [{\@glsuservi@{#1}{#2}}{\@glsuservi@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3943 \def\@glsuservi@#1#2[#3] {%
3944   \@gls@field@link{#1}{#2}{\glsentryuservi{#2}#3}%
3945 }
```

\Glsuservi behaves like \glsuservi except that the first letter is converted to uppercase.

\Glsuservi

```
3946 \newrobustcmd*{\Glsuservi}{\@gls@hyp@opt\@Glsuservi}
```

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

```
3947 \newcommand*{\@Glsuservi}[2] [] {%
3948   \new@ifnextchar [{\@Glsuservi@{#1}{#2}}{\@Glsuservi@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3949 \def\@Glsuservi@#1#2[#3] {%
3950   \@gls@field@link{#1}{#2}{\Glsentryuservi{#2}#3}%
3951 }
```

\Glsuservi behaves like \glsuservi except that the link text is converted to uppercase.

\GLSuservi

```
3952 \newrobustcmd*{\GLSuservi}{\@gls@hyp@opt\@GLSuservi}
```

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

```
3953 \newcommand*{\@GLSuservi}[2] [] {%
```

```
3954   \new@ifnextchar[{\@GLSuservi@{#1}{#2}}{\@GLSuservi@{#1}{#2} []}]
```

Read in the final optional argument:

```
3955 \def\@GLSuservi@#1#2[#3] {%
```

```
3956   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuservi{#2}{#3}}}%
```

```
3957 }
```

Now deal with acronym related keys. First the short form:

\acrshort

```
3958 \newrobustcmd*{\acrshort}{\@gls@hyp@opt\@ns@acrshort}
```

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

```
3959 \newcommand*{\@ns@acrshort}[2] [] {%
```

```
3960   \new@ifnextchar[{\@acrshort{#1}{#2}}{\@acrshort{#1}{#2} []}]
```

```
3961 }
```

Read in the final optional argument:

```
3962 \def\@acrshort#1#2[#3] {%
```

```
3963   \glsdoifexists{#2}%
```

```
3964   {%
```

```
3965     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
```

```
3966     \let\glsifplural\@secondoftwo
```

```
3967     \let\gls@caps@case\@firstofthree
```

```
3968     \let\glsinsert\@empty
```

```
3969     \def\gls@customtext{%
```

```
3970       \acronymfont{\glsentryshort{#2}}#3%
```

```
3971     }%
```

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

```
3972   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
```

```
3973   }%
```

```
3974   \gls@postlinkhook
```

```
3975 }
```

\Acrshort

```
3976 \newrobustcmd*{\Acrshort}{\@gls@hyp@opt\@ns@Acrshort}
```

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

```
3977 \newcommand*{\@ns@Acrshort}[2] [] {%
```

```
3978   \new@ifnextchar[{\@Acrshort{#1}{#2}}{\@Acrshort{#1}{#2} []}]
```

```
3979 }
```

Read in the final optional argument:

```
3980 \def\@Acrshort#1#2[#3]{%
3981   \glsdoifexists{#2}%
3982   {%
3983     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
3984     \def\glslabel{#2}%
3985     \let\glsifplural\@secondoftwo
3986     \let\glscapscase\@secondofthree
3987     \let\glsinsert\@empty
3988     \def\glscustomtext{%
3989       \acronymfont{\Glsentryshort{#2}}#3%
3990     }%
3991     \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
3992   }%
3993   \glspostlinkhook
3994 }
```

Call \@gl@link Note that \@gl@link sets \glstype.

\ACRshort

```
3995 \newrobustcmd*{\ACRshort}{\@gl@hyp@opt\@ns@ACRshort}
```

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

```
3996 \newcommand*{\@ns@ACRshort}[2][ ]{%
3997   \new@ifnextchar[{\@ACRshort{#1}{#2}}{\@ACRshort{#1}{#2}[]}%
3998 }
```

Read in the final optional argument:

```
3999 \def\@ACRshort#1#2[#3]{%
4000   \glsdoifexists{#2}%
4001   {%
4002     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4003     \def\glslabel{#2}%
4004     \let\glsifplural\@secondoftwo
4005     \let\glscapscase\@thirdofthree
4006     \let\glsinsert\@empty
4007     \def\glscustomtext{%
4008       \mfirstucMakeUppercase{\acronymfont{\Glsentryshort{#2}}#3}%
4009     }%
4010     \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
4011   }%
4012   \glspostlinkhook
4013 }
```

Call \@gl@link Note that \@gl@link sets \glstype.

```
4010   \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
4011 }%
4012 \glspostlinkhook
4013 }
```


Short plural:

`\acrshortpl`

```
4014 \newrobustcmd*{\acrshortpl}{\@gls@hyp@opt\ns@acrshortpl}
```

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

```
4015 \newcommand*{\ns@acrshortpl}[2][\%  
4016   \new@ifnextchar[\@acrshortpl{#1}{#2}]{\@acrshortpl{#1}{#2}[]}%  
4017 }
```

Read in the final optional argument:

```
4018 \def\@acrshortpl#1#2[#3]{%  
4019   \glsdoifexists{#2}%  
4020   {%  
  
4021     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper  
  
4022     \def\glslabel{#2}%  
4023     \let\glsifplural\@firstoftwo  
4024     \let\glsapscase\@firstofthree  
4025     \let\glsinsert\@empty  
4026     \def\glscustomtext{%  
4027       \acronymfont{\glsentryshortpl{#2}}#3%  
4028     }%
```

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

```
4029   \@gls@link[#1]{#2}{\csname gls@\glsstyle @entryfmt\endcsname}%  
4030   }%  
  
4031   \glspostlinkhook  
4032 }
```

`\Acrshortpl`

```
4033 \newrobustcmd*{\Acrshortpl}{\@gls@hyp@opt\ns@Acrshortpl}
```

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

```
4034 \newcommand*{\ns@Acrshortpl}[2][\%  
4035   \new@ifnextchar[\@Acrshortpl{#1}{#2}]{\@Acrshortpl{#1}{#2}[]}%  
4036 }
```

Read in the final optional argument:

```
4037 \def\@Acrshortpl#1#2[#3]{%  
4038   \glsdoifexists{#2}%  
4039   {%  
  
4040     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper  
  
4041     \def\glslabel{#2}%  
4042     \let\glsifplural\@firstoftwo  
4043     \let\glsapscase\@secondofthree  
4044     \let\glsinsert\@empty
```

```

4045 \def\glscustomtext{%
4046 \acronymfont{\Glsentryshortpl{#2}}#3%
4047 }%

Call \@gls@link Note that \@gls@link sets \glstype.
4048 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4049 }%

4050 \glspostlinkhook
4051 }

```

\ACRshortpl

```

4052 \newrobustcmd*{\ACRshortpl}{\@gls@hyp@opt\ns@ACRshortpl}

Define the un-starred form. Need to determine if there is a final optional argument
4053 \newcommand*{\ns@ACRshortpl}[2][{}]{%
4054 \new@ifnextchar[{\@ACRshortpl{#1}{#2}}{\@ACRshortpl{#1}{#2}[]}%
4055 }

Read in the final optional argument:
4056 \def\@ACRshortpl#1#2[#3]{%
4057 \glsdoifexists{#2}%
4058 {%

4059 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

4060 \def\glslabel{#2}%
4061 \let\glsifplural\@firstoftwo
4062 \let\glsupcase\@thirdofthree
4063 \let\glsinsert\@empty
4064 \def\glscustomtext{%
4065 \mfirstucMakeUppercase{\acronymfont{\Glsentryshortpl{#2}}#3}%
4066 }%

Call \@gls@link Note that \@gls@link sets \glstype.
4067 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4068 }%

4069 \glspostlinkhook
4070 }

```

\acrlong

```

4071 \newrobustcmd*{\acrlong}{\@gls@hyp@opt\ns@acrlong}

Define the un-starred form. Need to determine if there is a final optional argument
4072 \newcommand*{\ns@acrlong}[2][{}]{%
4073 \new@ifnextchar[{\@acrlong{#1}{#2}}{\@acrlong{#1}{#2}[]}%
4074 }

```

Read in the final optional argument:

```
4075 \def\@acrlong#1#2[#3]{%
4076   \glsdoifexists{#2}%
4077   {%
4078     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4079     \def\glslabel{#2}%
4080     \let\glsifplural\@secondoftwo
4081     \let\glscapscase\@firstofthree
4082     \let\glsinsert\@empty
```

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

```
4083   \def\glscustomtext{%
4084     \glsentrylong{#2}#3%
4085   }%
```

Call \@gl@link Note that \@gl@link sets \glstype.

```
4086   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4087   }%
4088   \glspostlinkhook
4089 }
```

\Acrlong

```
4090 \newrobustcmd*{\Acrlong}{\@gl@hyp@opt\@ns@Acrlong}
```

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

```
4091 \newcommand*{\ns@Acrlong}[2][ ]{%
4092   \new@ifnextchar{\@Acrlong{#1}{#2}}{\@Acrlong{#1}{#2}[ ]}%
4093 }
```

Read in the final optional argument:

```
4094 \def\@Acrlong#1#2[#3]{%
4095   \glsdoifexists{#2}%
4096   {%
4097     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4098     \def\glslabel{#2}%
4099     \let\glsifplural\@secondoftwo
4100     \let\glscapscase\@secondofthree
4101     \let\glsinsert\@empty
```

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

```
4102   \def\glscustomtext{%
4103     \Glsentrylong{#2}#3%
4104   }%
```

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

```
4105   \@gls@link[#1]{#2}{\csname gls@\gls@type @entryfmt\endcsname}%  
4106   }%  
  
4107   \gls@postlinkhook  
4108 }
```

\ACRlong

```
4109 \newrobustcmd*{\ACRlong}{\@gls@hyp@opt\@ns@ACRlong}
```

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

```
4110 \newcommand*{\ns@ACRlong}[2][{}]{%  
4111   \new@ifnextchar[{\@ACRlong{#1}{#2}}{\@ACRlong{#1}{#2}[]}%  
4112 }
```

Read in the final optional argument:

```
4113 \def\@ACRlong#1#2[#3]{%  
4114   \gls@ifexists{#2}%  
4115   {%  
  
4116     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper  
  
4117     \def\gls@label{#2}%  
4118     \let\gls@ifplural\@secondoftwo  
4119     \let\gls@scapscase\@thirdofthree  
4120     \let\gls@insert\@empty
```

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

```
4121   \def\gls@customtext{%  
4122     \mfirstucMakeUppercase{\gls@entrylong{#2}{#3}}%  
4123   }%
```

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

```
4124   \@gls@link[#1]{#2}{\csname gls@\gls@type @entryfmt\endcsname}%  
4125   }%  
  
4126   \gls@postlinkhook  
4127 }
```

Short plural:

\acrlongpl

```
4128 \newrobustcmd*{\acrlongpl}{\@gls@hyp@opt\@ns@acrlongpl}
```

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

```
4129 \newcommand*{\ns@acrlongpl}[2][{}]{%  
4130   \new@ifnextchar[{\@acrlongpl{#1}{#2}}{\@acrlongpl{#1}{#2}[]}%  
4131 }
```

Read in the final optional argument:

```
4132 \def\@acrlongpl#1#2[#3]{%
4133   \glsdoifexists{#2}%
4134   {%
4135     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4136     \def\glslabel{#2}%
4137     \let\glsifplural\@firstoftwo
4138     \let\glscapscase\@firstofthree
4139     \let\glsinsert\@empty
```

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

```
4140   \def\glscustomtext{%
4141     \glstrylongpl{#2}#3%
4142   }%
```

Call \@gl@link. Note that \@gl@link sets \glstype.

```
4143   \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
4144   }%
4145   \glspostlinkhook
4146 }
```

\Acrlongpl

```
4147 \newrobustcmd*{\Acrlongpl}{\@gl@hyp@opt\@ns@Acrlongpl}
```

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

```
4148 \newcommand*{\ns@Acrlongpl}[2][\@gl@hyp@opt\@ns@Acrlongpl]{%
4149   \new@ifnextchar[\@Acrlongpl{#1}{#2}}{\@Acrlongpl{#1}{#2}[]}%
4150 }
```

Read in the final optional argument:

```
4151 \def\@Acrlongpl#1#2[#3]{%
4152   \glsdoifexists{#2}%
4153   {%
4154     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4155     \def\glslabel{#2}%
4156     \let\glsifplural\@firstoftwo
4157     \let\glscapscase\@secondofthree
4158     \let\glsinsert\@empty
```

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

```
4159   \def\glscustomtext{%
4160     \Glsentrylongpl{#2}#3%
4161   }%
```

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

```
4162 \gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4163 }%

4164 \glspostlinkhook
4165 }
```

`\ACRlongpl`

```
4166 \newrobustcmd*{\ACRlongpl}{\@gls@hyp@opt\@ns@ACRlongpl}
```

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

```
4167 \newcommand*{\ns@ACRlongpl}[2][{}]{%
4168 \new@ifnextchar[{\@ACRlongpl{#1}{#2}}{\@ACRlongpl{#1}{#2}[]}%
4169 }
```

Read in the final optional argument:

```
4170 \def\@ACRlongpl#1#2[#3]{%
4171 \glsdoifexists{#2}%
4172 {%
4173 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

4174 \def\glslabel{#2}%
4175 \let\glsifplural\@firstoftwo
4176 \let\gls caps case\@thirdofthree
4177 \let\glsinsert\@empty
```

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

```
4178 \def\glscustomtext{%
4179 \mfirstucMakeUppercase{\glsentrylongpl{#2}{#3}%
4180 }%
```

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

```
4181 \gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4182 }%

4183 \glspostlinkhook
4184 }
```

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>}`

```
4185 \newcommand*{\@gls@entry@field}[2]{%
```

```

4186 \csname glo@\glsdetoklabel{#1}@#2\endcsname
4187 }

```

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

```

4188 \newcommand*{\glsletentryfield}[3]{%
4189   \letcs{#1}{glo@\glsdetoklabel{#2}@#3}%
4190 }

```

`\Gls@entry@field` Generic first letter uppercase version.

`\@Gls@entry@field{<label>}{<field>}`

```

4191 \newcommand*{\@Gls@entry@field}[2]{%
4192   \glsdoifexistsordo{#1}%
4193   {%
4194     \letcs{@glo@text}{glo@\glsdetoklabel{#1}@#2}%
4195     \ifdef{@glo@text
4196       {%
4197         \xmakefirstuc{@glo@text}%
4198       }%
4199     }%
4200     ??\PackageError{glossaries}{The field ‘#2’ doesn’t exist for glossary
4201       entry ‘\glsdetoklabel{#1}’}{Check you have correctly spelt the entry
4202       label and the field name}%
4203   }%
4204 }%
4205 {%
4206   ??%
4207 }%
4208 }

```

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`

```

4209 \newcommand*{\glsentryname}[1]{\@gls@entry@field{#1}{name}}

```

`\Glsentryname`

```

4210 \newrobustcmd*{\Glsentryname}[1]{%
4211   \@Gls@entryname{#1}%
4212 }

```

`\@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:

```

4213 \newcommand*{\@Gls@entryname}[1]{%
4214   \@Gls@entry@field{#1}{name}%
4215 }

```

ls@acrentryname Now the behaviour when \setacronymstyle is used:

```

4216 \newcommand*{\@Gls@acrentryname}[1]{%
4217   \ifglshaslong{#1}%
4218   {%
4219     \letcs\@glo@text{glo@glsetoklabel{#1}@name}%
4220     \expandafter\@gls@getbody\@glo@text{}\@nil
4221     \expandafter\ifx\@gls@body\glentrylong\relax
4222     \expandafter\Glsentrylong\@gls@rest
4223   \else
4224     \expandafter\ifx\@gls@body\glentryshort\relax
4225     \expandafter\Glsentryshort\@gls@rest
4226   \else
4227     \expandafter\ifx\@gls@body\acronymfont\relax

```

Temporarily make \glentryshort behave like \Glsentryshort. (This is on the assumption that the argument of \acronymfont is \glentryshort{\label}), as that's the behaviour of the predefined acronym styles.) This is scoped to localise the effect of the assignment.

```

4228     {%
4229       \let\glentryshort\Glsentryshort
4230       \@glo@text
4231     }%
4232   \else
4233     \xmakefirstuc{\@glo@text}%
4234   \fi
4235 \fi
4236 \fi
4237 }%
4238 {%

```

Not an acronym

```

4239   \@Gls@entry@field{#1}{name}%
4240 }%
4241 }

```

Get the entry description (as specified by the description when the entry was defined). The argument is the label associated with the entry. Note that unless you used `description=false` in the `sanitize` package option you may get unexpected results if the description key contained any commands.

\glentrydesc

```

4242 \newcommand*{\glentrydesc}[1]{\@gls@entry@field{#1}{desc}}

```

\Glsentrydesc

```

4243 \newrobustcmd*{\Glsentrydesc}[1]{%
4244   \@Gls@entry@field{#1}{desc}%
4245 }

```


Plural form:

entrydescplural

```
4246 \newcommand*{\glentrydescplural}[1]{%
4247   \@gls@entry@field{#1}{descplural}%
4248 }
```

entrydescplural

```
4249 \newrobustcmd*{\Glsentrydescplural}[1]{%
4250   \@Gls@entry@field{#1}{descplural}%
4251 }
```

Get the entry text, as specified by the text key when the entry was defined. The argument is the label associated with the entry:

\glentrytext

```
4252 \newcommand*{\glentrytext}[1]{\@gls@entry@field{#1}{text}}
```

\Glsentrytext

```
4253 \newrobustcmd*{\Glsentrytext}[1]{%
4254   \@Gls@entry@field{#1}{text}%
4255 }
```

Get the plural form:

\glentryplural

```
4256 \newcommand*{\glentryplural}[1]{%
4257   \@gls@entry@field{#1}{plural}%
4258 }
```

\Glsentryplural

```
4259 \newrobustcmd*{\Glsentryplural}[1]{%
4260   \@Gls@entry@field{#1}{plural}%
4261 }
```

Get the symbol associated with this entry. The argument is the label associated with the entry.

\glentrysymbol

```
4262 \newcommand*{\glentrysymbol}[1]{%
4263   \@gls@entry@field{#1}{symbol}%
4264 }
```

\Glsentrysymbol

```
4265 \newrobustcmd*{\Glsentrysymbol}[1]{%
4266   \@Gls@entry@field{#1}{symbol}%
4267 }
```

Plural form:

trysymbolplural

```
4268 \newcommand*{\glsentrysymbolplural}[1]{%
4269   \@gls@entry@field{#1}{symbolplural}%
4270 }
```

trysymbolplural

```
4271 \newrobustcmd*{\Glsentrysymbolplural}[1]{%
4272   \@Gls@entry@field{#1}{symbolplural}%
4273 }
```

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

```
4274 \newcommand*{\glsentryfirst}[1]{%
4275   \@gls@entry@field{#1}{first}%
4276 }
```

\Glsentryfirst

```
4277 \newrobustcmd*{\Glsentryfirst}[1]{%
4278   \@Gls@entry@field{#1}{first}%
4279 }
```

Get the plural form (as specified by the firstplural key when the entry was defined).

ntryfirstplural

```
4280 \newcommand*{\glsentryfirstplural}[1]{%
4281   \@gls@entry@field{#1}{firstpl}%
4282 }
```

ntryfirstplural

```
4283 \newrobustcmd*{\Glsentryfirstplural}[1]{%
4284   \@Gls@entry@field{#1}{firstpl}%
4285 }
```

sentrytitlecase

```
4286 \newrobustcmd*{@glsentrytitlecase}[2]{%
4287   \glsfieldfetch{#1}{#2}{\@gls@value}%
4288   \xcapitalisewords{\@gls@value}%
4289 }
4290 \ifdef\texorpdfstring
4291 {
4292   \newcommand*{\glsentrytitlecase}[2]{%
4293     \texorpdfstring
4294       {\@glsentrytitlecase{#1}{#2}}%
4295       {\@gls@entry@field{#1}{#2}}%
4296   }
4297 }
4298 {
```

```

4299 \newcommand*{\glsentrytitlecase}[2]{\@glsentrytitlecase{#1}{#2}}
4300 }

```

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

`\glsentrytype`

```

4301 \newcommand*{\glsentrytype}[1]{\@gls@entry@field{#1}{type}}

```

Display the sort text used for this entry. Note that the sort key is sanitized, so unexpected results may occur if the sort key contained commands.

`\glsentrysort`

```

4302 \newcommand*{\glsentrysort}[1]{%
4303   \@gls@entry@field{#1}{sort}%
4304 }

```

`\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.

```

4305 \newcommand*{\glsentryuseri}[1]{%
4306   \@gls@entry@field{#1}{useri}%
4307 }

```

`\Glsentryuseri`

```

4308 \newrobustcmd*{\Glsentryuseri}[1]{%
4309   \@Gls@entry@field{#1}{useri}%
4310 }

```

`\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.

```

4311 \newcommand*{\glsentryuserii}[1]{%
4312   \@gls@entry@field{#1}{userii}%
4313 }

```

`\Glsentryuserii`

```

4314 \newrobustcmd*{\Glsentryuserii}[1]{%
4315   \@Gls@entry@field{#1}{userii}%
4316 }

```

`\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.

```

4317 \newcommand*{\glsentryuseriii}[1]{%
4318   \@gls@entry@field{#1}{useriii}%
4319 }

```

`\Glsentryuseriii`

```

4320 \newrobustcmd*{\Glsentryuseriii}[1]{%
4321   \@Gls@entry@field{#1}{useriii}%
4322 }

```

`\glentryuseriv` Get the fourth user key (as specified by the user4 when the entry was defined). The argument is the label associated with the entry.

```
4323 \newcommand*{\glentryuseriv}[1]{%
4324   \@gls@entry@field{#1}{useriv}%
4325 }
```

`\Glsentryuseriv`

```
4326 \newrobustcmd*{\Glsentryuseriv}[1]{%
4327   \@Gls@entry@field{#1}{useriv}%
4328 }
```

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

```
4329 \newcommand*{\glentryuseriv}[1]{%
4330   \@gls@entry@field{#1}{useriv}%
4331 }
```

`\Glsentryuseriv`

```
4332 \newrobustcmd*{\Glsentryuseriv}[1]{%
4333   \@Gls@entry@field{#1}{useriv}%
4334 }
```

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

```
4335 \newcommand*{\glentryuseriv}[1]{%
4336   \@gls@entry@field{#1}{useriv}%
4337 }
```

`\Glsentryuseriv`

```
4338 \newrobustcmd*{\Glsentryuseriv}[1]{%
4339   \@Gls@entry@field{#1}{useriv}%
4340 }
```

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

```
4341 \newcommand*{\glentryshort}[1]{\@gls@entry@field{#1}{short}}
```

`\Glsentryshort`

```
4342 \newrobustcmd*{\Glsentryshort}[1]{%
4343   \@Gls@entry@field{#1}{short}%
4344 }
```

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

```
4345 \newcommand*{\glentryshortpl}[1]{\@gls@entry@field{#1}{shortpl}}
```

Glsentryshortpl

```
4346 \newrobustcmd*{\Glsentryshortpl}[1]{%
4347   \@Gls@entry@field{#1}{shortpl}%
4348 }
```

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

```
4349 \newcommand*{\glsentrylong}[1]{\@Gls@entry@field{#1}{long}}
```

`\Glsentrylong`

```
4350 \newrobustcmd*{\Glsentrylong}[1]{%
4351   \@Gls@entry@field{#1}{long}%
4352 }
```

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

```
4353 \newcommand*{\glsentrylongpl}[1]{\@Gls@entry@field{#1}{longpl}}
```

`\Glsentrylongpl`

```
4354 \newrobustcmd*{\Glsentrylongpl}[1]{%
4355   \@Gls@entry@field{#1}{longpl}%
4356 }
```

Short cut macros to access full form:

`\glsentryfull`

```
4357 \newcommand*{\glsentryfull}[1]{%
4358   \acrfullformat{\glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%
4359 }
```

`\Glsentryfull`

```
4360 \newrobustcmd*{\Glsentryfull}[1]{%
4361   \acrfullformat{\Glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%
4362 }
```

`\glsentryfullpl`

```
4363 \newcommand*{\glsentryfullpl}[1]{%
4364   \acrfullformat{\glsentrylongpl{#1}}{\acronymfont{\glsentryshortpl{#1}}}%
4365 }
```

`\Glsentryfullpl`

```
4366 \newrobustcmd*{\Glsentryfullpl}[1]{%
4367   \acrfullformat{\Glsentrylongpl{#1}}{\acronymfont{\glsentryshortpl{#1}}}%
4368 }
```

entrynumberlist Displays the number list as is.

```
4369 \newcommand*\glentrynumberlist[1]{%
4370   \glsdoifexists{#1}%
4371   {%
4372     \@gls@entry@field{#1}{numberlist}%
4373   }%
4374 }
```

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

```
4375 \@ifpackageloaded{hyperref} {%
4376   \newcommand*\glsdisplaynumberlist[1]{%
4377     \GlossariesWarning
4378     {%
4379       \string\glsdisplaynumberlist\space
4380       doesn't work with hyperref.^^JUsing
4381       \string\glentrynumberlist\space instead%
4382     }%
4383     \glentrynumberlist{#1}%
4384   }%
4385 }%
4386 {%
4387   \newcommand*\glsdisplaynumberlist[1]{%
4388     \glsdoifexists{#1}%
4389     {%
4390       \bgroup

4391         \edef\@glo@label{\glsdetoklabel{#1}}%
4392         \let\@org@glsnumberformat\glsnumberformat
4393         \def\glsnumberformat##1{##1}%
4394         \protected@edef\the@numberlist{%
4395           \csname glo@\@glo@label @numberlist\endcsname}%
4396         \def\@gls@numlist@sep{%
4397           \def\@gls@numlist@nextsep{%
4398             \def\@gls@numlist@lastsep{%
4399               \def\@gls@thislist{%
4400                 \def\@gls@donext@def{%
4401                   \renewcommand\do[1]{%
4402                     \protected@edef\@gls@thislist{%
4403                       \@gls@thislist
4404                       \noexpand\@gls@numlist@sep
4405                       ##1%
4406                     }%
4407                     \let\@gls@numlist@sep\@gls@numlist@nextsep
4408                     \def\@gls@numlist@nextsep{\glsnumlistsep}%
4409                     \@gls@donext@def
4410                     \def\@gls@donext@def{%
4411                       \def\@gls@numlist@lastsep{\glsnumlistlastsep}%
4412                     }%
4413                   }%

```

```

4414         \expandafter \glsnumlistparser \expandafter{\the@numberlist}%
4415         \let\@gls@numlist@sep\@gls@numlist@lastsep
4416         \@gls@thislist
4417     \egroup
4418 }%
4419 }
4420 }

```

`\glsnumlistsep`

```

4421 \newcommand*{\glsnumlistsep}{, }

```

`\glsnumlistlastsep`

```

4422 \newcommand*{\glsnumlistlastsep}{ \& }

```

`\gls hyperlink`

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

```

4423 \newcommand*{\gls hyperlink}[2][\gls entrytext{\@glo@label}]{%
4424   \def\@glo@label{#2}%
4425   \@gls link{\glo link prefix\gls detok label{#2}}{#1}}

```

1.12 Adding an entry to the glossary without generating text

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

```

4426 \define@key{gloss add}{counter}{\def\@gls@counter{#1}}
4427 \define@key{gloss add}{format}{\def\@gls number format{#1}}

```

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

```

4428 \define@key{gloss add}{types}{\def\@glo@type{#1}}

```

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

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

`\gls add`

```

4429 \newrobustcmd*{\gls add}[2][ ]{%

```

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

```

4430   \@gls@adjustmode
4431   \gls do if exists{#2}%
4432   {%
4433     \def\@gls number format{gls number format}%

```

```

4434 \edef\@gls@counter{\csname glo@\glsdetoklabel{#2}@counter\endcsname}%
4435 \setkeys{glossadd}{#1}%

Store the entry's counter in \theglsentrycounter
4436 \@gls@saveentrycounter

This should use \@do@wrglossary rather than \do@wrglossary since the whole point of
\glsadd is to add a line to the glossary.
4437 \@do@wrglossary{#2}%
4438 }%
4439 }

```

@gls@adjustmode

```

4440 \newcommand*{\@gls@adjustmode}{}
4441 \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

```

4442 \newrobustcmd*{\glsaddall}[1][]{%
4443 \edef\@glo@type{\@glo@types}%
4444 \setkeys{glossadd}{#1}%
4445 \forallglsentries[\@glo@type]{\@glo@entry}{%
4446 \glsadd[#1]{\@glo@entry}%
4447 }%
4448 }

```

\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.

```

4449 \newrobustcmd*{\glsaddallunused}[1][\@glo@types]{%
4450 \forallglsentries[#1]{\@glo@entry}%
4451 {%
4452 \ifglsused{\@glo@entry}{\glsadd[format=glsignore]{\@glo@entry}}%
4453 }%
4454 }

```

\glsignore

```

4455 \newcommand*{\glsignore}[1]{}

```


1.13 Creating associated files

The `\writeist` command creates the associated customized `.ist` `makeindex` style file. While defining this command, some characters have their catcodes temporarily changed to ensure they get written to the `.ist` file correctly. The `makeindex` actual character (usually `@`) is redefined to be a `?`, to allow internal commands to be written to the glossary file output file.

The special characters are stored in `\@gls@actualchar`, `\@gls@encapchar`, `\@gls@levelchar` and `\@gls@quotechar` to make them easier to use later, but don't change these values, because the characters are encoded in the command definitions that are used to escape the special characters (which means that the user no longer needs to worry about `makeindex` special characters).

The symbols and numbers label for group headings are hardwired into the `.ist` file as `glssymbols` and `glsnumbers`, the group titles can be translated (so that `\glssymbolsgroupname` replaces `glssymbols` and `\glsnumbersgroupname` replaces `glsnumbers`) using the command `\glsgroupname` which is defined in `.` This is done to prevent any problem characters in `\glssymbolsgroupname` and `\glsnumbersgroupname` from breaking hyperlinks.

```
\glsopenbrace  Define \glsopenbrace to make it easier to write an opening brace to a file.
4456 \edef\glsopenbrace{\expandafter\@gobble\string\{}}

\glsclosebrace Define \glsclosebrace to make it easier to write an opening brace to a file.
4457 \edef\glsclosebrace{\expandafter\@gobble\string\}}

\glsbackslash  Define \glsbackslash to make it easier to write a backslash to a file.
4458 \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.
4459 \edef\glsquote#1{\string"#1\string"}

\glspcentchar  Define \glspcentchar to make it easier to write a percent character to a file.
4460 \edef\glspcentchar{\expandafter\@gobble\string\%}

\glstildechar  Define \glstildechar to make it easier to write a tilde character to a file.
4461 \edef\glstildechar{\string~}

\@glsfirstletter Define the first letter to come after the digits 0,...,9. Only required for xindy.
4462 \ifglsxindy
4463   \newcommand*{\@glsfirstletter}{A}
4464 \fi

\@glsfirstletterAfterDigits Sets the first letter to come after the digits 0,...,9.
4465 \ifglsxindy
4466   \newcommand*{\GlsSetXdyFirstLetterAfterDigits}[1]{%
4467     \renewcommand*{\@glsfirstletter}{#1}}
4468 \else
```

```

4469 \newcommand*{\GlsSetXdyFirstLetterAfterDigits}[1]{%
4470 \glsnoxywarning\GlsSetXdyFirstLetterAfterDigits}
4471 \fi

```

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

```

4472 \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.

```

4473 \ifglsxindy
4474 \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
4475 \renewcommand*{\@glsminrange}{#1}}
4476 \else
4477 \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
4478 \glsnoxywarning\GlsSetXdyMinRangeLength}
4479 \fi

```

`\writeist`

```

4480 \ifglsxindy

```

Code to use if xindy is required.

```

4481 \def\writeist{%

```

Define write register if not already defined

```

4482 \ifundef{\glswrite}{\newwrite\glswrite}{}%

```

Update attributes list

```

4483 \@gls@addpredefinedattributes

```

Open the file.

```

4484 \openout\glswrite=\istfilename

```

Write header comment at the start of the file

```

4485 \write\glswrite{;; xindy style file created by the glossaries
4486 package}%
4487 \write\glswrite{;; for document '\jobname' on
4488 \the\year-\the\month-\the\day}%

```

Specify the required styles

```

4489 \write\glswrite{^^J; required styles^^J}
4490 \@for\@xdystyle:=\@xdyrequiredstyles\do{%
4491 \ifx\@xdystyle\@empty
4492 \else
4493 \protected@write\glswrite{{(require
4494 \string"\@xdystyle.xdy\string")}}%
4495 \fi
4496 }%

```

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

```

4497 \write\glswrite{^^J%
4498 ; list of allowed attributes (number formats)^^J}%
4499 \write\glswrite{(define-attributes ((\@xdyattributes)))}%

```

Define any additional alphabets

```
4500 \write\glswrite{^^J; user defined alphabets^^J}%
4501 \write\glswrite{\@xdyuseralphabets}%
```

Define location classes.

```
4502 \write\glswrite{^^J; location class definitions^^J}%
```

As from version 3.0, locations are now specified as $\{\langle Hprefix \rangle\}\{\langle number \rangle\}$, so need to add all possible combinations of location types.

```
4503 \@for\@gls@classI:=\@gls@xdy@locationlist\do{%
```

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

```
4504 \protected@write\glswrite{}\{(define-location-class
4505 \string"\@gls@classI\string"^^J\space\space\space
4506 (
4507 :sep "{"
4508 \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
4509 :sep "}"
4510 )
4511 ^^J\space\space\space
4512 :min-range-length \@glsminrange^^J%
4513 )
4514 }%
```

Nested iteration over all classes:

```
4515 {%
4516 \@for\@gls@classII:=\@gls@xdy@locationlist\do{%
4517 \protected@write\glswrite{}\{(define-location-class
4518 \string"\@gls@classII-\@gls@classI\string"
4519 ^^J\space\space\space
4520 (
4521 :sep "{"
4522 \csname @gls@xdy@Lclass@\@gls@classII\endcsname\space
4523 :sep "}"
4524 \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
4525 :sep "}"
4526 )
4527 ^^J\space\space\space
4528 :min-range-length \@glsminrange^^J%
4529 )
4530 }%
4531 }%
4532 }%
4533 }%
```

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

```
4534 \write\glswrite{^^J; user defined location classes}%
4535 \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.)

```

4536 \write\glswrite{^^J; define cross-reference class^^J}%
4537 \write\glswrite{(define-crossref-class \string"see\string"
4538 :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.)

```

4539 \write\glswrite{(markup-crossref-list
4540 :class \string"see\string"^^J\space\space\space
4541 :open \string"\string\glsseeformat\string"
4542 :close \string"{\string")}%

```

List the order to sort the classes.

```

4543 \write\glswrite{^^J; define the order of the location classes}%
4544 \write\glswrite{(define-location-class-order
4545 (\@xdylocationclassorder))}%

```

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

```

4546 \write\glswrite{^^J; define the glossary markup^^J}%

4547 \write\glswrite{(markup-index^^J\space\space\space
4548 :open \string"\string
4549 \glossarysection[\string\glossarytoctitle]{\string
4550 \glossarytitle}\string\glossarypreamble}%

```

Add all the xindy-only macro definitions (needed to prevent errors in the event that the user changes from xindy to makeindex)

```

4551 \@for\@this@ctr:=\@xdycounters\do{%
4552   {%
4553     \@for\@this@attr:=\@xdyattributelist\do{%
4554       \protected@write\glswrite{}{\string\providecommand*%
4555         \expandafter\string
4556         \csname glsX\@this@ctr X\@this@attr\endcsname[2]%
4557       {%
4558         \string\setentrycounter
4559         [\expandafter\@gobble\string\#1]{\@this@ctr}%
4560         \expandafter\string
4561         \csname\@this@attr\endcsname
4562         {\expandafter\@gobble\string\#2}%
4563       }%
4564     }%
4565   }%
4566 }%
4567 }%

```

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

```

4568 \write\glswrite{%
4569 \string\begin
4570 {theglossary}\string\glossaryheader\glstildechar n\string" ^^J\space
4571 \space\space:close \string"\glspercentchar\glstildechar n\string

```

```

4572         \end{theglossary}\string\glossarypostamble
4573         \glstildechar n\string" ^^J\space\space\space
4574         :tree)}}%

```

Specify what to put between letter groups

```

4575     \write\glswrite{(markup-letter-group-list
4576         :sep \string"\string\glsgroupskip\glstildechar n\string"}}%

```

Specify what to put between entries

```

4577     \write\glswrite{(markup-indexentry
4578         :open \string"\string\relax \string\glresetentrylist
4579         \glstildechar n\string"}}%

```

Specify how to format entries

```

4580     \write\glswrite{(markup-locclass-list :open
4581         \string"\glsoopenbrace\string\glossaryentrynumbers
4582         \glsoopenbrace\string\relax\space \string"^^J\space\space\space
4583         :sep \string", \string"
4584         :close \string"\glsclosebrace\glsclosebrace\string"}}%

```

Specify how to separate location numbers

```

4585     \write\glswrite{(markup-locref-list
4586         :sep \string"\string\delimN\space\string"}}%

```

Specify how to indicate location ranges

```

4587     \write\glswrite{(markup-range
4588         :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.

```

4589     \@onelevel@sanitize\gls@suffixF
4590     \@onelevel@sanitize\gls@suffixFF
4591     \ifx\gls@suffixF\@empty
4592     \else
4593         \write\glswrite{(markup-range
4594             :close "\gls@suffixF" :length 1 :ignore-end)}}%
4595     \fi
4596     \ifx\gls@suffixFF\@empty
4597     \else
4598         \write\glswrite{(markup-range
4599             :close "\gls@suffixFF" :length 2 :ignore-end)}}%
4600     \fi

```

Specify how to format locations.

```

4601     \write\glswrite{^^J; define format to use for locations^^J}%
4602     \write\glswrite{@xdylocref}%

```

Specify how to separate letter groups.

```

4603     \write\glswrite{^^J; define letter group list format^^J}%
4604     \write\glswrite{(markup-letter-group-list
4605         :sep \string"\string\glsgroupskip\glstildechar n\string"}}%

```

Define letter group headings.

```
4606 \write\glswrite{^^J; letter group headings^^J}%
4607 \write\glswrite{(markup-letter-group
4608 :open-head \string\string\glsgroupheading
4609 \glsoopenbrace\string^^J\space\space\space
4610 :close-head \string\glsclosebrace\string)}}%
```

Define additional letter groups.

```
4611 \write\glswrite{^^J; additional letter groups^^J}%
4612 \write\glswrite{\@xdylettergroups}%
```

Define additional sort rules

```
4613 \write\glswrite{^^J; additional sort rules^^J}
4614 \write\glswrite{\@xdysortrules}%
```

Close the style file

```
4615 \closeout\glswrite
```

Suppress any further calls.

```
4616 \let\writeist\relax
4617 }
4618 \else
```

Code to use if makeindex is required.

```
4619 \edef\@gls@actualchar{\string?}
4620 \edef\@gls@encapchar{\string|}
4621 \edef\@gls@levelchar{\string!}
4622 \edef\@gls@quotechar{\string"}
4623 \def\writeist{\relax
4624 \ifundef{\glswrite}{\newwrite\glswrite}{\relax
4625 \openout\glswrite=\istfilename
4626 \write\glswrite{\glspersentchar\space makeindex style file
4627 created by the glossaries package}
4628 \write\glswrite{\glspersentchar\space for document
4629 '\jobname' on \the\year-\the\month-\the\day}
4630 \write\glswrite{actual '\@gls@actualchar'}
4631 \write\glswrite{encap '\@gls@encapchar'}
4632 \write\glswrite{level '\@gls@levelchar'}
4633 \write\glswrite{quote '\@gls@quotechar'}
4634 \write\glswrite{keyword \string\string\glossaryentry\string"}
4635 \write\glswrite{preamble \string\string\glossarysection[\string
4636 \glossarytoctitle]{\string\glossarytitle}\string
4637 \glossarypreamble\string\n\string\begin{theglossary}\string
4638 \glossaryheader\string\n\string"}
4639 \write\glswrite{postamble \string\string%\string\n\string
4640 \end{theglossary}\string\glossarypostamble\string\n
4641 \string"}
4642 \write\glswrite{group_skip \string\string\glsgroupskip\string\n
4643 \string"}
4644 \write\glswrite{item_0 \string\string%\string\n\string"}
4645 \write\glswrite{item_1 \string\string%\string\n\string"}
```

```

4646 \write\glswrite{item_2 \string\string%\string\n\string"}
4647 \write\glswrite{item_01 \string\string%\string\n\string"}
4648 \write\glswrite{item_x1
4649 \string\string\relax \string\glresetentrylist\string\n
4650 \string"}
4651 \write\glswrite{item_12 \string\string%\string\n\string"}
4652 \write\glswrite{item_x2
4653 \string\string\relax \string\glresetentrylist\string\n
4654 \string"}

4655 \write\glswrite{delim_0 \string\string\{\string
4656 \glossaryentrynumbers\string\{\string\relax \string"}
4657 \write\glswrite{delim_1 \string\string\{\string
4658 \glossaryentrynumbers\string\{\string\relax \string"}
4659 \write\glswrite{delim_2 \string\string\{\string
4660 \glossaryentrynumbers\string\{\string\relax \string"}
4661 \write\glswrite{delim_t \string\string\}\string\}\string"}
4662 \write\glswrite{delim_n \string\string\delimN \string"}
4663 \write\glswrite{delim_r \string\string\delimR \string"}
4664 \write\glswrite{headings_flag 1}
4665 \write\glswrite{heading_prefix
4666 \string\string\glsgroupheading\string\{\string"}
4667 \write\glswrite{heading_suffix
4668 \string\string\}\string\relax
4669 \string\glresetentrylist \string"}
4670 \write\glswrite{symhead_positive \string\glssymbols\string"}
4671 \write\glswrite{numhead_positive \string\glnumbers\string"}
4672 \write\glswrite{page_compositor \string\glscpositor\string"}
4673 \@glscbsdq\glscsuffixF
4674 \@glscbsdq\glscsuffixFF
4675 \ifx\glscsuffixF\@empty
4676 \else
4677 \write\glswrite{suffix_2p \string\glscsuffixF\string"}
4678 \fi
4679 \ifx\glscsuffixFF\@empty
4680 \else
4681 \write\glswrite{suffix_3p \string\glscsuffixFF\string"}
4682 \fi
4683 \closeout\glswrite
4684 \let\writeist\relax
4685 }
4686 \fi

```

The command `\noist` will suppress the creation of the `.ist` file. Obviously you need to use this command before `\writeist` to have any effect.

`\noist`

```
4687 \newcommand{\noist}{%
```

Update attributes list

```

4688 \@gls@addpredefinedattributes
4689 \let\writeist\relax
4690 }

```

\@makeglossary is an internal command that takes an argument indicating the glossary type. This command will create the glossary file required by makeindex for the given glossary type, using the extension supplied by the *<out-ext>* parameter used in \newglossary (and it will also activate the \glossary command, and create the customized .ist makeindex style file).

Note that you can't use \@makeglossary for only some of the defined glossaries. You either need to have a \makeglossary for all glossaries or none (otherwise you will end up with a situation where T_EX is trying to write to a non-existent file). The relevant glossary must be defined prior to using \@makeglossary.

\@makeglossary

```

4691 \newcommand*{\@makeglossary}[1]{%
4692   \ifglossaryexists{#1}%
4693   {%

```

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

```

4694     \ifglssavewrites
4695       \expandafter\newtoks\csname glo@#1@filetok\endcsname
4696     \else
4697       \expandafter\newwrite\csname glo@#1@file\endcsname
4698       \expandafter\@glsoopenfile\csname glo@#1@file\endcsname{#1}%
4699     \fi
4700     \@gls@renewglossary
4701     \writeist
4702   }%
4703   {%
4704     \PackageError{glossaries}%
4705     {Glossary type ‘#1’ not defined}%
4706     {New glossaries must be defined before using \string\makeglossary}%
4707   }%
4708 }

```

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

```

4709 \newcommand*{\@glsoopenfile}[2]{%
4710   \immediate\openout#1=\jobname.\csname @glotype@#2@out\endcsname
4711   \PackageInfo{glossaries}{Writing glossary file
4712     \jobname.\csname @glotype@#2@out\endcsname}%
4713 }

```

\@closegls

```

4714 \newcommand*{\@closegls}[1]{%
4715   \closeout\csname glo@#1@file\endcsname
4716 }
4717 %   \end{macrocode}

```



```

4718%\end{macro}
4719%
4720%\begin{macro}{\@gls@automake}
4721%\changes{4.08}{2014-07-30}{new}
4722%   \begin{macrocode}
4723\ifglxsindy
4724 \newcommand*{\@gls@automake}[1]{%
4725   \ifglossaryexists{#1}
4726   {%
4727     \@closegls{#1}%
4728     \ifdefstring{\glsorder}{letter}%
4729     {\def\@gls@order{-M ord/letorder }}%
4730     {\let\@gls@order\@empty}%
4731     \ifcsundef{xdy@#1@language}%
4732     {\let\@gls@langmod\@xdy@main@language}%
4733     {\letcs\@gls@langmod{xdy@#1@language}}%
4734     \edef\@gls@dothiswrite{\noexpand\write18{xindy
4735       -I xindy
4736       \@gls@order
4737       -L \@gls@langmod\space
4738       -M \@gls@istfilebase\space
4739       -C \@gls@codepage\space
4740       -t \jobname.\csuse{@glotype@#1@log}
4741       -o \jobname.\csuse{@glotype@#1@in}
4742       \jobname.\csuse{@glotype@#1@out}}}%
4743     }%
4744     \@gls@dothiswrite
4745   }%
4746   {%
4747     \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
4748   }%
4749 }
4750\else
4751 \newcommand*{\@gls@automake}[1]{%
4752   \ifglossaryexists{#1}
4753   {%
4754     \@closegls{#1}%
4755     \ifdefstring{\glsorder}{letter}%
4756     {\def\@gls@order{-l }}%
4757     {\let\@gls@order\@empty}%
4758     \edef\@gls@dothiswrite{\noexpand\write18{makeindex \@gls@order
4759       -s \istfilename\space
4760       -t \jobname.\csuse{@glotype@#1@log}
4761       -o \jobname.\csuse{@glotype@#1@in}
4762       \jobname.\csuse{@glotype@#1@out}}}%
4763     }%
4764     \@gls@dothiswrite
4765   }%
4766   {%

```

```

4767     \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
4768 }%
4769 }
4770 \fi

```

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

```

4771 \newcommand*{\@warn@nomakeglossaries}{}

```

Only use this if warning if `\printglossary` has been used without `\makeglossaries`

```

4772 \newcommand*{\@warn@nomakeglossaries}{\@warn@nomakeglossaries}

```

`\makeglossaries` will use `\makeglossary` for each glossary type that has been defined. New glossaries need to be defined before using `\makeglossary`, so have `\makeglossaries` redefine `\newglossary` to prevent it being used afterwards.

`\makeglossaries`

```

4773 \newcommand*{\makeglossaries}{%

```

Define the write used for style file also used for all other output files if `savewrites=true`.

```

4774 \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.

```

4775 \protected@write\@auxout{}{\string\providecommand\string\@glsorder[1]{}

```

```

4776 \protected@write\@auxout{}{\string\providecommand\string\@istfilename[1]{}

```

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

```

4777 \protected@write\@auxout{}{\string\@istfilename{\istfilename}}%

```

```

4778 \protected@write\@auxout{}{\string\@glsorder{\glsorder}}

```

Iterate through each glossary type and activate it.

```

4779 \@for\@glo@type:=\@glo@types\do{%

```

```

4780 \ifthenelse{\equal{\@glo@type}{}}{}{%

```

```

4781 \makeglossary{\@glo@type}}%

```

```

4782 }%

```

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

```

4783 \renewcommand*\newglossary[4][]{%

```

```

4784 \PackageError{glossaries}{New glossaries

```

```

4785 must be created before \string\makeglossaries}{You need

```

```

4786 to move \string\makeglossaries\space after all your

```

```

4787 \string\newglossary\space commands}}%

```

Any subsequent instances of this command should have no effect

```

4788 \let\makeglossary\relax

```

```

4789 \let\makeglossary\relax

```

```

4790 \let\makeglossaries\relax

```

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

```

4791 \@disable@onlypremakeg

```

Allow see key:

```

4792 \let\gls@checkseeallowed\relax

```

Suppress warning about no \makeglossaries

```
4793 \let\warn@nomakeglossaries\relax
```

Activate warning about missing \printglossary

```
4794 \def\warn@noprintglossary{%
4795   \GlossariesWarningNoLine{No \string\printglossary\space
4796     or \string\printglossaries\space
4797     found.^^J(Remove \string\makeglossaries\space if you don't want
4798     any glossaries.)^^JThis document will not have a glossary}%
4799 }%
```

Declare list parser for \glsdisplaynumberlist

```
4800 \ifglssavenumberlist
4801   \edef\@gls@dodeflistparser{\noexpand\DeclareListParser
4802     {\noexpand\glsnumlistparser}{\delimN}}}%
4803   \@gls@dodeflistparser
4804 \fi
```

Prevent user from also using \makenoidxglossaries

```
4805 \let\makenoidxglossaries\@no@makeglossaries
```

Prohibit sort key in printgloss family:

```
4806 \renewcommand*{\@printgloss@setsort}{%
4807   \let\@glo@assign@sortkey\@glo@no@assign@sortkey
4808 }%
```

Check the automake setting:

```
4809 \ifglssautomake
4810   \renewcommand*{\@gls@doautomake}{%
4811     \@for\@gls@type:=\@glo@types\do{%
4812       \ifdefempty{\@gls@type}{}%
4813       {\@gls@automake{\@gls@type}}%
4814     }%
4815   }%
4816 \fi
4817 }
```

Must occur in the preamble:

```
4818 \@onlypreamble{\makeglossaries}
```

\glswrite The definition of \glswrite has now been moved to \makeglossaries so that it's only defined if needed.

The \makeglossary command is redefined to be identical to \makeglossaries. (This is done to reinforce the message that you must either use \@makeglossary for all the glossaries or for none of them.)

\makeglossary

```
4819 \let\makeglossary\makeglossaries
```

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

```
4820 \AtEndDocument{%
4821   \warn@nomakeglossaries
4822   \warn@noprintglossary
4823 }
```

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

```
4824 \newcommand*{\makenoidxglossaries}{%
```

Redefine empty glossary warning:

```
4825   \renewcommand{\@gls@noref@warn}[1]{%
4826     \GlossariesWarning{Empty glossary for
4827       \string\printnoidxglossary[type={##1}].
4828     Rerun may be required (or you may have forgotten to use
4829     commands like \string\gls).}%
4830   }%
```

Don't escape makeindex/xindy characters

```
4831   \let\@gls@checkmkidxchars\@gobble
```

Write glossary information to aux instead of glossary files

```
4832   \let\@do@wrglossary\gls@noidxglossary
```

Switch on group headings that use the character code:

```
4833   \let\@gls@getgrouptitle\@gls@noidx@getgrouptitle
```

Allow see key:

```
4834   \let\gls@checkseeallowed\relax
```

Redefine cross-referencing macro:

```
4835   \renewcommand{\@do@seeglossary}[2]{%
4836     \edef\@gls@label{\glsdetoklabel{##1}}%
4837     \protected@write\@auxout{}{%
4838       \string\@gls@reference
4839       {\csname glo@\@gls@label @type\endcsname}%
4840       {\@gls@label}%
4841       {%
4842         \string\glsseeformat##2}%
4843       }%
4844     }%
4845   }%
```

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.

```
4846   \AtBeginDocument
4847   {%
4848     \write\@auxout{\string\providecommand\string\@gls@reference[3]{}}%
4849   }%
```

Change warning about no glossares

```
4850 \def\warn@noprintglossary{%
4851   \GlossariesWarningNoLine{No \string\printnoidxglossary\space
4852     or \string\printnoidxglossaries ^^J
4853     found. (Remove \string\makenoidxglossaries\space if you
4854     don't want any glossaries.)^^JThis document will not have a glossary}%
4855 }%
```

Suppress warning about no \makeglossaries

```
4856 \let\warn@nomakeglossaries\relax
```

Prevent user from also using \makeglossaries

```
4857 \let\makeglossaries\@no@makeglossaries
```

Allow sort key in printgloss family:

```
4858 \renewcommand*{\@printgloss@setsort}{%
4859   \let\@glo@assign@sortkey\@glo@assign@sortkey
```

Initialise default sort order:

```
4860   \def\@glo@sorttype{\@glo@default@sorttype}%
4861 }%
```

All entries must be defined in the preamble:

```
4862 \renewcommand*\new@glossaryentry[2]{%
4863   \PackageError{glossaries}{Glossary entries must be
4864     defined in the preamble^^Jwhen you use
4865     \string\makenoidxglossaries}%
4866   {Either move your definitions to the preamble or use
4867     \string\makeglossaries}%
4868 }%
```

Redefine \glsentrynumberlist

```
4869 \renewcommand*{\glsentrynumberlist}[1]{%
4870   \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
4871   \ifdef\@gls@loclist
4872     {%
4873       \glsnoidxloclist{\@gls@loclist}%
4874     }%
4875     {%
4876       ??\glsdoifexists{##1}%
4877       {%
4878         \GlossariesWarning{Missing location list for '##1'. Either
4879           a rerun is required or you haven't referenced the entry.}%
4880       }%
4881     }%
4882 }%
```

Redefine \glsdisplaynumberlist

```
4883 \renewcommand*{\glsdisplaynumberlist}[1]{%
4884   \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
4885   \ifdef\@gls@loclist
4886     {%
```

```

4887 \def\@gls@noidxloclist@sep{%
4888 \def\@gls@noidxloclist@sep{%
4889 \def\@gls@noidxloclist@sep{%
4890 \glsnumlistsep
4891 }%
4892 \def\@gls@noidxloclist@finalsep{\glsnumlistlastsep}%
4893 }%
4894 }%
4895 \def\@gls@noidxloclist@finalsep{}}%
4896 \def\@gls@noidxloclist@prev{}}%
4897 \forlistloop{\glsnoidxdisplayloclisthandler}{\@gls@loclist}%
4898 \@gls@noidxloclist@finalsep
4899 \@gls@noidxloclist@prev
4900 }%
4901 {%
4902 ??\glsdoifexists{##1}%
4903 {%
4904 \GlossariesWarning{Missing location list for ‘##1’. Either
4905 a rerun is required or you haven’t referenced the entry.}%
4906 }%
4907 }%
4908 }%

```

Provide a generic way of iterating through the number list:

```

4909 \renewcommand*\@glsnumberlistloop}[3]{%
4910 \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
4911 \let\@gls@org\glsnoidxdisplayloc\glsnoidxdisplayloc
4912 \let\@gls@org\glsseeformat\glsseeformat
4913 \let\glsnoidxdisplayloc##2\relax
4914 \let\glsseeformat##3\relax
4915 \ifdef\@gls@loclist
4916 {%
4917 \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
4918 }%
4919 {%
4920 ??\glsdoifexists{##1}%
4921 {%
4922 \GlossariesWarning{Missing location list for ‘##1’. Either
4923 a rerun is required or you haven’t referenced the entry.}%
4924 }%
4925 }%
4926 \let\glsnoidxdisplayloc\@gls@org\glsnoidxdisplayloc
4927 \let\glsseeformat\@gls@org\glsseeformat
4928 }%

```

Modify sanitize sort function

```

4929 \let\@gls@sanitizesort\@gls@noidx@sanitizesort
4930 \let\@gls@nosanitizesort\@gls@noidx@nosanitizesort
4931 \@gls@noidx@setsanitizesort
4932 }

```

Preamble-only command:

```
4933 \@onlypreamble{\makenoidxglossaries}
```

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

```
4934 \newcommand*{\glsnumberlistloop}[2]{%
4935   \PackageError{glossaries}{\string\glsnumberlistloop\space
4936     only works with \string\makenoidxglossaries}{}%
4937 }
```

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

```
4938 \newcommand*{\glsnoidxnumberlistloophandler}[1]{%
4939   #1%
4940 }
```

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

```
4941 \newcommand*{\@no@makeglossaries}{%
4942   \PackageError{glossaries}{You can't use both
4943     \string\makeglossaries\space and \string\makenoidxglossaries}%
4944   {Either use one or other (or none) of those commands but not both
4945     together.}%
4946 }
```

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

```
4947 \newcommand{\@gls@noref@warn}[1]{%
4948   \GlossariesWarning{\string\makenoidxglossaries\space
4949     is required to make \string\printnoidxglossary[type={#1}] work}%
4950 }
```

s@noidxglossary Write the glossary information to the aux file:

```
4951 \newcommand*{\gls@noidxglossary}{%
4952   \protected@write\@auxout{}{%
4953     \string\@gls@reference
4954       {\csname glo@\@gls@label @type\endcsname}%
4955       {\@gls@label}%
4956       {\string\glsnoidxdisplayloc
4957         {\@glo@counterprefix}%
4958         {\@gls@counter}%
4959         {\@glsnumberformat}%
4960         {\@glslocref}%
4961       }%
4962   }%
4963 }
```

1.14 Writing information to associated files

`\istfile` Deprecated.

```
4964 \def\istfile{\glswrite}
```

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

```
4965 \AtEndDocument{%
```

```
4966   \glswritefiles
```

```
4967 }
```

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

```
4968 \newcommand*{\@glswritefiles}{%
```

Iterate through all the glossaries

```
4969   \forallglossaries{\@glo@type}{%
```

Check for empty glossaries (patch provided by Patrick Häcker)

```
4970     \ifcsundef{glo@\@glo@type @filetok}%
```

```
4971     {%
```

```
4972       \def\gls@tmp{}%
```

```
4973     }%
```

```
4974     {%
```

```
4975       \edef\gls@tmp{\expandafter\the
```

```
4976         \csname glo@\@glo@type @filetok\endcsname}%
```

```
4977     }%
```

```
4978     \ifx\gls@tmp\@empty
```

```
4979       \ifx\@glo@type\glsdefaulttype
```

```
4980         \GlossariesWarningNoLine{Glossary ‘\@glo@type’ has no
```

```
4981           entries.^^JRemember to use package option ‘nomain’ if
```

```
4982 you
```

```
4983           don’t want to^^Juse the main glossary}%
```

```
4984       \else
```

```
4985         \GlossariesWarningNoLine{Glossary ‘\@glo@type’ has no
```

```
4986           entries}%
```

```
4987       \fi
```

```
4988     \else
```

```
4989       \@glsopenfile{\glswrite}{\@glo@type}%
```

```
4990       \immediate\write\glswrite{%
```

```
4991         \expandafter\the
```

```
4992         \csname glo@\@glo@type @filetok\endcsname}%
```

```
4993       \immediate\closeout\glswrite
```

```
4994     \fi
```

```
4995   }%
```

```
4996 }
```

As from v4.10, the `\glossary` command is used by the `glossaries` package. Since the user isn't expected to use this command (as `glossaries` takes care of the particular format required for `makeindex/xindy`) there's no need for a user level command. Using a custom internal command prevents any conflict with other packages (and with the `\mark` mechanism).

In v4.10, the redefinition of `\glossary` was removed since it wasn't intended as a user level command, however it seems there are packages that have hacked the internal macros used by glossaries and no longer work with this redefinition removed, so it's been restored in v4.11 but is not used at all by glossaries. (This may be removed or moved to a compatibility mode in future.)

```
\glossary
4997 \if@gls@docloaded
4998 \else
4999   \renewcommand*{\glossary}[1][main]{\gls@glossary{#1}}
5000 \fi
```

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

```
\gls@glossary
5001 \newcommand*{\gls@glossary}[1]{%
5002   \@gls@glossary{#1}%
5003 }
```

`\@gls@glossary` (In v4.10, `\@glossary` was redefined to `\@gls@glossary` to avoid conflict with other packages.) Define internal `\@gls@glossary` to ignore its argument. This gets redefined in `\@makeglossary`. This is defined to just `\index` as memoir changes the definition of `\@index`. (Thanks to Dan Luecking for pointing this out.) The argument #1 is the glossary type.

```
5004 \newcommand*{\@gls@glossary}[1]{\index}
```

This is a convenience command to set `\@gls@glossary`. It's used by `\@makeglossary` and then redefined to do nothing, as it only needs to be done once.

```
\@renewglossary
5005 \newcommand{\@gls@renewglossary}{%
5006   \gdef\@gls@glossary##1{\@bsphack\begin{group}\gls@wrglossary{##1}}%
5007   \let\@gls@renewglossary\empty
5008 }
```

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
5009 \newcommand*{\gls@wrglossary}[2]{%
5010   \ifglssavewrites
5011     \protected@edef\@gls@tmp{\the\csname glo@#1@filetok\endcsname#2}%
5012     \expandafter\global\expandafter\csname glo@#1@filetok\endcsname
5013       \expandafter{\@gls@tmp~J}%
5014   \else
```

```

5015 \ifcsdef{glo@#1@file}%
5016 {%
5017 \expandafter\protected@write\csname glo@#1@file\endcsname{%
5018 \gls@disablepagerefexpansion}{#2}%
5019 }%
5020 {%
5021 \ifignoredglossary{#1}{}%
5022 {%
5023 \GlossariesWarning{No file defined for glossary ‘#1’}%
5024 }%
5025 }%
5026 \fi
5027 \endgroup\@esphack
5028 }

```

\do@wrglossary

```

5029 \newcommand*\do@wrglossary}[1]{%
5030 \glswriteentry{#1}{\do@wrglossary{#1}}%
5031 }

```

\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.

```

5032 \newcommand*\glswriteentry}[2]{%
5033 \ifglsindexonlyfirst
5034 \ifglsused{#1}{#2}%
5035 \else
5036 #2%
5037 \fi
5038 }

```

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

```

5039 \newcommand*\gls@protected@pagefmts{%
5040 \gls@numberpage,\gls@alphpage,\gls@Alphpage,\gls@romanpage,\gls@Romanpage,\gls@arabicpage%
5041 }

```

\gls@disablepagerefexpansion

```

5042 \newcommand*\gls@disablepagerefexpansion{%
5043 \@for\@gls@this:=\gls@protected@pagefmts\do
5044 {%
5045 \expandafter\let\@gls@this\relax
5046 }%
5047 }

```

\gls@alphpage

```

5048 \newcommand*\gls@alphpage{\@alph\c@page}

```

\gls@Alphpage

```

5049 \newcommand*\gls@Alphpage{\@Alph\c@page}

```

`\gls@numberpage`

```
5050 \newcommand*{\gls@numberpage}{\number\c@page}
```

`\gls@arabicpage`

```
5051 \newcommand*{\gls@arabicpage}{\@arabic\c@page}
```

`\gls@romanpage`

```
5052 \newcommand*{\gls@romanpage}{\romannumeral\c@page}
```

`\gls@Romanpage`

```
5053 \newcommand*{\gls@Romanpage}{\@Roman\c@page}
```

`protectedpagefmt`

```
\glsaddprotectedpagefmt{<cs name>}
```

Added a page format to the list of protected page formats. The argument should be the name (without a backslash) of the command that takes a \TeX register as the argument (`\<csname>\c@page` must be valid).

```
5054 \newcommand*{\glsaddprotectedpagefmt}[1]{%
5055   \eappto\gls@protected@pagefmts{,\expandonce{\csname gls#1page\endcsname}}%
5056   \csedef{gls#1page}{\expandonce{\csname#1\endcsname}\noexpand\c@page}%
5057   \eappto\@wrglossarynumberhook{%
5058     \noexpand\let\expandonce{\csname org@gls#1\endcsname}%
5059     \expandonce{\csname#1\endcsname}%
5060     \noexpand\def\expandonce{\csname#1\endcsname}{%
5061       \noexpand\@wrglossary@pageformat
5062       \expandonce{\csname gls#1page\endcsname}%
5063       \expandonce{\csname org@gls#1\endcsname}%
5064     }%
5065   }%
5066 }
```

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

```
5067 \newcommand*\@wrglossarynumberhook{}
```

`sary@pageformat`

```
5068 \newcommand{\@wrglossary@pageformat}[3]{%
5069   \ifx#3\c@page #1\else #2#3\fi
5070 }
```

`owprimitivemods` Conditional to determine whether or not `\@do@wrglossary` should be allowed to temporarily redefine `\the` and `\number`.

```
5071 \newif\ifglswrallowprimitivemods
5072 \glswrallowprimitivemodstrue
```

`@do@wrglossary` Write the glossary entry in the appropriate format. (Need to set `\@glsnumberformat` and `\@gls@counter` prior to use.) The argument is the entry's label.

```
5073 \newcommand*{\@do@wrglossary}[1]{%
5074   \begingroup
```

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

```
5075   \let\orgthe\the
5076   \let\orgnumber\number

5077   \let\orgarabic\@arabic
5078   \let\orgromannumeral\romannumeral
5079   \let\orgalph\@alph
5080   \let\orgAlph\@Alph
5081   \let\orgRoman\@Roman
```

Redefine:

```
5082   \ifglswrallowprimitivemods
5083     \def\the##1{%
5084       \ifx##1\c@page \gls@numberpage\else\orgthe##1\fi}%
5085     \def\number##1{%
5086       \ifx##1\c@page \gls@numberpage\else\orgnumber##1\fi}%
5087     \fi
5088     \def\@arabic##1{%
5089       \ifx##1\c@page \gls@arabicpage\else\orgarabic##1\fi}%
5090     \def\romannumeral##1{%
5091       \ifx##1\c@page \gls@romanpage\else\orgromannumeral##1\fi}%
5092     \def\@Roman##1{%
5093       \ifx##1\c@page \gls@Romanpage\else\orgRoman##1\fi}%
5094     \def\@alph##1{%
5095       \ifx##1\c@page \gls@alphpage\else\orgalph##1\fi}%
5096     \def\@Alph##1{%
5097       \ifx##1\c@page \gls@Alphpage\else\orgAlph##1\fi}%
```

Add hook to allow for other number formats:

```
5098   \@wrglossarynumberhook
```

Prevent expansion:

```
5099   \gls@disablepagerefexpansion
```

Now store location in `\@glslocref`:

```
5100   \protected@xdef\@glslocref{\theHglentrycounter}%
5101   \endgroup
```

Escape any special characters

```
5102   \@gls@checkmkidxchars\@glslocref
```

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

```
5103   \expandafter\ifx\theHglentrycounter\theHglentrycounter\relax
5104   \def\@glo@counterprefix{}%
5105   \else
5106     \protected@edef\@glsHlocref{\theHglentrycounter}%
5107     \@gls@checkmkidxchars\@glsHlocref
```

```

5108 \edef\@do@glS@getcounterprefix{\noexpand\@glS@getcounterprefix
5109 {\@glS@locref}{\@glS@Hlocref}%
5110 }%
5111 \@do@glS@getcounterprefix
5112 \fi

```

De-tok label if required

```

5113 \edef\@glS@label{\glSdetoklabel{#1}}%

```

Write the information to file:

```

5114 \@do@wrglossary
5115 }

```

@do@wrglossary

```

5116 \newcommand*{\@do@wrglossary}{%

```

Determine whether to use xindy or makeindex syntax

```

5117 \ifglSxindy

```

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

```

5118 \expandafter\@glo@check@mkidxrangechar\@glS@numberformat\@nil
5119 \def\@glo@range{}%
5120 \expandafter\if\@glo@prefix(\relax
5121 \def\@glo@range{:open-range}%
5122 \else
5123 \expandafter\if\@glo@prefix)\relax
5124 \def\@glo@range{:close-range}%
5125 \fi
5126 \fi

```

Write to the glossary file using xindy syntax.

```

5127 \glS@glossary{\csname glo@\@glS@label @type\endcsname}{%
5128 (indexentry :tkey (\csname glo@\@glS@label @index\endcsname)

5129 :locref \string"{\@glo@counterprefix}{\@glS@locref}\string" %
5130 :attr \string"\@glS@counter\@glo@suffix\string"
5131 \@glo@range
5132 )
5133 }%
5134 \else

```

Convert the format information into the format required for makeindex

```

5135 \@set@glo@numformat{\@glo@numfmt}{\@glS@counter}{\@glS@numberformat}%
5136 {\@glo@counterprefix}%

```

Write to the glossary file using makeindex syntax.

```

5137 \glS@glossary{\csname glo@\@glS@label @type\endcsname}{%
5138 \string\glossaryentry{\csname glo@\@glS@label @index\endcsname
5139 \@glS@encapchar\@glo@numfmt}{\@glS@locref}}%
5140 \fi
5141 }

```

`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.)

```

5142 \newcommand*\gls@getcounterprefix[2]{%
5143   \edef\@gls@thisloc{#1}\edef\@gls@thisHloc{#2}%
5144   \ifx\@gls@thisloc\@gls@thisHloc
5145     \def\@glo@counterprefix{}%
5146   \else
5147     \def\@gls@get@counterprefix##1.#1##2\end@getprefix{%
5148       \def\@glo@tmp{##2}%
5149       \ifx\@glo@tmp\@empty
5150         \def\@glo@counterprefix{}%
5151       \else
5152         \def\@glo@counterprefix{##1}%
5153       \fi
5154     }%
5155     \@gls@get@counterprefix#2.#1\end@getprefix

```

Warn if no prefix can be formed.

```

5156   \ifx\@glo@counterprefix\@empty
5157     \GlossariesWarning{Hyper target ‘#2’ can’t be formed by
5158       prefixing^^Jlocation ‘#1’. You need to modify the
5159       definition of \string\theH\@gls@counter^^Jotherwise you
5160       will get the warning: “name{\@gls@counter.#1}’ has been^^J
5161       referenced but does not exist"%
5162   \fi
5163 \fi
5164 }

```

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.

```

5165 \newcommand{\@do@seeglossary}[2]{%
5166   \def\@gls@xref{#2}%
5167   \@onelevel@sanitize\@gls@xref
5168   \@gls@checkmkidxchars\@gls@xref
5169   \ifglxindy
5170     \gls@glossary{\csname glo@#1@type\endcsname}{%
5171       (indexentry
5172         :tkey (\csname glo@#1@index\endcsname)
5173         :xref (\string"\@gls@xref\string")
5174         :attr \string"see\string"
5175       )
5176     }%

```

```

5177 \else
5178   \gls@glossary{\csname glo@#1@type\endcsname}{%
5179   \string\glossaryentry{\csname glo@#1@index\endcsname
5180   \@gls@encapchar glsseeformat\@gls@xref}{Z}}}%
5181 \fi
5182 }

```

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

```

5183 \def\@gls@fixbraces#1#2#3\@nil{%
5184   \ifx#2[\relax
5185     \@gls@fixbraces#1#2#3\@end@fixbraces
5186   \else
5187     \def#1{{#2#3}}%
5188   \fi
5189 }

```

`@@gls@fixbraces`

```

5190 \def@@gls@fixbraces#1[#2]#3\@end@fixbraces{%
5191   \def#1{[#2]{#3}}%
5192 }

```

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

```

5193 \DeclareRobustCommand*\glssee{3}[\seename]{%
5194   \@do@seeglossary{#2}{#1}{#3}}
5195 \newcommand*\@glssee{3}[\seename]{%
5196   \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.

