

Documented Code For glossaries v4.13

Nicola L.C. Talbot

Dickimaw Books

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

2015-02-03

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 v4.13: 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	30
1.4 Xindy	40
1.5 Loops and conditionals	49
1.6 Defining new glossaries	55
1.7 Defining new entries	59
1.8 Resetting and unsetting entry flags	80
1.9 Loading files containing glossary entries	82
1.10 Using glossary entries in the text	82
1.10.1 Links to glossary entries	93
1.10.2 Displaying entry details without adding information to the glossary	134
1.11 Adding an entry to the glossary without generating text	142
1.12 Creating associated files	144
1.13 Writing information to associated files	159
1.14 Glossary Entry Cross-References	165
1.15 Displaying the glossary	167
1.16 Acronyms	196
1.17 Predefined acronym styles	201
1.18 Predefined Glossary Styles	232
1.19 Debugging Commands	232
1.20 Compatibility with version 2.07 and below	238
2 Prefix Support (glossaries-prefix Code)	238
3 Mfirstuc Documented Code	244
4 Mfirstuc-english Documented Code	247
5 Glossary Styles	248
5.1 Glossary hyper-navigation definitions (glossary-hypernav package)	248
5.2 In-line Style (glossary-inline.sty)	250
5.3 List Style (glossary-list.sty)	253
5.4 Glossary Styles using longtable (the glossary-long package)	256
5.5 Glossary Styles using longtable (the glossary-longragged package)	262
5.6 Glossary Styles using multicol (glossary-mcols.sty)	267
5.7 Glossary Styles using supertabular environment (glossary-super package)	271
5.8 Glossary Styles using supertabular environment (glossary-superragged package)	278
5.9 Tree Styles (glossary-tree.sty)	284

6 glossaries-compatible-207	292
7 Accessibility Support (glossaries-accsupp Code)	312
7.1 Defining Replacement Text	313
7.2 Accessing Replacement Text	316
7.3 Displaying the Glossary	331
7.4 Acronyms	333
7.5 Debugging Commands	347
8 Multi-Lingual Support	348
8.1 Polyglossia Captions	349
Glossary	350
Change History	350
Index	373

1 Main Package Code

1.1 Package Definition

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

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries}[2015/02/03 v4.13 (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.

```

\if@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.

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

sorg@endtheglossary
22   \let\glsorg@endtheglossary\endtheglossary

\PrintChanges Now redefine \PrintChanges so that it uses the original theglossary environ-
ment.
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 redefined `\glossarysection`.

```
37 \define@choicekey{glossaries.sty}{section}{part,chapter,section,%
38 subsection,subsubsection,paragraph,subparagraph}[section]{%
39   \renewcommand*\@@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 [subsection 1.18](#).)

`ssary@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 [subsection 1.18](#).

```
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.

`\@gls@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”:

`lossaryentrynumbers`

```
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
```

o@seeautonumberlist

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

seeautonumberlist Automatically activates number list for entries containing the see key.

```
77 \@gls@declareoption{seeautonumberlist}{%
78   \renewcommand*{\@gls@seeautonumberlist}{%
79     \def\@gls@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 }

```

`\glspostdescription` 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

```

`entrycounterwithin` 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 }

```

`\@gls@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
```

`lo@default@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 }
```

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

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

```
128 \newcommand*\@glsprestandardsort[3]{%
129   \glsdosanitizesort
130 }
```

`@setupsort@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 argu-
  ment 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
```

`\glssortnumberfmt` 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 }
```

`\@gls@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 }
```

`\@gls@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 [subsection 1.9](#)).

`\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 [subsection 1.16](#) 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}%
217     }%
218   }%
219 }
```

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 }

```

`@gls@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{}

```

`@addtoacronymlists`

```

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

```

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

```

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

```

`\glsIfListOfAcronyms` `\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 }

```

`\if@glsisacronymlist` Convenient boolean.

```

273 \newif\if@glsisacronymlist

```

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

```

274 \newcommand*\@gls@checkisacronymlist}[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 [subsection 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}{}%
```

`sDeclareNoHyperList`

```
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 }
```

`\GlossariesWarning` Prints a warning message.

```
301 \newcommand*{\GlossariesWarning}[1]{%
```

```
302   \PackageWarning{glossaries}{#1}%
```

```
303 }
```

`sariesWarningNoLine` 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 }
```



```

@warnonglossdefined Issue a warning if overriding \printglossary
311 \newcommand*{\@gls@warnonglossdefined}{%
312   \GlossariesWarning{Overriding \string\printglossary}%
313 }

warnontheGLOSSdefined Issue a warning if overriding theGLOSSary
314 \newcommand*{\@gls@warnontheGLOSSdefined}{%
315   \GlossariesWarning{Overriding 'theGLOSSary' environment}%
316 }

norededefwarn Suppress warning on redefinition of \printglossary
317 \@gls@declareoption{norededefwarn}{%
318   \renewcommand*{\@gls@warnonglossdefined}{}%
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.

```

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

```

```

\glssetexpandfield \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 }

```

```

\glssetnoexpandfield \glssetnoexpandfield{<field>}

```

Sets field to never expand.

```

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

```

s@assign@type@field The type must always be expandable.

```

333 \glssetexpandfield{type}

```

s@assign@desc@field The description is not expanded by default:

```
334 \glssetnoexpandfield{desc}
```

gn@descplural@field

```
335 \glssetnoexpandfield{descplural}
```

\@gls@sanitizename

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

s@assign@name@field Don't expand name by default.

```
337 \glssetnoexpandfield{name}
```

@gls@sanitizesymbol

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

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

```
339 \glssetnoexpandfield{symbol}
```

@symbolplural@field

```
340 \glssetnoexpandfield{symbolplural}
```

Sanitizing stuff:

\@gls@sanitizesort

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

\@@gls@sanitizesort

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

@gls@nosanitizesort

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

@noidx@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 }

```

oidx@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 }

```

lsnoidxstripaccents

```

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     \glSsetnoexpandfield{sortvalue}%
439     \renewcommand*{\@glS@noidx@setsanitizesort}{%
440       \glSSanitizesorttrue
441       \glSsetnoexpandfield{sortvalue}%
442     }%
443   \else
444     \glSsetexpandfield{sortvalue}%
445     \renewcommand*{\@glS@noidx@setsanitizesort}{%

```

```

446      \glssanitizesortfalse
447      \glsssetexpandfield{sortvalue}%
448    }%
449    \fi
450 }

```

Default setting:

```

451 \glssanitizesorttrue
452 \glsssetnoexpandfield{sortvalue}%

```

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

```

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

457 \define@choicekey{gls}{sanitimize}{sort}{true,false}[true]{%
458   \setbool{glssanitizesort}{#1}%
459   \ifglssanitizesort
460     \glsssetnoexpandfield{sortvalue}%
461   \else
462     \glsssetexpandfield{sortvalue}%
463   \fi
464   \GlossariesWarning{sanitimize={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

```

`\ls@nottranslatorhook` `\@gls@nottranslatorhook` has been removed.

\@gls@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 }
```

fusedtranslatordict 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 }
```

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

```
505 \@gls@declareoption{nottranslate}{%
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   }
```

```

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 defined 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}}
```

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

```
595 \newif\ifglxindy
```

The default is makeindex:

```
596 \glxindyfalse
```

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

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

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

```
598 \define@boolkey[glx]{xindy}{glsnumbers}[true]{}
```

```
599 \gls{xindy@glsnumberstrue}
```

`\@xdy@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[glx]{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[glx]{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 \glxindytrue
```

```
609 \setkeys[glx]{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=glsnumbers=false that can be passed via the document class options.

```
614 \@gls@declareoption{xindynoglsnumbers}{%
615   \glsxindytrue
616   \gls@xindy@glsnumbersfalse
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*{\glswritefiles}{\@glswritefiles}%
638   \else
639     \let\glswritefiles\@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][]{%
684     \newglossaryentry{##2}%
685     {type={index},name={##2},description={\nopostdesc},##1}}
686 }%
```

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@ption
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. $\langle n \rangle . 0$ target, otherwise entries placed before the first section of a chapter will have undefined links.

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

```

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 {}

```

`\@gls@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 }
```

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

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

`\@disable@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}
```

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

```
757 \newcommand*{\glssettoctitle}[1]{%
758   \def\glossarytoctitle{\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:

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

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

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

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

`\glsupacrpluralsuffix`
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.

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

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

dglossarytocaptions 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*{\glstttitle}[1]{%
828     \ifcsdef{gls@tr@set@#1@ttitle}%
829     {%
830       \csuse{gls@tr@set@#1@ttitle}%
831     }%
832     {%
833       \def\glossaryttitle{\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*{\gl@istfilebase}{#1}%
  Just in case \istfilename has been modified.
860   \ifglsxindy
861     \def\istfilename{\gl@istfilebase.xdy}
862   \else
863     \def\istfilename{\gl@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 re-defining 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{\gl@istfilebase.xdy}
869 \else
870   \def\istfilename{\gl@istfilebase.ist}
871 \fi
```

`\gl@istfilebase`

```
872 \newcommand*{\gl@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}
```

`\glsSetAlphaCompositor` Sets the alpha compositor.

```
879 \ifglsxindy
880   \newcommand*\glsSetAlphaCompositor[1]{%
881     \renewcommand*\@glsAlphacompositor{#1}}
882 \else
883   \newcommand*\glsSetAlphaCompositor[1]{%
884     \glsnoxywarning\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`.

```

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

```

```

\setglossarypreamble \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 `theglossary` environment (again, this shouldn't be affected by the glossary style). It is, of course, possible to simply add the text after `\printglossary`, but if you only want the postamble to appear after the first glossary, but not after subsequent glossaries, you can do something like:

```

\renewcommand{\glossarypostamble}{For a complete list of terms
see \cite{blah}\gdef@glossarypreamble{}}

```

```

\glossarypostamble
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{\memUHead{#1}}{\memUHead{#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.

`\setglossarysection`

```

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.

`\@p@glossarysection`

```

973 \newcommand*{\@p@glossarysection}[2]{%
974   \gls@clearpage
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@doclearpage` The `\gls@doclearpage` command is used to issue a `\clearpage` (or `\cleardoublepage`) depending on whether the glossary sectional unit is a chapter. If the sectional unit is something else, do nothing.

```

986 \newcommand*{\gls@doclearpage}{%
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.

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

```

1019 \newcommand*{\glsnoxindywarning}[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 \ifglxsindy
1023   \edef\@xdyattributes{\string"default\string"}%
1024 \fi

```

\@xdyattributelist Comma-separated list of attributes.

```

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

```

\@xdylocref Define list of markup location references.

```

1028 \ifglxsindy
1029   \def\@xdylocref{}
1030 \fi

```

\@gls@ifinlist

```

1031 \newcommand*\@gls@ifinlist[4]{%
1032   \def\@do@ifinlist##1,#1,##2\end@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@ifinlist
1041 }

```

\GlsAddXdyCounters 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 \ifglxsindy
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

```

`\d@glssaddxdycounters` 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 }

```

`\GlsAddXdyAttribute` Adds an attribute.

```

1068 \ifglsxindy

```

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

```

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\@glsaddxdyattribute{#1}{\@this@counter}%
1093 }
1094 \gls@do@addxdyattribute
1095 }%

```

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

```

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

```

Only has an effect before `\writeist`:

```

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

```

`redefinedattributes` Add known attributes for all defined counters

```

1103 \ifglxsindy
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 \ifglsxindy
1134   \newcommand*{\GlsAddXdyAlphabet}[2]{%
1135     \edef\@xdyuseralphabets{%
1136       \@xdyuseralphabets ^^J
1137       (define-alphabet "#1" (#2))}%
1138   \else
1139     \newcommand*{\GlsAddXdyAlphabet}[2]{%
1140       \glsnnoxindywarning\GlsAddXdyAlphabet}
1141   \fi
```

This code is only required for xindy:

```
1142 \ifglsxindy
```

`ls@xdy@locationlist` List of predefined location names.

```
1143   \newcommand*{\@glx@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 `\@glx@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\@glx@roman{\@roman{0}\string"
1153     \string"roman-numbers-lowercase\string" :sep \string"}}%
1154   \@onelevel@sanitize\@glx@roman
1155   \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
1156     :sep \string"}%
1157   \@onelevel@sanitize\@tmp
1158   \ifx\@tmp\@glx@roman
1159     \expandafter
1160       \edef\csname @glx@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 }%

pendix-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 }

bic-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 }%

xdyuserlocationdefs List of additional location definitions (separated by ^^J)
1191 \def\@xdyuserlocationdefs{}

dyuserlocationnames 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 \ifglxindy
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     \glsnxindywarning\GlsAddXdyLocation}
1222 \fi
```

ylocationclassorder 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.

yLocationClassOrder

```

1236 \ifglxindy
1237   \newcommand*\GlsSetXdyLocationClassOrder[1]{%
1238     \def\@xdylocationclassorder{#1}}
1239 \else
1240   \newcommand*\GlsSetXdyLocationClassOrder[1]{%
1241     \glsnnoxindywarning\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     \glsnnoxindywarning\GlsAddSortRule}
1255 \fi

```

\@xdyrequiredstyles 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     \glsnnoxindywarning\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     \glsnxindywarning\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{}
```

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

```
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     \glsnxindywarning\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{}
```

`\GlsSetXdyCodePage` 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 \ifglx@xindy@glslnumbers
1303 \def\@xdylettergroups{(define-letter-group
1304 \string\glslnumbers\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"@glslfirstletter\string"))}
1310 \else
1311 \def\@xdylettergroups{}
1312 \fi
1313 \fi

```

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

```

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\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 }
```

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

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

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

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

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

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

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

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

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

```
1338 \newcommand\ifglossaryexists[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 redefining `\glsdetoklabel` can cause things to go badly wrong, I'm not going to mention it in the main user guide. Only advanced users who know what they're doing ought to attempt it.

`\glsdetoklabel`

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

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

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

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

```
1342 \newcommand{\ifglsentryexists}[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   \ifglstryexists{#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   \ifglstryexists{#1}{#2}{%
1361     \GlossariesWarning{Glossary entry ‘\glsdetoklabel{#1}’
1362       has not been defined}%
1363   }%
1364 }

```

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

```

1365 \newcommand{\ifglshaschildren}[3]{%
1366   \glsdoifexists{#1}%
1367   {%
1368     \def\do@glshaschildren{#3}%
1369     \edef\@gls@thislabel{\glsdetoklabel{#1}}%
1370     \expandafter\forglstryentries\expandafter
1371     [\csname glo@\@gls@thislabel @type\endcsname]
1372     {\glo@label}%
1373     {%
1374       \letcs\glo@parent{glo@\glo@label @parent}%
1375       \ifdefequal\@gls@thislabel\glo@parent
1376       {%
1377         \def\do@glshaschildren{#2}%
1378         \@endfortrue
1379       }%
1380     }%
1381   }%
1382   \do@glshaschildren
1383 }%
1384 }

```

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

```

1385 \newcommand{\ifglshasparent}[3]{%
1386   \glsdoifexists{#1}%
1387   {%
1388     \ifcsempy{glo@\glsdetoklabel{#1}@parent}{#3}{#2}%

```

```

1389 }%
1390 }

\ifglshasdesc \ifglshasdesc{<label>}{<true part>}{<false part>}
1391 \newcommand*{\ifglshasdesc}[3]{%
1392   \ifcsequal{glo@\glsdetoklabel{#1}@desc}%
1393   {#3}%
1394   {#2}%
1395 }

ifglstdescsuppressed \ifglstdescsuppressed{<label>}{<true part>}{<false part>} Does <true part>
if the description is just \nopostdesc otherwise does <false part>.
1396 \newcommand*{\ifglstdescsuppressed}[3]{%
1397   \ifcsequal{glo@\glsdetoklabel{#1}@desc}{@no@post@desc}%
1398   {#2}%
1399   {#3}%
1400 }

\ifglshassymbol \ifglshassymbol{<label>}{<true part>}{<false part>}
1401 \newcommand*{\ifglshassymbol}[3]{%
1402   \letcs{@glo@symbol}{glo@\glsdetoklabel{#1}@symbol}%
1403   \ifdefempty@glo@symbol
1404   {#3}%
1405   {%
1406     \ifdefequal@glo@symbol@gls@default@value
1407     {#3}%
1408     {#2}%
1409   }%
1410 }

\ifglshaslong \ifglshaslong{<label>}{<true part>}{<false part>}
1411 \newcommand*{\ifglshaslong}[3]{%
1412   \letcs{@glo@long}{glo@\glsdetoklabel{#1}@long}%
1413   \ifdefempty@glo@long
1414   {#3}%
1415   {%
1416     \ifdefequal@glo@long@gls@default@value
1417     {#3}%
1418     {#2}%
1419   }%
1420 }

\ifglshasshort \ifglshasshort{<label>}{<true part>}{<false part>}
1421 \newcommand*{\ifglshasshort}[3]{%
1422   \letcs{@glo@short}{glo@\glsdetoklabel{#1}@short}%
1423   \ifdefempty@glo@short
1424   {#3}%
1425   {%

```

```

1426 \ifdefequal\@glo@short\@gls@default@value
1427 {#3}%
1428 {#2}%
1429 }%
1430 }

```

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

```

1431 \newcommand*{\ifglshasfield}[4]{%
1432 \glsoifexists{#2}%
1433 {%
1434 \letcs{\@glo@thisvalue}{gls@detoklabel{#2}@#1}%

```

First check supplied field label is defined.

```

1435 \ifdef\@glo@thisvalue
1436 {%

```

Is defined, so now check if empty.

```

1437 \ifdefempty\@glo@thisvalue
1438 {%

```

Is empty, so doesn't have field set.

```

1439 #4%
1440 }%
1441 {%

```

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

```

1442 \ifdefequal\@glo@thisvalue\@gls@default@value{#4}{#3}%
1443 }%
1444 }%
1445 {%

```

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

```

1446 \@gls@fetchfield{\@gls@thisfield}{#1}%

```

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

```

1447 \ifdef\@gls@thisfield
1448 {%

```

Is defined, so now check if empty.

```

1449 \letcs{\@glo@thisvalue}{gls@detoklabel{#2}@\@gls@thisfield}%
1450 \ifdefempty\@glo@thisvalue
1451 {%

```

Is empty so field hasn't been set.

```

1452 #4%
1453 }%
1454 {%

```

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

```
1455         \ifdefequal\@glo@thisvalue\@gls@default@value{#4}{#3}%
1456     }%
1457 }%
1458 {%
```

Not defined.

```
1459     \GlossariesWarning{Unknown entry field '#1'}%
1460     #4%
1461 }%
1462 }%
1463 }%
1464 }
```

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

```
1465 \newcommand*{\@glo@types}{,}
```

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

```
1466 \newcommand*\@gls@provide@newglossary{%
1467     \protected@write\@auxout{}\string\providecommand\string\@newglossary[4]{}%
```

Only need to do this once.

```
1468     \let\@gls@provide@newglossary\relax
1469 }
```

\defglsentryfmt Allow different glossaries to have different display styles.

```
1470 \newcommand*\defglsentryfmt[2][\glsdefaulttype]{%
1471     \csgdef{gls@#1@entryfmt}{#2}%
1472 }
```

\gls@doentryfmt

```
1473 \newcommand*\gls@doentryfmt[1]{\csuse{gls@#1@entryfmt}}
```

\@gls@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.

```
1474 \newcommand*\@gls@forbidtexext[1]{%
1475     \ifboolexpr{test {\ifdefstring{#1}{tex}}}
```

```

1476         or test {\ifdefstring{#1}{TEX}}
1477 {%
1478   \def#1{nottex}%
1479   \PackageError{glossaries}%
1480     {Forbidden '.tex' extension replaced with '.nottex'}%
1481     {I'm sorry, I can't allow you to do something so reckless.\MessageBreak
1482       Don't use '.tex' as an extension for a temporary file.}%
1483 }%
1484 {%
1485 }%
1486 }

```

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`

```

1487 \newcommand*{\newglossary}{\@ifstar\s@newglossary\@ns@newglossary}

```

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

```

1488 \newcommand*{\s@newglossary}[2]{%
1489   \ns@newglossary[#1-glg]{#1}{#1-gls}{#1-glo}{#2}%
1490 }

```

`\ns@newglossary` Define the unstarred version.

```

1491 \newcommand*{\ns@newglossary}[5][glg]{%
1492   \ifglossaryexists{#2}%
1493   {%
1494     \PackageError{glossaries}{Glossary type '#2' already exists}{%
1495       You can't define a new glossary called '#2' because it already
1496       exists}%
1497   }%
1498   {%

```

Check if default has been set

```

1499   \ifundef\glsdefaulttype
1500   {%
1501     \gdef\glsdefaulttype{#2}%
1502   }{}%

```


Add this to the list of glossary types:

```
1503 \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.

```
1504 \expandafter\gdef\csname glolist@#2\endcsname{,}%
```

Store the file extensions:

```
1505 \expandafter\edef\csname @glo@type@#2@log\endcsname{#1}%
1506 \expandafter\edef\csname @glo@type@#2@in\endcsname{#3}%
1507 \expandafter\edef\csname @glo@type@#2@out\endcsname{#4}%
1508 \expandafter\@gls@forbidtexext\csname @glo@type@#2@log\endcsname
1509 \expandafter\@gls@forbidtexext\csname @glo@type@#2@in\endcsname
1510 \expandafter\@gls@forbidtexext\csname @glo@type@#2@out\endcsname
```

Store the title:

```
1511 \expandafter\def\csname @glo@type@#2@title\endcsname{#5}%
```

```
1512 \@gls@provide@newglossary
```

```
1513 \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 redefined by the user later if required (see \defglsentry). This may already have been defined if this has been specified as a list of acronyms.

```
1514 \ifcsundef{gls@#2@entryfmt}%
1515 {%
1516   \defglsentryfmt[#2]{\glsentryfmt}%
1517 }%
1518 {}%
```

Define sort counter if required:

```
1519 \@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.)

```
1520 \ifnextchar[{\@gls@setcounter{#2}}%
1521   {\@gls@setcounter{#2}[\glscounter]}%
1522 }
```

\altnewglossary

```
1523 \newcommand*\altnewglossary}[3]{%
1524   \newglossary[#2-glg]{#1}{#2-gls}{#2-glo}{#3}%
1525 }
```

Only define new glossaries in the preamble:

```
1526 \@onlypreamble{\newglossary}
```

Only define new glossaries before \makeglossaries

```
1527 \@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 \TeX , `\@newglossary` simply ignores its arguments.

`\@newglossary`

```
1528 \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`

```
1529 \def\@gls@setcounter#1[#2]{%
1530   \expandafter\def\csname @gls@#1@counter\endcsname{#2}%

```

Add counter to xindy list, if not already added:

```
1531   \ifglxindy
1532     \GlsAddXdyCounters{#2}%
1533   \fi
1534 }

```

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

`\@gls@getcounter`

```
1535 \newcommand*{\@gls@getcounter}[1]{%
1536   \csname @gls@#1@counter\endcsname
1537 }

```

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

```
1538 \glsdefmain

```

Define the “acronym” glossaries if required.

```
1539 \@gls@do@acronymsdef

```

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

```
1540 \@gls@do@symbolsdef
1541 \@gls@do@numbersdef
1542 \@gls@do@indexdef

```

`\newignoredglossary` 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.

```
1543 \newcommand*{\newignoredglossary}[1]{%
1544   \ifdefempty\@ignored@glossaries
1545     {%
1546       \edef\@ignored@glossaries{#1}%
1547     }%
1548     {%
1549       \eappto\@ignored@glossaries{, #1}%

```

```

1550 }%
1551 \csgdef{glolist@#1}{,}%
1552 \ifcsundef{gls@#1@entryfmt}%
1553 {%
1554   \defglsentryfmt[#1]{\glsentryfmt}%
1555 }%
1556 }%
1557 \ifdefempty\@gls@nohyperlist
1558 {%
1559   \renewcommand*{\@gls@nohyperlist}{#1}%
1560 }%
1561 {%
1562   \eappto\@gls@nohyperlist{, #1}%
1563 }%
1564 }

```

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

```

1565 \newcommand*{\@ignored@glossaries}{}

```

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

```

1566 \newcommand*{\ifignoredglossary}[3]{%
1567   \edef\@gls@igtype{#1}%
1568   \expandafter\DTLifinlist\expandafter
1569     {\@gls@igtype}{\@ignored@glossaries}{#2}{#3}%
1570 }

```

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.

```

1571 \define@key{glossentry}{name}{%
1572 \def\@glo@name{#1}%
1573 }

```

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`.

```
1574 \define@key{glossentry}{description}{%
1575 \def\@glo@desc{#1}%
1576 }
```

descriptionplural

```
1577 \define@key{glossentry}{descriptionplural}{%
1578 \def\@glo@descplural{#1}%
1579 }
```

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

```
1580 \define@key{glossentry}{sort}{%
1581 \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.

```
1582 \define@key{glossentry}{text}{%
1583 \def\@glo@text{#1}%
1584 }
```

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.

```
1585 \define@key{glossentry}{plural}{%
1586 \def\@glo@plural{#1}%
1587 }
```

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.

```
1588 \define@key{glossentry}{first}{%
1589 \def\@glo@first{#1}%
1590 }
```

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.