```

5197 \DeclareRobustCommand*\glsseeformat{3}[\seename]{%
5198   \emph{#1} \glsseelist{#2}}

```

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

```

5199 \DeclareRobustCommand*\glsseelist{1}{%

```

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

```

5200   \let\@gls@dolast\relax

```

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

```

5201   \let\@gls@donext\relax

```

Iterate through the labels

```

5202   \@for\@gls@thislabel:=#1\do{%

```

Check if on last iteration of loop

```

5203     \ifx\@xfor@nextelement\@nnil
5204       \@gls@dolast
5205     \else
5206       \@gls@donext
5207     \fi

```

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

```
5208 \expandafter\glsseeitem\expandafter{\@gls@thislabel}%
```

Update separators

```
5209 \let\@gls@dolast\glsseelastsep
```

```
5210 \let\@gls@donext\glsseesep
```

```
5211 }%
```

```
5212 }
```

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

```
5213 \newcommand*{\glsseelastsep}{\space\andname\space}
```

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

```
5214 \newcommand*{\glsseesep}{, }
```

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

```
5215 \DeclareRobustCommand*{\glsseeitem}[1]{\gls hyperlink[\glsseeitemformat{#1}]{#1}}
```

`\glsseeitemformat` As from v3.0, default is to use `\glsentrytext` instead of `\glsentryname`. (To avoid problems with the name key being sanitized.)

```
5216 \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.

```
5217 \newcommand*{\gls@save@numberlist}[1]{%
```

```
5218 \ifglssavenumberlist
```

```
5219 \toks@{#1}%
```

```
5220 \edef\@do@writeaux@info{%
```

```
5221 \noexpand\csgdef{glo@\glscurrententrylabel @numberlist}{\the\toks@}%
```

```
5222 }%
```

```
5223 \@onelevel@sanitize\@do@writeaux@info
```

```
5224 \protected@write\@auxout{}{\@do@writeaux@info}%
```

```
5225 \fi
```

```
5226 }
```

`\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.)

```
5227 \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.

```
5228 \ifcsundef{printglossary}{}%  
5229 {%
```

If `\printglossary` is already defined, issue a warning and undefine it.

```
5230 \@gls@warnonglossdefined  
5231 \undef\printglossary  
5232 }
```

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

```
5233 \newcommand*{\printglossary}[1][type=\glsdefaulttype]{%  
5234 \@printglossary{#1}{\@print@glossary}%  
5235 }
```

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`

```
5236 \newcommand*{\printglossaries}{%  
5237 \forallglossaries{\@glo@type}{\printglossary[type=\@glo@type]}%  
5238 }
```

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

```
5239 \newcommand*{\printnoidxglossary}[1][type=\glsdefaulttype]{%  
5240 \@printglossary{#1}{\@print@noidx@glossary}%  
5241 }
```

`printnoidxglossaries` Analogous to `\printglossaries`

```
5242 \newcommand*{\printnoidxglossaries}{%  
5243 \forallglossaries{\@glo@type}{\printnoidxglossary[type=\@glo@type]}%  
5244 }
```

`printgloss@setsort` Initialise to do nothing.

```
5245 \newcommand*{\@printgloss@setsort}{}%
```

`preglossaryhook`

```
5246 \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.

```
5247 \newcommand{\@printglossary}[2]{%
```

Set up defaults.

```
5248 \def\@glo@type{\glsdefaulttype}%
```

```
5249 \def\glossarytitle{\csname @glo@type\@glo@type @title\endcsname}%
```

```
5250 \def\glossarytoctitle{\glossarytitle}%
```

```
5251 \let\org@glossarytitle\glossarytitle
```

```
5252 \def\@glossarystyle{%
```

```
5253 \ifx\@glossary@default@style\relax
```

```
5254 \GlossariesWarning{No default glossary style provided \MessageBreak
```

```
5255 for the glossary ‘\@glo@type’. \MessageBreak
```

```
5256 Using deprecated fallback. \MessageBreak
```

```
5257 To fix this set the style with \MessageBreak
```

```
5258 \string\setglossarystyle\space or use the \MessageBreak
```

```
5259 style key=value option}%
```

```
5260 \fi
```

```
5261 }%
```

```
5262 \def\gls@dotoc@title{\gls@set@toc@title{\@glo@type}}%
```

Store current value of `\glossaryentrynumbers`. (This may be changed via the optional argument)

```
5263 \let\@org@glossaryentrynumbers\glossaryentrynumbers
```

Localise the effects of the optional argument

```
5264 \bgroup
```

Activate or deactivate sort key:

```
5265 \@printgloss@setsort
```

Determine settings specified in the optional argument.

```
5266 \setkeys{printgloss}{#1}%
```

If title has been set, but toctitle hasn't, make toctitle the same as given title (rather than the title used when the glossary was defined)

```
5267 \ifx\glossarytitle\org@glossarytitle
```

```
5268 \else
```

```
5269 \expandafter\let\csname @glo@type\@glo@type @title\endcsname
```

```
5270 \glossarytitle
```

```
5271 \fi
```

Allow a high-level user command to indicate the current glossary

```
5272 \let\currentglossary\@glo@type
```

Enable individual number lists to be suppressed.

```
5273 \let\org@glossaryentrynumbers\glossaryentrynumbers
```

```
5274 \let\glsnonextpages\glsnonextpages
```

Enable individual number list to be activated:

```
5275 \let\glsnextpages\@glsnextpages
```

Enable suppression of description terminators.

```
5276 \let\nopostdesc\@nopostdesc
```

Set up the entry for the TOC

```
5277 \gls@dotocitle
```

Set the glossary style

```
5278 \@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):

```
5279 \let\gls@org@glossaryentryfield\glossentry
5280 \let\gls@org@glossarysubentryfield\subglossentry
5281 \renewcommand{\glossentry}[1]{%
5282   \xdef\glscurrententrylabel{\glsdetoklabel{##1}}%
5283   \gls@org@glossaryentryfield{##1}%
5284 }%
5285 \renewcommand{\subglossentry}[2]{%
5286   \xdef\glscurrententrylabel{\glsdetoklabel{##2}}%
5287   \gls@org@glossarysubentryfield{##1}{##2}%
5288 }%
```

```
5289 \@gls@preglossaryhook
```

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

```
5290 #2%
```

End the current scope

```
5291 \egroup
```

Reset `\glossaryentrynumbers`

```
5292 \global\let\glossaryentrynumbers\@org@glossaryentrynumbers
```

Suppress warning about no `\printglossary`

```
5293 \global\let\warn@noprintglossary\relax
5294 }
```

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

```
5295 \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.)

```
5296 \makeatletter
```

Input the glossary file, if it exists.

```
5297 \@input@{\jobname.\csname @glo@type\@glo@type @in\endcsname}%
```

If the glossary file doesn't exist, do `\null`. (This ensures that the page is shipped out and all write commands are done.) This might produce an empty page, but at this point the document isn't complete, so it shouldn't matter.

```
5298 \IfFileExists{\jobname.\csname @glo@type\@glo@type @in\endcsname}%
5299 {}%
5300 {\null}%
```

If `xindy` is being used, need to write the language dependent information to the `.aux` file for `makeglossaries`.

```
5301 \ifglxindy
5302 \ifcsundef{@xdy@\@glo@type @language}%
5303 {%
5304     \edef\@do@auxoutstuff{%
5305         \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.

```
5306         \noexpand\immediate\noexpand\write\@auxout{%
5307             \string\providecommand\string\@xdylanguage[2]{}}%
5308         \noexpand\immediate\noexpand\write\@auxout{%
5309             \string\@xdylanguage{\@glo@type}{\@xdy@main@language}}%
5310     }%
5311 }%
5312 }%
5313 {%
5314     \edef\@do@auxoutstuff{%
5315         \noexpand\AtEndDocument{%
5316             \noexpand\immediate\noexpand\write\@auxout{%
5317                 \string\providecommand\string\@xdylanguage[2]{}}%
5318             \noexpand\immediate\noexpand\write\@auxout{%
5319                 \string\@xdylanguage{\@glo@type}{\csname @xdy@\@glo@type
5320                     @language\endcsname}}%
5321             }%
5322         }%
5323     }%
5324     \@do@auxoutstuff
5325     \edef\@do@auxoutstuff{%
5326         \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.

```
5327         \noexpand\immediate\noexpand\write\@auxout{%
5328             \string\providecommand\string\@gls@codepage[2]{}}%
5329         \noexpand\immediate\noexpand\write\@auxout{%
5330             \string\@gls@codepage{\@glo@type}{\@gls@codepage}}%
5331     }%
5332 }%
5333 \@do@auxoutstuff
5334 \fi
```

Activate warning if \makeglossaries hasn't been used.

```

5335 \renewcommand*{\@warn@nomakeglossaries}{%
5336   \GlossariesWarningNoLine{\string\makeglossaries\space
5337     hasn't been used,^^Jthe glossaries will not be updated}%
5338   }%
5339 }

```

The sort macros all have the syntax:

$$\backslash@glo@sortmacro@<order>\{<type>\}$$

where $<order>$ is the sort order as specified by the sort key and $<type>$ is the glossary type. (The referenced entry list is stored in $\backslash@gl{sref@<type>}$. The actual sorting is done by $\backslash@glo@sortentries\{<handler>\}\{<type>\}$).

$\backslash@glo@sortentries$

```

5340 \newcommand*{\@glo@sortentries}[2]{%
5341   \def\@glo@sortinglist{}%
5342   \def\@glo@sortinghandler{#1}%
5343   \edef\@glo@type{#2}%
5344   \forlistcsloop{\@glo@do@sortentries}\@gl{sref@#2}%
5345   \csdef{\@gl{sref@#2}}{}%
5346   \@for\@this@label:=\@glo@sortinglist\do{%

```

Has this entry already been added?

```

5347     \xifinlistcs{\@this@label}\@gl{sref@#2}%
5348     {}%
5349     {%
5350       \listcsxadd{\@gl{sref@#2}}{\@this@label}%
5351     }%
5352     \ifcsdef{\@glo@sortingchildren@\@this@label}%
5353     {%
5354       \@glo@addchildren{#2}\@this@label}%
5355     }%
5356     {}%
5357   }%
5358 }

```

$\backslash@glo@addchildren$

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

```

5359 \newcommand*{\@glo@addchildren}[2]{%

```

Scope to allow nesting.

```

5360   \bgroup
5361     \letcs{\@glo@childlist}\@glo@sortingchildren@#2}%
5362     \@for\@this@childlabel:=\@glo@childlist\do
5363     {%

```

Check this label hasn't already been added.

```
5364      \xifinlistcs{\@this@childlabel}{\@glsref@#1}%  
5365      {%  
5366      {%  
5367      \listcsxadd{\@glsref@#1}{\@this@childlabel}%  
5368      }%
```

Does this child have children?

```
5369      \ifcsdef{@glo@sortingchildren@\@this@childlabel}%  
5370      {%  
5371      \@glo@addchildren{#1}{\@this@childlabel}%  
5372      }%  
5373      {%  
5374      }%  
5375      }%  
5376 \egroup  
5377 }
```

@do@sortentries

```
5378 \newcommand*{\@glo@do@sortentries}[1]{%  
5379 \ifglshasparent{#1}%  
5380 {%
```

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

```
5381 \edef\@glo@parent{\csuse{glo@\glsdetoklabel{#1}@parent}}%  
5382 \ifcsundef{@glo@sortingchildren@\@glo@parent}%  
5383 {%  
5384 \csdef{@glo@sortingchildren@\@glo@parent}{}%  
5385 }%  
5386 {}%  
5387 \expandafter\@glo@sortedinsert  
5388 \csname @glo@sortingchildren@\@glo@parent\endcsname{#1}%
```

Has the parent been added?

```
5389 \xifinlistcs{\@glo@parent}{\@glsref@\@glo@type}%  
5390 {%
```

Yes, it has so do nothing.

```
5391 }%  
5392 {%
```

No, it hasn't so add it now.

```
5393 \expandafter\@glo@do@sortentries\expandafter{\@glo@parent}%  
5394 }%  
5395 }%  
5396 {%  
5397 \@glo@sortedinsert{\@glo@sortinglist}{#1}%  
5398 }%  
5399 }
```

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

Insert into list.

```
5400 \newcommand*{\@glo@sortedinsert}[2]{%
5401   \dtl@insertinto{#2}{#1}{\@glo@sortinghandler}%
5402 }%
```

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

```
5403 \newcommand*{\@glo@sorthandler@word}[2]{%
5404   \letcs\@gls@sort@A{glo@glstdetoklabel{#1}@sort}%
5405   \letcs\@gls@sort@B{glo@glstdetoklabel{#2}@sort}%
5406   \edef\glo@do@compare{%
5407     \noexpand\dtlwordindexcompare{\noexpand\dtl@sortresult}%
5408     {\expandonce\@gls@sort@B}%
5409     {\expandonce\@gls@sort@A}%
5410   }%
5411   \glo@do@compare
5412 }
```

thandler@letter

```
5413 \newcommand*{\@glo@sorthandler@letter}[2]{%
5414   \letcs\@gls@sort@A{glo@glstdetoklabel{#1}@sort}%
5415   \letcs\@gls@sort@B{glo@glstdetoklabel{#2}@sort}%
5416   \edef\glo@do@compare{%
5417     \noexpand\dtlletterindexcompare{\noexpand\dtl@sortresult}%
5418     {\expandonce\@gls@sort@B}%
5419     {\expandonce\@gls@sort@A}%
5420   }%
5421   \glo@do@compare
5422 }
```

orthandler@case Case-sensitive sort.

```
5423 \newcommand*{\@glo@sorthandler@case}[2]{%
5424   \letcs\@gls@sort@A{glo@glstdetoklabel{#1}@sort}%
5425   \letcs\@gls@sort@B{glo@glstdetoklabel{#2}@sort}%
5426   \edef\glo@do@compare{%
5427     \noexpand\dtlcompare{\noexpand\dtl@sortresult}%
5428     {\expandonce\@gls@sort@B}%
5429     {\expandonce\@gls@sort@A}%
5430   }%
5431   \glo@do@compare
5432 }
```

thandler@nocase Case-insensitive sort.

```

5433 \newcommand*{\@glo@sorthandler@nocase}[2]{%
5434   \letcs\@gls@sort@A{glo\glsdetoklabel{#1}@sort}%
5435   \letcs\@gls@sort@B{glo\glsdetoklabel{#2}@sort}%
5436   \edef\glo@do@compare{%
5437     \noexpand\dtlicompare{\noexpand\dtl@sortresult}%
5438     {\expandonce\@gls@sort@B}%
5439     {\expandonce\@gls@sort@A}%
5440   }%
5441   \glo@do@compare
5442 }

```

@sortmacro@word Sort macro for ‘word’

```

5443 \newcommand*{\@glo@sortmacro@word}[1]{%
5444   \ifdefstring{\@glo@default@sorttype}{standard}%
5445   {%
5446     \@glo@sortentries{\@glo@sorthandler@word}{#1}%
5447   }%
5448   {%
5449     \PackageError{glossaries}{Conflicting sort options:^^J
5450       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5451       \string\printnoidxglossary[sort=word]}{ }%
5452   }%
5453 }

```

ortmacro@letter Sort macro for ‘letter’

```

5454 \newcommand*{\@glo@sortmacro@letter}[1]{%
5455   \ifdefstring{\@glo@default@sorttype}{standard}%
5456   {%
5457     \@glo@sortentries{\@glo@sorthandler@letter}{#1}%
5458   }%
5459   {%
5460     \PackageError{glossaries}{Conflicting sort options:^^J
5461       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5462       \string\printnoidxglossary[sort=letter]}{ }%
5463   }%
5464 }

```

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

```

5465 \newcommand*{\@glo@sortmacro@standard}[1]{%
5466   \ifdefstring{\@glo@default@sorttype}{standard}%
5467   {%
5468     \ifcsdef{\@glo@sorthandler@\glsorder}%
5469     {%
5470       \@glo@sortentries{\csuse{\@glo@sorthandler@\glsorder}}{#1}%
5471     }%
5472     {%
5473       \PackageError{glossaries}{Unknown sort handler ‘\glsorder’}{ }%
5474     }%
5475   }%

```



```

5476 {%
5477   \PackageError{glossaries}{Conflicting sort options:^^J
5478     \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5479     \string\printnoidxglossary[sort=standard]}{}}%
5480 }%
5481 }

```

@sortmacro@case Sort macro for ‘case’

```

5482 \newcommand*{\@glo@sortmacro@case}[1]{%
5483   \ifdefstring{\@glo@default@sorttype}{standard}%
5484   {%
5485     \@glo@sortentries{\@glo@sorthandler@case}{#1}%
5486   }%
5487   {%
5488     \PackageError{glossaries}{Conflicting sort options:^^J
5489       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5490       \string\printnoidxglossary[sort=case]}{}}%
5491   }%
5492 }

```

@sortmacro@nocase Sort macro for ‘nocase’

```

5493 \newcommand*{\@glo@sortmacro@nocase}[1]{%
5494   \ifdefstring{\@glo@default@sorttype}{standard}%
5495   {%
5496     \@glo@sortentries{\@glo@sorthandler@nocase}{#1}%
5497   }%
5498   {%
5499     \PackageError{glossaries}{Conflicting sort options:^^J
5500       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5501       \string\printnoidxglossary[sort=nocase]}{}}%
5502   }%
5503 }

```

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

```

5504 \newcommand*{\@glo@sortmacro@def}[1]{%
5505   \def\@glo@sortinglist{%
5506     \for@gl@sentries[#1]{\@gl@thislabel}%
5507     {%
5508       \xifinlistcs{\@gl@thislabel}{\@gl@ref@#1}%
5509       {%
5510         \list@add{\@glo@sortinglist}{\@gl@thislabel}%
5511       }%
5512     }%

```

Hasn't been referenced.

```

5513   }%
5514 }%
5515 \cslet{\@gl@ref@#1}{\@glo@sortinglist}%
5516 }

```

ortmacro@def@do This won't include parent entries that haven't been referenced.

```
5517 \newcommand*{\@glo@sortmacro@def@do}[1]{%
5518   \ifinlistcs{#1}{\@glsref@\@glo@type}%
5519   {}%
5520   {%
5521     \listcsadd{\@glsref@\@glo@type}{#1}%
5522   }%
5523   \ifcsdef{\@glo@sortingchildren@#1}%
5524   {%
5525     \@glo@addchildren{\@glo@type}{#1}%
5526   }%
5527   {}%
5528 }
```

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

```
5529 \newcommand*{\@glo@sortmacro@use}[1]{}
```

@noidx@glossary Glossary handler for \printnoidxglossary which doesn't use an indexing application. Since \printnoidxglossary may occur at the start of the document, we can't just check if an entry has been used. Instead, the first pass needs to write information to the aux file every time an entry is referenced. This needs to be read in on the second run and stored in a list corresponding to the appropriate glossary.

```
5530 \newcommand*{\@print@noidx@glossary}{%
5531   \ifcsdef{\@glsref@\@glo@type}%
5532   {%
```

Sort the entries:

```
5533   \ifcsdef{\@glo@sortmacro@\@glo@sorttype}%
5534   {%
5535     \csuse{\@glo@sortmacro@\@glo@sorttype}{\@glo@type}%
5536   }%
5537   {%
5538     \PackageError{glossaries}{Unknown sort handler '\@glo@sorttype'}{}%
5539   }%
```

Do the glossary heading and preamble

```
5540   \glossarysection[\glossarytoctitle]{\glossarytitle}%
5541   \glossarypreamble
5542   \begin{theglossary}%
5543     \glossaryheader
5544     \glsresetentrylist
5545     \def\@gls@currentlettergroup{}
```

Iterate through the entries.

```
5546   \forlistcsloop{\@gls@noidx@do}{\@glsref@\@glo@type}%
```

Finally end the glossary and do the postamble:

```
5547   \end{theglossary}%
5548   \glossarypostamble
```

```

5549 }%
5550 {%
5551   \@gls@noref@warn{\@glo@type}%
5552 }%
5553 }

```

`\glo@grabfirst`

```

5554 \def\glo@grabfirst#1#2\@nil{%
5555   \def\@gls@firsttok{#1}%
5556   \ifdefempty\@gls@firsttok
5557   {%
5558     \def\@glo@thislettergrp{0}%
5559   }%
5560   {%

```

Sanitize it:

```

5561   \@onelevel@sanitize\@gls@firsttok

```

Fetch the first letter:

```

5562   \expandafter\@glo@grabfirst\@gls@firsttok{}{}\@nil
5563 }%
5564 }

```

`\@glo@grabfirst`

```

5565 \def\@glo@grabfirst#1#2\@nil{%
5566   \ifdefempty\@glo@thislettergrp
5567   {%
5568     \def\@glo@thislettergrp{glssymbols}%
5569   }%
5570   {%
5571     \count@=\uccode'#1\relax
5572     \ifnum\count@=0\relax
5573       \def\@glo@thislettergrp{glssymbols}%
5574     \else
5575       \ifdefstring\@glo@sorttype{case}%
5576       {%
5577         \count@='#1\relax
5578       }%
5579       {%
5580     }%
5581     \edef\@glo@thislettergrp{\the\count@}%
5582   \fi
5583 }%
5584 }

```

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

```

5585 \newcommand{\@gls@noidx@do}[1]{%

```

Get this entry's location list

```

5586   \global\letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%

```

Does this entry have a parent?

```
5587 \ifglshasparent{#1}%  
5588 {%
```

Has a parent.

```
5589 \gls@level=\csuse{glo@\glsdetoklabel{#1}@level}\relax  
5590 \ifdefvoid{\@gls@loclist}  
5591 {%  
5592 \subglossentry{\gls@level}{#1}{}%  
5593 }%  
5594 {%  
5595 \subglossentry{\gls@level}{#1}%  
5596 {%  
5597 \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%  
5598 }%  
5599 }%  
5600 }%  
5601 {%
```

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

```
5602 \letcs{\@gls@sort}{glo@\glsdetoklabel{#1}@sort}%
```

Fetch the first letter:

```
5603 \expandafter\glo@grabfirst\@gls@sort{}{}\@nil  
5604 \ifdefequal{\@glo@thislettergrp}{\@gls@currentlettergroup}%  
5605 {}%  
5606 {%
```

Do the group header:

```
5607 \ifdefempty{\@gls@currentlettergroup}{\@gls@groupskip}%  
5608 \gls@groupheading{\@glo@thislettergrp}%  
5609 }%  
5610 \let\@gls@currentlettergroup\@glo@thislettergrp
```

Do this entry:

```
5611 \ifdefvoid{\@gls@loclist}  
5612 {%  
5613 \glossentry{#1}{}%  
5614 }%  
5615 {%  
5616 \glossentry{#1}%  
5617 {%  
5618 \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%  
5619 }%  
5620 }%  
5621 }%  
5622 }
```

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

Display location list.

```

5623 \newcommand*\glsnoidxloclist}[1]{%
5624   \def\@gls@noidxloclist@sep{}%
5625   \def\@gls@noidxloclist@prev{}%
5626   \forlistloop{\glsnoidxloclisthandler}{#1}%
5627 }

```

`xloclisthandler` Handler for location list iterator.

```

5628 \newcommand*\glsnoidxloclisthandler}[1]{%
5629   \ifdefstring{\@gls@noidxloclist@prev}{#1}%
5630   {%

```

Same as previous location so skip.

```

5631   }%
5632   {%
5633     \@gls@noidxloclist@sep
5634     #1%
5635     \def\@gls@noidxloclist@sep{\delimN}%
5636     \def\@gls@noidxloclist@prev{#1}%
5637   }%
5638 }

```

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

```

5639 \newcommand*\glsnoidxdisplayloclisthandler}[1]{%
5640   \ifdefstring{\@gls@noidxloclist@prev}{#1}%
5641   {%

```

Same as previous location so skip.

```

5642   }%
5643   {%
5644     \@gls@noidxloclist@sep
5645     \@gls@noidxloclist@prev
5646     \def\@gls@noidxloclist@prev{#1}%
5647   }%
5648 }

```

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

Display a location in the location list.

```

5649 \newcommand*\glsnoidxdisplayloc[4]{%
5650   \setentrycounter[#1]{#2}%
5651   \csuse{#3}{#4}%
5652 }

```

`\@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.

```
5653 \newcommand*{\@gls@reference}[3]{%
```

Add to label list

```
5654 \glsdoifexistsorwarn{#2}%
5655 {%
5656 \ifcsundef{@glsref@#1}{\csgdef{@glsref@#1}{}}{}%
5657 \ifinlistcs{#2}{@glsref@#1}%
5658 {}%
5659 {\listcsgadd{@glsref@#1}{#2}}%
```

Add to location list

```
5660 \ifcsundef{glo@glstdetoklabel{#2}@loclist}%
5661 {\csgdef{glo@glstdetoklabel{#2}@loclist}{}}%
5662 {}%
5663 \listcsgadd{glo@glstdetoklabel{#2}@loclist}{#3}%
5664 }%
5665 }
```

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

```
5666 \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`.

```
5667 \define@key{printgloss}{title}{%
5668 \def@glossarytitle{#1}%
5669 \let@gls@dotoc@title\relax
5670 }
```

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

```
5671 \define@key{printgloss}{toctitle}{%
5672 \def@glossarytoctitle{#1}%
5673 \let@gls@dotoc@title\relax
5674 }
```

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

```
5675 \define@key{printgloss}{style}{%
5676 \ifcsundef{@glsstyle@#1}%
5677 {%
5678 \PackageError{glossaries}%
5679 {Glossary style ‘#1’ undefined}{}%
5680 }%
5681 {%
5682 \def@glossarystyle{\setglossentrycompatibility
5683 \csname @glsstyle@#1\endcsname}%
5684 }%
5685 }
```

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

```

5686 \define@choicekey{printgloss}{numberedsection}[\val\nr]{%
5687 false,nolabel,autolabel,nameref}[nolabel]{%
5688   \ifcase\nr\relax
5689     \renewcommand*{\@@glossarysecstar}{*}%
5690     \renewcommand*{\@@glossaryseclabel}{}%
5691   \or
5692     \renewcommand*{\@@glossarysecstar}{}%
5693     \renewcommand*{\@@glossaryseclabel}{}%
5694   \or
5695     \renewcommand*{\@@glossarysecstar}{}%
5696     \renewcommand*{\@@glossaryseclabel}{\label{\glsautoprefix\@glo@type}}%
5697   \or
5698     \renewcommand*{\@@glossarysecstar}{*}%
5699     \renewcommand*{\@@glossaryseclabel}{%
5700       \protected@edef\@currentlabelname{\glossarytoctitle}%
5701       \label{\glsautoprefix\@glo@type}}%
5702   \fi
5703 }
```

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

```

5704 \define@choicekey{printgloss}{nogroupskip}{true,false}[true]{%
5705   \csuse{glsnogroupskip#1}%
5706 }
```

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

```

5707 \define@choicekey{printgloss}{nopostdot}{true,false}[true]{%
5708   \csuse{glsnopostdot#1}%
5709 }
```

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

```

5710 \define@choicekey{printgloss}{entrycounter}{true,false}[true]{%
5711   \csuse{glsentrycounter#1}%
5712   \ifglsentrycounter
5713     \ifx\@gls@counterwithin\@empty
5714       \newcounter{glossaryentry}%
5715     \else
5716       \newcounter{glossaryentry}[\@gls@counterwithin]%
5717     \fi
5718     \def\theHglossaryentry{\currentglossary.\theglossaryentry}%
5719     \renewcommand*{\glsresetentrycounter}{%
5720       \setcounter{glossaryentry}{0}%
5721     }%
5722     \renewcommand*{\glsstepentry}[1]{%
5723       \refstepcounter{glossaryentry}%
5724       \label{glsentry-\glsdetoklabel{##1}}%
5725     }%
5726     \renewcommand*{\glsentrycounterlabel}{\theglossaryentry.\space}%
5727     \renewcommand*{\glsentryitem}[1]{%
```

```

5728     \glsstepentry{##1}\glsentrycounterlabel
5729   }%
5730 \else
5731   \renewcommand*{\glsresetentrycounter}{}%
5732   \renewcommand*{\glsstepentry}[1]{}%
5733   \renewcommand*{\glsentrycounterlabel}{}%
5734   \renewcommand*{\glsentryitem}[1]{\glsresetsubentrycounter}
5735 \fi
5736 }

```

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.

```

5737 \define@choicekey{printgloss}{subentrycounter}{true,false}[true]{%
5738   \csuse{glssubentrycounter#1}%
5739   \ifglssubentrycounter
5740     \ifundef\c@glossarysubentry
5741     {%
5742       \ifglsentrycounter
5743         \newcounter{glossarysubentry}[glossaryentry]%
5744       \else
5745         \newcounter{glossarysubentry}
5746       \fi
5747     }{}%
5748     \renewcommand*{\glsstepsubentry}[1]{%
5749       \edef\currentglssubentry{\glsdetoklabel{##1}}%
5750       \refstepcounter{glossarysubentry}%
5751       \label{glsentry-\currentglssubentry}%
5752     }%
5753     \renewcommand*{\glsresetsubentrycounter}{%
5754       \setcounter{glossarysubentry}{0}%
5755     }%
5756     \renewcommand*{\glssubentryitem}[1]{%
5757       \glsstepsubentry{##1}\glssubentrycounterlabel
5758     }%
5759     \renewcommand*{\glssubentrycounterlabel}{\theglossarysubentry}\space}%
5760     \def\theHglossarysubentry{\currentglssubentry.\theglossarysubentry}
5761   \else
5762     \renewcommand*{\glssubentryitem}[1]{}%
5763     \renewcommand*{\glsstepsubentry}[1]{}%
5764     \renewcommand*{\glsresetsubentrycounter}{}%
5765     \renewcommand*{\glssubentrycounterlabel}{}%
5766   \fi
5767 }

```

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

```

5768 \define@boolkey{printgloss}[gls]{nonnumberlist}[true]{%
5769 \ifglslsnonnumberlist
5770   \def\glossaryentrynumbers##1{%
5771 \else

```



```

5772 \def\glossaryentrynumbers##1{##1}%
5773 \fi}

```

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

```

5774 \define@key{printgloss}{sort}{\@glo@assign@sortkey{#1}}

```

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

```

5775 \newcommand*{\@glo@no@assign@sortkey}[1]{%
5776 \PackageError{glossaries}{‘sort’ key not permitted with
5777 \string\printglossary}%
5778 {The ‘sort’ key may only be used with \string\printnoidxglossary}%
5779 }

```

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

```

5780 \newcommand*{\@glo@assign@sortkey}[1]{%
5781 \def\@glo@sorttype{#1}%
5782 }

```

`@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.

```

5783 \newcommand*{\@glsnonextpages}{%
5784 \gdef\glossaryentrynumbers##1{%
5785 \glsresetentrylist
5786 }%
5787 }

```

`\@glsnextpages` Activate the next number list only. Global assignments required as it may not occur in the same level of grouping as the next numberlist. (For example, if `\glsnextpages` is placed in the entry’s description and 3 column tabular style glossary is used.) `\org@glossaryentrynumbers` needs to be set at the start of each glossary, in the event that `\glossaryentrynumber` is re-defined.

```

5788 \newcommand*{\@glsnextpages}{%
5789 \gdef\glossaryentrynumbers##1{%
5790 ##1\glsresetentrylist}}

```

`sresetentrylist` Resets `\glossaryentrynumbers`

```

5791 \newcommand*{\glsresetentrylist}{%
5792 \global\let\glossaryentrynumbers\org@glossaryentrynumbers}

```

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

```

5793 \newcommand*{\glsnonextpages}{}

```

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

```

5794 \newcommand*{\glsnextpages}{}

```

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

```

5795 \ifglentrycounter
5796   \ifx\@gls@counterwithin\@empty
5797     \newcounter{glossaryentry}
5798   \else
5799     \newcounter{glossaryentry}[\@gls@counterwithin]
5800   \fi
5801   \def\theHglossaryentry{\currentglossary.\theglossaryentry}
5802 \fi

```

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

```

5803 \ifglssubentrycounter
5804   \ifglentrycounter
5805     \newcounter{glossarysubentry}[glossaryentry]
5806   \else
5807     \newcounter{glossarysubentry}
5808   \fi
5809   \def\theHglossarysubentry{\currentglssubentry.\theglossarysubentry}
5810 \fi

```

subentrycounter Resets the `glossarysubentry` counter.

```

5811 \ifglssubentrycounter
5812   \newcommand*{\glsresetsubentrycounter}{%
5813     \setcounter{glossarysubentry}{0}%
5814   }
5815 \else
5816   \newcommand*{\glsresetsubentrycounter}{}
5817 \fi

```

subentrycounter Resets the `glossaryentry` counter.

```

5818 \ifglentrycounter
5819   \newcommand*{\glsresetentrycounter}{%
5820     \setcounter{glossaryentry}{0}%
5821   }
5822 \else
5823   \newcommand*{\glsresetentrycounter}{}
5824 \fi

```

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

```

5825 \ifglentrycounter
5826   \newcommand*{\glsstepentry}[1]{%
5827     \refstepcounter{glossaryentry}%
5828     \label{glsentry-\glsdetoklabel{#1}}%
5829   }
5830 \else

```

```

5831 \newcommand*{\glsstepentry}[1]{%
5832 \fi

```

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

```

5833 \ifglssubentrycounter
5834 \newcommand*{\glsstepsubentry}[1]{%
5835 \edef\currentglssubentry{\glsdetoklabel{#1}}%
5836 \refstepcounter{glossarysubentry}%
5837 \label{glentry-\currentglssubentry}%
5838 }
5839 \else
5840 \newcommand*{\glsstepsubentry}[1]{%
5841 \fi

```

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

```

5842 \ifglentrycounter
5843 \newcommand*{\glsrefentry}[1]{\ref{glentry-\glsdetoklabel{#1}}}
5844 \else
5845 \ifglssubentrycounter
5846 \newcommand*{\glsrefentry}[1]{\ref{glentry-\glsdetoklabel{#1}}}
5847 \else
5848 \newcommand*{\glsrefentry}[1]{\gls{#1}}
5849 \fi
5850 \fi

```

`\glsentrycounterlabel` Defines how to display the glossaryentry counter.

```

5851 \ifglentrycounter
5852 \newcommand*{\glsentrycounterlabel}{\theglossaryentry.\space}
5853 \else
5854 \newcommand*{\glsentrycounterlabel}{}
5855 \fi

```

`\glsentrysubentrycounterlabel` Defines how to display the glossarysubentry counter.

```

5856 \ifglssubentrycounter
5857 \newcommand*{\glsentrysubentrycounterlabel}{\theglossarysubentry)\space}
5858 \else
5859 \newcommand*{\glsentrysubentrycounterlabel}{}
5860 \fi

```

`\glsentryitem` Step and display glossaryentry counter, if appropriate.

```

5861 \ifglentrycounter
5862 \newcommand*{\glsentryitem}[1]{%
5863 \glsstepentry{#1}\glsentrycounterlabel
5864 }
5865 \else
5866 \newcommand*{\glsentryitem}[1]{\glsresetsubentrycounter}
5867 \fi

```

`glssubentryitem` Step and display `glossarysubentry` counter, if appropriate.

```
5868 \ifglssubentrycounter
5869   \newcommand*{\glssubentryitem}[1]{%
5870     \glssubentrycounterlabel
5871   }
5872 \else
5873   \newcommand*{\glssubentryitem}[1]{}
5874 \fi
```

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

```
5875 \ifcsundef{theglossary}%
5876 {%
5877   \newenvironment{theglossary}{}{}%
5878 }%
5879 {%
5880   \@gls@warnontheglossdefined
5881   \renewenvironment{theglossary}{}{}%
5882 }
```

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`

```
5883 \newcommand*{\glossaryheader}{}%
```

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

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

```
5884 \newcommand*{\glstarget}[2]{\@glstarget{\glo@linkprefix#1}{#2}}
```

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

`atibleglossentry`

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

```
5885 \providecommand*{\compatibleglossentry}[2]{%
5886   \toks@{#2}%
5887   \protected@edef\do@glossentry{\noexpand\glossaryentryfield{#1}%
5888     {\noexpand\glsnamefont
5889       {\expandafter\expandonce\csname glo@#1@name\endcsname}}}%
5889 }
```

```

5890     {\expandafter\expandonce\csname glo@#1@desc\endcsname}%
5891     {\expandafter\expandonce\csname glo@#1@symbol\endcsname}%
5892     {\the\toks@}%
5893   }%
5894   \@do@glossentry
5895 }

```

\glossentryname

```

5896 \newcommand*{\glossentryname}[1]{%
5897   \glsdoifexistsorwarn{#1}%
5898   {%
5899     \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
5900     \expandafter\glsnamefont\expandafter{\glo@name}%
5901   }%
5902 }

```

\Glossentryname

```

5903 \newcommand*{\Glossentryname}[1]{%
5904   \glsdoifexistsorwarn{#1}%
5905   {%
5906     \glsnamefont{\Glsentryname{#1}}%
5907   }%
5908 }

```

\glossentrydesc

```

5909 \newcommand*{\glossentrydesc}[1]{%
5910   \glsdoifexistsorwarn{#1}%
5911   {%
5912     \glsentrydesc{#1}%
5913   }%
5914 }

```

\Glossentrydesc

```

5915 \newcommand*{\Glossentrydesc}[1]{%
5916   \glsdoifexistsorwarn{#1}%
5917   {%
5918     \Glsentrydesc{#1}%
5919   }%
5920 }

```

\glossentrysymbol

```

5921 \newcommand*{\glossentrysymbol}[1]{%
5922   \glsdoifexistsorwarn{#1}%
5923   {%
5924     \glsentrysymbol{#1}%
5925   }%
5926 }

```

lossentrysymbol

```
5927 \newcommand*\Glossentrysymbol}[1]{%
5928   \glsdoifexistsorwarn{#1}%
5929   {%
5930     \Glsentrysymbol{#1}%
5931   }%
5932 }
```

blesubglossentry

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

```
5933 \providecommand*\compatiblesubglossentry}[3]{%
5934   \toks@{#3}%
5935   \protected@edef\@do@subglossentry{\noexpand\glossarysubentryfield{\number#1}%
5936     {#2}%
5937     {\noexpand\glsnamefont
5938       {\expandafter\expandonce\csname glo@#2@name\endcsname}}%
5939     {\expandafter\expandonce\csname glo@#2@desc\endcsname}%
5940     {\expandafter\expandonce\csname glo@#2@symbol\endcsname}%
5941     {\the\toks@}}%
5942   }%
5943   \@do@subglossentry
5944 }
```

rycompatibility

```
5945 \newcommand*\setglossentrycompatibility{%
5946   \let\glossentry\compatibleglossentry
5947   \let\subglossentry\compatiblesubglossentry
5948 }
5949 \setglossentrycompatibility
```

ossaryentryfield

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

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

```
5950 \newcommand{\glossaryentryfield}[5]{%
5951   \GlossariesWarning
5952   {Deprecated use of \string\glossaryentryfield.^^J
5953     I recommend you change to \string\glossentry.^^J
5954     If you've just upgraded, try removing your gls auxiliary
5955     files^^J and recompile}%
5956   \noindent\textbf{\glstarget{#1}{#2}} #4 #3. #5\par}
```

arysubentryfield

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

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

```
5957 \newcommand*{\glossarysubentryfield}[6]{%
5958   \GlossariesWarning
5959   {Deprecated use of \string\glossarysubentryfield.^^J
5960    I recommend you change to \string\subglossentry.^^J
5961    If you've just upgraded, try removing your gls auxiliary
5962    files^^J and recompile}%
5963   \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`

```
5964 \newcommand*{\glsgroupskip}{}%
```

Each of the 28 glossary groups described above is preceded by a group heading. This is formatted by the command `\glsgroupheading` which takes one argument which is the *label* assigned to that group (not the title). The corresponding labels are: `glssymbols`, `glsnumbers`, A, ..., Z. Glossary styles must redefine this command. (In between groups, `\glsgroupheading` comes immediately after `\glsgroupskip`.)

`\glsgroupheading`

```
5965 \newcommand*{\glsgroupheading}[1]{}
```

It is possible to “trick” `makeindex` into treating entries as though they belong to the same group, even if the terms don’t start with the same letter, by modifying the sort key. For example, all entries belonging to one group could be defined so that the sort key starts with an a, while entries belonging to another group could be defined so that the sort key starts with a b, and so on. If you want each group to have a heading, you would then need to modify the translation control sequences `\glsgetgrouptitle` and `\glsgetgrouplabel` so that the label is translated into the required title (and vice-versa).

`\glsgetgrouptitle{<label>}`

This command produces the title for the glossary group whose label is given by *<label>*. By default, the group labelled `glssymbols` produces `\glssymbolsgroupname`, the group labelled `glsnumbers` produces `\glsnumbersgroupname` and all the other groups simply produce their label. As mentioned above, the group labels are: `glssymbols`, `glsnumbers`, A, ..., Z. If you want to redefine the group titles, you will need to redefine this command. Languages

other than English may produce labels that are non-expandable, so we need to check for that otherwise it will create a “missing \endcsname inserted” error.

lsgetgrouptitle

```
5966 \newcommand*{\glsgetgrouptitle}[1]{%
5967   \@gls@getgrouptitle{#1}{\@gls@grptitle}%
5968   \@gls@grptitle
5969 }
```

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

```
5970 \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.

```
5971   \dtl@ifsingle{#1}%
5972   {%
5973     \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
5974   }%
5975   {%
5976     \ifboolexpr{test{\ifstrequal{#1}{glssymbols}}
5977                or test{\ifstrequal{#1}{glsnumbers}}}%
5978   {%
5979     \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
5980   }%
5981   {%
5982     \def#2{#1}%
5983   }%
5984 }%
5985 }
```

othergrouptitle Version for the no-indexing app option:

```
5986 \newcommand*{\@gls@noidx@getgrouptitle}[2]{%
5987   \DTLifint{#1}%
5988   {\edef#2{\char#1\relax}}%
5989   {%
5990     \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
5991   }%
5992 }
```

`\glsgetgrouplabel{<title>}`

This command does the reverse to the previous command. The argument is the group title, and it produces the group label. Note that if you redefine \glsgetgrouptitle, you will also need to redefine \glsgetgrouplabel.

lsgetgrouplabel

```
5993 \newcommand*{\glsgetgrouplabel}[1]{%
```



```

5994 \ifthenelse{\equal{#1}{\glssymbolsgroupname}}{\glssymbols}{%
5995 \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`

```

5996 \newcommand*{\setentrycounter}[2][]{%
5997   \def\@glo@counterprefix{#1}%
5998   \ifx\@glo@counterprefix\empty
5999     \def\@glo@counterprefix{.}%
6000   \else
6001     \def\@glo@counterprefix{.#1.}%
6002   \fi
6003   \def\glsentrycounter{#2}%
6004 }

```

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

`\setglossarystyle`

```

6005 \newcommand*{\setglossarystyle}[1]{%
6006   \ifcsundef{@glsstyle@#1}%
6007   {%
6008     \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
6009   }%
6010   {%
6011     \csname @glsstyle@#1\endcsname
6012   }%

```

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

```

6013   \ifx\@glossary@default@style\relax
6014     \protected@edef\@glossary@default@style{#1}%
6015   \fi
6016 }

```

`\glossarystyle`

```

6017 \newcommand*{\glossarystyle}[1]{%
6018   \ifcsundef{@glsstyle@#1}%
6019   {%
6020     \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
6021   }%
6022   {%
6023     \GlossariesWarning
6024     {Deprecated command \string\glossarystyle.^~J
6025      I recommend you switch to \string\setglossarystyle\space unless
6026      you want to maintain backward compatibility}%
6027     \setglossentrycompatibility
6028     \csname @glsstyle@#1\endcsname

```

```

6029 \ifcsdef{@glscompstyle@#1}%
6030 {\setglossentrycompatibility\csuse{@glscompstyle@#1}}%
6031 {}%
6032 }%

```

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

```

6033 \ifx@glossary@default@style\relax
6034 \protected@edef@glossary@default@style{#1}%
6035 \fi
6036 }

```

`\newglossarystyle` New glossary styles can be defined using:

```
\newglossarystyle{<name>}{<definition>}
```

The *<definition>* argument should redefine `\theglossary`, `\glossaryheader`, `\glsgroupheading`, `\glossaryentryfield` and `\glsgroupskip` (see [section 1.19](#) for the definitions of predefined styles). Glossary styles should not redefine `\glossarypreamble` and `\glossarypostamble`, as the user should be able to switch between styles without affecting the pre- and postambles.

```

6037 \newcommand{\newglossarystyle}[2]{%
6038 \ifcsundef{@glsstyle@#1}%
6039 {%
6040 \expandafter\def\csname @glsstyle@#1\endcsname{#2}%
6041 }%
6042 {%
6043 \PackageError{glossaries}{Glossary style ‘#1’ is already defined}{}%
6044 }%
6045 }

```

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

```

6046 \newcommand{\renewglossarystyle}[2]{%
6047 \ifcsundef{@glsstyle@#1}%
6048 {%
6049 \PackageError{glossaries}{Glossary style ‘#1’ isn’t already defined}{}%
6050 }%
6051 {%
6052 \csdef{@glsstyle@#1}{#2}%
6053 }%
6054 }

```

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`

```
6055 \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`

```
6056 \ifcsundef{hyperlink}%
6057 {%
6058   \def\glshypernumber#1{#1}%
6059 }%
6060 {%
6061   \def\glshypernumber#1{\@glshypernumber#1\nohyperpage{}}\@nil}
6062 }
```

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

```
6063 \def\@glshypernumber#1\nohyperpage#2#3\@nil{%
6064   \ifx\#1\%
6065     \else
6066       \@delimR#1\delimR\delimR\%
6067     \fi
6068     \ifx\#2\%
6069       \else
6070         #2%
6071       \fi
6072     \ifx\#3\%
6073       \else
6074         \@glshypernumber#3\@nil
6075       \fi
6076 }
```

`\@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`

```
6077 \def\@delimR#1\delimR #2\delimR #3\%
6078 \ifx\#2\%
6079   \@delimN{#1}%

```

```

6080 \else
6081   \@gls@numberlink{#1}\delimR\@gls@numberlink{#2}%
6082 \fi}

```

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

\@delimN

```

6083 \def\@delimN#1{\@delimN#1\delimN \delimN\}
6084 \def\@delimN#1\delimN #2\delimN#3\{\{%
6085 \ifx\#3\}%
6086   \@gls@numberlink{#1}%
6087 \else
6088   \@gls@numberlink{#1}\delimN\@gls@numberlink{#2}%
6089 \fi
6090 }

```

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

```

6091 \def\@gls@numberlink#1{%
6092 \begingroup
6093 \toks@={}%
6094 \@gls@removespaces#1 \@nil
6095 \endgroup}

6096 \def\@gls@removespaces#1 #2\@nil{%
6097 \toks@=\expandafter{\the\toks@#1}%
6098 \ifx\#2\}%
6099 \edef\x{\the\toks@}%
6100 \ifx\x\empty
6101 \else

6102   \hyperlink{\glsentrycounter\@glo@counterprefix\the\toks@}%
6103     {\the\toks@}%
6104 \fi
6105 \else
6106   \@gls@ReturnAfterFi{%
6107     \@gls@removespaces#2\@nil
6108   }%
6109 \fi
6110 }
6111 \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

```

6112 \newcommand*\hyperrm[1]{\textrm{\glshypernumber{#1}}}

```

\hypersf

```

6113 \newcommand*\hypersf[1]{\textsf{\glshypernumber{#1}}}

```

```

\hypertt
6114 \newcommand*{\hypertt}[1]{\texttt{\glshypernumber{#1}}}

\hyperbf
6115 \newcommand*{\hyperbf}[1]{\textbf{\glshypernumber{#1}}}

\hypermd
6116 \newcommand*{\hypermd}[1]{\textmd{\glshypernumber{#1}}}

\hyperit
6117 \newcommand*{\hyperit}[1]{\textit{\glshypernumber{#1}}}

\hypersl
6118 \newcommand*{\hypersl}[1]{\textsl{\glshypernumber{#1}}}

\hyperup
6119 \newcommand*{\hyperup}[1]{\textup{\glshypernumber{#1}}}

\hypersc
6120 \newcommand*{\hypersc}[1]{\textsc{\glshypernumber{#1}}}

\hyperemph
6121 \newcommand*{\hyperemph}[1]{\emph{\glshypernumber{#1}}}

```

1.17 Acronyms

```

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

```

This emulates the way the old package defined acronyms. It is equivalent to `\newacronym[⟨key-val list⟩]{⟨label⟩}{⟨abbrv⟩}{⟨long⟩}` and it additionally defines the command `\⟨label⟩` which is equivalent to `\gls{⟨label⟩}` (thus `⟨label⟩` must only contain alphabetical characters). If `⟨label⟩` is omitted, `⟨abbrv⟩` is used. This only emulates the syntax of the old package. The way the acronyms appear in the list of acronyms is determined by the definition of `\newacronym` and the glossary style.

Note that `\⟨label⟩` can't have an optional argument if the package is loaded. If hasn't been loaded then you can do `\⟨label⟩[⟨insert⟩]` but you can't do `\⟨label⟩[⟨key-val list⟩]`. For example if you define the acronym `svm`, then you can do `\svm['s]` but you can't do `\svm[format=textbf]`. If the package is loaded, `\svm['s]` will appear as `svm ['s]` which is unlikely to be the desired result. In this case, you will need to use `\gls` explicitly, e.g. `\gls{svm}['s]`. Note that it is up to the user to load if desired.

```

6122 \newcommand{\oldacronym}[4][\gls@label]{%
6123   \def\gls@label{#2}%
6124   \newacronym[#4]{#1}{#2}{#3}%
6125   \ifcsundef{xspace}%

```

```

6126 {%
6127   \expandafter\edef\csname#1\endcsname{%
6128     \noexpand@ifstar{\noexpand\Gls{#1}}{\noexpand\gls{#1}}}%
6129   }%
6130 }%
6131 {%
6132   \expandafter\edef\csname#1\endcsname{%
6133     \noexpand@ifstar{\noexpand\Gls{#1}\noexpand\xspace}{%
6134       \noexpand\gls{#1}\noexpand\xspace}%
6135     }%
6136   }%
6137 }

```

```
\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`

```
6138 \newcommand{\newacronym}[4][{}]{}
```

Set up some convenient short cuts. These need to be changed if `\newacronym` is changed (or if the description key is changed).

`\acrpluralsuffix` Plural suffix used by `\newacronym`. This just defaults to `\glspluralsuffix` but is changed to include `\textup` if the `smallcaps` option is used, so that the suffix doesn't appear in small caps as it doesn't look right. For example, `ABCS` looks as though the “s” is part of the acronym, but `ABCS` looks as though the “s” is a plural suffix. Since the entire text `abcs` is set in `\textsc`, `\textup` is need to cancel it out.

```
6139 \newcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}
```

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

`\glstextup`

```
6140 \newrobustcmd*{\glstextup}[1]{\ifdef\textulc{\textulc{#1}}{\textup{#1}}}
```

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

`\glsshortkey`

```
6141 \newcommand*{\glsshortkey}{short}
```

`\sshortpluralkey`

```
6142 \newcommand*{\glsshortpluralkey}{shortplural}
```

```

\glslongkey
6143 \newcommand*{\glslongkey}{long}

\glslongpluralkey
6144 \newcommand*{\glslongpluralkey}{longplural}

\acrfull  Full form of the acronym.
6145 \newrobustcmd*{\acrfull}{\@gls@hyp@opt\@ns@acrfull}

6146 \newcommand*\ns@acrfull[2] [] {%
6147   \new@ifnextchar[{\@acrfull{#1}{#2}}%
6148     {\@acrfull{#1}{#2} []}%
6149 }

\@acrfull  Low-level macro:
6150 \def\@acrfull#1#2[#3] {%
  Make it easier for acronym styles to change this:
6151   \acrfullfmt{#1}{#2}{#3}%
6152 }

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

\acrfullfmt  No case change full format.
6153 \newcommand*{\acrfullfmt}[3] {%
6154   \acrlinkfullformat{\@acrlong}{\@acrshort}{#1}{#2}{#3}%
6155 }

\acrlinkfullformat  Format for full links like \acrfull. Syntax: \acrlinkfullformat{<long cs>}{<short cs>}{<options>}{<label>}{<in
6156 \newcommand{\acrlinkfullformat}[5] {%
6157   \acrfullformat{#1}{#3}{#4}[#5]{#2}{#3}{#4} []}%
6158 }

\acrfullformat  Default full form is <long> (<short>).
6159 \newcommand{\acrfullformat}[2] {#1\glsspace(#2)}

\glsspace  Robust space to ensure it's written to the .glsdefs file.
6160 \newrobustcmd{\glsspace}{\space}

  Default format for full acronym

\Acrfull
6161 \newrobustcmd*{\Acrfull}{\@gls@hyp@opt\@ns@Acrfull}

6162 \newcommand*\ns@Acrfull[2] [] {%
6163   \new@ifnextchar[{\@Acrfull{#1}{#2}}%
6164     {\@Acrfull{#1}{#2} []}%
6165 }

```

Low-level macro:

```
6166 \def\@Acrfull#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6167 \Acrfullfmt{#1}{#2}{#3}%
6168 }
```

`\Acrfullfmt` First letter upper case full format.

```
6169 \newcommand*\@Acrfullfmt[3]{%
6170 \acrlinkfullformat{\@Acrlong}{\@acrshort}{#1}{#2}{#3}%
6171 }
```

`\ACRfull`

```
6172 \newrobustcmd*\@ACRfull{\@gls@hyp@opt\@ns@ACRfull}

6173 \newcommand*\@ns@ACRfull[2][{}]{%
6174 \new@ifnextchar[{\@ACRfull{#1}{#2}}{%
6175 {\@ACRfull{#1}{#2}[]}%
6176 }
```

Low-level macro:

```
6177 \def\@ACRfull#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6178 \ACRfullfmt{#1}{#2}{#3}%
6179 }
```

`\ACRfullfmt` All upper case full format.

```
6180 \newcommand*\@ACRfullfmt[3]{%
6181 \acrlinkfullformat{\@ACRlong}{\@ACRshort}{#1}{#2}{#3}%
6182 }
```

Plural:

`\acrfullpl`

```
6183 \newrobustcmd*\@acrfullpl{\@gls@hyp@opt\@ns@acrfullpl}

6184 \newcommand*\@ns@acrfullpl[2][{}]{%
6185 \new@ifnextchar[{\@acrfullpl{#1}{#2}}{%
6186 {\@acrfullpl{#1}{#2}[]}%
6187 }
```

Low-level macro:

```
6188 \def\@acrfullpl#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6189 \acrfullplfmt{#1}{#2}{#3}%
6190 }
```



```
\acrfullplfmt  No case change plural full format.
6191 \newcommand*{\acrfullplfmt}[3]{%
6192   \acrlinkfullformat{\@acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%
6193 }
```

```
\Acrfullpl
6194 \newrobustcmd*{\Acrfullpl}{\@gls@hyp@opt\ns@Acrfullpl}

6195 \newcommand*\ns@Acrfullpl[2][{}]{%
6196   \new@ifnextchar[{\@Acrfullpl{#1}{#2}}{%
6197     {\@Acrfullpl{#1}{#2}[]}%
6198 }
```

Low-level macro:

```
6199 \def\@Acrfullpl#1#2[#3]{%
    Make it easier for acronym styles to change this:
6200   \Acrfullplfmt{#1}{#2}{#3}%
6201 }
```

```
\Acrfullplfmt  First letter upper case plural full format.
6202 \newcommand*{\Acrfullplfmt}[3]{%
6203   \acrlinkfullformat{\@Acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%
6204 }
```

```
\ACRfullpl
6205 \newrobustcmd*{\ACRfullpl}{\@gls@hyp@opt\ns@ACRfullpl}

6206 \newcommand*\ns@ACRfullpl[2][{}]{%
6207   \new@ifnextchar[{\@ACRfullpl{#1}{#2}}{%
6208     {\@ACRfullpl{#1}{#2}[]}%
6209 }
```

Low-level macro:

```
6210 \def\@ACRfullpl#1#2[#3]{%
    Make it easier for acronym styles to change this:
6211   \ACRfullplfmt{#1}{#2}{#3}%
6212 }
```

```
\ACRfullplfmt  All upper case plural full format.
6213 \newcommand*{\ACRfullplfmt}[3]{%
6214   \acrlinkfullformat{\@ACRlongpl}{\@ACRshortpl}{#1}{#2}{#3}%
6215 }
```

1.18 Predefined acronym styles

`\acronymfont` This is only used with the additional acronym styles:
6216 `\newcommand{\acronymfont}[1]{#1}`

`\firstacronymfont` This is only used with the additional acronym styles:
6217 `\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.
6218 `\newcommand*{\acrnameformat}[2]{\acronymfont{#1}}`

Define some tokens used by `\newacronym`:

`\glskeylisttok`
6219 `\newtoks\glskeylisttok`

`\glslabeltok`
6220 `\newtoks\glslabeltok`

`\glsshorttok`
6221 `\newtoks\glsshorttok`

`\glslongtok`
6222 `\newtoks\glslongtok`

`\newacronymhook` Provide a hook for `\newacronym`:
6223 `\newcommand*{\newacronymhook}{}`

`\genericNewAcronym` New improved version of setting the acronym style.
6224 `\newcommand*{\SetGenericNewAcronym}{%`
Change the behaviour of `\Glsentryname` to workaround expansion issues that cause a problem for `\makefirstuc`
6225 `\let\@Gls@entryname\@Gls@acrentryname`
Change the way acronyms are defined:
6226 `\renewcommand{\newacronym}[4][\%`
6227 `\ifdefempty{\@glsacronymlists}%`
6228 `{%`
6229 `\def\@glo@type{\acronymtype}%`
6230 `\setkeys{glossentry}{##1}%`
6231 `\DeclareAcronymList{\@glo@type}%`
6232 `}%`
6233 `{}%`
6234 `\glskeylisttok{##1}%`
6235 `\glslabeltok{##2}%`
6236 `\glsshorttok{##3}%`
6237 `\glslongtok{##4}%`

```

6238 \newacronymhook
6239 \protected@edef\@do@newglossaryentry{%
6240 \noexpand\newglossaryentry{\the\glslabeltok}%
6241 {%
6242 type=\acronymtype,%
6243 name={\expandonce{\acronymentry{##2}}},%
6244 sort={\acronymsort{\the\glsshorttok}{\the\glslongtok}},%
6245 text={\the\glsshorttok},%
6246 short={\the\glsshorttok},%
6247 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6248 long={\the\glslongtok},%
6249 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6250 \GenericAcronymFields,%
6251 \the\glskeylisttok
6252 }%
6253 }%
6254 \@do@newglossaryentry
6255 }%

```

Make sure that \acrfull etc reflects the new style:

```

6256 \renewcommand*{\acrfullfmt}[3]{%
6257 \glslink[##1]{##2}{\genacrfullformat{##2}{##3}}}%
6258 \renewcommand*{\Acrfullfmt}[3]{%
6259 \glslink[##1]{##2}{\Genacrfullformat{##2}{##3}}}%
6260 \renewcommand*{\ACRfullfmt}[3]{%
6261 \glslink[##1]{##2}{%
6262 \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}}%
6263 \renewcommand*{\acrfullplfmt}[3]{%
6264 \glslink[##1]{##2}{\genplacrfullformat{##2}{##3}}}%
6265 \renewcommand*{\Acrfullplfmt}[3]{%
6266 \glslink[##1]{##2}{\Genplacrfullformat{##2}{##3}}}%
6267 \renewcommand*{\ACRfullplfmt}[3]{%
6268 \glslink[##1]{##2}{%
6269 \mfirstucMakeUppercase{\genplacrfullformat{##2}{##3}}}}%

```

Make sure that \glsentryfull etc reflects the new style:

```

6270 \renewcommand*{\glsentryfull}[1]{\genacrfullformat{##1}{}}%
6271 \renewcommand*{\Glsentryfull}[1]{\Genacrfullformat{##1}{}}%
6272 \renewcommand*{\glsentryfullpl}[1]{\genplacrfullformat{##1}{}}%
6273 \renewcommand*{\Glsentryfullpl}[1]{\Genplacrfullformat{##1}{}}%
6274 }

```

`\icAcronymFields` Fields used by \SetGenericNewAcronym that can be changed by the acronym style.

```

6275 \newcommand*{\GenericAcronymFields}{description={\the\glslongtok}}

```

`\acronymentry` `\acronymentry{<label>}`

Display style for the name field in the list of acronyms.

```

6276 \newcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{#1}}}

```

`\acronymsort` `\acronymsort{<short>}{<long>}`

Default sort format for acronyms.

```
6277 \newcommand*{\acronymsort}[2]{#1}
```

`\setacronymstyle` `\setacronymstyle{<style name>}`

```
6278 \newcommand*{\setacronymstyle}[1]{%
6279   \ifcsundef{@glsacr@dispstyle@#1}
6280   {%
6281     \PackageError{glossaries}{Undefined acronym style ‘#1’}{}%
6282   }%
6283   {%
6284     \ifdefempty{\@glsacronymlists}%
6285     {%
6286       \DeclareAcronymList{\acronymtype}%
6287     }%
6288     {}%
6289     \SetGenericNewAcronym
6290     \GlsUseAcrStyleDefs{#1}%
6291     \@for\@gls@type:=\@glsacronymlists\do{%
6292       \defglentryfmt[\@gls@type]{\GlsUseAcrEntryDispStyle{#1}}%
6293     }%
6294   }%
6295 }
```

`\newacronymstyle` `\newacronymstyle{<style name>}{<entry format definition>}{<display definitions>}`

Defines a new acronym style called *<style name>*.

```
6296 \newcommand*{\newacronymstyle}[3]{%
6297   \ifcsdef{@glsacr@dispstyle@#1}%
6298   {%
6299     \PackageError{glossaries}{Acronym style ‘#1’ already exists}{}%
6300   }%
6301   {%
6302     \csdef{@glsacr@dispstyle@#1}{#2}%
6303     \csdef{@glsacr@styledefs@#1}{#3}%
6304   }%
6305 }
```

`\renewacronymstyle` Redefines the given acronym style.

```
6306 \newcommand*{\renewacronymstyle}[3]{%
6307   \ifcsdef{@glsacr@dispstyle@#1}%
6308   {%
```

```

6309 \csdef{@glsacr@dispstyle@#1}{#2}%
6310 \csdef{@glsacr@styledefs@#1}{#3}%
6311 }%
6312 {%
6313 \PackageError{glossaries}{Acronym style ‘#1’ doesn’t exist}{}%
6314 }%
6315 }

```

rEntryDispStyle

```

6316 \newcommand*{\GlsUseAcrEntryDispStyle}[1]{\csuse{@glsacr@dispstyle@#1}}

```

UseAcrStyleDefs

```

6317 \newcommand*{\GlsUseAcrStyleDefs}[1]{\csuse{@glsacr@styledefs@#1}}

```

Predefined acronym styles:

long-short *<long>* (*<short>*) acronym style.

```

6318 \newacronymstyle{long-short}%
6319 {%

```

Check for long form in case this is a mixed glossary.

```

6320 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6321 }%
6322 {%
6323 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6324 \renewcommand*{\genacrfullformat}[2]{%
6325 \glentrylong{##1}##2\space
6326 (\protect\firstacronymfont{\glentryshort{##1}})%
6327 }%
6328 \renewcommand*{\Genacrfullformat}[2]{%
6329 \Glsentrylong{##1}##2\space
6330 (\protect\firstacronymfont{\glentryshort{##1}})%
6331 }%
6332 \renewcommand*{\genplacrfullformat}[2]{%
6333 \glentrylongpl{##1}##2\space
6334 (\protect\firstacronymfont{\glentryshortpl{##1}})%
6335 }%
6336 \renewcommand*{\Genplacrfullformat}[2]{%
6337 \Glsentrylongpl{##1}##2\space
6338 (\protect\firstacronymfont{\glentryshortpl{##1}})%
6339 }%
6340 \renewcommand*{\acronymentry}[1]{\acronymfont{\glentryshort{##1}}}%
6341 \renewcommand*{\acronymsort}[2]{##1}%
6342 \renewcommand*{\acronymfont}[1]{##1}%
6343 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
6344 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6345 }

```

long-sp-short Similar to the previous style but allows the space between the long and short form to be customized.

```

6346 \newacronymstyle{long-sp-short}%
6347 {%
    Check for long form in case this is a mixed glossary.
6348 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6349 }%
6350 {%
6351 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6352 \renewcommand*{\genacrfullformat}[2]{%
6353 \glentrylong{##1}##2\glsacspace{##1}%
6354 (\protect\firstacronymfont{\glentryshort{##1}})%
6355 }%
6356 \renewcommand*{\Genacrfullformat}[2]{%
6357 \Glsentrylong{##1}##2\glsacspace{##1}%
6358 (\protect\firstacronymfont{\glentryshort{##1}})%
6359 }%
6360 \renewcommand*{\genplacrfullformat}[2]{%
6361 \glentrylongpl{##1}##2\glsacspace{##1}%
6362 (\protect\firstacronymfont{\glentryshortpl{##1}})%
6363 }%
6364 \renewcommand*{\Genplacrfullformat}[2]{%
6365 \Glsentrylongpl{##1}##2\glsacspace{##1}%
6366 (\protect\firstacronymfont{\glentryshortpl{##1}})%
6367 }%
6368 \renewcommand*{\acronymentry}[1]{\acronymfont{\glentryshort{##1}}}%
6369 \renewcommand*{\acronymsort}[2]{##1}%
6370 \renewcommand*{\acronymfont}[1]{##1}%
6371 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
6372 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6373 }

```

`\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.

```

6374 \newcommand*{\glsacspace}[1]{%
6375 \settowidth{\dimen@}{(\firstacronymfont{\glentryshort{##1}})}%
6376 \ifdim\dimen@<3em~\else\space\fi
6377 }

```

`short-long` *(short)* (*long*) acronym style.

```

6378 \newacronymstyle{short-long}%
6379 {%
    Check for long form in case this is a mixed glossary.
6380 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6381 }%
6382 {%
6383 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6384 \renewcommand*{\genacrfullformat}[2]{%
6385 \protect\firstacronymfont{\glentryshort{##1}}##2\space
6386 (\glentrylong{##1})%

```

```

6387 }%
6388 \renewcommand*{\Genacrfullformat}[2]{%
6389   \protect\firstacronymfont{\Glsentryshort{##1}}##2\space
6390   (\glsentrylong{##1})%
6391 }%
6392 \renewcommand*{\genplacrfullformat}[2]{%
6393   \protect\firstacronymfont{\glsentryshortpl{##1}}##2\space
6394   (\glsentrylongpl{##1})%
6395 }%
6396 \renewcommand*{\Genplacrfullformat}[2]{%
6397   \protect\firstacronymfont{\Glsentryshortpl{##1}}##2\space
6398   (\glsentrylongpl{##1})%
6399 }%

6400 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6401 \renewcommand*{\acronymsort}[2]{##1}%
6402 \renewcommand*{\acronymfont}[1]{##1}%
6403 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
6404 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6405 }

```

long-sc-short *<long>* (\textsc{<short>}) acronym style.

```

6406 \newacronymstyle{long-sc-short}%
6407 {%
6408   \GlsUseAcrEntryDisplayStyle{long-short}%
6409 }%
6410 {%
6411   \GlsUseAcrStyleDefs{long-short}%
6412   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6413   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6414 }

```

long-sm-short *<long>* (\textsmaller{<short>}) acronym style.

```

6415 \newacronymstyle{long-sm-short}%
6416 {%
6417   \GlsUseAcrEntryDisplayStyle{long-short}%
6418 }%
6419 {%
6420   \GlsUseAcrStyleDefs{long-short}%
6421   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6422   \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6423 }

```

sc-short-long *<short>* (\textsc{<long>}) acronym style.

```

6424 \newacronymstyle{sc-short-long}%
6425 {%
6426   \GlsUseAcrEntryDisplayStyle{short-long}%
6427 }%
6428 {%

```

```

6429 \GlsUseAcrStyleDefs{short-long}%
6430 \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6431 \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6432 }

```

sm-short-long *<short>* (*\textsmaller{<long>}*) acronym style.

```

6433 \newacronymstyle{sm-short-long}%
6434 {%
6435   \GlsUseAcrEntryDispStyle{short-long}%
6436 }%
6437 {%
6438   \GlsUseAcrStyleDefs{short-long}%
6439   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6440   \renewcommand*{\acrpluralsuffix}{\glacrpluralsuffix}%
6441 }

```

long-short-desc *<long>* (*{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

6442 \newacronymstyle{long-short-desc}%
6443 {%
6444   \GlsUseAcrEntryDispStyle{long-short}%
6445 }%
6446 {%
6447   \GlsUseAcrStyleDefs{long-short}%
6448   \renewcommand*{\GenericAcronymFields}{}%
6449   \renewcommand*{\acronymsort}[2]{##2}%
6450   \renewcommand*{\acronymentry}[1]{%
6451     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6452 }

```

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`.

```

6453 \newacronymstyle{long-sp-short-desc}%
6454 {%
6455   \GlsUseAcrEntryDispStyle{long-sp-short}%
6456 }%
6457 {%
6458   \GlsUseAcrStyleDefs{long-sp-short}%
6459   \renewcommand*{\GenericAcronymFields}{}%
6460   \renewcommand*{\acronymsort}[2]{##2}%
6461   \renewcommand*{\acronymentry}[1]{%
6462     \glentrylong{##1}\glsacspace{##1}(\acronymfont{\glentryshort{##1}})}%
6463 }

```

g-sc-short-desc *<long>* (*\textsc{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

6464 \newacronymstyle{long-sc-short-desc}%
6465 {%

```



```

6466 \GlsUseAcrEntryDispStyle{long-sc-short}%
6467 }%
6468 {%
6469 \GlsUseAcrStyleDefs{long-sc-short}%
6470 \renewcommand*{\GenericAcronymFields}{}%
6471 \renewcommand*{\acronymsort}[2]{##2}%
6472 \renewcommand*{\acronymentry}[1]{%
6473     \glstrylong{##1}\space (\acronymfont{\glstryshort{##1}})}%
6474 }

```

g-sm-short-desc *<long>* (\textsmaller{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

6475 \newacronymstyle{long-sm-short-desc}%
6476 {%
6477 \GlsUseAcrEntryDispStyle{long-sm-short}%
6478 }%
6479 {%
6480 \GlsUseAcrStyleDefs{long-sm-short}%
6481 \renewcommand*{\GenericAcronymFields}{}%
6482 \renewcommand*{\acronymsort}[2]{##2}%
6483 \renewcommand*{\acronymentry}[1]{%
6484     \glstrylong{##1}\space (\acronymfont{\glstryshort{##1}})}%
6485 }

```

short-long-desc *<short>* ({<long>}) acronym style that has an accompanying description (which the user needs to supply).

```

6486 \newacronymstyle{short-long-desc}%
6487 {%
6488 \GlsUseAcrEntryDispStyle{short-long}%
6489 }%
6490 {%
6491 \GlsUseAcrStyleDefs{short-long}%
6492 \renewcommand*{\GenericAcronymFields}{}%
6493 \renewcommand*{\acronymsort}[2]{##2}%
6494 \renewcommand*{\acronymentry}[1]{%
6495     \glstrylong{##1}\space (\acronymfont{\glstryshort{##1}})}%
6496 }

```

short-long-desc *<long>* (\textsc{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

6497 \newacronymstyle{sc-short-long-desc}%
6498 {%
6499 \GlsUseAcrEntryDispStyle{sc-short-long}%
6500 }%
6501 {%
6502 \GlsUseAcrStyleDefs{sc-short-long}%
6503 \renewcommand*{\GenericAcronymFields}{}%
6504 \renewcommand*{\acronymsort}[2]{##2}%
6505 \renewcommand*{\acronymentry}[1]{%

```

```

6506      \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6507 }

```

short-long-desc *<long>* (\textsmaller{\i<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

6508 \newacronymstyle{sm-short-long-desc}%
6509 {%
6510   \GlsUseAcrEntryDispStyle{sm-short-long}%
6511 }%
6512 {%
6513   \GlsUseAcrStyleDefs{sm-short-long}%
6514   \renewcommand*{\GenericAcronymFields}{}%
6515   \renewcommand*{\acronymsort}[2]{##2}%
6516   \renewcommand*{\acronymentry}[1]{%
6517     \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6518 }

```

dua *<long>* only acronym style.

```

6519 \newacronymstyle{dua}%
6520 {%

```

Check for long form in case this is a mixed glossary.

```

6521   \ifdefempty\glscustomtext
6522   {%
6523     \ifglshaslong{\glslabel}%
6524     {%
6525       \glsifplural
6526       {%

```

Plural form:

```

6527       \glscapscase
6528       {%

```

Plural form, don't adjust case:

```

6529       \glsentrylongpl{\glslabel}\glsinsert
6530       }%
6531       {%

```

Plural form, make first letter upper case:

```

6532       \Glsentrylongpl{\glslabel}\glsinsert
6533       }%
6534       {%

```

Plural form, all caps:

```

6535       \mfirstucMakeUppercase
6536       {\glsentrylongpl{\glslabel}\glsinsert}%
6537       }%
6538       }%
6539       {%

```

Singular form

```
6540      \glscapscase
6541      {%
```

Singular form, don't adjust case:

```
6542      \glentrylong{\glslabel}\glsinsert
6543      }%
6544      {%
```

Subsequent singular form, make first letter upper case:

```
6545      \Glsentrylong{\glslabel}\glsinsert
6546      }%
6547      {%
```

Subsequent singular form, all caps:

```
6548      \mfirstucMakeUppercase
6549      {\glentrylong{\glslabel}\glsinsert}%
6550      }%
6551      }%
6552      }%
6553      {%
```

Not an acronym:

```
6554      \glsgenentryfmt
6555      }%
6556      }%
6557      {\glscustomtext\glsinsert}%
6558      }%
6559      {%
6560      \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%

6561      \renewcommand*{\acrfullfmt}[3]{%
6562          \glslink[##1]{##2}{\glentrylong{##2}##3\space
6563              (\acronymfont{\glentryshort{##2}})}}%
6564      \renewcommand*{\Acrfullfmt}[3]{%
6565          \glslink[##1]{##2}{\Glsentrylong{##2}##3\space
6566              (\acronymfont{\glentryshort{##2}})}}%
6567      \renewcommand*{\ACRfullfmt}[3]{%
6568          \glslink[##1]{##2}{%
6569              \mfirstucMakeUppercase{\glentrylong{##2}##3\space
6570              (\acronymfont{\glentryshort{##2}})}}}%

6571      \renewcommand*{\acrfullplfmt}[3]{%
6572          \glslink[##1]{##2}{\glentrylongpl{##2}##3\space
6573              (\acronymfont{\glentryshortpl{##2}})}}%

6574      \renewcommand*{\Acrfullplfmt}[3]{%
6575          \glslink[##1]{##2}{\Glsentrylongpl{##2}##3\space
6576              (\acronymfont{\glentryshortpl{##2}})}}%
6577      \renewcommand*{\ACRfullplfmt}[3]{%
6578          \glslink[##1]{##2}{%
```

```

6579      \mfirstucMakeUppercase{\glentrylongpl{##2}##3\space
6580      (\acronymfont{\glentryshortpl{##2}})}}}%
6581 \renewcommand*{\glentryfull}[1]{%
6582   \glentrylong{##1}\space(\acronymfont{\glentryshort{##1}})%
6583 }%
6584 \renewcommand*{\Glsentryfull}[1]{%
6585   \Glsentrylong{##1}\space(\acronymfont{\glentryshort{##1}})%
6586 }%
6587 \renewcommand*{\glentryfullpl}[1]{%
6588   \glentrylongpl{##1}\space(\acronymfont{\glentryshortpl{##1}})%
6589 }%
6590 \renewcommand*{\Glsentryfullpl}[1]{%
6591   \Glsentrylongpl{##1}\space(\acronymfont{\glentryshortpl{##1}})%
6592 }%
6593 \renewcommand*{\acronymentry}[1]{\acronymfont{\glentryshort{##1}}}%
6594 \renewcommand*{\acronymsort}[2]{##1}%
6595 \renewcommand*{\acronymfont}[1]{##1}%
6596 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6597 }

```

dua-desc *<long>* only acronym style with user-supplied description.

```

6598 \newacronymstyle{dua-desc}%
6599 {%
6600   \GlsUseAcrEntryDispStyle{dua}%
6601 }%
6602 {%
6603   \GlsUseAcrStyleDefs{dua}%
6604   \renewcommand*{\GenericAcronymFields}{}%
6605   \renewcommand*{\acronymentry}[1]{\acronymfont{\glentrylong{##1}}}%
6606   \renewcommand*{\acronymsort}[2]{##2}%
6607 }%

```

footnote *<short>*\footnote{*<long>*} acronym style.

```

6608 \newacronymstyle{footnote}%
6609 {%
6610   Check for long form in case this is a mixed glossary.
6611   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6612 }%
6613 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6614   Need to ensure hyperlinks are switched off on first use:
6615   \glshyperfirstfalse
6616   \renewcommand*{\genacrfullformat}[2]{%
6617     \protect\firstacronymfont{\glentryshort{##1}}##2%
6618     \protect\footnote{\glentrylong{##1}}%
6619   }%
6620   \renewcommand*{\Genacrfullformat}[2]{%

```

```

6620 \firstacronymfont{\Glsentryshort{##1}}##2%
6621 \protect\footnote{\glentrylong{##1}}%
6622 }%
6623 \renewcommand*{\genplacrfullformat}[2]{%
6624 \protect\firstacronymfont{\glentryshortpl{##1}}##2%
6625 \protect\footnote{\glentrylongpl{##1}}%
6626 }%
6627 \renewcommand*{\Genplacrfullformat}[2]{%
6628 \protect\firstacronymfont{\Glsentryshortpl{##1}}##2%
6629 \protect\footnote{\glentrylongpl{##1}}%
6630 }%
6631 \renewcommand*{\acronymentry}[1]{\acronymfont{\glentryshort{##1}}}%
6632 \renewcommand*{\acronymsort}[2]{##1}%
6633 \renewcommand*{\acronymfont}[1]{##1}%
6634 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%

```

Don't use footnotes for \acrfull:

```

6635 \renewcommand*{\acrfullfmt}[3]{%
6636 \glslink[##1]{##2}{\acronymfont{\glentryshort{##2}}##3\space
6637 (\glentrylong{##2})}%
6638 \renewcommand*{\Acrfullfmt}[3]{%
6639 \glslink[##1]{##2}{\acronymfont{\Glsentryshort{##2}}##3\space
6640 (\glentrylong{##2})}%
6641 \renewcommand*{\ACRfullfmt}[3]{%
6642 \glslink[##1]{##2}{%
6643 \mfirstucMakeUppercase{\acronymfont{\glentryshort{##2}}##3\space
6644 (\glentrylong{##2})}}}%
6645 \renewcommand*{\acrfullplfmt}[3]{%
6646 \glslink[##1]{##2}{\acronymfont{\glentryshortpl{##2}}##3\space
6647 (\glentrylongpl{##2})}%
6648 \renewcommand*{\Acrfullplfmt}[3]{%
6649 \glslink[##1]{##2}{\acronymfont{\Glsentryshortpl{##2}}##3\space
6650 (\glentrylongpl{##2})}%
6651 \renewcommand*{\ACRfullplfmt}[3]{%
6652 \glslink[##1]{##2}{%
6653 \mfirstucMakeUppercase{\acronymfont{\glentryshortpl{##2}}##3\space
6654 (\glentrylongpl{##2})}}}%

```

Similarly for \glentryfull etc:

```

6655 \renewcommand*{\glentryfull}[1]{%
6656 \acronymfont{\glentryshort{##1}}\space(\glentrylong{##1})}%
6657 \renewcommand*{\Glsentryfull}[1]{%
6658 \acronymfont{\Glsentryshort{##1}}\space(\glentrylong{##1})}%
6659 \renewcommand*{\glentryfullpl}[1]{%
6660 \acronymfont{\glentryshortpl{##1}}\space(\glentrylongpl{##1})}%
6661 \renewcommand*{\Glsentryfullpl}[1]{%
6662 \acronymfont{\Glsentryshortpl{##1}}\space(\glentrylongpl{##1})}%
6663 }

```

footnote-sc \textsc{<short>}\footnote{<long>} acronym style.

```

6664 \newacronymstyle{footnote-sc}%
6665 {%
6666   \GlsUseAcrEntryDisplayStyle{footnote}%
6667 }%
6668 {%
6669   \GlsUseAcrStyleDefs{footnote}%
6670   \renewcommand{\acronymentry}[1]{\acronymfont{\glentryshort{##1}}}
6671   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6672   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6673 }%

```

footnote-sm \textsmaller{<short>}\footnote{<long>} acronym style.

```

6674 \newacronymstyle{footnote-sm}%
6675 {%
6676   \GlsUseAcrEntryDisplayStyle{footnote}%
6677 }%
6678 {%
6679   \GlsUseAcrStyleDefs{footnote}%
6680   \renewcommand{\acronymentry}[1]{\acronymfont{\glentryshort{##1}}}
6681   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6682   \renewcommand*{\acrpluralsuffix}{\glacrpluralsuffix}%
6683 }%

```

footnote-desc <short>\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

6684 \newacronymstyle{footnote-desc}%
6685 {%
6686   \GlsUseAcrEntryDisplayStyle{footnote}%
6687 }%
6688 {%
6689   \GlsUseAcrStyleDefs{footnote}%
6690   \renewcommand*{\GenericAcronymFields}{}%
6691   \renewcommand*{\acronymsort}[2]{##2}%
6692   \renewcommand*{\acronymentry}[1]{%
6693     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6694 }

```

footnote-sc-desc \textsc{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

6695 \newacronymstyle{footnote-sc-desc}%
6696 {%
6697   \GlsUseAcrEntryDisplayStyle{footnote-sc}%
6698 }%
6699 {%
6700   \GlsUseAcrStyleDefs{footnote-sc}%
6701   \renewcommand*{\GenericAcronymFields}{}%
6702   \renewcommand*{\acronymsort}[2]{##2}%
6703   \renewcommand*{\acronymentry}[1]{%
6704     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%

```

6705 }

ootnote-sm-desc \textsmaller{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

6706 \newacronymstyle{footnote-sm-desc}%

6707 {%

6708 \GlsUseAcrEntryDispStyle{footnote-sm}%

6709 }%

6710 {%

6711 \GlsUseAcrStyleDefs{footnote-sm}%

6712 \renewcommand*{\GenericAcronymFields}{}%

6713 \renewcommand*{\acronymsort}[2]{##2}%

6714 \renewcommand*{\acronymentry}[1]{%

6715 \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%

6716 }

AcronymSynonyms

6717 \newcommand*{\DefineAcronymSynonyms}{%

Short form

\acs

6718 \let\acs\acrshort

First letter uppercase short form

\Acs

6719 \let\Acs\Acrshort

Plural short form

\acsp

6720 \let\acsp\acrshortpl

First letter uppercase plural short form

\Acsp

6721 \let\Acsp\Acrshortpl

Long form

\acl

6722 \let\acl\aclong

Plural long form

\aclp

6723 \let\aclp\aclongpl

First letter upper case long form

`\Ac1`
 6724 `\let\Ac1\Acrlong`
 First letter upper case plural long form

`\Ac1p`
 6725 `\let\Ac1p\Acrlongpl`
 Full form

`\acf`
 6726 `\let\acf\acrfull`
 Plural full form

`\acfp`
 6727 `\let\acfp\acrfullpl`
 First letter upper case full form

`\Acf`
 6728 `\let\Acf\Acrfull`
 First letter upper case plural full form

`\Acfp`
 6729 `\let\Acfp\Acrfullpl`
 Standard form

`\ac`
 6730 `\let\ac\gls`
 First upper case standard form

`\Ac`
 6731 `\let\Ac\Gls`
 Standard plural form

`\acp`
 6732 `\let\acp\glspl`
 Standard first letter upper case plural form

`\Acp`
 6733 `\let\Acp\Glspl`
 6734 }
 Define synonyms if required

6735 `\ifglsacrshortcuts`
 6736 `\DefineAcronymSynonyms`
 6737 `\fi`

These commands for setting the style are now deprecated but are kept for backward compatibility.

`\setAcronymDisplayStyle` Sets the default acronym display style for given glossary.

```
6738 \newcommand*{\SetDefaultAcronymDisplayStyle}[1]{%
6739   \def\glsentryfmt[#1]{\glsentryfmt}%
6740 }
```

`\setNewAcronymDef` Sets up the acronym definition for the default style. The information is provided by the tokens `\glslabeltok`, `\glsshorttok`, `\glslongtok` and `\glskeylisttok`.

```
6741 \newcommand*{\DefaultNewAcronymDef}{%
6742   \edef\@do@newglossaryentry{%
6743     \noexpand\newglossaryentry{\the\glslabeltok}%
6744     {%
6745       type=\acronymtype,%
6746       name={\the\glsshorttok},%
6747       sort={\the\glsshorttok},%
6748       text={\the\glsshorttok},%
6749       first={\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
6750       plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6751       firstplural={\acrfullformat{\noexpand\expandonce\noexpand\@glo@longpl}%
6752                     {\noexpand\expandonce\noexpand\@glo@shortpl}},%
6753       short={\the\glsshorttok},%
6754       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6755       long={\the\glslongtok},%
6756       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6757       description={\the\glslongtok},%
6758       descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
```

Remaining options specified by the user:

```
6759       \the\glskeylisttok
6760     }%
6761   }%
6762   \let\@org@gls@assign@firstpl\gls@assign@firstpl
6763   \let\@org@gls@assign@plural\gls@assign@plural
6764   \let\@org@gls@assign@descplural\gls@assign@descplural
6765   \def\gls@assign@firstpl##1##2{%
6766     \@@gls@expand@field{##1}{firstpl}{##2}%
6767   }%
6768   \def\gls@assign@plural##1##2{%
6769     \@@gls@expand@field{##1}{plural}{##2}%
6770   }%
6771   \def\gls@assign@descplural##1##2{%
6772     \@@gls@expand@field{##1}{descplural}{##2}%
6773   }%
6774   \@do@newglossaryentry
6775   \let\gls@assign@firstpl\@org@gls@assign@firstpl
6776   \let\gls@assign@plural\@org@gls@assign@plural
6777   \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
6778 }
```

ultAcronymStyle Set up the default acronym style:

```
6779 \newcommand*{\SetDefaultAcronymStyle}{%
```

Set the display style:

```
6780 \@for\@gls@type:=\@glsacronymlists\do{%
6781 \SetDefaultAcronymDisplayStyle{\@gls@type}%
6782 }%
```

Set up the definition of `\newacronym`:

```
6783 \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).

```
6784 \ifx\@glsacronymlists\@empty
6785 \def\@glo@type{\acronymtype}%
6786 \setkeys{glossentry}{##1}%
6787 \DeclareAcronymList{\@glo@type}%
6788 \SetDefaultAcronymDisplayStyle{\@glo@type}%
6789 \fi
6790 \glskeylisttok{##1}%
6791 \glslabeltok{##2}%
6792 \glsshorttok{##3}%
6793 \glslongtok{##4}%
6794 \newacronymhook
6795 \DefaultNewAcronymDef
6796 }%
6797 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6798 }
```

\acrfootnote Used by the footnote acronym styles.

```
6799 \newcommand*{\acrfootnote}[3]{\acrlinkfootnote{#1}{#2}{#3}}
```

acrlinkfootnote

```
6800 \newcommand*{\acrlinkfootnote}[3]{%
6801 \footnote{\glslink[#1]{#2}{#3}}%
6802 }
```

acrnolinkfootnote

```
6803 \newcommand*{\acrnolinkfootnote}[3]{%
6804 \footnote{#3}%
6805 }
```

acronymDisplayStyle Sets the acronym display style for given glossary for the description and footnote combination.

```
6806 \newcommand*{\SetDescriptionFootnoteAcronymDisplayStyle}[1]{%
6807 \defglsentryfmt[#1]{%
6808 \ifdefempty\glscustomtext
6809 {%
6810 \ifglsused{\glslabel}%
```

```

6811      {%
6812      \acronymfont{\glsentryfmt}%
6813      }%
6814      {%
6815      \firstacronymfont{\glsentryfmt}%
6816      \ifglshassymbol{\glslabel}%
6817      {%
6818      \expandafter\protect\expandafter\acrfootnote\expandafter
6819      {\@gls@link@opts}{\@gls@link@label}%
6820      {%
6821      \glsifplural
6822      {\glsentrysymbolplural{\glslabel}}%
6823      {\glsentrysymbol{\glslabel}}%
6824      }%
6825      }%
6826      }%
6827      }%
6828      {\glscustomtext\glsinsert}%
6829      }%
6830 }

```

teNewAcronymDef

```

6831 \newcommand*{\DescriptionFootnoteNewAcronymDef}{%
6832   \edef\@do@newglossaryentry{%
6833     \noexpand\newglossaryentry{\the\glslabeltok}%
6834     {%
6835       type=\acronymtype,%
6836       name={\noexpand\acronymfont{\the\glsshorttok}},%
6837       sort={\the\glsshorttok},%
6838       first={\the\glsshorttok},%
6839       firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6840       text={\the\glsshorttok},%
6841       plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6842       short={\the\glsshorttok},%
6843       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6844       long={\the\glslongtok},%
6845       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6846       symbol={\the\glslongtok},%
6847       symbolplural={\noexpand\expandonce\noexpand\@glo@longpl},%
6848       \the\glskeylisttok
6849     }%
6850   }%
6851   \let\@org@gls@assign@firstpl\gls@assign@firstpl
6852   \let\@org@gls@assign@plural\gls@assign@plural
6853   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
6854   \def\gls@assign@firstpl##1##2{%
6855     \@@gls@expand@field{##1}{firstpl}{##2}%
6856   }%
6857   \def\gls@assign@plural##1##2{%

```

```

6858 \@@gls@expand@field{##1}{plural}{##2}%
6859 }%
6860 \def\gls@assign@symbolplural##1##2{%
6861 \@@gls@expand@field{##1}{symbolplural}{##2}%
6862 }%
6863 \do@newglossaryentry
6864 \let\gls@assign@plural\@org@gls@assign@plural
6865 \let\gls@assign@firstpl\@org@gls@assign@firstpl
6866 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
6867 }

```

oteAcronymStyle If a description and footnote are both required, store the long form in the symbol key. Store the short form in text key. Note that since the long form is stored in the symbol key, if you want the long form to appear in the list of acronyms, you need to use a glossary style that displays the symbol key.

```

6868 \newcommand*\SetDescriptionFootnoteAcronymStyle{%
6869 \renewcommand{\newacronym}[4][\]{%
6870 \ifx\@glsacronymlists\@empty
6871 \def\@glo@type{\acronymtype}%
6872 \setkeys{glossentry}{##1}%
6873 \DeclareAcronymList{\@glo@type}%
6874 \SetDescriptionFootnoteAcronymDisplayStyle{\@glo@type}%
6875 \fi
6876 \glskeylisttok{##1}%
6877 \glslabeltok{##2}%
6878 \glsshorttok{##3}%
6879 \glslongtok{##4}%
6880 \newacronymhook
6881 \DescriptionFootnoteNewAcronymDef
6882 }%

```

If footnote package option is specified, set the first use to append the long form (stored in symbol) as a footnote.

```

6883 \@for\@gls@type:=\@glsacronymlists\do{%
6884 \SetDescriptionFootnoteAcronymDisplayStyle{\@gls@type}%
6885 }%

```

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.

```

6886 \ifglsacrsmallcaps
6887 \renewcommand*\acronymfont[1]{\textsc{##1}}%
6888 \renewcommand*\acrpluralsuffix{\glsupacrpluralsuffix}%
6889 \else
6890 \ifglsacrsmaller
6891 \renewcommand*\acronymfont[1]{\textsmaller{##1}}%
6892 \fi
6893 \fi

```

Check for package option clash

```

6894 \ifglsacrdua
6895   \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’
6896     can’t both be set}{}%
6897 \fi
6898 }%

```

nymDisplayStyle Sets the acronym display style for given glossary with description and dua combination.

```

6899 \newcommand*{\SetDescriptionDUAAcronymDisplayStyle}[1]{%
6900   \defglsentryfmt[#1]{\glsentryfmt}%
6901 }

```

UANewAcronymDef

```

6902 \newcommand*{\DescriptionDUANewAcronymDef}{%
6903   \edef\@do@newglossaryentry{%
6904     \noexpand\newglossaryentry{\the\glslabeltok}%
6905     {%
6906       type=\acronymtype,%
6907       name={\the\glslongtok},%
6908       sort={\the\glslongtok},%
6909       text={\the\glslongtok},%
6910       first={\the\glslongtok},%
6911       plural={\noexpand\expandonce\noexpand\@glo@longpl},%
6912       firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
6913       short={\the\glsshorttok},%
6914       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6915       long={\the\glslongtok},%
6916       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6917       symbol={\the\glsshorttok},%
6918       symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6919       \the\glskeylisttok
6920     }%
6921   }%
6922   \let\@org@gls@assign@firstpl\gls@assign@firstpl
6923   \let\@org@gls@assign@plural\gls@assign@plural
6924   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
6925   \def\gls@assign@firstpl##1##2{%
6926     \@@gls@expand@field{##1}{firstpl}{##2}%
6927   }%
6928   \def\gls@assign@plural##1##2{%
6929     \@@gls@expand@field{##1}{plural}{##2}%
6930   }%
6931   \def\gls@assign@symbolplural##1##2{%
6932     \@@gls@expand@field{##1}{symbolplural}{##2}%
6933   }%
6934   \@do@newglossaryentry
6935   \let\gls@assign@firstpl\@org@gls@assign@firstpl
6936   \let\gls@assign@plural\@org@gls@assign@plural
6937   \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
6938 }

```

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.

```

6939 \newcommand*{\SetDescriptionDUAACronymStyle}{%
6940   \ifglsmallcaps
6941     \PackageError{glossaries}{Option clash: 'smallcaps' and 'dua'
6942       can't both be set}{}%
6943   \else
6944     \ifglsmaller
6945       \PackageError{glossaries}{Option clash: 'smaller' and 'dua'
6946         can't both be set}{}%
6947     \fi
6948   \fi
6949   \renewcommand{\newacronym}[4][{}]{%
6950     \ifx\@glsacronymlists\@empty
6951       \def\@glo@type{\acronymtype}%
6952       \setkeys{glossentry}{##1}%
6953       \DeclareAcronymList{\@glo@type}%
6954       \SetDescriptionDUAACronymDisplayStyle{\@glo@type}%
6955     \fi
6956     \glskeylisttok{##1}%
6957     \glslabeltok{##2}%
6958     \glsshorttok{##3}%
6959     \glslongtok{##4}%
6960     \newacronymhook
6961     \DescriptionDUANewAcronymDef
6962   }%

```

Set display.

```

6963   \@for\@gls@type:=\@glsacronymlists\do{%
6964     \SetDescriptionDUAACronymDisplayStyle{\@gls@type}%
6965   }%
6966 }%

```

nymDisplayStyle Sets the acronym display style for given glossary using the description setting (but not footnote or dua).

```

6967 \newcommand*{\SetDescriptionAcronymDisplayStyle}[1]{%
6968   \defglentryfmt[#1]{%
6969     \ifdefempty\glscustomtext
6970     {%
6971       \ifglused{\glslabel}%
6972     }%

```

Move the inserted text outside of \acronymfont

```

6973       \let\gls@org@insert\glsinsert
6974       \let\glsinsert\@empty
6975       \acronymfont{\glsgenentryfmt}\gls@org@insert
6976     }%

```

```

6977     {%
6978         \glsgenentryfmt
6979         \ifglshassymbol{\glslabel}%
6980         {%
6981             \glsifplural
6982             {%
6983                 \def\@glo@symbol{\glsentrysymbolplural{\glslabel}}%
6984             }%
6985             {%
6986                 \def\@glo@symbol{\glsentrysymbol{\glslabel}}%
6987             }%
6988             \space(\protect\firstacronymfont
6989             {\glscapscase
6990              {\@glo@symbol}
6991              {\@glo@symbol}
6992              {\mfirstucMakeUppercase{\@glo@symbol}}})%
6993         }%
6994     }%
6995 }%
6996 }%
6997 {\glscustomtext\glsinsert}%
6998 }%
6999 }

```

onNewAcronymDef

```

7000 \newcommand*{\DescriptionNewAcronymDef}{%
7001     \edef\@do@newglossaryentry{%
7002         \noexpand\newglossaryentry{\the\glslabeltok}%
7003         {%
7004             type=\acronymtype,%
7005             name={\noexpand
7006                 \acrnameformat{\the\glsshorttok}{\the\glslongtok}},%
7007             sort={\the\glsshorttok},%
7008             first={\the\glslongtok},%
7009             firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7010             text={\the\glsshorttok},%
7011             plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7012             short={\the\glsshorttok},%
7013             shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7014             long={\the\glslongtok},%
7015             longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7016             symbol={\noexpand\@glo@text},%
7017             symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7018             \the\glskeylisttok}%
7019     }%
7020     \let\@org@gls@assign@firstpl\gls@assign@firstpl
7021     \let\@org@gls@assign@plural\gls@assign@plural
7022     \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7023     \def\gls@assign@firstpl##1##2{%

```

```

7024 \@@gls@expand@field{##1}{firstpl}{##2}%
7025 }%
7026 \def\gls@assign@plural##1##2{%
7027 \@@gls@expand@field{##1}{plural}{##2}%
7028 }%
7029 \def\gls@assign@symbolplural##1##2{%
7030 \@@gls@expand@field{##1}{symbolplural}{##2}%
7031 }%
7032 \do@newglossaryentry
7033 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7034 \let\gls@assign@plural\@org@gls@assign@plural
7035 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7036 }

```

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.

```

7037 \newcommand*{\SetDescriptionAcronymStyle}{%
7038 \renewcommand{\newacronym}[4][\]{%
7039 \ifx\@glsacronymlists\@empty
7040 \def\@glo@type{\acronymtype}%
7041 \setkeys{glossentry}{##1}%
7042 \DeclareAcronymList{\@glo@type}%
7043 \SetDescriptionAcronymDisplayStyle{\@glo@type}%
7044 \fi
7045 \glskeylisttok{##1}%
7046 \glslabeltok{##2}%
7047 \glsshorttok{##3}%
7048 \glslongtok{##4}%
7049 \newacronymhook
7050 \DescriptionNewAcronymDef
7051 }%

```

Set display.

```

7052 \@for\@gls@type:=\@glsacronymlists\do{%
7053 \SetDescriptionAcronymDisplayStyle{\@gls@type}%
7054 }%

```

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.

```

7055 \ifglsacrsmallcaps
7056 \renewcommand{\acronymfont}[1]{\textsc{##1}}
7057 \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7058 \else
7059 \ifglsacrsmaller
7060 \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
7061 \fi
7062 \fi
7063 }%

```


nymDisplayStyle Sets the acronym display style for given glossary with footnote setting (but not description or dua).

```

7064 \newcommand*{\SetFootnoteAcronymDisplayStyle}[1]{%
7065   \defglentryfmt[#1]{%

7066     \ifdefempty\glscustomtext
7067     {%

      Move the inserted text outside of \acronymfont

7068       \let\gls@org@insert\glsinsert
7069       \let\glsinsert\@empty
7070       \ifglused{\glslabel}%
7071       {%
7072         \acronymfont{\glsgenentryfmt}\gls@org@insert
7073       }%
7074       {%
7075         \firstacronymfont{\glsgenentryfmt}\gls@org@insert
7076         \ifglshaslong{\glslabel}%
7077         {%
7078           \expandafter\protect\expandafter\acrfootnote\expandafter
7079           {\@gls@link@opts}{\@gls@link@label}%
7080           {%
7081             \glsifplural
7082             {\glsgenentrylongpl{\glslabel}}%
7083             {\glsgenentrylong{\glslabel}}%
7084           }%
7085         }%
7086       }%
7087     }%
7088   }%
7089   {\glscustomtext\glsinsert}%
7090 }%
7091 }

```

teNewAcronymDef

```

7092 \newcommand*{\FootnoteNewAcronymDef}{%
7093   \edef\@do@newglossaryentry{%
7094     \noexpand\newglossaryentry{\the\glslabeltok}%
7095     {%
7096       type=\acronymtype,%
7097       name={\noexpand\acronymfont{\the\glsshorttok}},%
7098       sort={\the\glsshorttok},%
7099       text={\the\glsshorttok},%
7100       plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7101       first={\the\glsshorttok},%
7102       firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7103       short={\the\glsshorttok},%
7104       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7105       long={\the\glslongtok},%

```

```

7106     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7107     description={\the\glslongtok},%
7108     descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7109     \the\glskeylisttok
7110   }%
7111 }%
7112 \let\@org@gls@assign@plural\gls@assign@plural
7113 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7114 \let\@org@gls@assign@descplural\gls@assign@descplural
7115 \def\gls@assign@firstpl##1##2{%
7116   \@gls@expand@field{##1}{firstpl}{##2}%
7117 }%
7118 \def\gls@assign@plural##1##2{%
7119   \@gls@expand@field{##1}{plural}{##2}%
7120 }%
7121 \def\gls@assign@descplural##1##2{%
7122   \@gls@expand@field{##1}{descplural}{##2}%
7123 }%
7124 \do@newglossaryentry
7125 \let\gls@assign@plural\@org@gls@assign@plural
7126 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7127 \let\gls@assign@descplural\@org@gls@assign@descplural
7128 }

```

oteAcronymStyle If footnote package option is specified, set the first use to append the long form (stored in description) as a footnote. Use the description key to store the long form.

```

7129 \newcommand*{\SetFootnoteAcronymStyle}{%
7130   \renewcommand{\newacronym}[4][]{%
7131     \ifx\@glsacronymlists\@empty
7132       \def\@glo@type{\acronymtype}%
7133       \setkeys{glossentry}{##1}%
7134       \DeclareAcronymList{\@glo@type}%
7135       \SetFootnoteAcronymDisplayStyle{\@glo@type}%
7136     \fi
7137     \glskeylisttok{##1}%
7138     \glslabeltok{##2}%
7139     \glsshorttok{##3}%
7140     \glslongtok{##4}%
7141     \newacronymhook
7142     \FootnoteNewAcronymDef
7143   }%

```

Set display

```

7144   \@for\@gls@type:=\@glsacronymlists\do{%
7145     \SetFootnoteAcronymDisplayStyle{\@gls@type}%
7146   }%

```

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.

```

7147   \ifglsacrsmallcaps

```

```

7148     \renewcommand*{\acronymfont}[1]{\textsc{##1}}%
7149     \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7150 \else
7151     \ifglssacrsmaller
7152         \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
7153     \fi
7154 \fi

    Check for option clash
7155 \ifglssacrdua
7156     \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’
7157         can’t both be set}{}%
7158 \fi
7159 }%

```

`\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.

```

7160 \DeclareRobustCommand*{\glsdoparenifnotempty}[2]{%
7161     \protected@edef\gls@tmp{#1}%
7162     \ifdefempty\gls@tmp
7163     {%
7164     {%
7165         \ifx\gls@tmp\@gls@default@value
7166         \else
7167             \space (#2{#1})%
7168         \fi
7169     }%
7170 }

```

`\nymDisplayStyle` Sets the acronym display style for given glossary where neither footnote nor description is required, but smallcaps or smaller specified.

```

7171 \newcommand*{\SetSmallAcronymDisplayStyle}[1]{%
7172     \defglsenentryfmt[#1]{%

7173     \ifdefempty\glscustomtext
7174     {%

```

Move the inserted text outside of `\acronymfont`

```

7175         \let\gls@org@insert\glsinsert
7176         \let\glsinsert\@empty
7177         \ifglssused{\glslabel}%
7178         {%
7179             \acronymfont{\glsgenentryfmt}\gls@org@insert
7180         }%
7181         {%
7182             \glsgenentryfmt
7183             \ifglshassymbol{\glslabel}%
7184             {%
7185                 \glsifplural
7186                 {%

```

```

7187         \def\@glo@symbol{\glentrysymbolplural{\glslabel}}%
7188     }%
7189     {%
7190         \def\@glo@symbol{\glentrysymbol{\glslabel}}%
7191     }%
7192     \space
7193     (\glscapscase
7194     {\firstacronymfont{\@glo@symbol}}%
7195     {\firstacronymfont{\@glo@symbol}}%
7196     {\firstacronymfont{\mfirstucMakeUppercase{\@glo@symbol}}})%
7197 }%
7198 {}%
7199 }%
7200 }%
7201 {\glscustomtext\glsinsert}%
7202 }%
7203 }

```

11NewAcronymDef

```

7204 \newcommand*{\SmallNewAcronymDef}{%
7205     \edef\@do@newglossaryentry{%
7206         \noexpand\newglossaryentry{\the\glslabeltok}%
7207         {%
7208             type=\acronymtype,%
7209             name={\noexpand\acronymfont{\the\glsshorttok}},%
7210             sort={\the\glsshorttok},%
7211             text={\the\glsshorttok},%
7212             plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7213             first={\the\glslongtok},%
7214             firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7215             short={\the\glsshorttok},%
7216             shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7217             long={\the\glslongtok},%
7218             longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7219             description={\noexpand\@glo@first},%
7220             descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7221             symbol={\the\glsshorttok},%
7222             symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7223             \the\glskeylisttok
7224         }%
7225     }%
7226     \let\@org@gls@assign@firstpl\gls@assign@firstpl
7227     \let\@org@gls@assign@plural\gls@assign@plural
7228     \let\@org@gls@assign@descplural\gls@assign@descplural

```

```

7229 \let\org@gl@s@assign@symbolplural\gl@s@assign@symbolplural
7230 \def\gl@s@assign@firstpl##1##2{%
7231   \@@gl@s@expand@field{##1}{firstpl}{##2}%
7232 }%
7233 \def\gl@s@assign@plural##1##2{%
7234   \@@gl@s@expand@field{##1}{plural}{##2}%
7235 }%
7236 \def\gl@s@assign@descplural##1##2{%
7237   \@@gl@s@expand@field{##1}{descplural}{##2}%
7238 }%
7239 \def\gl@s@assign@symbolplural##1##2{%
7240   \@@gl@s@expand@field{##1}{symbolplural}{##2}%
7241 }%
7242 \do@newglossaryentry
7243 \let\gl@s@assign@firstpl\org@gl@s@assign@firstpl
7244 \let\gl@s@assign@plural\org@gl@s@assign@plural
7245 \let\gl@s@assign@descplural\org@gl@s@assign@descplural
7246 \let\gl@s@assign@symbolplural\org@gl@s@assign@symbolplural
7247 }

```

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.

```

7248 \newcommand*{\SetSmallAcronymStyle}{%
7249   \renewcommand{\newacronym}[4][]{%
7250     \ifx\org@gl@s@acronymlists\@empty
7251       \def\org@type{\acronymtype}%
7252       \setkeys{glossentry}{##1}%
7253       \DeclareAcronymList{\org@type}%
7254       \SetSmallAcronymDisplayStyle{\org@type}%
7255     \fi
7256     \gl@keylisttok{##1}%
7257     \gl@labeltok{##2}%
7258     \gl@shorttok{##3}%
7259     \gl@longtok{##4}%
7260     \newacronymhook
7261     \SmallNewAcronymDef
7262   }%

```

Change the display since first only contains long form.

```

7263 \@for\org@type:=\org@gl@s@acronymlists\do{%
7264   \SetSmallAcronymDisplayStyle{\org@type}%
7265 }%

```

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.

```

7266 \ifgl@acrsmallcaps
7267   \renewcommand*{\acronymfont}[1]{\textsc{##1}}
7268   \renewcommand*{\acrpluralsuffix}{\gl@supacrpluralsuffix}%
7269 \else
7270   \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}

```

```

7271 \fi
      check for option clash
7272 \ifglsacrdua
7273   \ifglsacrsmallcaps
7274     \PackageError{glossaries}{Option clash: ‘smallcaps’ and ‘dua’
7275       can’t both be set}{}%
7276   \else
7277     \PackageError{glossaries}{Option clash: ‘smaller’ and ‘dua’
7278       can’t both be set}{}%
7279   \fi
7280 \fi
7281 }%

```

DUADisplayStyle Sets the acronym display style for given glossary with dua setting.

```

7282 \newcommand*{\SetDUADisplayStyle}[1]{%
7283   \defglsentryfmt[#1]{\glsentryfmt}%
7284 }

```

UANewAcronymDef

```

7285 \newcommand*{\DUANewAcronymDef}{%
7286   \edef\@do@newglossaryentry{%
7287     \noexpand\newglossaryentry{\the\glslabeltok}%
7288     {%
7289       type=\acronymtype,%
7290       name={\the\glsshorttok},%
7291       text={\the\glslongtok},%
7292       first={\the\glslongtok},%
7293       plural={\noexpand\expandonce\noexpand\@glo@longpl},%
7294       firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7295       short={\the\glsshorttok},%
7296       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7297       long={\the\glslongtok},%
7298       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7299       description={\the\glslongtok},%
7300       descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7301       symbol={\the\glsshorttok},%
7302       symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7303       \the\glskeylisttok
7304     }%
7305   }%
7306   \let\@org@gls@assign@firstpl\gls@assign@firstpl
7307   \let\@org@gls@assign@plural\gls@assign@plural
7308   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7309   \let\@org@gls@assign@descplural\gls@assign@descplural
7310   \def\gls@assign@firstpl##1##2{%
7311     \@@gls@expand@field{##1}{firstpl}{##2}%
7312   }%
7313   \def\gls@assign@plural##1##2{%
7314     \@@gls@expand@field{##1}{plural}{##2}%

```

```

7315 }%
7316 \def\gls@assign@symbolplural##1##2{%
7317   \@gls@expand@field{##1}{symbolplural}{##2}%
7318 }%
7319 \def\gls@assign@descplural##1##2{%
7320   \@gls@expand@field{##1}{descplural}{##2}%
7321 }%
7322 \do@newglossaryentry
7323 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7324 \let\gls@assign@plural\@org@gls@assign@plural
7325 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7326 \let\gls@assign@descplural\@org@gls@assign@descplural
7327 }

```

\SetDUASyle Always expand acronyms.

```

7328 \newcommand*{\SetDUASyle}{%
7329   \renewcommand{\newacronym}[4][]{%
7330     \ifx\@glsacronymlists\@empty
7331       \def\@glo@type{\acronymtype}%
7332       \setkeys{glossentry}{##1}%
7333       \DeclareAcronymList{\@glo@type}%
7334       \SetDUADisplayStyle{\@glo@type}%
7335     \fi
7336     \glskeylisttok{##1}%
7337     \glslabeltok{##2}%
7338     \glsshorttok{##3}%
7339     \glslongtok{##4}%
7340     \newacronymhook
7341     \DUANewAcronymDef
7342   }%
7343   \Set the display
7344   \@for\@gls@type:=\@glsacronymlists\do{%
7345     \SetDUADisplayStyle{\@gls@type}%
7346 }%

```

SetAcronymStyle

```

7347 \newcommand*{\SetAcronymStyle}{%
7348   \SetDefaultAcronymStyle
7349   \ifglsacrdescription
7350     \ifglsacrfootnote
7351       \SetDescriptionFootnoteAcronymStyle
7352     \else
7353       \ifglsacrdua
7354         \SetDescriptionDUAAcronymStyle
7355       \else
7356         \SetDescriptionAcronymStyle
7357     \fi
7358   \fi

```

```

7359 \else
7360   \ifglsacrfootnote
7361     \SetFootnoteAcronymStyle
7362   \else
7363     \ifthenelse{\boolean{glsacrsmalldcaps}}\OR
7364       \boolean{glsacrsmalldr}}{
7365       {%
7366         \SetSmallAcronymStyle
7367       }%
7368     {%
7369       \ifglsacrdua
7370         \SetDUASStyle
7371       \fi
7372     }%
7373   \fi
7374 \fi
7375 }

```

Set the acronym style according to the package options

```
7376 \SetAcronymStyle
```

Allow user to define their own custom acronyms. (For compatibility with versions before v3.0, the short form is stored in the user1 key, the plural short form is stored in the user2 key, the long form is stored in the user3 key and the plural long form is stored in the user4 key.) Defaults to displaying only the acronym with the long form as the description.

`\setacronymstyle` Sets the acronym display style.

```

7377 \newcommand*{\SetCustomDisplayStyle}[1]{%
7378   \defglsentryfmt[#1]{\glsentryfmt}%
7379 }

```

`\setacronymfields`

```

7380 \newcommand*{\CustomAcronymFields}{%
7381   name={\the\glsshorttok},%
7382   description={\the\glslongtok},%
7383   first={\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
7384   firstplural={\acrfullformat
7385     {\noexpand\glsentrylongpl{\the\glslabeltok}}}%
7386     {\noexpand\glsentryshortpl{\the\glslabeltok}}},%

7387   text={\the\glsshorttok},%
7388   plural={\the\glsshorttok\noexpand\acrpluralsuffix}%
7389 }

```

`\setnewacronymdef`

```

7390 \newcommand*{\CustomNewAcronymDef}{%
7391   \protected@edef\@do@newglossaryentry{%
7392     \noexpand\newglossaryentry{\the\glslabeltok}%
7393     {%

```



```

7394     type=\acronymtype,%
7395     short={\the\glsshorttok},%
7396     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7397     long={\the\glslongtok},%
7398     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7399     user1={\the\glsshorttok},%
7400     user2={\the\glsshorttok\noexpand\acrpluralsuffix},%
7401     user3={\the\glslongtok},%
7402     user4={\the\glslongtok\noexpand\acrpluralsuffix},%
7403     \CustomAcronymFields,%
7404     \the\glskeylisttok
7405   }%
7406 }%
7407 \do@newglossaryentry
7408 }

```

\SetCustomStyle

```

7409 \newcommand*{\SetCustomStyle}{%
7410   \renewcommand{\newacronym}[4][]{%
7411     \ifx\@glsacronymlists\@empty
7412       \def\@gls@type{\acronymtype}%
7413       \setkeys{glossentry}{##1}%
7414       \DeclareAcronymList{\@gls@type}%
7415       \SetCustomDisplayStyle{\@gls@type}%
7416     \fi
7417     \glskeylisttok{##1}%
7418     \glslabeltok{##2}%
7419     \glsshorttok{##3}%
7420     \glslongtok{##4}%
7421     \newacronymhook
7422     \CustomNewAcronymDef
7423   }%
7424   \SetCustomDisplayStyle{\@gls@type}%
7425   \SetCustomDisplayStyle{\@gls@type}%
7426 }%
7427 }

```

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.

```
7428 \RequirePackage{glossary-hypernav}
```

The styles that use list-like environments. These are not loaded if the nolist option is used:

```
7429 \@gls@loadlist
```

The styles that use the longtable environment. These are not loaded if the nolong package option is used.

7430 \@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.

7431 \@gls@loadsuper

The tree-like styles. These are not loaded if the notree package option is used.

7432 \@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.

7433 \ifx\@glossary@default@style\relax

7434 \else

7435 \setglossarystyle{\@glossary@default@style}

7436 \fi

1.20 Debugging Commands

\showgloparent \showgloparent{<label>}

7437 \newcommand*{\showgloparent}[1]{%

7438 \expandafter\show\csname glo@\glsdetoklabel{#1}@parent\endcsname

7439 }

\showglolevel \showglolevel{<label>}

7440 \newcommand*{\showglolevel}[1]{%

7441 \expandafter\show\csname glo@\glsdetoklabel{#1}@level\endcsname

7442 }

\showglotext \showglotext{<label>}

7443 \newcommand*{\showglotext}[1]{%

7444 \expandafter\show\csname glo@\glsdetoklabel{#1}@text\endcsname

7445 }

\showgloplural \showgloplural{<label>}

```

7446 \newcommand*{\showgloplural}[1]{%
7447   \expandafter\show\csname glo@glstetoklabel{#1}@plural\endcsname
7448 }

```

\showglofirst `\showglofirst{<label>}`

```

7449 \newcommand*{\showglofirst}[1]{%
7450   \expandafter\show\csname glo@glstetoklabel{#1}@first\endcsname
7451 }

```

\showglofirstpl `\showglofirstpl{<label>}`

```

7452 \newcommand*{\showglofirstpl}[1]{%
7453   \expandafter\show\csname glo@glstetoklabel{#1}@firstpl\endcsname
7454 }

```

\showgloftype `\showgloftype{<label>}`

```

7455 \newcommand*{\showgloftype}[1]{%
7456   \expandafter\show\csname glo@glstetoklabel{#1}@type\endcsname
7457 }

```

\showglocounter `\showglocounter{<label>}`

```

7458 \newcommand*{\showglocounter}[1]{%
7459   \expandafter\show\csname glo@glstetoklabel{#1}@counter\endcsname
7460 }

```

\showglouserii `\showglouserii{<label>}`

```

7461 \newcommand*{\showglouserii}[1]{%
7462   \expandafter\show\csname glo@glstetoklabel{#1}@userii\endcsname
7463 }

```

\showglouserii `\showglouserii{<label>}`

```

7464 \newcommand*{\showglouserii}[1]{%
7465   \expandafter\show\csname glo@glstdetoklabel{#1}@userii\endcsname
7466 }

```

\showglouseriii \showglouseriii{<label>}

```

7467 \newcommand*{\showglouseriii}[1]{%
7468   \expandafter\show\csname glo@glstdetoklabel{#1}@useriii\endcsname
7469 }

```

\showglouseriv \showglouseriv{<label>}

```

7470 \newcommand*{\showglouseriv}[1]{%
7471   \expandafter\show\csname glo@glstdetoklabel{#1}@useriv\endcsname
7472 }

```

\showglouserv \showglouserv{<label>}

```

7473 \newcommand*{\showglouserv}[1]{%
7474   \expandafter\show\csname glo@glstdetoklabel{#1}@userv\endcsname
7475 }

```

\showglouservi \showglouservi{<label>}

```

7476 \newcommand*{\showglouservi}[1]{%
7477   \expandafter\show\csname glo@glstdetoklabel{#1}@uservi\endcsname
7478 }

```

\showgloname \showgloname{<label>}

```

7479 \newcommand*{\showgloname}[1]{%
7480   \expandafter\show\csname glo@glstdetoklabel{#1}@name\endcsname
7481 }

```

\showglodesc \showglodesc{<label>}

```

7482 \newcommand*{\showglodesc}[1]{%
7483   \expandafter\show\csname glo@\glsdetoklabel{#1}@desc\endcsname
7484 }

```

showglodescplural `\showglodescplural{<label>}`

```

7485 \newcommand*{\showglodescplural}[1]{%
7486   \expandafter\show\csname glo@\glsdetoklabel{#1}@descplural\endcsname
7487 }

```

\showglosort `\showglosort{<label>}`

```

7488 \newcommand*{\showglosort}[1]{%
7489   \expandafter\show\csname glo@\glsdetoklabel{#1}@sort\endcsname
7490 }

```

\showglosymbol `\showglosymbol{<label>}`

```

7491 \newcommand*{\showglosymbol}[1]{%
7492   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbol\endcsname
7493 }

```

showglosymbolplural `\showglosymbolplural{<label>}`

```

7494 \newcommand*{\showglosymbolplural}[1]{%
7495   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolplural\endcsname
7496 }

```

\showgloshort `\showgloshort{<label>}`

```

7497 \newcommand*{\showgloshort}[1]{%
7498   \expandafter\show\csname glo@\glsdetoklabel{#1}@short\endcsname
7499 }

```

\showglolong `\showglolong{<label>}`

```

7500 \newcommand*{\showglolong}[1]{%
7501   \expandafter\show\csname glo@\glsdetoklabel{#1}@long\endcsname
7502 }

```

\showgloindex \showgloindex{<label>}

```

7503 \newcommand*{\showgloindex}[1]{%
7504   \expandafter\show\csname glo@\glsdetoklabel{#1}@index\endcsname
7505 }

```

\showgloflag \showgloflag{<label>}

```

7506 \newcommand*{\showgloflag}[1]{%
7507   \expandafter\show\csname ifglo@\glsdetoklabel{#1}@flag\endcsname
7508 }

```

\showgloloclist \showgloloclist{<label>}

```

7509 \newcommand*{\showgloloclist}[1]{%
7510   \expandafter\show\csname glo@\glsdetoklabel{#1}@loclist\endcsname
7511 }

```

\showglofield \showglofield{<label>}{<field>}

```

7512 \newcommand*{\showglofield}[2]{%
7513   \csshow{glo@\glsdetoklabel{#1}@#2}%
7514 }

```

showacronymlists \showacronymlists

Show list of glossaries that have been flagged as a list of acronyms.

```

7515 \newcommand*{\showacronymlists}{%
7516   \show\@glsacronymlists
7517 }

```

\showglossaries \showglossaries

Show list of defined glossaries.

```

7518 \newcommand*{\showglossaries}{%

```

```

7519 \show\@glo@types
7520 }

```

`\showglossaryin` `\showglossaryin{<glossary-label>}`

Show the ‘in’ extension for the given glossary.

```

7521 \newcommand*{\showglossaryin}[1]{%
7522 \expandafter\show\csname @glotype@#1@in\endcsname
7523 }

```

`\showglossaryout` `\showglossaryout{<glossary-label>}`

Show the ‘out’ extension for the given glossary.

```

7524 \newcommand*{\showglossaryout}[1]{%
7525 \expandafter\show\csname @glotype@#1@out\endcsname
7526 }

```

`showglossarytitle` `\showglossarytitle{<glossary-label>}`

Show the title for the given glossary.

```

7527 \newcommand*{\showglossarytitle}[1]{%
7528 \expandafter\show\csname @glotype@#1@title\endcsname
7529 }

```

`wglossarycounter` `\showglossarycounter{<glossary-label>}`

Show the counter for the given glossary.

```

7530 \newcommand*{\showglossarycounter}[1]{%
7531 \expandafter\show\csname @glotype@#1@counter\endcsname
7532 }

```

`wglossaryentries` `\showglossaryentries{<glossary-label>}`

Show the list of entry labels for the given glossary.

```

7533 \newcommand*{\showglossaryentries}[1]{%
7534 \expandafter\show\csname glolist@#1\endcsname
7535 }

```

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.

```
7536 \csname ifglscpatible-2.07\endcsname
7537   \RequirePackage{glossaries-compatible-207}
7538 \fi
```


2 Prefix Support (glossaries-prefix Code)

This package provides a means of adding prefixes to your glossary entries. For example, you may want to use “`\gls{<label>}`” on first use but use “`\an \gls{<label>}`” on subsequent use.

```
7539 \NeedsTeXFormat{LaTeX2e}
```

```
7540 \ProvidesPackage{glossaries-prefix}[2016/04/30 v4.23 (NLCT)]
```

Pass all options to glossaries:

```
7541 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
7542 \ProcessOptions
```

Load glossaries:

```
7543 \RequirePackage{glossaries}
```

Add the new keys:

```
7544 \define@key{glossentry}{prefixfirst}{\def\@glo@entryprefixfirst{#1}}%
```

```
7545 \define@key{glossentry}{prefixfirstplural}{\def\@glo@entryprefixfirstplural{#1}}%
```

```
7546 \define@key{glossentry}{prefix}{\def\@glo@entryprefix{#1}}%
```

```
7547 \define@key{glossentry}{prefixplural}{\def\@glo@entryprefixplural{#1}}%
```

Add them to `\@gls@keymap`:

```
7548 \appto\@gls@keymap{,%
```

```
7549   {prefixfirst}{prefixfirst},%
```

```
7550   {prefixfirstplural}{prefixfirstplural},%
```

```
7551   {prefix}{prefix},%
```

```
7552   {prefixplural}{prefixplural}}%
```

```
7553 }
```

Set the default values:

```
7554 \appto\@newglossaryentryprehook{%
```

```
7555   \def\@glo@entryprefix{}
```

```
7556   \def\@glo@entryprefixplural{}
```

```
7557   \let\@glo@entryprefixfirst\@gls@default@value
```

```
7558   \let\@glo@entryprefixfirstplural\@gls@default@value
```

```
7559 }
```

Set the assignment code:

```
7560 \appto\@newglossaryentryposthook{%
```

```
7561   \gls@assign@field{\@glo@label}{prefix}{\@glo@entryprefix}%
```

```
7562   \gls@assign@field{\@glo@label}{prefixplural}{\@glo@entryprefixplural}%
```

If `prefixfirst` has not been supplied, make it the same as `prefix`.

```
7563   \expandafter\gls@assign@field\expandafter
```

```
7564     {\csname glo@\@glo@label @prefix\endcsname}{\@glo@label}{prefixfirst}%
```

```
7565     {\@glo@entryprefixfirst}}%
```

If prefixfirstplural has not been supplied, make it the same as prefixplural.

```

7566 \expandafter\gls@assign@field\expandafter
7567   {\csname glo@\@glo@label @prefixplural\endcsname}{\@glo@label}%
7568   {prefixfirstplural}{\@glo@entryprefixfirstplural}%
7569 }

```

Define commands to access these fields:

entryprefixfirst

```

7570 \newcommand*{\glsentryprefixfirst}[1]{\csuse{glo@#1@prefixfirst}}

```

entryfirstplural

```

7571 \newcommand*{\glsentryprefixfirstplural}[1]{\csuse{glo@#1@prefixfirstplural}}

```

\glsentryprefix

```

7572 \newcommand*{\glsentryprefix}[1]{\csuse{glo@#1@prefix}}

```

entryprefixplural

```

7573 \newcommand*{\glsentryprefixplural}[1]{\csuse{glo@#1@prefixplural}}

```

Now for the initial upper case variants:

entryprefixfirst

```

7574 \newrobustcmd*{\Glsentryprefixfirst}[1]{%
7575   \protected@edef\@glo@text{\csname glo@#1@prefixfirst\endcsname}%
7576   \xmakefirstuc\@glo@text
7577 }

```

entryfirstplural

```

7578 \newrobustcmd*{\Glsentryprefixfirstplural}[1]{%
7579   \protected@edef\@glo@text{\csname glo@#1@prefixfirstplural\endcsname}%
7580   \xmakefirstuc\@glo@text
7581 }

```

\Glsentryprefix

```

7582 \newrobustcmd*{\Glsentryprefix}[1]{%
7583   \protected@edef\@glo@text{\csname glo@#1@prefix\endcsname}%
7584   \xmakefirstuc\@glo@text
7585 }

```

entryprefixplural

```

7586 \newrobustcmd*{\Glsentryprefixplural}[1]{%
7587   \protected@edef\@glo@text{\csname glo@#1@prefixplural\endcsname}%
7588   \xmakefirstuc\@glo@text
7589 }

```

Define commands to determine if the prefix keys have been set:

\ifglshasprefix

```
7590 \newcommand*{\ifglshasprefix}[3]{%
7591   \ifcempty{glo@#1@prefix}%
7592   {#3}%
7593   {#2}%
7594 }
```

hasprefixplural

```
7595 \newcommand*{\ifglshasprefixplural}[3]{%
7596   \ifcempty{glo@#1@prefixplural}%
7597   {#3}%
7598   {#2}%
7599 }
```

shasprefixfirst

```
7600 \newcommand*{\ifglshasprefixfirst}[3]{%
7601   \ifcempty{glo@#1@prefixfirst}%
7602   {#3}%
7603   {#2}%
7604 }
```

efixfirstplural

```
7605 \newcommand*{\ifglshasprefixfirstplural}[3]{%
7606   \ifcempty{glo@#1@prefixfirstplural}%
7607   {#3}%
7608   {#2}%
7609 }
```

Define commands that insert the prefix before commands like \gls:

\pgls

```
7610 \newrobustcmd{\pgls}{\@gls@hyp@opt\@pgls}
```

\@pgls Unstarred version.

```
7611 \newcommand*{\@pgls}[2][ ]{%
7612   \new@ifnextchar[%
7613   {\@pgls@{#1}{#2}}%
7614   {\@pgls@{#1}{#2}[ ]}%
7615 }
```

\@pgls@ Read in the final optional argument:

```
7616 \def\@pgls@#1#2[#3]{%
7617   \glsdoifexists{#2}%
7618   {%
7619     \ifglsused{#2}%
7620     {%
7621       \glsentryprefix{#2}%
7622     }%

```

```

7623     {%
7624     \glsentryprefixfirst{#2}%
7625     }%
7626     \@gls@{#1}{#2}[#3]%
7627     }%
7628 }

```

Similarly for the plural version:

```

\pglsp1
7629 \newrobustcmd{\pglsp1}{\@gls@hyp@opt\@pglsp1}

```

\@pglsp1 Unstarred version.

```

7630 \newcommand*{\@pglsp1}[2][ ]{%
7631   \new@ifnextchar[%
7632   {\@pglsp1@{#1}{#2}}%
7633   {\@pglsp1@{#1}{#2}[ ]}%
7634 }

```

\@pglsp1@ Read in the final optional argument:

```

7635 \def\@pglsp1@#1#2[#3]{%
7636   \glsdoifexists{#2}%
7637   {%
7638     \ifglsused{#2}%
7639     {%
7640       \glsentryprefixplural{#2}%
7641     }%
7642     {%
7643       \glsentryprefixfirstplural{#2}%
7644     }%
7645     \@glspl@{#1}{#2}[#3]%
7646   }%
7647 }

```

Now for the first letter upper case versions:

```

\Pgls
7648 \newrobustcmd{\Pgls}{\@gls@hyp@opt\@Pgls}

```

\@Pgls Unstarred version.

```

7649 \newcommand*{\@Pgls}[2][ ]{%
7650   \new@ifnextchar[%
7651   {\@Pgls@{#1}{#2}}%
7652   {\@Pgls@{#1}{#2}[ ]}%
7653 }

```

\@Pgls@ Read in the final optional argument:

```

7654 \def\@Pgls@#1#2[#3]{%

```

```

7655 \glsdoifexists{#2}%
7656 {%
7657   \ifglsused{#2}%
7658   {%
7659     \ifglshasprefix{#2}%
7660     {%
7661       \Glsentryprefix{#2}%
7662       \@gls@{#1}{#2}[#3]%
7663     }%
7664     {\@Gls@{#1}{#2}[#3]}%
7665   }%
7666   {%
7667     \ifglshasprefixfirst{#2}%
7668     {%
7669       \Glsentryprefixfirst{#2}%
7670       \@gls@{#1}{#2}[#3]%
7671     }%
7672     {\@Gls@{#1}{#2}[#3]}%
7673   }%
7674 }%
7675 }

```

Similarly for the plural version:

```

\Pglspl
7676 \newrobustcmd{\Pglspl}{\@gls@hyp@opt\Pglspl}

```

\@Pglspl Unstarred version.

```

7677 \newcommand*{\@Pglspl}[2] [] {%
7678   \new@ifnextchar[%
7679   {\@Pglspl@{#1}{#2}}%
7680   {\@Pglspl@{#1}{#2} []}%
7681 }

```

\@Pglspl@ Read in the final optional argument:

```

7682 \def\@Pglspl@#1#2[#3] {%
7683   \glsdoifexists{#2}%
7684   {%
7685     \ifglsused{#2}%
7686     {%
7687       \ifglshasprefixplural{#2}%
7688       {%
7689         \Glsentryprefixplural{#2}%
7690         \@glspl@{#1}{#2}[#3]%
7691       }%
7692       {\@Glspl@{#1}{#2}[#3]}%
7693     }%
7694     {%
7695       \ifglshasprefixfirstplural{#2}%

```

```

7696      {%
7697      \Glsentryprefixfirstplural{#2}%
7698      \@glsp1@{#1}{#2}[#3]%
7699      }%
7700      {\@Glspl@{#1}{#2}[#3]}%
7701      }%
7702  }%
7703 }

```

Finally the all upper case versions:

\PGLS

```

7704 \newrobustcmd{\PGLS}{\@gls@hyp@opt\PGLS}

```

\@PGLS Unstarred version.

```

7705 \newcommand*{\@PGLS}[2][{}]{%
7706   \new@ifnextchar[%
7707   {\@PGLS@{#1}{#2}}%
7708   {\@PGLS@{#1}{#2}[]}%
7709 }

```

\@PGLS@ Read in the final optional argument:

```

7710 \def\@PGLS@#1#2[#3]{%
7711   \glsdoifexists{#2}%
7712   {%
7713     \ifglsused{#2}%
7714     {%
7715       \mfirstucMakeUppercase{\glsentryprefix{#2}}%
7716     }%
7717     {%
7718       \mfirstucMakeUppercase{\glsentryprefixfirst{#2}}%
7719     }%
7720     \@GLS@{#1}{#2}[#3]%
7721   }%
7722 }

```

Plural version:

\PGLSp1

```

7723 \newrobustcmd{\PGLSp1}{\@gls@hyp@opt\PGLSp1}

```

\@PGLSp1 Unstarred version.

```

7724 \newcommand*{\@PGLSp1}[2][{}]{%
7725   \new@ifnextchar[%
7726   {\@PGLSp1@{#1}{#2}}%
7727   {\@PGLSp1@{#1}{#2}[]}%
7728 }

```

\@PGLSp1@ Read in the final optional argument:

```
7729 \def\@PGLSp1@#1#2[#3]{%
7730   \glsdoifexists{#2}%
7731   {%
7732     \ifglsused{#2}%
7733     {%
7734       \mfirstucMakeUppercase{\glsentryprefixplural{#2}}%
7735     }%
7736     {%
7737       \mfirstucMakeUppercase{\glsentryprefixfirstplural{#2}}%
7738     }%
7739     \@GLSp1@{#1}{#2}[#3]%
7740   }%
7741 }
```

3 Glossary Styles

3.1 Glossary hyper-navigation definitions (glossary-hypernav package)

Package Definition:

```
7742 \ProvidesPackage{glossary-hypernav}[2016/04/30 v4.23 (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`

```
7743 \newcommand*{\glsnavhyperlink}[3][\@glo@type]{%
7744   \edef\gls@grplabel{#2}\protected@edef\gls@grptitle{#3}%
7745   \@glslink{glsn:#1@#2}{#3}}
```

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

This command makes `<text>` a hypertarget for the glossary group whose label is given by `<label>` in the glossary given by `<type>`. If `<type>` is omitted, `\@glo@type` is used which is set by `\printglossary` to the current glossary label.