```
1591 \define@key{glossentry}{firstplural}{%
1592 \def\@glo@firstplural{#1}%
1593 }
```

`\@gls@default@value`

```
1594 \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).

```
1595 \define@key{glossentry}{symbol}{%
1596 \def\@glo@symbol{#1}%
1597 }
```

symbolplural

```
1598 \define@key{glossentry}{symbolplural}{%
1599 \def\@glo@symbolplural{#1}%
1600 }
```

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

```
1601 \define@key{glossentry}{type}{%
1602 \def\@glo@type{#1}}
```

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

```
1603 \define@key{glossentry}{counter}{%
1604   \ifcsundef{c@#1}%
1605   {%
1606     \PackageError{glossaries}%
1607     {There is no counter called ‘#1’}%
1608     {%
1609       The counter key should have the name of a valid counter
1610       as its value%
1611     }%
1612   }%
1613   {%
1614     \def\@glo@counter{#1}%
1615   }%
1616 }
```

see The see key specifies a list of cross-references

```
1617 \define@key{glossentry}{see}{%
1618   \gls@checkseeallowed
1619   \def\@glo@see{#1}%
1620   \@glo@seeautonumberlist
1621 }
```

\gls@checkseeallowed

```
1622 \newcommand*{\gls@checkseeallowed}{%
1623   \PackageError{glossaries}%
```

```

1624  {'see' key may only be used after \string\makeglossaries\space
1625    or \string\makenoidxglossaries}%
1626  {You must use \string\makeglossaries\space
1627    or \string\makenoidxglossaries\space before defining
1628    any entries that have a 'see' key}%
1629 }

```

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

```

1630 \define@key{glossentry}{parent}{%
1631 \def\@glo@parent{#1}}

```

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

```

1632 \define@choicekey{glossentry}{nonumberlist}[\val\nr]{true,false}[true]{%
1633 \ifcase\nr\relax
1634 \def\@glo@prefix{\glsnonextpages}%
1635 \else
1636 \def\@glo@prefix{\glsnextpages}%
1637 \fi
1638 }

```

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

user1

```

1639 \define@key{glossentry}{user1}{%
1640 \def\@glo@useri{#1}%
1641 }

```

user2

```

1642 \define@key{glossentry}{user2}{%
1643 \def\@glo@userii{#1}%
1644 }

```

user3

```

1645 \define@key{glossentry}{user3}{%
1646 \def\@glo@useriii{#1}%
1647 }

```

user4

```

1648 \define@key{glossentry}{user4}{%
1649 \def\@glo@useriv{#1}%
1650 }

```

user5

```

1651 \define@key{glossentry}{user5}{%
1652 \def\@glo@userv{#1}%
1653 }

```

user6

```
1654 \define@key{glossentry}{user6}{%  
1655   \def\@glo@uservi{#1}%  
1656 }
```

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.

```
1657 \define@key{glossentry}{short}{%  
1658   \def\@glo@short{#1}%  
1659 }
```

shortplural This key is provided for use by \newacronym.

```
1660 \define@key{glossentry}{shortplural}{%  
1661   \def\@glo@shortpl{#1}%  
1662 }
```

long This key is provided for use by \newacronym.

```
1663 \define@key{glossentry}{long}{%  
1664   \def\@glo@long{#1}%  
1665 }
```

longplural This key is provided for use by \newacronym.

```
1666 \define@key{glossentry}{longplural}{%  
1667   \def\@glo@longpl{#1}%  
1668 }
```

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

```
1669 \newcommand*{\@glsnname}{%  
1670   \PackageError{glossaries}{name key required in  
1671   \string\newglossaryentry\space for entry '\@glo@label'}{You  
1672   haven't specified the entry name}}
```

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

```
1673 \newcommand*{\@glsnodelsc}{%  
1674   \PackageError{glossaries}  
1675   {%  
1676     description key required in \string\newglossaryentry\space  
1677     for entry '\@glo@label'%  
1678   }%  
1679   {%  
1680     You haven't specified the entry description%  
1681   }%  
1682 }%
```

\@glsdefaultplural Now obsolete. Don't use.

```
1683 \newcommand*{\@glsdefaultplural}{{}}
```

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

```
1684 \newcommand*{\@gls@missingnumberlist}[1]{%
1685   ??%
1686   \ifglssavenumberlist
1687     \GlossariesWarning{Missing number list for entry ‘#1’.
1688       Maybe makeglossaries + rerun required.}%
1689   \else
1690     \PackageError{glossaries}%
1691       {Package option ‘savenumberlist=true’ required.}%
1692     {%
1693       You must use the ‘savenumberlist’ package option
1694       to reference location lists.%
1695     }%
1696   \fi
1697 }
```

\@glsdefaultsort Define command to set default sort.

```
1698 \newcommand*{\@glsdefaultsort}{\@glo@name}
```

\gls@level Register to increment entry levels.

```
1699 \newcount\gls@level
```

@gls@noexpand@field

```
1700 \newcommand{\@gls@noexpand@field}[3]{%
1701   \expandafter\global\expandafter
1702     \let\csname glo@#1@#2\endcsname#3%
1703 }
```

gls@noexpand@fields

```
1704 \newcommand{\@gls@noexpand@fields}[4]{%
1705   \ifcsdef{gls@assign@#3@field}
1706     {%
1707       \ifdefequal{#4}{\@gls@default@value}%
1708       {%
1709         \edef\@gls@value{\expandonce{#1}}%
1710         \csuse{gls@assign@#3@field}{#2}{\@gls@value}%
1711       }%
1712     }%
1713     \csuse{gls@assign@#3@field}{#2}{#4}%
1714   }%
1715 }%
1716 {%
1717   \ifdefequal{#4}{\@gls@default@value}%
1718   {%
1719     \edef\@gls@value{\expandonce{#1}}%
1720     \@gls@noexpand@field{#2}{#3}{\@gls@value}%
1721   }%
1722   {%
```



```

1723     \@@gls@noexpand@field{#2}{#3}{#4}%
1724   }%
1725 }%
1726 }

```

\@@gls@expand@field

```

1727 \newcommand{\@@gls@expand@field}[3]{%
1728   \expandafter
1729     \protected@xdef\csname glo@#1@#2\endcsname{#3}%
1730 }

```

@gls@expand@fields

```

1731 \newcommand{\@gls@expand@fields}[4]{%
1732   \ifcsdef{gls@assign@#3@field}
1733   {%
1734     \ifdefequal{#4}{\@gls@default@value}%
1735     {%
1736       \edef\@gls@value{\expandonce{#1}}%
1737       \csuse{gls@assign@#3@field}{#2}{\@gls@value}%
1738     }%
1739     {%
1740       \expandafter\@gls@startswitexpandonce#4\relax\relax@gls@endcheck
1741       {%
1742         \@@gls@expand@field{#2}{#3}{#4}%
1743       }%
1744       {%
1745         \csuse{gls@assign@#3@field}{#2}{#4}%
1746       }%
1747     }%
1748   }%
1749   {%
1750     \ifdefequal{#4}{\@gls@default@value}%
1751     {%
1752       \@@gls@expand@field{#2}{#3}{#1}%
1753     }%
1754     {%
1755       \@@gls@expand@field{#2}{#3}{#4}%
1756     }%
1757   }%
1758 }

```

startswitexpandonce

```

1759 \def\@gls@expandonce{\expandonce}
1760 \def\@gls@startswitexpandonce#1#2@gls@endcheck#3#4{%
1761   \def\@gls@tmp{#1}%
1762   \ifdefequal{\@gls@expandonce}{\@gls@tmp}{#3}{#4}%
1763 }

```

<code>\gls@assign@field</code>	<div style="border: 1px solid black; background-color: #ffffcc; padding: 5px; margin-bottom: 10px;"> <code>\gls@assign@field{<def value>}{<glossary type>}{<field>}{<tmp cs>}</code> </div> <p>Assigns an entry field. Expansion performed by default (except for name, symbol and description where backward compatibility required). If <code><tmp cs></code> is <code><@gls@default@value></code>, <code><def value></code> is used instead.</p> <pre>1764 \let\gls@assign@field\@gls@expand@fields</pre>
<code>\glsexpandfields</code>	<p>Fully expand values when assigning fields (except for specific fields that are overridden by <code>\glssetnoexpandfield</code>).</p> <pre>1765 \newcommand*{\glsexpandfields}{% 1766 \let\gls@assign@field\@gls@expand@fields 1767 }</pre>
<code>\glsnoexpandfields</code>	<p>Don't expand values when assigning fields (except for specific fields that are overridden by <code>\glssetexpandfield</code>).</p> <pre>1768 \newcommand*{\glsnoexpandfields}{% 1769 \let\gls@assign@field\@gls@noexpand@fields 1770 }</pre>
<code>\newglossaryentry</code>	<p>Define <code>\newglossaryentry {<label>}{<key-val list>}</code>. There are two required fields in <code><key-val list></code>: name (or parent) and description. (See above.)</p> <pre>1771 \newrobustcmd{\newglossaryentry}[2]{% Check to see if this glossary entry has already been defined: 1772 \glsdoifnoexists{#1}% 1773 {% 1774 \gls@defglossaryentry{#1}{#2}% 1775 }% 1776 }</pre>
<code>\provideglossaryentry</code>	<p>Like <code>\newglossaryentry</code> but does nothing if the entry has already been defined.</p> <pre>1777 \newrobustcmd{\provideglossaryentry}[2]{% 1778 \ifglstryexists{#1}% 1779 {% 1780 {% 1781 \gls@defglossaryentry{#1}{#2}% 1782 }% 1783 } 1784 \@onlypreamble{\provideglossaryentry}</pre>
<code>\new@glossaryentry</code>	<p>For use in document environment.</p> <pre>1785 \newrobustcmd{\new@glossaryentry}[2]{% 1786 \ifundef\@gls@deffile 1787 {% 1788 \global\newwrite\@gls@deffile 1789 \immediate\openout\@gls@deffile=\jobname.glsdefs</pre>

```

1790 }%
1791 {}%
1792 \ifglentryexists{#1}{}%
1793 {%
1794   \gls@defglossaryentry{#1}{#2}%
1795 }%
1796 \@gls@writedef{#1}%
1797 }
1798 \AtBeginDocument
1799 {
1800   \makeatletter
1801   \InputIfFileExists{\jobname.glsdefs}{}{}%
1802   \makeatother
1803   \let\newglossaryentry\new@glossaryentry
1804 }
1805 \AtEndDocument{\ifdef\@gls@deffile{\closeout\@gls@deffile}{}%

```

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

```

1806 \newcommand*{\@gls@writedef}[1]{%
1807   \immediate\write\@gls@deffile
1808   {%
1809     \string\ifglentryexists{#1}{}\glspercentchar^^J%
1810     \expandafter\@gobble\string\{\glspercentchar^^J%
1811     \string\gls@defglossaryentry{\glsdetoklabel{#1}}\glspercentchar^^J%
1812     \expandafter\@gobble\string\{\glspercentchar%
1813   }%

```

Write key value information:

```

1814 \@for\@gls@map:=\@gls@keymap\do
1815 {%
1816   \edef\glo@value{\expandafter\expandonce
1817     \csname glo@\glsdetoklabel{#1}\@expandafter
1818     \@secondoftwo\@gls@map\endcsname}%
1819   \@onelevel@sanitize\glo@value
1820   \immediate\write\@gls@deffile
1821   {%
1822     \expandafter\@firstoftwo\@gls@map
1823     =\expandafter\@gobble\string\{\glo@value\expandafter\@gobble\string\},%
1824     \glspercentchar%
1825   }%
1826 }%

```

Provide hook:

```

1827 \gls.writedefhook
1828 \immediate\write\@gls@deffile
1829 {%
1830   \glspercentchar^^J%
1831   \expandafter\@gobble\string\}\glspercentchar^^J%
1832   \expandafter\@gobble\string\}\glspercentchar%
1833 }%

```

1834 }

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

```
1835 \newcommand*{\@gls@keymap}{%
1836   {name}{name},%
1837   {sort}{sortvalue},% unescaped sort value
1838   {type}{type},%
1839   {first}{first},%
1840   {firstplural}{firstpl},%
1841   {text}{text},%
1842   {plural}{plural},%
1843   {description}{desc},%
1844   {descriptionplural}{descplural},%
1845   {symbol}{symbol},%
1846   {symbolplural}{symbolplural},%
1847   {user1}{useri},%
1848   {user2}{userii},%
1849   {user3}{useriii},%
1850   {user4}{useriv},%
1851   {user5}{userv},%
1852   {user6}{uservi},%
1853   {long}{long},%
1854   {longplural}{longpl},%
1855   {short}{short},%
1856   {shortplural}{shortpl},%
1857   {counter}{counter},%
1858   {parent}{parent}}%
1859 }
```

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

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

```
1860 \newcommand*{\@gls@fetchfield}[2]{%
```

Ensure user field name is fully expanded

```
1861   \edef\@gls@thisval{#2}%
```

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

```
1862   \@for\@gls@map:=\@gls@keymap\do{%
1863     \edef\@this@key{\expandafter\@firstoftwo\@gls@map}%
1864     \ifdefequal{\@this@key}{\@gls@thisval}%
1865     {%
```

Found it.

```
1866     \edef#1{\expandafter\@secondoftwo\@gls@map}%
```

Break out of loop.

```
1867     \@endfortrue
```

```

1868 }%
1869 {}%
1870 }%
1871 }

```

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

Allow user to add their own custom keys.

```
1872 \newcommand*{\glsaddkey}{\@ifstar\sglsaddkey\@glsaddkey}
```

Starred version switches on expansion for this key.

```

1873 \newcommand*{\@sglsaddkey}[1]{%
1874   \key@ifundefined{glossentry}{#1}%
1875   {%
1876     \expandafter\newcommand\expandafter*\expandafter
1877     {\csname gls@assign@#1@field\endcsname}[2]{%
1878       \@gls@expand@field{##1}{#1}{##2}%
1879     }%
1880   }%
1881   {}%
1882   \@glsaddkey{#1}%
1883 }

```

Unstarred version doesn't override default expansion.

```
1884 \newcommand*{\@glsaddkey}[7]{%
```

Check the specified key doesn't already exist.

```

1885   \key@ifundefined{glossentry}{#1}%
1886   {%

```

Set up the key.

```

1887     \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
1888     \appto\@gls@keymap{, {#1}{#1}}%

```

Set the default value.

```
1889     \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
```

Assignment code.

```

1890     \appto\@newglossaryentryposthook{%
1891       \letcs{\@glo@tmp}{@glo@#1}%
1892       \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
1893     }%

```

Define the no-link commands.

```

1894     \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
1895     \newcommand*{#4}[1]{\@Gls@entry@field{##1}{#1}}%

```

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

```

1896     \ifcsdef{@gls@user@#1@}%
1897     {%

```

```

1898     \PackageError{glossaries}%
1899     {Can't define '\string#5' as helper command
1900     '\expandafter\string\csname @gls@user@#1@\endcsname' already exists}%
1901     {}%
1902 }%
1903 {%

1904     \expandafter\newcommand\expandafter*\expandafter
1905     {\csname @gls@user@#1@\endcsname}[2][\{%
1906         \new@ifnextchar[%
1907             {\csuse{@gls@user@#1@}{##1}{##2}}%
1908             {\csuse{@gls@user@#1@}{##1}{##2}[]}}%
1909     \csdef{@gls@user@#1@}##1##2[##3]{%
1910         \@gls@field@link{##1}{##2}{#3{##2}##3}%
1911     }%
1912     \newrobustcmd*{#5}{%
1913         \expandafter\@gls@hyp@opt\csname @gls@user@#1@\endcsname}%
1914     }%

```

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

```

1915     \ifcsdef{@Gls@user@#1@}%
1916     {%
1917         \PackageError{glossaries}%
1918         {Can't define '\string#6' as helper command
1919         '\expandafter\string\csname @Gls@user@#1@\endcsname' already exists}%
1920         {}%
1921     }%
1922     {%

1923     \expandafter\newcommand\expandafter*\expandafter
1924     {\csname @Gls@user@#1@\endcsname}[2][\{%
1925         \new@ifnextchar[%
1926             {\csuse{@Gls@user@#1@}{##1}{##2}}%
1927             {\csuse{@Gls@user@#1@}{##1}{##2}[]}}%
1928     \csdef{@Gls@user@#1@}##1##2[##3]{%
1929         \@gls@field@link{##1}{##2}{#4{##2}##3}%
1930     }%
1931     \newrobustcmd*{#6}{%
1932         \expandafter\@gls@hyp@opt\csname @Gls@user@#1@\endcsname}%
1933     }%

```

Finally the all caps version:

```

1934     \ifcsdef{@GLS@user@#1@}%
1935     {%
1936         \PackageError{glossaries}%
1937         {Can't define '\string#7' as helper command
1938         '\expandafter\string\csname @GLS@user@#1@\endcsname' already exists}%
1939         {}%
1940     }%
1941     {%

```

```

1942 \expandafter\newcommand\expandafter*\expandafter
1943 {\csname @GLS@user@#1\endcsname}[2][\{%
1944 \new@ifnextchar[%
1945 {\csuse{@GLS@user@#1@}{##1}{##2}}}%
1946 {\csuse{@GLS@user@#1@}{##1}{##2}[]}}}%
1947 \csdef{@GLS@user@#1@}##1##2[##3]{%
1948 \@gls@field@link{##1}{##2}{\mfirstucMakeUppercase{#3{##2}##3}}}%
1949 }%
1950 \newrobustcmd*{#7}{%
1951 \expandafter\@gls@hyp@opt\csname @GLS@user@#1\endcsname}%
1952 }%
1953 }%
1954 {%
1955 \PackageError{glossaries}{Key ‘#1’ already exists}{}%
1956 }%
1957 }

```

\gls.writedefhook

```

1958 \newcommand*\gls.writedefhook{}

```

\gls@assign@desc

```

1959 \newcommand*\gls@assign@desc[1]{%
1960 \gls@assign@field{#1}{desc}{\@glo@desc}%
1961 \gls@assign@field{\@glo@desc}{#1}{descplural}{\@glo@descplural}%
1962 }

```

\longnewglossaryentry

```

1963 \newcommand\longnewglossaryentry[3]{%
1964 \glsdoifnoexists{#1}%
1965 {%
1966 \bgroup
1967 \let\@org@newglossaryentryprehook\@newglossaryentryprehook
1968 \long\def\@newglossaryentryprehook{%
1969 \long\def\@glo@desc{#3\leavevmode\unskip\nopostdesc}%
1970 \@org@newglossaryentryprehook
1971 }%
1972 \renewcommand*\gls@assign@desc[1]{%
1973 \global\cslet{glo@glsetoklabel{#1}@desc}{\@glo@desc}%
1974 \global\cslet{glo@glsetoklabel{#1}@descplural}{\@glo@desc}%
1975 }
1976 \gls@defglossaryentry{#1}{#2}%
1977 \egroup
1978 }
1979 }

```

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

```

1980 \@onlypreamble{\longnewglossaryentry}

```

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

```
1981 \newcommand{\longprovideglossaryentry}[3]{%
1982   \ifglentryexists{#1}{}%
1983   {\longnewglossaryentry{#1}{#2}{#3}}%
1984 }
1985 \@onlypreamble{\longprovideglossaryentry}
```

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

Defines a new entry without checking if it already exists.

```
1986 \newcommand{\gls@defglossaryentry}[2]{%
```

Store label

```
1987   \edef\@glo@label{\glstoklabel{#1}}%
```

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

```
1988   \let\glslabel\@glo@label
```

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

```
1989   \let\@glo@name\@gls@name
```

```
1990   \let\@glo@desc\@gls@desc
```

```
1991   \let\@glo@descplural\@gls@default@value
```

```
1992   \let\@glo@type\@gls@default@value
```

```
1993   \let\@glo@symbol\@gls@default@value
```

```
1994   \let\@glo@symbolplural\@gls@default@value
```

```
1995   \let\@glo@text\@gls@default@value
```

```
1996   \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.)

```
1997   \let\@glo@first\@gls@default@value
```

```
1998   \let\@glo@firstplural\@gls@default@value
```

Set the default sort:

```
1999   \let\@glo@sort\@gls@default@value
```

Set the default counter:

```
2000   \let\@glo@counter\@gls@default@value
```

```
2001   \def\@glo@see{}%
```

```
2002   \def\@glo@parent{}%
```

```
2003   \def\@glo@prefix{}%
```



```

2004 \def\@glo@useri{}%
2005 \def\@glo@userii{}%
2006 \def\@glo@useriii{}%
2007 \def\@glo@useriv{}%
2008 \def\@glo@userv{}%
2009 \def\@glo@uservi{}%

2010 \def\@glo@short{}%
2011 \def\@glo@shortpl{}%
2012 \def\@glo@long{}%
2013 \def\@glo@longpl{}%

```

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

```
2014 \@newglossaryentryprehook
```

Extract key-val information from third parameter:

```
2015 \setkeys{glossentry}{#2}%
```

Check there is a default glossary.

```

2016 \ifundef\glsdefaulttype
2017 {%
2018 \PackageError{glossaries}%
2019 {No default glossary type (have you used ‘nomain’?)}%
2020 {If you use package option ‘nomain’ you must define
2021 a new glossary before you can define entries}%
2022 }%
2023 {}%

```

Assign type. This must be fully expandable

```

2024 \gls@assign@field{\glsdefaulttype}{\@glo@label}{type}{\@glo@type}%
2025 \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.

```

2026 \ifcsundef{glolist@\@glo@type}%
2027 {%
2028 \PackageError{glossaries}%
2029 {Glossary type ‘\@glo@type’ has not been defined}%
2030 {You need to define a new glossary type, before making entries
2031 in it}%
2032 }%
2033 {%

```

Check if it's an ignored glossary

```

2034 \ifignoredglossary\@glo@type
2035 {%

```

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

```

2036 \ifx\@glo@desc\glsnodels
2037 \let\@glo@desc\empty
2038 \fi
2039 }%

```

```

2040      {%
2041      }%
2042      \protected@edef\@glo@list@{\csname glo@list@\@glo@type\endcsname}%
2043      \expandafter\xdef\csname glo@list@\@glo@type\endcsname{%
2044      \@glo@list@{\@glo@label},}%
2045      }%

  Initialise level to 0.
2046      \gls@level=0\relax

  Has this entry been assigned a parent?
2047      \ifx\@glo@parent\@empty

    Doesn't have a parent. Set \glo@<label>@parent to empty.
2048      \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%
2049      \else

    Has a parent. Check to ensure this entry isn't its own parent.
2050      \ifdefequal\@glo@label\@glo@parent%
2051      {%
2052      \PackageError{glossaries}{Entry '@glo@label' can't be its own parent}{}%
2053      \def\@glo@parent{}%
2054      \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%
2055      }%
2056      {%

    Check the parent exists:
2057      \ifglentryexists{\@glo@parent}%
2058      {%

    Parent exists. Set \glo@<label>@parent.
2059      \expandafter\xdef\csname glo@\@glo@label @parent\endcsname{%
2060      \@glo@parent}%

    Determine level.
2061      \gls@level=\csname glo@\@glo@parent @level\endcsname\relax
2062      \advance\gls@level by 1\relax

    If name hasn't been specified, use same as the parent name
2063      \ifx\@glo@name\@gls@noname
2064      \expandafter\let\expandafter\@glo@name
2065      \csname glo@\@glo@parent @name\endcsname

    If name and plural haven't been specified, use same as the parent
2066      \ifx\@glo@plural\@gls@default@value
2067      \expandafter\let\expandafter\@glo@plural
2068      \csname glo@\@glo@parent @plural\endcsname
2069      \fi
2070      \fi
2071      }%
2072      {%

```

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

```

2073      \PackageError{glossaries}%
2074      {%
2075          Invalid parent '@glo@parent'
2076          for entry '@glo@label' - parent doesn't exist%
2077      }%
2078      {%
2079          Parent entries must be defined before their children%
2080      }%
2081      \def@glo@parent{%
2082          \expandafter\gdef\csname glo@\glo@label @parent\endcsname{%
2083      }%
2084      }%
2085      \fi

```

Set the level for this entry

```

2086      \expandafter\xdef\csname glo@\glo@label @level\endcsname{\number\gls@level}%

```

Define commands associated with this entry:

```

2087      \gls@assign@field{\glo@name}{\glo@label}{sortvalue}{\glo@sort}%
2088      \letcs@glo@sort{glo@\glo@label @sortvalue}%
2089      \gls@assign@field{\glo@name}{\glo@label}{text}{\glo@text}%
2090      \expandafter\gls@assign@field\expandafter
2091          {\csname glo@\glo@label @text\endcsname\glspluralsuffix}%
2092          {\glo@label}{plural}{\glo@plural}%
2093      \expandafter\gls@assign@field\expandafter
2094          {\csname glo@\glo@label @text\endcsname}%
2095          {\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.

```

2096      \ifx@glo@first@gls@default@value
2097          \expandafter\gls@assign@field\expandafter
2098              {\csname glo@\glo@label @plural\endcsname}%
2099              {\glo@label}{firstpl}{\glo@firstplural}%
2100      \else
2101          \expandafter\gls@assign@field\expandafter
2102              {\csname glo@\glo@label @first\endcsname\glspluralsuffix}%
2103              {\glo@label}{firstpl}{\glo@firstplural}%
2104      \fi

2105      \ifcsundef{glo@type@\glo@type @counter}%
2106      {%
2107          \def@glo@defaultcounter{\glscounter}%
2108      }%
2109      {%
2110          \letcs@glo@defaultcounter{glo@type@\glo@type @counter}%
2111      }%
2112      \gls@assign@field{\glo@defaultcounter}{\glo@label}{counter}{\glo@counter}%

```

```

2113 \gls@assign@field{\@glo@label}{useri}{\@glo@useri}%
2114 \gls@assign@field{\@glo@label}{userii}{\@glo@userii}%
2115 \gls@assign@field{\@glo@label}{useriii}{\@glo@useriii}%
2116 \gls@assign@field{\@glo@label}{useriv}{\@glo@useriv}%
2117 \gls@assign@field{\@glo@label}{userv}{\@glo@userv}%
2118 \gls@assign@field{\@glo@label}{uservi}{\@glo@uservi}%
2119 \gls@assign@field{\@glo@label}{short}{\@glo@short}%
2120 \gls@assign@field{\@glo@label}{shortpl}{\@glo@shortpl}%
2121 \gls@assign@field{\@glo@label}{long}{\@glo@long}%
2122 \gls@assign@field{\@glo@label}{longpl}{\@glo@longpl}%
2123 \ifx\@glo@name\@glsnname
2124 \@glsnname
2125 \let\@glo@name\@gls@default@value
2126 \fi
2127 \gls@assign@field{\@glo@label}{name}{\@glo@name}%

```

Set default numberlist if not defined:

```

2128 \ifcsundef{glo@\@glo@label @numberlist}%
2129 {%
2130 \csxdef{glo@\@glo@label @numberlist}{%
2131 \noexpand\@gls@missingnumberlist{\@glo@label}}%
2132 }%
2133 {}%

```

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.

```

2134 \def\@glo@@desc{\@glo@first}%
2135 \ifx\@glo@desc\@glo@@desc
2136 \let\@glo@desc\@glo@first
2137 \fi
2138 \ifx\@glo@desc\@glsnname
2139 \@glsnname
2140 \let\@glo@desc\@gls@default@value
2141 \fi
2142 \gls@assign@desc{\@glo@label}%

```

Set the sort key for this entry:

```

2143 \@gls@defsort{\@glo@type}{\@glo@label}%

2144 \def\@glo@@symbol{\@glo@text}%
2145 \ifx\@glo@symbol\@glo@@symbol
2146 \let\@glo@symbol\@glo@text
2147 \fi
2148 \gls@assign@field{\relax}{\@glo@label}{symbol}{\@glo@symbol}%
2149 \expandafter
2150 \gls@assign@field\expandafter
2151 {\csname glo@\@glo@label @symbol\endcsname}
2152 {\@glo@label}{symbolplural}{\@glo@symbolplural}%

```

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

```

2153 \expandafter\xdef\csname glo@\glo@label @flagfalse\endcsname{%
2154 \noexpand\global
2155 \noexpand\let\expandafter\noexpand
2156 \csname ifglo@\glo@label @flag\endcsname\noexpand\iffalse
2157 }%
2158 \expandafter\xdef\csname glo@\glo@label @flagtrue\endcsname{%
2159 \noexpand\global
2160 \noexpand\let\expandafter\noexpand
2161 \csname ifglo@\glo@label @flag\endcsname\noexpand\iftrue
2162 }%
2163 \csname glo@\glo@label @flagfalse\endcsname

```

Sort out any cross-referencing if required.