`glsnavhypertarget`

```
7746 \newcommand*{\glsnavhypertarget}[3][\@glo@type]{%
  Add this group to the aux file for re-run check.
7747   \protected@write\@auxout{}{\string\@gls@hypergroup{#1}{#2}}%
  Add the target.
7748   \@glstarget{glsn:#1@#2}{#3}%
  Check list of know groups to determine if a re-run is required.
7749   \expandafter\let
7750     \expandafter\@gls@list\csname @gls@hypergroup@#1\endcsname
  Iterate through list and terminate loop if this group is found.
7751   \@for\@gls@elem:=\@gls@list\do{%
7752     \ifthenelse{equal{\@gls@elem}{#2}}{\@endfortrue}{}%}
```


Check if list terminated prematurely.

```
7753 \if@endfor
7754 \else
```

This group was not included in the list, so issue a warning.

```
7755 \GlossariesWarningNoLine{Navigation panel
7756     for glossary type ‘#1’^^Jmissing group ‘#2’}%
7757 \gdef\gls@hypergrouprerun{%
7758     \GlossariesWarningNoLine{Navigation panel
7759     has changed. Rerun LaTeX}}%
7760 \fi
7761 }
```

hypergrouprerun Give a warning at the end if re-run required

```
7762 \let\gls@hypergrouprerun\relax
7763 \AtEndDocument{\gls@hypergrouprerun}
```

@gls@hypergroup This adds to (or creates) the command \@gls@hypergroup@list@<glossary type> which lists all groups for a given glossary, so that the navigation bar only contains those groups that are present. However it requires at least 2 runs to ensure the information is up-to-date.

```
7764 \newcommand*{\@gls@hypergroup}[2]{%
7765 \ifundefined{\@gls@hypergroup@list@#1}{%
7766     \expandafter\xdef\csname @gls@hypergroup@list@#1\endcsname{#2}%
7767 }{%
7768     \expandafter\let\expandafter\@gls@tmp
7769     \csname @gls@hypergroup@list@#1\endcsname
7770     \expandafter\xdef\csname @gls@hypergroup@list@#1\endcsname{%
7771         \@gls@tmp,#2}%
7772 }%
7773 }
```

The \glsnavigation command displays a simple glossary group navigation. The symbol and number elements are defined separately, so that they can be suppressed if need be. Note that this command will produce a link to all 28 groups, but some groups may not be defined if there are groups that do not contain any terms, in which case you will get an undefined hyperlink warning. Now for the whole navigation bit:

\glsnavigation

```
7774 \newcommand*{\glsnavigation}{%
7775 \def\@gls@between{}%
7776 \ifcsundef{\@gls@hypergroup@list@\@glo@type}%
7777 {%
7778     \def\@gls@list{}%
7779 }%
7780 {%
7781     \expandafter\let\expandafter\@gls@list
7782     \csname @gls@hypergroup@list@\@glo@type\endcsname
7783 }%
7784 \@for\@gls@tmp:=\@gls@list\do{%
```

```

7785 \gls@between
7786 \gls@getgrouptitle{\gls@tmp}{\gls@grptitle}%
7787 \glsnavhyperlink{\gls@tmp}{\gls@grptitle}%
7788 \let\gls@between\glshypernavsep
7789 }%
7790 }

```

`\glshypernavsep` Separator for the hyper navigation bar.

```

7791 \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`

```

7792 \newcommand*{\glssymbolnav}{%
7793 \glsnavhyperlink{glssymbols}{\glsgetgrouptitle{glssymbols}}%
7794 \glshypernavsep
7795 \glsnavhyperlink{glsnumbers}{\glsgetgrouptitle{glsnumbers}}%
7796 \glshypernavsep
7797 }

```

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.

```

7798 \ProvidesPackage{glossary-inline}[2016/04/30 v4.23 (NLCT)]

```

`inline` Define the inline style.

```

7799 \newglossarystyle{inline}{%
    Start of glossary sets up first empty separator between entries. (This is then changed by
    \glossentry)
7800 \renewenvironment{theglossary}%
7801     {%
7802         \def\gls@inlinesep{}%
7803         \def\gls@inlinesubsep{}%
7804         \def\gls@inlinepostchild{}%
7805     }%
7806     {\glspostinline}%
    No header:
7807 \renewcommand*{\glossaryheader}{}%
    No group headings (if heading is required, add \glsinlinedopostchild to start definition
    in case heading follows a child entry):
7808 \renewcommand*{\glsgroupheading}[1]{}%

```

Just display separator followed by name and description:

```

7809 \renewcommand{\glossentry}[2]{%
7810   \glsinlinedopostchild
7811   \gls@inlinesep
7812   \glsentryitem{##1}%
7813   \glsinlinenameformat{##1}{%
7814     \glossentryname{##1}%
7815   }%
7816   \ifglshasdescsuppressed{##1}%
7817   {%
7818     \glsinlineemptydescformat
7819     {%
7820       \glossentrysymbol{##1}%
7821     }%
7822     {%
7823       ##2%
7824     }%
7825   }%
7826   {%
7827     \ifglshasdesc{##1}%
7828     {\glsinlinedescformat{\glossentrydesc{##1}}{\glossentrysymbol{##1}}{##2}}%
7829     {\glsinlineemptydescformat{\glossentrysymbol{##1}}{##2}}%
7830   }%
7831   \ifglshaschildren{##1}%
7832   {%
7833     \glsresetsubentrycounter
7834     \glsinlineparentchildseparator
7835     \def\gls@inlinesubsep{%
7836       \def\gls@inlinepostchild{\glsinlinepostchild}%
7837     }%
7838     {}%
7839     \def\gls@inlinesep{\glsinlineseparator}%
7840   }%

```

Sub-entries display description:

```

7841 \renewcommand{\subglossentry}[3]{%
7842   \gls@inlinesubsep%
7843   \glsinlinesubnameformat{##2}{%
7844     \glossentryname{##2}%
7845   }%
7846   \glsinlinesubdescformat{\glossentrydesc{##2}}{\glossentrysymbol{##2}}{##3}%
7847   \def\gls@inlinesubsep{\glsinlinesubseparator}%
7848 }%

```

Nothing special between groups:

```

7849 \renewcommand*{\glsgroupskip}{}%
7850 }

```

linedopostchild

```

7851 \newcommand*{\glsinlinedopostchild}{%

```

```

7852 \gls@inlinepostchild
7853 \def\gls@inlinepostchild{}%
7854 }

```

`inlineseparator` Separator to use between entries.

```
7855 \newcommand*{\glsinlineseparator}{;\space}
```

`inlinesubseparator` Separator to use between sub-entries.

```
7856 \newcommand*{\glsinlinesubseparator}{,\space}
```

`parentchildseparator` Separator to use between parent and children.

```
7857 \newcommand*{\glsinlineparentchildseparator}{:\space}
```

`inlinepostchild` Hook to use between child and next entry

```
7858 \newcommand*{\glsinlinepostchild}{}
```

`\glspostinline` Terminator for inline glossary.

```
7859 \newcommand*{\glspostinline}{\glspostdescription\space}
```

`inlinenameformat` Formats the name of the entry (first argument label, second argument name):

```
7860 \newcommand*{\glsinlinenameformat}[2]{\glstarget{#1}{#2}}
```

`inlinedescformat` Formats the entry's description, symbol and location list:

```
7861 \newcommand*{\glsinlinedescformat}[3]{\space#1}
```

`emptydescformat` Formats the entry's symbol and location list when the description is empty:

```
7862 \newcommand*{\glsinlineemptydescformat}[2]{}
```

`inlinesubnameformat` Formats the name of the subentry (first argument label, second argument name):

```
7863 \newcommand*{\glsinlinesubnameformat}[2]{\glstarget{#1}{}}
```

`inlinesubdescformat` Formats the subentry's description, symbol and location list:

```
7864 \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.

```
7865 \ProvidesPackage{glossary-list}[2016/04/30 v4.23 (NLCT)]
```

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```

7866 \providecommand{\indexspace}{%
7867 \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
7868 }

```

tgrouphaderfmt Provide a way of adjusting the format of the group headings.

```
7869 \newcommand*{\glslistgrouphaderfmt}[1]{#1}
```

tnavigationitem Provide a way of adjusting the format of the navigation header. This puts the navigation line inside the optional argument of item to prevent unwanted space occurring at the start, but this can cause a problem if the navigation line is too long. With this command, it makes it easier for the user to customise the style without having to remember to modify \glossaryheader after the style has been set.

```
7870 \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.

```
7871 \newglossarystyle{list}{%
```

Use description environment:

```
7872 \renewenvironment{theglossary}%
```

```
7873 {\begin{description}}{\end{description}}%
```

No header at the start of the environment:

```
7874 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
7875 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries start a new item in the list:

```
7876 \renewcommand*{\glossentry}[2]{%
```

```
7877 \item[\glsentryitem{##1}]%
```

```
7878 \glstarget{##1}{\glossentryname{##1}}]
```

```
7879 \glossentrydesc{##1}\glspostdescription\space ##2}%
```

Sub-entries continue on the same line:

```
7880 \renewcommand*{\subglossentry}[3]{%
```

```
7881 \glssubentryitem{##2}%
```

```
7882 \glstarget{##2}{\strut}\space
```

```
7883 \glossentrydesc{##2}\glspostdescription\space ##3.}%
```

```
7884 % \end{macrocode}
```

```
7885 % Add vertical space between groups:
```

```
7886 %\changes{3.03}{2012/09/21}{added check for glsnogroupskip}
```

```
7887 % \begin{macrocode}
```

```
7888 \renewcommand*{\glsgroupskip}{\ifglsgroupskip\else\indexspace\fi}%
```

```
7889 }
```

listgroup The listgroup style is like the list style, but the glossary groups have headings.

```
7890 \newglossarystyle{listgroup}{%
```

Base it on the list style:

```
7891 \setglossarystyle{list}%
```

Each group has a heading:

```
7892 \renewcommand*{\glsgroupheading}[1]{%
7893   \item[\glslistgroupheaderfmt{\glsgrouptitle{##1}}]}
```

listhypergroup The listhypergroup style is like the listgroup style, but has a set of links to the groups at the start of the glossary.

```
7894 \newglossarystyle{listhypergroup}{%
```

Base it on the list style:

```
7895 \setglossarystyle{list}%
```

Add navigation links at the start of the environment.

```
7896 \renewcommand*{\glossaryheader}{%
7897   \glslstnavigationitem{\glslnavigation}}%
```

Each group has a heading with a hypertarget:

```
7898 \renewcommand*{\glsgroupheading}[1]{%
7899   \item[\glslistgroupheaderfmt
7900         {\glslnavhypertarget{##1}{\glsgrouptitle{##1}}}]}
```

altlist The altlist glossary style is like the list style, but places the description on a new line. Sub-entries follow in separate paragraphs without the sub-entry name. This style does not use the entry's symbol.

```
7901 \newglossarystyle{altlist}{%
```

Base it on the list style:

```
7902 \setglossarystyle{list}%
```

Main (level 0) entries start a new item in the list with a line break after the entry name:

```
7903 \renewcommand*{\glossentry}[2]{%
7904   \item[\glssentryitem{##1}%
7905         \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.

```
7906   \mbox{}\par\nobreak\@afterheading
7907   \glossentrydesc{##1}\glspostdescription\space ##2}%
```

Sub-entries start a new paragraph:

```
7908 \renewcommand{\subglossentry}[3]{%
7909   \par
7910   \glssubentryitem{##2}%
7911   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space ##3}%
7912 }
```

altlistgroup The altlistgroup glossary style is like the altlist style, but the glossary groups have headings.

```
7913 \newglossarystyle{altlistgroup}{%
```

Base it on the altlist style:

```
7914 \setglossarystyle{altlist}%
```

Each group has a heading:

```
7915 \renewcommand*{\glsgroupheading}[1]{%
7916 \item[\glslistgroupheaderfmt{\glsgetgrouptitle{##1}}]}
```

`altlisthypergroup` The `altlisthypergroup` glossary style is like the `altlistgroup` style, but has a set of links to the groups at the start of the glossary.

```
7917 \newglossarystyle{altlisthypergroup}{%
```

Base it on the `altlist` style:

```
7918 \setglossarystyle{altlist}%
```

Add navigation links at the start of the environment.

```
7919 \renewcommand*{\glossaryheader}{%
7920 \glslistnavigationitem{\glsnavigation}}%
```

Each group has a heading with a `hypertarget`:

```
7921 \renewcommand*{\glsgroupheading}[1]{%
7922 \item[\glslistgroupheaderfmt
7923 {\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.

```
7924 \newglossarystyle{listdotted}{%
```

Base it on the `list` style:

```
7925 \setglossarystyle{list}%
```

Each main (level 0) entry starts a new item:

```
7926 \renewcommand*{\glossentry}[2]{%
7927 \item[\makebox[\glslistdottedwidth][l]{%
7928 \glsentryitem{##1}%
7929 \glstarget{##1}{\glossentryname{##1}}}%
7930 \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##1}}%
```

Sub entries have the same format as main entries:

```
7931 \renewcommand*{\subglossentry}[3]{%
7932 \item[\makebox[\glslistdottedwidth][l]{%
7933 \glssubentryitem{##2}%
7934 \glstarget{##2}{\glossentryname{##2}}}%
7935 \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##2}}%
7936 }
```

`listdottedwidth`

```
7937 \newlength\glslistdottedwidth
7938 \setlength{\glslistdottedwidth}{.5\hspace}
```

`sublistdotted` This style is similar to the `glostylelistdotted` style, except that the main entries just have the name displayed.

```
7939 \newglossarystyle{sublistdotted}{%
```

Base it on the listdotted style:

```
7940 \setglossarystyle{listdotted}%
```

Main (level 0) entries just display the name:

```
7941 \renewcommand*{\glossentry}[2]{%
7942   \item[\glentryitem{##1}\glstarget{##1}{\glossentryname{##1}}}%
7943 }
```

3.4 Glossary Styles using longtable (the glossary-long package)

The glossary styles defined in the package used the longtable environment in the glossary.

```
7944 \ProvidesPackage{glossary-long}[2016/04/30 v4.23 (NLCT)]
```

Requires the package:

```
7945 \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`.)

```
7946 \@ifundefined{glsdescwidth}{%
7947   \newlength{glsdescwidth}
7948   \setlength{glsdescwidth}{0.6\hsize}
7949 }{}
```

`\glspagelistwidth` This is a length that governs the width of the page list column.

```
7950 \@ifundefined{glspagelistwidth}{%
7951   \newlength{glspagelistwidth}
7952   \setlength{glspagelistwidth}{0.1\hsize}
7953 }{}
```

`long` The long glossary style command which uses the longtable environment:

```
7954 \newglossarystyle{long}{%
```

Use longtable with two columns:

```
7955 \renewenvironment{theglossary}%
7956   {\begin{longtable}[lp{glsdescwidth}]}%
7957   {\end{longtable}}%
```

Do nothing at the start of the environment:

```
7958 \renewcommand*{\glossaryheader}{}%
```

No heading between groups:

```
7959 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries displayed in a row:

```
7960 \renewcommand{\glossentry}[2]{%
7961   \glentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
7962   \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline
7963 }%
```


Sub entries displayed on the following row without the name:

```
7964 \renewcommand{\subglossentry}[3]{%
7965     &
7966     \glssubentryitem{##2}%
7967     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
7968     ##3\tabularnewline
7969 }%
```

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>)

```
7970 \ifglsgroupskip
7971 \renewcommand*{\glsgroupskip}{}%
7972 \else
7973 \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
7974 \fi
7975 }
```

longborder The longborder style is like the above, but with horizontal and vertical lines:

```
7976 \newglossarystyle{longborder}{%
```

Base it on the glostylelong style:

```
7977 \setglossarystyle{long}%
```

Use longtable with two columns with vertical lines between each column:

```
7978 \renewenvironment{theglossary}{%
7979 \begin{longtable}{|l|p{\glsgdescwidth}|}{\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
7980 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
7981 }
```

longheader The longheader style is like the long style but with a header:

```
7982 \newglossarystyle{longheader}{%
```

Base it on the glostylelong style:

```
7983 \setglossarystyle{long}%
```

Set the table's header:

```
7984 \renewcommand*{\glossaryheader}{%
7985 \bfseries \entryname & \bfseries \descriptionname\tabularnewline\endhead}%
7986 }
```

longheaderborder The longheaderborder style is like the long style but with a header and border:

```
7987 \newglossarystyle{longheaderborder}{%
```

Base it on the glostylelongborder style:

```
7988 \setglossarystyle{longborder}%
```

Set the table's header and add horizontal line to table's foot:

```
7989 \renewcommand*{\glossaryheader}{%
7990 \hline\bfseries \entryname & \bfseries
7991 \descriptionname\tabularnewline\hline
```

```

7992 \endhead
7993 \hline\endfoot}%
7994 }

```

long3col The long3col style is like long but with 3 columns

```

7995 \newglossarystyle{long3col}{%
    Use a longtable with 3 columns:
7996 \renewenvironment{theglossary}%
7997 {\begin{longtable}{lp{\glstdescwidth}p{\glspagelistwidth}}}%
7998 {\end{longtable}}%

```

No table header:

```

7999 \renewcommand*\glossaryheader{}%

```

No headings between groups:

```

8000 \renewcommand*\glsgroupheading[1]{}%

```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```

8001 \renewcommand{\glossentry}[2]{%
8002 \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8003 \glossentrydesc{##1} & ##2\tabularnewline
8004 }%

```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```

8005 \renewcommand{\subglossentry}[3]{%
8006 &
8007 \glssubentryitem{##2}%
8008 \glstarget{##2}{\strut}\glossentrydesc{##2} &
8009 ##3\tabularnewline
8010 }%

```

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>)

```

8011 \ifglsgroupskip
8012 \renewcommand*\glsgroupskip{}%
8013 \else
8014 \renewcommand*\glsgroupskip{{ & & \tabularnewline}%
8015 \fi
8016 }

```

long3colborder The long3colborder style is like the long3col style but with a border:

```

8017 \newglossarystyle{long3colborder}{%
    Base it on the glostylelong3col style:
8018 \setglossarystyle{long3col}%
    Use a longtable with 3 columns with vertical lines around them:
8019 \renewenvironment{theglossary}%
8020 {\begin{longtable}{|l|p{\glstdescwidth}|p{\glspagelistwidth}|}%
8021 {\end{longtable}}%

```

Place horizontal lines at the head and foot of the table:

```
8022 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8023 }
```

`long3colheader` The `long3colheader` style is like `long3col` but with a header row:

```
8024 \newglossarystyle{long3colheader}{%
    Base it on the glostylelong3col style:
8025 \setglossarystyle{long3col}%
    Set the table's header:
8026 \renewcommand*{\glossaryheader}{%
8027 \bfseries\entryname&\bfseries\descriptionname&
8028 \bfseries\pagelistname\tabularnewline\endhead}%
8029 }
```

`colheaderborder` The `long3colheaderborder` style is like the above but with a border

```
8030 \newglossarystyle{long3colheaderborder}{%
    Base it on the glostylelong3colborder style:
8031 \setglossarystyle{long3colborder}%
    Set the table's header and add horizontal line at table's foot:
8032 \renewcommand*{\glossaryheader}{%
8033 \hline
8034 \bfseries\entryname&\bfseries\descriptionname&
8035 \bfseries\pagelistname\tabularnewline\hline\endhead
8036 \hline\endfoot}%
8037 }
```

`long4col` The `long4col` style has four columns where the third column contains the value of the associated symbol key.

```
8038 \newglossarystyle{long4col}{%
    Use a longtable with 4 columns:
8039 \renewenvironment{theglossary}%
8040 {\begin{longtable}{l1l1l}}%
8041 {\end{longtable}}%
    No table header:
8042 \renewcommand*{\glossaryheader}{}%
    No group headings:
8043 \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):
8044 \renewcommand{\glossentry}[2]{%
8045 \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8046 \glossentrydesc{##1} &
8047 \glossentrysymbol{##1} &
8048 ##2\tabularnewline
8049 }%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
8050 \renewcommand{\subglossentry}[3]{%
8051   &
8052   \glssubentryitem{##2}%
8053   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8054   \glossentrysymbol{##2} & ##3\tabularnewline
8055 }%
```

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>)

```
8056 \ifglsgroupskip
8057   \renewcommand*{\glsgroupskip}{}%
8058 \else
8059   \renewcommand*{\glsgroupskip}{ & & \tabularnewline}%
8060 \fi
8061 }
```

long4colheader The long4colheader style is like long4col but with a header row.

```
8062 \newglossarystyle{long4colheader}{%
```

Base it on the glostylelong4col style:

```
8063 \setglossarystyle{long4col}{%
```

Table has a header:

```
8064 \renewcommand*{\glossaryheader}{%
8065   \bfseries\entryname&\bfseries\descriptionname&
8066   \bfseries \symbolname&
8067   \bfseries\pagelistname\tabularnewline\endhead}%
8068 }
```

long4colborder The long4colborder style is like long4col but with a border.

```
8069 \newglossarystyle{long4colborder}{%
```

Base it on the glostylelong4col style:

```
8070 \setglossarystyle{long4col}{%
```

Use a longtable with 4 columns surrounded by vertical lines:

```
8071 \renewenvironment{theglossary}%
8072   {\begin{longtable}{|l|l|l|l|}}%
8073   {\end{longtable}}%
```

Add horizontal lines to the head and foot of the table:

```
8074 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8075 }
```

colheaderborder The long4colheaderborder style is like the above but with a border.

```
8076 \newglossarystyle{long4colheaderborder}{%
```

Base it on the glostylelong4col style:

```
8077 \setglossarystyle{long4col}{%
```

Use a longtable with 4 columns surrounded by vertical lines:

```
8078 \renewenvironment{theglossary}%
8079   {\begin{longtable}{|l|l|l|l|}}%
8080   {\end{longtable}}%
```

Add table header and horizontal line at the table's foot:

```
8081 \renewcommand*{\glossaryheader}{%
8082   \hline\bfseries\entryname&\bfseries\descriptionname&
8083   \bfseries \symbolname&
8084   \bfseries\pagelistname\tabularnewline\hline\endhead
8085   \hline\endfoot}%
8086 }
```

altlong4col The altlong4col style is like the long4col style but can have multiline descriptions and page lists.

```
8087 \newglossarystyle{altlong4col}{%
```

Base it on the glostylelong4col style:

```
8088 \setglossarystyle{long4col}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8089 \renewenvironment{theglossary}%
8090   {\begin{longtable}{lp{\glsdescwidth}lp{\glspagelistwidth}}}%
8091   {\end{longtable}}%
8092 }
```

altlong4colheader The altlong4colheader style is like altlong4col but with a header row.

```
8093 \newglossarystyle{altlong4colheader}{%
```

Base it on the glostylelong4colheader style:

```
8094 \setglossarystyle{long4colheader}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8095 \renewenvironment{theglossary}%
8096   {\begin{longtable}{lp{\glsdescwidth}lp{\glspagelistwidth}}}%
8097   {\end{longtable}}%
8098 }
```

altlong4colborder The altlong4colborder style is like altlong4col but with a border.

```
8099 \newglossarystyle{altlong4colborder}{%
```

Base it on the glostylelong4colborder style:

```
8100 \setglossarystyle{long4colborder}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8101 \renewenvironment{theglossary}%
8102   {\begin{longtable}{|lp{\glsdescwidth}|lp{\glspagelistwidth}|}}%
8103   {\end{longtable}}%
8104 }
```

colheaderborder The altlong4colheaderborder style is like the above but with a header as well as a border.

```
8105 \newglossarystyle{altlong4colheaderborder}{%
```

Base it on the glostylelong4colheaderborder style:

```
8106 \setglossarystyle{long4colheaderborder}{%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8107 \renewenvironment{theglossary}{%
```

```
8108 {\begin{longtable}{|l|p{\glstdescwidth}|l|p{\glspagelistwidth}|}}%
```

```
8109 {\end{longtable}}%
```

```
8110 }
```

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>

```
8111 \ProvidesPackage{glossary-longbooktabs}[2016/04/30 v4.23 (NLCT)]
```

Requires booktabs package:

```
8112 \RequirePackage{booktabs}
```

and the base packages for long styles:

```
8113 \RequirePackage{glossary-long}
```

```
8114 \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.

```
8115 \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.

```
8116 \glspatchLToutput
```

As with the longheader style, use the long style as a base.

```
8117 \setglossarystyle{long}{%
```

Add a header with rules.

```
8118 \renewcommand*{\glossaryheader}{%
```

```
8119 \toprule \bfseries \entryname & \bfseries
```

```
8120 \descriptionname\tabularnewline\midrule\endhead
```

```
8121 \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.

```
8122 \ifglsnogroupskip
```

```

8123   \renewcommand*{\glsgroupskip}{}%
8124   \else
8125     \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
8126   \fi
8127 }

```

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.

```
8128 \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.

```
8129   \glspatchLToutput
```

Use the long3col style as a base.

```
8130   \setglossarystyle{long3col}{%
```

Add a header with rules.

```

8131   \renewcommand*{\glossaryheader}{%
8132     \toprule \bfseries \entryname &
8133     \bfseries \descriptionname &
8134     \bfseries \pagelistname
8135     \tabularnewline\midrule\endhead
8136     \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.

```

8137   \ifglsgroupskip
8138     \renewcommand*{\glsgroupskip}{}%
8139   \else
8140     \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
8141   \fi
8142 }

```

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.

```
8143 \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.

```
8144   \glspatchLToutput
```

Use the long4col style as a base.

```
8145   \setglossarystyle{long4col}{%
```

Add a header with rules.

```

8146   \renewcommand*{\glossaryheader}{%
8147     \toprule \bfseries \entryname &
8148     \bfseries \descriptionname &
8149     \bfseries \symbolname &

```

```

8150 \bfseries \pagelistname
8151 \tabularnewline\midrule\endhead
8152 \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.

```

8153 \ifglsgnogroupskip
8154 \renewcommand*{\glsgroupskip}{}%
8155 \else
8156 \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
8157 \fi
8158 }

```

`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.

```

8159 \newglossarystyle{altlong4col-booktabs}{%

```

If the style change is scoped, the patch will only have a local effect, which may be useful if it conflicts with other tables in the document.

```

8160 \glspatchLToutput

```

Use the `long4col-booktabs` style as a base.

```

8161 \setglossarystyle{long4col-booktabs}%

```

Change the column specifications:

```

8162 \renewenvironment{theglossary}%
8163 {\begin{longtable}{lp{\glsgdescwidth}lp{\glspagelistwidth}}}%
8164 {\end{longtable}}%
8165 }

```

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.

```

8166 \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.

```

8167 \glspatchLToutput

```

Use the `long-booktabs` style as a base.

```

8168 \setglossarystyle{long-booktabs}%

```

Adjust the column specification.

```

8169 \renewenvironment{theglossary}%
8170 {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}}}%
8171 {\end{longtable}}%
8172 }

```


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.

```
8173 \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.

```
8174 \glspatchLToutput
```

Use the long3col-booktabs style as a base.

```
8175 \setglossarystyle{long3col-booktabs}{%
```

Adjust the column specification.

```
8176 \renewenvironment{theglossary}{%
```

```
8177 {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}}%
```

```
8178 >{\raggedright}p{\glspagelistwidth}}}%
```

```
8179 {\end{longtable}}}%
```

```
8180 }
```

ed4col-booktabs The altlongragged4col-booktabs style is similar to the altlongragged4col style but uses the booktabs rules and patches longtable to check for group skip occurring at a page break.

```
8181 \newglossarystyle{altlongragged4col-booktabs}{%
```

If the style change is scoped, the patch will only have a local effect, which may be useful if it conflicts with other tables in the document.

```
8182 \glspatchLToutput
```

Use the altlong4col-booktabs style as a base.

```
8183 \setglossarystyle{altlong4col-booktabs}{%
```

Adjust the column specification.

```
8184 \renewenvironment{theglossary}{%
```

```
8185 {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}l%
```

```
8186 >{\raggedright}p{\glspagelistwidth}}}%
```

```
8187 {\end{longtable}}}%
```

```
8188 }
```

sLTpenaltycheck

```
8189 \newcommand*{\glsLTpenaltycheck}{%
```

```
8190 \ifnum\outputpenalty=-50\vskip-\normalbaselineskip\relax\fi
```

```
8191 }
```

enaltygroupskip

```
8192 \newcommand{\glspenaltygroupskip}{%
```

```
8193 \noalign{\penalty-50\vskip\normalbaselineskip}}
```

restoreLToutput Provide a way of restoring \LT@output for the user.

```
8194 \let\@gls@org@LT@output\LT@output
```

```
8195 \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

```

8196 \newcommand*{\glspatchLToutput}{%
8197   \renewcommand*{\LT@output}{%
8198     \ifnum\outputpenalty <-\@Mi
8199       \ifnum\outputpenalty > -\LT@end@pen
8200         \LT@err{floats and marginpars not allowed in a longtable}\@ehc
8201       \else
8202         \setbox\z@\vbox{\unvbox\@cclv}%
8203         \ifdim \ht\LT@lastfoot>\ht\LT@foot
8204           \dimen@ \pagegoal
8205           \advance\dimen@-\ht\LT@lastfoot
8206           \ifdim\dimen@<\ht\z@
8207             \setbox\@cclv\vbox{\unvbox\z@\copy\LT@foot\vss}%
8208             \@makecol
8209             \@outputpage
8210             \setbox\z@\vbox{\box\LT@head\glslTpenaltycheck}%
8211           \fi
8212         \fi
8213         \global\@colroom\@colht
8214         \global\vsizel\@colht
8215         {\unvbox\z@\box\ifvoid\LT@lastfoot\LT@foot\else\LT@lastfoot\fi}%
8216       \fi
8217     \else
8218       \setbox\@cclv\vbox{\unvbox\@cclv\copy\LT@foot\vss}%
8219       \@makecol
8220       \@outputpage
8221       \global\vsizel\@colroom
8222       \copy\LT@head
8223       \glslTpenaltycheck
8224       \nobreak
8225     \fi
8226   }%
8227 }
```

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.

```
8228 \ProvidesPackage{glossary-longragged}[2016/04/30 v4.23 (NLCT)]
```

Requires the package:

```
8229 \RequirePackage{array}
```

Requires the package:

```
8230 \RequirePackage{longtable}
```

`\glsdescwidth` This is a length that governs the width of the description column. This may have already been defined.

```

8231 \@ifundefined{glsdescwidth}{%
8232   \newlength{glsdescwidth
8233   \setlength{glsdescwidth}{0.6\hsize}
8234 }{}

```

`lspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```

8235 \@ifundefined{glspagelistwidth}{%
8236   \newlength{glspagelistwidth
8237   \setlength{glspagelistwidth}{0.1\hsize}
8238 }{}

```

`longragged` The longragged glossary style is like the long but uses ragged right formatting for the description column.

```

8239 \newglossarystyle{longragged}{%
    Use longtable with two columns:
8240   \renewenvironment{theglossary}%
8241     {\begin{longtable}{l>{\raggedright}p{glsdescwidth}}}%
8242     {\end{longtable}}%
    Do nothing at the start of the environment:
8243   \renewcommand*{\glossaryheader}{}%
    No heading between groups:
8244   \renewcommand*{\glsgroupheading}[1]{}%
    Main (level 0) entries displayed in a row:
8245   \renewcommand{\glossentry}[2]{%
8246     \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8247     \glossentrydesc{##1}\glspostdescription\space ##2%
8248     \tabularnewline
8249   }%
    Sub entries displayed on the following row without the name:
8250   \renewcommand{\subglossentry}[3]{%
8251     &
8252     \glssubentryitem{##2}%
8253     \glstarget{##2}{\strut}\glossentrydesc{##2}%
8254     \glspostdescription\space ##3%
8255     \tabularnewline
8256   }%
    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)
8257   \ifglsnogroupskip
8258     \renewcommand*{\glsgroupskip}{}%
8259   \else
8260     \renewcommand*{\glsgroupskip}{ & \tabularnewline}%

```

```

8261 \fi
8262 }

```

`longraggedborder` The `longraggedborder` style is like the above, but with horizontal and vertical lines:

```

8263 \newglossarystyle{longraggedborder}{%
    Base it on the glostylelongragged style:
8264 \setglossarystyle{longragged}%
    Use longtable with two columns with vertical lines between each column:
8265 \renewenvironment{theglossary}{%
8266 \begin{longtable}{|l|>{\raggedright}p{\glsgdescwidth}|}}%
8267 {\end{longtable}}%
    Place horizontal lines at the head and foot of the table:
8268 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8269 }

```

`longraggedheader` The `longraggedheader` style is like the `longragged` style but with a header:

```

8270 \newglossarystyle{longraggedheader}{%
    Base it on the glostylelongragged style:
8271 \setglossarystyle{longragged}%
    Set the table's header:
8272 \renewcommand*{\glossaryheader}{%
8273 \bfseries \entryname & \bfseries \descriptionname
8274 \tabularnewline\endhead}%
8275 }

```

`longraggedheaderborder` The `longraggedheaderborder` style is like the `longragged` style but with a header and border:

```

8276 \newglossarystyle{longraggedheaderborder}{%
    Base it on the glostylelongraggedborder style:
8277 \setglossarystyle{longraggedborder}%
    Set the table's header and add horizontal line to table's foot:
8278 \renewcommand*{\glossaryheader}{%
8279 \hline\bfseries \entryname & \bfseries \descriptionname
8280 \tabularnewline\hline
8281 \endhead
8282 \hline\endfoot}%
8283 }

```

`longragged3col` The `longragged3col` style is like `longragged` but with 3 columns

```

8284 \newglossarystyle{longragged3col}{%
    Use a longtable with 3 columns:
8285 \renewenvironment{theglossary}{%
8286 {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}%
8287 >{\raggedright}p{\glspagelistwidth}}}%
8288 {\end{longtable}}%

```

No table header:

```
8289 \renewcommand*\glossaryheader{}\%
```

No headings between groups:

```
8290 \renewcommand*\glsgroupheading[1]{}\%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
8291 \renewcommand\glossentry[2]{%
8292   \glentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8293   \glossentrydesc{##1} & ##2\tabularnewline
8294 }%
```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```
8295 \renewcommand\subglossentry[3]{%
8296   &
8297   \glssubentryitem{##2}%
8298   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8299   ##3\tabularnewline
8300 }%
```

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>)

```
8301 \ifglsgroupskip
8302 \renewcommand*\glsgroupskip{}\%
8303 \else
8304 \renewcommand*\glsgroupskip{\ & \tabularnewline}%
8305 \fi
8306 }
```

agged3colborder The longragged3colborder style is like the longragged3col style but with a border:

```
8307 \newglossarystyle{longragged3colborder}{%
```

Base it on the glostylelongragged3col style:

```
8308 \setglossarystyle{longragged3col}%
```

Use a longtable with 3 columns with vertical lines around them:

```
8309 \renewenvironment{theglossary}%
8310 {\begin{longtable}{|l|>{\raggedright}p{\glstdescwidth}|}%
8311 >{\raggedright}p{\glspagelistwidth}|}%
8312 {\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
8313 \renewcommand*\glossaryheader{\hline\endhead\hline\endfoot}%
8314 }
```

agged3colheader The longragged3colheader style is like longragged3col but with a header row:

```
8315 \newglossarystyle{longragged3colheader}{%
```

Base it on the glostylelongragged3col style:

```
8316 \setglossarystyle{longragged3col}%
```

Set the table's header:

```
8317 \renewcommand*{\glossaryheader}{%
8318   \bfseries\entryname&\bfseries\descriptionname&
8319   \bfseries\pagelistname\tabularnewline\endhead}%
8320 }
```

`colheaderborder` The `longragged3colheaderborder` style is like the above but with a border

```
8321 \newglossarystyle{longragged3colheaderborder}{%
```

Base it on the `glostylelongragged3colborder` style:

```
8322 \setglossarystyle{longragged3colborder}%
```

Set the table's header and add horizontal line at table's foot:

```
8323 \renewcommand*{\glossaryheader}{%
8324   \hline
8325   \bfseries\entryname&\bfseries\descriptionname&
8326   \bfseries\pagelistname\tabularnewline\hline\endhead
8327   \hline\endfoot}%
8328 }
```

`longragged4col` The `altlongragged4col` style is like the `altlong4col` style defined in the package, except that ragged right formatting is used for the description and page list columns.

```
8329 \newglossarystyle{altlongragged4col}{%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8330 \renewenvironment{theglossary}%
8331   {\begin{longtable}{1>{\raggedright}p{\glstdescwidth}1%
8332     >{\raggedright}p{\glspagelistwidth}}}%
8333   {\end{longtable}}%
```

No table header:

```
8334 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8335 \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):

```
8336 \renewcommand{\glossentry}[2]{%
8337   \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8338   \glossentrydesc{##1} & \glossentrysymbol{##1} &
8339   ##2\tabularnewline
8340   }%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
8341 \renewcommand{\subglossentry}[3]{%
8342   &
8343   \glssubentryitem{##2}%
8344   \glstarget{##2}{\strut}\glossentrydesc{##2} &
```

```

8345 \glossentrysymbol{##2} & ##3\tabularnewline
8346 }%

```

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>)

```

8347 \ifglsgroupskip
8348 \renewcommand*{\glsgroupskip}{}%
8349 \else
8350 \renewcommand*{\glsgroupskip}{ & & \tabularnewline}%
8351 \fi
8352 }

```

agged4colheader The altlongragged4colheader style is like altlongragged4col but with a header row.

```
8353 \newglossarystyle{altlongragged4colheader}{%
```

Base it on the glostylealtlongragged4col style:

```
8354 \setglossarystyle{altlongragged4col}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines
in each row:

```

8355 \renewenvironment{theglossary}%
8356 {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}l%
8357 >{\raggedright}p{\glspagelistwidth}}}%
8358 {\end{longtable}}%

```

Table has a header:

```

8359 \renewcommand*{\glossaryheader}{%
8360 \bfseries\entryname&\bfseries\descriptionname&
8361 \bfseries \symbolname&
8362 \bfseries\pagelistname\tabularnewline\endhead}%
8363 }

```

agged4colborder The altlongragged4colborder style is like altlongragged4col but with a border.

```
8364 \newglossarystyle{altlongragged4colborder}{%
```

Base it on the glostylealtlongragged4col style:

```
8365 \setglossarystyle{altlongragged4col}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines
in each row:

```

8366 \renewenvironment{theglossary}%
8367 {\begin{longtable}{|l|>{\raggedright}p{\glsdescwidth}|l|}%
8368 >{\raggedright}p{\glspagelistwidth}|}%
8369 {\end{longtable}}%

```

Add horizontal lines to the head and foot of the table:

```

8370 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8371 }

```

colheaderborder The altlongragged4colheaderborder style is like the above but with a header as well as a border.

```
8372 \newglossarystyle{altlongragged4colheaderborder}{%
```

Base it on the `glostylealtlongragged4col` style:

```
8373 \setglossarystyle{altlongragged4col}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8374 \renewenvironment{theglossary}%  
8375 {\begin{longtable}{|l|>{\raggedright}p{\glstdescwidth}|l|}%  
8376 >{\raggedright}p{\glspagelistwidth}|}}%  
8377 {\end{longtable}}%
```

Add table header and horizontal line at the table's foot:

```
8378 \renewcommand*{\glossaryheader}{%  
8379 \hline\bfseries\entryname&\bfseries\descriptionname&  
8380 \bfseries \symbolname&  
8381 \bfseries\pagelistname\tabularnewline\hline\endhead  
8382 \hline\endfoot}%  
8383 }
```

3.7 Glossary Styles using multicols (glossary-mcols.sty)

The style file defines glossary styles that use the `multicols` package. These use the tree-like glossary styles in a multicols environment.

```
8384 \ProvidesPackage{glossary-mcols}[2016/04/30 v4.23 (NLCT)]
```

Required packages:

```
8385 \RequirePackage{multicols}  
8386 \RequirePackage{glossary-tree}
```

`\indexspace` The are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```
8387 \providecommand{\indexspace}{%  
8388 \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax  
8389 }
```

`\glsmcols` Define macro in which to store the number of columns. (Defaults to 2.)

```
8390 \newcommand*{\glsmcols}{2}
```

`mcolindex` Multi-column index style. Same as the index, but puts the glossary in multiple columns. (Ideally the glossary title should go in the optional argument of `multicols`, but the title isn't part of the glossary style.)

```
8391 \newglossarystyle{mcolindex}{%  
8392 \setglossarystyle{index}%  
8393 \renewenvironment{theglossary}%  
8394 {%  
  
8395 \begin{multicols}{\glsmcols}  
8396 \setlength{\parindent}{0pt}%  
8397 \setlength{\parskip}{0pt plus 0.3pt}%
```



```

8398     \let\item\@idxitem}%
8399     {\end{multicols}}}%
8400 }

```

`mcindexgroup` As `mcindex` but has headings:

```

8401 \newglossarystyle{mcindexgroup}{%
8402   \setglossarystyle{mcindex}%
8403   \renewcommand*{\glsgroupheading}[1]{%
8404     \item\glstreegroupheaderfmt{\glsgrouptitle{##1}}\indexspace}%
8405 }

```

`indexhypergroup` The `mcindexhypergroup` style is like the `mcindexgroup` style but has hyper navigation.

```

8406 \newglossarystyle{mcindexhypergroup}{%

```

Base it on the `glostylemcindex` style:

```

8407   \setglossarystyle{mcindex}%

```

Put navigation links to the groups at the start of the glossary:

```

8408   \renewcommand*{\glossaryheader}{%
8409     \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.

```

8410   \renewcommand*{\glsgroupheading}[1]{%
8411     \item\glstreegroupheaderfmt
8412       {\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}%
8413     \indexspace}%
8414 }

```

`colindexspannav` Similar to `mcindexhypergroup`, but puts the navigation line in the optional argument of `multicols`.

```

8415 \newglossarystyle{colindexspannav}{%
8416   \setglossarystyle{index}%
8417   \renewenvironment{theglossary}%
8418     {%
8419       \begin{multicols}{\glsmcols}[\noindent\glstreenavigationfmt{\glsnavigation}]
8420       \setlength{\parindent}{0pt}%
8421       \setlength{\parskip}{0pt plus 0.3pt}%
8422       \let\item\@idxitem}%
8423     {\end{multicols}}}%

```

Add a heading for each group (with a target). The group's title is in bold followed by a vertical gap.

```

8424   \renewcommand*{\glsgroupheading}[1]{%
8425     \item\glstreegroupheaderfmt
8426       {\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}%
8427     \indexspace}%
8428 }

```

mcoltree Multi-column index style. Same as the tree, but puts the glossary in multiple columns.

```

8429 \newglossarystyle{mcoltree}{%
8430   \setglossarystyle{tree}%
8431   \renewenvironment{theglossary}%
8432   {%
8433     \begin{multicols}{\glsmcols}
8434     \setlength{\parindent}{0pt}%
8435     \setlength{\parskip}{0pt plus 0.3pt}%
8436   }%
8437   {\end{multicols}}%
8438 }
```

mcoltreegroup Like the mcoltree style but the glossary groups have headings.

```

8439 \newglossarystyle{mcoltreegroup}{%
      Base it on the glostylemcoltree style:
8440   \setglossarystyle{mcoltree}%
      Each group has a heading (in bold) followed by a vertical gap):
8441   \renewcommand{\glsgroupheading}[1]{\par
8442     \noindent\glstreegroupheaderfmt{\glsgrouptitle{##1}}\par\indexspace}%
8443 }
```

1treehypergroup The mcoltreehypergroup style is like the treegroup style, but has a set of links to the groups at the start of the glossary.

```

8444 \newglossarystyle{mcoltreehypergroup}{%
      Base it on the glostylemcoltree style:
8445   \setglossarystyle{mcoltree}%
      Put navigation links to the groups at the start of the theglossary environment:
8446   \renewcommand*{\glossaryheader}{%
8447     \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
      Each group has a heading (in bold with a target) followed by a vertical gap):
8448   \renewcommand*{\glsgroupheading}[1]{%
8449     \par\noindent
8450     \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}\par
8451     \indexspace}%
8452 }
```

mcoltreespannav Similar to the mcoltreehypergroup style but the navigation line is put in the optional argument of the multicols environment.

```

8453 \newglossarystyle{mcoltreespannav}{%
8454   \setglossarystyle{tree}%
8455   \renewenvironment{theglossary}%
8456   {%
```

```

8457     \begin{multicols}{\glsmcols}[\noindent\glstreenavigationfmt{\glsnavigation}]
8458     \setlength{\parindent}{0pt}%
8459     \setlength{\parskip}{0pt plus 0.3pt}%
8460 }%
8461 {\end{multicols}}%

```

Each group has a heading (in bold with a target) followed by a vertical gap):

```

8462 \renewcommand*{\glsgroupheading}[1]{%
8463   \par\noindent
8464   \glstreegroupheaderfmt{\glsnahypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8465   \indexspace}%
8466 }

```

mcoltreenoname Multi-column index style. Same as the **treenoname**, but puts the glossary in multiple columns.

```

8467 \newglossarystyle{mcoltreenoname}{%
8468   \setglossarystyle{treenoname}%
8469   \renewenvironment{theglossary}%
8470   {%
8471     \begin{multicols}{\glsmcols}
8472     \setlength{\parindent}{0pt}%
8473     \setlength{\parskip}{0pt plus 0.3pt}%
8474   }%
8475   {\end{multicols}}%
8476 }

```

treenonamegroup Like the **mcoltreenoname** style but the glossary groups have headings.

```

8477 \newglossarystyle{mcoltreenonamegroup}{%
  Base it on the glostylemcoltreenoname style:
8478   \setglossarystyle{mcoltreenoname}%
  Give each group a heading:
8479   \renewcommand{\glsgroupheading}[1]{\par
8480     \noindent\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}\par\indexspace}%
8481 }

```

onamehypergroup The **mcoltreenonamehypergroup** style is like the **mcoltreenonamegroup** style, but has a set of links to the groups at the start of the glossary.

```

8482 \newglossarystyle{mcoltreenonamehypergroup}{%
  Base it on the glostylemcoltreenoname style:
8483   \setglossarystyle{mcoltreenoname}%
  Put navigation links to the groups at the start of the theglossary environment:
8484   \renewcommand*{\glossaryheader}{%
8485     \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
  Each group has a heading (in bold with a target) followed by a vertical gap):
8486   \renewcommand*{\glsgroupheading}[1]{%
8487     \par\noindent

```

```

8488 \glstreegroupheaderfmt{\glshypertarget{##1}{\glsgrouptitle{##1}}}\par
8489 \indexspace}%
8490 }

```

reenonamespannav Similar to the `mcoltreenonamehypergroup` style but the navigation line is put in the optional argument of the `multicols` environment.

```

8491 \newglossarystyle{mcoltreenonamespannav}{%
8492 \setglossarystyle{treenoname}%
8493 \renewenvironment{theglossary}%
8494 {%
8495 \begin{multicols}{\glsmcols}[\noindent\glstreenavigationfmt{\glsnavigation}]
8496 \setlength{\parindent}{0pt}%
8497 \setlength{\parskip}{0pt plus 0.3pt}%
8498 }%
8499 {\end{multicols}}%

```

Each group has a heading (in bold with a target) followed by a vertical gap):

```

8500 \renewcommand*{\glsgroupheading}[1]{%
8501 \par\noindent
8502 \glstreegroupheaderfmt{\glshypertarget{##1}{\glsgrouptitle{##1}}}\par
8503 \indexspace}%
8504 }

```

mcolalmtree Multi-column index style. Same as the `almtree`, but puts the glossary in multiple columns.

```

8505 \newglossarystyle{mcolalmtree}{%
8506 \setglossarystyle{almtree}%
8507 \renewenvironment{theglossary}%
8508 {%
8509 \begin{multicols}{\glsmcols}
8510 \def\@gls@prevlevel{-1}%
8511 \mbox{}\par
8512 }%
8513 {\par\end{multicols}}%
8514 }

```

colalmtreegroup Like the `mcolalmtree` style but the glossary groups have headings.

```

8515 \newglossarystyle{colalmtreegroup}{%

```

Base it on the `glostylemcolalmtree` style:

```

8516 \setglossarystyle{mcolalmtree}%

```

Give each group a heading.

```

8517 \renewcommand{\glsgroupheading}[1]{\par
8518 \def\@gls@prevlevel{-1}%
8519 \hangindent0pt\relax
8520 \parindent0pt\relax
8521 \glstreegroupheaderfmt{\glsgrouptitle{##1}}\par\indexspace}%
8522 }

```

treehypergroup The mcolalttreehypergroup style is like the mcolalttreegroup style, but has a set of links to the groups at the start of the glossary.

```
8523 \newglossarystyle{mcolalttreehypergroup}{%
```

Base it on the glostylemcolalttree style:

```
8524 \setglossarystyle{mcolalttree}{%
```

Put the navigation links in the header

```
8525 \renewcommand*{\glossaryheader}{%
```

```
8526 \par
```

```
8527 \def\@gls@prevlevel{-1}%
```

```
8528 \hangindent0pt\relax
```

```
8529 \parindent0pt\relax
```

```
8530 \glstreenavigationfmt{\glsnavigation}\par\indexspace}%
```

Put a hypertarget at the start of each group

```
8531 \renewcommand*{\glsgroupheading}[1]{%
```

```
8532 \par
```

```
8533 \def\@gls@prevlevel{-1}%
```

```
8534 \hangindent0pt\relax
```

```
8535 \parindent0pt\relax
```

```
8536 \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
```

```
8537 \indexspace}%
```

```
8538 }
```

alttreespannav Similar to the mcolalttreehypergroup style but the navigation line is put in the optional argument of the multicols environment.

```
8539 \newglossarystyle{mcolalttreespannav}{%
```

```
8540 \setglossarystyle{alttree}{%
```

```
8541 \renewenvironment{theglossary}{%
```

```
8542 {%
```

```
8543 \begin{multicols}{\glsncols}[\noindent\glstreenavigationfmt{\glsnavigation}]
```

```
8544 \def\@gls@prevlevel{-1}%
```

```
8545 \mbox{}\par
```

```
8546 }%
```

```
8547 {\par\end{multicols}}}%
```

Put a hypertarget at the start of each group

```
8548 \renewcommand*{\glsgroupheading}[1]{%
```

```
8549 \par
```

```
8550 \def\@gls@prevlevel{-1}%
```

```
8551 \hangindent0pt\relax
```

```
8552 \parindent0pt\relax
```

```
8553 \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
```

```
8554 \indexspace}
```

```
8555 }
```

3.8 Glossary Styles using supertabular environment (glossary-super package)

The glossary styles defined in the package use the supertabular environment.

8556 \ProvidesPackage{glossary-super}[2016/04/30 v4.23 (NLCT)]

Requires the package:

8557 \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.

8558 \@ifundefined{glsdescwidth}{%

8559 \newlength{glsdescwidth

8560 \setlength{glsdescwidth}{0.6\hsize}

8561 }{}

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined if has been loaded.

8562 \@ifundefined{glspagelistwidth}{%

8563 \newlength{glspagelistwidth

8564 \setlength{glspagelistwidth}{0.1\hsize}

8565 }{}

`super` The super glossary style uses the supertabular environment (it uses lengths defined in the package.)

8566 \newglossarystyle{super}{%

Put the glossary in a supertabular environment with two columns and no head or tail:

8567 \renewenvironment{theglossary}{%

8568 {\tablehead{}\tabletail{}}%

8569 \begin{supertabular}{lp{glsdescwidth}}%

8570 {\end{supertabular}}%

Do nothing at the start of the table:

8571 \renewcommand*{\glossaryheader}{}%

No group headings:

8572 \renewcommand*{\glsgroupheading}[1]{}%

Main (level 0) entries put in a row (name in first column, description and page list in second column):

8573 \renewcommand{\glossentry}[2]{%

8574 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &

8575 \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline

8576 }%

Sub entries put in a row (no name, description and page list in second column):

8577 \renewcommand{\subglossentry}[3]{%

8578 &

8579 \glssubentryitem{##2}%

```

8580      \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
8581      ##3\tabularnewline
8582  }%

```

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>)

```

8583  \ifglsgroupskip
8584    \renewcommand*{\glsgroupskip}{}%
8585  \else
8586    \renewcommand*{\glsgroupskip}{& \tabularnewline}%
8587  \fi
8588 }

```

superborder The superborder style is like the above, but with horizontal and vertical lines:

```
8589 \newglossarystyle{superborder}{%
```

Base it on the glostylesuper style:

```
8590 \setglossarystyle{super}%
```

Put the glossary in a supertabular environment with two columns and a horizontal line in the head and tail:

```

8591 \renewenvironment{theglossary}%
8592   {\tablehead{\hline}\tabletail{\hline}%
8593    \begin{supertabular}{|l|p{\glsgdescwidth}|}%
8594    {\end{supertabular}}%
8595 }

```

superheader The superheader style is like the super style, but with a header:

```
8596 \newglossarystyle{superheader}{%
```

Base it on the glostylesuper style:

```
8597 \setglossarystyle{super}%
```

Put the glossary in a supertabular environment with two columns, a header and no tail:

```

8598 \renewenvironment{theglossary}%
8599   {\tablehead{\bfseries \entryname &
8600    \bfseries\descriptionname\tabularnewline}%
8601    \tabletail{}}%
8602   \begin{supertabular}{lp{\glsgdescwidth}}%
8603   {\end{supertabular}}%
8604 }

```

superheaderborder The superheaderborder style is like the super style but with a header and border:

```
8605 \newglossarystyle{superheaderborder}{%
```

Base it on the glostylesuper style:

```
8606 \setglossarystyle{super}%
```

Put the glossary in a supertabular environment with two columns, a header and horizontal lines above and below the table:

```
8607 \renewenvironment{theglossary}%
```

```

8608   {\tablehead{\hline\bfseries \entryname &
8609             \bfseries \descriptionname\tabularnewline\hline}%
8610     \tabletail{\hline}
8611     \begin{supertabular}{|l|p{\glstdescwidth}|}%
8612   {\end{supertabular}}%
8613 }

```

super3col The super3col style is like the super style, but with 3 columns:

```
8614 \newglossarystyle{super3col}{%
```

Put the glossary in a supertabular environment with three columns and no head or tail:

```

8615   \renewenvironment{theglossary}%
8616     {\tablehead{}\tabletail{}}%
8617     \begin{supertabular}{|lp{\glstdescwidth}p{\glspagelistwidth}|}%
8618     {\end{supertabular}}%

```

Do nothing at the start of the table:

```
8619   \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8620   \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```

8621   \renewcommand{\glossentry}[2]{%
8622     \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8623     \glossentrydesc{##1} & ##2\tabularnewline
8624   }%

```

Sub entries on a row (no name, description in second column, page list in last column):

```

8625   \renewcommand{\subglossentry}[3]{%
8626     &
8627     \glssubentryitem{##2}%
8628     \glstarget{##2}{\strut}\glossentrydesc{##2} &
8629     ##3\tabularnewline
8630   }%

```

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>)

```

8631   \ifglsgroupskip
8632     \renewcommand*{\glsgroupskip}{}%
8633   \else
8634     \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
8635   \fi
8636 }

```

super3colborder The super3colborder style is like the super3col style, but with a border:

```
8637 \newglossarystyle{super3colborder}{%
```

Base it on the glostylesuper3col style:

```
8638   \setglossarystyle{super3col}%

```


Put the glossary in a supertabular environment with three columns and a horizontal line in the head and tail:

```
8639 \renewenvironment{theglossary}%
8640 {\tablehead{\hline}\tabletail{\hline}%
8641 \begin{supertabular}{|l|p{\glsgdescwidth}|p{\glspagelistwidth}|}%
8642 {\end{supertabular}}}%
8643 }
```

super3colheader The super3colheader style is like the super3col style but with a header row:

```
8644 \newglossarystyle{super3colheader}{%
      Base it on the glostylessuper3col style:
8645 \setglossarystyle{super3col}%
      Put the glossary in a supertabular environment with three columns, a header and no tail:
8646 \renewenvironment{theglossary}%
8647 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8648 \bfseries\pagelistname\tabularnewline}\tabletail{}}%
8649 \begin{supertabular}{lp{\glsgdescwidth}p{\glspagelistwidth}}}%
8650 {\end{supertabular}}}%
8651 }
```

colheaderborder The super3colheaderborder style is like the super3col style but with a header and border:

```
8652 \newglossarystyle{super3colheaderborder}{%
      Base it on the glostylessuper3colborder style:
8653 \setglossarystyle{super3colborder}%
      Put the glossary in a supertabular environment with three columns, a header with horizontal
      lines and a horizontal line in the tail:
8654 \renewenvironment{theglossary}%
8655 {\tablehead{\hline
8656 \bfseries\entryname&\bfseries\descriptionname&
8657 \bfseries\pagelistname\tabularnewline\hline}%
8658 \tabletail{\hline}%
8659 \begin{supertabular}{|l|p{\glsgdescwidth}|p{\glspagelistwidth}|}%
8660 {\end{supertabular}}}%
8661 }
```

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.

```
8662 \newglossarystyle{super4col}{%
      Put the glossary in a supertabular environment with four columns and no head or tail:
8663 \renewenvironment{theglossary}%
8664 {\tablehead{}\tabletail{}}%
8665 \begin{supertabular}{llll}}}%
8666 \end{supertabular}}}%
      Do nothing at the start of the table:
8667 \renewcommand*{\glossaryheader}{}%

```

No group headings:

```
8668 \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:

```
8669 \renewcommand{\glossentry}[2]{%
8670   \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8671   \glossentrydesc{##1} &
8672   \glossentrysymbol{##1} & ##2\tabularnewline
8673 }%
```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```
8674 \renewcommand{\subglossentry}[3]{%
8675   &
8676   \glssubentryitem{##2}%
8677   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8678   \glossentrysymbol{##2} & ##3\tabularnewline
8679 }%
```

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>)

```
8680 \ifglsnogroupskip
8681 \renewcommand*\glsgroupskip{}{}%
8682 \else
8683 \renewcommand*\glsgroupskip{& & & \tabularnewline}%
8684 \fi
8685 }
```

super4colheader The super4colheader style is like the super4col but with a header row.

```
8686 \newglossarystyle{super4colheader}{%
```

Base it on the glostylesuper4col style:

```
8687 \setglossarystyle{super4col}%
```

Put the glossary in a supertabular environment with four columns, a header and no tail:

```
8688 \renewenvironment{theglossary}%
8689   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8690     \bfseries\symbolname &
8691     \bfseries\pagelistname\tabularnewline}%
8692   \tabletail{}}%
8693   \begin{supertabular}{l1111}}%
8694   {\end{supertabular}}%
8695 }
```

super4colborder The super4colborder style is like the super4col but with a border.

```
8696 \newglossarystyle{super4colborder}{%
```

Base it on the glostylesuper4col style:

```
8697 \setglossarystyle{super4col}%
```

Put the glossary in a supertabular environment with four columns and a horizontal line in the head and tail:

```
8698 \renewenvironment{theglossary}%
8699 {\tablehead{\hline}\tabletail{\hline}%
8700 \begin{supertabular}{|l|l|l|l|}%
8701 {\end{supertabular}}}%
8702 }
```

colheaderborder The super4colheaderborder style is like the super4col but with a header and border.

```
8703 \newglossarystyle{super4colheaderborder}{%
```

Base it on the glostylesuper4col style:

```
8704 \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:

```
8705 \renewenvironment{theglossary}%
8706 {\tablehead{\hline\bfseries\entryname&\bfseries\descriptionname&
8707 \bfseries\symbolname &
8708 \bfseries\pagelistname\tabularnewline\hline}%
8709 \tabletail{\hline}%
8710 \begin{supertabular}{|l|l|l|l|}%
8711 {\end{supertabular}}}%
8712 }
```

altsuper4col The altsuper4col glossary style is like super4col but has provision for multiline descriptions.

```
8713 \newglossarystyle{altsuper4col}{%
```

Base it on the glostylesuper4col style:

```
8714 \setglossarystyle{super4col}%
```

Put the glossary in a supertabular environment with four columns and no head or tail:

```
8715 \renewenvironment{theglossary}%
8716 {\tablehead{}\tabletail{}}%
8717 \begin{supertabular}{lp{\glsgdescwidth}lp{\glspagelistwidth}}%
8718 {\end{supertabular}}}%
8719 }
```

super4colheader The altsuper4colheader style is like the altsuper4col but with a header row.

```
8720 \newglossarystyle{altsuper4colheader}{%
```

Base it on the glostylesuper4colheader style:

```
8721 \setglossarystyle{super4colheader}%
```

Put the glossary in a supertabular environment with four columns, a header and no tail:

```
8722 \renewenvironment{theglossary}%
8723 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8724 \bfseries\symbolname &
8725 \bfseries\pagelistname\tabularnewline}\tabletail{}}%
8726 \begin{supertabular}{lp{\glsgdescwidth}lp{\glspagelistwidth}}%
8727 {\end{supertabular}}}%
8728 }
```

`super4colborder` The `altsuper4colborder` style is like the `altsuper4col` but with a border.

```
8729 \newglossarystyle{altsuper4colborder}{%
```

Base it on the `glostylesuper4colborder` style:

```
8730 \setglossarystyle{super4colborder}{%
```

Put the glossary in a `supertabular` environment with four columns and a horizontal line in the head and tail:

```
8731 \renewenvironment{theglossary}%  
8732 {\tablehead{\hline}\tabletail{\hline}%  
8733 \begin{supertabular}%  
8734 {lllp{\glsgdescwidth}lllp{\glspagelistwidth}}}%  
8735 {\end{supertabular}}}%  
8736 }
```

`colheaderborder` The `altsuper4colheaderborder` style is like the `altsuper4col` but with a header and border.

```
8737 \newglossarystyle{altsuper4colheaderborder}{%
```

Base it on the `glostylesuper4colheaderborder` style:

```
8738 \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:

```
8739 \renewenvironment{theglossary}%  
8740 {\tablehead{\hline  
8741 \bfseries\entryname &  
8742 \bfseries\descriptionname &  
8743 \bfseries\symbolname &  
8744 \bfseries\pagelistname\tabularnewline\hline}%  
8745 \tabletail{\hline}%  
8746 \begin{supertabular}%  
8747 {lllp{\glsgdescwidth}lllp{\glspagelistwidth}}}%  
8748 {\end{supertabular}}}%  
8749 }
```

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.

```
8750 \ProvidesPackage{glossary-superragged}[2016/04/30 v4.23 (NLCT)]
```

Requires the package:

```
8751 \RequirePackage{array}
```

Requires the package:

```
8752 \RequirePackage{supertabular}
```

`\glsdescwidth` This is a length that governs the width of the description column. This may already have been defined.

```

8753 \@ifundefined{glsdescwidth}{%
8754   \newlength{glsdescwidth
8755   \setlength{glsdescwidth}{0.6\hsize}
8756 }{}

```

`lspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```

8757 \@ifundefined{glspagelistwidth}{%
8758   \newlength{glspagelistwidth
8759   \setlength{glspagelistwidth}{0.1\hsize}
8760 }{}

```

`superragged` The superragged glossary style uses the supertabular environment.

```

8761 \newglossarystyle{superragged}{%

```

Put the glossary in a supertabular environment with two columns and no head or tail:

```

8762   \renewenvironment{theglossary}%
8763     {\tablehead{}\tabletail{}}%
8764     \begin{supertabular}{1>{\raggedright}p{glsdescwidth}}}%
8765     {\end{supertabular}}%

```

Do nothing at the start of the table:

```

8766   \renewcommand*{\glossaryheader}{}%

```

No group headings:

```

8767   \renewcommand*{\glsgroupheading}[1]{}%

```

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```

8768   \renewcommand{\glossentry}[2]{%
8769     \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8770     \glossentrydesc{##1}\glspostdescription\space ##2%
8771     \tabularnewline
8772   }%

```

Sub entries put in a row (no name, description and page list in second column):

```

8773   \renewcommand{\subglossentry}[3]{%
8774     &
8775     \glssubentryitem{##2}%
8776     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
8777     ##3%
8778     \tabularnewline
8779   }%

```

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>)

```

8780   \ifglsnogroupskip
8781     \renewcommand*{\glsgroupskip}{}%
8782   \else

```

```

8783 \renewcommand*{\glsgroupskip}{& \tabularnewline}%
8784 \fi
8785 }

```

superraggedborder The superraggedborder style is like the above, but with horizontal and vertical lines:

```

8786 \newglossarystyle{superraggedborder}{%
    Base it on the glostylesuperragged style:
8787 \setglossarystyle{superragged}%
    Put the glossary in a supertabular environment with two columns and a horizontal line in the
    head and tail:
8788 \renewenvironment{theglossary}%
8789 {\tablehead{\hline}\tabletail{\hline}%
8790 \begin{supertabular}{|l|>{\raggedright}p{\glsgdescwidth}|}}%
8791 {\end{supertabular}}%
8792 }

```

superraggedheader The superraggedheader style is like the super style, but with a header:

```

8793 \newglossarystyle{superraggedheader}{%
    Base it on the glostylesuperragged style:
8794 \setglossarystyle{superragged}%
    Put the glossary in a supertabular environment with two columns, a header and no tail:
8795 \renewenvironment{theglossary}%
8796 {\tablehead{\bfseries \entryname & \bfseries \descriptionname
8797 \tabularnewline}%
8798 \tabletail{}}%
8799 \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}}}%
8800 {\end{supertabular}}%
8801 }

```

superraggedheaderborder The superraggedheaderborder style is like the superragged style but with a header and border:

```

8802 \newglossarystyle{superraggedheaderborder}{%
    Base it on the glostylesuper style:
8803 \setglossarystyle{superragged}%
    Put the glossary in a supertabular environment with two columns, a header and horizontal
    lines above and below the table:
8804 \renewenvironment{theglossary}%
8805 {\tablehead{\hline\bfseries \entryname &
8806 \bfseries \descriptionname\tabularnewline\hline}%
8807 \tabletail{\hline}
8808 \begin{supertabular}{|l|>{\raggedright}p{\glsgdescwidth}|}}%
8809 {\end{supertabular}}%
8810 }

```

superragged3col The superragged3col style is like the superragged style, but with 3 columns:

```

8811 \newglossarystyle{superragged3col}{%

```

Put the glossary in a supertabular environment with three columns and no head or tail:

```
8812 \renewenvironment{theglossary}%
8813 {\tablehead{}\tabletail{}}%
8814 \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}%
8815 >{\raggedright}p{\glspagelistwidth}}}%
8816 {\end{supertabular}}%
```

Do nothing at the start of the table:

```
8817 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8818 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
8819 \renewcommand{\glossentry}[2]{%
8820 \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8821 \glossentrydesc{##1} &
8822 ##2\tabularnewline
8823 }%
```

Sub entries on a row (no name, description in second column, page list in last column):

```
8824 \renewcommand{\subglossentry}[3]{%
8825 &
8826 \glssubentryitem{##2}%
8827 \glstarget{##2}{\strut}\glossentrydesc{##2} &
8828 ##3\tabularnewline
8829 }%
```

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>)

```
8830 \ifglsgroupskip
8831 \renewcommand*{\glsgroupskip}{}%
8832 \else
8833 \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
8834 \fi
8835 }
```

ragged3colborder The superragged3colborder style is like the superragged3col style, but with a border:

```
8836 \newglossarystyle{superragged3colborder}{%
```

Base it on the glostylessuperragged3col style:

```
8837 \setglossarystyle{superragged3col}%
```

Put the glossary in a supertabular environment with three columns and a horizontal line in the head and tail:

```
8838 \renewenvironment{theglossary}%
8839 {\tablehead{\hline}\tabletail{\hline}%
8840 \begin{supertabular}{|l|>{\raggedright}p{\glsgdescwidth}|}%
8841 >{\raggedright}p{\glspagelistwidth}|}}%
8842 {\end{supertabular}}%
8843 }
```

ragged3colheader The superragged3colheader style is like the superragged3col style but with a header row:

```
8844 \newglossarystyle{superragged3colheader}{%
```

Base it on the glostylesuperragged3col style:

```
8845 \setglossarystyle{superragged3col}{%
```

Put the glossary in a supertabular environment with three columns, a header and no tail:

```
8846 \renewenvironment{theglossary}{%
8847   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8848     \bfseries\pagelistname\tabularnewline}\tabletail{}}%
8849   \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}%
8850     >{\raggedright}p{\glspagelistwidth}}}%
8851   {\end{supertabular}}}%
8852 }
```

colheaderborder The superragged3colheaderborder style is like the superragged3col style but with a header and border:

```
8853 \newglossarystyle{superragged3colheaderborder}{%
```

Base it on the glostylesuperragged3colborder style:

```
8854 \setglossarystyle{superragged3colborder}{%
```

Put the glossary in a supertabular environment with three columns, a header with horizontal lines and a horizontal line in the tail:

```
8855 \renewenvironment{theglossary}{%
8856   {\tablehead{\hline
8857     \bfseries\entryname&\bfseries\descriptionname&
8858     \bfseries\pagelistname\tabularnewline\hline}%
8859   \tabletail{\hline}%
8860   \begin{supertabular}{|l|>{\raggedright}p{\glsgdescwidth}|%
8861     >{\raggedright}p{\glspagelistwidth}|}%
8862   {\end{supertabular}}}%
8863 }
```

superragged4col The altsuperragged4col glossary style is like altsuper4col style in the package but uses ragged right formatting in the description and page list columns.

```
8864 \newglossarystyle{altsuperragged4col}{%
```

Put the glossary in a supertabular environment with four columns and no head or tail:

```
8865 \renewenvironment{theglossary}{%
8866   {\tablehead{}\tabletail{}}%
8867   \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}l%
8868     >{\raggedright}p{\glspagelistwidth}}}%
8869   {\end{supertabular}}%
```

Do nothing at the start of the table:

```
8870 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8871 \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:

```
8872 \renewcommand{\glossentry}[2]{%
8873   \glentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8874   \glossentrydesc{##1} &
8875   \glossentrysymbol{##1} & ##2\tabularnewline
8876 }%
```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```
8877 \renewcommand{\subglossentry}[3]{%
8878   &
8879   \glssubentryitem{##2}%
8880   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8881   \glossentrysymbol{##2} & ##3\tabularnewline
8882 }%
```

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>)

```
8883 \ifglsgroupskip
8884   \renewcommand*{\glsgroupskip}{}%
8885 \else
8886   \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
8887 \fi
8888 }
```

agged4colheader The altsuperragged4colheader style is like the altsuperragged4col style but with a header row.

```
8889 \newglossarystyle{altsuperragged4colheader}{%
```

Base it on the glostylealtsuperragged4col style:

```
8890 \setglossarystyle{altsuperragged4col}{%
```

Put the glossary in a supertabular environment with four columns, a header and no tail:

```
8891 \renewenvironment{theglossary}%
8892   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8893     \bfseries\symbolname &
8894     \bfseries\pagelistname\tabularnewline}\tabletail{}}%
8895   \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}l%
8896     >{\raggedright}p{\glspagelistwidth}}}%
8897   {\end{supertabular}}%
8898 }
```

agged4colborder The altsuperragged4colborder style is like the altsuperragged4col style but with a border.

```
8899 \newglossarystyle{altsuperragged4colborder}{%
```

Base it on the glostylealtsuperragged4col style:

```
8900 \setglossarystyle{altsuper4col}{%
```

Put the glossary in a supertabular environment with four columns and a horizontal line in the head and tail:

```
8901 \renewenvironment{theglossary}{%
```

```

8902   {\tablehead{\hline}\tabletail{\hline}%
8903   \begin{supertabular}%
8904       {||>{\raggedright}p{\glsgdescwidth}||}%
8905       >{\raggedright}p{\glspagelistwidth}||}%
8906   {\end{supertabular}}}%
8907 }

```

`colheaderborder` The `altsuperragged4colheaderborder` style is like the `altsuperragged4col` style but with a header and border.

```

8908 \newglossarystyle{altsuperragged4colheaderborder}{%

```

Base it on the `glostylealtsuperragged4col` style:

```

8909 \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:

```

8910 \renewenvironment{theglossary}%
8911   {\tablehead{\hline
8912     \bfseries\entryname &
8913     \bfseries\descriptionname &
8914     \bfseries\symbolname &
8915     \bfseries\pagelistname\tabularnewline\hline}%
8916   \tabletail{\hline}%
8917   \begin{supertabular}%
8918       {||>{\raggedright}p{\glsgdescwidth}||}%
8919       >{\raggedright}p{\glspagelistwidth}||}%
8920   {\end{supertabular}}}%
8921 }

```

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.

```

8922 \ProvidesPackage{glossary-tree}[2016/04/30 v4.23 (NLCT)]

```

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```

8923 \providecommand{\indexspace}{%
8924   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
8925 }

```

`\glstreenamefmt` Format used to display the name in the tree styles. (This may be counteracted by `\glsglfont`.) This command was previously also used to format the group headings.

```

8926 \newcommand*{\glstreenamefmt}[1]{\textbf{#1}}

```

`egroupheaderfmt` 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.

```
8927 \newcommand*{\glstreegroupheaderfmt}[1]{\glstreenamefmt{#1}}
```

`eenavigationfmt` Format used to display the navigation header in the tree styles.

```
8928 \newcommand*{\glstreenavigationfmt}[1]{\glstreenamefmt{#1}}
```

`index` The index glossary style is similar in style to the way indices are usually typeset using `\item`, `\subitem` and `\subsubitem`. The entry name is set in bold. If an entry has a symbol, it is placed in brackets after the name. Then the description is displayed, followed by the number list. This style allows up to three levels.

```
8929 \newglossarystyle{index}{%
```

Set the paragraph indentation and skip and define `\item` to be the same as that used by `theindex`:

```
8930 \renewenvironment{theglossary}%
8931 {\setlength{\parindent}{0pt}%
8932 \setlength{\parskip}{0pt plus 0.3pt}%
8933 \let\item\@idxitem}%
8934 {\par}%
8935 \renewcommand*{\glossaryheader}{}%
8936 \renewcommand*{\glsgroupheading}[1]{}%
8937 \renewcommand*{\glosseentry}[2]{%
8938 \item\glseentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glosseentryname{##1}}}%
8939 \ifglshassymbol{##1}{\space(\glosseentrysymbol{##1})}{}%
8940 \space \glosseentrydesc{##1}\glspostdescription\space ##2%
8941 }%
```

Do nothing at the start of the environment:

```
8935 \renewcommand*{\glossaryheader}{}%
```

No group headers:

```
8936 \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.

```
8937 \renewcommand*{\glosseentry}[2]{%
8938 \item\glseentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glosseentryname{##1}}}%
8939 \ifglshassymbol{##1}{\space(\glosseentrysymbol{##1})}{}%
8940 \space \glosseentrydesc{##1}\glspostdescription\space ##2%
8941 }%
```

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 `\glosseentry`, 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.

```
8942 \renewcommand{\subglosseentry}[3]{%
8943 \ifcase##1\relax
8944 % level 0
8945 \item
8946 \or
8947 % level 1
8948 \subitem
```

```

8949     \glssubentryitem{##2}%
8950   \else
8951     % all other levels
8952     \subsubitem
8953   \fi
8954   \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
8955   \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
8956   \space\glossentrydesc{##2}\glspostdescription\space ##3%
8957 }%

```

Vertical gap between groups is the same as that used by indices:

```

8958 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}

```

indexgroup The indexgroup style is like the index style but has headings.

```

8959 \newglossarystyle{indexgroup}{%

```

Base it on the glostyleindex style:

```

8960 \setglossarystyle{index}%

```

Add a heading for each group. This puts the group's title in bold followed by a vertical gap.

```

8961 \renewcommand*{\glsgroupheading}[1]{%
8962   \item\glstreegroupheaderfmt{\glsggetgrouptitle{##1}}%
8963   \indexspace
8964 }%
8965 }

```

indexhypergroup The indexhypergroup style is like the indexgroup style but has hyper navigation.

```

8966 \newglossarystyle{indexhypergroup}{%

```

Base it on the glostyleindex style:

```

8967 \setglossarystyle{index}%

```

Put navigation links to the groups at the start of the glossary:

```

8968 \renewcommand*{\glossaryheader}{%
8969   \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.

```

8970 \renewcommand*{\glsgroupheading}[1]{%
8971   \item\glstreegroupheaderfmt
8972     {\glsnavhypertarget{##1}{\glsggetgrouptitle{##1}}}%
8973   \indexspace}%
8974 }

```

tree The tree glossary style is similar in style to the index style, but can have arbitrary levels.

```

8975 \newglossarystyle{tree}{%

```

Set the paragraph indentation and skip:

```

8976 \renewenvironment{theglossary}%
8977   {\setlength{\parindent}{0pt}%
8978   \setlength{\parskip}{0pt plus 0.3pt}}%
8979   {}%

```

Do nothing at the start of the theglossary environment:

```
8980 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8981 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries: name in bold, followed by symbol in brackets (if it exists), the description and the page list:

```
8982 \renewcommand{\glossentry}[2]{%
8983   \hangindent0pt\relax
8984   \parindent0pt\relax
8985   \glstentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
8986   \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
8987   \space\glossentrydesc{##1}\glspostdescription\space##2\par
8988 }%
```

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.

```
8989 \renewcommand{\subglossentry}[3]{%
8990   \hangindent##1\glstreeindent\relax
8991   \parindent##1\glstreeindent\relax
8992   \ifnum##1=1\relax
8993     \glssubentryitem{##2}%
8994   \fi
8995   \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
8996   \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
8997   \space\glossentrydesc{##2}\glspostdescription\space ##3\par
8998 }%
```

Vertical gap between groups is the same as that used by indices:

```
8999 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}}
```

treegroup Like the tree style but the glossary groups have headings.

```
9000 \newglossarystyle{treegroup}{%
```

Base it on the `glostyletree` style:

```
9001 \setglossarystyle{tree}%
```

Each group has a heading (in bold) followed by a vertical gap):

```
9002 \renewcommand{\glsgroupheading}[1]{\par
9003   \noindent\glstreegroupheaderfmt{\glsgrouptitle{##1}}\par
9004   \indexspace}%
9005 }
```

treehypergroup The `treehypergroup` style is like the `treegroup` style, but has a set of links to the groups at the start of the glossary.

```
9006 \newglossarystyle{treehypergroup}{%
```

Base it on the `glostyletree` style:

```
9007 \setglossarystyle{tree}%
```

Put navigation links to the groups at the start of the theglossary environment:

```
9008 \renewcommand*{\glossaryheader}{%
9009 \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
9010 \renewcommand*{\glsgroupheading}[1]{%
9011 \par\noindent
9012 \glstreegroupheaderfmt
9013 {\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}\par
9014 \indexspace}%
9015 }
```

`\glstreeindent` Length governing left indent for each level of the tree style.

```
9016 \newlength\glstreeindent
9017 \setlength{\glstreeindent}{10pt}
```

`treenoname` The `treenoname` glossary style is like the tree style, but doesn't print the name or symbol for sub-levels.

```
9018 \newglossarystyle{treenoname}{%
```

Set the paragraph indentation and skip:

```
9019 \renewenvironment{theglossary}%
9020 {\setlength{\parindent}{0pt}%
9021 \setlength{\parskip}{0pt plus 0.3pt}}%
9022 {}%
```

No header:

```
9023 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9024 \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.

```
9025 \renewcommand{\glossentry}[2]{%
9026 \hangindent0pt\relax
9027 \parindent0pt\relax
9028 \glstryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
9029 \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
9030 \space\glossentrydesc{##1}\glspostdescription\space##2\par
9031 }%
```

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.

```
9032 \renewcommand{\subglossentry}[3]{%
9033 \hangindent##1\glstreeindent\relax
9034 \parindent##1\glstreeindent\relax
9035 \ifnum##1=1\relax
9036 \glssubentryitem{##2}%
9037 \fi
9038 \glstarget{##2}{\strut}%
```

```

9039 \glossentrydesc{##2}\glspostdescription\space##3\par
9040 }%

```

Vertical gap between groups is the same as that used by indices:

```

9041 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
9042 }

```

treenonamegroup Like the `treenoname` style but the glossary groups have headings.

```

9043 \newglossarystyle{treenonamegroup}{%

```

Base it on the `glostyletreenoname` style:

```

9044 \setglossarystyle{treenoname}%

```

Give each group a heading:

```

9045 \renewcommand{\glsgroupheading}[1]{\par
9046 \noindent\glstreegroupheaderfmt
9047 {\glsgrouptitle{##1}}\par\indexspace}%
9048 }

```

onamehypergroup The `treenonamehypergroup` style is like the `treenonamegroup` style, but has a set of links to the groups at the start of the glossary.

```

9049 \newglossarystyle{treenonamehypergroup}{%

```

Base it on the `glostyletreenoname` style:

```

9050 \setglossarystyle{treenoname}%

```

Put navigation links to the groups at the start of the `theglossary` environment:

```

9051 \renewcommand*{\glossaryheader}{%
9052 \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%

```

Each group has a heading (in bold with a target) followed by a vertical gap):

```

9053 \renewcommand*{\glsgroupheading}[1]{%
9054 \par\noindent
9055 \glstreegroupheaderfmt
9056 {\glsnavigationtarget{##1}}{\glsgrouptitle{##1}}\par
9057 \indexspace}%
9058 }

```

esttoplevelname Find the widest name over all parentless entries in the given glossary or glossaries.

```

9059 \newrobustcmd*{\glsfindwidesttoplevelname}[1][\@glo@types]{%
9060 \dimen@=0pt\relax
9061 \glstmplen=0pt\relax
9062 \forallglossaries[#1]{\@gls@type}%
9063 {%
9064 \forallglsentries[\@gls@type]{\@glo@label}%
9065 {%
9066 \ifglshasparent{\@glo@label}%
9067 }%
9068 {%
9069 \settowidth{\dimen@}%
9070 {\glstreenamfmt{\glsentryname{\@glo@label}}}%

```

```

9071      \ifdim\dimen@>\gls@tmplen
9072      \gls@tmplen=\dimen@
9073      \letcs{\@glswidestname}{glo@\glsdetoklabel{\@glo@label}@name}%
9074      \fi
9075    }%
9076  }%
9077 }%
9078 }

```

`\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.

```

9079 \newcommand*{\glssetwidest}[2][0]{%
9080   \expandafter\def\csname @glswidestname\romannumeral#1\endcsname{%
9081     #2}%
9082 }

```

`\@glswidestname` Initialise `\@glswidestname`.

```

9083 \newcommand*{\@glswidestname}{}

```

`\glstreenamebox` Used by the alttree style to create the box for the name and associated information.

```

9084 \newcommand*{\glstreenamebox}[2]{%
9085   \makebox[#1][l]{#2}%
9086 }

```

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`.

```

9087 \newglossarystyle{alttree}{%

```

Redefine theglossary environment.

```

9088   \renewenvironment{theglossary}%
9089     {\def\@gls@prevlevel{-1}%
9090      \mbox{}\par}%
9091     {\par}%

```

Set the header and group headers to nothing.

```

9092   \renewcommand*{\glossaryheader}{}%
9093   \renewcommand*{\glsgroupheading}[1]{}%

```

Redefine the way that the level 0 entries are displayed.

```

9094   \renewcommand{\glosseentry}[2]{%
9095     \ifnum\@gls@prevlevel=0\relax
9096     \else

```

Find out how big the indentation should be by measuring the widest entry.

```

9097       \settowidth{\glstreeindent}{\glstreenamefmt{\@glswidestname\space}}%
9098     \fi

```

Set the hangindent and paragraph indent.

```

9099     \hangindent\glstreeindent
9100     \parindent\glstreeindent

```


Put the name to the left of the paragraph block.

```
9101 \makebox[Opt][r]{\glstreenamebox{\glstreeindent}{%
9102 \glstryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}}}%
```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
9103 \ifglshassymbol{##1}{(\glossentrysymbol{##1})\space}{}%
```

Do the description followed by the description terminator and location list.

```
9104 \glossentrydesc{##1}\glspostdescription \space ##2\par
```

Set the previous level to 0.

```
9105 \def\@gls@prevlevel{0}%
9106 }%
```

Redefine the way sub-entries are displayed.

```
9107 \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.

```
9108 \ifnum##1=1\relax
9109 \glssubentryitem{##2}%
9110 \fi
```

If the level hasn't changed, keep the same settings, otherwise adjust \glstreeindent accordingly.

```
9111 \ifnum\@gls@prevlevel=##1\relax
9112 \else
```

Compute the widest entry for this level, or for level 0 if not defined for this level. Store in \gls@tmplen

```
9113 \@ifundefined{@glswidestname\romannumeral##1}{%
9114 \settowidth{\gls@tmplen}{\glstreenamefmt{\@glswidestname\space}}}%
9115 \settowidth{\gls@tmplen}{\glstreenamefmt{%
9116 \csname @glswidestname\romannumeral##1\endcsname\space}}}%
```

Determine if going up or down a level

```
9117 \ifnum\@gls@prevlevel<##1\relax
```

Depth has increased, so add the width of the widest entry to \glstreeindent.

```
9118 \setlength\glstreeindent\gls@tmplen
9119 \addtolength\glstreeindent\parindent
9120 \parindent\glstreeindent
9121 \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.

```
9122 \@ifundefined{@glswidestname\romannumeral\@gls@prevlevel}{%
9123 \settowidth{\glstreeindent}{\glstreenamefmt{%
9124 \@glswidestname\space}}}%
9125 \settowidth{\glstreeindent}{\glstreenamefmt{%
9126 \csname @glswidestname\romannumeral\@gls@prevlevel
9127 \endcsname\space}}}%
```

Subtract this length from the previous level's paragraph indent and set to `\glstreeindent`.

```
9128      \addtolength\parindent{-\glstreeindent}%
9129      \setlength\glstreeindent\parindent
9130      \fi
9131  \fi
```

Set the hanging indentation.

```
9132  \hangindent\glstreeindent
```

Put the name to the left of the paragraph block

```
9133  \makebox[0pt][r]{\glstreenamebox{\gls@tmplen}{%
9134    \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}}}%
```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
9135  \ifglshassymbol{##2}{(\glossentrysymbol{##2})\space}{}%
```

Do the description followed by the description terminator and location list.

```
9136  \glossentrydesc{##2}\glspostdescription\space ##3\par
```

Set the previous level macro to the current level.

```
9137  \def\@gls@prevlevel{##1}%
9138  }%
```

Vertical gap between groups is the same as that used by indices:

```
9139  \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
9140 }
```

almtreegroup Like the `almtree` style but the glossary groups have headings.

```
9141 \newglossarystyle{almtreegroup}{%
```

Base it on the `glostylealmtree` style:

```
9142  \setglossarystyle{almtree}%
```

Give each group a heading.

```
9143  \renewcommand{\glsgroupheading}[1]{\par
9144    \def\@gls@prevlevel{-1}%
9145    \hangindent0pt\relax
9146    \parindent0pt\relax
9147    \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}}%
9148  \par\indexspace}%
9149 }
```

almtreehypergroup The `almtreehypergroup` style is like the `almtreegroup` style, but has a set of links to the groups at the start of the glossary.

```
9150 \newglossarystyle{almtreehypergroup}{%
```

Base it on the `glostylealmtree` style:

```
9151  \setglossarystyle{almtree}%
```

Put the navigation links in the header

```
9152  \renewcommand*{\glossaryheader}{%
9153  \par
```

```

9154     \def\@gls@prevlevel{-1}%
9155     \hangindent0pt\relax
9156     \parindent0pt\relax
9157     \glstreenavigationfmt{\glsnavigation}\par\indexspace}%

```

Put a hypertarget at the start of each group

```

9158     \renewcommand*{\glsgroupheading}[1]{%
9159     \par
9160     \def\@gls@prevlevel{-1}%
9161     \hangindent0pt\relax
9162     \parindent0pt\relax
9163     \glstreegroupheaderfmt
9164     {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
9165     \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.

```
9166 \NeedsTeXFormat{LaTeX2e}
9167 \ProvidesPackage{glossaries-compatible-207}[2016/04/30 v4.23 (NLCT)]
```

AddXdyAttribute Adds an attribute in old format.

```
9168 \ifglsxindy
9169   \renewcommand*\GlsAddXdyAttribute[1]{%
9170     \edef\@xdyattributes{\@xdyattributes ^^J \string"#1\string"}%
9171     \expandafter\toks@\expandafter{\@xdylocref}%
9172     \edef\@xdylocref{\the\toks@ ^^J}%
9173     (markup-locref
9174     :open \string"\string~n\string\setentrycounter
9175       {\noexpand\glscounter}%
9176       \expandafter\string\csname#1\endcsname
9177       \expandafter\@gobble\string\{\string" ^^J
9178       :close \string"\expandafter\@gobble\string\}\string" ^^J
9179       :attr \string"#1\string"))}}
```

Only has an effect before `\writeist`:

```
9180 \fi
```

sAddXdyCounters

```
9181 \renewcommand*\GlsAddXdyCounters[1]{%
9182   \GlossariesWarning{\string\GlsAddXdyCounters\space not available
9183     in compatibility mode.}%
9184 }
```

Add predefined attributes

```
9185 \GlsAddXdyAttribute{glsnumberformat}
9186 \GlsAddXdyAttribute{textrm}
9187 \GlsAddXdyAttribute{textsf}
9188 \GlsAddXdyAttribute{texttt}
9189 \GlsAddXdyAttribute{textbf}
9190 \GlsAddXdyAttribute{textmd}
9191 \GlsAddXdyAttribute{textit}
9192 \GlsAddXdyAttribute{textup}
9193 \GlsAddXdyAttribute{textsl}
```

```

9194 \GlsAddXdyAttribute{textsc}
9195 \GlsAddXdyAttribute{emph}
9196 \GlsAddXdyAttribute{glshypernumber}
9197 \GlsAddXdyAttribute{hyperrm}
9198 \GlsAddXdyAttribute{hypersf}
9199 \GlsAddXdyAttribute{hypertt}
9200 \GlsAddXdyAttribute{hyperbf}
9201 \GlsAddXdyAttribute{hypermd}
9202 \GlsAddXdyAttribute{hyperit}
9203 \GlsAddXdyAttribute{hyperup}
9204 \GlsAddXdyAttribute{hypersl}
9205 \GlsAddXdyAttribute{hypersc}
9206 \GlsAddXdyAttribute{hyperemph}

```

sAddXdyLocation Restore v2.07 definition:

```

9207 \ifglxindy
9208   \renewcommand*{\GlsAddXdyLocation}[2]{%
9209     \edef\@xdyuserlocationdefs{%
9210       \@xdyuserlocationdefs ^^J%
9211       (define-location-class \string"#1\string"^^J\space\space
9212       \space(#2))
9213     }%
9214     \edef\@xdyuserlocationnames{%
9215       \@xdyuserlocationnames^^J\space\space\space
9216       \string"#1\string"}%
9217   }
9218 \fi

```

\@do@wrglossary

```

9219 \renewcommand{\@do@wrglossary}[1]{%
  Determine whether to use xindy or makeindex syntax
9220 \ifglxindy
  Need to determine if the formatting information starts with a ( or ) indicating a range.
9221 \expandafter\@glo@check@mkidxrangechar\@glsnumberformat\@nil
9222 \def\@glo@range{}%
9223 \expandafter\if\@glo@prefix(\relax
9224   \def\@glo@range{:open-range}%
9225 \else
9226   \expandafter\if\@glo@prefix)\relax
9227   \def\@glo@range{:close-range}%
9228 \fi
9229 \fi

  Get the location and escape any special characters
9230 \protected@edef\@glslocref{\theglentrycounter}%
9231 \@gls@checkmkidxchars\@glslocref

  Write to the glossary file using xindy syntax.
9232 \glossary[\csname glo@#1@type\endcsname]{%

```

```

9233 (indexentry :tkey (\csname glo@#1@index\endcsname)
9234 :locoref \string"\@glslocoref\string" %
9235 :attr \string"\@glo@suffix\string" \@glo@range
9236 )
9237 }%
9238 \else

```

Convert the format information into the format required for makeindex

```

9239 \@set@glo@numformat\@glo@numfmt\@gls@counter\@glsnumberformat

```

Write to the glossary file using makeindex syntax.

```

9240 \glossary[\csname glo@#1@type\endcsname]{%
9241 \string\glossaryentry{\csname glo@#1@index\endcsname
9242 \@gls@encapchar\@glo@numfmt}{\theglsentrycounter}}%
9243 \fi
9244 }

```

t@glo@numformat Only had 3 arguments in v2.07

```

9245 \def\@set@glo@numformat#1#2#3{%
9246 \expandafter\@glo@check@mkidxrangechar#3\@nil
9247 \protected@edef#1{%
9248 \@glo@prefix setentrycounter[]{\#2}%
9249 \expandafter\string\csname\@glo@suffix\endcsname
9250 }%
9251 \@gls@checkmkidxchars#1%
9252 }

```

\writeist Redefine \writeist back to the way it was in v2.07, but change \istfile to \glswrite.

```

9253 \ifglxindy
9254 \def\writeist{%
9255 \openout\glswrite=\istfilename
9256 \write\glswrite{;; xindy style file created by the glossaries
9257 package in compatible-2.07 mode}%
9258 \write\glswrite{;; for document '\jobname' on
9259 \the\year-\the\month-\the\day}%
9260 \write\glswrite{^^J; required styles^^J}
9261 \@for\@xdystyle:=\@xdyrequiredstyles\do{%
9262 \ifx\@xdystyle\@empty
9263 \else
9264 \protected@write\glswrite{{(require
9265 \string"\@xdystyle.xdy\string")}}%
9266 \fi
9267 }%
9268 \write\glswrite{^^J%
9269 ; list of allowed attributes (number formats)^^J}%
9270 \write\glswrite{(define-attributes ((\@xdyattributes)))}%
9271 \write\glswrite{^^J; user defined alphabets^^J}%
9272 \write\glswrite{\@xdyuseralphabets}%
9273 \write\glswrite{^^J; location class definitions^^J}%
9274 \protected@edef\@gls@roman{\@roman{0}\string"

```

```

9275     \string"roman-numbers-lowercase\string" :sep \string"}}%
9276 \@onelevel@sanitize\@gls@roman
9277 \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
9278     :sep \string"}}%
9279 \@onelevel@sanitize\@tmp
9280 \ifx\@tmp\@gls@roman
9281     \write\glswrite{(define-location-class
9282         \string"roman-page-numbers\string"^^J\space\space\space
9283         (\string"roman-numbers-lowercase\string")
9284         :min-range-length \@glsminrange)}}%
9285 \else
9286     \write\glswrite{(define-location-class
9287         \string"roman-page-numbers\string"^^J\space\space\space
9288         (:sep "\@gls@roman")
9289         :min-range-length \@glsminrange)}}%
9290 \fi
9291 \write\glswrite{(define-location-class
9292     \string"Roman-page-numbers\string"^^J\space\space\space
9293     (\string"roman-numbers-uppercase\string")
9294     :min-range-length \@glsminrange)}}%
9295 \write\glswrite{(define-location-class
9296     \string"arabic-page-numbers\string"^^J\space\space\space
9297     (\string"arabic-numbers\string")
9298     :min-range-length \@glsminrange)}}%
9299 \write\glswrite{(define-location-class
9300     \string"alpha-page-numbers\string"^^J\space\space\space
9301     (\string"alpha\string")
9302     :min-range-length \@glsminrange)}}%
9303 \write\glswrite{(define-location-class
9304     \string"Alpha-page-numbers\string"^^J\space\space\space
9305     (\string"ALPHA\string")
9306     :min-range-length \@glsminrange)}}%
9307 \write\glswrite{(define-location-class
9308     \string"Appendix-page-numbers\string"^^J\space\space\space
9309     (\string"ALPHA\string"
9310     :sep \string"\@glsAlphacompositor\string"
9311     \string"arabic-numbers\string")
9312     :min-range-length \@glsminrange)}}%
9313 \write\glswrite{(define-location-class
9314     \string"arabic-section-numbers\string"^^J\space\space\space
9315     (\string"arabic-numbers\string"
9316     :sep \string"\glscompositor\string"
9317     \string"arabic-numbers\string")
9318     :min-range-length \@glsminrange)}}%
9319 \write\glswrite{^^J; user defined location classes}%
9320 \write\glswrite{\@xdyuserlocationdefs}%
9321 \write\glswrite{^^J; define cross-reference class^^J}%
9322 \write\glswrite{(define-crossref-class \string"see\string"
9323     :unverified )}%

```

```

9324 \write\glswrite{(markup-crossref-list
9325 :class \string"see\string"^^J\space\space\space
9326 :open \string"\string\glseeformat\string"
9327 :close \string"{}\string")}%
9328 \write\glswrite{^^J; define the order of the location classes}%
9329 \write\glswrite{(define-location-class-order
9330 (\@xdylocationclassorder))}%
9331 \write\glswrite{^^J; define the glossary markup^^J}%
9332 \write\glswrite{(markup-index^^J\space\space\space
9333 :open \string"\string
9334 \glossarysection[\string\glossarytoctitle]{\string
9335 \glossarytitle}\string\glossarypreamble\string~n\string\begin
9336 {theglossary}\string\glossaryheader\string~n\string" ^^J\space
9337 \space\space:close \string"\expandafter\@gobble
9338 \string%\string~n\string
9339 \end{theglossary}\string\glossarypostamble
9340 \string~n\string" ^^J\space\space\space
9341 :tree)}}%
9342 \write\glswrite{(markup-letter-group-list
9343 :sep \string"\string\glsgroupskip\string~n\string")}%
9344 \write\glswrite{(markup-indexentry
9345 :open \string"\string\relax \string\glresetentrylist
9346 \string~n\string")}%
9347 \write\glswrite{(markup-locclass-list :open
9348 \string"\glsoopenbrace\string\glossaryentrynumbers
9349 \glsoopenbrace\string\relax\space \string"^^J\space\space\space
9350 :sep \string", \string"
9351 :close \string"\glsclosebrace\glsclosebrace\string")}%
9352 \write\glswrite{(markup-locref-list
9353 :sep \string"\string\delimN\space\string")}%
9354 \write\glswrite{(markup-range
9355 :sep \string"\string\delimR\space\string")}%
9356 \@onelevel@sanitize\gls@suffixF
9357 \@onelevel@sanitize\gls@suffixFF
9358 \ifx\gls@suffixF\@empty
9359 \else
9360 \write\glswrite{(markup-range
9361 :close "\gls@suffixF" :length 1 :ignore-end)}%
9362 \fi
9363 \ifx\gls@suffixFF\@empty
9364 \else
9365 \write\glswrite{(markup-range
9366 :close "\gls@suffixFF" :length 2 :ignore-end)}%
9367 \fi
9368 \write\glswrite{^^J; define format to use for locations^^J}%
9369 \write\glswrite{\@xdylocref}%
9370 \write\glswrite{^^J; define letter group list format^^J}%
9371 \write\glswrite{(markup-letter-group-list
9372 :sep \string"\string\glsgroupskip\string~n\string")}%

```



```

9373 \write\glswrite{^^J; letter group headings^^J}%
9374 \write\glswrite{(markup-letter-group
9375   :open-head \string"\string\glsgroupheading
9376   \glsoopenbrace\string"^^J\space\space\space
9377   :close-head \string"\glsclosebrace\string")}%
9378 \write\glswrite{^^J; additional letter groups^^J}%
9379 \write\glswrite{\@xdylettergroups}%
9380 \write\glswrite{^^J; additional sort rules^^J}
9381 \write\glswrite{\@xdysortrules}%
9382 \noist}
9383 \else
9384 \edef\@gls@actualchar{\string?}
9385 \edef\@gls@encapchar{\string|}
9386 \edef\@gls@levelchar{\string!}
9387 \edef\@gls@quotechar{\string"}
9388 \def\writeist{\relax
9389   \openout\glswrite=\istfilename
9390   \write\glswrite{\expandafter\@gobble\string\% makeindex style file
9391     created by the glossaries package}
9392   \write\glswrite{\expandafter\@gobble\string\% for document
9393     '\jobname' on \the\year-\the\month-\the\day}
9394   \write\glswrite{actual '\@gls@actualchar'}
9395   \write\glswrite{encap '\@gls@encapchar'}
9396   \write\glswrite{level '\@gls@levelchar'}
9397   \write\glswrite{quote '\@gls@quotechar'}
9398   \write\glswrite{keyword \string"\string\glossaryentry\string"}
9399   \write\glswrite{preamble \string"\string\glossarysection[\string
9400     \glossarytoctitle]{\string\glossarytitle}\string
9401     \glossarypreamble\string\n\string\begin{theglossary}\string
9402     \glossaryheader\string\n\string"}
9403   \write\glswrite{postamble \string"\string%\string\n\string
9404     \end{theglossary}\string\glossarypostamble\string\n
9405     \string"}
9406   \write\glswrite{group_skip \string"\string\glsgroupskip\string\n
9407     \string"}
9408   \write\glswrite{item_0 \string"\string%\string\n\string"}
9409   \write\glswrite{item_1 \string"\string%\string\n\string"}
9410   \write\glswrite{item_2 \string"\string%\string\n\string"}
9411   \write\glswrite{item_01 \string"\string%\string\n\string"}
9412   \write\glswrite{item_x1
9413     \string"\string\relax \string\glsresetentrylist\string\n
9414     \string"}
9415   \write\glswrite{item_12 \string"\string%\string\n\string"}
9416   \write\glswrite{item_x2
9417     \string"\string\relax \string\glsresetentrylist\string\n
9418     \string"}
9419   \write\glswrite{delim_0 \string"\string\{\string
9420     \glossaryentrynumbers\string\{\string\relax \string"}
9421   \write\glswrite{delim_1 \string"\string\{\string

```

```

9422     \glossaryentrynumbers\string\{\string\relax \string}
9423     \write\glswrite{delim_2 \string"\string\{\string
9424     \glossaryentrynumbers\string\{\string\relax \string}
9425     \write\glswrite{delim_t \string"\string\}\string\}\string}
9426     \write\glswrite{delim_n \string"\string\delimN \string}
9427     \write\glswrite{delim_r \string"\string\delimR \string}
9428     \write\glswrite{headings_flag 1}
9429     \write\glswrite{heading_prefix
9430         \string"\string\glsgroupheading\string\{\string}
9431     \write\glswrite{heading_suffix
9432         \string"\string\}\string\relax
9433         \string\glsgroupresetentrylist \string}
9434     \write\glswrite{symhead_positive \string"glssymbols\string}
9435     \write\glswrite{numhead_positive \string"glslnumbers\string}
9436     \write\glswrite{page_compositor \string"glscpositor\string}
9437     \@gls@escbsdq\gls@suffixF
9438     \@gls@escbsdq\gls@suffixFF
9439     \ifx\gls@suffixF\@empty
9440     \else
9441         \write\glswrite{suffix_2p \string"\gls@suffixF\string}
9442     \fi
9443     \ifx\gls@suffixFF\@empty
9444     \else
9445         \write\glswrite{suffix_3p \string"\gls@suffixFF\string}
9446     \fi
9447     \noist
9448 }
9449 \fi
\noist
9450 \renewcommand*{\noist}{\let\writeist\relax}

```

4.2 glossaries-compatible-307

```

9451 \NeedsTeXFormat{LaTeX2e}
9452 \ProvidesPackage{glossaries-compatible-307}[2016/04/30 v4.23 (NLCT)]

```

Compatibility macros for predefined glossary styles:

`\atglossarystyle` Defines a compatibility glossary style.

```

9453 \newcommand{\compatglossarystyle}[2]{%
9454     \ifcsundef{@glscompstyle@#1}%
9455     {%
9456         \csdef{@glscompstyle@#1}{#2}%
9457     }%
9458     {%
9459         \PackageError{glossaries}{Glossary compatibility style ‘#1’ is already defined}{}%
9460     }%
9461 }

```

Backward compatible inline style.

```
9462 \compatglossarystyle{inline}{%
9463   \renewcommand{\glossaryentryfield}[5]{%
9464     \glsinlinedopostchild
9465     \gls@inlinesep
9466     \def\glo@desc{##3}%
9467     \def\@no@post@desc{\nopostdesc}%
9468     \glstentryitem{##1}\glsinlinenameformat{##1}{##2}%
9469     \ifx\glo@desc\@no@post@desc
9470       \glsinlineemptydescformat{##4}{##5}%
9471     \else
9472       \ifstrepty{##3}%
9473         {\glsinlineemptydescformat{##4}{##5}}%
9474         {\glsinlinedescformat{##3}{##4}{##5}}%
9475     \fi
9476     \ifglshaschildren{##1}%
9477     {%
9478       \glsresetsubentrycounter
9479       \glsinlineparentchildseparator
9480       \def\gls@inlinesubsep{}%
9481       \def\gls@inlinepostchild{\glsinlinepostchild}%
9482     }%
9483   }%
9484   \def\gls@inlinesep{\glsinlineseparator}%
9485 }%
```

Sub-entries display description:

```
9486 \renewcommand{\glossarysubentryfield}[6]{%
9487   \gls@inlinesubsep%
9488   \glsinlinesubnameformat{##2}{##3}%
9489   \glssubentryitem{##2}\glsinlinesubdescformat{##4}{##5}{##6}%
9490   \def\gls@inlinesubsep{\glsinlinesubseparator}%
9491 }%
9492 }
```

Backward compatible list style.

```
9493 \compatglossarystyle{list}{%
9494   \renewcommand*{\glossaryentryfield}[5]{%
9495     \item[\glstentryitem{##1}\glstarget{##1}{##2}]
9496     ##3\glspostdescription\space ##5}%
9497 }
```

Sub-entries continue on the same line:

```
9497 \renewcommand*{\glossarysubentryfield}[6]{%
9498   \glssubentryitem{##2}%
9499   \glstarget{##2}{\strut}##4\glspostdescription\space ##6.}%
9500 }
```

Backward compatible listgroup style.

```
9501 \compatglossarystyle{listgroup}{%
9502   \csuse{@glscompstyle@list}%
9503 }%
```

Backward compatible listhypergroup style.

```
9504 \compatglossarystyle{listhypergroup}{%
9505   \csuse{@glscompstyle@list}%
9506 }%
```

Backward compatible altlist style.

```
9507 \compatglossarystyle{altlist}{%
9508   \renewcommand*{\glossaryentryfield}[5]{%
9509     \item[\glstentryitem{##1}\glstarget{##1}{##2}]%
9510     \mbox{}\par\nobreak\@afterheading
9511     ##3\glspostdescription\space ##5}%
9512   \renewcommand{\glossarysubentryfield}[6]{%
9513     \par
9514     \glssubentryitem{##2}%
9515     \glstarget{##2}{\strut}##4\glspostdescription\space ##6}%
9516 }%
```

Backward compatible altlistgroup style.

```
9517 \compatglossarystyle{altlistgroup}{%
9518   \csuse{@glscompstyle@altlist}%
9519 }%
```

Backward compatible altlisthypergroup style.

```
9520 \compatglossarystyle{altlisthypergroup}{%
9521   \csuse{@glscompstyle@altlist}%
9522 }%
```

Backward compatible listdotted style.

```
9523 \compatglossarystyle{listdotted}{%
9524   \renewcommand*{\glossaryentryfield}[5]{%
9525     \item[]\makebox[\glslistdottedwidth][l]{%
9526       \glstentryitem{##1}\glstarget{##1}{##2}%
9527       \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##3}%
9528   \renewcommand*{\glossarysubentryfield}[6]{%
9529     \item[]\makebox[\glslistdottedwidth][l]{%
9530       \glssubentryitem{##2}%
9531       \glstarget{##2}{##3}%
9532       \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##4}%
9533 }%
```

Backward compatible sublistdotted style.

```
9534 \compatglossarystyle{sublistdotted}{%
9535   \csuse{@glscompstyle@listdotted}%
9536   \renewcommand*{\glossaryentryfield}[5]{%
9537     \item[\glstentryitem{##1}\glstarget{##1}{##2}]}%
9538 }%
```

Backward compatible long style.

```
9539 \compatglossarystyle{long}{%
9540   \renewcommand*{\glossaryentryfield}[5]{%
9541     \glstentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
9542   \renewcommand*{\glossarysubentryfield}[6]{%

```

```

9543      &
9544      \glssubentryitem{##2}%
9545      \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
9546 }%

```

Backward compatible longborder style.

```

9547 \compatglossarystyle{longborder}{%
9548   \csuse{@glscmpstyle@long}%
9549 }%

```

Backward compatible longheader style.

```

9550 \compatglossarystyle{longheader}{%
9551   \csuse{@glscmpstyle@long}%
9552 }%

```

Backward compatible longheaderborder style.

```

9553 \compatglossarystyle{longheaderborder}{%
9554   \csuse{@glscmpstyle@long}%
9555 }%

```

Backward compatible long3col style.

```

9556 \compatglossarystyle{long3col}{%
9557   \renewcommand*{\glossaryentryfield}[5]{%
9558     \glstarget{##1}\glstarget{##1}{##2} & ##3 & ##5\\}%
9559   \renewcommand*{\glossarysubentryfield}[6]{%
9560     &
9561     \glssubentryitem{##2}%
9562     \glstarget{##2}{\strut}##4 & ##6\\}%
9563 }%

```

Backward compatible long3colborder style.

```

9564 \compatglossarystyle{long3colborder}{%
9565   \csuse{@glscmpstyle@long3col}%
9566 }%

```

Backward compatible long3colheader style.

```

9567 \compatglossarystyle{long3colheader}{%
9568   \csuse{@glscmpstyle@long3col}%
9569 }%

```

Backward compatible long3colheaderborder style.

```

9570 \compatglossarystyle{long3colheaderborder}{%
9571   \csuse{@glscmpstyle@long3col}%
9572 }%

```

Backward compatible long4col style.

```

9573 \compatglossarystyle{long4col}{%
9574   \renewcommand*{\glossaryentryfield}[5]{%
9575     \glstarget{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%
9576   \renewcommand*{\glossarysubentryfield}[6]{%
9577     &
9578     \glssubentryitem{##2}%

```

9579 \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%
 9580 }%

Backward compatible long4colheader style.

9581 \compatglossarystyle{long4colheader}{%
 9582 \csuse{@glscompstyle@long4col}%
 9583 }%

Backward compatible long4colborder style.

9584 \compatglossarystyle{long4colborder}{%
 9585 \csuse{@glscompstyle@long4col}%
 9586 }%

Backward compatible long4colheaderborder style.

9587 \compatglossarystyle{long4colheaderborder}{%
 9588 \csuse{@glscompstyle@long4col}%
 9589 }%

Backward compatible altlong4col style.

9590 \compatglossarystyle{altlong4col}{%
 9591 \csuse{@glscompstyle@long4col}%
 9592 }%

Backward compatible altlong4colheader style.

9593 \compatglossarystyle{altlong4colheader}{%
 9594 \csuse{@glscompstyle@long4col}%
 9595 }%

Backward compatible altlong4colborder style.

9596 \compatglossarystyle{altlong4colborder}{%
 9597 \csuse{@glscompstyle@long4col}%
 9598 }%

Backward compatible altlong4colheaderborder style.

9599 \compatglossarystyle{altlong4colheaderborder}{%
 9600 \csuse{@glscompstyle@long4col}%
 9601 }%

Backward compatible long style.

9602 \compatglossarystyle{longragged}{%
 9603 \renewcommand*{\glossaryentryfield}[5]{%
 9604 \glssentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
 9605 \tabularnewline}%
 9606 \renewcommand*{\glossarysubentryfield}[6]{%
 9607 &
 9608 \glssubentryitem{##2}%
 9609 \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
 9610 \tabularnewline}%
 9611 }%

Backward compatible longraggedborder style.

9612 \compatglossarystyle{longraggedborder}{%
 9613 \csuse{@glscompstyle@longragged}%
 9614 }%

Backward compatible longraggedheader style.

```
9615 \compatglossarystyle{longraggedheader}{%  
9616 \csuse{@glscompstyle@longragged}%  
9617 }%
```

Backward compatible longraggedheaderborder style.

```
9618 \compatglossarystyle{longraggedheaderborder}{%  
9619 \csuse{@glscompstyle@longragged}%  
9620 }%
```

Backward compatible longragged3col style.

```
9621 \compatglossarystyle{longragged3col}{%  
9622 \renewcommand*{\glossaryentryfield}[5]{%  
9623 \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%  
9624 \renewcommand*{\glossarysubentryfield}[6]{%  
9625 &  
9626 \glssubentryitem{##2}%  
9627 \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%  
9628 }%
```

Backward compatible longragged3colborder style.

```
9629 \compatglossarystyle{longragged3colborder}{%  
9630 \csuse{@glscompstyle@longragged3col}%  
9631 }%
```

Backward compatible longragged3colheader style.

```
9632 \compatglossarystyle{longragged3colheader}{%  
9633 \csuse{@glscompstyle@longragged3col}%  
9634 }%
```

Backward compatible longragged3colheaderborder style.

```
9635 \compatglossarystyle{longragged3colheaderborder}{%  
9636 \csuse{@glscompstyle@longragged3col}%  
9637 }%
```

Backward compatible altlongragged4col style.

```
9638 \compatglossarystyle{altlongragged4col}{%  
9639 \renewcommand*{\glossaryentryfield}[5]{%  
9640 \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%  
9641 \renewcommand*{\glossarysubentryfield}[6]{%  
9642 &  
9643 \glssubentryitem{##2}%  
9644 \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%  
9645 }%
```

Backward compatible altlongragged4colheader style.

```
9646 \compatglossarystyle{altlongragged4colheader}{%  
9647 \csuse{@glscompstyle@altlong4col}%  
9648 }%
```

Backward compatible altlongragged4colborder style.

```
9649 \compatglossarystyle{altlongragged4colborder}{%
```

```

9650 \csuse{@glscompstyle@altlong4col}%
9651 }%

```

Backward compatible altlongragged4colheaderborder style.

```

9652 \compatglossarystyle{altlongragged4colheaderborder}{%
9653 \csuse{@glscompstyle@altlong4col}%
9654 }%

```

Backward compatible index style.

```

9655 \compatglossarystyle{index}{%
9656 \renewcommand*{\glossaryentryfield}[5]{%
9657 \item\glsentryitem{##1}\textbf{\glstarget{##1}{##2}}%
9658 \ifx\relax##4\relax
9659 \else
9660 \space{##4}%
9661 \fi
9662 \space ##3\glspostdescription \space ##5}%
9663 \renewcommand*{\glossarysubentryfield}[6]{%
9664 \ifcase##1\relax
9665 % level 0
9666 \item
9667 \or
9668 % level 1
9669 \subitem
9670 \glssubentryitem{##2}%
9671 \else
9672 % all other levels
9673 \subsubitem
9674 \fi
9675 \textbf{\glstarget{##2}{##3}}%
9676 \ifx\relax##5\relax
9677 \else
9678 \space{##5}%
9679 \fi
9680 \space##4\glspostdescription\space ##6}%
9681 }%

```

Backward compatible indexgroup style.

```

9682 \compatglossarystyle{indexgroup}{%
9683 \csuse{@glscompstyle@index}%
9684 }%

```

Backward compatible indexhypergroup style.

```

9685 \compatglossarystyle{indexhypergroup}{%
9686 \csuse{@glscompstyle@index}%
9687 }%

```

Backward compatible tree style.

```

9688 \compatglossarystyle{tree}{%
9689 \renewcommand{\glossaryentryfield}[5]{%
9690 \hangindent0pt\relax

```



```

9691 \parindent0pt\relax
9692 \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
9693 \ifx\relax##4\relax
9694 \else
9695 \space{##4}%
9696 \fi
9697 \space ##3\glspostdescription \space ##5\par}%
9698 \renewcommand{\glossarysubentryfield}[6]{%
9699 \hangindent##1\glstreeindent\relax
9700 \parindent##1\glstreeindent\relax
9701 \ifnum##1=1\relax
9702 \glssubentryitem{##2}%
9703 \fi
9704 \textbf{\glstarget{##2}{##3}}%
9705 \ifx\relax##5\relax
9706 \else
9707 \space{##5}%
9708 \fi
9709 \space##4\glspostdescription\space ##6\par}%
9710 }%

```

Backward compatible treegroup style.

```

9711 \compatglossarystyle{treegroup}{%
9712 \csuse{@glscmpstyle@tree}%
9713 }%

```

Backward compatible treehypergroup style.

```

9714 \compatglossarystyle{treehypergroup}{%
9715 \csuse{@glscmpstyle@tree}%
9716 }%

```

Backward compatible treenoname style.

```

9717 \compatglossarystyle{treenoname}{%
9718 \renewcommand{\glossaryentryfield}[5]{%
9719 \hangindent0pt\relax
9720 \parindent0pt\relax
9721 \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
9722 \ifx\relax##4\relax
9723 \else
9724 \space{##4}%
9725 \fi
9726 \space ##3\glspostdescription \space ##5\par}%
9727 \renewcommand{\glossarysubentryfield}[6]{%
9728 \hangindent##1\glstreeindent\relax
9729 \parindent##1\glstreeindent\relax
9730 \ifnum##1=1\relax
9731 \glssubentryitem{##2}%
9732 \fi
9733 \glstarget{##2}{\strut}%
9734 ##4\glspostdescription\space ##6\par}%
9735 }%

```

Backward compatible treenonamegroup style.

```
9736 \compatglossarystyle{treenonamegroup}{%
9737   \csuse{@glscompstyle@treenoname}%
9738 }%
```

Backward compatible treenonamehypergroup style.

```
9739 \compatglossarystyle{treenonamehypergroup}{%
9740   \csuse{@glscompstyle@treenoname}%
9741 }%
```

Backward compatible alttree style.

```
9742 \compatglossarystyle{alttree}{%
9743   \renewcommand{\glossaryentryfield}[5]{%
9744     \ifnum\@gls@prevlevel=0\relax
9745     \else
9746       \settowidth{\glstreeindent}{\textbf{\@glswidestname\space}}%
9747       \hangindent\glstreeindent
9748       \parindent\glstreeindent
9749     \fi
9750     \makebox[0pt][r]{\makebox[\glstreeindent][l]{%
9751       \glssentryitem{##1}\textbf{\glstarget{##1}{##2}}}%
9752     \ifx\relax##4\relax
9753     \else
9754       (##4)\space
9755     \fi
9756     ##3\glspostdescription \space ##5\par
9757     \def\@gls@prevlevel{0}%
9758   }%
9759   \renewcommand{\glossarysubentryfield}[6]{%
9760     \ifnum##1=1\relax
9761       \glssubentryitem{##2}%
9762     \fi
9763     \ifnum\@gls@prevlevel=##1\relax
9764     \else
9765       \@ifundefined{@glswidestname\romannumeral##1}{%
9766         \settowidth{\gls@tmplen}{\textbf{\@glswidestname\space}}{%
9767         \settowidth{\gls@tmplen}{\textbf{%
9768           \csname @glswidestname\romannumeral##1\endcsname\space}}}%
9769       \ifnum\@gls@prevlevel<##1\relax
9770         \setlength\glstreeindent\gls@tmplen
9771         \addtolength\glstreeindent\parindent
9772         \parindent\glstreeindent
9773       \else
9774         \@ifundefined{@glswidestname\romannumeral\@gls@prevlevel}{%
9775         \settowidth{\glstreeindent}{\textbf{%
9776           \@glswidestname\space}}{%
9777         \settowidth{\glstreeindent}{\textbf{%
9778           \csname @glswidestname\romannumeral\@gls@prevlevel
9779           \endcsname\space}}}%
9780         \addtolength\parindent{-\glstreeindent}%

```

```

9781         \setlength\glstreeindent\parindent
9782     \fi
9783 \fi
9784 \hangindent\glstreeindent
9785 \makebox[0pt][r]{\makebox[\glstemplen][l]{%
9786     \textbf{\glstarget{##2}{##3}}}%
9787 \ifx##5\relax\relax
9788 \else
9789     (##5)\space
9790 \fi
9791 ##4\glspostdescription\space ##6\par
9792 \def\@gls@prevlevel{##1}%
9793 }%
9794 }%

```

Backward compatible alttreegroup style.

```

9795 \compatglossarystyle{alttreegroup}{%
9796 \csuse{@glscompstyle@alttree}%
9797 }%

```

Backward compatible alttreehypergroup style.

```

9798 \compatglossarystyle{alttreehypergroup}{%
9799 \csuse{@glscompstyle@alttree}%
9800 }%

```

Backward compatible mcolindex style.

```

9801 \compatglossarystyle{mcolindex}{%
9802 \csuse{@glscompstyle@index}%
9803 }%

```

Backward compatible mcolindexgroup style.

```

9804 \compatglossarystyle{mcolindexgroup}{%
9805 \csuse{@glscompstyle@index}%
9806 }%

```

Backward compatible mcolindexhypergroup style.

```

9807 \compatglossarystyle{mcolindexhypergroup}{%
9808 \csuse{@glscompstyle@index}%
9809 }%

```

Backward compatible mcoltree style.

```

9810 \compatglossarystyle{mcoltree}{%
9811 \csuse{@glscompstyle@tree}%
9812 }%

```

Backward compatible mcoltreegroup style.

```

9813 \compatglossarystyle{mcolindextreegroup}{%
9814 \csuse{@glscompstyle@tree}%
9815 }%

```

Backward compatible mcoltreehypergroup style.

```

9816 \compatglossarystyle{mcolindextreehypergroup}{%

```

9817 \csuse{@glscompstyle@tree}%
 9818 }%

Backward compatible mcoltreenoname style.

9819 \compatglossarystyle{mcoltreenoname}{%
 9820 \csuse{@glscompstyle@tree}%
 9821 }%

Backward compatible mcoltreenonamegroup style.

9822 \compatglossarystyle{mcoltreenonamegroup}{%
 9823 \csuse{@glscompstyle@tree}%
 9824 }%

Backward compatible mcoltreenonamehypergroup style.

9825 \compatglossarystyle{mcoltreenonamehypergroup}{%
 9826 \csuse{@glscompstyle@tree}%
 9827 }%

Backward compatible mcolalttree style.

9828 \compatglossarystyle{mcolalttree}{%
 9829 \csuse{@glscompstyle@alttree}%
 9830 }%

Backward compatible mcolalttreegroup style.

9831 \compatglossarystyle{mcolalttreegroup}{%
 9832 \csuse{@glscompstyle@alttree}%
 9833 }%

Backward compatible mcolalttreehypergroup style.

9834 \compatglossarystyle{mcolalttreehypergroup}{%
 9835 \csuse{@glscompstyle@alttree}%
 9836 }%

Backward compatible superragged style.

9837 \compatglossarystyle{superragged}{%
 9838 \renewcommand*{\glossaryentryfield}[5]{%
 9839 \glsentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
 9840 \tabularnewline}%
 9841 \renewcommand*{\glossarysubentryfield}[6]{%
 9842 &
 9843 \glssubentryitem{##2}%
 9844 \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
 9845 \tabularnewline}%
 9846 }%

Backward compatible superraggedborder style.

9847 \compatglossarystyle{superraggedborder}{%
 9848 \csuse{@glscompstyle@superragged}%
 9849 }%

Backward compatible superraggedheader style.

9850 \compatglossarystyle{superraggedheader}{%
 9851 \csuse{@glscompstyle@superragged}%
 9852 }%

Backward compatible superraggedheaderborder style.

```
9853 \compatglossarystyle{superraggedheaderborder}{%
9854   \csuse{@glscompstyle@superragged}%
9855 }%
```

Backward compatible superragged3col style.

```
9856 \compatglossarystyle{superragged3col}{%
9857   \renewcommand*{\glossaryentryfield}[5]{%
9858     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%
9859   \renewcommand*{\glossarysubentryfield}[6]{%
9860     &
9861     \glssubentryitem{##2}%
9862     \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%
9863 }%
```

Backward compatible superragged3colborder style.

```
9864 \compatglossarystyle{superragged3colborder}{%
9865   \csuse{@glscompstyle@superragged3col}%
9866 }%
```

Backward compatible superragged3colheader style.

```
9867 \compatglossarystyle{superragged3colheader}{%
9868   \csuse{@glscompstyle@superragged3col}%
9869 }%
```

Backward compatible superragged3colheaderborder style.

```
9870 \compatglossarystyle{superragged3colheaderborder}{%
9871   \csuse{@glscompstyle@superragged3col}%
9872 }%
```

Backward compatible altsuperragged4col style.

```
9873 \compatglossarystyle{altsuperragged4col}{%
9874   \renewcommand*{\glossaryentryfield}[5]{%
9875     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%
9876   \renewcommand*{\glossarysubentryfield}[6]{%
9877     &
9878     \glssubentryitem{##2}%
9879     \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%
9880 }%
```

Backward compatible altsuperragged4colheader style.

```
9881 \compatglossarystyle{altsuperragged4colheader}{%
9882   \csuse{@glscompstyle@altsuperragged4col}%
9883 }%
```

Backward compatible altsuperragged4colborder style.

```
9884 \compatglossarystyle{altsuperragged4colborder}{%
9885   \csuse{@glscompstyle@altsuperragged4col}%
9886 }%
```

Backward compatible altsuperragged4colheaderborder style.

```
9887 \compatglossarystyle{altsuperragged4colheaderborder}{%
```

```

9888 \csuse{@glscompstyle@altsuperragged4col}%
9889 }%

```

Backward compatible super style.

```

9890 \compatglossarystyle{super}{%
9891   \renewcommand*{\glossaryentryfield}[5]{%
9892     \glstarget{##1}{\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
9893   \renewcommand*{\glossarysubentryfield}[6]{%
9894     &
9895     \glssubentryitem{##2}%
9896     \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
9897 }%

```

Backward compatible superborder style.

```

9898 \compatglossarystyle{superborder}{%
9899 \csuse{@glscompstyle@super}%
9900 }%

```

Backward compatible superheader style.

```

9901 \compatglossarystyle{superheader}{%
9902 \csuse{@glscompstyle@super}%
9903 }%

```

Backward compatible superheaderborder style.

```

9904 \compatglossarystyle{superheaderborder}{%
9905 \csuse{@glscompstyle@super}%
9906 }%

```

Backward compatible super3col style.

```

9907 \compatglossarystyle{super3col}{%
9908   \renewcommand*{\glossaryentryfield}[5]{%
9909     \glstarget{##1}{\glstarget{##1}{##2} & ##3 & ##5\\}%
9910   \renewcommand*{\glossarysubentryfield}[6]{%
9911     &
9912     \glssubentryitem{##2}%
9913     \glstarget{##2}{\strut}##4 & ##6\\}%
9914 }%

```

Backward compatible super3colborder style.

```

9915 \compatglossarystyle{super3colborder}{%
9916 \csuse{@glscompstyle@super3col}%
9917 }%

```

Backward compatible super3colheader style.

```

9918 \compatglossarystyle{super3colheader}{%
9919 \csuse{@glscompstyle@super3col}%
9920 }%

```

Backward compatible super3colheaderborder style.

```

9921 \compatglossarystyle{super3colheaderborder}{%
9922 \csuse{@glscompstyle@super3col}%
9923 }%

```

Backward compatible super4col style.

```
9924 \compatglossarystyle{super4col}{%
9925   \renewcommand*{\glossaryentryfield}[5]{%
9926     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%
9927   \renewcommand*{\glossarysubentryfield}[6]{%
9928     &
9929     \glssubentryitem{##2}%
9930     \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%
9931 }%
```

Backward compatible super4colheader style.

```
9932 \compatglossarystyle{super4colheader}{%
9933   \csuse{@glscompstyle@super4col}%
9934 }%
```

Backward compatible super4colborder style.

```
9935 \compatglossarystyle{super4colborder}{%
9936   \csuse{@glscompstyle@super4col}%
9937 }%
```

Backward compatible super4colheaderborder style.

```
9938 \compatglossarystyle{super4colheaderborder}{%
9939   \csuse{@glscompstyle@super4col}%
9940 }%
```

Backward compatible altsuper4col style.

```
9941 \compatglossarystyle{altsuper4col}{%
9942   \csuse{@glscompstyle@super4col}%
9943 }%
```

Backward compatible altsuper4colheader style.

```
9944 \compatglossarystyle{altsuper4colheader}{%
9945   \csuse{@glscompstyle@super4col}%
9946 }%
```

Backward compatible altsuper4colborder style.

```
9947 \compatglossarystyle{altsuper4colborder}{%
9948   \csuse{@glscompstyle@super4col}%
9949 }%
```

Backward compatible altsuper4colheaderborder style.

```
9950 \compatglossarystyle{altsuper4colheaderborder}{%
9951   \csuse{@glscompstyle@super4col}%
9952 }%
```

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.

```
9953 \NeedsTeXFormat{LaTeX2e}
```

Package version number now in line with main glossaries package number.

```
9954 \ProvidesPackage{glossaries-accsupp}[2016/04/30 v4.23 (NLCT)]
```

```
9955 Experimental glossaries accessibility]
```

Pass all options to glossaries:

```
9956 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
9957 \ProcessOptions
```

This package should be loaded before glossaries-extra, so complain if that has already been loaded.

```
9958 \@ifpackageloaded{glossaries-extra}
```

```
9959 {%
```

```
9960 \PackageWarning{glossaries-accsupp}{The ‘glossaries-accsupp’
```

```
9961 package has been loaded after the ‘glossaries-extra’
```

```
9962 package. This can cause a failure to integrate both
```

```
9963 packages. Either use the ‘accsupp’ option when you
```

```
9964 load ‘glossaries-extra’ or load ‘glossaries-accsupp’
```

```
9965 before loading ‘glossaries-extra’}%
```

```
9966 }
```

```
9967 {}
```

`\compatibleglossentry` Override style compatibility macros:

```
9968 \def\compatibleglossentry#1#2{%
```

```
9969 \toks@{#2}%
```

```
9970 \protected@edef\@do@glossentry{%
```

```
9971 \noexpand\accsuppglossaryentryfield{#1}%
```

```
9972 {\noexpand\glsnamefont
```

```
9973 {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@name\endcsname}}%
```

```
9974 {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@desc\endcsname}%
```

```
9975 {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@symbol\endcsname}%
```

```
9976 {\the\toks@}%
```

```
9977 }%
```

```
9978 \@do@glossentry
```

```
9979 }
```


lesubglossentry

```
9980 \def\compatiblesubglossentry#1#2#3{%
9981   \toks@{#3}%
9982   \protected@edef\@do@subglossentry{%
9983     \noexpand\accsuppglossarysubentryfield{\number#1}%
9984     {#2}%
9985     {\noexpand\glsnamefont
9986       {\expandafter\expandonce\csname glo@\glsdetoklabel{#2}@name\endcsname}}%
9987     {\expandafter\expandonce\csname glo@\glsdetoklabel{#2}@desc\endcsname}%
9988     {\expandafter\expandonce\csname glo@\glsdetoklabel{#2}@symbol\endcsname}%
9989     {\the\toks@}%
9990   }%
9991   \@do@subglossentry
9992 }
```

Required packages:

```
9993 \RequirePackage{glossaries}
9994 \RequirePackage{accsupp}
```

5.1 Defining Replacement Text

The version 0.1 stored the replacement text in the symbol key. This has been changed to use the new keys defined here. Example of use:

```
\newglossaryentry{dr}{name=Dr,description={},access={Doctor}}
```

access The replacement text corresponding to the name key:

```
9995 \define@key{glossentry}{access}{%
9996   \def\@glo@access{#1}%
9997 }
```

textaccess The replacement text corresponding to the text key:

```
9998 \define@key{glossentry}{textaccess}{%
9999   \def\@glo@textaccess{#1}%
10000 }
```

firstaccess The replacement text corresponding to the first key:

```
10001 \define@key{glossentry}{firstaccess}{%
10002   \def\@glo@firstaccess{#1}%
10003 }
```

pluralaccess The replacement text corresponding to the plural key:

```
10004 \define@key{glossentry}{pluralaccess}{%
10005   \def\@glo@pluralaccess{#1}%
10006 }
```

firstpluralaccess The replacement text corresponding to the firstplural key:

```
10007 \define@key{glossentry}{firstpluralaccess}{%
10008   \def\@glo@firstpluralaccess{#1}%
10009 }
```

symbolaccess The replacement text corresponding to the symbol key:

```
10010 \define@key{glossentry}{symbolaccess}{%
10011   \def\@glo@symbolaccess{#1}%
10012 }
```

symbolpluralaccess The replacement text corresponding to the symbolplural key:

```
10013 \define@key{glossentry}{symbolpluralaccess}{%
10014   \def\@glo@symbolpluralaccess{#1}%
10015 }
```

descriptionaccess The replacement text corresponding to the description key:

```
10016 \define@key{glossentry}{descriptionaccess}{%
10017   \def\@glo@descaccess{#1}%
10018 }
```

descriptionpluralaccess The replacement text corresponding to the descriptionplural key:

```
10019 \define@key{glossentry}{descriptionpluralaccess}{%
10020   \def\@glo@descpluralaccess{#1}%
10021 }
```

shortaccess The replacement text corresponding to the short key:

```
10022 \define@key{glossentry}{shortaccess}{%
10023   \def\@glo@shortaccess{#1}%
10024 }
```

shortpluralaccess The replacement text corresponding to the shortplural key:

```
10025 \define@key{glossentry}{shortpluralaccess}{%
10026   \def\@glo@shortpluralaccess{#1}%
10027 }
```

longaccess The replacement text corresponding to the long key:

```
10028 \define@key{glossentry}{longaccess}{%
10029   \def\@glo@longaccess{#1}%
10030 }
```

longpluralaccess The replacement text corresponding to the longplural key:

```
10031 \define@key{glossentry}{longpluralaccess}{%
10032   \def\@glo@longpluralaccess{#1}%
10033 }
```

There are no equivalent keys for the user1...user6 keys. The replacement text would have to be explicitly put in the value, e.g., user1={\glsacccsupp{inches}{in}}.

Append these new keys to \@gls@keymap:

```

10034 \appto\@gls@keymap{,%
10035   {access}{access},%
10036   {textaccess}{textaccess},%
10037   {firstaccess}{firstaccess},%
10038   {pluralaccess}{pluralaccess},%
10039   {firstpluralaccess}{firstpluralaccess},%
10040   {symbolaccess}{symbolaccess},%
10041   {symbolpluralaccess}{symbolpluralaccess},%
10042   {descaccess}{descaccess},%
10043   {descpluralaccess}{descpluralaccess},%
10044   {shortaccess}{shortaccess},%
10045   {shortpluralaccess}{shortpluralaccess},%
10046   {longaccess}{longaccess},%
10047   {longpluralaccess}{longpluralaccess}%
10048 }
```

\@gls@noaccess Indicates that no replacement text has been provided.

```

10049 \def\@gls@noaccess{\relax}
```

Add to the start hook (the access key is initialised to the value of the symbol key at the start for backwards compatibility):

```

10050 \let\@gls@oldnewglossaryentryprehook\@newglossaryentryprehook
10051 \renewcommand*{\@newglossaryentryprehook}{%
10052   \@gls@oldnewglossaryentryprehook
10053   \def\@glo@access{\@glo@symbol}%
```

Initialise the other keys:

```

10054   \def\@glo@textaccess{\@glo@access}%
10055   \def\@glo@firstaccess{\@glo@access}%
10056   \def\@glo@pluralaccess{\@glo@textaccess}%
10057   \def\@glo@firstpluralaccess{\@glo@pluralaccess}%
10058   \def\@glo@symbolaccess{\relax}%
10059   \def\@glo@symbolpluralaccess{\@glo@symbolaccess}%
10060   \def\@glo@descaccess{\relax}%
10061   \def\@glo@descpluralaccess{\@glo@descaccess}%
10062   \def\@glo@shortaccess{\relax}%
10063   \def\@glo@shortpluralaccess{\@glo@shortaccess}%
10064   \def\@glo@longaccess{\relax}%
10065   \def\@glo@longpluralaccess{\@glo@longaccess}%
10066 }
```

Add to the end hook:

```

10067 \let\@gls@oldnewglossaryentryposthook\@newglossaryentryposthook
10068 \renewcommand*{\@newglossaryentryposthook}{%
10069   \@gls@oldnewglossaryentryposthook
```

Store the access information:

```

10070   \expandafter
10071   \protected@xdef\csname glo@\@glo@label @access\endcsname{%
```

```

10072      \@glo@access}%
10073 \expandafter
10074   \protected@xdef\csname glo@\@glo@label @textaccess\endcsname{%
10075     \@glo@textaccess}%
10076 \expandafter
10077   \protected@xdef\csname glo@\@glo@label @firstaccess\endcsname{%
10078     \@glo@firstaccess}%
10079 \expandafter
10080   \protected@xdef\csname glo@\@glo@label @pluralaccess\endcsname{%
10081     \@glo@pluralaccess}%
10082 \expandafter
10083   \protected@xdef\csname glo@\@glo@label @firstpluralaccess\endcsname{%
10084     \@glo@firstpluralaccess}%
10085 \expandafter
10086   \protected@xdef\csname glo@\@glo@label @symbolaccess\endcsname{%
10087     \@glo@symbolaccess}%
10088 \expandafter
10089   \protected@xdef\csname glo@\@glo@label @symbolpluralaccess\endcsname{%
10090     \@glo@symbolpluralaccess}%
10091 \expandafter
10092   \protected@xdef\csname glo@\@glo@label @descaccess\endcsname{%
10093     \@glo@descaccess}%
10094 \expandafter
10095   \protected@xdef\csname glo@\@glo@label @descpluralaccess\endcsname{%
10096     \@glo@descpluralaccess}%
10097 \expandafter
10098   \protected@xdef\csname glo@\@glo@label @shortaccess\endcsname{%
10099     \@glo@shortaccess}%
10100 \expandafter
10101   \protected@xdef\csname glo@\@glo@label @shortpluralaccess\endcsname{%
10102     \@glo@shortpluralaccess}%
10103 \expandafter
10104   \protected@xdef\csname glo@\@glo@label @longaccess\endcsname{%
10105     \@glo@longaccess}%
10106 \expandafter
10107   \protected@xdef\csname glo@\@glo@label @longpluralaccess\endcsname{%
10108     \@glo@longpluralaccess}%
10109 }

```

5.2 Accessing Replacement Text

`\glsentryaccess` Get the value of the access key for the entry with the given label:

```

10110 \newcommand*{\glsentryaccess}[1]{%
10111   \@gls@entry@field{#1}{access}%
10112 }

```

`\glsentrytextaccess` Get the value of the textaccess key for the entry with the given label:

```

10113 \newcommand*{\glsentrytextaccess}[1]{%

```

```

10114 \@gls@entry@field{#1}{textaccess}%
10115 }

entryfirstaccess Get the value of the firstaccess key for the entry with the given label:
10116 \newcommand*{\glsentryfirstaccess}[1]{%
10117 \@gls@entry@field{#1}{firstaccess}%
10118 }

entrypluralaccess Get the value of the pluralaccess key for the entry with the given label:
10119 \newcommand*{\glsentrypluralaccess}[1]{%
10120 \@gls@entry@field{#1}{pluralaccess}%
10121 }

entryfirstpluralaccess Get the value of the firstpluralaccess key for the entry with the given label:
10122 \newcommand*{\glsentryfirstpluralaccess}[1]{%
10123 \csname glo@#1@firstpluralaccess\endcsname
10124 }

entrysymbolaccess Get the value of the symbolaccess key for the entry with the given label:
10125 \newcommand*{\glsentrysymbolaccess}[1]{%
10126 \@gls@entry@field{#1}{symbolaccess}%
10127 }

entrysymbolpluralaccess Get the value of the symbolpluralaccess key for the entry with the given label:
10128 \newcommand*{\glsentrysymbolpluralaccess}[1]{%
10129 \@gls@entry@field{#1}{symbolpluralaccess}%
10130 }

entrydescaccess Get the value of the descriptionaccess key for the entry with the given label:
10131 \newcommand*{\glsentrydescaccess}[1]{%
10132 \@gls@entry@field{#1}{descaccess}%
10133 }

entrydescpluralaccess Get the value of the descriptionpluralaccess key for the entry with the given label:
10134 \newcommand*{\glsentrydescpluralaccess}[1]{%
10135 \@gls@entry@field{#1}{descaccess}%
10136 }

entryshortaccess Get the value of the shortaccess key for the entry with the given label:
10137 \newcommand*{\glsentryshortaccess}[1]{%
10138 \@gls@entry@field{#1}{shortaccess}%
10139 }

entryshortpluralaccess Get the value of the shortpluralaccess key for the entry with the given label:
10140 \newcommand*{\glsentryshortpluralaccess}[1]{%
10141 \@gls@entry@field{#1}{shortpluralaccess}%
10142 }

```

entrylongaccess Get the value of the longaccess key for the entry with the given label:

```
10143 \newcommand*{\glsentrylongaccess}[1]{%
10144   \@gls@entry@field{#1}{longaccess}%
10145 }
```

ongpluralaccess Get the value of the longpluralaccess key for the entry with the given label:

```
10146 \newcommand*{\glsentrylongpluralaccess}[1]{%
10147   \@gls@entry@field{#1}{longpluralaccess}%
10148 }
```

\glsaccsupp \glsaccsupp{<replacement text>}{<text>}

This can be redefined to use E or Alt instead of ActualText. (I don't have the software to test the E or Alt options.)