```

2164 \ifdefined\glo@see
2165 {}%
2166 {%
2167 \protected@edef\do@glsee{%
2168 \noexpand\gls@fixbraces\noexpand\glo@list\glo@see
2169 \noexpand\nil
2170 \noexpand\expandafter\noexpand\glsee\noexpand\glo@list{\glo@label}}%
2171 \@do@glsee
2172 }%

```

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

```

2173 \ifignoredglossary\glo@type
2174 {%
2175 \csdef{glo@\glo@label @index}{}%
2176 }
2177 {%
2178 \do@glo@storeentry{\glo@label}%
2179 }%

```

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

```

2180 \@newglossaryentryposthook
2181 }

```

`\glossaryentryprehook` Allow extra information to be added to glossary entries:

```

2182 \newcommand*\@newglossaryentryprehook{}

```

`\glossaryentryposthook` Allow extra information to be added to glossary entries:

```

2183 \newcommand*\@newglossaryentryposthook{}

```

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

```

2184 \newcommand*\glsmoveentry[2]{%
2185 \edef\glo@thislabel{\glsdetoklabel{#1}}%
2186 \edef\glo@type{\csname glo@\glo@thislabel @type\endcsname}%

```

```

2187 \def\glo@list{,}%
2188 \forlslentries[\glo@type]{\glo@label}%
2189 {%
2190     \ifdefequal\@glo@thislabel\glo@label
2191     }\eappto\glo@list{\glo@label,}%
2192 }%
2193 \cslet{glolist@\glo@type}{\glo@list}%
2194 \csdef{glo@\@glo@thislabel @type}{#2}%
2195 }

```

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

```

2196 \ifglxindy
2197 \newcommand*{\@glossaryentryfield}{\string\glossentry}
2198 \else
2199 \newcommand*{\@glossaryentryfield}{\string\glossentry}
2200 \fi

```

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

```

2201 \ifglxindy
2202 \newcommand*{\@glossarysubentryfield}{%
2203     \string\subglossentry}
2204 \else
2205 \newcommand*{\@glossarysubentryfield}{%
2206     \string\subglossentry}
2207 \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.)

```

2208 \newcommand{\@glo@storeentry}[1]{%

```

Escape makeindex/xindy special characters in the label:

```

2209 \edef\@glo@esclabel{#1}%
2210 \@gls@checkmkidxchars\@glo@esclabel

```

Get the sort string and escape any special characters

```

2211 \protected@edef\@glo@sort{\csname glo@#1@sort\endcsname}%
2212 \@gls@checkmkidxchars\@glo@sort

```

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

```

2213 \@gls@checkmkidxchars\@glo@prefix

```

Get the parent, if one exists

```

2214 \edef\@glo@parent{\csname glo@#1@parent\endcsname}%

Write the information to the glossary file.

2215 \ifglxindy

Store using xindy syntax.

2216 \ifx\@glo@parent\@empty

Entry doesn't have a parent

2217 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2218 (\string"\@glo@sort\string" %
2219 \string"\@glo@prefix\@glossaryentryfield{\@glo@esclabel}\string") %
2220 }%
2221 \else

Entry has a parent

2222 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2223 \csname glo@\@glo@parent @index\endcsname
2224 (\string"\@glo@sort\string" %
2225 \string"\@glo@prefix\@glossarysubentryfield
2226 {\csname glo@#1@level\endcsname}{\@glo@esclabel}\string") %
2227 }%
2228 \fi
2229 \else

Store using makeindex syntax.

2230 \ifx\@glo@parent\@empty

Sanitize \@glo@prefix

2231 \@onelevel@sanitize\@glo@prefix

Entry doesn't have a parent

2232 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2233 \@glo@sort\@gls@actualchar\@glo@prefix
2234 \@glossaryentryfield{\@glo@esclabel}%
2235 }%
2236 \else

Entry has a parent

2237 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2238 \csname glo@\@glo@parent @index\endcsname\@gls@levelchar
2239 \@glo@sort\@gls@actualchar\@glo@prefix
2240 \@glossarysubentryfield
2241 {\csname glo@#1@level\endcsname}{\@glo@esclabel}%
2242 }%
2243 \fi
2244 \fi
2245 }

```

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.

`\gls@ifnotmeasuring`

```
2246 \AtBeginDocument{%
2247   \ifpackageloaded{amsmath}%
2248   {\let\gls@ifnotmeasuring\@gls@ifnotmeasuring}%
2249   }%
2250 }
2251 \newcommand*{\@gls@ifnotmeasuring}[1]{%
2252   \ifmeasuring@
2253   \else
2254     #1%
2255   \fi
2256 }
2257 \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.

```
2258 \newcommand*{\glsreset}[1]{%
2259   \gls@ifnotmeasuring
2260   {%
2261     \glsdoifexists{#1}%
2262     {%
2263       \expandafter\global\csname glo@\glsdetoklabel{#1}@flagfalse\endcsname
2264     }%
2265   }%
2266 }
```

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

```
2267 \newcommand*{\glslocalreset}[1]{%
2268   \gls@ifnotmeasuring
2269   {%
2270     \glsdoifexists{#1}%
2271     {%
2272       \expandafter\let\csname ifglo@\glsdetoklabel{#1}@flag\endcsname\iffalse
2273     }%
2274   }%
2275 }
```

`\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.

```
2276 \newcommand*{\glsunset}[1]{%
```



```

2277 \gls@ifnotmeasuring
2278 {%
2279     \glsdoifexists{#1}%
2280     {%
2281         \expandafter\global\csname glo@\glsdetoklabel{#1}@flagtrue\endcsname
2282     }%
2283 }%
2284 }

```

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

```

2285 \newcommand*\glslocalunset}[1]{%
2286     \gls@ifnotmeasuring
2287     {%
2288         \glsdoifexists{#1}%
2289         {%
2290             \expandafter\let\csname ifglo@\glsdetoklabel{#1}@flag\endcsname\iftrue
2291         }%
2292     }%
2293 }

```

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

Syntax: `\glsresetall[⟨glossary-list⟩]`

`\glsresetall`

```

2294 \newcommand*\glsresetall}[1][\@glo@types]{%
2295     \forallglsentries[#1]{\@glsentry}%
2296     {%
2297         \glsreset{\@glsentry}%
2298     }%
2299 }

```

As above, but with only a local effect:

`\glslocalresetall`

```

2300 \newcommand*\glslocalresetall}[1][\@glo@types]{%
2301     \forallglsentries[#1]{\@glsentry}%
2302     {%
2303         \glslocalreset{\@glsentry}%
2304     }%
2305 }

```

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

Syntax: `\glsunsetall[⟨glossary-list⟩]`

`\glsunsetall`

```

2306 \newcommand*\glsunsetall}[1][\@glo@types]{%
2307     \forallglsentries[#1]{\@glsentry}%
2308     {%
2309         \glsunset{\@glsentry}%
2310     }%
2311 }

```

As above, but with only a local effect:

```
\glslocalunsetall
```

```
2312 \newcommand*{\glslocalunsetall}[1][\@gls@types]{%
2313   \forallglsentries[#1]{\@glsentry}%
2314   {%
2315     \glslocalunset{\@glsentry}%
2316   }%
2317 }
```

1.9 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
```

```
2318 \newcommand*{\loadglsentries}[2][\@gls@default]{%
2319   \let\@gls@default\glsdefaulttype
2320   \def\glsdefaulttype{#1}\input{#2}%
2321   \let\glsdefaulttype\@gls@default
2322 }
```

`\loadglsentries` can only be used in the preamble:

```
2323 \@onlypreamble{\loadglsentries}
```

1.10 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 `\glsformat`. By default this just displays the link text “as is”.

```
\glsformat
```

```
2324 \newcommand*{\glsformat}[1]{#1}
```

¹and any other valid \LaTeX code that can be used in the preamble.

`\glsentryfmt` As from version 3.11a, the way in which an entry is displayed is now governed by `\glsentryfmt`. This doesn't take any arguments. The required information is set by commands like `\gls`. To ensure backward compatibility, the default use the old `\glsdisplay` and `\glsdisplayfirst` style of commands

```
2325 \newcommand*{\glsentryfmt}{%
2326   \@@gls@default@entryfmt\glsdisplayfirst\glsdisplay
2327 }
```

Format that provides backwards compatibility:

```
2328 \newcommand*{\@@gls@default@entryfmt}[2]{%
2329   \ifdefempty\glscustomtext
2330     {%
2331       \glsifplural
2332       {%
```

Plural form

```
2333       \glscapscase
2334       {%
```

Don't adjust case

```
2335       \ifglsused\glslabel
2336       {%
```

Subsequent use

```
2337         #2{\glsentryplural{\glslabel}}%
2338         {\glsentrydescplural{\glslabel}}%
2339         {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2340       }%
2341     {%
```

First use

```
2342         #1{\glsentryfirstplural{\glslabel}}%
2343         {\glsentrydescplural{\glslabel}}%
2344         {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2345       }%
2346     }%
2347     {%
```

Make first letter upper case

```
2348       \ifglsused\glslabel
2349       {%
```

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

```
2350       \ifbool{glscompatible-3.07}%
2351       {%
2352         \protected@edef\@glo@etext{%
2353           #2{\glsentryplural{\glslabel}}%
2354           {\glsentrydescplural{\glslabel}}%
```

```

2355         {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2356     \xmakefirstuc\@glo@etext
2357 }%
2358 {%
2359     #2{\Glsentryplural{\glslabel}}%
2360     {\glsentrydescplural{\glslabel}}%
2361     {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2362 }%
2363 }%
2364 {%

```

First use

```

2365     \ifbool{glscompatible-3.07}%
2366     {%
2367         \protected@edef\@glo@etext{%
2368             #1{\glsentryfirstplural{\glslabel}}%
2369             {\glsentrydescplural{\glslabel}}%
2370             {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2371         \xmakefirstuc\@glo@etext
2372     }%
2373     {%
2374         #1{\Glsentryfirstplural{\glslabel}}%
2375         {\glsentrydescplural{\glslabel}}%
2376         {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2377     }%
2378 }%
2379 }%
2380 {%

```

Make all upper case

```

2381     \ifglsused\glslabel
2382     {%

```

Subsequent use

```

2383         \mfirstucMakeUppercase{#2{\glsentryplural{\glslabel}}%
2384         {\glsentrydescplural{\glslabel}}%
2385         {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2386     }%
2387     {%

```

First use

```

2388         \mfirstucMakeUppercase{#1{\glsentryfirstplural{\glslabel}}%
2389         {\glsentrydescplural{\glslabel}}%
2390         {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2391     }%
2392 }%
2393 }%
2394 {%

```

Singular form

```

2395     \glscapscase
2396     {%

```

Don't adjust case

```
2397      \ifglsused\glslabel
2398      {%
```

Subsequent use

```
2399      #2{\glsentrytext{\glslabel}}%
2400      {\glsentrydesc{\glslabel}}%
2401      {\glsentrysymbol{\glslabel}}{\glsinsert}%
2402      }%
2403      {%
```

First use

```
2404      #1{\glsentryfirst{\glslabel}}%
2405      {\glsentrydesc{\glslabel}}%
2406      {\glsentrysymbol{\glslabel}}{\glsinsert}%
2407      }%
2408      }%
2409      {%
```

Make first letter upper case

```
2410      \ifglsused\glslabel
2411      {%
```

Subsequent use

```
2412      \ifbool{glscompatible-3.07}%
2413      {%
2414      \protected@edef\@glo@etext{%
2415      #2{\glsentrytext{\glslabel}}%
2416      {\glsentrydesc{\glslabel}}%
2417      {\glsentrysymbol{\glslabel}}{\glsinsert}}%
2418      \xmakefirstuc\@glo@etext
2419      }%
2420      {%
2421      #2{\Glsentrytext{\glslabel}}%
2422      {\glsentrydesc{\glslabel}}%
2423      {\glsentrysymbol{\glslabel}}{\glsinsert}%
2424      }%
2425      }%
2426      {%
```

First use

```
2427      \ifbool{glscompatible-3.07}%
2428      {%
2429      \protected@edef\@glo@etext{%
2430      #1{\glsentryfirst{\glslabel}}%
2431      {\glsentrydesc{\glslabel}}%
2432      {\glsentrysymbol{\glslabel}}{\glsinsert}}%
2433      \xmakefirstuc\@glo@etext
2434      }%
2435      {%
2436      #1{\Glsentryfirst{\glslabel}}%
```

```

2437         {\glsentrydesc{\glslabel}}}%
2438         {\glsentrysymbol{\glslabel}}{\glsinsert}}%
2439     }%
2440 }%
2441 }%
2442 {%

    Make all upper case
2443     \ifglsused\glslabel
2444     {%

        Subsequent use
2445         \mfirstucMakeUppercase{#2{\glsentrytext{\glslabel}}}%
2446         {\glsentrydesc{\glslabel}}}%
2447         {\glsentrysymbol{\glslabel}}{\glsinsert}}}%
2448     }%
2449     {%

        First use
2450         \mfirstucMakeUppercase{#1{\glsentryfirst{\glslabel}}}%
2451         {\glsentrydesc{\glslabel}}}%
2452         {\glsentrysymbol{\glslabel}}{\glsinsert}}}%
2453     }%
2454 }%
2455 }%
2456 }%
2457 {%

    Custom text provided in \glsdisp
2458     \ifglsused{\glslabel}}%
2459     {%

        Subsequent use
2460         #2{\glscustomtext}}%
2461         {\glsentrydesc{\glslabel}}}%
2462         {\glsentrysymbol{\glslabel}}{\glsinsert}}}%
2463     }%
2464     {%

        First use
2465         #1{\glscustomtext}}%
2466         {\glsentrydesc{\glslabel}}}%
2467         {\glsentrysymbol{\glslabel}}{\glsinsert}}}%
2468     }%
2469 }%
2470 }

```

`\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.

```

2471 \newcommand*{\glsgenentryfmt}{%
2472     \ifdefempty\glscustomtext

```

```

2473   {%
2474   \glsifplural
2475   {%

    Plural form

2476   \glscapscase
2477   {%

    Don't adjust case

2478   \ifglsused\glslabel
2479   {%

    Subsequent use

2480   \glsentryplural{\glslabel}\glsinsert
2481   }%
2482   {%

    First use

2483   \glsentryfirstplural{\glslabel}\glsinsert
2484   }%
2485   }%
2486   {%

    Make first letter upper case

2487   \ifglsused\glslabel
2488   {%

    Subsequent use.

2489   \Glsentryplural{\glslabel}\glsinsert
2490   }%
2491   {%

    First use

2492   \Glsentryfirstplural{\glslabel}\glsinsert
2493   }%
2494   }%
2495   {%

    Make all upper case

2496   \ifglsused\glslabel
2497   {%

    Subsequent use

2498   \mfirstucMakeUppercase
2499   {\glsentryplural{\glslabel}\glsinsert}%
2500   }%
2501   {%

    First use

2502   \mfirstucMakeUppercase
2503   {\glsentryfirstplural{\glslabel}\glsinsert}%
2504   }%
2505   }%

```

2506 }%
 2507 {%

Singular form

2508 \glscapscase
 2509 {%

Don't adjust case

2510 \ifglused\glslabel
 2511 {%

Subsequent use

2512 \glstrytext{\glslabel}\glinsert
 2513 }%
 2514 {%

First use

2515 \glstryfirst{\glslabel}\glinsert
 2516 }%
 2517 }%
 2518 {%

Make first letter upper case

2519 \ifglused\glslabel
 2520 {%

Subsequent use

2521 \Glstrytext{\glslabel}\glinsert
 2522 }%
 2523 {%

First use

2524 \Glstryfirst{\glslabel}\glinsert
 2525 }%
 2526 }%
 2527 {%

Make all upper case

2528 \ifglused\glslabel
 2529 {%

Subsequent use

2530 \mfirstucMakeUppercase{\glstrytext{\glslabel}\glinsert}%
 2531 }%
 2532 {%

First use

2533 \mfirstucMakeUppercase{\glstryfirst{\glslabel}\glinsert}%
 2534 }%
 2535 }%
 2536 }%
 2537 }%
 2538 {%

Custom text provided in `\glsdisp`. (The insert is most likely to be empty at this point.)

```
2539     \glscustomtext\glsinsert
2540 }%
2541 }
```

`\glsngenacfmt` Define a generic acronym format that uses the long and short keys (or their plurals) and `\acrfullformat`, `\firstacronymfont` and `\acronymfont`.

```
2542 \newcommand*{\glsngenacfmt}{%
2543   \ifdefempty\glscustomtext
2544   {%
2545     \ifglsused\glslabel
2546     {%
```

Subsequent use:

```
2547     \glsifplural
2548     {%
```

Subsequent plural form:

```
2549     \glscapscase
2550     {%
```

Subsequent plural form, don't adjust case:

```
2551     \acronymfont{\glsentryshortpl{\glslabel}}\glsinsert
2552     }%
2553     {%
```

Subsequent plural form, make first letter upper case:

```
2554     \acronymfont{\Glsentryshortpl{\glslabel}}\glsinsert
2555     }%
2556     {%
```

Subsequent plural form, all caps:

```
2557     \mfirstucMakeUppercase
2558     {\acronymfont{\glsentryshortpl{\glslabel}}\glsinsert}%
2559     }%
2560     }%
2561     {%
```

Subsequent singular form

```
2562     \glscapscase
2563     {%
```

Subsequent singular form, don't adjust case:

```
2564     \acronymfont{\glsentryshort{\glslabel}}\glsinsert
2565     }%
2566     {%
```

Subsequent singular form, make first letter upper case:

```
2567     \acronymfont{\Glsentryshort{\glslabel}}\glsinsert
2568     }%
2569     {%
```

Subsequent singular form, all caps:

```
2570      \mfirstucMakeUppercase
2571      {\acronymfont{\glsentryshort{\glslabel}}\glsinsert}%
2572      }%
2573      }%
2574      }%
2575      {%
```

First use:

```
2576      \glsifplural
2577      {%
```

First use plural form:

```
2578      \glscapscase
2579      {%
```

First use plural form, don't adjust case:

```
2580      \genplacrfullformat{\glslabel}{\glsinsert}%
2581      }%
2582      {%
```

First use plural form, make first letter upper case:

```
2583      \Genplacrfullformat{\glslabel}{\glsinsert}%
2584      }%
2585      {%
```

First use plural form, all caps:

```
2586      \mfirstucMakeUppercase
2587      {\genplacrfullformat{\glslabel}{\glsinsert}}%
2588      }%
2589      }%
2590      {%
```

First use singular form

```
2591      \glscapscase
2592      {%
```

First use singular form, don't adjust case:

```
2593      \genacrfullformat{\glslabel}{\glsinsert}%
2594      }%
2595      {%
```

First use singular form, make first letter upper case:

```
2596      \Genacrfullformat{\glslabel}{\glsinsert}%
2597      }%
2598      {%
```

First use singular form, all caps:

```
2599      \mfirstucMakeUppercase
2600      {\genacrfullformat{\glslabel}{\glsinsert}}%
2601      }%
2602      }%
2603      }%
```

```

2604 }%
2605 {%
    User supplied text.
2606 \glscustomtext
2607 }%
2608 }

```

`\genacrfullformat` `\genacrfullformat{<label>}{<insert>}`

The full format used by `\glsgenacfmt` (singular).

```

2609 \newcommand*{\genacrfullformat}[2]{%
2610 \glentrylong{#1}#2\space
2611 (\protect\firstacronymfont{\glentryshort{#1}})%
2612 }

```

`\Genacrfullformat` `\Genacrfullformat{<label>}{<insert>}`

As above but makes the first letter upper case.

```

2613 \newcommand*{\Genacrfullformat}[2]{%
2614 \protected@edef\gls@text{\genacrfullformat{#1}{#2}}%
2615 \xmakefirstuc\gls@text
2616 }

```

`\genplacrfullformat` `\genplacrfullformat{<label>}{<insert>}`

The full format used by `\glsgenacfmt` (plural).

```

2617 \newcommand*{\genplacrfullformat}[2]{%
2618 \glentrylongpl{#1}#2\space
2619 (\protect\firstacronymfont{\glentryshortpl{#1}})%
2620 }

```

`\Genplacrfullformat` `\Genplacrfullformat{<label>}{<insert>}`

As above but makes the first letter upper case.

```

2621 \newcommand*{\Genplacrfullformat}[2]{%
2622 \protected@edef\gls@text{\genplacrfullformat{#1}{#2}}%
2623 \xmakefirstuc\gls@text
2624 }

```

`\glsdisplayfirst` Deprecated. Kept for backward compatibility.

```

2625 \newcommand*{\glsdisplayfirst}[4]{#1#4}

```

`\glsdisplay` Deprecated. Kept for backward compatibility.

```
2626 \newcommand*{\glsdisplay}[4]{#1#4}
```

`\defglsdisplay` Deprecated. Kept for backward compatibility.

```
2627 \newcommand*{\defglsdisplay}[2][\glsdefaulttype]{%
2628   \GlossariesWarning{\string\defglsdisplay\space is now obsolete.^^J
2629   Use \string\defglsentryfmt\space instead}%
2630   \expandafter\def\csname gls@#1@display\endcsname##1##2##3##4{#2}%
2631   \edef\@gls@doentrydef{%
2632     \noexpand\defglsentryfmt[#1]{%
2633       \noexpand\ifcsdef{gls@#1@displayfirst}%
2634       {%
2635         \noexpand\@gls@default@entryfmt
2636         {\noexpand\csuse{gls@#1@displayfirst}}}%
2637         {\noexpand\csuse{gls@#1@display}}}%
2638       }%
2639       {%
2640         \noexpand\@gls@default@entryfmt
2641         {\noexpand\glsdisplayfirst}%
2642         {\noexpand\csuse{gls@#1@display}}}%
2643       }%
2644     }%
2645   }%
2646   \@gls@doentrydef
2647 }
```

`\defglsdisplayfirst` Deprecated. Kept for backward compatibility.

```
2648 \newcommand*{\defglsdisplayfirst}[2][\glsdefaulttype]{%
2649   \GlossariesWarning{\string\defglsdisplayfirst\space is now obsolete.^^J
2650   Use \string\defglsentryfmt\space instead}%
2651   \expandafter\def\csname gls@#1@displayfirst\endcsname##1##2##3##4{#2}%
2652   \edef\@gls@doentrydef{%
2653     \noexpand\defglsentryfmt[#1]{%
2654       \noexpand\ifcsdef{gls@#1@display}%
2655       {%
2656         \noexpand\@gls@default@entryfmt
2657         {\noexpand\csuse{gls@#1@displayfirst}}}%
2658         {\noexpand\csuse{gls@#1@display}}}%
2659       }%
2660       {%
2661         \noexpand\@gls@default@entryfmt
2662         {\noexpand\csuse{gls@#1@displayfirst}}}%
2663         {\noexpand\glsdisplay}%
2664       }%
2665     }%
2666   }%
2667   \@gls@doentrydef
2668 }
```

1.10.1 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.

```
2669 \define@key{glslink}{counter}{%
2670   \ifcsundef{c@#1}%
2671   {%
2672     \PackageError{glossaries}%
2673     {There is no counter called ‘#1’}%
2674     {%
2675       The counter key should have the name of a valid counter
2676       as its value%
2677     }%
2678   }%
2679   {%
2680     \def\@gls@counter{#1}%
2681   }%
2682 }
```

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.

```
2683 \define@key{glslink}{format}{%
2684   \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.

```
2685 \define@boolkey{glslink}{hyper}[true]{}
```

Initialise hyper key.

```
2686 \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.

```
2687 \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.

```
2688 \newcommand*{\glslinkvar}[3]{#1}
```

`\glsifhyper` Now deprecated.

```
2689 \newcommand*{\glsifhyper}[2]{%
2690 \glslinkvar{#1}{#2}{#1}%
2691 \GlossariesWarning{\string\glsifhyper\space is deprecated. Did
2692 you mean \string\glsifhyperon\space or \string\glslinkvar?}%
2693 }
```

`\@gls@hyp@opt` Used by the commands such as `\glslink` to determine whether to modify the hyper option.

```
2694 \newcommand*{\@gls@hyp@opt}[1]{%
2695 \let\glslinkvar\@firstofthree
2696 \let\@gls@hyp@opt@cs#1\relax
2697 \@ifstar{\s@gls@hyp@opt}%
2698 {\@ifnextchar{\@firstoftwo{\p@gls@hyp@opt}}{#1}}%
2699 }
```

`\s@gls@hyp@opt` Starred version

```
2700 \newcommand*{\s@gls@hyp@opt}[1] []{%
2701 \let\glslinkvar\@secondofthree
2702 \@gls@hyp@opt@cs[hyper=false,#1]}
```

`\p@gls@hyp@opt` Plus version

```
2703 \newcommand*{\p@gls@hyp@opt}[1] []{%
2704 \let\glslinkvar\@thirdofthree
2705 \@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`

```
2706 \newrobustcmd*{\glslink}{%
2707   \@gls@hyp@opt\@gls@link
2708 }
```

`\@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.

```
2709 \newcommand*{\@gls@link}[3][]{%
2710   \ifglsentryexists{#2}%
2711   {%
2712     \let\do@gls@link@checkfirsthyper\relax
2713     \@gls@link[#1]{#2}{#3}%
2714   }{%
2715     \PackageError{glossaries}{Glossary entry ‘#2’ has not been
2716       defined}{You need to define a glossary entry before you
2717       can use it.}%

```

Display the specified text. (The entry doesn't exist so there's nothing to link it to.)

```
2718   \glstextformat{#3}%
2719 }%
2720 }
```

`\@gls@link@checkfirsthyper` Check for first use and switch off hyper key if hyperlink not wanted. (Should be off if first use and `hyper=false` is on or if first use and both the entry is in an acronym list and the `acrfootnote` setting is on.) This assumes the glossary type is stored in `\glstype` and the label is stored in `\glslabel`.

```
2721 \newcommand*{\@gls@link@checkfirsthyper}{%
2722   \ifglsused{\glslabel}%
2723   {%
2724   }%
2725   {%
2726     \gls@checkisacronymlist\glstype
2727     \ifglshyperfirst
2728       \ifglsisacronymlist
2729         \ifglsacrfootnote
2730           \KV@glslink@hyperfalse
2731         \fi
2732       \fi
2733     \else
2734       \KV@glslink@hyperfalse

```

```

2735     \fi
2736   }%

   Allow user to hook into this
2737   \glslinkcheckfirsthyperhook
2738 }

checkfirsthyperhook Allow used to hook into the \gls@link@checkfirsthyper macro
2739 \newcommand*{\glslinkcheckfirsthyperhook}{}

\@gls@link
2740 \def\@gls@link[#1]#2#3{%
   Inserting \leavevmode suggested by Donald Arseneau (avoids problem with
   tabularx).
2741   \leavevmode
2742   \edef\glslabel{\glsdetoklabel{#2}}%
   Save options in \@gls@link@opts and label in \@gls@link@label
2743   \def\@gls@link@opts{#1}%
2744   \let\@gls@link@label\glslabel
2745   \def\@glsnumberformat{\glsnumberformat}%
2746   \edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%

   If this is in one of the “nohypertypes” glossaries, suppress the hyperlink by de-
   fault
2747   \edef\glstype{\csname glo@\glslabel @type\endcsname}%
   Save original setting
2748   \let\org@ifKV@glslink@hyper\ifKV@glslink@hyper
   Switch off hyper setting if the glossary type has been identified in nohyperlist.
2749   \expandafter\DTLifinlist\expandafter
2750     {\glstype}{\@gls@nohyperlist}%
2751   {%
2752     \KV@glslink@hyperfalse
2753   }%
2754   {%
2755   }%

   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.
2756   \do@gls@link@checkfirsthyper
2757   \setkeys{glslink}{#1}%

   Define \glsifhyperon
2758   \ifKV@glslink@hyper
2759     \let\glsifhyperon\@firstoftwo
2760   \else
2761     \let\glsifhyperon\@secondoftwo
2762   \fi

```


Store the entry's counter in \theglsentrycounter

```

2763 \gls@saveentrycounter
    Define sort key if necessary:
2764 \gls@setsort{\glslabel}%
    (De-tok'ing done by \@do@wrglossary)
2765 \@do@wrglossary{#2}%
2766 \ifKV@glslink@hyper
2767 \glslink{\glolinkprefix\glslabel}{\glstextformat{#3}}%
2768 \else
2769 \glstextformat{#3}%
2770 \fi
    Restore original setting
2771 \let\ifKV@glslink@hyper\org@ifKV@glslink@hyper
2772 }
```

\glolinkprefix

```

2773 \newcommand*{\glolinkprefix}{glo:}
```

\glsentrycounter Set default value of entry counter

```

2774 \def\glsentrycounter{\glscounter}%
```

\gls@saveentrycounter Need to check if using equation counter in align environment:

```

2775 \newcommand*{\gls@saveentrycounter}{%
2776 \def\gls@Hcounter{%
    Are we using equation counter?
2777 \ifthenelse{\equal{\gls@counter}{equation}}{%
2778 {
```

If we're in align environment, \xatlevel@ will be defined. (Can't test for \@currenvir as may be inside an inner environment.)

```

2779 \ifcsundef{xatlevel@}%
2780 {%
2781 \edef\theglsentrycounter{\expandafter\noexpand
2782 \csname the\gls@counter\endcsname}%
2783 }%
2784 {%
2785 \ifx\xatlevel@\@empty
2786 \edef\theglsentrycounter{\expandafter\noexpand
2787 \csname the\gls@counter\endcsname}%
2788 \else
2789 \savecounters@
2790 \advance\c@equation by 1\relax
2791 \edef\theglsentrycounter{\csname the\gls@counter\endcsname}%
```

Check if hyperref version of this counter

```

2792     \ifcsundef{theH\@gls@counter}%
2793     {%
2794         \def\@gls@Hcounter{\theglsentrycounter}%
2795     }%
2796     {%
2797         \def\@gls@Hcounter{\csname theH\@gls@counter\endcsname}%
2798     }%
2799     \protected@edef\theHglentrycounter{\@gls@Hcounter}%
2800     \restorecounters@
2801 \fi
2802 }%
2803 }%
2804 {%

```

Not using equation counter so no special measures:

```

2805     \edef\theglsentrycounter{\expandafter\noexpand
2806         \csname the\@gls@counter\endcsname}%
2807 }%

```

Check if hyperref version of this counter

```

2808 \ifx\@gls@Hcounter\@empty
2809     \ifcsundef{theH\@gls@counter}%
2810     {%
2811         \def\theHglentrycounter{\theglsentrycounter}%
2812     }%
2813     {%
2814         \protected@edef\theHglentrycounter{\expandafter\noexpand
2815             \csname theH\@gls@counter\endcsname}%
2816     }%
2817 \fi
2818 }

```

`\@set@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.

```

2819 \def\@set@glo@numformat#1#2#3#4{%
2820     \expandafter\@glo@check@mkidxrangechar#3\@nil
2821     \protected@edef#1{%
2822         \@glo@prefix setentrycounter[#4]{#2}%
2823         \expandafter\string\csname\@glo@suffix\endcsname
2824     }%
2825     \@gls@checkmkidxchars#1%
2826 }

```

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.

```

2827 \def\@glo@check@mkidxrangechar#1#2\@nil{%
2828 \if#1(\relax
2829   \def\@glo@prefix{)%
2830   \if\relax#2\relax
2831     \def\@glo@suffix{glsnumberformat}%
2832   \else
2833     \def\@glo@suffix{#2}%
2834   \fi
2835 \else
2836   \if#1)\relax
2837     \def\@glo@prefix{)%
2838     \if\relax#2\relax
2839       \def\@glo@suffix{glsnumberformat}%
2840     \else
2841       \def\@glo@suffix{#2}%
2842     \fi
2843   \else
2844     \def\@glo@prefix{}\def\@glo@suffix{#1#2}%
2845   \fi
2846 \fi}

```

\@gls@escbsdq Escape backslashes and double quote marks. The argument must be a control sequence.

```

2847 \newcommand*\@gls@escbsdq[1]{%
2848   \def\@gls@checkedmkidx{%
2849     \let\gls@xdystring=#1\relax
2850     \@onelevel@sanitize\gls@xdystring
2851     \edef\do@gls@xdycheckbackslash{%
2852       \noexpand\@gls@xdycheckbackslash\gls@xdystring\noexpand\@nil
2853       \@backslashchar\@backslashchar\noexpand\null}%
2854     \do@gls@xdycheckbackslash
2855     \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdystring}%
2856     \def\@gls@checkedmkidx{%
2857       \expandafter\@gls@xdycheckquote\gls@xdystring\@nil""\null
2858       \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdystring}%

```

Unsanitize \gls@numberpage, \gls@alphpage, \gls@Alphpage and \gls@romanpage (thanks to David Carlisle for the suggestion.)

```

2859 \@for\@gls@tmp:=\gls@protected@pagefmts\do
2860 {%
2861   \edef\@gls@sanitized@tmp{\expandafter\@gobble\string\\expandonce\@gls@tmp}%
2862   \@onelevel@sanitize\@gls@sanitized@tmp
2863   \edef\gls@dosubst{%
2864     \noexpand\DTLsubstituteall\noexpand\gls@xdystring
2865     {\@gls@sanitized@tmp}{\expandonce\@gls@tmp}%
2866   }%
2867   \gls@dosubst

```

2868 }%

Assign to required control sequence

2869 \let#1=\gls@xdyststring

2870 }

Catch special characters (argument must be a control sequence):

\gls@checkmkidxchars

```
2871 \newcommand{\@gls@checkmkidxchars}[1]{%
2872   \ifglxsindy
2873     \@gls@escbsdq{#1}%
2874   \else
2875     \def\@gls@checkedmkidx{%
2876       \expandafter\@gls@checkquote#1\@nil""\null
2877       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
2878     \def\@gls@checkedmkidx{%
2879       \expandafter\@gls@checkescquote#1\@nil""\null
2880       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
2881     \def\@gls@checkedmkidx{%
2882       \expandafter\@gls@checkescactual#1\@nil"??\null
2883       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
2884     \def\@gls@checkedmkidx{%
2885       \expandafter\@gls@checkactual#1\@nil??\null
2886       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
2887     \def\@gls@checkedmkidx{%
2888       \expandafter\@gls@checkbar#1\@nil||\null
2889       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
2890     \def\@gls@checkedmkidx{%
2891       \expandafter\@gls@checkescbar#1\@nil\\|\null
2892       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
2893     \def\@gls@checkedmkidx{%
2894       \expandafter\@gls@checklevel#1\@nil!!\null
2895       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
2896     \fi
2897 }
```

Update the control sequence and strip trailing \@nil:

\@gls@updatechecked

```
2898 \def\@gls@updatechecked#1\@nil#2{\def#2{#1}}
```

\@gls@tmpb Define temporary token

```
2899 \newtoks\@gls@tmpb
```

\@gls@checkquote Replace " with "" since " is a makeindex special character.

```
2900 \def\@gls@checkquote#1"#2"#3\null{%
2901   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
2902   \toks@={#1}%
2903   \ifx\@nil#2\@nil
```

```

2904 \ifx\null#3\null
2905 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
2906 \def\@gls@checkquote{\relax}%
2907 \else
2908 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
2909 \@gls@quotechar\@gls@quotechar\@gls@quotechar\@gls@quotechar}%
2910 \def\@gls@checkquote{\@gls@checkquote#3\null}%
2911 \fi
2912 \else
2913 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
2914 \@gls@quotechar\@gls@quotechar}%
2915 \ifx\null#3\null
2916 \def\@gls@checkquote{\@gls@checkquote#2""\null}%
2917 \else
2918 \def\@gls@checkquote{\@gls@checkquote#2"#3\null}%
2919 \fi
2920 \fi
2921 \@gls@checkquote
2922 }

```

\@gls@checkescquote Do the same for \":

```

2923 \def\@gls@checkescquote#1\"#2\"#3\null{%
2924 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
2925 \toks@={#1}%
2926 \ifx\null#2\null
2927 \ifx\null#3\null
2928 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
2929 \def\@gls@checkescquote{\relax}%
2930 \else
2931 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
2932 \@gls@quotechar\string\"@gls@quotechar
2933 \@gls@quotechar\string\"@gls@quotechar}%
2934 \def\@gls@checkescquote{\@gls@checkescquote#3\null}%
2935 \fi
2936 \else
2937 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
2938 \@gls@quotechar\string\"@gls@quotechar}%
2939 \ifx\null#3\null
2940 \def\@gls@checkescquote{\@gls@checkescquote#2\"\" \null}%
2941 \else
2942 \def\@gls@checkescquote{\@gls@checkescquote#2\"#3\null}%
2943 \fi
2944 \fi
2945 \@gls@checkescquote
2946 }

```

\@gls@checkescactual Similarly for \? (which is replaces @ as makeindex's special character):

```

2947 \def\@gls@checkescactual#1\?#2\?#3\null{%
2948 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%

```

```

2949 \toks@={#1}%
2950 \ifx\null#2\null
2951   \ifx\null#3\null
2952     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
2953     \def\@@gls@checkescactual{\relax}%
2954   \else
2955     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
2956       \@gls@quotearchar\string"\@gls@actualchar
2957       \@gls@quotearchar\string"\@gls@actualchar}%
2958     \def\@@gls@checkescactual{\@gls@checkescactual#3\null}%
2959   \fi
2960 \else
2961   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
2962     \@gls@quotearchar\string"\@gls@actualchar}%
2963   \ifx\null#3\null
2964     \def\@@gls@checkescactual{\@gls@checkescactual#2\?\?\null}%
2965   \else
2966     \def\@@gls@checkescactual{\@gls@checkescactual#2\?#3\null}%
2967   \fi
2968 \fi
2969 \@@gls@checkescactual
2970 }

```

\@gls@checkescbar Similarly for \|:

```

2971 \def\@gls@checkescbar#1\|#2\|#3\null{%
2972   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
2973   \toks@={#1}%
2974   \ifx\null#2\null
2975     \ifx\null#3\null
2976       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
2977       \def\@@gls@checkescbar{\relax}%
2978     \else
2979       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
2980         \@gls@quotearchar\string"\@gls@encapchar
2981         \@gls@quotearchar\string"\@gls@encapchar}%
2982       \def\@@gls@checkescbar{\@gls@checkescbar#3\null}%
2983     \fi
2984   \else
2985     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
2986       \@gls@quotearchar\string"\@gls@encapchar}%
2987     \ifx\null#3\null
2988       \def\@@gls@checkescbar{\@gls@checkescbar#2\|\|\null}%
2989     \else
2990       \def\@@gls@checkescbar{\@gls@checkescbar#2\|#3\null}%
2991     \fi
2992   \fi
2993 \@@gls@checkescbar
2994 }

```

\@gls@checkesclevel Similarly for \!:

```
2995 \def\@gls@checkesclevel#1\!#2\!#3\null{%
2996   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
2997   \toks@={#1}%
2998   \ifx\null#2\null
2999     \ifx\null#3\null
3000       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3001       \def\@gls@checkesclevel{\relax}%
3002     \else
3003       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3004         \@gls@quotechar\string"\@gls@levelchar
3005         \@gls@quotechar\string"\@gls@levelchar}%
3006       \def\@gls@checkesclevel{\@gls@checkesclevel#3\null}%
3007     \fi
3008   \else
3009     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3010       \@gls@quotechar\string"\@gls@levelchar}%
3011     \ifx\null#3\null
3012       \def\@gls@checkesclevel{\@gls@checkesclevel#2\!\!\null}%
3013     \else
3014       \def\@gls@checkesclevel{\@gls@checkesclevel#2\!#3\null}%
3015     \fi
3016   \fi
3017 \@gls@checkesclevel
3018 }
```

\@gls@checkbar and for |:

```
3019 \def\@gls@checkbar#1|#2|#3\null{%
3020   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3021   \toks@={#1}%
3022   \ifx\null#2\null
3023     \ifx\null#3\null
3024       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3025       \def\@gls@checkbar{\relax}%
3026     \else
3027       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3028         \@gls@quotechar\@gls@encapchar\@gls@quotechar\@gls@encapchar}%
3029       \def\@gls@checkbar{\@gls@checkbar#3\null}%
3030     \fi
3031   \else
3032     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3033       \@gls@quotechar\@gls@encapchar}%
3034     \ifx\null#3\null
3035       \def\@gls@checkbar{\@gls@checkbar#2||\null}%
3036     \else
3037       \def\@gls@checkbar{\@gls@checkbar#2|#3\null}%
3038     \fi
3039   \fi
3040 \@gls@checkbar
```

3041 }

\@gls@checklevel and for !:

```
3042 \def\@gls@checklevel#1!#2!#3\null{%
3043   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3044   \toks@={#1}%
3045   \ifx\null#2\null
3046     \ifx\null#3\null
3047       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3048       \def\@gls@checklevel{\relax}%
3049     \else
3050       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3051         \@gls@quotechar\@gls@levelchar\@gls@quotechar\@gls@levelchar}%
3052       \def\@gls@checklevel{\@gls@checklevel#3\null}%
3053     \fi
3054   \else
3055     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3056       \@gls@quotechar\@gls@levelchar}%
3057     \ifx\null#3\null
3058       \def\@gls@checklevel{\@gls@checklevel#2!\null}%
3059     \else
3060       \def\@gls@checklevel{\@gls@checklevel#2!#3\null}%
3061     \fi
3062   \fi
3063   \@gls@checklevel
3064 }
```

\@gls@checkactual and for ?:

```
3065 \def\@gls@checkactual#1?#2?#3\null{%
3066   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3067   \toks@={#1}%
3068   \ifx\null#2\null
3069     \ifx\null#3\null
3070       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3071       \def\@gls@checkactual{\relax}%
3072     \else
3073       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3074         \@gls@quotechar\@gls@actualchar\@gls@quotechar\@gls@actualchar}%
3075       \def\@gls@checkactual{\@gls@checkactual#3\null}%
3076     \fi
3077   \else
3078     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3079       \@gls@quotechar\@gls@actualchar}%
3080     \ifx\null#3\null
3081       \def\@gls@checkactual{\@gls@checkactual#2??\null}%
3082     \else
3083       \def\@gls@checkactual{\@gls@checkactual#2?#3\null}%
3084     \fi
3085   \fi
```



```

3086 \@@gls@checkactual
3087 }

```

\@gls@xdycheckquote As before but for use with xindy

```

3088 \def\@gls@xdycheckquote#1"#2"#3\null{%
3089 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3090 \toks@={#1}%
3091 \ifx\null#2\null
3092 \ifx\null#3\null
3093 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3094 \def\@@gls@xdycheckquote{\relax}%
3095 \else
3096 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3097 \string\}\string\}%
3098 \def\@@gls@xdycheckquote{\@gls@xdycheckquote#3\null}%
3099 \fi
3100 \else
3101 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3102 \string\}%
3103 \ifx\null#3\null
3104 \def\@@gls@xdycheckquote{\@gls@xdycheckquote#2""\null}%
3105 \else
3106 \def\@@gls@xdycheckquote{\@gls@xdycheckquote#2"#3\null}%
3107 \fi
3108 \fi
3109 \@gls@xdycheckquote
3110 }

```

s@xdycheckbackslash Need to escape all backslashes for xindy. Define command that will define
\@gls@xdycheckbackslash

```

3111 \edef\def\@gls@xdycheckbackslash{%
3112 \noexpand\def\noexpand\@gls@xdycheckbackslash##1\@backslashchar
3113 ##2\@backslashchar##3\noexpand\null{%
3114 \noexpand\@gls@tmpb=\noexpand\expandafter
3115 {\noexpand\@gls@checkedmkidx}%
3116 \noexpand\toks@={##1}%
3117 \noexpand\ifx\noexpand\null##2\noexpand\null
3118 \noexpand\ifx\noexpand\null##3\noexpand\null
3119 \noexpand\edef\noexpand\@gls@checkedmkidx{%
3120 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
3121 \noexpand\def\noexpand\@@gls@xdycheckbackslash{\relax}%
3122 \noexpand\else
3123 \noexpand\edef\noexpand\@gls@checkedmkidx{%
3124 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
3125 \@backslashchar\@backslashchar\@backslashchar\@backslashchar}%
3126 \noexpand\def\noexpand\@@gls@xdycheckbackslash{%
3127 \noexpand\@gls@xdycheckbackslash##3\noexpand\null}%
3128 \noexpand\fi
3129 \noexpand\else

```

```

3130 \noexpand\edef\noexpand\@gls@checkedmkidx{%
3131 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
3132 \@backslashchar\@backslashchar}%
3133 \noexpand\ifx\noexpand\null##3\noexpand\null
3134 \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3135 \noexpand\@gls@xdycheckbackslash##2\@backslashchar
3136 \@backslashchar\noexpand\null}%
3137 \noexpand\else
3138 \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3139 \noexpand\@gls@xdycheckbackslash##2\@backslashchar
3140 ##3\noexpand\null}%
3141 \noexpand\fi
3142 \noexpand\fi
3143 \noexpand\@gls@xdycheckbackslash
3144 }%
3145 }

```

Now go ahead and define \@gls@xdycheckbackslash

```

3146 \def@gls@xdycheckbackslash

```

\glsdohypertarget

```

3147 \newlength@gls@tmplen
3148 \newcommand*\glsdohypertarget}[2]{%
3149 \settoheight{\gls@tmplen}{#2}%
3150 \raisebox{\gls@tmplen}{\hypertarget{#1}}{#2}%
3151 }

```

\glsdohyperlink

```

3152 \newcommand*\glsdohyperlink}[2]{\hyperlink{#1}{#2}}

```

\@glslink If \hyperlink is not defined \@glslink ignores its first argument and just does the second argument, otherwise it is equivalent to \hyperlink.

```

3153 \ifcsundef{hyperlink}%
3154 {%
3155 \let\@glslink\@secondoftwo
3156 }%
3157 {%
3158 \let\@glslink\glsdohyperlink
3159 }

```

\@glstarget If \hypertarget is not defined, \@glstarget ignores its first argument and just does the second argument, otherwise it is equivalent to \hypertarget.

```

3160 \ifcsundef{hypertarget}%
3161 {%
3162 \let\@glstarget\@secondoftwo
3163 }%
3164 {%
3165 \let\@glstarget\glsdohypertarget
3166 }

```

Glossary hyperlinks can be disabled using `\glsdisablehyper` (effect can be localised):

`\glsdisablehyper`

```
3167 \newcommand{\glsdisablehyper}{%
3168   \KV@glslink@hyperfalse
3169   \let\@glslink\@secondoftwo
3170   \let\@glstarget\@secondoftwo
3171 }
```

Glossary hyperlinks can be enabled using `\glsenablehyper` (effect can be localised):

`\glsenablehyper`

```
3172 \newcommand{\glsenablehyper}{%
3173   \KV@glslink@hypertrue
3174   \let\@glslink\glsdohyperlink
3175   \let\@glstarget\glsdohypertarget
3176 }
```

Provide some convenience commands if not already defined:

```
3177 \providecommand{\@firstofthree}[3]{#1}
3178 \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`

```
3179 \newrobustcmd*{\gls}{\@gls{\@gls@hyp@opt\@gls}}
```

Defined the un-starred form. Need to determine if there is a final optional argument

`\@gls`

```
3180 \newcommand*{\@gls}[2][ ]{%
3181   \new@ifnextchar[\@gls@{#1}{#2}}{\@gls@{#1}{#2}[ ]}%
3182 }
```

\@gls@ Read in the final optional argument:

```
3183 \def\@gls@#1#2[#3]{%
3184   \glsdoifexists{#2}%
3185   {%
3186     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3187     \let\glsifplural\@secondoftwo
3188     \let\glscapscase\@firstofthree
3189     \let\glscustomtext\@empty
3190     \def\glsinsert{#3}%
```

Determine what the link text should be (this is stored in \@glo@text) Note that \@gls@link sets \glstyle.

```
3191   \def\@glo@text{\csname gls@\glstyle @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.

```
3192   \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3193   \ifKV@glslink@local
3194     \glslocalunset{#2}%
3195   \else
3196     \glsunset{#2}%
3197   \fi
3198 }%
3199 }
```

\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

```
3200 \newrobustcmd*{\Gls}{\@gls@hyp@opt\@Gls}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3201 \newcommand*{\@Gls}[2] [] {%
3202   \new@ifnextchar[{\@Gls@{#1}{#2}}{\@Gls@{#1}{#2} []}%
3203 }
```

\@Gls@ Read in the final optional argument:

```
3204 \def\@Gls@#1#2[#3]{%
3205   \glsdoifexists{#2}%
3206   {%
3207     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
```

```

3208 \let\glsifplural\@secondoftwo
3209 \let\glscapscase\@secondofthree
3210 \let\glscustomtext\@empty
3211 \def\glsinsert{#3}%

```

Determine what the link text should be (this is stored in \glo@text) Note that \gls@link sets \glstype.

```

3212 \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.

```

3213 \gls@link[#1]{#2}{\glo@text}%

```

Indicate that this entry has now been used

```

3214 \ifKV@glslink@local
3215 \glslocalunset{#2}%
3216 \else
3217 \glsunset{#2}%
3218 \fi
3219 }%
3220 }

```

\GLS behaves like \gls, but the link text is converted to uppercase:

\GLS

```

3221 \newrobustcmd*{\GLS}{\gls@hyp@opt\@GLS}

```

Defined the un-starred form. Need to determine if there is a final optional argument

```

3222 \newcommand*{\@GLS}[2][{}]{%
3223 \new@ifnextchar[{\@GLS@{#1}{#2}}{\@GLS@{#1}{#2}[]}%
3224 }

```

\@GLS@ Read in the final optional argument:

```

3225 \def\@GLS@#1#2[#3]{%
3226 \glsdoifexists{#2}%
3227 {%
3228 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3229 \let\glsifplural\@secondoftwo
3230 \let\glscapscase\@thirdofthree
3231 \let\glscustomtext\@empty
3232 \def\glsinsert{#3}%

```

Determine what the link text should be (this is stored in \glo@text). Note that \gls@link sets \glstype.

```

3233 \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.

```
3234 \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3235 \ifKV@glslink@local
3236 \glslocalunset{#2}%
3237 \else
3238 \glsunset{#2}%
3239 \fi
3240 }%
3241 }
```

`\glspl` behaves in the same way as `\gls` except it uses the plural form.

`\glspl`

```
3242 \newrobustcmd*{\glspl}{\@gls@hyp@opt\@glspl}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3243 \newcommand*{\@glspl}[2][{}]{%
3244 \new@ifnextchar[{\@glspl@{#1}{#2}}{\@glspl@{#1}{#2}[]}%
3245 }
```

`\@glspl@` Read in the final optional argument:

```
3246 \def\@glspl@#1#2[#3]{%
3247 \glsdoifexists{#2}%
3248 {%
3249 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3250 \let\glsifplural\@firstoftwo
3251 \let\glsupcase\@firstofthree
3252 \let\glsustomtext\@empty
3253 \def\glsinsert{#3}%
```

Determine what the link text should be (this is stored in `\@glo@text`) Note that `\@gls@link` sets `\glstyle`.

```
3254 \def\@glo@text{\csname gls@\glstyle @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.

```
3255 \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3256 \ifKV@glslink@local
3257 \glslocalunset{#2}%
3258 \else
3259 \glsunset{#2}%
```

```

3260   \fi
3261 }%
3262 }

```

`\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`

```

3263 \newrobustcmd*{\Glspl}{\@gls@hyp@opt\@Glspl}

```

Defined the un-starred form. Need to determine if there is a final optional argument

```

3264 \newcommand*{\@Glspl}[2][\@Glspl@#1]{%
3265   \new@ifnextchar[\@Glspl@#1]{#2}}{\@Glspl@#1}{#2}[]}%
3266 }

```

`\@Glspl@` Read in the final optional argument:

```

3267 \def\@Glspl@#1#2[#3]{%
3268   \glsdoifexists{#2}%
3269   {%
3270     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3271     \let\glsifplural\@firstoftwo
3272     \let\glsifscapscase\@secondofthree
3273     \let\glsifcustomtext\@empty
3274     \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`.

```

3275   \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.

```

3276   \@gls@link[#1]{#2}{\@glo@text}%

```

Indicate that this entry has now been used

```

3277   \ifKV@glslink@local
3278     \glslocalunset{#2}%
3279   \else
3280     \glsunset{#2}%
3281   \fi
3282 }%
3283 }

```

`\GLSp1` behaves like `\glsp1` except that all the link text is converted to uppercase.

\GLSp1

```
3284 \newrobustcmd*{\GLSp1}{\@gls@hyp@opt\GLSp1}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3285 \newcommand*{\GLSp1}[2] [] {%
```

```
3286   \new@ifnextchar[{\GLSp1@{#1}{#2}}{\GLSp1@{#1}{#2} []}%
```

```
3287 }
```

\@GLSp1 Read in the final optional argument:

```
3288 \def\GLSp1@#1#2[#3] {%
```

```
3289   \glsdoifexists{#2}%
```

```
3290   {%
```

```
3291     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
```

```
3292     \let\glsifplural\@firstoftwo
```

```
3293     \let\glscapscase\@thirdofthree
```

```
3294     \let\glscustomtext\@empty
```

```
3295     \def\glsinsert{#3}%
```

Determine what the link text should be (this is stored in \@glo@text) Note that \@gls@link sets \glstype.

```
3296   \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.

```
3297   \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3298   \ifKV@glslink@local
```

```
3299     \glslocalunset{#2}%
```

```
3300   \else
```

```
3301     \glsunset{#2}%
```

```
3302   \fi
```

```
3303   }%
```

```
3304 }
```

\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:

```
3305 \newrobustcmd*{\glsdisp}{\@gls@hyp@opt\glsdisp}
```

Defined the un-starred form.

\@glsdisp

```
3306 \newcommand*{\@glsdisp}[3] [] {%
```

```
3307   \glsdoifexists{#2}{%
```



```

3308 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3309 \let\glsifplural\@secondoftwo
3310 \let\glsupcase\@firstofthree
3311 \def\glsustomtext{#3}%
3312 \def\glsinsert{}%

```

Determine what the link text should be (this is stored in \@glo@text) Note that \@gls@link sets \glstype.

```

3313 \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.