```
10149 \newcommand*{\glsaccsupp}[2]{%
10150   \BeginAccSupp{ActualText=#1}#2\EndAccSupp{}%
10151 }
```

\xglsaccsupp Fully expands replacement text before calling \glsaccsupp

```
10152 \newcommand*{\xglsaccsupp}[2]{%
10153   \protected@edef\@gls@replacementtext{#1}%
10154   \expandafter\glsaccsupp\expandafter{\@gls@replacementtext}{#2}%
10155 }
```

@access@display

```
10156 \newcommand*{\@gls@access@display}[2]{%
10157   \protected@edef\@glo@access{#2}%
10158   \ifx\@glo@access\@gls@noaccess
10159     #1%
10160   \else
10161     \xglsaccsupp{\@glo@access}{#1}%
10162   \fi
10163 }
```

meaccessdisplay Displays the first argument with the accessibility text for the entry with the label given by the second argument (if set).

```
10164 \DeclareRobustCommand*{\glsnameaccessdisplay}[2]{%
10165   \@gls@access@display{#1}{\glsentryaccess{#2}}%
10166 }
```

xtaccessdisplay As above but for the textaccess replacement text.

```
10167 \DeclareRobustCommand*{\glsstextaccessdisplay}[2]{%
10168   \@gls@access@display{#1}{\glsentrytextaccess{#2}}%
10169 }
```

alaccessdisplay As above but for the pluralaccess replacement text.

```
10170 \DeclareRobustCommand*{\glspluralaccessdisplay}[2]{%
10171   \@gls@access@display{#1}{\glsentrypluralaccess{#2}}%
10172 }
```

staccessdisplay As above but for the firstaccess replacement text.

```

10173 \DeclareRobustCommand*\glfirstaccessdisplay}[2]{%
10174   \@gls@access@display{#1}{\glentryfirstaccess{#2}}%
10175 }

```

alaccessdisplay As above but for the firstpluralaccess replacement text.

```

10176 \DeclareRobustCommand*\glfirstpluralaccessdisplay}[2]{%
10177   \@gls@access@display{#1}{\glentryfirstpluralaccess{#2}}%
10178 }

```

olaccessdisplay As above but for the symbolaccess replacement text.

```

10179 \DeclareRobustCommand*\glssymbolaccessdisplay}[2]{%
10180   \@gls@access@display{#1}{\glentrysymbolaccess{#2}}%
10181 }

```

alaccessdisplay As above but for the symbolpluralaccess replacement text.

```

10182 \DeclareRobustCommand*\glssymbolpluralaccessdisplay}[2]{%
10183   \@gls@access@display{#1}{\glentrysymbolpluralaccess{#2}}%
10184 }

```

onaccessdisplay As above but for the descriptionaccess replacement text.

```

10185 \DeclareRobustCommand*\glsdescriptionaccessdisplay}[2]{%
10186   \@gls@access@display{#1}{\glentrydescaccess{#2}}%
10187 }

```

alaccessdisplay As above but for the descriptionpluralaccess replacement text.

```

10188 \DeclareRobustCommand*\glsdescriptionpluralaccessdisplay}[2]{%
10189   \@gls@access@display{#1}{\glentrydescpluralaccess{#2}}%
10190 }

```

rtaccessdisplay As above but for the shortaccess replacement text.

```

10191 \DeclareRobustCommand*\glsshortaccessdisplay}[2]{%
10192   \@gls@access@display{#1}{\glentryshortaccess{#2}}%
10193 }

```

alaccessdisplay As above but for the shortpluralaccess replacement text.

```

10194 \DeclareRobustCommand*\glsshortpluralaccessdisplay}[2]{%
10195   \@gls@access@display{#1}{\glentryshortpluralaccess{#2}}%
10196 }

```

ngaccessdisplay As above but for the longaccess replacement text.

```

10197 \DeclareRobustCommand*\glslongaccessdisplay}[2]{%
10198   \@gls@access@display{#1}{\glentrylongaccess{#2}}%
10199 }

```

alaccessdisplay As above but for the longpluralaccess replacement text.

```

10200 \DeclareRobustCommand*\glslongpluralaccessdisplay}[2]{%
10201   \@gls@access@display{#1}{\glentrylongpluralaccess{#2}}%
10202 }

```

`\glsaccessdisplay` Gets the replacement text corresponding to the named key given by the first argument and calls the appropriate command defined above.

```

10203 \DeclareRobustCommand*\glsaccessdisplay}[3]{%
10204   \@ifundefined{gls#1accessdisplay}%
10205   {%
10206     \PackageError{glossaries-accsupp}{No accessibility support
10207       for key ‘#1’}{}%
10208   }%
10209   {%
10210     \csname gls#1accessdisplay\endcsname{#2}{#3}%
10211   }%
10212 }

```

`\default@entryfmt` Redefine the default entry format to use accessibility information

```

10213 \renewcommand*\@@gls@default@entryfmt}[2]{%
10214   \ifdefempty\glscustomtext
10215   {%
10216     \glsifplural
10217     {%

```

Plural form

```

10218     \glscapscase
10219     {%

```

Don't adjust case

```

10220     \ifglsused\glslabel
10221     {%

```

Subsequent use

```

10222     #2{\glspluralaccessdisplay
10223       {\glsentryplural{\glslabel}}{\glslabel}}%
10224     {\glsdescriptionpluralaccessdisplay
10225       {\glsentrydescplural{\glslabel}}{\glslabel}}%
10226     {\glsymbolpluralaccessdisplay
10227       {\glsentrysymbolplural{\glslabel}}{\glslabel}}
10228     {\glsinsert}%
10229   }%
10230   {%

```

First use

```

10231     #1{\glsfirstpluralaccessdisplay
10232       {\glsentryfirstplural{\glslabel}}{\glslabel}}%
10233     {\glsdescriptionpluralaccessdisplay
10234       {\glsentrydescplural{\glslabel}}{\glslabel}}%
10235     {\glsymbolpluralaccessdisplay
10236       {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10237     {\glsinsert}%
10238   }%
10239   }%
10240   {%

```


Make first letter upper case

```
10241      \ifglsused\glslabel
10242      {%
```

Subsequent use.

```
10243      #2{\glspluralaccessdisplay
10244          {\Glsentryplural{\glslabel}}{\glslabel}}%
10245          {\glsdescriptionpluralaccessdisplay
10246          {\glsentrydescplural{\glslabel}}{\glslabel}}%
10247          {\glsymbolpluralaccessdisplay
10248          {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10249          {\glsinsert}}%
10250      }%
10251      {%
```

First use

```
10252      #1{\glsfirstpluralaccessdisplay
10253          {\Glsentryfirstplural{\glslabel}}{\glslabel}}%
10254          {\glsdescriptionpluralaccessdisplay
10255          {\glsentrydescplural{\glslabel}}{\glslabel}}%
10256          {\glsymbolpluralaccessdisplay
10257          {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10258          {\glsinsert}}%
10259      }%
10260      }%
10261      {%
```

Make all upper case

```
10262      \ifglsused\glslabel
10263      {%
```

Subsequent use

```
10264      \MakeUppercase{%
10265      #2{\glspluralaccessdisplay
10266          {\glsentryplural{\glslabel}}{\glslabel}}%
10267          {\glsdescriptionpluralaccessdisplay
10268          {\glsentrydescplural{\glslabel}}{\glslabel}}%
10269          {\glsymbolpluralaccessdisplay
10270          {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10271          {\glsinsert}}}%
10272      }%
10273      {%
```

First use

```
10274      \MakeUppercase{%
10275      #1{\glsfirstpluralaccessdisplay
10276          {\glsentryfirstplural{\glslabel}}{\glslabel}}%
10277          {\glsdescriptionpluralaccessdisplay
10278          {\glsentrydescplural{\glslabel}}{\glslabel}}%
10279          {\glsymbolpluralaccessdisplay
10280          {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
```

```

10281             {\glsinsert}}}%
10282         }%
10283     }%
10284 }%
10285 {%

```

Singular form

```

10286     \glscapscase
10287     {%

```

Don't adjust case

```

10288     \ifglsused\glslabel
10289     {%

```

Subsequent use

```

10290     #2{\glstextaccessdisplay
10291         {\glsentrytext{\glslabel}}{\glslabel}}%
10292     {\glsdescriptionaccessdisplay
10293         {\glsentrydesc{\glslabel}}{\glslabel}}%
10294     {\glssymbolaccessdisplay
10295         {\glsentrysymbol{\glslabel}}{\glslabel}}%
10296     {\glsinsert}}%
10297 }%
10298 {%

```

First use

```

10299     #1{\glsfirstaccessdisplay
10300         {\glsentryfirst{\glslabel}}{\glslabel}}%
10301     {\glsdescriptionaccessdisplay
10302         {\glsentrydesc{\glslabel}}{\glslabel}}%
10303     {\glssymbolaccessdisplay
10304         {\glsentrysymbol{\glslabel}}{\glslabel}}%
10305     {\glsinsert}}%
10306 }%
10307 }%
10308 {%

```

Make first letter upper case

```

10309     \ifglsused\glslabel
10310     {%

```

Subsequent use

```

10311     #2{\glstextaccessdisplay
10312         {\Glsentrytext{\glslabel}}{\glslabel}}%
10313     {\glsdescriptionaccessdisplay
10314         {\glsentrydesc{\glslabel}}{\glslabel}}%
10315     {\glssymbolaccessdisplay
10316         {\glsentrysymbol{\glslabel}}{\glslabel}}%
10317     {\glsinsert}}%
10318 }%
10319 {%

```

First use

```

10320      #1{\glsfirstaccessdisplay
10321          {\Glsentryfirst{\glslabel}}{\glslabel}}%
10322          {\glsdescriptionaccessdisplay
10323              {\glsentrydesc{\glslabel}}{\glslabel}}%
10324          {\glssymbolaccessdisplay
10325              {\glsentrysymbol{\glslabel}}{\glslabel}}%
10326          {\glsinsert}}%
10327      }%
10328  }%
10329  {%

```

Make all upper case

```

10330      \ifglsused\glslabel
10331      {%

```

Subsequent use

```

10332      \MakeUppercase{%
10333          #2{\glstextaccessdisplay
10334              {\glsentrytext{\glslabel}}{\glslabel}}%
10335              {\glsdescriptionaccessdisplay
10336                  {\glsentrydesc{\glslabel}}{\glslabel}}%
10337                  {\glssymbolaccessdisplay
10338                      {\glsentrysymbol{\glslabel}}{\glslabel}}%
10339                      {\glsinsert}}}%
10340      }%
10341  {%

```

First use

```

10342      \MakeUppercase{%
10343          #1{\glsfirstaccessdisplay
10344              {\glsentryfirst{\glslabel}}{\glslabel}}%
10345              {\glsdescriptionaccessdisplay
10346                  {\glsentrydesc{\glslabel}}{\glslabel}}%
10347                  {\glssymbolaccessdisplay
10348                      {\glsentrysymbol{\glslabel}}{\glslabel}}%
10349                      {\glsinsert}}}%
10350      }%
10351  }%
10352  }%
10353  }%
10354  {%

```

Custom text provided in \glsdisp

```

10355      \ifglsused{\glslabel}%
10356      {%

```

Subsequent use

```

10357      #2{\glscustomtext}%
10358      {\glsdescriptionaccessdisplay
10359          {\glsentrydesc{\glslabel}}{\glslabel}}%

```

```

10360      {\glssymbolaccessdisplay
10361        {\glentrysymbol{\glslabel}}{\glslabel}}%
10362      {\glsinsert}%
10363    }%
10364    {%

```

First use

```

10365      #1{\glscustomtext}%
10366      {\glsdescriptionaccessdisplay
10367        {\glentrydesc{\glslabel}}{\glslabel}}%
10368      {\glssymbolaccessdisplay
10369        {\glentrysymbol{\glslabel}}{\glslabel}}%
10370      {\glsinsert}%
10371    }%
10372  }%
10373 }

```

`\glsgenentryfmt` Redefine to use accessibility information.

```

10374 \renewcommand*{\glsgenentryfmt}{%
10375   \ifdefempty\glscustomtext
10376     {%
10377       \glsifplural
10378       {%

```

Plural form

```

10379       \glscapscase
10380       {%

```

Don't adjust case

```

10381       \ifglused\glslabel
10382       {%

```

Subsequent use

```

10383       \glspluralaccessdisplay
10384         {\glentryplural{\glslabel}}{\glslabel}%
10385       \glsinsert
10386     }%
10387     {%

```

First use

```

10388       \glsfirstpluralaccessdisplay
10389         {\glentryfirstplural{\glslabel}}{\glslabel}%
10390       \glsinsert
10391     }%
10392   }%
10393   {%

```

Make first letter upper case

```

10394       \ifglused\glslabel
10395       {%

```

Subsequent use.

```
10396      \glspluralaccessdisplay
10397      {\Glsentryplural{\glslabel}}{\glslabel}%
10398      \glsinsert
10399      }%
10400      {%
```

First use

```
10401      \glsfirstpluralaccessdisplay
10402      {\Glsentryfirstplural{\glslabel}}{\glslabel}%
10403      \glsinsert
10404      }%
10405      }%
10406      {%
```

Make all upper case

```
10407      \ifglsused\glslabel
10408      {%
```

Subsequent use

```
10409      \glspluralaccessdisplay
10410      {\mfirstucMakeUppercase{\glsentryplural{\glslabel}}}%
10411      {\glslabel}%
10412      \mfirstucMakeUppercase{\glsinsert}%
10413      }%
10414      {%
```

First use

```
10415      \glsfirstpluralaccessdisplay
10416      {\mfirstucMakeUppercase{\glsentryfirstplural{\glslabel}}}%
10417      {\glslabel}%
10418      \mfirstucMakeUppercase{\glsinsert}%
10419      }%
10420      }%
10421      }%
10422      {%
```

Singular form

```
10423      \glscapscase
10424      {%
```

Don't adjust case

```
10425      \ifglsused\glslabel
10426      {%
```

Subsequent use

```
10427      \glstextaccessdisplay{\glsentrytext{\glslabel}}{\glslabel}%
10428      \glsinsert
10429      }%
10430      {%
```

First use

```
10431      \glsfirstaccessdisplay{\glsentryfirst{\glslabel}}{\glslabel}%
10432      \glsinsert
10433      }%
10434      }%
10435      {%
```

Make first letter upper case

```
10436      \ifglsused\glslabel
10437      {%
```

Subsequent use

```
10438      \glstextaccessdisplay{\Glsentrytext{\glslabel}}{\glslabel}%
10439      \glsinsert
10440      }%
10441      {%
```

First use

```
10442      \glsfirstaccessdisplay{\Glsentryfirst{\glslabel}}{\glslabel}%
10443      \glsinsert
10444      }%
10445      }%
10446      {%
```

Make all upper case

```
10447      \ifglsused\glslabel
10448      {%
```

Subsequent use

```
10449      \glstextaccessdisplay
10450      {\mfirstucMakeUppercase{\glsentrytext{\glslabel}}}{\glslabel}%
10451      \mfirstucMakeUppercase{\glsinsert}%
10452      }%
10453      {%
```

First use

```
10454      \glsfirstaccessdisplay
10455      {\mfirstucMakeUppercase{\glsentryfirst{\glslabel}}}{\glslabel}%
10456      \mfirstucMakeUppercase{\glsinsert}%
10457      }%
10458      }%
10459      }%
10460      }%
10461      {%
```

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.

```
10462      \glscustomtext\glsinsert
10463      }%
10464      }
```

`\glsgenacfmt` Redefine to include accessibility information.

```
10465 \renewcommand*{\glsgenacfmt}{%
10466   \ifdefempty\glscustomtext
10467   {%
10468     \ifglused\glslabel
10469     {%
```

Subsequent use:

```
10470     \glsifplural
10471     {%
```

Subsequent plural form:

```
10472     \glscapscase
10473     {%
```

Subsequent plural form, don't adjust case:

```
10474     \acronymfont
10475     {\glsshortpluralaccessdisplay
10476      {\glentryshortpl{\glslabel}}{\glslabel}}%
10477     \glsinsert
10478   }%
10479   {%
```

Subsequent plural form, make first letter upper case:

```
10480     \acronymfont
10481     {\glsshortpluralaccessdisplay
10482      {\Glsentryshortpl{\glslabel}}{\glslabel}}%
10483     \glsinsert
10484   }%
10485   {%
```

Subsequent plural form, all caps:

```
10486     \mfirstucMakeUppercase
10487     {\acronymfont
10488      {\glsshortpluralaccessdisplay
10489       {\glentryshortpl{\glslabel}}{\glslabel}}%
10490      \glsinsert}%
10491   }%
10492   }%
10493   {%
```

Subsequent singular form

```
10494     \glscapscase
10495     {%
```

Subsequent singular form, don't adjust case:

```
10496     \acronymfont
10497     {\glsshortaccessdisplay{\glentryshort{\glslabel}}{\glslabel}}%
10498     \glsinsert
10499   }%
10500   {%
```

Subsequent singular form, make first letter upper case:

```
10501      \acronymfont
10502      {\glsshortaccessdisplay{\Glsentryshort{\glslabel}}{\glslabel}}%
10503      \glsinsert
10504      }%
10505      {%
```

Subsequent singular form, all caps:

```
10506      \mfirstucMakeUppercase
10507      {\acronymfont{%
10508      \glsshortaccessdisplay{\Glsentryshort{\glslabel}}{\glslabel}}%
10509      \glsinsert}%
10510      }%
10511      }%
10512      }%
10513      {%
```

First use:

```
10514      \glsifplural
10515      {%
```

First use plural form:

```
10516      \glscapscase
10517      {%
```

First use plural form, don't adjust case:

```
10518      \genplacrfullformat{\glslabel}{\glsinsert}%
10519      }%
10520      {%
```

First use plural form, make first letter upper case:

```
10521      \Genplacrfullformat{\glslabel}{\glsinsert}%
10522      }%
10523      {%
```

First use plural form, all caps:

```
10524      \mfirstucMakeUppercase
10525      {\genplacrfullformat{\glslabel}{\glsinsert}}%
10526      }%
10527      }%
10528      {%
```

First use singular form

```
10529      \glscapscase
10530      {%
```

First use singular form, don't adjust case:

```
10531      \genacrfullformat{\glslabel}{\glsinsert}%
10532      }%
10533      {%
```


First use singular form, make first letter upper case:

```
10534      \Genacrfullformat{\glslabel}{\glsinsert}%
10535      }%
10536      {%
```

First use singular form, all caps:

```
10537      \mfirstucMakeUppercase
10538      {\genacrfullformat{\glslabel}{\glsinsert}}%
10539      }%
10540      }%
10541      }%
10542      }%
10543      {%
```

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.

```
10544      \glscustomtext
10545      }%
10546 }
```

enacrfullformat Redefine to include accessibility information.

```
10547 \renewcommand*{\genacrfullformat}[2]{%
10548   \glslongaccessdisplay{\glsentrylong{#1}}{#1}#2\space
10549   (\glsshortaccessdisplay{\protect\firstacronymfont{\glsentryshort{#1}}}{#1})%
10550 }
```

enacrfullformat Redefine to include accessibility information.

```
10551 \renewcommand*{\Genacrfullformat}[2]{%
10552   \glslongaccessdisplay{\Glsentrylong{#1}}{#1}#2\space
10553   (\glsshortaccessdisplay{\protect\firstacronymfont{\Glsentryshort{#1}}}{#1})%
10554 }
```

placrfullformat Redefine to include accessibility information.

```
10555 \renewcommand*{\genplacrfullformat}[2]{%
10556   \glslongpluralaccessdisplay{\glsentrylongpl{#1}}{#1}#2\space
10557   (\glsshortpluralaccessdisplay
10558     {\protect\firstacronymfont{\glsentryshortpl{#1}}}{#1})%
10559 }
```

placrfullformat Redefine to include accessibility information.

```
10560 \renewcommand*{\Genplacrfullformat}[2]{%
10561   \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}#2\space
10562   (\glsshortpluralaccessdisplay
10563     {\protect\firstacronymfont{\Glsentryshortpl{#1}}}{#1})%
10564 }
```

\@acrshort

```
10565 \def\@acrshort#1#2[#3]{%
10566   \glsdoifexists{#2}%
```

```

10567 {%
10568   \let\do@gl@s@link@checkfirsthyper\relax

10569   \let\gl@sifplural\@secondoftwo
10570   \let\gl@scapscase\@firstofthree
10571   \let\gl@sinsert\@empty
10572   \def\glscustomtext{%
10573     \acronymfont{\glsshortaccessdisplay{\gl@sentryshort{#2}}{#2}}#3%
10574   }%

   Call \@gl@s@link
10575   \@gl@s@link[#1]{#2}{\csname gl@s\glstype @entryfmt\endcsname}%
10576   }%

10577   \glspostlinkhook
10578 }

```

\@Acrshort

```

10579 \def\@Acrshort#1#2[#3]{%
10580   \gl@sdoifexists{#2}%
10581   {%
10582     \let\do@gl@s@link@checkfirsthyper\relax

10583     \let\gl@sifplural\@secondoftwo
10584     \let\gl@scapscase\@secondofthree
10585     \let\gl@sinsert\@empty
10586     \def\glscustomtext{%
10587       \acronymfont{\glsshortaccessdisplay{\gl@sentryshort{#2}}{#2}}#3%
10588     }%

     Call \@gl@s@link
10589     \@gl@s@link[#1]{#2}{\csname gl@s\glstype @entryfmt\endcsname}%
10590     }%

10591     \glspostlinkhook
10592 }

```

\@ACRshort

```

10593 \def\@ACRshort#1#2[#3]{%
10594   \gl@sdoifexists{#2}%
10595   {%
10596     \let\do@gl@s@link@checkfirsthyper\relax

10597     \let\gl@sifplural\@secondoftwo
10598     \let\gl@scapscase\@thirdofthree
10599     \let\gl@sinsert\@empty
10600     \def\glscustomtext{%
10601       \acronymfont{\glsshortaccessdisplay
10602         {\MakeUppercase{\gl@sentryshort{#2}}}{#2}}#3%
10603     }%

```

```

Call \@gls@link
10604   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10605   }%

10606   \glspostlinkhook
10607 }

```

\@acrlong

```

10608 \def\@acrlong#1#2[#3]{%
10609   \glsdoifexists{#2}%
10610   {%
10611     \let\do@gls@link@checkfirsthyper\relax

10612     \let\glsifplural\@secondoftwo
10613     \let\glscapscase\@firstofthree
10614     \let\glsinsert\@empty
10615     \def\glscustomtext{%
10616       \acronymfont{\glslongaccessdisplay{\glsentrylong{#2}}{#2}}#3%
10617     }%

```

```

Call \@gls@link
10618   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10619   }%

10620   \glspostlinkhook
10621 }

```

\@Acrlong

```

10622 \def\@Acrlong#1#2[#3]{%
10623   \glsdoifexists{#2}%
10624   {%
10625     \let\do@gls@link@checkfirsthyper\relax

10626     \let\glsifplural\@secondoftwo
10627     \let\glscapscase\@firstofthree
10628     \let\glsinsert\@empty
10629     \def\glscustomtext{%
10630       \acronymfont{\glslongaccessdisplay{\Glsentrylong{#2}}{#2}}#3%
10631     }%

```

```

Call \@gls@link
10632   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10633   }%

10634   \glspostlinkhook
10635 }

```

\@ACRlong

```

10636 \def\@ACRlong#1#2[#3]{%
10637   \glsdoifexists{#2}%
10638   {%
10639     \let\do@gls@link@checkfirsthyper\relax

```

```

10640 \let\glsifplural\@secondoftwo
10641 \let\glsifscapscase\@firstofthree
10642 \let\glsinsert\@empty
10643 \def\glscustomtext{%
10644     \acronymfont{\glslongaccessdisplay{%
10645         \MakeUppercase{\glsentrylong{#2}}}{#2}#3}%
10646     }%

    Call \@gls@link
10647     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10648 }%

10649 \glspostlinkhook
10650 }

```

5.3 Displaying the Glossary

We need to redefine the way the glossary entries are formatted to include the accessibility support. The predefined glossary styles use `\glossentryname`, `\glossentrydesc` and `\glossentrysymbol`, but we need to provide compatibility with earlier versions in case users have defined their own styles using `\accsuppglossaryentryfield` and `\accsuppglossarysubentryfield`.

Now redefine `\glossentryname`, `\glossentrydesc` and `\glossentrysymbol` etc so they use the accessibility stuff.

```

10651 \renewcommand*{\glossentryname}[1]{%
10652     \glsdoifexists{#1}%
10653     {%
10654         \glsnamefont{\glsnameaccessdisplay{\glsentryname{#1}}{#1}}%
10655     }%
10656 }

10657 \renewcommand*{\glossentrydesc}[1]{%
10658     \glsdoifexists{#1}%
10659     {%
10660         \glsnamefont{\glsnameaccessdisplay{\Glsentryname{#1}}{#1}}%
10661     }%
10662 }

10663 \renewcommand*{\glossentrydesc}[1]{%
10664     \glsdoifexists{#1}%
10665     {%
10666         \glsdescriptionaccessdisplay{\glsentrydesc{#1}}{#1}%
10667     }%
10668 }

10669 \renewcommand*{\Glossentrydesc}[1]{%
10670     \glsdoifexists{#1}%
10671     {%
10672         \glsdescriptionaccessdisplay{\Glsentrydesc{#1}}{#1}%
10673     }%
10674 }

```

```

10675 \renewcommand*{\glossentrysymbol}[1]{%
10676   \glsdoifexists{#1}%
10677   {%
10678     \glssymbolaccessdisplay{\glentrysymbol{#1}}{#1}%
10679   }%
10680 }

10681 \renewcommand*{\Glossentrysymbol}[1]{%
10682   \glsdoifexists{#1}%
10683   {%
10684     \glssymbolaccessdisplay{\Glentrysymbol{#1}}{#1}%
10685   }%
10686 }

```

ssaryentryfield

```

10687 \newcommand*{\accsuppglossaryentryfield}[5]{%
10688   \glossaryentryfield{#1}%
10689   {\glsnameaccessdisplay{#2}{#1}}%
10690   {\glsdescriptionaccessdisplay{#3}{#1}}%
10691   {\glssymbolaccessdisplay{#4}{#1}}{#5}%
10692 }

```

rysubentryfield

```

10693 \newcommand*{\accsuppglossarysubentryfield}[6]{%
10694   \glossarysubentryfield{#1}{#2}%
10695   {\glsnameaccessdisplay{#3}{#2}}%
10696   {\glsdescriptionaccessdisplay{#4}{#2}}%
10697   {\glssymbolaccessdisplay{#5}{#2}}{#6}%
10698 }

```

5.4 Acronyms

Redefine acronym styles provided by glossaries:

long-short *<long>* (*<short>*) acronym style.

```

10699 \renewacronymstyle{long-short}%
10700 {%

```

Check for long form in case this is a mixed glossary.

```

10701   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
10702 }%
10703 {%
10704   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
10705   \renewcommand*{\genacrfullformat}[2]{%
10706     \glslongaccessdisplay{\glsentrylong{##1}}{##1}##2\space
10707     (\glsshortaccessdisplay
10708       {\protect\firstacronymfont{\glsentryshort{##1}}}{##1})%
10709   }%
10710   \renewcommand*{\Genacrfullformat}[2]{%

```

```

10711 \glslongaccessdisplay{\Glsentrylong{##1}}{##1}##2\space
10712 (\glsshortaccessdisplay
10713   {\protect\firstacronymfont{\glsentryshort{##1}}{##1}})%
10714 }%
10715 \renewcommand*{\genplacrfullformat}[2]{%
10716   \glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}##2\space
10717   (\glsshortpluralaccessdisplay
10718     {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}})%
10719   }%
10720 \renewcommand*{\Genplacrfullformat}[2]{%
10721   \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}##2\space
10722   (\glsshortpluralaccessdisplay
10723     {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}})%
10724   }%
10725 \renewcommand*{\acronymentry}[1]{%
10726   \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}{##1}}
10727 \renewcommand*{\acronymsort}[2]{##1}%
10728 \renewcommand*{\acronymfont}[1]{##1}%
10729 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
10730 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
10731 }

```

short-long (*short*) (*long*) acronym style.

```

10732 \renewacronymstyle{short-long}%
10733 {%

```

Check for long form in case this is a mixed glossary.

```

10734 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
10735 }%
10736 {%
10737 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
10738 \renewcommand*{\genacrfullformat}[2]{%
10739   \glsshortaccessdisplay
10740     {\protect\firstacronymfont{\glsentryshort{##1}}{##1}##2\space
10741     (\glslongaccessdisplay{\glsentrylong{##1}}{##1}})%
10742   }%
10743 \renewcommand*{\Genacrfullformat}[2]{%
10744   \glsshortaccessdisplay
10745     {\protect\firstacronymfont{\Glsentryshort{##1}}{##1}##2\space
10746     (\glslongaccessdisplay{\glsentrylong{##1}}{##1}})%
10747   }%
10748 \renewcommand*{\genplacrfullformat}[2]{%
10749   \glsshortpluralaccessdisplay
10750     {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}##2\space
10751     (\glslongpluralaccessdisplay
10752       {\glsentrylongpl{##1}}{##1}})%
10753   }%
10754 \renewcommand*{\Genplacrfullformat}[2]{%
10755   \glsshortpluralaccessdisplay
10756     {\protect\firstacronymfont{\Glsentryshortpl{##1}}{##1}##2\space

```

```

10757 (\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1})%
10758 }%
10759 \renewcommand*{\acronymentry}[1]{%
10760   \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}%
10761 \renewcommand*{\acronymsort}[2]{##1}%
10762 \renewcommand*{\acronymfont}[1]{##1}%
10763 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
10764 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
10765 }

```

long-short-desc *long* (*short*) acronym style that has an accompanying description (which the user needs to supply).

```

10766 \renewacronymstyle{long-short-desc}%
10767 {%
10768   \GlsUseAcrEntryDispStyle{long-short}%
10769 }%
10770 {%
10771   \GlsUseAcrStyleDefs{long-short}%
10772   \renewcommand*{\GenericAcronymFields}{}%
10773   \renewcommand*{\acronymsort}[2]{##2}%
10774   \renewcommand*{\acronymentry}[1]{%
10775     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10776     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10777 }

```

g-sc-short-desc *long* (\textsc{short}) acronym style that has an accompanying description (which the user needs to supply).

```

10778 \renewacronymstyle{long-sc-short-desc}%
10779 {%
10780   \GlsUseAcrEntryDispStyle{long-sc-short}%
10781 }%
10782 {%
10783   \GlsUseAcrStyleDefs{long-sc-short}%
10784   \renewcommand*{\GenericAcronymFields}{}%
10785   \renewcommand*{\acronymsort}[2]{##2}%
10786   \renewcommand*{\acronymentry}[1]{%
10787     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10788     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10789 }

```

g-sm-short-desc *long* (\textsmaller{short}) acronym style that has an accompanying description (which the user needs to supply).

```

10790 \renewacronymstyle{long-sm-short-desc}%
10791 {%
10792   \GlsUseAcrEntryDispStyle{long-sm-short}%
10793 }%
10794 {%
10795   \GlsUseAcrStyleDefs{long-sm-short}%
10796   \renewcommand*{\GenericAcronymFields}{}%

```

```

10797 \renewcommand*{\acronymsort}[2]{##2}%
10798 \renewcommand*{\acronymentry}[1]{%
10799     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10800     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10801 }

```

short-long-desc *<short>* (*<long>*) acronym style that has an accompanying description (which the user needs to supply).

```

10802 \renewacronymstyle{short-long-desc}%
10803 {%
10804     \GlsUseAcrEntryDispStyle{short-long}%
10805 }%
10806 {%
10807     \GlsUseAcrStyleDefs{short-long}%
10808     \renewcommand*{\GenericAcronymFields}{}%
10809     \renewcommand*{\acronymsort}[2]{##2}%
10810     \renewcommand*{\acronymentry}[1]{%
10811         \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10812         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10813 }

```

short-long-desc *<long>* (*\textsc{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

10814 \renewacronymstyle{sc-short-long-desc}%
10815 {%
10816     \GlsUseAcrEntryDispStyle{sc-short-long}%
10817 }%
10818 {%
10819     \GlsUseAcrStyleDefs{sc-short-long}%
10820     \renewcommand*{\GenericAcronymFields}{}%
10821     \renewcommand*{\acronymsort}[2]{##2}%
10822     \renewcommand*{\acronymentry}[1]{%
10823         \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10824         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10825 }

```

short-long-desc *<long>* (*\textsmaller{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

10826 \renewacronymstyle{sm-short-long-desc}%
10827 {%
10828     \GlsUseAcrEntryDispStyle{sm-short-long}%
10829 }%
10830 {%
10831     \GlsUseAcrStyleDefs{sm-short-long}%
10832     \renewcommand*{\GenericAcronymFields}{}%
10833     \renewcommand*{\acronymsort}[2]{##2}%
10834     \renewcommand*{\acronymentry}[1]{%
10835         \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10836         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%

```


10837 }

dua *<long>* only acronym style.

10838 \renewacronymstyle{dua}%
10839 {%

Check for long form in case this is a mixed glossary.

10840 \ifdefempty\glscustomtext
10841 {%
10842 \ifglshaslong{\glslabel}%
10843 {%
10844 \glsifplural
10845 {%

Plural form:

10846 \glscapscase
10847 {%

Plural form, don't adjust case:

10848 \glslongpluralaccessdisplay{\glsentrylongpl{\glslabel}}{\glslabel}%
10849 \glsinsert
10850 }%
10851 {%

Plural form, make first letter upper case:

10852 \glslongpluralaccessdisplay{\Glentrylongpl{\glslabel}}{\glslabel}%
10853 \glsinsert
10854 }%
10855 {%

Plural form, all caps:

10856 \glslongpluralaccessdisplay
10857 {\mfirstucMakeUppercase{\glsentrylongpl{\glslabel}}}{\glslabel}%
10858 \mfirstucMakeUppercase{\glsinsert}%
10859 }%
10860 }%
10861 {%

Singular form

10862 \glscapscase
10863 {%

Singular form, don't adjust case:

10864 \glslongaccessdisplay{\glsentrylong{\glslabel}}{\glslabel}\glsinsert
10865 }%
10866 {%

Subsequent singular form, make first letter upper case:

10867 \glslongaccessdisplay{\Glentrylong{\glslabel}}{\glslabel}\glsinsert
10868 }%
10869 {%

Subsequent singular form, all caps:

```

10870         \glslongaccessdisplay
10871         {\mfirstucMakeUppercase
10872          {\glsentrylong{\glslabel}\glsinsert}}{\glslabel}%
10873         \mfirstucMakeUppercase{\glsinsert}%
10874     }%
10875 }%
10876 }%
10877 {%

```

Not an acronym:

```

10878     \glsgenentryfmt
10879 }%
10880 }%
10881 {\glscustomtext\glsinsert}%
10882 }%
10883 {%
10884 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
10885 \renewcommand*{\acrfullfmt}[3]{%
10886     \glslink[##1]{##2}{%
10887         \glslongaccessdisplay{\glsentrylong{##2}}{##2}##3\space
10888         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}})%
10889 \renewcommand*{\Acrfullfmt}[3]{%
10890     \glslink[##1]{##2}{%
10891         \glslongaccessdisplay{\Glsentrylong{##2}}{##2}##3\space
10892         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}})%
10893 \renewcommand*{\ACRfullfmt}[3]{%
10894     \glslink[##1]{##2}{%
10895         \glslongaccessdisplay
10896         {\mfirstucMakeUppercase{\glsentrylong{##2}}{##2}##3\space
10897         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}})%
10898 \renewcommand*{\acrfullplfmt}[3]{%
10899     \glslink[##1]{##2}{%
10900         \glslongpluralaccessdisplay
10901         {\glsentrylongpl{##2}}{##2}##3\space
10902         (\glsshortpluralaccessdisplay
10903         {\acronymfont{\glsentryshortpl{##2}}}{##2}})%
10904 \renewcommand*{\Acrfullplfmt}[3]{%
10905     \glslink[##1]{##2}{%
10906         \glslongpluralaccessdisplay
10907         {\Glsentrylongpl{##2}}{##2}##3\space
10908         (\glsshortpluralaccessdisplay
10909         {\acronymfont{\glsentryshortpl{##2}}}{##2}})%
10910 \renewcommand*{\ACRfullplfmt}[3]{%
10911     \glslink[##1]{##2}{%
10912         \glslongpluralaccessdisplay
10913         {\mfirstucMakeUppercase{\glsentrylongpl{##2}}{##2}##3\space
10914         (\glsshortpluralaccessdisplay
10915         {\acronymfont{\glsentryshortpl{##2}}}{##2}})%
10916 \renewcommand*{\glsentryfull}[1]{%

```

```

10917 \glslongaccessdisplay{\glsentrylong{##1}}\space
10918 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})%
10919 }%
10920 \renewcommand*{\Glsentryfull}[1]{%
10921 \glslongaccessdisplay{\Glsentrylong{##1}}{##1}\space
10922 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})%
10923 }%
10924 \renewcommand*{\glsentryfullpl}[1]{%
10925 \glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}\space
10926 (\glsshortpluralaccessdisplay{\acronymfont{\glsentryshortpl{##1}}}{##1})%
10927 }%
10928 \renewcommand*{\Glsentryfullpl}[1]{%
10929 \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}\space
10930 (\glsshortpluralaccessdisplay{\acronymfont{\glsentryshortpl{##1}}}{##1})%
10931 }%
10932 \renewcommand*{\acronymentry}[1]{%
10933 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}%
10934 \renewcommand*{\acronymsort}[2]{##1}%
10935 \renewcommand*{\acronymfont}[1]{##1}%
10936 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
10937 }

```

dua-desc *<long>* only acronym style with user-supplied description.

```

10938 \renewacronymstyle{dua-desc}%
10939 {%
10940 \GlsUseAcrEntryDispStyle{dua}%
10941 }%
10942 {%
10943 \GlsUseAcrStyleDefs{dua}%
10944 \renewcommand*{\GenericAcronymFields}{}%
10945 \renewcommand*{\acronymentry}[1]{%
10946 \glslongaccessdisplay{\acronymfont{\glsentrylong{##1}}}{##1}}%
10947 \renewcommand*{\acronymsort}[2]{##2}%
10948 }%

```

footnote *<short>*\footnote{*<long>*} acronym style.

```

10949 \renewacronymstyle{footnote}%
10950 {%
10951 \ifglshaslong{\glslabel}{\glsacronymfont}{\glsacronymentryfmt}%
10952 }%
10953 {%
10954 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%

```

Need to ensure hyperlinks are switched off on first use:

```

10955 \glshyperfirstfalse
10956 \renewcommand*{\genacrfullformat}[2]{%
10957 \glsshortaccessdisplay
10958 {\protect\firstacronymfont{\glsentryshort{##1}}}{##1}##2%

```

```

10959 \protect\footnote{\glslongaccessdisplay{\glsentrylong{##1}}{##1}}%
10960 }%
10961 \renewcommand*{\Genacrfullformat}[2]{%
10962 \glsshortaccessdisplay
10963   {\firstacronymfont{\Glsentryshort{##1}}{##1}##2%
10964 \protect\footnote{\glslongaccessdisplay{\glsentrylong{##1}}{##1}}%
10965 }%
10966 \renewcommand*{\genplacrfullformat}[2]{%
10967 \glsshortpluralaccessdisplay
10968   {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}##2%
10969 \protect\footnote{\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}}%
10970 }%
10971 \renewcommand*{\Genplacrfullformat}[2]{%
10972 \glsshortpluralaccessdisplay
10973   {\protect\firstacronymfont{\Glsentryshortpl{##1}}{##1}##2%
10974 \protect\footnote{\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}}%
10975 }%
10976 \renewcommand*{\acronymentry}[1]{%
10977 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}{##1}}%
10978 \renewcommand*{\acronymsort}[2]{##1}%
10979 \renewcommand*{\acronymfont}[1]{##1}%
10980 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%

```

Don't use footnotes for \acrfull:

```

10981 \renewcommand*{\acrfullfmt}[3]{%
10982 \glslink{##1}{##2}{%
10983 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}{##2}##3\space
10984 (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}}%
10985 \renewcommand*{\Acrfullfmt}[3]{%
10986 \glslink{##1}{##2}{%
10987 \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##2}}{##2}##3\space
10988 (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}}%
10989 \renewcommand*{\ACRfullfmt}[3]{%
10990 \glslink{##1}{##2}{%
10991 \glsshortaccessdisplay
10992   {\mfirstucMakeUppercase
10993   {\acronymfont{\glsentryshort{##2}}{##2}##3\space
10994   (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}}%
10995 \renewcommand*{\acrfullplfmt}[3]{%
10996 \glslink{##1}{##2}{%
10997 \glsshortpluralaccessdisplay
10998   {\acronymfont{\glsentryshortpl{##2}}{##2}##3\space
10999   (\glslongpluralaccessdisplay{\glsentrylongpl{##2}}{##2})}}}%
11000 \renewcommand*{\ACRfullplfmt}[3]{%
11001 \glslink{##1}{##2}{%
11002 \glsshortpluralaccessdisplay
11003   {\acronymfont{\Glsentryshortpl{##2}}{##2}##3\space
11004   (\glslongpluralaccessdisplay{\glsentrylongpl{##2}}{##2})}}}%
11005 \renewcommand*{\ACRfullplfmt}[3]{%
11006 \glslink{##1}{##2}{%

```

```

11007      \glsshortpluralaccessdisplay
11008      {\mfirstucMakeUppercase
11009      {\acronymfont{\glentryshortpl{##2}}}{##2}##3\space
11010      (\glslongpluralaccessdisplay{\glentrylongpl{##2}}{##2}}}%

```

Similarly for \glentryfull etc:

```

11011 \renewcommand*{\glentryfull}[1]{%
11012     \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}}{##1}\space
11013     (\glslongaccessdisplay{\glentrylong{##1}}{##1}})%
11014 \renewcommand*{\Glsentryfull}[1]{%
11015     \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##1}}}{##1}\space
11016     (\glslongaccessdisplay{\glentrylong{##1}}{##1}})%
11017 \renewcommand*{\glentryfullpl}[1]{%
11018     \glsshortpluralaccessdisplay
11019     {\acronymfont{\glentryshortpl{##1}}}{##1}\space
11020     (\glslongpluralaccessdisplay{\glentrylongpl{##1}}{##1}})%
11021 \renewcommand*{\Glsentryfullpl}[1]{%
11022     \glsshortpluralaccessdisplay
11023     {\acronymfont{\Glsentryshortpl{##1}}}{##1}\space
11024     (\glslongpluralaccessdisplay{\glentrylongpl{##1}}{##1}})%
11025 }

```

footnote-sc \textsc{<short>}\footnote{<long>} acronym style.

```

11026 \renewacronymstyle{footnote-sc}%
11027 {%
11028     \GlsUseAcrEntryDispStyle{footnote}%
11029 }%
11030 {%
11031     \GlsUseAcrStyleDefs{footnote}%
11032     \renewcommand{\acronymentry}[1]{%
11033         \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}}{##1}}
11034     \renewcommand{\acronymfont}[1]{\textsc{##1}}%
11035     \renewcommand*{\acrpluralsuffix}{\glstextup{\glspluralsuffix}}%
11036 }%

```

footnote-sm \textsmaller{<short>}\footnote{<long>} acronym style.

```

11037 \renewacronymstyle{footnote-sm}%
11038 {%
11039     \GlsUseAcrEntryDispStyle{footnote}%
11040 }%
11041 {%
11042     \GlsUseAcrStyleDefs{footnote}%
11043     \renewcommand{\acronymentry}[1]{%
11044         \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}}{##1}}
11045     \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
11046     \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11047 }%

```

footnote-desc <short>\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

11048 \renewacronymstyle{footnote-desc}%
11049 {%
11050   \GlsUseAcrEntryDisplayStyle{footnote}%
11051 }%
11052 {%
11053   \GlsUseAcrStyleDefs{footnote}%
11054   \renewcommand*{\GenericAcronymFields}{}%
11055   \renewcommand*{\acronymsort}[2]{##2}%
11056   \renewcommand*{\acronymentry}[1]{%
11057     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11058     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11059 }

```

ootnote-sc-desc \textsc{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

11060 \renewacronymstyle{footnote-sc-desc}%
11061 {%
11062   \GlsUseAcrEntryDisplayStyle{footnote-sc}%
11063 }%
11064 {%
11065   \GlsUseAcrStyleDefs{footnote-sc}%
11066   \renewcommand*{\GenericAcronymFields}{}%
11067   \renewcommand*{\acronymsort}[2]{##2}%
11068   \renewcommand*{\acronymentry}[1]{%
11069     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11070     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11071 }

```

ootnote-sm-desc \textsmaller{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

11072 \renewacronymstyle{footnote-sm-desc}%
11073 {%
11074   \GlsUseAcrEntryDisplayStyle{footnote-sm}%
11075 }%
11076 {%
11077   \GlsUseAcrStyleDefs{footnote-sm}%
11078   \renewcommand*{\GenericAcronymFields}{}%
11079   \renewcommand*{\acronymsort}[2]{##2}%
11080   \renewcommand*{\acronymentry}[1]{%
11081     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11082     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11083 }

```

Use \newacronymhook to modify the key list to set the access text to the long version by default.

```

11084 \renewcommand*{\newacronymhook}{%
11085   \edef\@gls@keylist{shortaccess=\the\glslongtok,%
11086     \the\glskeylisttok}%
11087   \expandafter\glskeylisttok\expandafter{\@gls@keylist}%

```

11088 }

1tNewAcronymDef Modify default style to use access text:

```
11089 \renewcommand*{\DefaultNewAcronymDef}{%
11090   \edef\@do@newglossaryentry{%
11091     \noexpand\newglossaryentry{\the\glslabeltok}%
11092     {%
11093       type=\acronymtype,%
11094       name={\the\glsshorttok},%
11095       description={\the\glslongtok},%
11096       descriptionaccess=\relax,%
11097       text={\the\glsshorttok},%
11098       access={\noexpand\@glo@textaccess},%
11099       sort={\the\glsshorttok},%
11100       short={\the\glsshorttok},%
11101       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11102       shortaccess={\the\glslongtok},%
11103       long={\the\glslongtok},%
11104       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11105       descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11106       first={\noexpand\glslongaccessdisplay
11107         {\the\glslongtok}{\the\glslabeltok}\space
11108         (\noexpand\glsshortaccessdisplay
11109         {\the\glsshorttok}{\the\glslabeltok})},%
11110       plural={\the\glsshorttok\acrpluralsuffix},%
11111       firstplural={\noexpand\glslongpluralaccessdisplay
11112         {\noexpand\@glo@longpl}{\the\glslabeltok}\space
11113         (\noexpand\glsshortpluralaccessdisplay
11114         {\noexpand\@glo@shortpl}{\the\glslabeltok})},%
11115       firstaccess=\relax,%
11116       firstpluralaccess=\relax,%
11117       textaccess={\noexpand\@glo@shortaccess},%
11118       \the\glskeylisttok
11119     }%
11120   }%
11121   \let\@org@gls@assign@firstpl\gls@assign@firstpl
11122   \let\@org@gls@assign@plural\gls@assign@plural
11123   \let\@org@gls@assign@descplural\gls@assign@descplural
11124   \def\gls@assign@firstpl##1##2{%
11125     \@gls@expand@field{##1}{firstpl}{##2}%
11126   }%
11127   \def\gls@assign@plural##1##2{%
11128     \@gls@expand@field{##1}{plural}{##2}%
11129   }%
11130   \def\gls@assign@descplural##1##2{%
11131     \@gls@expand@field{##1}{descplural}{##2}%
11132   }%
11133   \@do@newglossaryentry
11134   \let\gls@assign@firstpl\@org@gls@assign@firstpl
```

```

11135 \let\gls@assign@plural\@org@gls@assign@plural
11136 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11137 }

```

teNewAcronymDef

```

11138 \renewcommand*{\DescriptionFootnoteNewAcronymDef}{%
11139 \edef\@do@newglossaryentry{%
11140 \noexpand\newglossaryentry{\the\glslabeltok}%
11141 {%
11142 type=\acronymtype,%
11143 name={\noexpand\acronymfont{\the\glsshorttok}},%
11144 sort={\the\glsshorttok},%
11145 text={\the\glsshorttok},%
11146 short={\the\glsshorttok},%
11147 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11148 shortaccess={\the\glslongtok},%
11149 long={\the\glslongtok},%
11150 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11151 access={\noexpand\@glo@textaccess},%
11152 plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11153 symbol={\the\glslongtok},%
11154 symbolplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11155 firstpluralaccess=\relax,
11156 textaccess={\noexpand\@glo@shortaccess},%
11157 \the\glskeylisttok
11158 }%
11159 }%
11160 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11161 \let\@org@gls@assign@plural\gls@assign@plural
11162 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
11163 \def\gls@assign@firstpl##1##2{%
11164 \@@gls@expand@field{##1}{firstpl}{##2}%
11165 }%
11166 \def\gls@assign@plural##1##2{%
11167 \@@gls@expand@field{##1}{plural}{##2}%
11168 }%
11169 \def\gls@assign@symbolplural##1##2{%
11170 \@@gls@expand@field{##1}{symbolplural}{##2}%
11171 }%
11172 \do@newglossaryentry
11173 \let\gls@assign@plural\@org@gls@assign@plural
11174 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11175 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11176 }

```

onNewAcronymDef

```

11177 \renewcommand*{\DescriptionNewAcronymDef}{%
11178 \edef\@do@newglossaryentry{%
11179 \noexpand\newglossaryentry{\the\glslabeltok}%

```



```

11180   {%
11181     type=\acronymtype,%
11182     name={\noexpand
11183       \acrnameformat{\the\glsshorttok}{\the\glslongtok}},%
11184     access={\noexpand\@glo@textaccess},%
11185     sort={\the\glsshorttok},%
11186     short={\the\glsshorttok},%
11187     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11188     shortaccess={\the\glslongtok},%
11189     long={\the\glslongtok},%
11190     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11191     first={\the\glslongtok},%
11192     firstaccess=\relax,
11193     firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11194     text={\the\glsshorttok},%
11195     textaccess={\the\glslongtok},%
11196     plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11197     symbol={\noexpand\@glo@text},%
11198     symbolaccess={\noexpand\@glo@textaccess},%
11199     symbolplural={\noexpand\@glo@plural},%
11200     firstpluralaccess=\relax,
11201     textaccess={\noexpand\@glo@shortaccess},%
11202     \the\glskeylisttok}%
11203   }%
11204   \let\@org@gls@assign@firstpl\gls@assign@firstpl
11205   \let\@org@gls@assign@plural\gls@assign@plural
11206   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
11207   \def\gls@assign@firstpl##1##2{%
11208     \@@gls@expand@field{##1}{firstpl}{##2}%
11209   }%
11210   \def\gls@assign@plural##1##2{%
11211     \@@gls@expand@field{##1}{plural}{##2}%
11212   }%
11213   \def\gls@assign@symbolplural##1##2{%
11214     \@@gls@expand@field{##1}{symbolplural}{##2}%
11215   }%
11216   \do@newglossaryentry
11217   \let\gls@assign@firstpl\@org@gls@assign@firstpl
11218   \let\gls@assign@plural\@org@gls@assign@plural
11219   \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11220 }

```

teNewAcronymDef

```

11221 \renewcommand*{\FootnoteNewAcronymDef}{%
11222   \edef\@do@newglossaryentry{%
11223     \noexpand\newglossaryentry{\the\glslabeltok}%
11224     {%
11225       type=\acronymtype,%
11226       name={\noexpand\acronymfont{\the\glsshorttok}},%

```

```

11227     sort={\the\glsshorttok},%
11228     text={\the\glsshorttok},%
11229     textaccess={\the\glslongtok},%
11230     access={\noexpand\@glo@textaccess},%
11231     plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11232     short={\the\glsshorttok},%
11233     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11234     long={\the\glslongtok},%
11235     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11236     description={\the\glslongtok},%
11237     descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11238     \the\glskeylisttok
11239   }%
11240 }%
11241 \let\@org@gls@assign@plural\gls@assign@plural
11242 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11243 \let\@org@gls@assign@descplural\gls@assign@descplural
11244 \def\gls@assign@firstpl##1##2{%
11245   \@@gls@expand@field{##1}{firstpl}{##2}%
11246 }%
11247 \def\gls@assign@plural##1##2{%
11248   \@@gls@expand@field{##1}{plural}{##2}%
11249 }%
11250 \def\gls@assign@descplural##1##2{%
11251   \@@gls@expand@field{##1}{descplural}{##2}%
11252 }%
11253 \do@newglossaryentry
11254 \let\gls@assign@plural\@org@gls@assign@plural
11255 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11256 \let\gls@assign@descplural\@org@gls@assign@descplural
11257 }

```

11NewAcronymDef

```

11258 \renewcommand*{\SmallNewAcronymDef}{%
11259   \edef\@do@newglossaryentry{%
11260     \noexpand\newglossaryentry{\the\glslabeltok}%
11261     {%
11262       type=\acronymtype,%
11263       name={\noexpand\acronymfont{\the\glsshorttok}},%
11264       access={\noexpand\@glo@symbolaccess},%
11265       sort={\the\glsshorttok},%
11266       short={\the\glsshorttok},%
11267       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11268       shortaccess={\the\glslongtok},%
11269       long={\the\glslongtok},%
11270       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11271       text={\noexpand\@glo@short},%
11272       textaccess={\noexpand\@glo@shortaccess},%
11273       plural={\noexpand\@glo@shortpl},%

```

```

11274     first={\the\glslongtok},%
11275     firstaccess=\relax,
11276     firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11277     description={\noexpand\@glo@first},%
11278     descriptionplural={\noexpand\@glo@firstplural},%
11279     symbol={\the\glsshorttok},%
11280     symbolaccess={\the\glslongtok},%
11281     symbolplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11282     \the\glskeylisttok
11283   }%
11284 }%
11285 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11286 \let\@org@gls@assign@plural\gls@assign@plural
11287 \let\@org@gls@assign@descplural\gls@assign@descplural
11288 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
11289 \def\gls@assign@firstpl##1##2{%
11290   \@@gls@expand@field{##1}{firstpl}{##2}%
11291 }%
11292 \def\gls@assign@plural##1##2{%
11293   \@@gls@expand@field{##1}{plural}{##2}%
11294 }%
11295 \def\gls@assign@descplural##1##2{%
11296   \@@gls@expand@field{##1}{descplural}{##2}%
11297 }%
11298 \def\gls@assign@symbolplural##1##2{%
11299   \@@gls@expand@field{##1}{symbolplural}{##2}%
11300 }%
11301 \do@newglossaryentry
11302 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11303 \let\gls@assign@plural\@org@gls@assign@plural
11304 \let\gls@assign@descplural\@org@gls@assign@descplural
11305 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11306 }

```

The following are kept for compatibility with versions before 3.0:

sshortaccesskey

```

11307   \newcommand*{\glsshortaccesskey}{\glsshortkey access}%

```

pluralaccesskey

```

11308   \newcommand*{\glsshortpluralaccesskey}{\glsshortpluralkey access}%

```

lslongaccesskey

```

11309   \newcommand*{\glslongaccesskey}{\glslongkey access}%

```

pluralaccesskey

```

11310   \newcommand*{\glslongpluralaccesskey}{\glslongpluralkey access}%

```

5.5 Debugging Commands

owglonameaccess

```
11311 \newcommand*{\showglonameaccess}[1]{%
11312   \expandafter\show\csname glo@\glsdetoklabel{#1}@textaccess\endcsname
11313 }
```

owglotextaccess

```
11314 \newcommand*{\showglotextaccess}[1]{%
11315   \expandafter\show\csname glo@\glsdetoklabel{#1}@textaccess\endcsname
11316 }
```

glopluralaccess

```
11317 \newcommand*{\showglopluralaccess}[1]{%
11318   \expandafter\show\csname glo@\glsdetoklabel{#1}@pluralaccess\endcsname
11319 }
```

wglofirstaccess

```
11320 \newcommand*{\showglofirstaccess}[1]{%
11321   \expandafter\show\csname glo@\glsdetoklabel{#1}@firstaccess\endcsname
11322 }
```

rstpluralaccess

```
11323 \newcommand*{\showglofirstpluralaccess}[1]{%
11324   \expandafter\show\csname glo@\glsdetoklabel{#1}@firstpluralaccess\endcsname
11325 }
```

glosymbolaccess

```
11326 \newcommand*{\showglosymbolaccess}[1]{%
11327   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolaccess\endcsname
11328 }
```

bolpluralaccess

```
11329 \newcommand*{\showglosymbolpluralaccess}[1]{%
11330   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolpluralaccess\endcsname
11331 }
```

owglodescaccess

```
11332 \newcommand*{\showglodescaccess}[1]{%
11333   \expandafter\show\csname glo@\glsdetoklabel{#1}@descaccess\endcsname
11334 }
```

escpluralaccess

```
11335 \newcommand*{\showglodescpluralaccess}[1]{%
11336   \expandafter\show\csname glo@\glsdetoklabel{#1}@descpluralaccess\endcsname
11337 }
```

wgloshortaccess

```
11338 \newcommand*{\showgloshortaccess}[1]{%
11339   \expandafter\show\csname glo@\glsdetoklabel{#1}@shortaccess\endcsname
11340 }
```

ortpluralaccess

```
11341 \newcommand*{\showgloshortpluralaccess}[1]{%
11342   \expandafter\show\csname glo@\glsdetoklabel{#1}@shortpluralaccess\endcsname
11343 }
```

owglolongaccess

```
11344 \newcommand*{\showglolongaccess}[1]{%
11345   \expandafter\show\csname glo@\glsdetoklabel{#1}@longaccess\endcsname
11346 }
```

ongpluralaccess

```
11347 \newcommand*{\showglolongpluralaccess}[1]{%
11348   \expandafter\show\csname glo@\glsdetoklabel{#1}@longpluralaccess\endcsname
11349 }
```

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.

```
11350 \NeedsTeXFormat{LaTeX2e}
11351 \ProvidesPackage{glossaries-babel}[2016/04/30 v4.23 (NLCT)]
```

Load tracklang to obtain language settings.

```
11352 \RequirePackage{tracklang}
11353 \let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
11354 \AnyTrackedLanguages
11355 {%
11356   \ForEachTrackedDialect{\this@dialect}{%
11357     \IfTrackedLanguageFileExists{\this@dialect}%
11358       {glossaries-}% prefix
11359       {.ldf}%
11360       {%
11361         \RequireGlossariesLang{\CurrentTrackedTag}%
11362       }%
11363       {%
11364         \PackageWarningNoLine{glossaries}%
11365           {No language module detected for ‘\this@dialect’.\MessageBreak
11366             Language modules need to be installed separately.\MessageBreak
11367             Please check on CTAN for a bundle called\MessageBreak
11368             ‘glossaries-\CurrentTrackedLanguage’ or similar}%
11369       }%
11370     }%
11371   }%
11372 }
```

6.1 Polyglossia Captions

Language support has now been split off into independent language modules.

```
11373 \NeedsTeXFormat{LaTeX2e}
11374 \ProvidesPackage{glossaries-polyglossia}[2016/04/30 v4.23 (NLCT)]
```

Load tracklang to obtain language settings.

```
11375 \RequirePackage{tracklang}
11376 \let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
11377 \AnyTrackedLanguages
```

```

11378 {%
11379     \ForEachTrackedDialect{\this@dialect}{%
11380         \IfTrackedLanguageFileExists{\this@dialect}%
11381         {glossaries-}% prefix
11382         {.ldf}%
11383         {%
11384             \RequireGlossariesLang{\CurrentTrackedTag}%
11385         }%
11386         {%
11387             \PackageWarningNoLine{glossaries}%
11388             {No language module detected for ‘\this@dialect’.\MessageBreak
11389             Language modules need to be installed separately.\MessageBreak
11390             Please check on CTAN for a bundle called\MessageBreak
11391             ‘glossaries-\CurrentTrackedLanguage’ or similar}%
11392         }%
11393     }%
11394 }%
11395 {}%

```

Glossary

`makeindex` An indexing application. [9](#), [24](#), [25](#), [168](#)

`xindy` An flexible indexing application with multilingual support written in Perl. [9](#), [24](#), [25](#), [168](#)

Change History

1.01 (2007-05-17)	numberline: numberline option added . . . 5
General: Added range facility in format key 107	1.12 (2008-03-08)
\writeist: Added spaces after \delimN and \delimR in ist file 154	\@GLSpl: now uses \glentrydescplural and \glentrysymbolplural instead of \glentrydesc and \glentrysymbol 121
1.04 (2007-08-03)	\@Glspl@: now uses \glentrydescplural and \glentrysymbolplural instead of \glentrydesc and \glentrysymbol 120
General: Added \glstextformat 91	\@Glspl@: now uses \glentrydescplural and \glentrysymbolplural instead of \glentrydesc and \glentrysymbol 119
1.05 (2007-08-10)	General: added check for \hypertarget separate to \hyperlink (memoir defines \hyperlink but not \hypertarget) 115
\glossarysection: added \@mkboth to \glossarysection 36	descriptionplural: new 59
\gls@defglossaryentry: Changed the default value of the sort key to just the value of the name key 76	\gls@defglossaryentry: Changed default first plural to be first key with s appended (was text key with s appended) 76
1.07 (2007-09-13)	descriptionplural support added 75
\@gls@link: fixed bug caused by \theglentrycounter setting the page number too soon 105	symbolplural support added 75
\glsadd: fixed bug caused by \theglentrycounter setting the page number too soon 151	\Glsentrydescplural: New 145
1.08 (2007-10-13)	\glentrydescplural: New 145
General: Added babel support 30	\Glsentrysymbolplural: New 146
listgroup: changed listgroup style to use \glsgrouptitle 261	\glentrysymbolplural: New 146
altlistgroup: changed altlistgroup style to use \glsgrouptitle 262	\SetDescriptionFootnoteAcronymStyle: Added \protect before \footnote and \glslink 228
1.1 (2008-02-22)	\SetFootnoteAcronymStyle: Added \protect before \footnote and \glslink 234
\@glossarysection: numbered sections and auto label added 37	symbolplural: new 60
\@gls@tmpb: changed \toksdef to \newtoks 109	1.13 (2008-05-10)
\@gls@toc: numberline added 39	General: fixed bug that ignored 3rd parameter 123–130
\@p@glossarysection: numbered sections and auto label added 38	\ACRfullpl: new 209
General: amsgen now loaded (\new@ifnextchar needed) 4	
translate: translate option added 21	
\setglossarysection: new 37	
numberedsection: numberedsection package option added 6	

\Acrfullpl: new	209	\@gls@: Test glossary type is \acronymtype	
\acrfullpl: new	208	in addition to checking if footnote op-	
\acrpluralsuffix: New	206	tion has been used	117
\gls@defglossaryentry: Changed de-		\@glsdisp: Test glossary type is	
fault first value	76	\acronymtype in addition to check-	
Changed default firstplural value	76	ing if footnote option has been used	122
Removed restriction on only using		\@glspl@: Test glossary type is	
\newglossaryentry in the preamble	81	\acronymtype in addition to check-	
\newacronym: Removed restriction on		ing if footnote option has been used	119
only using \newacronym in the pream-		\@glstarget: raised the hypertarget so	
ble	206	the target text doesn't scroll off the top	
1.14 (2008-06-17)		of the page	116
\@gls@hypergroup: new	257	\gls@defglossaryentry: Changed def	
General: added nonumberlist key to		to let	76
\printglossary	192	1.17 (2008-12-26)	
added numberedsection key to		\@@do@wrglossary: new	172
\printglossary	191	\@do@seeglossary: new	174
\firstacronymfont: new	210	\@glo@storeentry: new	82
\glsautoprefix: new	6	\@gls@glossary: changed definition to	
\glsnavhyperlink: changed \edef to		use \index instead of \@index	169
\protected@edef	256	\@glsdefaultplural: new	63
\glsnavhypertarget: added write to aux		\@glsdefaultsort: new	63
file	256	\@glshypernumber: new	203
\glsnavigation: changed to only use la-		\@glsnoname: new	63
bel for groups that are present	257	\@glsnonextpages: new	193
1.15 (2008-08-15)		General: added xindy support	24
\@gls@link: added \glslabel	105	parent: new	61
\gls@defglossaryentry: check for		see: new	61
\@glo@first in description	79	\gls@defglossaryentry: added non-	
check for \@glo@text in symbol	80	umberlist key	76
\gls@hypergroup: new	257	added parent key	76
\glsnavhypertarget: added check if re-		added see key	76
run required	256	Stored main part of entry format when	
\glssettoctitle: new	29	entry is defined	80
\printglossary: changed the way the		\gls@suffixF: new	34
TOC title is set	177	\gls@suffixFF: new	35
1.16 (2008-08-27)		\gls@wrglossary: modified to allow for	
\@GLS@: Test glossary type is \acronymtype		xindy support	169
in addition to checking if footnote op-		\glshyperlink: new	151
tion has been used	119	\glshypernumber: modified to allow ma-	
\@GLSpl: Test glossary type is		terial to be attached to location	203
\acronymtype in addition to check-		\glsnavhyperlink: replaced \hyperlink	
ing if footnote option has been used	121	to \@glslink	256
\@Gls@: Test glossary type is \acronymtype		\glsnavhypertarget: replaced	
in addition to checking if footnote op-		\hypertarget to \@glstarget ...	256
tion has been used	118	\glssee: new	175
\@Glspl@: Test glossary type is		\glsseeformat: new	175
\acronymtype in addition to check-		\glsSetSuffixF: new	34
ing if footnote option has been used	120	\glsSetSuffixFF: new	35