```

3314 \@gls@link[#1]{#2}{\@glo@text}%

```

Indicate that this entry has now been used

```

3315 \ifKV@glslink@local
3316 \glslocalunset{#2}%
3317 \else
3318 \glsunset{#2}%
3319 \fi
3320 }%
3321 }

```

\@gls@field@link

```

3322 \newcommand{\@gls@field@link}[3]{%
3323 \glsdoifexists{#2}%
3324 {%
3325 \let\do@gls@link@checkfirsthyper\relax
3326 \@gls@link[#1]{#2}{#3}%
3327 }%
3328 }

```

\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

```

3329 \newrobustcmd*{\glstext}{\@gls@hyp@opt\@glstext}

```

Defined the un-starred form. Need to determine if there is a final optional argument

```

3330 \newcommand*{\@glstext}[2][ ]{%
3331 \new@ifnextchar[{\@glstext@{#1}{#2}}{\@glstext@{#1}{#2}[ ]}]

```

Read in the final optional argument:

```

3332 \def\@glstext@#1#2[#3]{%
3333 \@gls@field@link{#1}{#2}{\glstext{#2}#3}%
3334 }

```

\GLSText behaves like \glstext except the text is converted to uppercase.

`\GLStext`

```
3335 \newrobustcmd*{\GLStext}{\@gls@hyp@opt\GLStext}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3336 \newcommand*{\@GLStext}[2] [] {%
```

```
3337   \new@ifnextchar[{\@GLStext@{#1}{#2}}{\@GLStext@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3338 \def\@GLStext@#1#2[#3] {%
```

```
3339   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glstentrytext{#2}#3}}%
```

```
3340 }
```

`\Glstext` behaves like `\glstext` except that the first letter of the text is converted to uppercase.

`\Glstext`

```
3341 \newrobustcmd*{\Glstext}{\@gls@hyp@opt\Glstext}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3342 \newcommand*{\@Glstext}[2] [] {%
```

```
3343   \new@ifnextchar[{\@Glstext@{#1}{#2}}{\@Glstext@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3344 \def\@Glstext@#1#2[#3] {%
```

```
3345   \@gls@field@link{#1}{#2}{\Glstentrytext{#2}#3}%
```

```
3346 }
```

`\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`

```
3347 \newrobustcmd*{\glsfirst}{\@gls@hyp@opt\glsfirst}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3348 \newcommand*{\@glsfirst}[2] [] {%
```

```
3349   \new@ifnextchar[{\@glsfirst@{#1}{#2}}{\@glsfirst@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3350 \def\@glsfirst@#1#2[#3] {%
```

```
3351   \@gls@field@link{#1}{#2}{\glstentryfirst{#2}#3}%
```

```
3352 }
```

`\Glsfirst` behaves like `\glsfirst` except it displays the first letter in uppercase.

`\Glsfirst`

```
3353 \newrobustcmd*{\Glsfirst}{\@gls@hyp@opt\Glsfirst}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3354 \newcommand*{\@Glsfirst}[2] [] {%
3355   \new@ifnextchar[{\@Glsfirst@{#1}{#2}}{\@Glsfirst@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3356 \def\@Glsfirst@#1#2[#3] {%
3357   \@gls@field@link{#1}{#2}{\Glsentryfirst{#2}#3}%
3358 }
```

\GLSfirst behaves like \Glsfirst except it displays the text in uppercase.

\GLSfirst

```
3359 \newrobustcmd*{\GLSfirst}{\@gls@hyp@opt\@GLSfirst}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3360 \newcommand*{\@GLSfirst}[2] [] {%
3361   \new@ifnextchar[{\@GLSfirst@{#1}{#2}}{\@GLSfirst@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3362 \def\@GLSfirst@#1#2[#3] {%
3363   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentryfirst{#2}#3}}%
3364 }
```

\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

```
3365 \newrobustcmd*{\glsplural}{\@gls@hyp@opt\@glsplural}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3366 \newcommand*{\@glsplural}[2] [] {%
3367   \new@ifnextchar[{\@glsplural@{#1}{#2}}{\@glsplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3368 \def\@glsplural@#1#2[#3] {%
3369   \@gls@field@link{#1}{#2}{\Glsentryplural{#2}#3}%
3370 }
```

\Glsplural behaves like \glsplural except that the first letter is converted to uppercase.

\Glsplural

```
3371 \newrobustcmd*{\Glsplural}{\@gls@hyp@opt\@Glsplural}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3372 \newcommand*{\@Glsplural}[2] [] {%
3373   \new@ifnextchar[{\@Glsplural@{#1}{#2}}{\@Glsplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3374 \def\@Glsplural@#1#2[#3]{%
3375   \@gls@field@link{#1}{#2}{\Glsentryplural{#2}#3}%
3376 }
```

\Glsplural behaves like \glsplural except that the text is converted to uppercase.

\Glsplural

```
3377 \newrobustcmd*{\Glsplural}{\@gls@hyp@opt\@Glsplural}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3378 \newcommand*{\@Glsplural}[2][{}]{%
3379   \new@ifnextchar[{\@Glsplural@{#1}{#2}}{\@Glsplural@{#1}{#2}[]}}
```

Read in the final optional argument:

```
3380 \def\@Glsplural@#1#2[#3]{%
3381   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentryplural{#2}#3}}%
3382 }
```

\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

```
3383 \newrobustcmd*{\glsfirstplural}{\@gls@hyp@opt\@glsfirstplural}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3384 \newcommand*{\@glsfirstplural}[2][{}]{%
3385   \new@ifnextchar[{\@glsfirstplural@{#1}{#2}}{\@glsfirstplural@{#1}{#2}[]}}
```

Read in the final optional argument:

```
3386 \def\@glsfirstplural@#1#2[#3]{%
3387   \@gls@field@link{#1}{#2}{\Glsentryfirstplural{#2}#3}%
3388 }
```

\Glsfirstplural behaves like \glsfirstplural except that the first letter is converted to uppercase.

\Glsfirstplural

```
3389 \newrobustcmd*{\Glsfirstplural}{\@gls@hyp@opt\@Glsfirstplural}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3390 \newcommand*{\@Glsfirstplural}[2][{}]{%
3391   \new@ifnextchar[{\@Glsfirstplural@{#1}{#2}}{\@Glsfirstplural@{#1}{#2}[]}}
```

Read in the final optional argument:

```
3392 \def\@Glsfirstplural@#1#2[#3]{%
3393   \@gls@field@link{#1}{#2}{\Glsentryfirstplural{#2}#3}%
3394 }
```


Define the un-starred form. Need to determine if there is a final optional argument

```
3414 \newcommand*{\@GLSname}[2] [] {%
3415   \new@ifnextchar[{\@GLSname@{#1}{#2}}{\@GLSname@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3416 \def\@GLSname@#1#2[#3] {%
3417   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryname{#2}#3}}%
3418 }
```

\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

```
3419 \newrobustcmd*{\glsdesc}{\@gls@hyp@opt\@glsdesc}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3420 \newcommand*{\@glsdesc}[2] [] {%
3421   \new@ifnextchar[{\@glsdesc@{#1}{#2}}{\@glsdesc@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3422 \def\@glsdesc@#1#2[#3] {%
3423   \@gls@field@link{#1}{#2}{\glsentrydesc{#2}#3}%
3424 }
```

\Glsdesc behaves like \glsdesc except that the first letter is converted to uppercase.

\Glsdesc

```
3425 \newrobustcmd*{\Glsdesc}{\@gls@hyp@opt\@Glsdesc}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3426 \newcommand*{\@Glsdesc}[2] [] {%
3427   \new@ifnextchar[{\@Glsdesc@{#1}{#2}}{\@Glsdesc@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3428 \def\@Glsdesc@#1#2[#3] {%
3429   \@gls@field@link{#1}{#2}{\Glsentrydesc{#2}#3}%
3430 }
```

\GLSdesc behaves like \glsdesc except that the link text is converted to uppercase.

\GLSdesc

```
3431 \newrobustcmd*{\GLSdesc}{\@gls@hyp@opt\@GLSdesc}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3432 \newcommand*{\@GLSdesc}[2] [] {%
3433   \new@ifnextchar[{\@GLSdesc@{#1}{#2}}{\@GLSdesc@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3434 \def\@GLSdesc@#1#2[#3]{%
3435   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrydesc{#2}#3}}%
3436 }
```

`\glsdescplural` behaves like `\gls` except it always uses the value given by the descriptionplural key and it doesn't mark the entry as used.

`\glsdescplural`

```
3437 \newrobustcmd*{\glsdescplural}{\@gls@hyp@opt\@glsdescplural}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3438 \newcommand*{\@glsdescplural}[2][\@glsdescplural@{#1}{#2}[]]{%
3439   \new@ifnextchar[{\@glsdescplural@{#1}{#2}}{\@glsdescplural@{#1}{#2}[]}]}
```

Read in the final optional argument:

```
3440 \def\@glsdescplural@#1#2[#3]{%
3441   \@gls@field@link{#1}{#2}{\glsentrydescplural{#2}#3}%
3442 }
```

`\Glsdescplural` behaves like `\glsdescplural` except that the first letter is converted to uppercase.

`\Glsdescplural`

```
3443 \newrobustcmd*{\Glsdescplural}{\@gls@hyp@opt\@Glsdescplural}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3444 \newcommand*{\@Glsdescplural}[2][\@Glsdescplural@{#1}{#2}[]]{%
3445   \new@ifnextchar[{\@Glsdescplural@{#1}{#2}}{\@Glsdescplural@{#1}{#2}[]}]}
```

Read in the final optional argument:

```
3446 \def\@Glsdescplural@#1#2[#3]{%
3447   \@gls@field@link{#1}{#2}{\Glsentrydescplural{#2}#3}%
3448 }
```

`\GLSdescplural` behaves like `\glsdescplural` except that the link text is converted to uppercase.

`\GLSdescplural`

```
3449 \newrobustcmd*{\GLSdescplural}{\@gls@hyp@opt\@GLSdescplural}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3450 \newcommand*{\@GLSdescplural}[2][\@GLSdescplural@{#1}{#2}[]]{%
3451   \new@ifnextchar[{\@GLSdescplural@{#1}{#2}}{\@GLSdescplural@{#1}{#2}[]}]}
```

Read in the final optional argument:

```
3452 \def\@GLSdescplural@#1#2[#3]{%
3453   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrydescplural{#2}#3}}%
3454 }
```

`\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`

```
3455 \newrobustcmd*{\glssymbol}{\@gls@hyp@opt\@glssymbol}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3456 \newcommand*{\@glssymbol}[2] [] {%
```

```
3457   \new@ifnextchar[{\@glssymbol@{#1}{#2}}{\@glssymbol@{#1}{#2} []}]
```

Read in the final optional argument:

```
3458 \def\@glssymbol@#1#2[#3] {%
```

```
3459   \@gls@field@link{#1}{#2}{\glstentrysymbol{#2}#3}%
```

```
3460 }
```

`\Glsymbol` behaves like `\glssymbol` except that the first letter is converted to uppercase.

`\Glsymbol`

```
3461 \newrobustcmd*{\Glsymbol}{\@gls@hyp@opt\@Glsymbol}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3462 \newcommand*{\@Glsymbol}[2] [] {%
```

```
3463   \new@ifnextchar[{\@Glsymbol@{#1}{#2}}{\@Glsymbol@{#1}{#2} []}]
```

Read in the final optional argument:

```
3464 \def\@Glsymbol@#1#2[#3] {%
```

```
3465   \@gls@field@link{#1}{#2}{\Glstentrysymbol{#2}#3}%
```

```
3466 }
```

`\GLSsymbol` behaves like `\glssymbol` except that the link text is converted to uppercase.

`\GLSsymbol`

```
3467 \newrobustcmd*{\GLSsymbol}{\@gls@hyp@opt\@GLSsymbol}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3468 \newcommand*{\@GLSsymbol}[2] [] {%
```

```
3469   \new@ifnextchar[{\@GLSsymbol@{#1}{#2}}{\@GLSsymbol@{#1}{#2} []}]
```

Read in the final optional argument:

```
3470 \def\@GLSsymbol@#1#2[#3] {%
```

```
3471   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glstentrysymbol{#2}#3}}%
```

```
3472 }
```

`\glssymbolplural` behaves like `\gls` except it always uses the value given by the `symbolplural` key and it doesn't mark the entry as used.

`\glssymbolplural`

```
3473 \newrobustcmd*{\glssymbolplural}{\@gls@hyp@opt\@glssymbolplural}
```


Define the un-starred form. Need to determine if there is a final optional argument

```
3474 \newcommand*{\@glssymbolplural}[2] [] {%
3475   \new@ifnextchar[{\@glssymbolplural@{#1}{#2}}{\@glssymbolplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3476 \def\@glssymbolplural@#1#2[#3] {%
3477   \@gls@field@link{#1}{#2}{\glstentrysymbolplural{#2}#3}%
3478 }
```

\Glsymbolplural behaves like \glssymbolplural except that the first letter is converted to uppercase.

\Glsymbolplural

```
3479 \newrobustcmd*{\Glsymbolplural}{\@gls@hyp@opt\@Glsymbolplural}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3480 \newcommand*{\@GLssymbolplural}[2] [] {%
3481   \new@ifnextchar[{\@GLssymbolplural@{#1}{#2}}{\@GLssymbolplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3482 \def\@GLssymbolplural@#1#2[#3] {%
3483   \@gls@field@link{#1}{#2}{\GLstentrysymbolplural{#2}#3}%
3484 }
```

\GLSsymbolplural behaves like \glssymbolplural except that the link text is converted to uppercase.

\GLSsymbolplural

```
3485 \newrobustcmd*{\GLSsymbolplural}{\@gls@hyp@opt\@GLSsymbolplural}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3486 \newcommand*{\@GLSsymbolplural}[2] [] {%
3487   \new@ifnextchar[{\@GLSsymbolplural@{#1}{#2}}{\@GLSsymbolplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3488 \def\@GLSsymbolplural@#1#2[#3] {%
3489   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glstentrysymbolplural{#2}#3}}%
3490 }
```

\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

```
3491 \newrobustcmd*{\glsuseri}{\@gls@hyp@opt\@glsuseri}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3492 \newcommand*{\@glsuseri}[2] [] {%
3493   \new@ifnextchar[{\@glsuseri@{#1}{#2}}{\@glsuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3494 \def\@glsuseri@#1#2[#3]{%
3495   \@gls@field@link{#1}{#2}{\glsentryuseri{#2}#3}%
3496 }
```

`\Glsuseri` behaves like `\glsuseri` except that the first letter is converted to uppercase.

`\Glsuseri`

```
3497 \newrobustcmd*{\Glsuseri}{\@gls@hyp@opt\@Glsuseri}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3498 \newcommand*{\@Glsuseri}[2][]{%
3499   \new@ifnextchar[{\@Glsuseri@{#1}{#2}}{\@Glsuseri@{#1}{#2}[]}]}
```

Read in the final optional argument:

```
3500 \def\@Glsuseri@#1#2[#3]{%
3501   \@gls@field@link{#1}{#2}{\glsentryuseri{#2}#3}%
3502 }
```

`\GLSuseri` behaves like `\glsuseri` except that the link text is converted to uppercase.

`\GLSuseri`

```
3503 \newrobustcmd*{\GLSuseri}{\@gls@hyp@opt\@GLSuseri}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3504 \newcommand*{\@GLSuseri}[2][]{%
3505   \new@ifnextchar[{\@GLSuseri@{#1}{#2}}{\@GLSuseri@{#1}{#2}[]}]}
```

Read in the final optional argument:

```
3506 \def\@GLSuseri@#1#2[#3]{%
3507   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseri{#2}#3}}%
3508 }
```

`\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`

```
3509 \newrobustcmd*{\glsuserii}{\@gls@hyp@opt\@glsuserii}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3510 \newcommand*{\@glsuserii}[2][]{%
3511   \new@ifnextchar[{\@glsuserii@{#1}{#2}}{\@glsuserii@{#1}{#2}[]}]}
```

Read in the final optional argument:

```
3512 \def\@glsuserii@#1#2[#3]{%
3513   \@gls@field@link{#1}{#2}{\glsentryuserii{#2}#3}%
3514 }
```

`\Glsuserii` behaves like `\glsuserii` except that the first letter is converted to uppercase.

`\Glsuserii`

```
3515 \newrobustcmd*{\Glsuserii}{\@gls@hyp@opt\@Glsuserii}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3516 \newcommand*{\@Glsuserii}[2] [] {%
```

```
3517   \new@ifnextchar[{\@Glsuserii@{#1}{#2}}{\@Glsuserii@{#1}{#2} []}]
```

Read in the final optional argument:

```
3518 \def\@Glsuserii@#1#2[#3] {%
```

```
3519   \@gls@field@link{#1}{#2}{\Glsentryuserii{#2}#3}%
```

```
3520 }
```

`\GLSuserii` behaves like `\glsuserii` except that the link text is converted to uppercase.

`\GLSuserii`

```
3521 \newrobustcmd*{\GLSuserii}{\@gls@hyp@opt\@GLSuserii}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3522 \newcommand*{\@GLSuserii}[2] [] {%
```

```
3523   \new@ifnextchar[{\@GLSuserii@{#1}{#2}}{\@GLSuserii@{#1}{#2} []}]
```

Read in the final optional argument:

```
3524 \def\@GLSuserii@#1#2[#3] {%
```

```
3525   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentryuserii{#2}#3}}%
```

```
3526 }
```

`\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`

```
3527 \newrobustcmd*{\glsuseriii}{\@gls@hyp@opt\@glsuseriii}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3528 \newcommand*{\@glsuseriii}[2] [] {%
```

```
3529   \new@ifnextchar[{\@glsuseriii@{#1}{#2}}{\@glsuseriii@{#1}{#2} []}]
```

Read in the final optional argument:

```
3530 \def\@glsuseriii@#1#2[#3] {%
```

```
3531   \@gls@field@link{#1}{#2}{\Glsentryuseriii{#2}#3}%
```

```
3532 }
```

`\Glsuseriii` behaves like `\glsuseriii` except that the first letter is converted to uppercase.

`\Glsuseriii`

```
3533 \newrobustcmd*{\Glsuseriii}{\@gls@hyp@opt\@Glsuseriii}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3534 \newcommand*{\@Glsuseriii}[2] [] {%
3535   \new@ifnextchar[{\@Glsuseriii@{#1}{#2}}{\@Glsuseriii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3536 \def\@Glsuseriii@#1#2[#3] {%
3537   \@gls@field@link{#1}{#2}{\Glsentryuseriii{#2}#3}%
3538 }
```

\Glsuseriii behaves like \glsuseriii except that the link text is converted to uppercase.

\Glsuseriii

```
3539 \newrobustcmd*{\Glsuseriii}{\@gls@hyp@opt\@Glsuseriii}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3540 \newcommand*{\@Glsuseriii}[2] [] {%
3541   \new@ifnextchar[{\@Glsuseriii@{#1}{#2}}{\@Glsuseriii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3542 \def\@Glsuseriii@#1#2[#3] {%
3543   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentryuseriii{#2}#3}}%
3544 }
```

\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

```
3545 \newrobustcmd*{\glsuseriv}{\@gls@hyp@opt\@glsuseriv}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3546 \newcommand*{\@glsuseriv}[2] [] {%
3547   \new@ifnextchar[{\@glsuseriv@{#1}{#2}}{\@glsuseriv@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3548 \def\@glsuseriv@#1#2[#3] {%
3549   \@gls@field@link{#1}{#2}{\Glsentryuseriv{#2}#3}%
3550 }
```

\Glsuseriv behaves like \glsuseriv except that the first letter is converted to uppercase.

\Glsuseriv

```
3551 \newrobustcmd*{\Glsuseriv}{\@gls@hyp@opt\@Glsuseriv}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3552 \newcommand*{\@Glsuseriv}[2] [] {%
3553   \new@ifnextchar[{\@Glsuseriv@{#1}{#2}}{\@Glsuseriv@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3554 \def\@Glsuseriv@#1#2[#3]{%
3555   \@gls@field@link{#1}{#2}{\Glsentryuseriv{#2}#3}%
3556 }
```

\Glsuseriv behaves like \glsuseriv except that the link text is converted to uppercase.

\Glsuseriv

```
3557 \newrobustcmd*{\Glsuseriv}{\@gls@hyp@opt\@Glsuseriv}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3558 \newcommand*{\@Glsuseriv}[2][ ]{%
3559   \new@ifnextchar[{\@Glsuseriv@{#1}{#2}}{\@Glsuseriv@{#1}{#2}[ ]}]}
```

Read in the final optional argument:

```
3560 \def\@Glsuseriv@#1#2[#3]{%
3561   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentryuseriv{#2}#3}}%
3562 }
```

\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

```
3563 \newrobustcmd*{\glsuserv}{\@gls@hyp@opt\@glsuserv}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3564 \newcommand*{\@glsuserv}[2][ ]{%
3565   \new@ifnextchar[{\@glsuserv@{#1}{#2}}{\@glsuserv@{#1}{#2}[ ]}]}
```

Read in the final optional argument:

```
3566 \def\@glsuserv@#1#2[#3]{%
3567   \@gls@field@link{#1}{#2}{\Glsentryuserv{#2}#3}%
3568 }
```

\Glsuserv behaves like \glsuserv except that the first letter is converted to uppercase.

\Glsuserv

```
3569 \newrobustcmd*{\Glsuserv}{\@gls@hyp@opt\@Glsuserv}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3570 \newcommand*{\@Glsuserv}[2][ ]{%
3571   \new@ifnextchar[{\@Glsuserv@{#1}{#2}}{\@Glsuserv@{#1}{#2}[ ]}]}
```

Read in the final optional argument:

```
3572 \def\@Glsuserv@#1#2[#3]{%
3573   \@gls@field@link{#1}{#2}{\Glsentryuserv{#2}#3}%
3574 }
```

`\GLSuserv` behaves like `\glsuserv` except that the link text is converted to uppercase.

`\GLSuserv`

```
3575 \newrobustcmd*{\GLSuserv}{\@gls@hyp@opt\@GLSuserv}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3576 \newcommand*{\@GLSuserv}[2] [] {%
```

```
3577 \new@ifnextchar [{\@GLSuserv@{#1}{#2}}{\@GLSuserv@{#1}{#2} []}]
```

Read in the final optional argument:

```
3578 \def\@GLSuserv@#1#2[#3] {%
```

```
3579 \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuserv{#2}#3}}%
```

```
3580 }
```

`\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`

```
3581 \newrobustcmd*{\glsuservi}{\@gls@hyp@opt\@glsuservi}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3582 \newcommand*{\@glsuservi}[2] [] {%
```

```
3583 \new@ifnextchar [{\@glsuservi@{#1}{#2}}{\@glsuservi@{#1}{#2} []}]
```

Read in the final optional argument:

```
3584 \def\@glsuservi@#1#2[#3] {%
```

```
3585 \@gls@field@link{#1}{#2}{\glsentryuservi{#2}#3}}%
```

```
3586 }
```

`\Glsuservi` behaves like `\glsuservi` except that the first letter is converted to uppercase.

`\Glsuservi`

```
3587 \newrobustcmd*{\Glsuservi}{\@gls@hyp@opt\@Glsuservi}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3588 \newcommand*{\@Glsuservi}[2] [] {%
```

```
3589 \new@ifnextchar [{\@Glsuservi@{#1}{#2}}{\@Glsuservi@{#1}{#2} []}]
```

Read in the final optional argument:

```
3590 \def\@Glsuservi@#1#2[#3] {%
```

```
3591 \@gls@field@link{#1}{#2}{\Glsentryuservi{#2}#3}}%
```

```
3592 }
```

`\GLSuservi` behaves like `\glsuservi` except that the link text is converted to uppercase.

`\GLSuservi`

```
3593 \newrobustcmd*{\GLSuservi}{\@gls@hyp@opt\@GLSuservi}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3594 \newcommand*{\@GLSuservi}[2] [] {%
3595   \new@ifnextchar[{\@GLSuservi@{#1}{#2}}{\@GLSuservi@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3596 \def\@GLSuservi@#1#2[#3] {%
3597   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuservi{#2}#3}}%
3598 }
```

Now deal with acronym related keys. First the short form:

\acrshort

```
3599 \newrobustcmd*{\acrshort}{\@gls@hyp@opt\@ns@acrshort}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3600 \newcommand*{\ns@acrshort}[2] [] {%
3601   \new@ifnextchar[{\@acrshort{#1}{#2}}{\@acrshort{#1}{#2} []}]%
3602 }
```

Read in the final optional argument:

```
3603 \def\@acrshort#1#2[#3] {%
3604   \glsdoifexists{#2}%
3605   {%
3606     \let\do@gls@link@checkfirsthyper\relax
3607     \let\glsifplural\@secondoftwo
3608     \let\glsapscase\@firstofthree
3609     \let\glsinsert\@empty
3610     \def\glscustomtext{%
3611       \acronymfont{\glsentryshort{#2}}#3%
3612     }%
```

Call \@gls@link Note that \@gls@link sets \glsstyle.

```
3613   \@gls@link{#1}{#2}{\csname gls@\glsstyle @entryfmt\endcsname}%
3614   }%
3615 }
```

\Acrshort

```
3616 \newrobustcmd*{\Acrshort}{\@gls@hyp@opt\@ns@Acrshort}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3617 \newcommand*{\ns@Acrshort}[2] [] {%
3618   \new@ifnextchar[{\@Acrshort{#1}{#2}}{\@Acrshort{#1}{#2} []}]%
3619 }
```

Read in the final optional argument:

```
3620 \def\@Acrshort#1#2[#3] {%
3621   \glsdoifexists{#2}%
```

```

3622  {%
3623    \let\do@gl@link@checkfirsthyper\relax

3624    \def\glslabel{#2}%
3625    \let\gl@ifplural\@secondoftwo
3626    \let\glscapscase\@secondofthree
3627    \let\glinsert\@empty
3628    \def\glscustomtext{%
3629      \acronymfont{\glentryshort{#2}}#3%
3630    }%

    Call \@gl@link Note that \@gl@link sets \glstype.
3631    \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3632  }%
3633 }

```

\ACRshort

```

3634 \newrobustcmd*{\ACRshort}{\@gl@hyp@opt\ns@ACRshort}

```

Define the un-starred form. Need to determine if there is a final optional argument

```

3635 \newcommand*{\ns@ACRshort}[2][{}]{%
3636   \new@ifnextchar[{\@ACRshort{#1}{#2}}{\@ACRshort{#1}{#2}[]}%
3637 }

```

Read in the final optional argument:

```

3638 \def\@ACRshort#1#2[#3]{%
3639   \gl@ifexists{#2}%
3640   {%
3641     \let\do@gl@link@checkfirsthyper\relax

3642     \def\glslabel{#2}%
3643     \let\gl@ifplural\@secondoftwo
3644     \let\glscapscase\@thirdofthree
3645     \let\glinsert\@empty
3646     \def\glscustomtext{%
3647       \mfirstucMakeUppercase{\acronymfont{\glentryshort{#2}}#3}%
3648     }%

```

Call \@gl@link Note that \@gl@link sets \glstype.

```

3649   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3650 }%
3651 }

```

Short plural:

\acrshortpl

```

3652 \newrobustcmd*{\acrshortpl}{\@gl@hyp@opt\ns@acrshortpl}

```


Define the un-starred form. Need to determine if there is a final optional argument

```
3653 \newcommand*{\ns@acrshortpl}[2] [] {%
3654   \new@ifnextchar[{\@acrshortpl{#1}{#2}}{\@acrshortpl{#1}{#2} []}%
3655 }
```

Read in the final optional argument:

```
3656 \def\@acrshortpl#1#2[#3] {%
3657   \glsdoifexists{#2}%
3658   {%
3659     \let\do@gl@link@checkfirsthyper\relax

3660     \def\glslabel{#2}%
3661     \let\glsifplural\@firstoftwo
3662     \let\glscapscase\@firstofthree
3663     \let\glsinsert\@empty
3664     \def\glscustomtext{%
3665       \acronymfont{\glsentryshortpl{#2}}#3%
3666     }%
```

Call \@gl@link Note that \@gl@link sets \glstype.

```
3667   \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
3668   }%
3669 }
```

\Acrshortpl

```
3670 \newrobustcmd*{\Acrshortpl}{\@gl@hyp@opt\ns@Acrshortpl}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3671 \newcommand*{\ns@Acrshortpl}[2] [] {%
3672   \new@ifnextchar[{\@Acrshortpl{#1}{#2}}{\@Acrshortpl{#1}{#2} []}%
3673 }
```

Read in the final optional argument:

```
3674 \def\@Acrshortpl#1#2[#3] {%
3675   \glsdoifexists{#2}%
3676   {%
3677     \let\do@gl@link@checkfirsthyper\relax

3678     \def\glslabel{#2}%
3679     \let\glsifplural\@firstoftwo
3680     \let\glscapscase\@secondofthree
3681     \let\glsinsert\@empty
3682     \def\glscustomtext{%
3683       \acronymfont{\Glsentryshortpl{#2}}#3%
3684     }%
```

Call \@gl@link Note that \@gl@link sets \glstype.

```
3685   \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
3686   }
```

```

3686 }%
3687 }

```

\ACRshortpl

```

3688 \newrobustcmd*{\ACRshortpl}{\@gls@hyp@opt\ns@ACRshortpl}

```

Define the un-starred form. Need to determine if there is a final optional argument

```

3689 \newcommand*{\ns@ACRshortpl}[2][ ]{%
3690   \new@ifnextchar[{\@ACRshortpl{#1}{#2}}{\@ACRshortpl{#1}{#2}[ ]}%
3691 }

```

Read in the final optional argument:

```

3692 \def\@ACRshortpl#1#2[#3]{%
3693   \glsdoifexists{#2}%
3694   {%
3695     \let\do@gls@link@checkfirsthyper\relax

3696     \def\glslabel{#2}%
3697     \let\glsifplural\@firstoftwo
3698     \let\glscapscase\@thirdofthree
3699     \let\glsinsert\@empty
3700     \def\glscustomtext{%
3701       \mfirstucMakeUppercase{\acronymfont{\glsentryshortpl{#2}}#3}%
3702     }%

```

Call \@gls@link Note that \@gls@link sets \glsstyle.

```

3703   \@gls@link[#1]{#2}{\csname gls@glsstyle @entryfmt\endcsname}%
3704 }%
3705 }

```

\acrlong

```

3706 \newrobustcmd*{\acrlong}{\@gls@hyp@opt\ns@acrlong}

```

Define the un-starred form. Need to determine if there is a final optional argument

```

3707 \newcommand*{\ns@acrlong}[2][ ]{%
3708   \new@ifnextchar[{\@acrlong{#1}{#2}}{\@acrlong{#1}{#2}[ ]}%
3709 }

```

Read in the final optional argument:

```

3710 \def\@acrlong#1#2[#3]{%
3711   \glsdoifexists{#2}%
3712   {%
3713     \let\do@gls@link@checkfirsthyper\relax

3714     \def\glslabel{#2}%
3715     \let\glsifplural\@secondoftwo
3716     \let\glscapscase\@firstofthree
3717     \let\glsinsert\@empty

```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```
3718 \def\glscustomtext{%
3719 \glentrylong{#2}#3%
3720 }%
```

Call \@gls@link Note that \@gls@link sets \glstype.

```
3721 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3722 }%
3723 }
```

\Acrlong

```
3724 \newrobustcmd*{\Acrlong}{\@gls@hyp@opt\ns@Acrlong}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3725 \newcommand*{\ns@Acrlong}[2][{}]{%
3726 \new@ifnextchar[{\@Acrlong{#1}{#2}}{\@Acrlong{#1}{#2}[]}%
3727 }
```

Read in the final optional argument:

```
3728 \def\@Acrlong#1#2[#3]{%
3729 \glsdoifexists{#2}%
3730 {%
3731 \let\do@gls@link@checkfirsthyper\relax

3732 \def\glslabel{#2}%
3733 \let\glsifplural\@secondoftwo
3734 \let\glscapscase\@secondofthree
3735 \let\glsinsert\@empty
```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```
3736 \def\glscustomtext{%
3737 \Glsentrylong{#2}#3%
3738 }%
```

Call \@gls@link. Note that \@gls@link sets \glstype.

```
3739 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3740 }%
3741 }
```

\ACRlong

```
3742 \newrobustcmd*{\ACRlong}{\@gls@hyp@opt\ns@ACRlong}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3743 \newcommand*{\ns@ACRlong}[2][{}]{%
3744 \new@ifnextchar[{\@ACRlong{#1}{#2}}{\@ACRlong{#1}{#2}[]}%
3745 }
```

Read in the final optional argument:

```
3746 \def\@ACRlong#1#2[#3]{%
3747   \glsdoifexists{#2}%
3748   {%
3749     \let\do@gl@link@checkfirsthyper\relax

3750   \def\glslabel{#2}%
3751   \let\glsifplural\@secondoftwo
3752   \let\glscapscase\@thirdofthree
3753   \let\glsinsert\@empty
```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```
3754   \def\glscustomtext{%
3755     \mfirstucMakeUppercase{\glentrylong{#2}#3}%
3756   }%
```

Call \@gl@link. Note that \@gl@link sets \glstype.

```
3757   \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
3758   }%
3759 }
```

Short plural:

\acrlongpl

```
3760 \newrobustcmd*{\acrlongpl}{\@gl@hyp@opt\@ns@acrlongpl}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3761 \newcommand*{\ns@acrlongpl}[2][ ]{%
3762   \new@ifnextchar[{\@acrlongpl{#1}{#2}}{\@acrlongpl{#1}{#2}[ ]}%
3763 }
```

Read in the final optional argument:

```
3764 \def\@acrlongpl#1#2[#3]{%
3765   \glsdoifexists{#2}%
3766   {%
3767     \let\do@gl@link@checkfirsthyper\relax

3768   \def\glslabel{#2}%
3769   \let\glsifplural\@firstoftwo
3770   \let\glscapscase\@firstofthree
3771   \let\glsinsert\@empty
```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```
3772   \def\glscustomtext{%
3773     \glentrylongpl{#2}#3%
3774   }%
```

Call \@gls@link. Note that \@gls@link sets \glstype.

```
3775 \gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
3776 }%
3777 }
```

\Acrlongpl

```
3778 \newrobustcmd*{\Acrlongpl}{\@gls@hyp@opt\@ns@Acrlongpl}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3779 \newcommand*{\@ns@Acrlongpl}[2] [] {%
3780 \new@ifnextchar[{\@Acrlongpl{#1}{#2}}{\@Acrlongpl{#1}{#2} []}%
3781 }
```

Read in the final optional argument:

```
3782 \def\@Acrlongpl#1#2[#3] {%
3783 \glsdoifexists{#2}%
3784 {%
3785 \let\do@gls@link@checkfirsthyper\relax

3786 \def\glslabel{#2}%
3787 \let\glsifplural\@firstoftwo
3788 \let\glscapscase\@secondofthree
3789 \let\glsinsert\@empty
```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```
3790 \def\glscustomtext{%
3791 \Glsentrylongpl{#2}#3%
3792 }%
```

Call \@gls@link. Note that \@gls@link sets \glstype.

```
3793 \gls@link[#1]{#2}{\csname gls@@glo@type @entryfmt\endcsname}%
3794 }%
3795 }
```

\ACRlongpl

```
3796 \newrobustcmd*{\ACRlongpl}{\@gls@hyp@opt\@ns@ACRlongpl}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3797 \newcommand*{\@ns@ACRlongpl}[2] [] {%
3798 \new@ifnextchar[{\@ACRlongpl{#1}{#2}}{\@ACRlongpl{#1}{#2} []}%
3799 }
```

Read in the final optional argument:

```
3800 \def\@ACRlongpl#1#2[#3] {%
3801 \glsdoifexists{#2}%
3802 {%
3803 \let\do@gls@link@checkfirsthyper\relax
```

```

3804 \def\glslabel{#2}%
3805 \let\glsifplural\@firstoftwo
3806 \let\glscapscase\@thirdofthree
3807 \let\glsinsert\@empty

```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```

3808 \def\glscustomtext{%
3809 \mfirstucMakeUppercase{\glsentrylongpl{#2}#3}%
3810 }%

```

Call \@gls@link. Note that \@gls@link sets \glstype.

```

3811 \@gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
3812 }%
3813 }

```

1.10.2 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>}
```

```

3814 \newcommand*{\@gls@entry@field}[2]{%
3815 \csname glo@\glsdetoklabel{#1}@#2\endcsname
3816 }

```

\glsletentryfield \glsletentryfield{<cs>}{<label>}{<field>}

```

3817 \newcommand*{\glsletentryfield}[3]{%
3818 \letcs{#1}{glo@\glsdetoklabel{#2}@#3}%
3819 }

```

\@Gls@entry@field Generic first letter uppercase version.

```
\@Gls@entry@field{<label>}{<field>}
```

```

3820 \newcommand*{\@Gls@entry@field}[2]{%
3821 \letcs\@glo@text{glo@\glsdetoklabel{#1}@#2}%
3822 \ifdef\@glo@text
3823 {%
3824 \xmakefirstuc{\@glo@text}%
3825 }%
3826 {%

```

```

3827 \PackageError{glossaries}{Either glossary entry
3828   '\glsdetoklabel{#1}' doesn't exist or the field '#2'
3829   doesn't exist}{Check you have correctly spelt the entry
3830   label and the field name}%
3831 }%
3832 }

```

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`

```

3833 \newcommand*{\glsentryname}[1]{\@gls@entry@field{#1}{name}}

```

`\Glsentryname`

```

3834 \newrobustcmd*{\Glsentryname}[1]{%
3835   \@Gls@entryname{#1}%
3836 }

```

`\@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:

```

3837 \newcommand*{\@Gls@entryname}[1]{%
3838   \@Gls@entry@field{#1}{name}%
3839 }

```

`\@Gls@acrentryname` Now the behaviour when `\setacronymstyle` is used:

```

3840 \newcommand*{\@Gls@acrentryname}[1]{%
3841   \ifglshaslong{#1}%
3842   {%
3843     \letcs\@glo@text{glo@\glsdetoklabel{#1}@name}%
3844     \expandafter\@gls@getbody\@glo@text{}\@nil
3845     \expandafter\ifx\@gls@body\glsentrylong\relax
3846     \expandafter\Glsentrylong\@gls@rest
3847   }else
3848     \expandafter\ifx\@gls@body\glsentryshort\relax
3849     \expandafter\Glsentryshort\@gls@rest
3850   }else
3851     \expandafter\ifx\@gls@body\acronymfont\relax

```

Temporarily make `\glsentryshort` behave like `\Glsentryshort`. (This is on the assumption that the argument of `\acronymfont` is `\glsentryshort{<label>}`, as that's the behaviour of the predefined acronym styles.) This is scoped to localise the effect of the assignment.

```

3852     {%
3853       \let\glsentryshort\Glsentryshort
3854       \@glo@text

```

```

3855         }%
3856     \else
3857         \xmakefirstuc{\@gls@text}%
3858     \fi
3859 \fi
3860 \fi
3861 }%
3862 {%
    Not an acronym
3863     \@Gls@entry@field{#1}{name}%
3864 }%
3865 }

```

Get the entry description (as specified by the description when the entry was defined). The argument is the label associated with the entry. Note that unless you used `description=false` in the `sanitize` package option you may get unexpected results if the description key contained any commands.

```

\glsentrydesc
3866 \newcommand*{\glsentrydesc}[1]{\@Gls@entry@field{#1}{desc}}

\Glsentrydesc
3867 \newrobustcmd*{\Glsentrydesc}[1]{%
3868     \@Gls@entry@field{#1}{desc}%
3869 }

```

Plural form:

```

\glsentrydescplural
3870 \newcommand*{\glsentrydescplural}[1]{%
3871     \@Gls@entry@field{#1}{descplural}%
3872 }

\Glsentrydescplural
3873 \newrobustcmd*{\Glsentrydescplural}[1]{%
3874     \@Gls@entry@field{#1}{descplural}%
3875 }

```

Get the entry text, as specified by the text key when the entry was defined. The argument is the label associated with the entry:

```

\glsentrytext
3876 \newcommand*{\glsentrytext}[1]{\@Gls@entry@field{#1}{text}}

\Glsentrytext
3877 \newrobustcmd*{\Glsentrytext}[1]{%
3878     \@Gls@entry@field{#1}{text}%
3879 }

```


Get the plural form:

`\glsentryplural`

```
3880 \newcommand*{\glsentryplural}[1]{%
3881   \@gls@entry@field{#1}{plural}%
3882 }
```

`\Glsentryplural`

```
3883 \newrobustcmd*{\Glsentryplural}[1]{%
3884   \@Gls@entry@field{#1}{plural}%
3885 }
```

Get the symbol associated with this entry. The argument is the label associated with the entry.

`\glsentrysymbol`

```
3886 \newcommand*{\glsentrysymbol}[1]{%
3887   \@gls@entry@field{#1}{symbol}%
3888 }
```

`\Glsentrysymbol`

```
3889 \newrobustcmd*{\Glsentrysymbol}[1]{%
3890   \@Gls@entry@field{#1}{symbol}%
3891 }
```

Plural form:

`\glsentrysymbolplural`

```
3892 \newcommand*{\glsentrysymbolplural}[1]{%
3893   \@gls@entry@field{#1}{symbolplural}%
3894 }
```

`\Glsentrysymbolplural`

```
3895 \newrobustcmd*{\Glsentrysymbolplural}[1]{%
3896   \@Gls@entry@field{#1}{symbolplural}%
3897 }
```

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`

```
3898 \newcommand*{\glsentryfirst}[1]{%
3899   \@gls@entry@field{#1}{first}%
3900 }
```

`\Glsentryfirst`

```
3901 \newrobustcmd*{\Glsentryfirst}[1]{%
3902   \@Gls@entry@field{#1}{first}%
3903 }
```

Get the plural form (as specified by the firstplural key when the entry was defined).

glsentryfirstplural

```
3904 \newcommand*{\glsentryfirstplural}[1]{%
3905   \@gls@entry@field{#1}{firstpl}%
3906 }
```

Glsentryfirstplural

```
3907 \newrobustcmd*{\Glsentryfirstplural}[1]{%
3908   \@Gls@entry@field{#1}{firstpl}%
3909 }
```

Display the glossary type with which this entry is associated (as specified by the type key used when the entry was defined)

\glsentrytype

```
3910 \newcommand*{\glsentrytype}[1]{\@gls@entry@field{#1}{type}}
```

Display the sort text used for this entry. Note that the sort key is sanitize, so unexpected results may occur if the sort key contained commands.

\glsentrysort

```
3911 \newcommand*{\glsentrysort}[1]{%
3912   \@gls@entry@field{#1}{sort}%
3913 }
```

\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.

```
3914 \newcommand*{\glsentryuseri}[1]{%
3915   \@gls@entry@field{#1}{useri}%
3916 }
```

\Glsentryuseri

```
3917 \newrobustcmd*{\Glsentryuseri}[1]{%
3918   \@Gls@entry@field{#1}{useri}%
3919 }
```

\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.

```
3920 \newcommand*{\glsentryuserii}[1]{%
3921   \@gls@entry@field{#1}{userii}%
3922 }
```

\Glsentryuserii

```
3923 \newrobustcmd*{\Glsentryuserii}[1]{%
3924   \@Gls@entry@field{#1}{userii}%
3925 }
```

`\glentryuseriii` Get the third user key (as specified by the user3 when the entry was defined).
The argument is the label associated with the entry.

```
3926 \newcommand*{\glentryuseriii}[1]{%  
3927   \@gls@entry@field{#1}{useriii}%  
3928 }
```

`\Glsentryuseriii`

```
3929 \newrobustcmd*{\Glsentryuseriii}[1]{%  
3930   \@Gls@entry@field{#1}{useriii}%  
3931 }
```

`\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.

```
3932 \newcommand*{\glentryuseriv}[1]{%  
3933   \@gls@entry@field{#1}{useriv}%  
3934 }
```

`\Glsentryuseriv`

```
3935 \newrobustcmd*{\Glsentryuseriv}[1]{%  
3936   \@Gls@entry@field{#1}{useriv}%  
3937 }
```

`\glentryuserv` Get the fifth user key (as specified by the user5 when the entry was defined).
The argument is the label associated with the entry.

```
3938 \newcommand*{\glentryuserv}[1]{%  
3939   \@gls@entry@field{#1}{userv}%  
3940 }
```

`\Glsentryuserv`

```
3941 \newrobustcmd*{\Glsentryuserv}[1]{%  
3942   \@Gls@entry@field{#1}{userv}%  
3943 }
```

`\glentryuservi` Get the sixth user key (as specified by the user6 when the entry was defined).
The argument is the label associated with the entry.

```
3944 \newcommand*{\glentryuservi}[1]{%  
3945   \@gls@entry@field{#1}{uservi}%  
3946 }
```

`\Glsentryuservi`

```
3947 \newrobustcmd*{\Glsentryuservi}[1]{%  
3948   \@Gls@entry@field{#1}{uservi}%  
3949 }
```

`\glentryshort` Get the short key (as specified by the short the entry was defined). The argument is the label associated with the entry.

```
3950 \newcommand*{\glentryshort}[1]{\@gls@entry@field{#1}{short}}
```

```

\Glsentryshort
3951 \newrobustcmd*{\Glsentryshort}[1]{%
3952   \@Gls@entry@field{#1}{short}%
3953 }

\glsentryshortpl  Get the short plural key (as specified by the shortplural the entry was defined).
                  The argument is the label associated with the entry.
3954 \newcommand*{\glsentryshortpl}[1]{\@Gls@entry@field{#1}{shortpl}}

\Glsentryshortpl
3955 \newrobustcmd*{\Glsentryshortpl}[1]{%
3956   \@Gls@entry@field{#1}{shortpl}%
3957 }

\glsentrylong  Get the long key (as specified by the long the entry was defined). The argument
               is the label associated with the entry.
3958 \newcommand*{\glsentrylong}[1]{\@Gls@entry@field{#1}{long}}

\Glsentrylong
3959 \newrobustcmd*{\Glsentrylong}[1]{%
3960   \@Gls@entry@field{#1}{long}%
3961 }

\glsentrylongpl  Get the long plural key (as specified by the longplural the entry was defined).
                 The argument is the label associated with the entry.
3962 \newcommand*{\glsentrylongpl}[1]{\@Gls@entry@field{#1}{longpl}}

\Glsentrylongpl
3963 \newrobustcmd*{\Glsentrylongpl}[1]{%
3964   \@Gls@entry@field{#1}{longpl}%
3965 }

Short cut macros to access full form:

\glsentryfull
3966 \newcommand*{\glsentryfull}[1]{%
3967   \acrfullformat{\glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%
3968 }

\Glsentryfull
3969 \newrobustcmd*{\Glsentryfull}[1]{%
3970   \acrfullformat{\Glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%
3971 }

\glsentryfullpl
3972 \newcommand*{\glsentryfullpl}[1]{%
3973   \acrfullformat{\glsentrylongpl{#1}}{\acronymfont{\glsentryshortpl{#1}}}%
3974 }

```

\Glsentryfullpl

```
3975 \newrobustcmd*{\Glsentryfullpl}[1]{%
3976   \acrfullformat{\Glsentrylongpl{#1}}{\acronymfont{\Glsentryshortpl{#1}}}%
3977 }
```

\glsentrynumberlist Displays the number list as is.

```
3978 \newcommand*{\glsentrynumberlist}[1]{%
3979   \glsdoifexists{#1}%
3980   {%
3981     \@gls@entry@field{#1}{numberlist}%
3982   }%
3983 }
```

\glsdisplaynumberlist Formats the number list for the given entry label. Doesn't work with hyperref.

```
3984 \@ifpackageloaded{hyperref} {%
3985   \newcommand*{\glsdisplaynumberlist}[1]{%
3986     \GlossariesWarning
3987     {%
3988       \string\glsdisplaynumberlist\space
3989       doesn't work with hyperref.^^JUsing
3990       \string\glsentrynumberlist\space instead%
3991     }%
3992     \glsentrynumberlist{#1}%
3993   }%
3994 }%
3995 {%
3996   \newcommand*{\glsdisplaynumberlist}[1]{%
3997     \glsdoifexists{#1}%
3998     {%
3999       \bgroup
4000
4001       \edef\@glo@label{\glsdetoklabel{#1}}%
4002       \let\@org@glsnnumberformat\glsnnumberformat
4003       \def\glsnnumberformat##1{##1}%
4004       \protected@edef\the@numberlist{%
4005         \csname glo@\@glo@label @numberlist\endcsname}%
4006       \def\@gls@numlist@sep{}%
4007       \def\@gls@numlist@nextsep{}%
4008       \def\@gls@numlist@lastsep{}%
4009       \def\@gls@thislist{}%
4010       \def\@gls@donext@def{}%
4011       \renewcommand\do[1]{%
4012         \protected@edef\@gls@thislist{%
4013           \@gls@thislist
4014           \noexpand\@gls@numlist@sep
4015           ##1%
4016         }%
4017         \let\@gls@numlist@sep\@gls@numlist@nextsep
4018         \def\@gls@numlist@nextsep{\glsnnumlistsep}%

```

```

4018         \@gls@donext@def
4019         \def\@gls@donext@def{%
4020             \def\@gls@numlist@lastsep{\glsnumlistlastsep}%
4021         }%
4022     }%
4023     \expandafter \glsnumlistparser \expandafter{\the@numberlist}%
4024     \let\@gls@numlist@sep\@gls@numlist@lastsep
4025     \@gls@thislist
4026 \egroup
4027 }%
4028 }
4029 }

```

`\glsnumlistsep`

```
4030 \newcommand*{\glsnumlistsep}{, }
```

`\glsnumlistlastsep`

```
4031 \newcommand*{\glsnumlistlastsep}{ \& }
```

`\glshyperlink` Provide a hyperlink to a glossary entry without adding information to the glossary file. The entry needs to be added using a command like `\glslink` or `\glsadd` to ensure that the target is defined. The first (optional) argument specifies the link text. The entry name is used by default. The second argument is the entry label.

```

4032 \newcommand*{\glshyperlink}[2][\glsentrytext{\@glo@label}]{%
4033   \def\@glo@label{#2}%
4034   \@glslink{\glo@linkprefix\glsdetoklabel{#2}}{#1}}

```

1.11 Adding an entry to the glossary without generating text

The following keys are provided for `\glsadd` and `\glsaddall`:

```

4035 \define@key{glossadd}{counter}{\def\@gls@counter{#1}}
4036 \define@key{glossadd}{format}{\def\@glsnumberformat{#1}}

```

This key is only used by `\glsaddall`:

```
4037 \define@key{glossadd}{types}{\def\@glo@type{#1}}
```

`\glsadd[<options>]{<label>}`

Add a term to the glossary without generating any link text. The optional argument indicates which counter to use, and how to format it (using a key-value list) the second argument is the entry label. Note that *<options>* only has two keys: counter and format (the types key will be ignored).

`\glsadd`

```
4038 \newrobustcmd*{\glsadd}[2][ ]{%
```

Need to move to horizontal mode if not already in it, but only if not in preamble.

```

4039 \@gls@adjustmode
4040 \glsdoifexists{#2}%
4041 {%
4042 \def\@glsnumberformat{glsnumberformat}%
4043 \edef\@gls@counter{\csname glo@glsetoklabel{#2}@counter\endcsname}%
4044 \setkeys{glossadd}{#1}%

Store the entry's counter in \theglsentrycounter
4045 \@gls@saveentrycounter
4046 \@do@wrglossary{#2}%
4047 }%
4048 }

```

\@gls@adjustmode

```

4049 \newcommand*{\@gls@adjustmode}{}
4050 \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

```

4051 \newrobustcmd*{\glsaddall}[1][]{%
4052 \edef\@glo@type{\@glo@types}%
4053 \setkeys{glossadd}{#1}%
4054 \forallglsentries[\@glo@type]{\@glo@entry}{%
4055 \glsadd[#1]{\@glo@entry}%
4056 }%
4057 }

```

\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.

```

4058 \newrobustcmd*{\glsaddallunused}[1][\@glo@types]{%
4059 \forallglsentries[#1]{\@glo@entry}%
4060 {%
4061 \ifglsused{\@glo@entry}{\glsadd[format=glsignore]{\@glo@entry}}%
4062 }%
4063 }

```

\glsignore

```

4064 \newcommand*{\glsignore}[1]{}

```

1.12 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 `\glssetgrouptitle` 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.

```
4065 \edef\glsopenbrace{\expandafter\@gobble\string\{}
```

`\glsclosebrace` Define `\glsclosebrace` to make it easier to write an opening brace to a file.

```
4066 \edef\glsclosebrace{\expandafter\@gobble\string\}}
```

`\glsbackslash` Define `\glsbackslash` to make it easier to write a backslash to a file.

```
4067 \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.

```
4068 \edef\glsquote#1{\string"#1\string"}
```

`\glspercentchar` Define `\glspercentchar` to make it easier to write a percent character to a file.

```
4069 \edef\glspercentchar{\expandafter\@gobble\string\%}
```

`\glstildechar` Define `\glstildechar` to make it easier to write a tilde character to a file.

```
4070 \edef\glstildechar{\string~}
```

`\@glsfirstletter` Define the first letter to come after the digits 0,...,9. Only required for `xindy`.

```
4071 \ifglsxindy
```

```
4072 \newcommand*{\@glsfirstletter}{A}
```

```
4073 \fi
```

`stLetterAfterDigits` Sets the first letter to come after the digits 0,...,9.

```
4074 \ifglsxindy
```

```
4075 \newcommand*{\GlsSetXdyFirstLetterAfterDigits}[1]{%
```



```

4076 \renewcommand*{\@glsfirstletter}{#1}}
4077 \else
4078 \newcommand*{\GlsSetXdyFirstLetterAfterDigits}[1]{%
4079 \glsnoxywarning\GlsSetXdyFirstLetterAfterDigits}
4080 \fi

\@glsminrange Define the minimum number of successive location references to merge into a
range.
4081 \newcommand*{\@glsminrange}{2}

etXdyMinRangeLength 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.
4082 \ifglxindy
4083 \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
4084 \renewcommand*{\@glsminrange}{#1}}
4085 \else
4086 \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
4087 \glsnoxywarning\GlsSetXdyMinRangeLength}
4088 \fi

\writeist
4089 \ifglxindy
Code to use if xindy is required.
4090 \def\writeist{%
Define write register if not already defined
4091 \ifundef{\glswrite}{\newwrite\glswrite}{}%
Update attributes list
4092 \@gls@addpredefinedattributes
Open the file.
4093 \openout\glswrite=\istfilename
Write header comment at the start of the file
4094 \write\glswrite{;; xindy style file created by the glossaries
4095 package}%
4096 \write\glswrite{;; for document '\jobname' on
4097 \the\year-\the\month-\the\day}%
Specify the required styles
4098 \write\glswrite{^^J; required styles^^J}
4099 \@for\@xdystyle:=\@xdyrequiredstyles\do{%
4100 \ifx\@xdystyle\@empty
4101 \else
4102 \protected@write\glswrite{{(require
4103 \string"\@xdystyle.xdy\string")}}%
4104 \fi
4105 }%

```

List the allowed attributes (possible values used by the format key)

```
4106 \write\glswrite{^^J%
4107 ; list of allowed attributes (number formats)^^J}%
4108 \write\glswrite{(define-attributes ((\@xdyattributes)))}%
```

Define any additional alphabets

```
4109 \write\glswrite{^^J; user defined alphabets^^J}%
4110 \write\glswrite{\@xdyuseralphabets}%
```

Define location classes.

```
4111 \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.

```
4112 \@for\@gls@classI:=\@gls@xdy@locationlist\do{%
```

Case were $\langle Hprefix \rangle$ is empty:

```
4113 \protected@write\glswrite{}\{(define-location-class
4114 \string"\@gls@classI\string"^^J\space\space\space
4115 (
4116 :sep "{{"
4117 \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
4118 :sep "}"
4119 )
4120 ^^J\space\space\space
4121 :min-range-length \@glsminrange^^J%
4122 )
4123 }%
```

Nested iteration over all classes:

```
4124 {%
4125 \@for\@gls@classII:=\@gls@xdy@locationlist\do{%
4126 \protected@write\glswrite{}\{(define-location-class
4127 \string"\@gls@classII-\@gls@classI\string"
4128 ^^J\space\space\space
4129 (
4130 :sep "{"
4131 \csname @gls@xdy@Lclass@\@gls@classII\endcsname\space
4132 :sep "{{"
4133 \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
4134 :sep "}"
4135 )
4136 ^^J\space\space\space
4137 :min-range-length \@glsminrange^^J%
4138 )
4139 }%
4140 }%
4141 }%
4142 }%
```

User defined location classes (needs checking for new location format).