\ifglxsindy: new	24	before term is displayed to prevent unwanted whatsit	106
\listfilename: added xindy support ...	33		
\newglossarystyle: made \newglossarystyle long	202	\foralllglossaries: replaced \ifthenelse with \ifx	48
\nopostdesc: new	32	\forallgsentries: replaced \ifthenelse with \ifx	48
nonumberlist: new	61	\glsgdefmain: new	12
\printglossary: added check to determine if \printglossary is already defined	177	\glsgdescwidth: changed \linewidth to \hsize	264, 286
added print language to aux file	177	\glsglistdottedwidth: changed \linewidth to \hsize	263
order: order package option added	24	\glsgpagelistwidth: changed \linewidth to \hsize	264, 286
\writeist: added xindy support	154	nomain: added nomain package option .	12
1.18 (2009-01-14)		\writeist: removed item_02 - no such makeindex key	158
\@gls@loadlist: new	8	2.02 (2007-07-13)	
\@gls@loadlong: new	7	\@printglossary: suppressed warning globally rather than locally	179
\@gls@loadsuper: new	7	2.02 (2009-07-13)	
\@gls@loadtree: new	8	\glossarysection: changed \@mkboth to \glossarymark	36
\glsgdefglossaryentry: Changed default value of sort to \@glsgdefaultsort	76	\glsgglossarymark: New	36
moved sort sanitization to \newglossaryentry	80	2.03 (2009-09-23)	
\glstarget: new	196	\@GLS@: Added check for hyperfirst	119
\oldacronym: new	205	\@GLSp1: Added check for hyperfirst ...	121
nolist: new	8	\@Gls@: Added check for hyperfirst	118
nolong: new	7	\@Glspl@: Added check for hyperfirst ..	120
sort: moved sanitization to \newglossaryentry	59	\@gls@: Added check for hyperfirst	117
nostyles: new	8	\@gls@link: new	104
nosuper: new	8	\@gls@link: added \leavevmode	105
notree: new	8	Moved entry existence check to avoid duplicate code	105
1.19 (2009-03-02)		\@glsgdisp: Added check for hyperfirst .	122
\glsclearpage: new	39	\@glsgpl@: Added check for hyperfirst ..	119
\glsgdisp: new	121	\glsgglossarymark: Added check to see if it's already defined	36
\SetDescriptionAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	232	hyperfirst: new	23
\SetDescriptionFootnoteAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	228	2.04 (2009-11-10)	
\SetFootnoteAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	234	\@GLS@: Changed test to check if glossary type has been identified as a list of acronyms	119
\SetSmallAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	237	\@GLSp1: Changed test to check if glossary type has been identified as a list of acronyms	121
2.01 (2009 May 30)		\@Gls@: Changed test to check if glossary type has been identified as a list of acronyms	118
\@gls@link: moved \@do@wrglossary			

\@Glspl@: Changed test to check if glossary type has been identified as a list of acronyms	120	\SetDUADisplayStyle: new	238
\@glossaryentryfield: new	81	\SetFootnoteAcronymDisplayStyle: new	233
\@glossarysubentryfield: new	81	\SetSmallAcronymDisplayStyle: new	235
\@gls@: Changed test to check if glossary type has been identified as a list of acronyms	117	2.05 (2010-02-06)	
\@glsacronymlists: new	13	\@glsdisp: Added closing brace. Patch provided by Sergiu Dotenco	122
\@glsdisp: Changed test to check if glossary type has been identified as a list of acronyms	122	Removed spurious brace. Patch provided by Sergiu Dotenco	122
\@glspl@: Changed test to check if glossary type has been identified as a list of acronyms	119	\writeist: Added \string before opening and closing braces. Patch provided by Segiu Dotenco	159
\@newglossaryentryposthook: new ..	81	2.06 (2010-06-14)	
\@newglossaryentryprehook: new ..	81	\altnewglossary: new	57
acronymlists: new	15	\CustomAcronymFields: new	240
\DeclareAcronymList: new	14	\CustomNewAcronymDef: new	240
\DefineAcronymSynonyms: new	223	\SetCustomDisplayStyle: new	240
\gls@defglossaryentry: added userl-6 keys	76	\SetCustomStyle: new	241
\glsadd: fixed bug that ignored counter	151	2.07 (2010-07-10)	
\Glsentryuseri: new	147	General: glsadd format key stored in \@glsnumberformat (was mistakenly stored in \@glo@format)	151
\Glsentryuseri: new	147	3.0 (2010-07-12)	
\Glsentryuserii: new	147	\@makeglossary: Added check for savewrites	160
\Glsentryuserii: new	147	\gls@wrglossary: modified to take into account savewrites	169
\Glsentryuseriii: new	147	3.0 (2010/03/31)	
\Glsentryuseriii: new	147	\@set@glo@numformat: added 4th argument	107
\Glsentryuseriv: new	148	3.0 (2011-04-02)	
\Glsentryuseriv: new	148	\@do@wrglossary: added check for hyper location prefix	172
\Glsentryuserv: new	148	modified to use new format	172
\Glsentryuserv: new	148	\@glossarysec: replaced \@ifundefined with \@ifcsundef	5
\Glsentryuservi: new	148	\@do@seeglossary: Sanitize and escape cross-referencing information	174
\ns@newglossary: added check to determine if \gls@{type}@display and \gls@{type}@displayfirst have been defined.	56	\@gls@counterwithin: new	9
\SetAcronymLists: new	15	\@gls@ifinlist: new	40
\SetDefaultAcronymDisplayStyle: new	225	\@gls@link: added \@gls@saveentrycounter	106
\SetDefaultAcronymStyle: new	226	added \@gls@setsort	106
\SetDescriptionAcronymDisplayStyle: new	230	\@gls@saveentrycounter: new	106
\SetDescriptionDUAAcronymDisplayStyle: new	229	\@gls@setupsort@def: new	10
\SetDescriptionFootnoteAcronymDisplayStyle: new	226	\@gls@setupsort@standard: new	10
		\@gls@setupsort@use: new	11
		\@gls@xdy@locationlist: new	43

\@glslink: replaced \@ifundefined with \ifcsundef	115	\glsentryshort: new	148
\@glsnextpages: new	193	\Glsentryshortpl: new	149
\@print@glossary: replaced \@ifundefined with \ifcsundef	180	\glsentryshortpl: new	148
\@printglossary: added \currentglossary	178	\glsgetgrouptitle: replaced \@ifundefined with \ifcsundef	200
added \glsnextpages	179	\gls glossarymark: replaced \@ifundefined with \ifcsundef	36
make toctitle default to title	178	\gls hyperlink: changed default from \glsentryname to \glsentrytext	151
\@xdyattributelist: new	39	\gls hypernumber: replaced \@ifundefined with \ifcsundef	203
General: added prefix to hyperlink	204	\gls numberformat: replaced \@ifundefined with \ifcsundef	35
etoolbox now loaded	4	\gls refentry: new	195
replaced \@ifundefined with \ifcsundef	28, 31, 102, 190	\gls resetsubentrycounter: new ...	194
\acrfootnote: new	226	\gls seeitem: hyperlink uses \glsseeitemformat instead of \glsentryname	176
\ACRfull: added starred version	208	\gls seeitemformat: new	176
\Acrfull: added starred version	207	\gls sortnumberfmt: new	10
\acrfull: added starred version	207	\gls stepentry: new	194
\ACRfullpl: added starred version ...	209	\gls steps subentry: new	195
\Acrfullpl: added starred version ...	209	\gls subentrycounterlabel: new ...	195
\acrfullpl: added starred version ...	208	\gls subentryitem: new	196
\acrlinkfootnote: new	226	theglossary: replaced \@ifundefined with \ifcsundef	196
\acrno linkfootnote: new	226	short: new	62
save writes: new	25	shortplural: new	62
see: added \@glo@seeautonumberlist	61	\if glossaryexists: replaced \@ifundefined with \ifcsundef	49
seeautonumberlist: new	7	\if glsentryexists: replaced \@ifundefined with \ifcsundef	49
\glossarysection: replaced \@ifundefined with \ifcsundef	36	\istfile: deprecated	168
\glossarystyle: replaced \@ifundefined with \ifcsundef	201	glossaryentry: new	194
\gls@codepage: replaced \@ifundefined with \ifcsundef	24	glossarysubentry: new	194
\gls@def glossaryentry: added \@gls@def sort	80	\new glossaryentry: replaced \DeclareRobustCommand with \newrobustcmd	65
added short and long keys	76	\new glossarystyle: replaced \@ifundefined with \ifcsundef	202
replaced \@ifundefined with \ifcsundef	77	\ns@new glossary: added \@gls@def sort count	57
\gls@doc clearpage: replaced \@ifundefined with \ifcsundef	38	replaced \@ifundefined with \ifcsundef	56
\gls add: added \@gls@saveentrycounter	152	entrycounter: new	9
\GlsAddXdyCounters: new	40	entrycounterwithin: new	9
\glsentrycounterlabel: new	195	\oldacronym: replaced \@ifundefined with \ifcsundef	205
\glsentryitem: new	195	compatible-2.07: compatible-2.07 op- tion added	26
\Glsentrylong: new	149	long: new	62
\glsentrylong: new	149		
\Glsentrylongpl: new	149		
\glsentrylongpl: new	149		
\Glsentryshort: new	148		

longplural: new	62	\Acrfull: made robust	207
nonumberlist: now boolean	61	\acrfull: made robust	207
sort: new	9	\acrfullformat: removed \acronymfont	
counter: replaced \@ifundefined with		as it should already be set in the sec-	
\ifcsundef	60	ond argument.	207
\printglossary: replaced \@ifundefined		\ACRfullpl: made robust	209
with \ifcsundef	177	\Acrfullpl: made robust	209
\SetDescriptionFootnoteAcronymDisplayStyle		\acrfullpl: made robust	208
expanded options link options	226	\ACRlong: made robust	140
\setentrycounter: added optional ar-		\Acrlong: made robust	139
gument	201	\acrlong: made robust	138
\showacronymlists: new	246	\ACRlongpl: made robust	142
\showglocounter: new	243	\Acrlongpl: made robust	141
\showglodesc: new	244	\acrlongpl: made robust	140
\showglodescplural: new	245	\ACRshort: made robust	136
\showglofirst: new	243	\Acrshort: made robust	135
\showglofirsttpl: new	243	\acrshort: made robust	135
\showgloflag: new	246	\ACRshortpl: made robust	138
\showgloindex: new	246	\Acrshortpl: made robust	137
\showglolevel: new	242	\acrshortpl: made robust	137
\showglongame: new	244	\Gls: made robust	117
\showgloparent: new	242	\glsadd: made robust	151
\showgloplural: new	242	\glsaddall: made robust	152
\showglosort: new	245	\GLSdesc: made robust	127
\showglossaries: new	246	\Glsdesc: made robust	127
\showglossarycounter: new	247	\glsdesc: made robust	127
\showglossaryentries: new	247	\GLSdescplural: made robust	128
\showglossaryin: new	247	\Glsdescplural: made robust	128
\showglossaryout: new	247	\glsdescplural: made robust	127
\showglossarytitle: new	247	\glsfirst: made robust	123
\showglosymbol: new	245	\GLSfirstplural: made robust	126
\showglosymbolplural: new	245	\Glsfirstplural: made robust	125
\showglotext: new	242	\glsfirstplural: made robust	125
\showglotype: new	243	\glslink: made robust	104
\showglouseri: new	243	\GLSname: made robust	126
\showglouserii: new	243	\Glsname: made robust	126
\showglouseriii: new	244	\glsname: made robust	126
\showglouseriv: new	244	\GLSpl: made robust	121
\showglouserv: new	244	\Glspl: made robust	120
\showglouservi: new	244	\glspl: made robust	119
subentrycounter: new	9	\GLSplural: made robust	125
\writeist: added xindy-only macro defi-		\GLSsymbol: made robust	129
nitions to glossary open tag	156	\Glsymbol: made robust	129
modified to support new format	154	\glssymbol: made robust	128
3.01 (2011-04-12)		\GLSsymbolplural: made robust	130
\@glswritefiles: added check for		\Glsymbolplural: made robust	129
empty glossaries	168	\glssymbolplural: made robust	129
General: made robust	118	\Glstext: made robust	123
\ACRfull: made robust	208	\glstext: made robust	123

\GLSuseri: made robust	130	\glstentryfullpl: fixed bug (re-	
\Glsuseri: made robust	130	placed \glstentryshort with	
\glsuseri: made robust	130	\glstentryshortpl)	149
\GLSuserii: made robust	131	\glstentrynumberlist: new	150
\Glsuserii: made robust	131	\glsmoveentry: new	81
\glsuserii: made robust	131	\glsresetsubentrycounter: new ...	194
\GLSuseriii: made robust	132	\ifglshaschildren: new	51
\Glsuseriii: made robust	132	\ifglshasparent: new	51
\glsuseriii: made robust	132	\makeglossaries: added list parser ..	163
\GLSuseriv: made robust	133	indexonlyfirst: new	23
\Glsuseriv: made robust	133	\renewglossarystyle: new	202
\glsuseriv: made robust	132	\showglossaryentries: fixed misspelt	
\GLSuserv: made robust	134	command	247
\Glsuserv: made robust	133	\SmallNewAcronymDef: fixed broken	
\glsuserv: made robust	133	short and long plural	236
\GLSuservi: made robust	135	3.03 (2012/09/21)	
\Glsuservi: made robust	134	\@gls@sanitizesort: new	17
\glsuservi: made robust	134	\@gls@setupsort@standard: used	
		\@gls@sanitizesort	10
3.02 (2012-05-19)		\@printglossary: allow title to override	
\glsnumlistlastsep: new	151	default toctitle	178
\glsnumlistsep: new	151	General: allow title to set toctitle	190
3.02 (2012-05-21)		\glsinlinedescformat: new	260
\@do@wrglossary: changed \@gls@loc		\glsinlineemptydescformat: new ..	260
ref to \theglstentrycounter	173	\glsinlinenameformat: new	260
\@do@wrglossary: changed \@do@wr@glossary		\glsinlinepostchild: new	260
to test for indexonlyfirst option; put		\glsinlinesubdescformat: new	260
old \@do@wr@glossary code into		\glsinlinesubnameformat: new	260
\@do@wrglossary	170	\glspostinline: replaced “.” with	
\@gls@missingnumberlist: new	63	\glspostdescription	260
\@gls@writefiles: added check		altlongragged4col: added check for	
for existence of token in case		glsnogroupskip	279
\makeglossaries has been		altsuperragged4col: added check for	
omitted	168	glsnogroupskip	297
\@printglossary: add a way to fetch		alttree: added check for glsnogroupskip	
current entry label	179	306
savenumberlist: new	7	index: added check for glsnogroupskip	300
ucmark: new	8	nogroupskip: new	8
\gls@defglossaryentry: added num-		long: added check for glsnogroupskip	265
berlist element	79	long3col: added check for glsnogroup-	
\gls@save@numberlist: new	176	skip	266
\gls@wrglossary: added check for glos-		long4col: added check for glsnogroup-	
sary file defined	170	skip	268
\glsdisplaynumberlist: new	150	longragged: added check for glsnogroup-	
\glstentrycounter: set default value ..	106	skip	275
\Glstentryfull: fixed bug (re-		longragged3col: added check for	
placed \glstentryshortpl with		glsnogroupskip	277
\glstentryshort)	149	nopostdot: new	8
		tree: added check for glsnogroupskip	301

treenoname: added check for glsnogroup-skip	303	mcolindexspannav: replaced '2' with \glsmcols	281
super: added check for glsnogroupskip	287	mcoltree: replaced '2' with \glsmcols	282
super3col: added check for glsnogroup-skip	288	mcoltreenoname: replaced '2' with \glsmcols	283
super4col: added check for glsnogroup-skip	290	mcoltreespannav: replaced '2' with \glsmcols	283
superragged: added check for glsnogroupskip	293	\gls@protected@pagefmts: added Roman to list	170
superragged3col: added check for glsnogroupskip	295	\gls@Romanpage: new	171
3.04 (2012-11-11)		\glsgetgrouplabel: fixed bug (typo in \equal)	200
altlist: replaced \newline with paragraph break	262	\nopostdesc: made robust	32
3.04 (2012-11-18)		3.05 (2013/04/21)	
\@do@wrglossary: changed \theglsentrycount back to \glslocref	173	\@gls@nohyperlist: new	15
\@do@wrglossary: modified to compensate for possible incorrect page number	172	\@gls@DeclareNoHyperList: new	15
\@gls@escbsdq: unsanitize \gls@numberpage, \gls@alphpage, \gls@Alphpage and \gls@romanpage	108	nohypertypes: new	15
\@print@glossary: Moved aux write to end of document to prevent unwanted whatsit occurring here.	180	3.06 (2013/06/17)	
General: Added check for doc package ...	4	\@xdy@main@language: Changed back to using \language	24
added datatool-base as a required package	4	\findrootlanguage: Obsoleted	46
added local key	102	3.07 (2013-07-05)	
\gls@Alphpage: new	170	\@gls@link: fixed bug that failed to find entry in list	105
\gls@alphpage: new	170	\glossarypreamble: modified to work with \setglossarypreamble	35
\gls@disablepagerefexpansion: new	170	\gls@docclearpage: added check for openright	38
\gls@numberpage: new	171	\glspostdescription: Added spacefactor code	8
\gls@protected@pagefmts: new	170	\GlsSetXdyCodePage: Added check for fontspec	47
\gls@romanpage: new	171	\SetDescriptionAcronymDisplayStyle: now using \glsdoparenifnotempty	230
\glsdefmain: added check for doc package	12	\setglossarypreamble: new	36
\glsorg@endtheglossary: new	5	3.08a (2013-08-30)	
\glsorg@theglossary: new	5	list: updated list style to use \glossentry and \subglossentry	261
\PrintChanges: new	5	listdotted: updated listdotted style to use \glossentry and \subglossentry	263
3.05 (2013-04-21)		altlist: updated altlist style to use \glossentry and \subglossentry	262
\@do@wrglossary: add Roman case. Fixed bugs in the else statements ..	172	inline: updated inline style to use \glossentry and \subglossentry	259
\@gls@link: added check for "nohypertypes"	105	3.08a (2013-09-28)	
mcolalttree: replaced '2' with \glsmcols	284	\@gls@storeentry: no longer need to check for special characters in any of the fields other than sort	82
mcolindex: replaced '2' with \glsmcols	280		

updated for \glossentry	82	tree: updated to use \glossentry and	
\@glossaryentryfield: switched to		\subglossentry	301
\glossentry	81	\setglossarystyle:new	201
\@glossarysubentryfield: switched to		\setglossentrycompatibility:new	198
\subglossentry	81	superragged: updated to use	
General: added nogroupskip key to		\glossentry and \subglossentry	293
\printglossary	191	3.09a (2013-10-09)	
removed definition of \@glossaryentryfield		\@gls@assign@symbolplural@field:	
.....	348	new	17
removed definition of \@glossarysubentryfield		\@gls@default@value: new	60
.....	348	\Glsentrydesc: made robust	144
\compatibleglossentry: new	196	\Glsentrydescplural: made robust ..	145
\compatiblesubglossentry: new ...	198	\Glsentryfirst: made robust	146
\glossaryentryfield: deprecated ...	198	\Glsentryfirstplural: made robust ..	146
\Glossentrydesc: new	197	\Glsentryfull: made robust	149
\glossentrydesc: new	197	\Glsentryfullpl: made robust	149
\Glossentryname: new	197	\Glsentrylong: made robust	149
\glossentryname: new	197	\Glsentrylongpl: made robust	149
\Glossentrysymbol: new	198	\Glsentryname: made robust	143
\glossentrysymbol: new	197	\Glsentryplural: made robust	145
\gls@assign@desc@field: new	17	\Glsentryshort: made robust	148
\gls@assign@descplural@field: new	17	\Glsentryshortpl: made robust	149
\gls@assign@field: new	65	\Glsentrysymbol: made robust	145
\gls@ifnotmeasuring: new	83	\Glsentrysymbolplural: made robust	146
\glsaddallunused: new	152	\Glsentrytext: made robust	145
\glsexpandfields: new	65	\Glsentryuseri: made robust	147
\glsnoexpandfields: new	65	\Glsentryuserii: made robust	147
\glssee: made robust	175	\Glsentryuseriii: made robust	147
\glsseeformat: made robust	175	\Glsentryuseriv: made robust	148
\glsseeitem: made robust	176	\Glsentryuserv: made robust	148
\glsseelist: made robust	175	\Glsentryuservi: made robust	148
\ifglshdescsuppressed: new	52	\glstextup: new	206
\ifglshasdesc: new	52	\ifglshassymbol: changed test to check	
\ifglshassymbol: new	52	for \@gls@default@symbol	52
altlongragged4col: updated to use		3.10a (2013-09-28)	
\glossentry and \subglossentry	278	\gls@assign@type@field: new	17
alttree: updated to use \glossentry		3.10a (2013-10-13)	
and \subglossentry	304	\@gls@keymap: new	67
index: added paragraph break at end of		\@gls@provide@newglossary: new ...	55
environment	299	\@gls@writedef: new	66
updated to use \glossentry and		\@gls@defaultplural: Obsolete	63
\subglossentry	299	\@glsnodesc: new	63
long: updated to use \glossentry and		\@print@glossary: Added providecom-	
\subglossentry	264	mand code to aux file	180
longragged: updated to use \glossentry		\gls@defglossaryentry: Changed to	
and \subglossentry	275	using \@gls@default@value	76
longragged3col: updated to use		new	75
\glossentry and \subglossentry	277	\glswritedefhook: new	74

\makeglossaries: Added providecom-	\@acrshort: added \glslabel,
mand code to aux file 162	\glsifplural, \glscapscase,
\new@glossaryentry: new 66	\glsinsert and \glscustomtext 346
\ns@newglossary: added \@gls@provide@newglossary	\@gls@: add \glslabel, \glsifplural,
..... 56	\glscapscase, \glscustomtext and
3.11a (2013-10-15)	\glsinsert 117
\@ACRlong: added \glslabel,	change to using \glentryfmt style
\glsifplural, \glscapscase,	commands 117
\glsinsert and \glscustomtext 348	\@gls@noexpand@fields: Fixed bug
\@ACRshort: added \glslabel,	expand replaced with noexpand 64
\glsifplural, \glscapscase,	\@glsdisp: add \glslabel, \glsifplural,
\glsinsert and \glscustomtext 346	\glscapscase, \glscustomtext and
\@Acrlong: added \glslabel,	\glsinsert 122
\glsifplural, \glscapscase,	change to using \glentryfmt style
\glsinsert and \glscustomtext 347	commands 122
\@Acrshort: added \glslabel,	\@glspl@: add \glslabel, \glsifplural,
\glsifplural, \glscapscase,	\glscapscase, \glscustomtext and
\glsinsert and \glscustomtext 346	\glsinsert 119
\@GLS@: add \glslabel, \glsifplural,	change to using \glentryfmt style
\glscapscase, \glscustomtext and	commands 119
\glsinsert 118	General: added \glslabel, \glsifplural,
change to using \glentryfmt style	\glscapscase, \glsinsert and
commands 119	\glscustomtext 135-142
removed \MakeUppercase (now moved	changed to just use \Glsentrydescplural
to \glentryfmt) 119 128
\@GLSpl: add \glslabel, \glsifplural,	changed to just use \glentrydescplural
\glscapscase, \glscustomtext and 128
\glsinsert 121	changed to just use \Glsentrydesc . 127
change to using \glentryfmt style	changed to just use \glentrydesc . 127
commands 121	changed to just use \Glsentryfirstplural
removed \MakeUppercase as now dealt 125
with in \glentryfmt 121	changed to just use \glentryfirstplural
\@Gls@: add \glsifplural, \glscapscase, 125, 126
\glscustomtext and \glsinsert 118	changed to just use \Glsentryfirst 124
change to using \glentryfmt style	changed to just use \glentryfirst 124
commands 118	changed to just use \Glsentryname . 126
removed \makefirstuc (now dealt	changed to just use \glentryname .
with in \glentryfmt) 118 126, 127
\@Glspl@: add \glsifplural,	changed to just use \Glsentryplural 125
\glscapscase, \glscustomtext and	changed to just use \glentryplural
\glsinsert 120 124, 125
change to using \glentryfmt style	changed to just use \Glsentrysymbolplural
commands 120 130
removed \makefirstuc (now dealt	changed to just use \glentrysymbolplural
with in \glentryfmt) 120 129, 130
\@acrlong: added \glslabel,	changed to just use \Glsentrysymbol 129
\glsifplural, \glscapscase,	changed to just use \glentrysymbol
\glsinsert and \glscustomtext 347 128, 129
	Changed to just use \Glsentrytext . 123

changed to just use \glentrytext .	123	\ns@newglossary: replaced \glsdisplay	
changed to just use \Glsentryuseriii		and \glsdisplayfirst with	
.....	132	\glentryfmt	56
changed to just use \glentryuseriii		compatible-3.07: cnew	26
.....	132	\SetCustomDisplayStyle: updated to	
changed to just use \Glsentryuserii	131	use \defglentryfmt	240
changed to just use \glentryuserii	131	\SetDefaultAcronymDisplayStyle:	
changed to just use \Glsentryuseriv	133	changed to use \defglentryfmt .	225
changed to just use \glentryuseriv	133	\SetDescriptionAcronymDisplayStyle:	
changed to just use \Glsentryuseri	130	updated to use \defglentryfmt .	230
changed to just use \glentryuseri		\SetDescriptionDUAAcronymDisplayStyle:	
.....	130, 131	updated to use \defglentryfmt .	229
changed to just use \Glsentryuservi	134	\SetDescriptionFootnoteAcronymDisplayStyle:	
changed to just use \glentryuservi		updated to use \defglentryfmt .	226
.....	134, 135	\SetDUADisplayStyle: updated to use	
changed to just use \Glsentryuserv	134	\defglentryfmt	238
changed to just use \glentryuserv		\SetFootnoteAcronymDisplayStyle:	
.....	133, 134	updated to use \defglentryfmt .	233
Now requires textcase	4	\SetSmallAcronymDisplayStyle: up-	
acronymlists: replaced \@addtoacronymlists		dated to use \defglentryfmt ...	235
with \DeclareAcronymList	15	\setupglossaries: new	27
\defgldisplay: obsoleted	101	\showglolong: new	245
\defgldisplayfirst: obsoleted ...	101	\showgloshort: new	245
\defglentryfmt: new	55	numbers: new	26
\forglentries: replaced \ifx with		symbols: new	26
\ifdefempty	48	3.12a (2013-10-16)	
\gls@assign@desc: new	74	\gls@defglossaryentry: added	
\gls@defglossaryentry: Fixed default		\glslabel	75
counter if none supplied	79	\glsaddkey: new	69
\gls@doentryfmt: new	55	3.13a (2013-11-05)	
\glsdisplay: obsoleted	101	\@gls@assign@symbol@field: changed	
\glsdisplayfirst: obsoleted	100	to use \glssetnoexpandfield ...	17
\glsgenentryfmt: new	95	\@gls@assign@symbolplural@field:	
\glsgetgrouptitle: Added check in		changed to use \glssetnoexpandfield	
case non-Latin alphabet in use ...	200	17
\glsglossarymark: replaced \MakeUppercase		\@gls@link: removed \relax	106
with \mfirstucMakeUppercase ...	36	\@gls@notranslatorhook: new	21
\glsnavigation: switched to using		\@gls@setupsort@standard: moved	
\@gls@getgrouptitle	258	\@gls@santizesort to \glsprestandardsort	
\ifglshasdesc: replaced \ifdefempty		10
with \ifcempty	52	ucmark: added check for memoir	9
\ifglshaslong: new	52	see: added \gls@checkseeallowed ...	61
\ifglshasshort: new	52	\glossarysection: changed \glossarymark	
\ifglshassymbol: replaced \ifdefempty		to \glsglossarymark	36
with \ifcempty	52	\glossarystyle: fixed bug caused by us-	
\ifglused: replaced \ifthenelse with		ing \ifdef instead of \ifcsdef ...	202
\ifbool	50	\gls@assign@desc@field: changed to	
\longnewglossaryentry: new	74	use \glssetnoexpandfield	17

<code>\gls@assign@descplural@field:</code> changed to use <code>\glssetnoexpandfield</code> 17	<code>super3colheader:</code> switched to <code>\tabularnewline</code> 289
<code>\gls@assign@name@field:</code> changed to use <code>\glssetnoexpandfield</code> 17	<code>super4col:</code> switched to <code>\tabularnewline</code> 290
<code>\gls@assign@type@field:</code> changed to use <code>\glssetexpandfield</code> 17	<code>super4colheader:</code> switched to <code>\tabularnewline</code> 290
<code>\gls@checkseeallowed:</code> new 61	<code>super4colheaderborder:</code> switched to <code>\tabularnewline</code> 291
<code>\glsaddallunused:</code> set default to <code>\@glo@types</code> 152	<code>superheader:</code> switched to <code>\tabularnewline</code> 287
<code>\Glsentryfull:</code> changed to use <code>\acrfullformat</code> 149	<code>superheaderborder:</code> switched to <code>\tabularnewline</code> 287
<code>\glsentryfull:</code> changed to use <code>\acrfullformat</code> 149	3.14a (2013-11-12)
<code>\Glsentryfullpl:</code> changed to use <code>\acrfullformat</code> 149	<code>\@glswritefiles:</code> renamed <code>\glswritefiles</code> to <code>\@glswritefiles</code> and used “savewrites” option to set <code>\glswritefiles</code> 168
<code>\glsentryfullpl:</code> changed to use <code>\acrfullformat</code> 149	General: new 249
<code>\gls glossarymark:</code> renamed <code>\glossarymark</code> to <code>\gls glossarymark</code> to avoid con- flict with memoir 36	acronyms: new 13
<code>\glsprestandardsort:</code> new 9	<code>\gls@def glossaryentry:</code> added check for existence of default glossary 76
<code>\glssetexpandfield:</code> new 16	set the default for firstplural to be the value of plural 78
<code>\glssetnoexpandfield:</code> new 16	<code>xindy gloss:</code> new 25
<code>altsuper4colheader:</code> switched to <code>\tabularnewline</code> 291	<code>\long provide glossaryentry:</code> new ... 75
<code>altsuper4colheaderborder:</code> switched to <code>\tabularnewline</code> 292	compatible-2.07: added check for 2.07 before setting 3.07 compatibility 26
<code>long:</code> switched to <code>\tabularnewline</code> 264, 265	<code>notranslate:</code> new 21
<code>long3col:</code> switched to <code>\tabularnewline</code> 266	<code>\provide glossaryentry:</code> new 66
<code>long3colheader:</code> switched to <code>\tabularnewline</code> 267	4.0 (2013-11-14)
<code>long3colheaderborder:</code> switched to <code>\tabularnewline</code> 267	<code>\gls@def glossaryentry:</code> added check for first key 78
<code>long4col:</code> switched to <code>\tabularnewline</code> 267	<code>super:</code> fixed typo in <code>\sub glossentry</code> (<code>\glossentrydesc</code>) 286
<code>long4colheader:</code> switched to <code>\tabularnewline</code> 268	4.01 (2013-11-16)
<code>longheader:</code> switched to <code>\tabularnewline</code> 265	General: fixed non-value options so that they can be passed to document class . 6
<code>longheaderborder:</code> switched to <code>\tabularnewline</code> 265	<code>\CustomAcronymFields:</code> inserted miss- ing comma 240
<code>\SetFootnoteAcronymDisplayStyle:</code> fixed missing argument bug 233	4.02 (2013-12-05)
<code>super:</code> switched to <code>\tabularnewline</code> . 286	<code>\@acrfull:</code> now using <code>\acrfullfmt</code> .. 207
<code>super3col:</code> switched to <code>\tabularnewline</code> 288	<code>\@gls@indexdef:</code> new 27
	<code>\@gls@numbersdef:</code> new 26
	<code>\@gls@symbolsdef:</code> new 26
	General: Removed <code>\acronymfont</code> . 139–142
	<code>\ACRfullfmt:</code> new 208
	<code>\Acrfullfmt:</code> new 208
	<code>\acrfullfmt:</code> new 207
	<code>\ACRfullplfmt:</code> new 209

\Acrfullplfmt: new	209	\SetDescriptionFootnoteAcronymDisplayStyle:	
\acrfullplfmt: new	209	Moved check for empty custom text to	
\acronymentry: new	211	prevent unwanted parenthetical ma-	
sanitize: fixed bug that caused an error		terial	226
here	20	\SetFootnoteAcronymDisplayStyle:	
sc-short-long: new	215	Moved check for empty custom text to	
sc-short-long-desc: new	217	prevent unwanted parenthetical ma-	
\Genacrfullformat: new	100	terial	233
\genacrfullformat: new	100	\SetGenericNewAcronym: new	210
\GenericAcronymFields: new	211	\SetSmallAcronymDisplayStyle:	
\Genplacrfullformat: new	100	Moved check for empty custom text to	
\genplacrfullformat: new	100	prevent unwanted parenthetical ma-	
\Glsentryfull: bug fix: added missing		terial	235
\acronymfont	149	dua: new	218
\glentryfull: bug fix: added missing		dua-desc: new	220
\acronymfont	149	numberedsection: added nameref op-	
\Glsentryfullpl: bug fix: added miss-		tion	6
ing \acronymfont	149	4.02 (2013-13-05)	
\glentryfullpl: bug fix: added miss-		\makeglossaries: made preamble only	163
ing \acronymfont	149	4.03 (2014-01-17)	
\glsgenacfmt: new	98	General: changed default to \@empty in-	
\GlsUseAcrEntryDispStyle: new ...	213	stead of \relax	26
\GlsUseAcrStyleDefs: new	213	4.03 (2014-01-20)	
short-long: new	214	\@do@wrglossary: added \glsdetoklabel	
short-long-desc: new	217	173
xindynoglsnumbers: new	25	\@ACRlong: removed \glslabel (defined	
sm-short-long: new	216	in \@gls@link)	348
sm-short-long-desc: new	218	\@ACRshort: removed \glslabel (de-	
index: new	27	fined in \@gls@link)	346
\newacronymstyle: new	212	\@Acrlong: removed \glslabel (defined	
long-sc-short: new	215	in \@gls@link)	347
long-sc-short-desc: new	216	\@Acrshort: removed \glslabel (de-	
long-short: new	213	fined in \@gls@link)	346
long-short-desc: new	216	\@GLS@: removed \glslabel (defined in	
long-sm-short: new	215	\@gls@link)	118
long-sm-short-desc: new	217	\@GLSpl: removed \glslabel (defined in	
long-sp-short-desc: new	216	\@gls@link)	121
footnote: new	220	\@Gls@: removed \glslabel (defined in	
footnote-desc: new	222	\@gls@link)	118
footnote-sc: new	221	\@Gls@entry@field: new	143
footnote-sc-desc: new	222	\@Glspl@: removed \glslabel (defined	
footnote-sm: new	222	in \@gls@link)	120
footnote-sm-desc: new	223	\@acrlong: removed \glslabel (defined	
\setacronymstyle: new	212	in \@gls@link)	347
\SetDescriptionAcronymDisplayStyle:		\@acrshort: removed \glslabel (de-	
Moved check for empty custom text to		fined in \@gls@link)	346
prevent unwanted parenthetical ma-		\@gls@: removed \glslabel (defined in	
terial	230	\@gls@link)	117
		\@gls@access@display: new	334

<code>\@gls@entry@field</code> : new	142	<code>\glsentrydescpluralaccess</code> : switched to using <code>\@gls@entry@field</code>	333
<code>\@gls@fetchfield</code> : new	68	<code>\glsentryfirstaccess</code> : switched to using <code>\@gls@entry@field</code>	333
<code>\@gls@field@link</code> : new	122	<code>\glsentryfirstplural</code> : added <code>\glsdetoklabel</code>	146
<code>\@gls@link</code> : added <code>\glsdetoklabel</code> .	105	<code>\glsentrylongaccess</code> : switched to using <code>\@gls@entry@field</code>	334
moved <code>\@gls@link@opts</code> and		<code>\glsentrylongpluralaccess</code> : switched to using <code>\@gls@entry@field</code>	334
<code>\@gls@link@label</code> to <code>\@gls@link</code>	105	<code>\glsentrypluralaccess</code> : switched to using <code>\@gls@entry@field</code>	333
<code>\@gls@writedef</code> : added <code>\glsdetoklabel</code>	66	<code>\glsentryshortaccess</code> : switched to using <code>\@gls@entry@field</code>	333
<code>\@glsdisp</code> : removed <code>\glslabel</code> (defined in <code>\@gls@link</code>)	122	<code>\glsentryshortpluralaccess</code> : switched to using <code>\@gls@entry@field</code>	333
<code>\@glspl@</code> : removed <code>\glslabel</code> (defined in <code>\@gls@link</code>)	119	<code>\glsentrysymbolaccess</code> : switched to using <code>\@gls@entry@field</code>	333
<code>\@printglossary</code> : added <code>\glsdetoklabel</code>	179	<code>\glsentrysymbolpluralaccess</code> : switched to using <code>\@gls@entry@field</code>	333
General: removed <code>\glslabel</code> (defined in <code>\@gls@link</code>)	135	<code>\glsentrysymbolpluralaccess</code> : switched to using <code>\@gls@entry@field</code>	333
sc-short-long-desc: redefined to use accessibility information	352	<code>\glsentrytextaccess</code> : switched to using <code>\@gls@entry@field</code>	332
<code>\compatibleglossentry</code> : added <code>\glsdetoklabel</code>	328	<code>\glsgenacfmt</code> : redefined to use accessibility information	343
<code>\compatiblesubglossentry</code> : added <code>\glsdetoklabel</code>	329	<code>\glsgenentryfmt</code> : redefined to use accessibility information	340
<code>\Genacrfullformat</code> : redefined to use accessibility information	345	<code>\glshyperlink</code> : added <code>\glsdetoklabel</code>	151
<code>\genacrfullformat</code> : redefined to use accessibility information	345	<code>\glslocalreset</code> : added <code>\glsdetoklabel</code>	84
<code>\Genplacrfullformat</code> : redefined to use accessibility information	345	<code>\glslocalunset</code> : added <code>\glsdetoklabel</code>	84
<code>\genplacrfullformat</code> : redefined to use accessibility information	345	<code>\glsmoveentry</code> : added <code>\glsdetoklabel</code>	81
<code>\glossentryname</code> : added <code>\glsdetoklabel</code>	197	replaced <code>\ifthenelse</code> with <code>\ifdefequal</code>	81
<code>\gls@defglossaryentry</code> : added <code>\glsdetoklabel</code>	75	<code>\glsrefentry</code> : added <code>\glsdetoklabel</code>	195
replaced #1 with <code>\@glo@label</code>	76	<code>\glsreset</code> : added <code>\glsdetoklabel</code> ...	83
replaced <code>\ifthenelse</code> with <code>\ifdefequal</code>	77	<code>\glsseelist</code> : added <code>\expandafter</code> commands	176
<code>\glsadd</code> : added <code>\glsdetoklabel</code>	151	<code>\glsstepentry</code> : added <code>\glsdetoklabel</code>	194
<code>\glsaddkey</code> : switched to using <code>\@gls@field@link</code>	70	<code>\glsstepsubentry</code> : added <code>\glsdetoklabel</code>	195
<code>\glsdetoklabel</code> : new	49	<code>\glsunset</code> : added <code>\glsdetoklabel</code> ...	84
<code>\glsdisplaynumberlist</code> : added <code>\glsdetoklabel</code>	150	short-long: commented spurious EOL	215
<code>\glsdoifexistsorwarn</code> : new	50	redefined to use accessibility information	350
<code>\glsentryaccess</code> : switched to using <code>\@gls@entry@field</code>	332		
<code>\glsentrydescaccess</code> : switched to using <code>\@gls@entry@field</code>	333		

short-long-desc: redefined to use accessibility information	352	\showglodescaccess:added\glsdetoklabel	364
\ifglsdscsuppressed: added		\showglodescplural:added\glsdetoklabel	245
\glsdetoklabel	52	\showglodescpluralaccess: added	
fixed typo	52	\glsdetoklabel	364
\ifglstryexists:added\glsdetoklabel	49	\showglofirst:added\glsdetoklabel	243
\ifglshaschildren:added\glsdetoklabel	51	\showglofirstaccess:added\glsdetoklabel	364
\ifglshasdesc:added\glsdetoklabel	52	\showglofirsttpl:added\glsdetoklabel	243
\ifglshasfield: new	53	\showglofirstpluralaccess: added	
\ifglshaslong:added\glsdetoklabel	52	\glsdetoklabel	364
\ifglshasparent:added\glsdetoklabel	51	\showgloflag:added\glsdetoklabel	246
\ifglshasshort:added\glsdetoklabel	52	\showgloindex:added\glsdetoklabel	246
\ifglshassymbol:added\glsdetoklabel	52	\showglolevel:added\glsdetoklabel	242
replaced\ifcsempy with\ifdefempty and replaced\ifx with\ifdefequal	52	\showglolevel:added\glsdetoklabel	245
\ifglused:added\glsdetoklabel ..	50	\showglolevelaccess:added\glsdetoklabel	365
sm-short-long-desc: redefined to use accessibility information	352	\showglolevelpluralaccess: added	
long-sc-short-desc: redefined to use accessibility information	351	\glsdetoklabel	365
long-short: redefined to use accessibility information	349	\showglolevel:added\glsdetoklabel	244
long-short-desc: redefined to use accessibility information	351	\showglolevelaccess:added\glsdetoklabel	364
long-sm-short-desc: redefined to use accessibility information	351	\showgloparent:added\glsdetoklabel	242
footnote: redefined to use accessibility information	355	\showgloparent:added\glsdetoklabel	242
footnote-desc: redefined to use accessibility information	357	\showgloparentaccess:added\glsdetoklabel	245
footnote-sc: redefined to use accessibility information	357	\showgloparentpluralaccess: added	
footnote-sc-desc: redefined to use accessibility information	358	\glsdetoklabel	365
footnote-sm: redefined to use accessibility information	357	\showgloparentplural:added\glsdetoklabel	242
footnote-sm-desc: redefined to use accessibility information	358	\showgloparentpluralaccess: added	
\renewacronymstyle: new	212	\glsdetoklabel	364
\showglocounter:added\glsdetoklabel	243	\showgloparentplural:added\glsdetoklabel	245
\showglodesc:added\glsdetoklabel	244	\showgloparentpluralaccess: added	
		\glsdetoklabel	364
		\showgloparentplural:added	
		\glsdetoklabel	245
		\showgloparentpluralaccess: added	
		\glsdetoklabel	364
		\showgloparenttext:added\glsdetoklabel	242

\showglotextaccess:added\glsdetoklabel	\@gls@noidx@do:new	187
..... 364	\@gls@noref@warn:new	167
\showglotype:added\glsdetoklabel	\@gls@reference:new	189
243	\@gls@warnonglossdefined:new	16
\showglouserii:added\glsdetoklabel	\@gls@warnontheGLOSSdefined:new	16
..... 243	\@no@makeGLOSSaries:new	167
\showglouseriii:added\glsdetoklabel	\@print@GLOSSary:new	179
..... 243	\@print@noidx@GLOSSary:new	186
\showglouseriiii:added\glsdetoklabel	\@printGLOSS@setsort:new	177
..... 244	\@printGLOSSary:new	178
\showglouseriv:added\glsdetoklabel	General: added sort key to printGLOSS	
..... 244	group	193
\showglouserv:added\glsdetoklabel	\compatibleGLOSSentry: changed	
..... 244	\newcommand to \def as is may or may	
\showglouservi:added\glsdetoklabel	not be defined	328
..... 244	\compatibleSUBGLOSSentry: changed	
dua: fixed bug in \acrfullfmt	\newcommand to \def as is may or may	
..... 219	not be defined	329
fixed bug in \Acrrfullplfmt	\defGLSdisplayfirst: fixed unwanted	
..... 219	space	101
fixed bug in \acrfullplfmt	\glo@grabfirst:new	187
..... 219	\gls@defGLOSSaryentry: replaced \ifx	
redefined to use accessibility informa-	with \ifdefvoid	80
tion	\glsnoidxdisplayloc:new	189
..... 353	\glsnoidxdisplayloc:clisthandler:	
dua-desc: commented spurious EOL	new	189
.. 220	\glsnoidxloc:clist:new	188
redefined to use accessibility informa-	\glsnoidxloc:clisthandler:new	189
tion	\glsnoidxstripaccents:new	18
..... 355	alttree: moved hangindent and	
4.04 (2014-03-04)	parindent assignments outside level	
\@gls@getcounterprefix: added warn-	test	304
ing if no prefix can be formed	\makeGLOSSaries: Moved definition of	
..... 174	\glswrite to \makeGLOSSaries	162
4.04 (2014-03-06)	\makenoidxGLOSSaries:new	164
\@@gls@noidx@nosanitizesort:new	\printGLOSSary: changed to use new	
.. 18	\@printGLOSSary	177
\@@gls@noidx@sanitizesort:new	\printnoidxGLOSSaries:new	177
... 18	\printnoidxGLOSSary:new	177
\@@gls@nosanitizesort:new	\showgloloclist:new	246
..... 18	\warn@noprintGLOSSary: Activate warn-	
\@@gls@sanitizesort:new	ing in \makeGLOSSaries	176
..... 17	\writeist: checked for definition of	
\@glo@addchildren:new	\glswrite	154, 158
..... 181	4.06 (2014-03-12)	
\@glo@do@sortentries:new	\@GLS@: added \glsifhyper	119
..... 182	\@GLSpl: added \glsifhyper	121
\@glo@grabfirst:new	\@Gls@: added \glsifhyper	118
..... 187	\@GLspl: added \glsifhyper	120
\@glo@sortedinsert:new		
..... 183		
\@glo@sortentries:new		
..... 181		
\@glo@sorthandler@case:new		
..... 183		
\@glo@sorthandler@letter:new		
... 183		
\@glo@sorthandler@nocase:new		
... 183		
\@glo@sorthandler@word:new		
..... 183		
\@glo@sortmacro@case:new		
..... 185		
\@glo@sortmacro@def:new		
..... 185		
\@glo@sortmacro@def@do:new		
..... 186		
\@glo@sortmacro@letter:new		
..... 184		
\@glo@sortmacro@nocase:new		
..... 185		
\@glo@sortmacro@standard:new		
... 184		
\@glo@sortmacro@use:new		
..... 186		
\@glo@sortmacro@word:new		
..... 184		
\@gls@getothergrouptitle:new		
... 200		

\@gls@: added \glsifhyper	117	\@gls@field@link: added assignment of	
\@gls@numbersdef: added hook to set		\do@gls@link@checkfirsthyper	122
toc title	27	\@gls@forbidtexext: new	55
\@gls@symbolsdef: added hook to set		\@gls@hyp@opt: new	103
toc title	26	\@gls@link: removed redundancy	105
\@glsdisp: added \glsifhyper	122	renamed \gls@type to \glstype ...	105
\@glspl@: added \glsifhyper	119	\@glsdisp: moved \glsifhyper	122
General: added \glsifhyper	135–142	moved check for first use to	
acronym: added hook to set toc title	13	\@gls@link	122
acronyms: added hook to set toc title ...	13	\@glspl@: moved \glsifhyper	119
\glsdefmain: added hook to set toc title	12	moved check for first use to	
4.07 (2014-04-04)		\@gls@link	119
\@glossarysection: added optional ar-		\@ignored@glossaries: new	58
gument when using unstarred version	37	General: added entrycounter option to	
\@gls@noidx@do: added \global in case		printgloss family	191
it's used in a tabular-like style	187	added nopostdot option to printgloss	
\Acrfullplfmt: fixed no case change		family	191
bug	209	added subentrycounter option to	
\glsletentryfield: new	143	printgloss family	192
4.08 (2014-07-30)		explicitly initialise hyper key	102
\@ACRlong: added \do@gls@link@checkfirsthyper		moved \glsifhyper	135–142
.....	347	removed \@sACRlongpl	142
\@ACRshort: added \do@gls@link@checkfirsthyper		removed \@sAcrlongpl	141
.....	346	removed \@sacrlongpl	140
\@Acrlong: added \do@gls@link@checkfirsthyper		removed \@sACRlong	140
.....	347	removed \@sAcrlong	139
\@Acrshort: added \do@gls@link@checkfirsthyper		removed \@sacrlong	138
.....	346	removed \@sACRshortpl	138
\@GLS@: moved \glsifhyper	119	removed \@sAcrshortpl	137
moved check for first use to		removed \@sacrshortpl	137
\@gls@link	119	removed \@sACRshort	136
\@GLSpl: moved \glsifhyper	121	removed \@sAcrshort	135
moved check for first use to		removed \@sacrshort	135
\@gls@link	121	removed \@sgls@link	104
\@Gls@: moved \glsifhyper	118	removed \@sGLSdescplural	128
moved check for first use to		removed \@sGlsdescplural	128
\@gls@link	118	removed \@sglsdescplural	128
\@Glspl@: moved \glsifhyper	120	removed \@sGLSdesc	127
moved check for first use to		removed \@sGlsdesc	127
\@gls@link	120	removed \@sglsdesc	127
\@acrlong: added \do@gls@link@checkfirsthyper		removed \@sglsdisp	122
.....	347	removed \@sGLSfirstplural	126
\@acrshort: added \do@gls@link@checkfirsthyper		removed \@sGlsfirstplural	125
.....	345	removed \@sglsfirstplural	125
\@closegls: new	160	removed \@sGLSfirst	124
\@gls@: moved \glsifhyper	117	removed \@sGlsfirst	124
moved check for first use to		removed \@sglsfirst	123
\@gls@link	117	removed \@sGLSname	126
\@gls@doautomake: new	25	removed \@sGlsname	126

removed \@sglsname	126	\acrfull: removed \@sacrfull	207
removed \@sGLSplural	125	switched to using \@gls@hyp@opt ..	207
removed \@sGlsplural	125	\ACRfullpl: removed \s@ACRfullpl ..	209
removed \@sglsplural	124	switched to using \@gls@hyp@opt ..	209
removed \@sGLSpl	121	\Acrfullpl: removed \s@Acrfullpl ..	209
removed \@sGlspl	120	switched to using \@gls@hyp@opt ..	209
removed \@sglspl	119	\acrfullpl: removed \s@acrfullpl ..	208
removed \@sGLSsymbolplural	130	switched to using \@gls@hyp@opt ..	208
removed \@sGlsymbolplural	129	\ACRlong: switched to using \@gls@hyp@opt	
removed \@sglsymbolplural	129	140
removed \@sGLSsymbol	129	\Acrlong: switched to using \@gls@hyp@opt	
removed \@sGlsymbol	129	139
removed \@sglsymbol	128	\acrlong: switched to using \@gls@hyp@opt	
removed \@sGLStext	123	138
removed \@sGlstext	123	\ACRlongpl: switched to using	
removed \@sglstext	123	\@gls@hyp@opt	142
removed \@sGLSuseriii	132	\Acrlongpl: switched to using	
removed \@sGlsuseriii	132	\@gls@hyp@opt	141
removed \@sglsuseriii	132	\acrlongpl: switched to using	
removed \@sGLSuserii	131	\@gls@hyp@opt	140
removed \@sGlsuserii	131	\ACRshort: switched to using	
removed \@sglsuserii	131	\@gls@hyp@opt	136
removed \@sGLSuseriv	133	\Acrshort: switched to using	
removed \@sGlsuseriv	133	\@gls@hyp@opt	135
removed \@sglsuseriv	132	\acrshort: switched to using	
removed \@sGLSuseri	131	\@gls@hyp@opt	135
removed \@sGlsuseri	130	\ACRshortpl: switched to using	
removed \@sglsuseri	130	\@gls@hyp@opt	138
removed \@sGLSuservi	135	\Acrshortpl: switched to using	
removed \@sGlsuservi	134	\@gls@hyp@opt	137
removed \@sglsuservi	134	\acrshortpl: switched to using	
removed \@sGLSuserv	134	\@gls@hyp@opt	137
removed \@sGlsuserv	134	\forallacronyms: new	48
removed \@sglsuserv	133	\GLS: switched to using \@gls@hyp@opt	118
removed \@sGLS	118	\Gls: switched to using \@gls@hyp@opt	117
removed \@sGls	117	\gls: switched to using \@gls@hyp@opt	116
removed \@sgls	117	\gls@defglossaryentry: added check	
removed \@thirdofthree (defined in		for ignored glossary	77
kernel)	116	\gls@istfilebase: new	33
removed sPGLS	254	\glsaddkey: removed \@sGLS@user@<key>	
removed sPgls	252	71
removed spgls	251	removed \@sGls@user@<key>	71
removed sPGLSpl	254	removed \@sgls@user@<key>	70
removed sPglspl	253	switched to using \@gls@hyp@opt	70, 71
removed spglspl	252	\GLSdesc: switched to using \@gls@hyp@opt	
\ACRfull: removed \s@ACRfull	208	127
switched to using \@gls@hyp@opt ..	208	\Glsdesc: switched to using \@gls@hyp@opt	
\Acrfull: removed \s@Acrfull	207	127
switched to using \@gls@hyp@opt ..	207		

\glsdesc: switched to using \@gls@hyp@opt	127	\glsplural: switched to using \@gls@hyp@opt	124
\GLSdescplural: switched to using \@gls@hyp@opt	128	\glsspace: new	207
\Glsdescplural: switched to using \@gls@hyp@opt	128	\GLSsymbol: switched to using \@gls@hyp@opt	129
\glsdescplural: switched to using \@gls@hyp@opt	127	\Glsymbol: switched to using \@gls@hyp@opt	129
\glsdisablehyper: added \KV@glslink@hyperfalse to definition	116	\glssymbol: switched to using \@gls@hyp@opt	128
\glsdisp: switched to using \@gls@hyp@opt	121	\GLSsymbolplural: switched to using \@gls@hyp@opt	130
\glsdohyperlink: new	115	\Glsymbolplural: switched to using \@gls@hyp@opt	129
\glsdohypertarget: new	115	\glssymbolplural: switched to using \@gls@hyp@opt	129
\glsenablehyper: added \KV@glslink@hypertrue to definition	116	\GLStext: switched to using \@gls@hyp@opt	123
\GLSfirst: switched to using \@gls@hyp@opt	124	\Glstext: switched to using \@gls@hyp@opt	123
\Glsfirst: switched to using \@gls@hyp@opt	124	\glstext: switched to using \@gls@hyp@opt	123
\glsfirst: switched to using \@gls@hyp@opt	123	\glstreenamfmt: new	298
\GLSfirstplural: switched to using \@gls@hyp@opt	126	\GLSuseri: switched to using \@gls@hyp@opt	130
\Glsfirstplural: switched to using \@gls@hyp@opt	125	\Glsuseri: switched to using \@gls@hyp@opt	130
\glsfirstplural: switched to using \@gls@hyp@opt	125	\glsuseri: switched to using \@gls@hyp@opt	130
\glsifhyper: deprecated	103	\GLSuserii: switched to using \@gls@hyp@opt	131
\glslink: switched to using \@gls@hyp@opt	104	\Glsuserii: switched to using \@gls@hyp@opt	131
\glslinkcheckfirsthyperhook: new	105	\glsuserii: switched to using \@gls@hyp@opt	131
\glslinkvar: new	103	\GLSuseriii: switched to using \@gls@hyp@opt	132
\GLSname: switched to using \@gls@hyp@opt	126	\Glsuseriii: switched to using \@gls@hyp@opt	132
\Glsname: switched to using \@gls@hyp@opt	126	\glsuseriii: switched to using \@gls@hyp@opt	132
\glsname: switched to using \@gls@hyp@opt	126	\GLSuseriv: switched to using \@gls@hyp@opt	133
\GLSpl: switched to using \@gls@hyp@opt	121	\Glsuseriv: switched to using \@gls@hyp@opt	133
\Glspl: switched to using \@gls@hyp@opt	120	\glsuseriv: switched to using \@gls@hyp@opt	132
\glspl: switched to using \@gls@hyp@opt	119	\GLSserv: switched to using \@gls@hyp@opt	134
\GLSplural: switched to using \@gls@hyp@opt	125		
\Glsplural: switched to using \@gls@hyp@opt	124		

\Glsuserv: switched to using	4.12 (2014-11-22)
\@gls@hyp@opt	133
\glsuserv: switched to using	
\@gls@hyp@opt	133
\GLSuservi: switched to using	
\@gls@hyp@opt	135
\Glsuservi: switched to using	
\@gls@hyp@opt	134
\glsuservi: switched to using	
\@gls@hyp@opt	134
\ifignoredglossary: new	58
altlongragged4col: fixed bug that displayed description instead of symbol	278
\newglossary: added starred version ..	56
\newignoredglossary: new	58
\ns@newglossary: added \@glotype@<name>@log	56
new	56
\p@gls@hyp@opt: new	103
\PGLS: changed to use \@gls@hyp@opt	254
\Pgls: changed to use \@gls@hyp@opt	252
\pgls: changed to use \@gls@hyp@opt	251
\PGLSpl: changed to use \@gls@hyp@opt	254
\Pglspl: changed to use \@gls@hyp@opt	253
\pglspl: changed to use \@gls@hyp@opt	252
\s@gls@hyp@opt: new	103
\s@newglossary: new	56
automake: new	25
4.09 (2014-08-12)	
\glsaddkey: fixed bug in user commands	70
4.10 (2014-08-27)	
\@Gls@acentryname: new	144
\@Gls@entryname: new	143
\@gls@glossary: Renamed \@glossary to \@gls@glossary	169
\glspercentchar: new	153
\glstildechar: new	153
alttree: moved space after symbol	305, 306
4.11 (2014-09-01)	
\@do@wrglossary: added hook	172
sanitize: none option	20
\gls@wrglossary: renamed from \@wrglossary to \gls@wrglossary	169
\glsaddprotectedpagefmt: new	171
\glsbackslash: new	153
\@gls@addpredefinedattributes: Added glsignore attribute	42
\@gls@adjustmode: new	152
\@gls@notranslatorhook: removed ...	21
\@gls@toc: added \protect to \numberline	39
\@gls@usetranslator: new	21
\glsacrpluralsuffix: new	30
\glsadd: added check for vertical mode	151
\glsaddallunused: replaced @gobble with glsignore	152
\glsifusedtranslatorordict: new	21
\glsignore: new	152
\glsupacrpluralsuffix: new	30
\ProvidesGlossariesLang: new	30
\RequireGlossariesLang: new	30
4.13 (2015-02-03)	
\indexspace: new	260, 280, 298
4.14 (2015-02-28)	
\@@glslocalreset: new	85
\@@glslocalunset: new	84
\@@glsreset: new	85
\@@glsunset: new	85
\@newglossaryentry@defcounters: new	86
\cGls: new	89
\cGls@: new	89
\cGlspl@: new	90
\cglsl: new	89
\cglsl@: new	89
\cglspl: new	90
\cglspl@: new	90
\@gls@entry@count: new	89
\@gls@increment@currcount: new ...	88
\@gls@local@increment@currcount: new	88
\@gls@write@entrycounts: new	88
\@glslocalreset: new	85
\@glslocalunset: new	84
\@glsreset: new	85
\@glsunset: new	84
\@newglossaryentry@defcounters: new	81
\cGls: new	89
\cglsl: new	89
\cGlsformat: new	90
\cglslformat: new	89
\cGlspl: new	90

\cglspl:new	90	\glswriteentry:new	170
\cGlsplformat:new	91	\ifglsfieldcseq:new	74
\cglsplformat:new	90	\ifglsfielddefeq:new	74
\gls@defdocnewglossaryentry:new ..	66	\ifglsfieldeq:new	73
\glsenableentrycount:new	86	long-sp-short:new	213
\glslocalreset:switched to \@glslocalreset	84	\showglofield:new	246
4.18 (2015-09-09)			
\glslocalunset:switched to \@glslocalunset	84	General: split mfirstuc into separate bundle	4
4.19 (2015-10-31)			
\glsreset:switched to \@glsreset ...	83	\glstreenamibox:new	304
\glsunset:switched to \@glsunset ...	84	4.19 (2015-11-22)	
4.15 (2015-03-16)			
General: bug fix replaced \@glo@type with \glstype	142	\@gls@link@nocheckfirsthyper:new	122
4.16 (2015-06-18)			
\glsaddstoragekey:new	68	\@gls@preglossaryhook:new	177
4.16 (2015-07-08)			
\@ACRlong: added \glspostlinkhook	348	\@printglossary: added \@gls@preglossaryhook	179
\@ACRshort: added \glspostlinkhook	347	\do@gl:disablehyperinlist:new ..	105
\@Acrlong: added \glspostlinkhook	347	\doifglossarynoexistsordo:new ...	51
\@Acrshort: added \glspostlinkhook	346	\gls@gobbleopt:new	55
\@GLS@: added \glspostlinkhook ...	119	\glsdoifexistsordo:new	50
\@GLSpl: added \glspostlinkhook ..	121	4.20 (2015-11-30)	
\@Gls@: added \glspostlinkhook ...	118	\@gls@link: added \@gls@setdefault@glslink@opts	105
\@Glspl@: added \glspostlinkhook ..	121	added \glsdonohyperlink when hyperlink is suppressed	106
\@acrlong: added \glspostlinkhook	347	\@gls@setdefault@glslink@opts: new	105
\@acrshort: added \glspostlinkhook	346	\gls@checkseeallowed@preambleonly: new	61
\@gls@: added \glspostlinkhook ...	117	\glsdonohyperlink:new	115
\@gls@@link: added \glspostlinkhook	104	4.21 (2016-01-24)	
\@gls@field@link: added \glspostlinkhook	122	\@printglossary: warn if no style has been set	178
\@gls@link: moved definition of \glsifhyperon outside of this macro	106	General: changed checkfirsthyper assignment	135–142
\@glsdisp: added \glspostlinkhook	122	\glossarystyle: set default style if not already set	202
\@glspl@: added \glspostlinkhook ..	120	\glsLTpenaltycheck:new	273
General: added \glspostlinkhook	135–142	\glspatchLToutput:new	274
\glsacspace:new	214	\glspenaltygroupskip:new	273
\glsadd: changed \@do@wrglossary to \@do@wrglossary	152	altlong4col-booktabs:new	272
\glsfielddef:new	72	altlongragged4col-booktabs:new ..	273
\glsfieldedef:new	72	long-booktabs:new	270
\glsfieldfetch:new	73	long3col-booktabs:new	271
\glsfieldgdef:new	72	long4col-booktabs:new	271
\glsfieldxdef:new	71	longragged-booktabs:new	272
\glsifhyperon: moved definition of \glsifhyperon	105	longragged3col-booktabs:new	273
\glslinkpostsetkeys:new	105	\setglossarystyle: set default style if not already set	201
\glspostlinkhook:new	104		

4.22 (2016-04-19)

\@do@wrglossary: added check for \@arabic	172	altlongragged4col: check for nogroup- skip changed	279
added test to allow temporary primitive modifications and added arabic case	172	altsuperragged4col: check for nogroupskip changed	297
mcolalttreespannav: new	285	long: check for nogroupskip changed ..	265
mcolindexspannav: new	281	long-booktabs: check for nogroupskip changed	270
mcoltreenonamespannav: new	284	long3col: check for nogroupskip changed	266
mcoltreespannav: new	282	long3col-booktabs: check for nogroup- skip changed	271
\gls@arabicpage: new	171	long4col: check for nogroupskip changed	268
\gls@protected@pagefmts: added ara- bic to list	170	long4col-booktabs: check for nogroup- skip changed	272
\glsentrytitlecase: new	146	longragged: check for nogroupskip changed	275
\glsfindwidesttoplevelname: new .	303	longragged3col: check for nogroupskip changed	277
\glslistgroupheaderfmt: new	261	super: check for nogroupskip changed .	287
\glslistnavigationitem: new	261	super3col: check for nogroupskip changed	288
\glstreegroupheaderfmt: new	298	super4col: check for nogroupskip changed	290
\glstreenavigationfmt: new	299	superragged: check for nogroupskip changed	293
\ifglswrallowprimitivemods: new .	171	superragged3col: check for nogroup- skip changed	295
list: fixed missing space before descrip- tion	261		
long: fixed typo in \glossentrydesc .	265		
super4col: fixed bug in \glossentry .	290		
4.23 (2016-04-30)			
\glscurrentfieldvalue: new	54		
\ifglshasfield: added \glscurrentfieldvaluesuperragged3col: check for nogroup-	53, 54		

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	
\!	112
\"	18, 109–112, 114
\#	156
\%	153, 158, 159, 312, 313
\&	30, 151
\'	18
\.	8, 18
\=	18
\?	109, 111
\@@delimN	204
\@@do@@wrglossary	164, 173
\@@do@wrglossary	152, 170
\@@glo@assign@sortkey	165
\@@glo@list	48
\@@glo@sort	18
\@@glo@type	177
\@@glossarysec	5, 37, 38
\@@glossaryseclabel	6, 38, 191
\@@glossarysecstar	6, 37, 38, 191
\@@gls@checkactual	113, 114
\@@gls@checkbar	112, 113
\@@gls@checkescactual	111
\@@gls@checkescbar	111, 112
\@@gls@checkesclevel	112
\@@gls@checkescquote	110
\@@gls@checklevel	113
\@@gls@checkquote	110
\@@gls@default@entryfmt	92, 101
\@@gls@expand@field	16, 64, 65, 68, 69, 225, 227–229, 232, 234, 237–239, 359–363
\@@gls@fixbraces	175
\@@gls@noexpand@field	17, 63, 64
\@@gls@noidx@no@sanitizesort	18
\@@gls@noidx@nosanitizesort	166
\@@gls@nosanitizesort	17, 166
\@@gls@sanitizesort	17, 166
\@@gls@xdycheckbackslash	114, 115
\@@gls@xdycheckquote	114
\@@gls@localreset	85, 87
\@@gls@localunset	84, 87
\@@gls@reset	85, 87
\@@gls@unset	84, 87
\@@newglossaryentry@defcounters	86
\@@this@glo@	49
\@ACRfull	208
\@ACRfullpl	209
\@ACRlong	140, 208
\@ACRlongpl	142, 209
\@ACRshort	136, 208
\@ACRshortpl	138, 209
\@Acrfull	207, 208
\@Acrfullpl	209
\@Acrlong	139, 208
\@Acrlongpl	141, 209
\@Acrshort	135, 136
\@Acrshortpl	137
\@Alph	170, 172
\@GLS	118
\@GLS@	118, 254
\@GLSdesc	127
\@GLSdesc@	127
\@GLSdescplural	128
\@GLSdescplural@	128
\@GLSfirst	124
\@GLSfirst@	124
\@GLSfirstplural	126
\@GLSfirstplural@	126
\@GLSname	126
\@GLSname@	126, 127
\@GLSpl	121
\@GLSpl@	121, 255
\@GLSplural	125
\@GLSplural@	125

\@GLSsymbol	129	\@Glsuseriv	133
\@GLSsymbol@	129	\@Glsuseriv@	133
\@GLSsymbolplural	130	\@Glsuserv	133, 134
\@GLSsymbolplural@	130	\@Glsuserv@	134
\@GLStext	123	\@Glsuservi	134
\@GLStext@	123	\@Glsuservi@	134
\@GLSuseri	130, 131	\@Mi	274
\@GLSuseri@	131	\@PGLS	254
\@GLSuserii	131	\@PGLS@	254
\@GLSuserii@	131	\@PGLSpl	254
\@GLSuseriii	132	\@PGLSpl@	254
\@GLSuseriii@	132	\@Pgls	252
\@GLSuseriv	133	\@Pgls@	252
\@GLSuseriv@	133	\@Pglspl	253
\@GLSuserv	134	\@Pglspl@	253
\@GLSuserv@	134	\@Roman	171, 172
\@GLSuservi	135	\@acrfull	207
\@GLSuservi@	135	\@acrfullpl	208
\@Gls	117	\@acrlong	138, 139, 207
\@Gls@	88, 90, 117, 253	\@acrlongpl	140, 141, 209
\@Gls@acrentryname	210	\@acrshort	135, 207, 208
\@Gls@entry@field	70, 144–149	\@acrshortpl	137, 209
\@Gls@entryname	143, 210	\@addtoacronymlists	14
\@Glsdesc	127	\@after	14
\@Glsdesc@	127	\@afterheading	262, 316
\@Glsdescplural	128	\@alph	170, 172
\@Glsdescplural@	128	\@arabic	171, 172
\@Glsfirst	124	\@auxout	55, 56, 89, 162, 164, 167, 176, 180, 256
\@Glsfirst@	124	\@backslashchar	108, 114, 115
\@Glsfirstplural	125	\@before	14
\@Glsfirstplural@	125	\@bsphack	169
\@Glsname	126	\@cGls	89
\@Glsname@	126	\@cGls@	88, 89
\@Glspl	120	\@cGlspl	90
\@Glspl@	88, 91, 120, 253, 254	\@cGlspl@	88, 90
\@Glsplural	124, 125	\@cclv	274
\@Glsplural@	125	\@cgls	89
\@Glsymbol	129	\@cgls@	87, 89
\@Glsymbol@	129	\@cglspl	90
\@Glsymbolplural	129	\@cglspl@	87, 90
\@Glsymbolplural@	129, 130	\@chapter	28
\@Glstext	123	\@classoptionslist	27
\@Glstext@	123	\@colht	274
\@Glsuseri	130	\@colroom	274
\@Glsuseri@	130	\@currentlabelname	6, 191
\@Glsuserii	131	\@curroptions	27
\@Glsuserii@	131	\@declaredoptions	27
\@Glsuseriii	132	\@delimN	203
\@Glsuseriii@	132	\@delimR	203