```

4143 \write\glswrite{^^J; user defined location classes}%
4144 \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.)

```

4145 \write\glswrite{^^J; define cross-reference class^^J}%
4146 \write\glswrite{(define-crossref-class \string"see\string"
4147 :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.)

```

4148 \write\glswrite{(markup-crossref-list
4149 :class \string"see\string"^^J\space\space\space
4150 :open \string"\string\glsseeformat\string"
4151 :close \string"{}\string")}%

```

List the order to sort the classes.

```

4152 \write\glswrite{^^J; define the order of the location classes}%
4153 \write\glswrite{(define-location-class-order
4154 (\@xdylocationclassorder))}%

```

Specify what to write to the start and end of the glossary file.

```

4155 \write\glswrite{^^J; define the glossary markup^^J}%

4156 \write\glswrite{(markup-index^^J\space\space\space
4157 :open \string"\string
4158 \glossarysection[\string\glossarytoctitle]{\string
4159 \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`)

```

4160 \@for\@this@ctr:=\@xdycounters\do{%
4161   {%
4162     \@for\@this@attr:=\@xdyattributelist\do{%
4163       \protected\write\glswrite{{}\string\providecommand*%
4164         \expandafter\string
4165         \csname glsX\@this@ctr X\@this@attr\endcsname[2]%
4166         {%
4167           \string\setentrycounter
4168           [\expandafter\@gobble\string\#1]{\@this@ctr}%
4169           \expandafter\string
4170           \csname\@this@attr\endcsname
4171           {\expandafter\@gobble\string\#2}%
4172         }%
4173       }%
4174     }%
4175   }%
4176 }%

```

Add the end part of the open tag and the rest of the markup-index information:

```
4177 \write\glswrite{%
4178   \string\begin
4179   {theglossary}\string\glossaryheader\glstildechar n\string" ^^J\space
4180   \space\space:close \string"\glpercentchar\glstildechar n\string
4181   \end{theglossary}\string\glossarypostamble
4182   \glstildechar n\string" ^^J\space\space\space
4183   :tree}}%
```

Specify what to put between letter groups

```
4184 \write\glswrite{(markup-letter-group-list
4185   :sep \string"\string\glsgroupskip\glstildechar n\string"}}%
```

Specify what to put between entries

```
4186 \write\glswrite{(markup-indexentry
4187   :open \string"\string\relax \string\glsresetentrylist
4188   \glstildechar n\string"}}%
```

Specify how to format entries

```
4189 \write\glswrite{(markup-locclass-list :open
4190   \string"\glsoopenbrace\string\glossaryentrynumbers
4191   \glsoopenbrace\string\relax\space \string"^^J\space\space\space
4192   :sep \string", \string"
4193   :close \string"\glsclosebrace\glsclosebrace\string"}}%
```

Specify how to separate location numbers

```
4194 \write\glswrite{(markup-locref-list
4195   :sep \string"\string\delimN\space\string"}}%
```

Specify how to indicate location ranges

```
4196 \write\glswrite{(markup-range
4197   :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.

```
4198 \@onelevel@sanitize\gls@suffixF
4199 \@onelevel@sanitize\gls@suffixFF
4200 \ifx\gls@suffixF\@empty
4201 \else
4202   \write\glswrite{(markup-range
4203     :close "\gls@suffixF" :length 1 :ignore-end)}}%
4204 \fi
4205 \ifx\gls@suffixFF\@empty
4206 \else
4207   \write\glswrite{(markup-range
4208     :close "\gls@suffixFF" :length 2 :ignore-end)}}%
4209 \fi
```

Specify how to format locations.

```
4210 \write\glswrite{^^J; define format to use for locations^^J}%
4211 \write\glswrite{\@xdylocref}%
```

Specify how to separate letter groups.

```
4212 \write\glswrite{^^J; define letter group list format^^J}%
4213 \write\glswrite{(markup-letter-group-list
4214 :sep \string"\string\glsgroupskip\glstildechar n\string")}%
```

Define letter group headings.

```
4215 \write\glswrite{^^J; letter group headings^^J}%
4216 \write\glswrite{(markup-letter-group
4217 :open-head \string"\string\glsgroupheading
4218 \glsoopenbrace\string"^^J\space\space\space
4219 :close-head \string"\glsclosebrace\string")}%
```

Define additional letter groups.

```
4220 \write\glswrite{^^J; additional letter groups^^J}%
4221 \write\glswrite{\@xdylettergroups}%
```

Define additional sort rules

```
4222 \write\glswrite{^^J; additional sort rules^^J}
4223 \write\glswrite{\@xdysortrules}%
```

Close the style file

```
4224 \closeout\glswrite
```

Suppress any further calls.

```
4225 \let\writeist\relax
4226 }
4227 \else
```

Code to use if makeindex is required.

```
4228 \edef\@gls@actualchar{\string?}
4229 \edef\@gls@encapchar{\string|}
4230 \edef\@gls@levelchar{\string!}
4231 \edef\@gls@quotechar{\string"}
4232 \def\writeist{\relax
4233 \ifundef{\glswrite}{\newwrite\glswrite}{\relax
4234 \openout\glswrite=\istfilename
4235 \write\glswrite{\glspersentchar\space makeindex style file
4236 created by the glossaries package}
4237 \write\glswrite{\glspersentchar\space for document
4238 '\jobname' on \the\year-\the\month-\the\day}
4239 \write\glswrite{actual '\@gls@actualchar'}
4240 \write\glswrite{encap '\@gls@encapchar'}
4241 \write\glswrite{level '\@gls@levelchar'}
4242 \write\glswrite{quote '\@gls@quotechar'}
4243 \write\glswrite{keyword \string"\string\glossaryentry\string"}
4244 \write\glswrite{preamble \string"\string\glossarysection[\string
4245 \glossarytoctitle]{\string\glossarytitle}\string
4246 \glossarypreamble\string\n\string\begin{theglossary}\string
4247 \glossaryheader\string\n\string"}
4248 \write\glswrite{postamble \string"\string%\string\n\string
4249 \end{theglossary}\string\glossarypostamble\string\n
4250 \string"}
```

```

4251 \write\glswrite{group_skip \string"\string\glsgroupskip\string\n
4252 \string"}
4253 \write\glswrite{item_0 \string"\string%\string\n\string"}
4254 \write\glswrite{item_1 \string"\string%\string\n\string"}
4255 \write\glswrite{item_2 \string"\string%\string\n\string"}
4256 \write\glswrite{item_01 \string"\string%\string\n\string"}
4257 \write\glswrite{item_x1
4258 \string"\string\relax \string\glresetentrylist\string\n
4259 \string"}
4260 \write\glswrite{item_12 \string"\string%\string\n\string"}
4261 \write\glswrite{item_x2
4262 \string"\string\relax \string\glresetentrylist\string\n
4263 \string"}

4264 \write\glswrite{delim_0 \string"\string\{\string
4265 \glssymbols\string\{\string\relax \string"}
4266 \write\glswrite{delim_1 \string"\string\{\string
4267 \glssymbols\string\{\string\relax \string"}
4268 \write\glswrite{delim_2 \string"\string\{\string
4269 \glssymbols\string\{\string\relax \string"}
4270 \write\glswrite{delim_t \string"\string\}\string\}\string"}
4271 \write\glswrite{delim_n \string"\string\delimN \string"}
4272 \write\glswrite{delim_r \string"\string\delimR \string"}
4273 \write\glswrite{headings_flag 1}
4274 \write\glswrite{heading_prefix
4275 \string"\string\glsgroupheading\string\{\string"}
4276 \write\glswrite{heading_suffix
4277 \string"\string\}\string\relax
4278 \string\glresetentrylist \string"}
4279 \write\glswrite{symhead_positive \string"glssymbols\string"}
4280 \write\glswrite{numhead_positive \string"glssymbols\string"}
4281 \write\glswrite{page_compositor \string"glscpositor\string"}
4282 \@gls@escbsdq\gls@suffixF
4283 \@gls@escbsdq\gls@suffixFF
4284 \ifx\gls@suffixF\@empty
4285 \else
4286 \write\glswrite{suffix_2p \string"\gls@suffixF\string"}
4287 \fi
4288 \ifx\gls@suffixFF\@empty
4289 \else
4290 \write\glswrite{suffix_3p \string"\gls@suffixFF\string"}
4291 \fi
4292 \closeout\glswrite
4293 \let\writeist\relax
4294 }
4295 \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`

```
4296 \newcommand{\noist}{%
```

Update attributes list

```
4297 \@gls@addpredefinedattributes
```

```
4298 \let\writeist\relax
```

```
4299 }
```

`\@makeglossary` is an internal command that takes an argument indicating the glossary type. This command will create the glossary file required by `makeindex` for the given glossary type, using the extension supplied by the `<out-ext>` parameter used in `\newglossary` (and it will also activate the `\glossary` command, and create the customized `.ist` `makeindex` style file).

Note that you can't use `\@makeglossary` for only some of the defined glossaries. You either need to have a `\makeglossary` for all glossaries or none (otherwise you will end up with a situation where \TeX is trying to write to a non-existent file). The relevant glossary must be defined prior to using `\@makeglossary`.

`\@makeglossary`

```
4300 \newcommand*{\@makeglossary}[1]{%
```

```
4301 \ifglossaryexists{#1}%
```

```
4302 {%
```

Only create a new write if `savewrites=false` otherwise create a token to collect the information.

```
4303 \ifglssavewrites
```

```
4304 \expandafter\newtoks\csname glo@#1@filetok\endcsname
```

```
4305 \else
```

```
4306 \expandafter\newwrite\csname glo@#1@file\endcsname
```

```
4307 \expandafter\@glsopenfile\csname glo@#1@file\endcsname{#1}%
```

```
4308 \fi
```

```
4309 \@gls@renewglossary
```

```
4310 \writeist
```

```
4311 }%
```

```
4312 {%
```

```
4313 \PackageError{glossaries}%
```

```
4314 {Glossary type ‘#1’ not defined}%
```

```
4315 {New glossaries must be defined before using \string\makeglossary}%
```

```
4316 }%
```

```
4317 }
```

`\@glsopenfile` Open write file associated with the given glossary.

```
4318 \newcommand*{\@glsopenfile}[2]{%
```

```
4319 \immediate\openout#1=\jobname.\csname @glotype@#2@out\endcsname
```

```
4320 \PackageInfo{glossaries}{Writing glossary file
```

```
4321 \jobname.\csname @glotype@#2@out\endcsname}%
```

```
4322 }
```

\@closegls

```

4323 \newcommand*{\@closegls}[1]{%
4324   \closeout\csname glo@#1@file\endcsname
4325 }
4326 %   \end{macrocode}
4327 %\end{macro}
4328 %
4329 %\begin{macro}{\@gls@automake}
4330 %\changes{4.08}{2014-07-30}{new}
4331 %   \begin{macrocode}
4332 \ifglxindy
4333   \newcommand*{\@gls@automake}[1]{%
4334     \ifglossaryexists{#1}
4335     {%
4336       \@closegls{#1}%
4337       \ifdefstring{glsorder}{letter}%
4338         {\def\@gls@order{-M ord/letorder }}%
4339         {\let\@gls@order\@empty}%
4340       \ifcsundef{xdy@#1@language}%
4341         {\let\@gls@langmod\@xdy@main@language}%
4342         {\letcs\@gls@langmod{xdy@#1@language}}%
4343       \edef\@gls@dothiswrite{\noexpand\write18{xindy
4344         -I xindy
4345         \@gls@order
4346         -L \@gls@langmod\space
4347         -M \@gls@istfilebase\space
4348         -C \@gls@codepage\space
4349         -t \jobname.\csuse{@glotype@#1@log}
4350         -o \jobname.\csuse{@glotype@#1@in}
4351         \jobname.\csuse{@glotype@#1@out}}}%
4352     }%
4353     \@gls@dothiswrite
4354   }%
4355   {%
4356     \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
4357   }%
4358 }
4359 \else
4360   \newcommand*{\@gls@automake}[1]{%
4361     \ifglossaryexists{#1}
4362     {%
4363       \@closegls{#1}%
4364       \ifdefstring{glsorder}{letter}%
4365         {\def\@gls@order{-l }}%
4366         {\let\@gls@order\@empty}%
4367       \edef\@gls@dothiswrite{\noexpand\write18{makeindex \@gls@order
4368         -s \istfilename\space
4369         -t \jobname.\csuse{@glotype@#1@log}
4370         -o \jobname.\csuse{@glotype@#1@in}

```



```

4371      \jobname.\csuse{@glotype@#1@out}}%
4372    }%
4373    \@gls@dothiswrite
4374  }%
4375  {%
4376    \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
4377  }%
4378 }
4379 \fi

```

`\nomakeglossaries` Issue warning that `\makeglossaries` hasn't been used.

```

4380 \newcommand*{\@warn@nomakeglossaries}{%

```

Only use this if warning if `\printglossary` has been used without `\makeglossaries`

```

4381 \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`

```

4382 \newcommand*{\makeglossaries}{%

```

Define the write used for style file also used for all other output files if `savewrites=true`.

```

4383 \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.

```

4384 \protected@write\@auxout{}{\string\providecommand\string\@glsorder[1]{}%

```

```

4385 \protected@write\@auxout{}{\string\providecommand\string\@istfilename[1]{}%

```

Write the name of the style file to the aux file (needed by `makeglossaries`)

```

4386 \protected@write\@auxout{}{\string\@istfilename{\istfilename}}%

```

```

4387 \protected@write\@auxout{}{\string\@glsorder{\glsorder}}

```

Iterate through each glossary type and activate it.

```

4388 \@for\@glo@type:=\@glo@types\do{%

```

```

4389   \ifthenelse{\equal{\@glo@type}{}}{}{}%

```

```

4390   \@makeglossary{\@glo@type}%

```

```

4391 }%

```

New glossaries must be created before `\makeglossaries` so disable `\newglossary`.

```

4392 \renewcommand*\newglossary[4][]{%

```

```

4393 \PackageError{glossaries}{New glossaries

```

```

4394 must be created before \string\makeglossaries}{You need

```

```

4395 to move \string\makeglossaries\space after all your

```

```

4396 \string\newglossary\space commands}}%

```

Any subsequence instances of this command should have no effect

```
4397 \let\@makeglossary\relax
4398 \let\makeglossary\relax
4399 \let\makeglossaries\relax
```

Disable all commands that have no effect after \makeglossaries

```
4400 \@disable@onlypremakeg
```

Allow see key:

```
4401 \let\gls@checkseeallowed\relax
```

Suppress warning about no \makeglossaries

```
4402 \let\warn@nomakeglossaries\relax
```

Activate warning about missing \printglossary

```
4403 \def\warn@noprntglossary{%
4404   \GlossariesWarningNoLine{No \string\printglossary\space
4405     or \string\printglossaries\space
4406     found.^^J(Remove \string\makeglossaries\space if you don't want
4407     any glossaries.)^^JThis document will not have a glossary}%
4408 }%
```

Declare list parser for \glsdisplaynumberlist

```
4409 \ifglssavenumberlist
4410   \edef\@gls@dodolistparser{\noexpand\DeclareListParser
4411     {\noexpand\glsnumlistparser}{\delimN}}}%
4412   \@gls@dodolistparser
4413 \fi
```

Prevent user from also using \makenoidxglossaries

```
4414 \let\makenoidxglossaries\@no@makeglossaries
```

Prohibit sort key in printgloss family:

```
4415 \renewcommand*{\@printgloss@setsort}{}%
4416 \let\@glo@assign@sortkey\@glo@no@assign@sortkey
4417 }%
```

Check the automake setting:

```
4418 \ifglssautomake
4419   \renewcommand*{\@gls@doautomake}{%
4420     \@for\@gls@type:=\@glo@types\do{%
4421       \ifdefempty{\@gls@type}{}%
4422       {\@gls@automake{\@gls@type}}%
4423     }%
4424   }%
4425 \fi
4426 }
```

Must occur in the preamble:

```
4427 \@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`

```
4428 \let\makeglossary\makeglossaries
```

If `\makeglossaries` hasn't been used, issue a warning. Also issue a warning
if neither `\printglossaries` nor `\printglossary` have been used.

```
4429 \AtEndDocument{%
4430   \warn@nomakeglossaries
4431   \warn@noprintglossary
4432 }
```

`\makenoidxglossaries` Analogous to `\makeglossaries` this activates the commands needed for `\printnoidxglossary`

```
4433 \newcommand*\makenoidxglossaries{%
```

Redefine empty glossary warning:

```
4434   \renewcommand{\@gls@noref@warn}[1]{%
4435     \GlossariesWarning{Empty glossary for
4436     \string\printnoidxglossary[type={##1}].
4437     Rerun may be required (or you may have forgotten to use
4438     commands like \string\gls).}%
4439   }%
```

Don't escape makeindex/xindy characters

```
4440   \let\@gls@checkmkidxchars\@gobble
```

Write glossary information to aux instead of glossary files

```
4441   \let\@@do@@wrglossary\gls@noidxglossary
```

Switch on group headings that use the character code:

```
4442   \let\@gls@getgrouptitle\@gls@noidx@getgrouptitle
```

Allow see key:

```
4443   \let\gls@checkseeallowed\relax
```

Redefine cross-referencing macro:

```
4444   \renewcommand{\@do@seeglossary}[2]{%
4445     \edef\@gls@label{\glsdetoklabel{##1}}%
4446     \protected@write\@auxout{%
4447       \string\@gls@reference
4448       {\csname glo@\@gls@label @type\endcsname}%
4449       {\@gls@label}%
4450       {%
4451         \string\glsseeformat##2}%
4452       }%
4453     }%
4454   }%
```

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.

```

4455 \AtBeginDocument
4456 {%
4457   \write\@auxout{\string\providecommand\string\@gls@reference[3]{}}%
4458 }%
```

Change warning about no glossares

```

4459 \def\warn@noprintglossary{%
4460   \GlossariesWarningNoLine{No \string\printnoidxglossary\space
4461     or \string\printnoidxglossaries ^^J
4462     found. (Remove \string\makenoidxglossaries\space if you
4463     don't want any glossaries.)^^JThis document will not have a glossary}%
4464 }%
```

Suppress warning about no \makeglossaries

```

4465 \let\warn@nomakeglossaries\relax
```

Prevent user from also using \makeglossaries

```

4466 \let\makeglossaries\@no@makeglossaries
```

Allow sort key in printgloss family:

```

4467 \renewcommand*{\@printgloss@setsort}{%
4468   \let\@glo@assign@sortkey\@glo@assign@sortkey
```

Initialise default sort order:

```

4469 \def\@glo@sorttype{\@glo@default@sorttype}%
4470 }%
```

All entries must be defined in the preamble:

```

4471 \renewcommand*\new@glossaryentry[2]{%
4472   \PackageError{glossaries}{Glossary entries must be
4473     defined in the preamble^^Jwhen you use
4474     \string\makenoidxglossaries}%
4475   {Either move your definitions to the preamble or use
4476     \string\makeglossaries}%
4477 }%
```

Redefine \glsentrynumberlist

```

4478 \renewcommand*\glsentrynumberlist[1]{%
4479   \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
4480   \ifdef\@gls@loclist
4481     {%
4482       \glsnoidxloclist{\@gls@loclist}%
4483     }%
4484     {%
4485       \ifglsentryexists{##1}%
4486         {%
4487           \GlossariesWarning{Missing location list for ‘##1’. Either
4488             a rerun is required or you haven't referenced the entry.}%
4489         }%
```

```

4490     {%
4491         \PackageError{glossaries}{Glossary entry ‘##1’ has not been
4492             defined.}{}%
4493     }%
4494 }%
4495 }%

Redefine \glsdisplaynumberlist
4496 \renewcommand*{\glsdisplaynumberlist}[1]{%
4497     \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
4498     \ifdef\@gls@loclist
4499     {%
4500         \def\@gls@noidxloclist@sep{%
4501             \def\@gls@noidxloclist@sep{%
4502                 \def\@gls@noidxloclist@sep{%
4503                     \glsnumlistsep
4504                 }%
4505                 \def\@gls@noidxloclist@finalsep{\glsnumlistlastsep}%
4506             }%
4507         }%
4508         \def\@gls@noidxloclist@finalsep{}%
4509         \def\@gls@noidxloclist@prev{}%
4510         \forlistloop{\glsnoidxdisplayloclisthandler}{\@gls@loclist}%
4511         \@gls@noidxloclist@finalsep
4512         \@gls@noidxloclist@prev
4513     }%
4514     {%
4515         ??\ifglsentryexists{##1}%
4516         {%
4517             \GlossariesWarning{Missing location list for ‘##1’. Either
4518                 a rerun is required or you haven’t referenced the entry.}%
4519         }%
4520         {%
4521             \PackageError{glossaries}{Glossary entry ‘##1’ has not been
4522                 defined.}{}%
4523         }%
4524     }%
4525 }%

```

Provide a generic way of iterating through the number list:

```

4526 \renewcommand*{\glsnumberlistloop}[3]{%
4527     \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
4528     \let\@gls@org@glsnoidxdisplayloc\glsnoidxdisplayloc
4529     \let\@gls@org@glsseeformat\glsseeformat
4530     \let\glsnoidxdisplayloc##2\relax
4531     \let\glsseeformat##3\relax
4532     \ifdef\@gls@loclist
4533     {%
4534         \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
4535     }%

```

```

4536   {%
4537       \ifglsentryexists{##1}%
4538       {%
4539           \GlossariesWarning{Missing location list for ‘##1’. Either
4540               a rerun is required or you haven’t referenced the entry.}%
4541       }%
4542       {%
4543           \PackageError{glossaries}{Glossary entry ‘##1’ has not been
4544               defined.}{}%
4545       }%
4546   }%
4547   \let\glsnoidxdisplayloc\@gls@org\glsnoidxdisplayloc
4548   \let\glsseeformat\@gls@org\glsseeformat
4549 }%

```

Modify sanitize sort function

```

4550 \let\@gls@sanitizesort\@gls@noidx@sanitizesort
4551 \let\@gls@nosanitizesort\@gls@noidx@nosanitizesort
4552 \@gls@noidx@setsanitizesort
4553 }

```

Preamble-only command:

```

4554 \onlypreamble{\makenoidxglossaries}

```

`\glsnumberlistloop` `\glsnumberlistloop{<label>}{<handler>}`

```

4555 \newcommand*{\glsnumberlistloop}[2]{%
4556     \PackageError{glossaries}{\string\glsnumberlistloop\space
4557         only works with \string\makenoidxglossaries}{}%
4558 }

```

`numberlistloophandler` Handler macro for `\glsnumberlistloop`. (The argument should be in the form `\glsnoidxdisplayloc{<prefix>}{<counter>}{<format>}{<n>}`)

```

4559 \newcommand*{\glsnoidxnumberlistloophandler}[1]{%
4560     #1%
4561 }

```

`\@no@makeglossaries` Can’t use both `\makeglossaries` and `\makenoidxglossaries`

```

4562 \newcommand*{\@no@makeglossaries}{%
4563     \PackageError{glossaries}{You can’t use both
4564         \string\makeglossaries\space and \string\makenoidxglossaries}%
4565     {Either use one or other (or none) of those commands but not both
4566         together.}%
4567 }

```

`\@gls@noref@warn` Warning when no instances of `\@gls@reference` found.

```

4568 \newcommand{\@gls@noref@warn}[1]{%
4569     \GlossariesWarning{\string\makenoidxglossaries\space

```

```

4570   is required to make \string\printnoidxglossary[type={#1}] work}%
4571 }

```

`\gls@noidxglossary` Write the glossary information to the aux file:

```

4572 \newcommand*{\gls@noidxglossary}{%
4573   \protected@write\@auxout{}{%
4574     \string\@gls@reference
4575     {\csname glo@\@gls@label @type\endcsname}%
4576     {\@gls@label}%
4577     {\string\glsnoidxdisplayloc
4578      {\@glo@counterprefix}%
4579      {\@gls@counter}%
4580      {\@glsnumberformat}%
4581      {\@glslocref}%
4582     }%
4583   }%
4584 }

```

1.13 Writing information to associated files

`\istfile` Deprecated.

```

4585 \def\istfile{\glswrite}

```

At the end of the document, the files should be created if `savewrites=true`.

```

4586 \AtEndDocument{%
4587   \glswritefiles
4588 }

```

`\@glswritefiles` Only write the files if `savewrites=true`

```

4589 \newcommand*{\@glswritefiles}{%

```

Iterate through all the glossaries

```

4590   \foralllglossaries{\@glo@type}{%

```

Check for empty glossaries (patch provided by Patrick Häcker)

```

4591     \ifcsundef{glo@\@glo@type @filetok}%
4592     {%
4593       \def\gls@tmp{}%
4594     }%
4595     {%
4596       \edef\gls@tmp{\expandafter\the
4597         \csname glo@\@glo@type @filetok\endcsname}%
4598     }%
4599     \ifx\gls@tmp\@empty
4600       \ifx\@glo@type\glsdefaulttype
4601         \GlossariesWarningNoLine{Glossary '\@glo@type' has no
4602           entries.^^JRemember to use package option 'nomain' if
4603 you
4604           don't want to^^Juse the main glossary}%

```

```

4605         \else
4606             \GlossariesWarningNoLine{Glossary ‘\@glo@type’ has no
4607                 entries}%
4608         \fi
4609     \else
4610         \@glsopenfile{\glswrite}{\@glo@type}%
4611         \immediate\write\glswrite{%
4612             \expandafter\the
4613             \csname glo@\@glo@type @filetok\endcsname}%
4614         \immediate\closeout\glswrite
4615     \fi
4616 }%
4617 }

```

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`

```

4618 \if@gls@docloaded
4619 \else
4620   \renewcommand*{\glossary}[1][main]{\gls@glossary{#1}}
4621 \fi

```

The associated number should be stored in `\theglsentrycounter` before using `\gls@glossary`.

`\gls@glossary`

```

4622 \newcommand*{\gls@glossary}[1]{%
4623   \@gls@glossary{#1}%
4624 }

```

`\@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.

```

4625 \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.