\@disable@onlypremakeg	162	\@glo@descplural	59, 74, 75
\@disable@premakecs	29	\@glo@descpluralaccess	330–332
\@disabled@gl\$addxdycounters	41	\@glo@do@sortentries	181
\@do@addcounter	40	\@glo@entry	152
\@do@auxoutstuff	180	\@glo@entryprefix	249
\@do@glossentry	196, 197, 328	\@glo@entryprefixfirst	249
\@do@gl\$@getcounterprefix	173	\@glo@entryprefixfirstplural ..	249, 250
\@do@gl\$@islistofacronyms	14	\@glo@entryprefixplural	249
\@do@gl\$see	80	\@glo@esclabel	82, 83
\@do@ifinlist	40	\@glo@etext	92–94
\@do@newglossaryentry	211,	\@glo@first	60, 76, 78, 79, 236, 363
225, 227–229, 231–234, 236–241,	359–363	\@glo@firstaccess	329, 331, 332
\@do@seeglossary	164, 175	\@glo@firstplural	60, 76, 79, 363
\@do@subglossentry	198, 329	\@glo@firstpluralaccess	330–332
\@do@wrglossary	106	\@glo@grabfirst	187
\@do@writeaux@info	176	\@glo@label	63, 69–80,
\@ehc	274	86, 150, 151, 249, 250, 303, 304,	331, 332
\@empty	11, 14, 25–27, 29,	\@glo@list	80
40, 41, 44, 47, 48, 77, 82, 107,	117–121,	\@glo@long	52, 62, 76, 79
135–142, 154, 157, 159, 161, 168,	169,	\@glo@longaccess	330–332
174, 191, 194, 201, 226, 228, 230,	232–	\@glo@longpl	62,
235, 237, 239, 241, 310, 312, 314,	346–348	76, 79, 225, 227, 229, 231, 234,	236, 238, 359
\@end@fixbraces	175	\@glo@longpluralaccess	330–332
\@end@fortrue	22, 51, 68, 256	\@glo@name	10, 59, 63, 75, 78, 79
\@esphack	170	\@glo@no@assign@sortkey	163
\@expandtwoargs	27	\@glo@numfmt	173, 310
\@first@ofone	18, 19	\@glo@parent	11, 61, 76–78, 82, 83, 182
\@first@ofthree	103, 116,	\@glo@plural	59, 76, 78, 361
117, 119, 122, 135, 137, 139, 141,	346–348	\@glo@pluralaccess	329, 331, 332
\@first@oftwo	21,	\@glo@prefix 7, 61, 76, 82, 83,	108, 173, 309, 310
22, 67, 68, 103, 119–121, 137,	138, 141, 142	\@glo@range	173, 309, 310
\@for	22,	\@glo@see	61, 76, 80
27, 29, 40, 41, 48, 67, 68, 108,	154–156,	\@glo@seeautonumberlist	7, 61
162, 163, 170, 175, 181, 212,	226, 228,	\@glo@short	53, 62, 76, 79, 362
230, 232, 234, 237, 239, 241,	256, 257, 310	\@glo@shortaccess	330–332, 359–362
\@glo@@desc	79	\@glo@shortpl	62, 76, 79,
\@glo@@symbol	80	225, 227, 229, 231, 233, 236,	238, 359, 362
\@glo@access	329, 331, 332, 334	\@glo@shortpluralaccess	330–332
\@glo@addchildren	181, 186	\@glo@sort	10, 17, 18, 59, 76, 78, 82, 83
\@glo@assign@sortkey	163, 165, 193	\@glo@sortedinsert	182
\@glo@check@mkidxrangechar	107, 108, 173, 309, 310	\@glo@sortentries	184, 185
\@glo@childlist	181	\@glo@sorthandler@case	185
\@glo@counter	61, 76, 79	\@glo@sorthandler@letter	184
\@glo@counterprefix 167, 172–174,	201, 204	\@glo@sorthandler@nocase	185
\@glo@default@sorttype ...	9, 165, 184, 185	\@glo@sorthandler@word	184
\@glo@defaultcounter	79	\@glo@sortinghandler	181, 183
\@glo@desc	59, 74, 75, 77, 79	\@glo@sortinglist	181, 182, 185
\@glo@descaccess	330–332	\@glo@sorttype	165, 186, 187, 193
		\@glo@storeentry	10, 12

\@glo@suffix	108, 173, 310	\@gls@checkescbar	109
\@glo@symbol	52, 60, 75, 80, 231, 236, 331	\@gls@checkescquote	109
\@glo@symbolaccess	330–332, 362	\@gls@checklevel	109
\@glo@symbolplural	60, 75, 80	\@gls@checkmkidxchars	
\@glo@symbolpluralaccess	330–332	82, 108, 164, 172, 174, 309, 310
\@glo@text	59,	\@gls@checkquote	109
76, 78, 80, 117–122, 143, 144, 231, 250, 361		\@gls@classI	155
\@glo@textaccess ...	329, 331, 332, 359–362	\@gls@classII	155
\@glo@thislabel	81	\@gls@codepage	180
\@glo@thislettergrp	187, 188	\@gls@counter	
\@glo@thisvalue	53, 54	102, 105–107, 151, 152, 167, 173, 174, 310	
\@glo@tmp	69, 70, 174	\@gls@counterwithin	9, 191, 194
\@glo@type	6,	\@gls@ctr	40
11, 60, 75–77, 79, 80, 151, 152, 162, 168,		\@gls@currentlettergroup	186, 188
178–182, 186, 187, 190, 191, 210, 226,		\@gls@declareoption	
228, 230, 232, 234, 237, 239, 241, 256, 257		7, 8, 12, 13, 16, 21, 24–27
\@glo@types		\@gls@default	91
. 48, 49, 56, 85, 86, 152, 162, 163, 247, 303		\@gls@default@value	
\@glo@useri	61, 76, 79	52–54, 64, 65, 75, 76, 78, 79, 235, 249
\@glo@userii	62, 76, 79	\@gls@deffile	66, 67
\@glo@useriii	62, 76, 79	\@gls@defsort	10, 11, 80
\@glo@useriv	62, 76, 79	\@gls@defsortcount	10, 11, 57
\@glo@userv	62, 76, 79	\@gls@do@acronymsdef	13, 28, 58
\@glo@uservi	62, 76, 79	\@gls@do@indexdef	27, 28, 58
\@glodesc	79	\@gls@do@numbersdef	26, 28, 58
\@glolist@	77	\@gls@do@symbolsdef	26, 58
\@gloname	79	\@gls@do@symbolssdef	28
\@glossary@default@style		\@gls@doautomake	25, 163
.....	6, 8, 178, 201, 202, 242	\@gls@docloadedfalse	4
\@glossaryentryfield	82, 83	\@gls@docloadedtrue	4
\@glossarysection	36	\@gls@dodeflistparser	163
\@glossarystyle	178, 179, 190	\@gls@doentrydef	101
\@glossarysubentryfield	82, 83	\@gls@dolast	175, 176
\@gls	116	\@gls@donext	175, 176
\@gls@	87, 89, 117, 252, 253	\@gls@donext@def	150
\@gls@@link	104	\@gls@dothiswrite	161
\@gls@Hcounter	106, 107	\@gls@elem	256
\@gls@ReturnAfterFi	204	\@gls@encapchar	
\@gls@access@display	334, 335	111, 112, 158, 173, 175, 310, 313
\@gls@actualchar	83, 111, 113, 158, 313	\@gls@entry@count	88, 89
\@gls@addpredefinedattributes .	154, 160	\@gls@entry@field	
\@gls@adjustmode	151	69, 70, 87, 143–150, 332–334
\@gls@automake	161, 163	\@gls@escbsdq	109, 159, 314
\@gls@between	257, 258	\@gls@expand@fields	64, 65
\@gls@body	144	\@gls@expandonce	65
\@gls@checkactual	109	\@gls@fetchfield	53
\@gls@checkbar	109	\@gls@field@link	70, 71, 123–135
\@gls@checkedmkidx	108–115	\@gls@firsttok	187
\@gls@checkescactual	109	\@gls@fixbraces	80

<code>\@gls@forbidtexext</code>	56	<code>\@gls@numberlink</code>	204
<code>\@gls@get@counterprefix</code>	174	<code>\@gls@numbersdef</code>	26
<code>\@gls@getbody</code>	144	<code>\@gls@numlist@lastsep</code>	150, 151
<code>\@gls@getcounterprefix</code>	173	<code>\@gls@numlist@nextsep</code>	150
<code>\@gls@getgrouptitle</code>	164, 200, 258	<code>\@gls@numlist@sep</code>	150, 151
<code>\@gls@glossary</code>	169	<code>\@gls@old@chapter</code>	28
<code>\@gls@gobbleopt</code>	55	<code>\@gls@oldnewglossaryentryposthook</code> ..	331
<code>\@gls@grptitle</code>	200, 256, 258	<code>\@gls@oldnewglossaryentryprehook</code> ..	331
<code>\@gls@hyp@opt</code>	70, 71, 89, 90, 104, 116–121, 123–142, 207–209, 251–254	<code>\@gls@onlypremakeg</code>	29
<code>\@gls@hyp@opt@cs</code>	103	<code>\@gls@order</code>	161
<code>\@gls@hypergroup</code>	256	<code>\@gls@org@LT@output</code>	273
<code>\@gls@ifinlist</code>	40	<code>\@gls@org@glsnoidxdisplayloc</code>	166
<code>\@gls@ifnotmeasuring</code>	83	<code>\@gls@org@glssseeformat</code>	166
<code>\@gls@igtype</code>	58	<code>\@gls@preglossaryhook</code>	179
<code>\@gls@increment@currcount</code>	87	<code>\@gls@prevlevel</code> ..	284, 285, 304–307, 322, 323
<code>\@gls@indexdef</code>	27	<code>\@gls@provide@newglossary</code>	56
<code>\@gls@islistofacronyms</code>	14	<code>\@gls@quotechar</code>	110–113, 158, 313
<code>\@gls@keylist</code>	358	<code>\@gls@reference</code>	164, 167
<code>\@gls@keymap</code>	67–70, 249, 331	<code>\@gls@removespaces</code>	204
<code>\@gls@label</code>	164, 167, 173	<code>\@gls@renewglossary</code>	160
<code>\@gls@langmod</code>	161	<code>\@gls@replacementtext</code>	334
<code>\@gls@levelchar</code>	83, 112, 113, 158, 313	<code>\@gls@rest</code>	144
<code>\@gls@link</code> ..	104, 117–122, 135–142, 346–348	<code>\@gls@roman</code>	43, 310, 311
<code>\@gls@link@checkfirsthyper</code> ..	104, 117–122	<code>\@gls@sanitized@tmp</code>	108, 109
<code>\@gls@link@label</code>	105, 227, 233	<code>\@gls@sanitizedesc</code>	23
<code>\@gls@link@nocheckfirsthyper</code> ..	122, 135–142	<code>\@gls@sanitizesort</code>	10
<code>\@gls@link@opts</code>	105, 227, 233	<code>\@gls@sanitizesymbol</code>	23
<code>\@gls@list</code>	256, 257	<code>\@gls@saveentrycounter</code>	106, 152
<code>\@gls@listsuffix</code>	40	<code>\@gls@setacrstyle</code>	23, 28
<code>\@gls@loadlist</code>	8, 241	<code>\@gls@setcounter</code>	57
<code>\@gls@loadlong</code>	7, 8, 242	<code>\@gls@setdefault@glslink@opts</code>	105
<code>\@gls@loadsuper</code>	8, 242	<code>\@gls@setsort</code>	10, 11, 106
<code>\@gls@loadtree</code>	8, 242	<code>\@gls@setupshortcuts</code>	28
<code>\@gls@local@increment@currcount</code> ..	87	<code>\@gls@sort</code>	188
<code>\@gls@loclist</code>	165, 166, 187, 188	<code>\@gls@sort@A</code>	183, 184
<code>\@gls@map</code>	67, 68	<code>\@gls@sort@B</code>	183, 184
<code>\@gls@missingnumberlist</code>	79	<code>\@gls@startswithexpandonce</code>	64
<code>\@gls@noaccess</code>	334	<code>\@gls@symbolsdef</code>	26
<code>\@gls@noexpand@fields</code>	65	<code>\@gls@this</code>	170
<code>\@gls@nohyperlist</code>	15, 58, 105	<code>\@gls@thisHloc</code>	174
<code>\@gls@noidx@do</code>	186	<code>\@gls@thisfield</code>	53, 54
<code>\@gls@noidx@getgrouptitle</code>	164, 200	<code>\@gls@thislabel</code>	51, 175, 176, 185
<code>\@gls@noidx@sanitizesort</code>	18, 166	<code>\@gls@thislist</code>	150, 151
<code>\@gls@noidx@setsanitizesort</code>	20, 166	<code>\@gls@thisloc</code>	174
<code>\@gls@noidx@loclist@finalsep</code>	166	<code>\@gls@thisval</code>	68
<code>\@gls@noidx@loclist@prev</code>	166, 189	<code>\@gls@title</code>	36
<code>\@gls@noidx@loclist@sep</code>	166, 189	<code>\@gls@tmp</code> ..	11, 31, 44, 65, 108, 109, 169, 257, 258
<code>\@gls@noref@warn</code>	164, 187	<code>\@gls@tmpb</code>	110–115
		<code>\@gls@toc</code>	38

\@gls@type	163, 212, 226, 228, 230, 232, 234, 237, 239, 241, 303
\@gls@updatechecked	108, 109
\@gls@usetranslator	21, 22, 31
\@gls@value	64, 146
\@gls@warnonglossdefined	16, 177
\@gls@warnontheGLOSSdefined	16, 196
\@gls@write@entrycounts	88
\@gls@writedef	66
\@gls@xdy@locationlist	155
\@gls@xdycheckbackslash	108
\@gls@xdycheckquote	108
\@gls@xref	174, 175
\@gls@Alphacompositor	34, 44, 311
\@gls@Hlocref	172, 173
\@gls@acronymlists ..	14, 15, 48, 210, 212, 226, 228, 230, 232, 234, 237, 239, 241, 246
\@gls@addkey	69
\@gls@addstoragekey	68, 69
\@gls@addxdyattribute	41
\@gls@defaultsort	10
\@gls@desc	127
\@gls@desc@	127
\@gls@descplural	127, 128
\@gls@descplural@	128
\@gls@disp	121
\@gls@entry	85, 86, 89
\@gls@entrytitlecase	146, 147
\@gls@first	123
\@gls@first@	123, 124
\@gls@firstletter	48, 153
\@gls@firstplural	125
\@gls@firstplural@	125
\@gls@hypernumber	203
\@gls@isacronymlistfalse	14
\@gls@isacronymlisttrue	14
\@gls@link	106, 116, 151, 256
\@gls@localreset	84, 87
\@gls@localunset	84, 87
\@gls@locref	167, 172, 173, 309, 310
\@gls@minrange	154, 155, 311
\@gls@name	126
\@gls@name@	126
\@gls@nextpages	179
\@gls@nodesc	75, 77, 79
\@gls@noname	75, 78, 79
\@gls@nonextpages	178
\@gls@numberformat	102, 105, 151, 167, 173, 309, 310
\@gls@openfile	160, 168
\@gls@order	162
\@gls@pl	119
\@gls@pl@	87, 90, 119, 252-254
\@gls@plural	124
\@gls@plural@	124
\@gls@reset	83, 87
\@gls@see	80, 175
\@gls@symbol	128
\@gls@symbol@	128
\@gls@symbolplural	129
\@gls@symbolplural@	129
\@gls@target	116, 196, 256
\@gls@text	123
\@gls@text@	123
\@gls@unset	84, 87
\@gls@useri	130
\@gls@useri@	130
\@gls@userii	131
\@gls@userii@	131
\@gls@useriii	132
\@gls@useriii@	132
\@gls@useriv	132
\@gls@useriv@	132, 133
\@gls@userv	133
\@gls@userv@	133
\@gls@uservi	134
\@gls@uservi@	134
\@gls@widestname	304, 305, 322
\@gls@writefiles	25
\@gobble	11, 66, 67, 108, 153, 156, 164, 308, 312, 313
\@idxitem	281, 299
\@ifclassloaded	4, 9, 37
\@ifnextchar	57, 103
\@ifpackageloaded	4, 21, 22, 31, 47, 83, 150, 328
\@ifstar	56, 68, 69, 103, 206
\@ifundefined	30, 257, 264, 275, 286, 293, 305, 322, 336
\@ignored@glossaries	58
\@input@	179
\@istfilename	162
\@makecol	274
\@makeglossary	162
\@minus	260, 280, 298
\@mkboth	37
\@newglossary	55, 56
\@newglossaryentry@defcounters ..	80, 86

<code>\csuse</code>	32, 35, 55, 64, 70, 71, 101, 161, 182, 184, 186, 188, 189, 191, 192, 202, 213, 250, 315–327
<code>\csxdef</code>	79, 88
<code>\currentglossary</code>	35, 178, 191, 194
<code>\currentglssubentry</code>	192, 194, 195
<code>\CurrentOption</code>	27, 249, 328
<code>\CurrentTrackedLanguage</code>	32, 366, 367
<code>\CurrentTrackedTag</code>	32, 366, 367
<code>\CustomAcronymFields</code>	241
<code>\CustomNewAcronymDef</code>	241
D	
<code>\d</code>	18
<code>datatool package</code>	183
<code>\day</code>	154, 158, 310, 313
<code>\DeclareAcronymList</code>	13, 15, 210, 212, 226, 228, 230, 232, 234, 237, 239, 241
<code>\DeclareListParser</code>	163
<code>\DeclareOption</code>	7, 249, 328
<code>\DeclareOptionX</code>	7
<code>\DeclareRobustCommand</code>	32, 175, 176, 235, 334–336
<code>\def</code> ..	7, 10, 11, 14, 18, 19, 24, 28, 29, 32, 33, 36, 40, 42–48, 51, 55–57, 59–62, 65, 73, 75–81, 87–91, 101, 102, 105–115, 117– 142, 150, 151, 154, 158, 161, 163, 165, 166, 168, 171–175, 178, 181, 185–187, 189–194, 200–205, 207–210, 225–232, 234, 236–239, 241, 249, 251–255, 257– 260, 284, 285, 304–307, 309, 310, 313, 315, 322, 323, 328–331, 345–348, 359–363
<code>\def@gl@xdycheckbackslash</code>	114, 115
<code>\DefaultNewAcronymDef</code>	226
<code>\defgl@entryfmt</code>	56, 58, 101, 212, 225, 226, 229, 230, 233, 235, 238, 240
<code>\define@boolkey</code>	5, 7–9, 12, 19, 20, 23–26, 102, 192
<code>\define@choicekey</code>	5, 6, 9, 20, 21, 24, 61, 191, 192
<code>\define@key</code>	6, 9, 15, 20, 24, 25, 59– 62, 69, 70, 102, 151, 190, 193, 249, 329, 330
<code>\DefineAcronymSynonyms</code>	28, 224
<code>\delimN</code>	157, 163, 189, 204, 312
<code>\delimR</code>	157, 203, 204, 312
<code>\DescriptionDUANewAcronymDef</code>	230
<code>\DescriptionFootnoteNewAcronymDef</code> .	228
<code>\descriptionname</code>	32, 265, 267– 271, 276, 278–280, 287–292, 294, 296–298
<code>\DescriptionNewAcronymDef</code>	232
<code>\dimen@</code>	214, 274, 303, 304
<code>\disable@keys</code>	27
<code>\do</code>	22, 27, 29, 40, 41, 48, 67, 68, 108, 150, 154–156, 162, 163, 170, 175, 181, 212, 226, 228, 230, 232, 234, 237, 239, 241, 256, 257, 310
<code>\do@glo@storeentry</code>	10, 11, 80
<code>\do@gl@link@checkfirsthyper</code>	104, 106, 117–122, 135–142, 346, 347
<code>\do@gl@xdycheckbackslash</code>	108
<code>\do@gl@disablehyperinlist</code>	106
<code>\do@gl@shaschildren</code>	51
<code>doc package</code>	4, 5, 12
<code>\doifglossarynoexistsordo</code>	56
<code>\dtl@ifsingle</code>	200
<code>\dtl@insertinto</code>	183
<code>\dtl@sortresult</code>	183, 184
<code>\dtlcompare</code>	183
<code>\dtlicompare</code>	184
<code>\DTLifinlist</code>	58, 105
<code>\DTLifint</code>	200
<code>\dtlletterindexcompare</code>	183
<code>\DTLsubstituteall</code>	109
<code>\dtlwordindexcompare</code>	183
<code>\DUANewAcronymDef</code>	239
E	
<code>\eappto</code>	58, 81, 171
<code>\edef</code> ..	11, 14, 29, 31, 39–46, 48, 51, 56, 58, 64, 67, 68, 71–76, 81, 82, 101, 105–115, 150, 152, 153, 158, 161, 163, 164, 168, 173, 174, 176, 180–184, 187, 192, 195, 200, 204, 206, 225, 227, 229, 231, 233, 236, 238, 256, 308, 309, 311, 313, 358–362
<code>\egroup</code>	18, 75, 151, 179, 182
<code>\else</code>	8, 11–14, 17, 19, 20, 25, 27– 29, 33, 34, 37–43, 45–48, 61, 63, 77, 79, 81–83, 87, 88, 105–115, 117–122, 144, 153, 154, 157–161, 168–175, 178, 187, 191, 192, 194–196, 201, 203, 204, 214, 228, 230, 232, 235, 237–240, 242, 257, 261, 265, 266, 268, 271, 272, 274, 275, 277, 279, 287, 288, 290, 293, 295, 297, 300, 301, 303–306, 309–315, 320–323, 334
<code>\emph</code>	175, 205
<code>\empty</code>	204
<code>\end</code>	104, 157, 160, 161, 186, 261, 264–270, 272, 273, 275–298, 312

glossary styles:

altlist	262, 263, 316	longheaderborder	265, 317
altlistgroup	262, 263, 316	longragged	272, 275, 276
altlisthypergroup	263, 316	longragged-booktabs	272
altlong4col	269, 278, 318	longragged3col	273, 276, 277, 319
altlong4col-booktabs	272, 273	longragged3col-booktabs	273
altlong4colborder	269, 318	longragged3colborder	277, 319
altlong4colheader	269, 272, 318	longragged3colheader	277, 319
altlong4colheaderborder	270, 318	longragged3colheaderborder .	278, 319
altlongragged4col ...	273, 278, 279, 319	longraggedborder	276, 318
altlongragged4col-booktabs	273	longraggedheader	276, 319
altlongragged4colborder	279, 319	longraggedheaderborder	276, 319
altlongragged4colheader	279, 319	mcolalttree	284, 324
altlongragged4colheaderborder	279, 320	mcolalttreegroup	285, 324
altsuper4col	291, 292, 296, 327	mcolalttreehypergroup	285, 324
altsuper4colborder	292, 327	mcolindex	281, 323
altsuper4colheader	291, 327	mcolindexgroup	281, 323
altsuper4colheaderborder ...	292, 327	mcolindexhypergroup	281, 323
altsuperragged4col	296–298, 325	mcoltree	282, 323
altsuperragged4colborder ...	297, 325	mcoltreegroup	323
altsuperragged4colheader ...	297, 325	mcoltreehypergroup	282, 323
altsuperragged4colheaderborder .		mcoltreenoname	283, 324
.....	298, 325	mcoltreenonamegroup	283, 324
alttree	284, 304, 306, 322	mcoltreenonamehypergroup	283, 284, 324
alttreegroup	306, 323	sublistdotted	316
alttreehypergroup	306, 323	super	286–288, 294, 326
index	280, 299, 300, 320	super3col	288, 289, 326
indexgroup	300, 320	super3colborder	288, 326
indexhypergroup	300, 320	super3colheader	289, 326
inline	315	super3colheaderborder	289, 326
list	6, 261–263, 315	super4col	289–291, 327
listdotted	263, 264, 316	super4colborder	290, 327
listgroup	261, 262, 315	super4colheader	290, 327
listhypergroup	262, 316	super4colheaderborder	291, 327
long	264–266, 270, 275, 316, 318	superborder	287, 326
long-booktabs	270, 272	superheader	287, 326
long3col	266, 267, 271, 317	superheaderborder	287, 326
long3col-booktabs	271, 273	superragged	293, 294, 324
long3colborder	266, 317	superragged3col	294–296, 325
long3colheader	267, 271, 317	superragged3colborder	295, 325
long3colheaderborder	267, 317	superragged3colheader	296, 325
long4col	267–269, 271, 317	superragged3colheaderborder	296, 325
long4col-booktabs	271, 272	superraggedborder	294, 324
long4colborder	268, 318	superraggedheader	294, 324
long4colheader	268, 271, 318	superraggedheaderborder	294, 325
long4colheaderborder	268, 318	tree	282, 300–302, 304, 320
longborder	265, 317	treegroup	282, 301, 321
longheader	265, 270, 317	treehypergroup	301, 321
		treenoname	283, 302, 303, 321
		treenonamegroup	303, 322

treenonamehypergroup	303, 322	\gls@assign@descplural	225, 234, 236–239, 359, 362, 363
glossary-hypernav package	153	\gls@assign@field	65, 69, 70, 74, 76, 78–80, 249, 250
glossary-list package	6, 8, 260	\gls@assign@firstpl	225, 227–229, 231, 232, 234, 236–239, 359–363
glossary-long package	7, 264, 278, 286	\gls@assign@plural	225, 227–229, 231, 232, 234, 236–239, 359–363
glossary-longragged package	274	\gls@assign@symbolplural	225, 227–229, 231, 232, 237–239, 360, 361, 363
glossary-mcols package	280	\gls@checkisacronymlist	104
glossary-super package .. 7, 8, 264, 286, 292, 296		\gls@checkseeallowed	61, 66, 162, 164
glossary-superragged package	292	\gls@checkseeallowed@preambleonly ..	66
glossary-tree package	8, 298	\gls@codepage	47, 161, 180
\glossaryentry	173, 175, 310	\gls@defdocnewglossaryentry	66, 86
glossaryentry (counter)	9, 194, 195	\gls@defglossaryentry	65–67, 75
\glossaryentryfield	196, 315–322, 324–327, 349	\gls@disablepagerefexpansion ..	170, 172
\glossaryentrynumbers	7, 157, 178, 179, 188, 192, 193, 312	\gls@do@addxdyattribute	41
\glossaryheader	156, 186, 258, 261–271, 275–283, 285, 286, 288, 289, 293, 295, 296, 299–304, 306, 312	\gls@doclearpage	39
\glossarymark	36	\gls@dosubst	109
\glossaryname	12, 31, 32	\gls@dotocitle	178, 179, 190
\glossarypostamble	157, 186, 312	\gls@end@sanitizesort	18
\glossarypreamble	156, 186, 312	\gls@endcheck	64, 65
\glossarysection	156, 186, 312	\gls@glossary	169, 173–175
glossarysubentry (counter)	9, 194–196	\gls@gobbleopt	57
\glossarysubentryfield	198, 315–322, 324–327, 349	\gls@grplabel	256
\glossarytitle	156, 178, 186, 190, 312	\gls@hypergroup prerun	257
\glossarytocitle	6, 12, 13, 26, 27, 29, 32, 36, 156, 178, 186, 190, 191, 312	\gls@ifnotmeasuring	83, 84
\glossentry	81, 179, 188, 198, 259, 261–264, 266, 267, 275, 277, 278, 286, 288, 290, 293, 295, 297, 299, 301, 302, 304	\gls@inlinepostchild	258–260, 315
\Glossentrydesc	348	\gls@inlinesep	258, 259, 315
\glossentrydesc	259, 261–268, 275, 277, 278, 286–288, 290, 293, 295, 297, 299–303, 305, 306, 348	\gls@inlinesubsep	258, 259, 315
\glossentryname	259, 261–264, 266, 267, 275, 277, 278, 286, 288, 290, 293, 295, 297, 299–302, 305, 306, 348	\gls@islistofacronyms	14
\Glossentrysymbol	349	\gls@istfilebase	33, 161
\glossentrysymbol	259, 267, 268, 278, 279, 290, 297, 299–302, 305, 306, 349	\gls@label	205
\Gls	90, 206, 224	\gls@level	77, 78, 188
\gls	89, 164, 195, 206, 224	\gls@noidxglossary	164
\gls@Alphpage	170, 172	\gls@numberpage	170, 172
\gls@alphpage	170, 172	\gls@org@glossaryentryfield	179
\gls@arabicpage	170, 172	\gls@org@glossarysubentryfield	179
\gls@assign@desc	75, 80	\gls@org@insert	230, 233, 235
		\gls@protected@pagefmts	108, 170, 171
		\gls@Romanpage	170, 172
		\gls@romanpage	170, 172
		\gls@save@numberlist	7
		\gls@suffixF	34, 157, 159, 312, 314
		\gls@suffixFF	35, 157, 159, 312, 314
		\gls@text	100
		\gls@thissty	22
		\gls@tmp	168, 235

<code>\gls@tmplen</code>	115, 303–306, 322, 323	<code>\glsdetoklabel</code>	
<code>\gls@tr@set@acronym@toctitle</code>	13	... 49–54, 67, 71–75, 81, 84, 85, 87, 88,	
<code>\gls@tr@set@main@toctitle</code>	12	105, 143, 144, 150–152, 164–166, 173,	
<code>\gls@tr@set@numbers@toctitle</code>	27	179, 182–184, 187, 188, 190–192, 194,	
<code>\gls@tr@set@symbols@toctitle</code>	26	195, 197, 242–246, 304, 328, 329, 364, 365	
<code>\gls@wrglossary</code>	169	<code>\glsdisplay</code>	92, 101
<code>\gls@xdyststring</code>	108, 109	<code>\glsdisplayfirst</code>	92, 101
<code>\gls@xindy@glslnumbersfalse</code>	25	<code>\glsdisplaynumberlist</code>	165
<code>\gls@xindy@glslnumberstrue</code>	24	<code>\glsdohyperlink</code>	115, 116
<code>\glsaccsupp</code>	334	<code>\glsdohypertarget</code>	116
<code>\glsacronymtrue</code>	13	<code>\glsdoifexists</code>	
<code>\glsacrpluralsuffix</code> 51–53, 71–74, 83, 84, 117–122, 135–	
..... 30, 206, 215, 216, 220–222, 226		142, 150, 151, 165, 166, 251–255, 345–349	
<code>\glsacrshortcutsfalse</code>	28	<code>\glsdoifexistsordo</code>	104, 143
<code>\glsacrshortcutstrue</code>	28	<code>\glsdoifexistsorwarn</code>	190, 197, 198
<code>\glsacspace</code>	214, 216	<code>\glsdoifnoexists</code>	65, 75
<code>\glsadd</code>	152	<code>\glsdonohyperlink</code>	106, 115, 116
<code>\glsadd options</code>		<code>\glsdosanitizesort</code>	9, 10
counter	151	<code>\glsentryaccess</code>	334
format	151, 203	<code>\glsentrycounter</code>	201, 204
<code>\glsaddall options</code>		<code>\glsentrycounterfalse</code>	9
types	151, 152	<code>\glsentrycounterlabel</code>	191, 192, 195
<code>\GlsAddXdyAttribute</code>	40–42, 308, 309	<code>\glsentrycountertrue</code>	9
<code>\GlsAddXdyCounters</code>	40, 41, 57	<code>\glsentrycurrcount</code>	87–89
<code>\glsautomakefalse</code>	25	<code>\Glsentrydesc</code>	127, 197, 348
<code>\glsautoprefix</code>	6, 191	<code>\glsentrydesc .</code> 94, 95, 127, 197, 338–340, 348	
<code>\glscapscase</code>	92, 93, 96–99,	<code>\glsentrydescaccess</code>	335
117–122, 135–142, 218, 219, 231, 236,		<code>\Glsentrydescplural</code>	128
336, 338, 340, 341, 343, 344, 346–348, 353		<code>\glsentrydescplural</code> .. 92, 93, 128, 336, 337	
<code>\glsclearpage</code>	38	<code>\glsentrydescpluralaccess</code>	335
<code>\glsclosebrace</code>	45, 157, 158, 312, 313	<code>\Glsentryfirst</code>	90, 94, 97, 124, 339, 342
<code>\glscompositor</code>	34, 44, 159, 311, 314	<code>\glsentryfirst</code> 89, 94, 95, 97, 124, 338, 339, 342	
<code>\glscounter</code>	15, 28, 40, 57, 79, 106, 308	<code>\glsentryfirstaccess</code>	335
<code>\glscurrententrylabel</code>	176, 179	<code>\Glsentryfirstplural</code> 91, 93, 96, 125, 337, 341	
<code>\glscurrentfieldvalue</code>	53, 54	<code>\glsentryfirstplural</code>	90,
<code>\glscustomtext</code>		92, 93, 96, 125, 126, 336, 337, 340, 341	
92, 95, 98, 100, 117–122, 135–142, 218,		<code>\glsentryfirstpluralaccess</code>	335
219, 226, 227, 230, 231, 233, 235, 236,		<code>\glsentryfmt</code>	56, 58
336, 339, 340, 342, 343, 345–348, 353, 354		<code>\Glsentryfull</code>	211, 220, 221, 355, 357
<code>\GlsDeclareNoHyperList</code>	15	<code>\glsentryfull</code>	211, 220, 221, 354, 357
<code>\glsdefaulttype</code>	12,	<code>\Glsentryfullpl</code> 211, 220, 221, 355, 357	
36, 47, 48, 55, 56, 76, 91, 101, 168, 177, 178		<code>\glsentryfullpl</code> 211, 220, 221, 355, 357	
<code>\glsdefmain</code>	12, 58	<code>\glsentryitem</code>	
<code>\glsdescriptionaccessdisplay</code> 191, 192, 259, 261–264, 266, 267,	
..... 338–340, 348, 349		275, 277, 278, 286, 288, 290, 293, 295,	
<code>\glsdescriptionpluralaccessdisplay</code>		297, 299, 301, 302, 305, 315–322, 324–327	
..... 336, 337		<code>\Glsentrylong</code>	90, 139, 144, 149,
<code>\glsdescwidth</code>	264–266, 269,	213, 214, 219, 220, 345, 347, 350, 353–355	
270, 272, 273, 275–280, 286–289, 291–298			

<code>\glstrylong</code>	89, 100, 139, 140, 144, 149, 213–223, 233, 345, 347–358
<code>\glstrylongaccess</code>	335
<code>\Glsstrylongpl</code>	91, 141, 149, 213, 214, 218–220, 345, 350, 353–355
<code>\glstrylongpl</code> 90, 100, 141, 142, 149, 213–215, 218–221, 233, 240, 345, 350, 351, 353–357
<code>\glstrylongpluralaccess</code>	335
<code>\Glsstryname</code>	126, 197, 348
<code>\glstryname</code>	126, 127, 303, 348
<code>\glstrynumberlist</code>	150, 165
<code>\Glsstryplural</code>	93, 96, 125, 337, 341
<code>\glstryplural</code> 92, 93, 96, 124, 125, 336, 337, 340, 341
<code>\glstrypluralaccess</code>	334
<code>\Glsstryprefix</code>	253
<code>\glstryprefix</code>	251, 254
<code>\Glsstryprefixfirst</code>	253
<code>\glstryprefixfirst</code>	252, 254
<code>\Glsstryprefixfirstplural</code>	254
<code>\glstryprefixfirstplural</code>	252, 255
<code>\Glsstryprefixplural</code>	253
<code>\glstryprefixplural</code>	252, 255
<code>\glstryprevcount</code>	87, 88
<code>\Glsstryshort</code>	98, 136, 144, 215, 221, 344–346, 350, 356, 357
<code>\glstryshort</code> 98–100, 135, 136, 144, 149, 211, 213–223, 343–346, 349–352, 354–358	
<code>\glstryshortaccess</code>	335
<code>\Glsstryshortpl</code> 98, 138, 215, 221, 343, 350, 356, 357
<code>\glstryshortpl</code> 98, 100, 137, 138, 149, 213– 215, 219–221, 240, 343, 345, 350, 354–357
<code>\glstryshortpluralaccess</code>	335
<code>\Glsstrysymbol</code>	129, 198, 349
<code>\glstrysymbol</code>	94, 95, 128, 129, 197, 227, 231, 236, 338–340, 349
<code>\glstrysymbolaccess</code>	335
<code>\Glsstrysymbolplural</code>	130
<code>\glstrysymbolplural</code> 92, 93, 129, 130, 227, 231, 236, 336, 337
<code>\glstrysymbolpluralaccess</code>	335
<code>\Glsstrytext</code>	94, 97, 123, 338, 342
<code>\glstrytext</code>	94, 95, 97, 123, 151, 176, 338, 339, 341, 342
<code>\glstrytextaccess</code>	334
<code>\glstrytype</code>	76
<code>\Glsstryuseri</code>	130
<code>\glstryuseri</code>	130, 131
<code>\Glsstryuserii</code>	131
<code>\glstryuserii</code>	131
<code>\Glsstryuseriii</code>	132
<code>\glstryuseriii</code>	132
<code>\Glsstryuseriv</code>	133
<code>\glstryuseriv</code>	133
<code>\Glsstryuserv</code>	134
<code>\glstryuserv</code>	133, 134
<code>\Glsstryuservi</code>	134
<code>\glstryuservi</code>	134, 135
<code>\glsfldfetch</code>	146
<code>\glsfirstaccessdisplay</code>	338, 339, 342
<code>\glsfirstpluralaccessdisplay</code> 336, 337, 340, 341
<code>\glsfirstpluralacessdisplay</code>	341
<code>\glsgenacfmt</code>	213, 214, 220, 349, 350, 355
<code>\glsgenentryfmt</code> 213, 214, 219, 220, 225, 227, 229– 231, 233, 235, 238, 240, 349, 350, 354, 355
<code>\glsgetgroupitle</code>	258, 262, 263, 281–285, 300–303, 306, 307
<code>\glsglossarymark</code>	36
<code>\glsgroupheading</code> 158, 188, 258, 261–264, 266, 267, 275, 277, 278, 281–286, 288, 290, 293, 295, 296, 299–304, 306, 307, 313	
<code>\glsgroupskip</code> 157, 188, 259, 261, 265, 266, 268, 271, 272, 275, 277, 279, 287, 288, 290, 293–295, 297, 300, 301, 303, 306, 312	
<code>\glshyperfirstfalse</code>	220, 355
<code>\glshyperfirsttrue</code>	23
<code>\glshyperlink</code>	176
<code>\glshypernavsep</code>	258
<code>\glshypernumber</code>	35, 204, 205
<code>\glshyperon</code>	103
<code>\glshyperon</code>	14
<code>\glshyperplural</code>	92, 96, 98, 99, 117–122, 135–142, 218, 227, 231, 233, 235, 336, 340, 343, 344, 346–348, 353
<code>\glshypertranslator</code>	21, 22, 31, 32, 366
<code>\glshyperonlyfirstfalse</code>	23
<code>\glshyperdescformat</code>	259, 315
<code>\glshyperdopostchild</code>	259, 315
<code>\glshyperemptydescformat</code>	259, 315
<code>\glshypernameformat</code>	259, 315
<code>\glshyperparentchildseparator</code> 259, 315	
<code>\glshyperpostchild</code>	259, 315
<code>\glshyperseparator</code>	259, 315

<code>\glsinlinesubdescformat</code>	259, 315	<code>\glsnoidxloclisthandler</code>	189
<code>\glsinlinesubnameformat</code>	259, 315	<code>\glsnoidxnumberlistloophandler</code>	166
<code>\glsinlinesubseparator</code>	259, 315	<code>\glsnoidxstripaccents</code>	18
<code>\glsinsert</code>	92–	<code>\glsnnextpages</code>	61, 178
99, 117–122, 135–142, 218, 219, 227,		<code>\glsnopostdotfalse</code>	8
230, 231, 233, 235, 236, 336–348, 353, 354		<code>\glsnoxindywarning</code> 34, 40, 42, 43, 45–47, 154	
<code>\glskeylisttok</code>	210,	<code>\glsnumberformat</code>	150
211, 225–232, 234, 236–239, 241, 358–363		<code>\glsnumberlistloop</code>	166
<code>\glslabel</code> ... 75, 92–99, 104–106, 136–142,		<code>\glsnumbersgroupname</code>	26, 32, 201
213, 214, 218–220, 226, 227, 230, 231,		<code>\glsnumlistlastsep</code>	150, 166
233, 235, 236, 336–345, 349, 350, 353–355		<code>\glsnumlistparser</code>	151, 163
<code>\glslabeltok</code>		<code>\glsnumlistsep</code>	150, 166
.... 210, 211, 225–234, 236–241, 359–362		<code>\glsopenbrace</code>	45, 157, 158, 312, 313
<code>\glslink</code>	211, 219, 221, 226, 354, 356	<code>\glsorder</code>	24, 161, 162, 184
<code>\glslink options</code>		<code>\glsorg@endtheglossary</code>	5
counter	102, 116, 248	<code>\glsorg@PrintChanges</code>	5
format	102, 116, 203	<code>\glsorg@theglossary</code>	5
hyper	102, 105, 116	<code>\glspagelistwidth</code> ... 266, 269, 270, 272,	
local	102	273, 276–280, 288, 289, 291, 292, 295–298	
<code>\glslinkcheckfirsthyperhook</code>	105	<code>\glspatchLTOoutput</code>	270–273
<code>\glslinkpostsetkeys</code>	106	<code>\glspenaltygroupskip</code>	271, 272
<code>\glslinkvar</code>	103	<code>\glspercentchar</code>	66, 67, 156, 158
<code>\glslistdottedwidth</code>	263, 316	<code>\Glspl</code>	91, 224
<code>\glslistgroupheaderfmt</code>	262, 263	<code>\glspl</code>	90, 224
<code>\glslistnavigationitem</code>	262, 263	<code>\glspluralaccessdisplay</code> 336, 337, 340, 341	
<code>\glslocalreset</code>	85	<code>\glspluralsuffix</code>	
<code>\glslocalunset</code>	86, 117–122	... 30, 78, 79, 213–215, 350, 351, 355–357	
<code>\glslongaccessdisplay</code>	345, 347–359	<code>\glspostdescription</code>	33,
<code>\glslongkey</code>	363	260–262, 264, 265, 275, 286, 287, 293,	
<code>\glslongpluralaccessdisplay</code>		299–303, 305, 306, 315–318, 320–324, 326	
.... 345, 350, 351, 353–357, 359		<code>\glspostinline</code>	258
<code>\glslongpluralkey</code>	363	<code>\glspostlinkhook</code>	
<code>\glslongtok</code> 104, 117–122, 135–142, 346–348	
.... 210, 211, 213, 214, 219, 220, 225–		<code>\glsprestandardsort</code>	10
234, 236–241, 349, 350, 354, 355, 358–363		<code>\glsreset</code>	85
<code>\glsLTpenaltycheck</code>	274	<code>\glsresetentrycounter</code>	191, 192, 194
<code>\glsmcols</code>	280–285	<code>\glsresetentrylist</code>	157, 186, 193, 312
<code>\glsnameaccessdisplay</code>	348, 349	<code>\glsresetsubentrycounter</code> 192, 195, 259, 315	
<code>\glsnamefont</code>	196–198, 328, 329, 348	<code>\glsanitizesortfalse</code>	20
<code>\glsnavhyperlink</code>	258	<code>\glsanitizesorttrue</code>	20
<code>\glsnavhypertarget</code>		<code>\glsavenumberlistfalse</code>	7
.... 262, 263, 281–285, 300, 302, 303, 307		<code>\glsavewritesfalse</code>	26
<code>\glsnavigation</code>		<code>\glsseeformat</code>	156, 164, 166, 312
.... 262, 263, 281–285, 300, 302, 303, 307		<code>\glsseeitem</code>	176
<code>\glsnextpages</code>	7, 61, 179	<code>\glsseeitemformat</code>	176
<code>\glsnogroupskipfalse</code>	8	<code>\glsseelastsep</code>	176
<code>\glsnoidxdisplayloc</code>	166, 167	<code>\glsseelist</code>	175
<code>\glsnoidxdisplayloclisthandler</code>	166	<code>\glsseesep</code>	176
<code>\glsnoidxloclist</code>	165, 188	<code>\glssetexpandfield</code>	17, 19, 20

<code>\ifdefstring</code> ..	31, 55, 161, 184, 185, 187, 189
<code>\ifdefvoid</code>	18, 80, 188
<code>\ifdim</code>	214, 274, 304
<code>\iffalse</code>	80, 85
<code>\IfFileExists</code>	7, 21, 22, 180
<code>\ifglossaryexists</code>	36, 47, 51, 160, 161
<code>\ifgl@sanitize@description</code>	19
<code>\ifgl@sanitize@name</code>	19
<code>\ifgl@sanitize@symbol</code>	19
<code>\ifgl@xindy@glslnumbers</code>	47
<code>\ifgl@acrdescription</code>	239
<code>\ifgl@acrdua</code>	229, 235, 238–240
<code>\ifgl@acrfootnote</code>	104, 239, 240
<code>\ifgl@acrronym</code>	12, 13
<code>\ifgl@acrshortcuts</code>	28, 224
<code>\ifgl@acrsmallcaps</code>	228, 230, 232, 234, 237, 238
<code>\ifgl@acrsmaller</code>	228, 230, 232, 235
<code>\ifgl@automake</code>	25, 163
<code>\ifgl@descsuppressed</code>	259
<code>\ifgl@entrycounter</code>	191, 192, 194, 195
<code>\ifgl@entryexists</code>	50, 51, 66, 75, 77
<code>\ifgl@haschildren</code>	259, 315
<code>\ifgl@hasdesc</code>	259
<code>\ifgl@haslong</code>	89–91, 144, 213, 214, 218, 220, 233, 349, 350, 353, 355
<code>\ifgl@hasparent</code>	182, 188, 303
<code>\ifgl@hasprefix</code>	253
<code>\ifgl@hasprefixfirst</code>	253
<code>\ifgl@hasprefixfirstplural</code>	253
<code>\ifgl@hasprefixplural</code>	253
<code>\ifgl@hassymbol</code>	227, 231, 235, 299–302, 305, 306
<code>\ifgl@hyperfirst</code>	104
<code>\ifgl@indexonlyfirst</code>	170
<code>\ifgl@nogroupskip</code>	261, 265, 266, 268, 270–272, 275, 277, 279, 287, 288, 290, 293, 295, 297, 300, 301, 303, 306
<code>\ifgl@nonumberlist</code>	192
<code>\ifgl@nopostdot</code>	8
<code>\ifgl@numberline</code>	39
<code>\ifgl@sanitizesort</code>	17, 20
<code>\ifgl@savenumberlist</code>	63, 163, 176
<code>\ifgl@savewrites</code>	25, 160, 169
<code>\ifgl@subentrycounter</code>	192, 194–196
<code>\ifgl@stoc</code>	39
<code>\ifgl@stranslate</code>	31
<code>\ifgl@sucmark</code>	37
<code>\ifgl@sused</code>	89, 92–98, 104, 152, 170, 226, 230, 233, 235, 251–255, 336–343
<code>\ifgl@swallowprimitivemods</code>	172
<code>\ifgl@xindy</code>	33, 34, 39–47, 57, 81, 82, 109, 153, 154, 161, 173, 174, 180, 308–310
<code>\ifignoredglossary</code>	77, 80, 170
<code>\ifin@</code>	27
<code>\ifinlistcs</code>	186, 190
<code>\ifKV@gl@link@hyper</code>	105, 106
<code>\ifKV@gl@link@local</code>	117–122
<code>\ifmeasuring@</code>	83
<code>\ifnum</code>	10, 87, 88, 187, 273, 274, 301, 302, 304, 305, 321, 322
<code>\ifstreempty</code>	315
<code>\ifstrequal</code>	200
<code>\ifthenelse</code>	20, 28, 38, 106, 162, 201, 240, 256
<code>\IfTrackedLanguageFileExists</code>	32, 366, 367
<code>\iftrue</code>	80, 84
<code>\ifundef</code>	56, 66, 76, 154, 158, 162, 192
<code>\ifvmode</code>	152
<code>\ifvoid</code>	274
<code>\ifx</code>	10, 11, 14, 27, 29, 40, 41, 43, 44, 47, 48, 77–80, 82, 107, 110–115, 144, 154, 157, 159, 168, 171, 172, 174, 175, 178, 191, 194, 201–204, 226, 228, 230, 232, 234, 235, 237, 239, 241, 242, 310–312, 314, 315, 320–323, 334
<code>\immediate</code>	66, 67, 89, 160, 168, 180
<code>\in@</code>	27
<code>\index</code>	169
<code>\indexname</code>	27
<code>\indexspace</code>	261, 281–285, 300–303, 306, 307
<code>\input</code>	30, 91
<code>\inputencodingname</code>	24
<code>\InputIfFileExists</code>	66
<code>\istfilename</code>	33, 154, 158, 161, 162, 310, 313
<code>\item</code>	261–264, 281, 299, 300, 315, 316, 320
J	
<code>\jobname</code>	33, 66, 154, 158, 160, 161, 179, 180, 310, 313
K	
<code>\key@ifundefined</code>	68–70
<code>\KV@gl@link@hyperfalse</code>	102, 104, 105, 116
<code>\KV@gl@link@hypertrue</code>	102, 116
L	
<code>\L</code>	19
<code>\l</code>	19

<code>\label</code>	6, 191, 192, 194, 195	<code>\markboth</code>	37
<code>\language</code>	24	<code>\mbox</code>	152, 262, 284, 285, 304, 316
<code>\leaders</code>	263, 316	memoir class	169
<code>\leavevmode</code>	75, 105	<code>\memUChead</code>	37
<code>\let</code>	5, 8, 10– 13, 18, 19, 21, 22, 25–28, 31, 33, 41, 42, 53–55, 63, 65, 66, 75–80, 83–86, 88, 91, 103–106, 108, 109, 115–122, 135–142, 144, 150, 151, 158–166, 169–172, 175, 176, 178, 179, 188, 190, 193, 198, 210, 223–225, 227–239, 249, 256–258, 273, 281, 299, 314, 331, 346–348, 359–363, 366	<code>\MessageBreak</code>	32, 55, 178, 366, 367
<code>\letcs</code>	51– 54, 69, 70, 73, 78, 79, 143, 144, 161, 165, 166, 181, 183, 184, 187, 188, 197, 200, 304	mfirstuc package	1
link text	91	<code>\mfirstucMakeUppercase</code>	4, 37, 71, 93, 95–99, 123– 136, 138, 140, 142, 211, 218–221, 231, 236, 254, 255, 341–345, 353, 354, 356, 357
<code>\listcsadd</code>	186	<code>\midrule</code>	270–272
<code>\listcsgadd</code>	190	<code>\month</code>	154, 158, 310, 313
<code>\listcsxadd</code>	181, 182	multicol package	280
<code>\listead</code>	185	N	
<code>\loadglsentries</code>	91	<code>\n</code>	158, 159, 313
<code>\long</code>	75, 204	<code>\NeedsTeXFormat</code> ...	4, 249, 308, 314, 328, 366
<code>\longnewglossaryentry</code>	75	<code>\new@glossaryentry</code>	66, 165
longtable package	264, 270, 274	<code>\new@ifnextchar</code>	55, 70, 71, 89, 90, 117–121, 123–142, 207–209, 251–254
<code>\LT@end@pen</code>	274	<code>\newacronym</code>	205, 210, 226, 228, 230, 232, 234, 237, 239, 241
<code>\LT@err</code>	274	<code>\newacronymhook</code>	211, 226, 228, 230, 232, 234, 237, 239, 241, 358
<code>\LT@foot</code>	274	<code>\newacronymstyle</code> ...	213–218, 220, 222, 223
<code>\LT@head</code>	274	<code>\newcommand</code>	5– 18, 20, 21, 23–31, 33–58, 60, 61, 63–75, 81–92, 95, 98, 100, 101, 103–106, 108, 109, 115–154, 159–162, 164, 167–179, 181–187, 189, 190, 193–214, 223, 225– 247, 250–254, 256–261, 273, 274, 280, 298, 299, 304, 314, 332–334, 349, 363–365
<code>\LT@lastfoot</code>	274	<code>\newcount</code>	10, 11, 63
<code>\LT@output</code>	273, 274	<code>\newcounter</code>	191, 192, 194
M		<code>\newenvironment</code>	196
<code>\makeatletter</code>	66, 179	<code>\newglossary</code>	12, 13, 26, 27, 57, 162
<code>\makeatother</code>	66	<code>\newglossaryentry</code>	27, 63, 66, 86, 211, 225, 227, 229, 231, 233, 236, 238, 240, 359–362
<code>\makebox</code>	263, 304–306, 316, 322, 323	<code>\newglossaryentry options</code>	
makeglossaries	24, 33, 47, 56, 162, 180	access	331, 332
<code>\makeglossaries</code>	25, 29, 61, 163, 165, 167, 181	counter	60
<code>\makeglossary</code>	160, 162	description	23, 59, 63, 65, 75, 127, 144, 206, 234, 330
makeindex	368	descriptionaccess	333, 335
makeindex	9, 24, 25, 30, 33–35, 55–57, 59, 82, 107, 111, 153, 156, 158, 160, 168, 173, 199, 309, 310	descriptionplural	127, 330
delim_n	35	descriptionpluralaccess	333, 335
delim_r	35	first	60, 78, 116, 123, 146, 232, 237, 329
page_compositor	34	firstaccess	333, 335
special characters	109, 110, 153	firstplural	60, 125, 146, 330
<code>\makenoidxglossaries</code>	61, 163, 167		
<code>\MakeTextUppercase</code>	4		
<code>\MakeUppercase</code>	337, 339, 346, 348		

firstpluralaccess	333, 335	206, 211, 225, 227, 229, 231, 233, 234,
format	154	236, 238, 240, 241, 308, 328, 329, 359–363
long	98, 149, 330	\nohyperpage 203
longaccess	334, 335	\noindent 198, 281–285, 301–303
longplural	149, 330	\noist 313, 314
longpluralaccess	334, 335	\nopostdesc 27, 33, 75, 179, 315
name	59, 63, 65, 75, 126, 143, 176, 329	\normalbaselineskip 273
nonumberlist	61	\nr 6, 21, 22, 61, 191
parent	61, 65	\ns@ACRfull 208
plural	59, 78, 124, 329	\ns@Acrfull 207
pluralaccess	333, 334	\ns@acrfull 207
prefix	249	\ns@ACRfullpl 209
prefixfirst	249	\ns@Acrfullpl 209
prefixfirstplural	250	\ns@acrfullpl 208
prefixplural	250	\ns@ACRlong 140
see	7, 61, 66, 162, 164	\ns@Acrlong 139
short	98, 148, 330	\ns@acrlong 138
shortaccess	333, 335	\ns@ACRlongpl 142
shortplural	148, 330	\ns@Acrlongpl 141
shortpluralaccess	333, 335	\ns@acrlongpl 140
sort	59, 147, 199	\ns@ACRshort 136
symbol	59, 60,	\ns@Acrshort 135
	128, 228, 230, 232, 237, 267, 289, 329–331	\ns@acrshort 135
symbolaccess	333, 335	\ns@ACRshortpl 138
symbolplural	129, 330	\ns@Acrshortpl 137
symbolpluralaccess	333, 335	\ns@acrshortpl 137
text	59, 60, 116, 122, 145, 228, 232, 329	\ns@newglossary 56
textaccess	332, 334	\null 108–115, 180
type	12, 60, 91, 147	\number 10, 78, 88, 171, 172, 198, 329
user1	130, 147, 330	\numberline 39
user2	131, 147	\numexpr 88
user3	131, 147	
user4	132, 148	
user5	133, 148	
user6	134, 148, 330	
\newglossarystyle	258, 261–273, 275–304, 306	
\newif	4, 14, 21, 24, 171	
\newlength	115, 263, 264, 275, 286, 293, 302	
\newrobustcmd		
	65, 66, 70, 71, 89, 90, 104, 116–121,	
	123–149, 151, 152, 206–209, 250–254, 303	
\newterm	27	
\newtoks	109, 160, 210	
\newwrite	66, 154, 158, 160, 162	
\noalign	273	
\nobreak	262, 274, 316	
\noexpand	14, 29, 40, 41, 79, 80,	
	101, 106–109, 114, 115, 150, 161, 163,	
	171, 173, 176, 180, 183, 184, 196, 198,	

O	
\O	19
\o	19
\OE	19
\oe	19
\openout	66, 154, 158, 160, 310, 313
\OR	240
\or	6, 22, 191, 299, 320
\org@glossaryentrynumbers	178, 193
\org@glossarytitle	178
\org@glspostdescription	33
\org@ifKV@glslink@hyper	105, 106
\orgAlph	172
\orgalph	172
\orgarabic	172
\orgnumber	172
\orgRoman	172

O

\O	19
\o	19
\OE	19
\oe	19
\openout	66, 154, 158, 160, 310, 313
\OR	240
\or	6, 22, 191, 299, 320
\org@glossaryentrynumbers	178, 193
\org@glossarytitle	178
\org@glspostdescription	33
\org@ifKV@glslink@hyper	105, 106
\orgAlpha	172
\orgalph	172
\orgarabic	172
\orgnumber	172
\orgRoman	172

<code>\orgromannumeral</code>	172	162, 165, 167, 184–186, 190, 193, 201,
<code>\orgthe</code>	172	202, 212, 213, 229, 230, 235, 238, 314, 336
<code>\outputpenalty</code>	273, 274	
P		
<code>\p@</code>	260, 280, 298	
<code>\p@glshyp@opt</code>	103	
package options:		
acronym	12, 13, 29, 177, 206	
true	13	
counter	15	
description	232, 233	
dua	230, 232, 233	
entrycounter	192, 194	
true	9	
footnote	117–122, 228, 230, 232, 234	
hyperfirst		
false	117–122	
indexonlyfirst	375	
makeindex	156, 248	
nogroupskip	265, 266, 268, 270–272,	
	275, 277, 279, 287, 288, 290, 293, 295, 297	
nolist	241	
nolong	242, 264	
nomain	12	
nonumberlist	7	
nosuper	242	
notree	242	
numberline	5	
sanitize	19, 59, 143, 144	
sanitizesort	16	
savewrites	25, 372	
false	160	
true	162, 168	
section	5, 37	
sort		
def	9, 10	
standard	9	
use	9, 10	
style	6, 241, 242	
subentrycounter	192, 194	
toc	5	
true	5	
translate	21	
false	21	
translator	21	
xindy	24, 25, 156, 248	
<code>\PackageError</code> 25, 29, 40, 47, 50, 51, 55, 60,		
61, 63, 69–74, 76–78, 86, 102, 143, 160,		
162, 165, 167, 184–186, 190, 193, 201,		
202, 212, 213, 229, 230, 235, 238, 314, 336		
<code>\PackageInfo</code>	160	
<code>\PackageWarning</code>	15, 328	
<code>\PackageWarningNoLine</code>	16, 32, 366, 367	
<code>\pagegoal</code>	274	
<code>\pagelistname</code>	32, 267–	
	269, 271, 272, 278–280, 289–292, 296–298	
<code>\par</code>	33, 198, 199, 260, 262, 280,	
	282–285, 298, 299, 301–307, 316, 321–323	
<code>\parindent</code> 280–285, 299–302, 304–307, 321–323		
<code>\parskip</code>	280–284, 299, 300, 302	
<code>\PassOptionsToPackage</code>	249, 328	
<code>\penalty</code>	273	
<code>\phantomsection</code>	38	
polyglossia package	21, 31	
<code>\printglossaries</code>	163	
<code>\printglossary</code> ..	13, 16, 26, 27, 163, 177, 193	
<code>\printglossary options</code>		
entrycounter	191	
nogroupskip	191	
nonumberlist	192	
nopostdot	191	
numberedsection	191	
style	190	
subentrycounter	192	
title	190	
toctitle	190	
type	12, 176, 190	
<code>\printindex</code>	27	
<code>\printnoidxglossaries</code>	165	
<code>\printnoidxglossary</code>		
	164, 165, 167, 177, 184, 185, 193	
<code>\printnoidxglossary options</code>		
sort	193	
<code>\printnumbers</code>	26	
<code>\printsymbols</code>	26	
<code>\ProcessOptions</code>	249, 328	
<code>\ProcessOptionsX</code>	27	
<code>\protect</code>	39, 100, 213–215, 220,	
	221, 227, 231, 233, 345, 349, 350, 355, 356	
<code>\protected@edef</code>	6, 41, 43,	
	46, 48, 77, 80, 82, 92–94, 100, 107, 150,	
	169, 172, 191, 196, 198, 201, 202, 211,	
	235, 240, 250, 256, 309, 310, 328, 329, 334	
<code>\protected@write</code>	55, 56,	
	154–156, 162, 164, 167, 170, 176, 256, 310	
<code>\protected@xdef</code>		
	10, 11, 14, 18, 64, 82, 83, 172, 331, 332	

<code>\providecommand</code>	13, 29, 30, 37, 55, 89, 116, 156, 162, 164, 180, 196, 198, 260, 280, 298
<code>\ProvidesFile</code>	30
<code>\ProvidesPackage</code> 4, 249, 256, 258, 260, 264, 270, 274, 280, 286, 292, 298, 308, 314, 328, 366
R	
<code>\r</code>	18
<code>\raggedright</code> 272, 273, 275–280, 293–298
<code>\raisebox</code>	115
<code>\ref</code>	195
<code>\refstepcounter</code> 191, 192, 194, 195
<code>\relax</code>	6, 8, 11–13, 21, 22, 28, 42, 55, 60, 61, 64, 77, 78, 80, 87, 88, 103, 104, 107, 108, 110–114, 144, 157–160, 162–166, 170, 172, 173, 175, 178, 179, 187, 188, 190, 191, 200–202, 242, 257, 260, 273, 280, 284, 285, 298, 299, 301–307, 309, 312– 314, 320–323, 331, 346, 347, 359–361, 363
<code>\renewacronymstyle</code>	. 349–353, 355, 357, 358
<code>\renewcommand</code> 4–9, 12, 13, 15, 16, 20, 22, 23, 25, 28, 31–35, 47, 58, 75, 86–88, 150, 152–154, 162–166, 169, 179, 181, 191, 192, 210, 211, 213– 223, 226, 228, 230, 232, 234, 235, 237, 239, 241, 258, 259, 261–272, 274–290, 293–297, 299–309, 314–322, 324–327, 331, 336, 340, 343, 345, 348–352, 354–362
<code>\renewenvironment</code> 196, 258, 261, 264–270, 272, 273, 275–300, 302, 304
<code>\RequireGlossariesLang</code> 32, 366, 367
<code>\RequirePackage</code> 4, 7, 8, 21, 22, 27, 31, 241, 248, 249, 264, 270, 274, 280, 286, 292, 329, 366
<code>\restorecounters@</code>	107
<code>\romannumeral</code> 171, 172, 304, 305, 322
S	
<code>\s@gls@hyp@opt</code>	103
<code>\s@newglossary</code>	56
<code>\savecounters@</code>	107
<code>\seename</code>	175
<code>\SetAcronymStyle</code>	23
<code>\setbool</code>	20
<code>\setbox</code>	274
<code>\setcounter</code> 191, 192, 194
<code>\SetCustomDisplayStyle</code>	241
<code>\SetDefaultAcronymDisplayStyle</code> 226
<code>\SetDefaultAcronymStyle</code>	239
<code>\SetDescriptionAcronymDisplayStyle</code>	232
<code>\SetDescriptionAcronymStyle</code> 239
<code>\SetDescriptionDUAAcronymDisplayStyle</code> 230
<code>\SetDescriptionDUAAcronymStyle</code> 239
<code>\SetDescriptionFootnoteAcronymDisplayStyle</code> 228
<code>\SetDescriptionFootnoteAcronymStyle</code>	239
<code>\SetDUADisplayStyle</code>	239
<code>\SetDUASStyle</code>	240
<code>\setentrycounter</code> 41, 156, 189, 308
<code>\SetFootnoteAcronymDisplayStyle</code>	... 234
<code>\SetFootnoteAcronymStyle</code> 240
<code>\SetGenericNewAcronym</code> 212
<code>\setglossarystyle</code> 178, 201, 242, 261–273, 276– 285, 287–292, 294–298, 300, 301, 303, 306
<code>\setglossentrycompatibility</code>	190, 201, 202
<code>\setkeys</code>	.. 20, 25, 28, 37, 76, 106, 152, 178, 210, 226, 228, 230, 232, 234, 237, 239, 241
<code>\setlength</code> 263, 264, 275, 280–284, 286, 293, 299, 300, 302, 305, 306, 322, 323
<code>\SetSmallAcronymDisplayStyle</code> 237
<code>\SetSmallAcronymStyle</code> 240
<code>\settoheight</code>	115
<code>\settowidth</code> 214, 303–305, 322
<code>\sfcode</code> 8
<code>\show</code> 242–247, 364, 365
<code>\SmallNewAcronymDef</code> 237
<code>\space</code>	25, 29, 40, 41, 45, 47, 48, 61, 63, 86, 89– 91, 100, 101, 103, 150, 155–158, 161– 163, 165, 167, 176, 178, 181, 191, 192, 195, 201, 207, 213–223, 231, 235, 236, 258, 260–262, 264, 265, 275, 286, 287, 293, 299–306, 308, 309, 311–313, 315– 318, 320–324, 326, 345, 349–352, 354–359
<code>\spacefactor</code> 8
<code>\SS</code> 19
<code>\ss</code> 19
<code>\string</code>	16, 25, 29, 39–41, 43–48, 55, 56, 61, 63, 66, 67, 70, 71, 81, 82, 86, 89–91, 101, 103, 108, 110–112, 114, 150, 153–160, 162–165, 167, 173–175, 178, 180, 181, 184, 185, 193, 198, 199, 201, 256, 308–314
<code>\strut</code> 199, 261–263, 265, 266, 268, 275, 277, 278, 287, 288, 290, 293, 295, 297, 302, 315–319, 321, 324–327
<code>\subglossentry</code> 81, 179, 188, 198, 199, 259, 261–

263, 265, 266, 268, 275, 277, 278, 286, 288, 290, 293, 295, 297, 299, 301, 302, 305	\toprule 270, 271
\subitem 299, 320	tracklang package 31, 366
\subsubitem 300, 320	\trans@languages 31
supertabular package 7, 242, 286, 292	\translate 31, 32
\symbolname 32, 268, 269, 271, 279, 280, 290–292, 297, 298	\translatelet 12, 13, 26, 27
	translator package . 12, 13, 21, 26, 27, 31, 32, 177
	U
T	\u 18
\t 18	\uccode 187
\tablehead 286–298	\undef 177
\tabletail 286–298	\unskip 75, 263, 316
\tabularnewline 264– 272, 275–280, 286–298, 318, 319, 324, 325	\unvbox 274
\texorpdfstring 146	\usedictionary 31
\textbar 258	\usepackage 184, 185
\textbf 198, 205, 298, 320–323	
textcase package 4	V
\textit 205	\v 18
\textmd 205	\val 6, 21, 61, 191
\textrm 204	\vbox 274
\textsc 205, 215, 216, 222, 228, 232, 235, 237, 357	\vsize 274
\textsf 204	\vskip 260, 273, 280, 298
\textsl 205	\vss 274
\textsmaller 215, 216, 222, 228, 232, 235, 237, 357	
\texttt 205	W
\textulc 206	\warn@nomakeglossaries 163–165
\textup 205, 206	\warn@noprintglossary 163–165, 179
\th 19	\write 66, 67, 89, 154–159, 161, 164, 168, 180, 310–314
\the 29, 31, 41, 46, 48, 56, 110–115, 154, 158, 168, 169, 172, 176, 187, 197, 198, 204, 211, 213, 214, 219, 220, 225, 227, 229, 231, 233, 234, 236, 238, 240, 241, 308, 310, 313, 328, 329, 349, 350, 354, 355, 358–363	\writeist 160, 314
\the@numberlist 150, 151	
\theglossary 5	X
\theglossaryentry 191, 194, 195	\x 204
\theglossarysubentry 192, 194, 195	\xatlevel@ 107
\theglentrycounter 106, 107, 172, 309, 310	\xcapitalisewords 146
\theH 174	\xdef 71, 77, 78, 80, 179, 257
\theHglossaryentry 191, 194	\xglaccsupp 334
\theHglossarysubentry 192, 194	\xifinlistcs 181, 182, 185
\theHglentrycounter 107, 172	xindy 368
\thesection 28	xindy 9, 24, 25, 33, 34, 39, 42, 44, 46–48, 82, 114, 153–155, 168, 173, 180, 199, 248, 309
\this@dialect 32, 366, 367	\xmakefirstuc 93, 94, 100, 143, 144, 250
\toks@ 29, 31, 41, 46, 48, 56, 110–115, 176, 196–198, 204, 308, 328, 329	\xspace 206
	xspace package 4, 205
	Y
	\year 154, 158, 310, 313
	Z
	\z@ 274