\@gls@renewglossary

```

4626 \newcommand{\@gls@renewglossary}{%
4627   \gdef\@gls@glossary##1{\@bsphack\begingroup\gls@wrglossary{##1}}%
4628   \let\@gls@renewglossary\@empty
4629 }
```

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

```

4630 \newcommand*{\gls@wrglossary}[2]{%
4631   \ifglssavewrites
4632     \protected@edef\@gls@tmp{\the\csname glo@#1@filetok\endcsname#2}%
4633     \expandafter\global\expandafter\csname glo@#1@filetok\endcsname
4634       \expandafter{\@gls@tmp^^J}%
4635   \else
4636     \ifcsdef{glo@#1@file}%
4637     {%
4638       \expandafter\protected@write\csname glo@#1@file\endcsname{%
4639         \gls@disablepagerefexpansion}{#2}%
4640     }%
4641     {%
4642       \ifignoredglossary{#1}{}%
4643       {%
4644         \GlossariesWarning{No file defined for glossary ‘#1’}%
4645       }%
4646     }%
4647   \fi
4648   \endgroup\@esphack
4649 }
```

\@do@wrglossary

```

4650 \newcommand*{\@do@wrglossary}[1]{%
4651   \ifglsindexonlyfirst
4652     \ifglsused{#1}{\@do@wrglossary{#1}}%
4653   \else
4654     \@do@wrglossary{#1}%
4655   \fi
4656 }
```

@protected@pagefmts List of page formats to be protected against expansion.

```

4657 \newcommand{\gls@protected@pagefmts}{%
4658   \gls@numberpage,\gls@alphpage,\gls@Alphpage,\gls@romanpage,\gls@Romanpage%
4659 }
```

blepagerefexpansion

```

4660 \newcommand*{\gls@disablepagerefexpansion}{%
```

```

4661 \@for\@gls@this:=\gls@protected@pagefmts\do
4662 {%
4663     \expandafter\let\@gls@this\relax
4664 }%
4665 }

```

`\gls@alphpage`

```
4666 \newcommand*\gls@alphpage{\@alph\c@page}
```

`\gls@Alphpage`

```
4667 \newcommand*\gls@Alphpage{\@Alph\c@page}
```

`\gls@numberpage`

```
4668 \newcommand*\gls@numberpage{\number\c@page}
```

`\gls@romanpage`

```
4669 \newcommand*\gls@romanpage{\romannumeral\c@page}
```

`\gls@Romanpage`

```
4670 \newcommand*\gls@Romanpage{\@Roman\c@page}
```

`\glsaddprotectedpagefmt`

```
\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).

```

4671 \newcommand*\glsaddprotectedpagefmt[1]{%
4672     \eappto\gls@protected@pagefmts{\expandonce{\csname gls#1page\endcsname}}%
4673     \csedef{gls#1page}{\expandonce{\csname#1\endcsname}\noexpand\c@page}%
4674     \eappto\@wrglossarynumberhook{%
4675         \noexpand\let\expandonce{\csname org@gls#1\endcsname}%
4676         \expandonce{\csname#1\endcsname}%
4677         \noexpand\def\expandonce{\csname#1\endcsname}{%
4678             \noexpand\@wrglossary@pageformat
4679             \expandonce{\csname gls#1page\endcsname}%
4680             \expandonce{\csname org@gls#1\endcsname}%
4681         }%
4682     }%
4683 }

```

`\@wrglossarynumberhook` Hook used by `\@do@wrglossary`

```
4684 \newcommand*\@wrglossarynumberhook{}
```

`\@wrglossary@pageformat`

```

4685 \newcommand{\@wrglossary@pageformat}[3]{%
4686     \ifx#3\c@page #1\else #2#3\fi
4687 }

```

\@@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.

```
4688 \newcommand*{\@@do@wrglossary}[1]{%
4689   \begingroup
```

First a bit of hackery to prevent premature expansion of \@c@page. Store original definitions:

```
4690   \let\orgthe\the
4691   \let\orgnumber\number
4692   \let\orgromannumeral\romannumeral
4693   \let\orgalph\@alph
4694   \let\orgAlph\@Alph
4695   \let\orgRoman\@Roman
```

Redefine:

```
4696   \def\the##1{%
4697     \ifx##1\c@page \gls@numberpage\else\orgthe##1\fi}%
4698   \def\number##1{%
4699     \ifx##1\c@page \gls@numberpage\else\orgnumber##1\fi}%
4700   \def\romannumeral##1{%
4701     \ifx##1\c@page \gls@romanpage\else\orgromannumeral##1\fi}%
4702   \def\@Roman##1{%
4703     \ifx##1\c@page \gls@Romanpage\else\orgRoman##1\fi}%
4704   \def\@alph##1{%
4705     \ifx##1\c@page \gls@alphpage\else\orgalph##1\fi}%
4706   \def\@Alph##1{%
4707     \ifx##1\c@page \gls@Alphpage\else\orgAlph##1\fi}%
```

Add hook to allow for other number formats:

```
4708   \@wrglossarynumberhook
```

Prevent expansion:

```
4709   \gls@disablepagerefexpansion
```

Now store location in \@glslocref:

```
4710   \protected\xdef\@glslocref{\theHglentrycounter}%
4711   \endgroup
```

Escape any special characters

```
4712   \@gls@checkmkidxchars\@glslocref
```

Check if the hyper-location is the same as the location and set the hyper prefix.

```
4713   \expandafter\ifx\theHglentrycounter\theHglentrycounter\relax
4714   \def\@glo@counterprefix{%
4715   \else
4716     \protected\edef\@glsHlocref{\theHglentrycounter}%
4717     \@gls@checkmkidxchars\@glsHlocref
4718     \edef\@do@gls@getcounterprefix{\noexpand\@gls@getcounterprefix
4719       {\@glslocref}{\@glsHlocref}%
4720     }%
```

```

4721 \do@glsglscounterprefix
4722 \fi

```

De-tok label if required

```

4723 \edef\@glsglscounterlabel{\glsglscounterlabel{#1}}%

```

Write the information to file:

```

4724 \do@do@wrglossary
4725 }

```

\do@do@wrglossary

```

4726 \newcommand*\do@do@wrglossary{%

```

Determine whether to use xindy or makeindex syntax

```

4727 \ifglsglscounter

```

Need to determine if the formatting information starts with a (or) indicating a range.

```

4728 \expandafter\@glo@check@mkidxrangear\@glsglscounterformat\@nil
4729 \def\@glo@range{%
4730 \expandafter\if\@glo@prefix(\relax
4731 \def\@glo@range{:open-range}%
4732 \else
4733 \expandafter\if\@glo@prefix)\relax
4734 \def\@glo@range{:close-range}%
4735 \fi
4736 \fi

```

Write to the glossary file using xindy syntax.

```

4737 \glsglsglossary{\csname glo@\@glsglscounter @type\endcsname}{%
4738 (indexentry :tkey (\csname glo@\@glsglscounter @index\endcsname)
4739 :locoref \string"\@glo@counterprefix}\@glsglscounter\string" %
4740 :attr \string"\@glsglscounter\@glo@sufffix\string"
4741 \@glo@range
4742 )
4743 }%
4744 \else

```

Convert the format information into the format required for makeindex

```

4745 \set@glsglscounterformat{\@glo@numfmt}{\@glsglscounter}{\@glsglscounterformat}%
4746 {\@glo@counterprefix}%

```

Write to the glossary file using makeindex syntax.

```

4747 \glsglsglossary{\csname glo@\@glsglscounter @type\endcsname}{%
4748 \string\glossaryentry{\csname glo@\@glsglscounter @index\endcsname
4749 \@glsglscounter\@glo@numfmt}\@glsglscounter}%
4750 \fi
4751 }

```

ls@getcounterprefix 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.)

```

4752\newcommand*\@gls@get@counterprefix[2]{%
4753  \edef\@gls@thisloc{#1}\edef\@gls@thisHloc{#2}%
4754  \ifx\@gls@thisloc\@gls@thisHloc
4755    \def\@glo@counterprefix{}%
4756  \else
4757    \def\@gls@get@counterprefix##1.#1##2\end@getprefix{%
4758      \def\@glo@tmp{##2}%
4759      \ifx\@glo@tmp\@empty
4760        \def\@glo@counterprefix{}%
4761      \else
4762        \def\@glo@counterprefix{##1}%
4763      \fi
4764    }%
4765    \@gls@get@counterprefix#2.#1\end@getprefix

Warn if no prefix can be formed.

4766  \ifx\@glo@counterprefix\@empty
4767    \GlossariesWarning{Hyper target ‘#2’ can’t be formed by
4768      prefixing^^Jlocation ‘#1’. You need to modify the
4769      definition of \string\theH\@gls@counter^^Jotherwise you
4770      will get the warning: “‘name{\@gls@counter.#1}’ has been^^J
4771      referenced but does not exist”}%
4772  \fi
4773 \fi
4774 }
```

1.14 Glossary Entry Cross-References

`\do@seeglossary` Write the glossary entry with a cross reference. The first argument is the entry's label, the second must be in the form `[\langle tag \rangle]{\langle list \rangle}`, where `\langle tag \rangle` is a tag such as “see” and `\langle list \rangle` is a list of labels.

```

4775 \newcommand{\@do@seeglossary}[2]{%
4776 \def\@gls@xref{#2}%
4777 \@onelevel@sanitize\@gls@xref
4778 \@gls@checkmkidxchars\@gls@xref
4779 \ifglsxindy
4780   \gls@glossary{\csname glo@#1type\endcsname}{%
4781     (indexentry
4782       :key (\csname glo@#1index\endcsname)
4783       :xref (\string"\@gls@xref\string")
4784       :attr \string"see\string"
4785     )
4786   }%
4787 \else
4788   \gls@glossary{\csname glo@#1type\endcsname}{%
4789     \string\glossaryentry{\csname glo@#1index\endcsname

```

```

4790 \@gls@encapchar glsseeformat\@gls@xref}{Z}}%
4791 \fi
4792 }

```

`\@gls@fixbraces` If no optional argument is specified, list needs to be enclosed in a set of braces.

```

4793 \def\@gls@fixbraces#1#2#3\@nil{%
4794   \ifx#2[\relax
4795     \@gls@fixbraces#1#2#3\@end@fixbraces
4796   \else
4797     \def#1{{#2#3}}%
4798   \fi
4799 }

```

`\@@gls@fixbraces`

```

4800 \def\@@gls@fixbraces#1[#2]#3\@end@fixbraces{%
4801   \def#1{[#2]{#3}}%
4802 }

```

`\glssee` `\glssee{<label>}{<cross-ref list>}`

```

4803 \DeclareRobustCommand*\glssee[3][\seename]{%
4804   \@do@seeglossary{#2}{[#1]{#3}}}
4805 \newcommand*\@glssee[3][\seename]{%
4806   \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.

```

4807 \DeclareRobustCommand*\glsseeformat[3][\seename]{%
4808   \emph{#1} \glsseelist{#2}}

```

`\glsseelist` `\glsseelist{<list>}` formats list of entry labels.

```

4809 \DeclareRobustCommand*\glsseelist[1]{%

```

If there is only one item in the list, set the last separator to do nothing.

```

4810 \let\@gls@dolast\relax

```

Don’t display separator on the first iteration of the loop

```

4811 \let\@gls@donext\relax

```

Iterate through the labels

```

4812 \@for\@gls@thislabel:=#1\do{%

```

Check if on last iteration of loop

```

4813   \ifx\@xfor@nextelement\@nnil

```

```

4814     \@gls@dolast

```

```

4815   \else

```

```

4816     \@gls@donext

```

```

4817   \fi

```

Display the entry for this label. (Expanding label as it’s a temporary control sequence that’s used elsewhere.)

```

4818   \expandafter\glsseeitem\expandafter{\@gls@thislabel}%

```

Update separators

```
4819 \let\@gls@dolast\glsseelastsep
4820 \let\@gls@donext\glsseesep
4821 }%
4822 }
```

`\glsseelastsep` Separator to use between penultimate and ultimate entries in a cross-referencing list.

```
4823 \newcommand*{\glsseelastsep}{\space\andname\space}
```

`\glsseesep` Separator to use between entires in a cross-referencing list.

```
4824 \newcommand*{\glsseesep}{, }
```

`\glsseeitem` `\glsseeitem{<label>}` formats individual entry in a cross-referencing list.

```
4825 \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.)

```
4826 \newcommand*{\glsseeitemformat}[1]{\glsentrytext{#1}}
```

1.15 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.

`\gls@save@numberlist` Provide command to store number list.

```
4827 \newcommand*{\gls@save@numberlist}[1]{%
4828   \ifglssavenumberlist
4829     \toks@{#1}%
4830     \edef\@do@writeaux@info{%
4831       \noexpand\csgdef{glo@\glscurrententrylabel @numberlist}{\the\toks@}%
4832     }%
4833     \@onelevel@sanitize\@do@writeaux@info
4834     \protected@write\@auxout{}\@do@writeaux@info}%
4835   \fi
4836 }
```

`\warn@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.)

```
4837 \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.

```
4838 \ifcsundef{printglossary}{}%  
4839 {%
```

If `\printglossary` is already defined, issue a warning and undefine it.

```
4840 \@gls@warnonglossdefined  
4841 \undef\printglossary  
4842 }
```

`\printglossary` has an optional argument. The default value is to set the glossary type to the main glossary.

```
4843 \newcommand*{\printglossary}[1][type=\glsdefaulttype]{%  
4844 \@printglossary{#1}{\@print@glossary}%  
4845 }
```

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`

```
4846 \newcommand*{\printglossaries}{%  
4847 \forallglossaries{\@glo@type}{\printglossary[type=\@glo@type]}}%  
4848 }
```

`\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.

```
4849 \newcommand*{\printnoidxglossary}[1][type=\glsdefaulttype]{%  
4850 \@printglossary{#1}{\@print@noidx@glossary}%  
4851 }
```

`\printnoidxglossaries` Analogous to `\printglossaries`

```
4852 \newcommand*{\printnoidxglossaries}{%  
4853 \forallglossaries{\@glo@type}{\printnoidxglossary[type=\@glo@type]}}%  
4854 }
```

`@printgloss@setsort` Initialise to do nothing.

```
4855 \newcommand*{\@printgloss@setsort}{}
```

`\@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.

```
4856 \newcommand{\@printglossary}[2]{%
```


Set up defaults.

```
4857 \def\@glo@type{\glsdefaulttype}%
4858 \def\glossarytitle{\csname @glo@type\@glo@type @title\endcsname}%

4859 \def\glossarytoctitle{\glossarytitle}%
4860 \let\org@glossarytitle\glossarytitle
4861 \def\@glossarystyle{}%
4862 \def\gls@dotoc@title{\gls@dotoc@title{\@glo@type}}%
```

Store current value of `\glossaryentrynumbers`. (This may be changed via the optional argument)

```
4863 \let\@org@glossaryentrynumbers\glossaryentrynumbers
```

Localise the effects of the optional argument

```
4864 \bgroup
```

Activate or deactivate sort key:

```
4865 \@printgloss@setsort
```

Determine settings specified in the optional argument.

```
4866 \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)

```
4867 \ifx\glossarytitle\org@glossarytitle
4868 \else
4869 \expandafter\let\csname @glo@type\@glo@type @title\endcsname
4870 \glossarytitle
4871 \fi
```

Allow a high-level user command to indicate the current glossary

```
4872 \let\currentglossary\@glo@type
```

Enable individual number lists to be suppressed.

```
4873 \let\org@glossaryentrynumbers\glossaryentrynumbers
4874 \let\glsnonextpages\@glsnonextpages
```

Enable individual number list to be activated:

```
4875 \let\glsnextpages\@glsnextpages
```

Enable suppression of description terminators.

```
4876 \let\nopostdesc\@nopostdesc
```

Set up the entry for the TOC

```
4877 \gls@dotoc@title
```

Set the glossary style

```
4878 \@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):

```
4879 \let\gls@org@glossaryentryfield\glossentry
```

```

4880 \let\gls@org@glossarysubentryfield\subglossentry
4881 \renewcommand{\glossentry}[1]{%
4882   \xdef\glscurrententrylabel{\glsdetoklabel{##1}}%
4883   \gls@org@glossaryentryfield{##1}%
4884 }%
4885 \renewcommand{\subglossentry}[2]{%
4886   \xdef\glscurrententrylabel{\glsdetoklabel{##2}}%
4887   \gls@org@glossarysubentryfield{##1}{##2}%
4888 }%

```

Now do the handler macro that deals with the actual glossary:

```

4889   #2%

End the current scope
4890 \egroup

Reset \glossaryentrynumbers
4891 \global\let\glossaryentrynumbers\@org@glossaryentrynumbers

Suppress warning about no \printglossary
4892 \global\let\warn@noprintglossary\relax
4893 }

```

\@print@glossary Internal workings of \printglossary dealing with reading the external file.

```

4894 \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.)

```

4895 \makeatletter

```

Input the glossary file, if it exists.

```

4896 \@input{\jobname.\csname @glotype@\@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.

```

4897 \IfFileExists{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
4898 {}%
4899 {\null}%

```

If xindy is being used, need to write the language dependent information to the .aux file for makeglossaries.

```

4900 \ifglxindy
4901   \ifcsundef{@xdy@\@glo@type @language}%
4902   {%
4903     \edef\do@auxoutstuff{%
4904       \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.

```

4905         \noexpand\immediate\noexpand\write\@auxout{%
4906             \string\providecommand\string\@xdylanguage[2]{}%
4907         \noexpand\immediate\noexpand\write\@auxout{%
4908             \string\@xdylanguage{\@glo@type}{\@xdy@main@language}}%
4909     }%
4910 }%
4911 }%
4912 {%
4913     \edef\@do@auxoutstuff{%
4914         \noexpand\AtEndDocument{%
4915             \noexpand\immediate\noexpand\write\@auxout{%
4916                 \string\providecommand\string\@xdylanguage[2]{}%
4917             \noexpand\immediate\noexpand\write\@auxout{%
4918                 \string\@xdylanguage{\@glo@type}{\csname @xdy@\@glo@type
4919                     @language\endcsname}}%
4920         }%
4921     }%
4922 }%
4923 \do@auxoutstuff
4924 \edef\@do@auxoutstuff{%
4925     \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.

```

4926     \noexpand\immediate\noexpand\write\@auxout{%
4927         \string\providecommand\string\@gls@codepage[2]{}%
4928     \noexpand\immediate\noexpand\write\@auxout{%
4929         \string\@gls@codepage{\@glo@type}{\gls@codepage}}%
4930     }%
4931 }%
4932 \do@auxoutstuff
4933 \fi

```

Activate warning if \makeglossaries hasn't been used.

```

4934 \renewcommand*{\@warn@nomakeglossaries}{%
4935     \GlossariesWarningNoLine{\string\makeglossaries\space
4936         hasn't been used,^^Jthe glossaries will not be updated}%
4937 }%
4938 }

```

The sort macros all have the syntax:

`\@glo@sortmacro@<order>{<type>}`

where <order> is the sort order as specified by the sort key and <type> is the glossary type. (The referenced entry list is stored in \@glsref@<type>). The actual sorting is done by \@glo@sortentries{<handler>}{<type>}.

\@glo@sortentries

```
4939 \newcommand*{\@glo@sortentries}[2]{%
4940   \def\@glo@sortinglist{}%
4941   \def\@glo@sortinghandler{#1}%
4942   \edef\@glo@type{#2}%
4943   \forlistcsloop{\@glo@do@sortentries}{\@glsref@#2}%
4944   \csdef{\@glsref@#2}{}%
4945   \@for\@this@label:=\@glo@sortinglist\do{%
      Has this entry already been added?
4946     \xifinlistcs{\@this@label}{\@glsref@#2}%
4947     {}%
4948     {%
4949       \listcsxadd{\@glsref@#2}{\@this@label}%
4950     }%
4951     \ifcsdef{\@glo@sortingchildren@\@this@label}%
4952     {%
4953       \@glo@addchildren{#2}{\@this@label}%
4954     }%
4955     {}%
4956   }%
4957 }
```

\@glo@addchildren

\@glo@addchildren{<type>}{<parent>}

```
4958 \newcommand*{\@glo@addchildren}[2]{%
      Scope to allow nesting.
4959   \bgroup
4960   \letcs{\@glo@childlist}{\@glo@sortingchildren@#2}%
4961   \@for\@this@childlabel:=\@glo@childlist\do
4962   {%
      Check this label hasn't already been added.
4963     \xifinlistcs{\@this@childlabel}{\@glsref@#1}%
4964     {}%
4965     {%
4966       \listcsxadd{\@glsref@#1}{\@this@childlabel}%
4967     }%
      Does this child have children?
4968     \ifcsdef{\@glo@sortingchildren@\@this@childlabel}%
4969     {%
4970       \@glo@addchildren{#1}{\@this@childlabel}%
4971     }%
4972     {%
4973     }%
4974   }%
4975   \egroup
4976 }
```

@glo@do@sortentries

```
4977 \newcommand*{\@glo@do@sortentries}[1]{%
4978   \ifglshasparent{#1}%
4979   {%
```

This entry has a parent, so add it to the child list

```
4980   \edef\@glo@parent{\csuse{glo@\glstoklabel{#1}@parent}}%
4981   \ifcsundef{\@glo@sortingchildren@\@glo@parent}%
4982   {%
4983     \csdef{\@glo@sortingchildren@\@glo@parent}{}%
4984   }%
4985   {}%
4986   \expandafter\@glo@sortedinsert
4987     \csname @glo@sortingchildren@\@glo@parent\endcsname{#1}%
```

Has the parent been added?

```
4988   \xifinlistcs{\@glo@parent}{\glstoklabel{\@glo@parent}}%
4989   {%
```

Yes, it has so do nothing.

```
4990   }%
4991   {}%
```

No, it hasn't so add it now.

```
4992   \expandafter\@glo@do@sortentries\expandafter{\@glo@parent}%
4993   }%
4994   }%
4995   {}%
4996   \@glo@sortedinsert{\@glo@sortinglist}{#1}%
4997   }%
4998 }
```

\@glo@sortedinsert `\@glo@sortedinsert{<list>}{<entry label>}`

Insert into list.

```
4999 \newcommand*{\@glo@sortedinsert}[2]{%
5000   \dtl@insertinto{#2}{#1}{\@glo@sortinghandler}%
5001   }%
```

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).

@glo@sorthandler@word

```
5002 \newcommand*{\@glo@sorthandler@word}[2]{%
5003   \letcs\@gls@sortA{glo@\glstoklabel{#1}@sort}%
5004   \letcs\@gls@sortB{glo@\glstoklabel{#2}@sort}%
5005   \edef\@glo@do@compare{%
5006     \noexpand\dtlwordindexcompare{\noexpand\dtl@sortresult}%
```

```

5007     {\expandonce\@gls@sort@B}%
5008     {\expandonce\@gls@sort@A}%
5009 }%
5010 \glo@do@compare
5011 }

```

@sorthandler@letter

```

5012 \newcommand*{\@glo@sorthandler@letter}[2]{%
5013   \letcs\@gls@sort@A{glo\@glsdetoklabel{#1}@sort}%
5014   \letcs\@gls@sort@B{glo\@glsdetoklabel{#2}@sort}%
5015   \edef\glo@do@compare{%
5016     \noexpand\dtlletterindexcompare{\noexpand\dtl@sortresult}%
5017     {\expandonce\@gls@sort@B}%
5018     {\expandonce\@gls@sort@A}%
5019   }%
5020   \glo@do@compare
5021 }

```

lo@sorthandler@case Case-sensitive sort.

```

5022 \newcommand*{\@glo@sorthandler@case}[2]{%
5023   \letcs\@gls@sort@A{glo\@glsdetoklabel{#1}@sort}%
5024   \letcs\@gls@sort@B{glo\@glsdetoklabel{#2}@sort}%
5025   \edef\glo@do@compare{%
5026     \noexpand\dtlcompare{\noexpand\dtl@sortresult}%
5027     {\expandonce\@gls@sort@B}%
5028     {\expandonce\@gls@sort@A}%
5029   }%
5030   \glo@do@compare
5031 }

```

@sorthandler@nocase Case-insensitive sort.

```

5032 \newcommand*{\@glo@sorthandler@nocase}[2]{%
5033   \letcs\@gls@sort@A{glo\@glsdetoklabel{#1}@sort}%
5034   \letcs\@gls@sort@B{glo\@glsdetoklabel{#2}@sort}%
5035   \edef\glo@do@compare{%
5036     \noexpand\dtlicompare{\noexpand\dtl@sortresult}%
5037     {\expandonce\@gls@sort@B}%
5038     {\expandonce\@gls@sort@A}%
5039   }%
5040   \glo@do@compare
5041 }

```

@glo@sortmacro@word Sort macro for ‘word’

```

5042 \newcommand*{\@glo@sortmacro@word}[1]{%
5043   \ifdefstring{\@glo@default@sorttype}{standard}%
5044   {%
5045     \@glo@sortentries{\@glo@sorthandler@word}{#1}%
5046   }%
5047   {%

```

```

5048 \PackageError{glossaries}{Conflicting sort options:^^J
5049 \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5050 \string\printnoidxglossary[sort=word]}{}%
5051 }%
5052 }

\lo@sortmacro@letter Sort macro for 'letter'

5053 \newcommand*\@glo@sortmacro@letter[1]{%
5054 \ifdefstring{\@glo@default@sorttype}{standard}%
5055 {%
5056 \@glo@sortentries{\@glo@sorthandler@letter}{#1}%
5057 }%
5058 {%
5059 \PackageError{glossaries}{Conflicting sort options:^^J
5060 \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5061 \string\printnoidxglossary[sort=letter]}{}%
5062 }%
5063 }

\sortmacro@standard Sort macro for 'standard'. (Use either 'word' or 'letter' order.)

5064 \newcommand*\@glo@sortmacro@standard[1]{%
5065 \ifdefstring{\@glo@default@sorttype}{standard}%
5066 {%
5067 \ifcsdef{@glo@sorthandler@\glsorder}%
5068 {%
5069 \@glo@sortentries{\csuse{@glo@sorthandler@\glsorder}}{#1}%
5070 }%
5071 {%
5072 \PackageError{glossaries}{Unknown sort handler '@glsorder'}{}%
5073 }%
5074 }%
5075 {%
5076 \PackageError{glossaries}{Conflicting sort options:^^J
5077 \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5078 \string\printnoidxglossary[sort=standard]}{}%
5079 }%
5080 }

\glo@sortmacro@case Sort macro for 'case'

5081 \newcommand*\@glo@sortmacro@case[1]{%
5082 \ifdefstring{\@glo@default@sorttype}{standard}%
5083 {%
5084 \@glo@sortentries{\@glo@sorthandler@case}{#1}%
5085 }%
5086 {%
5087 \PackageError{glossaries}{Conflicting sort options:^^J
5088 \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5089 \string\printnoidxglossary[sort=case]}{}%
5090 }%

```

```

5091 }

\lo@sortmacro@nocase Sort macro for ‘nocase’
5092 \newcommand*{\@glo@sortmacro@nocase}[1]{%
5093   \ifdefstring{\@glo@default@sorttype}{standard}%
5094   {%
5095     \@glo@sortentries{\@glo@sorthandler@nocase}{#1}%
5096   }%
5097   {%
5098     \PackageError{glossaries}{Conflicting sort options:^^J
5099       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5100       \string\printnoidxglossary[sort=nocase]}{}%
5101   }%
5102 }

\@glo@sortmacro@def Sort macro for ‘def’. The order of definition is given in \glo@list@<type>.
5103 \newcommand*{\@glo@sortmacro@def}[1]{%
5104   \def\@glo@sortinglist{}%
5105   \for@gl@sentries[#1]{\@gls@thislabel}%
5106   {%
5107     \xifinlistcs{\@gls@thislabel}{\@glsref@#1}%
5108     {%
5109       \listeadadd{\@glo@sortinglist}{\@gls@thislabel}%
5110     }%
5111     {%
5112       }%
5113     }%
5114     \cslet{\@glsref@#1}{\@glo@sortinglist}%
5115   }

Hasn’t been referenced.

\lo@sortmacro@def@do This won’t include parent entries that haven’t been referenced.
5116 \newcommand*{\@glo@sortmacro@def@do}[1]{%
5117   \ifinlistcs{#1}{\@glsref@\@glo@type}%
5118   {}%
5119   {%
5120     \listcsadd{\@glsref@\@glo@type}{#1}%
5121   }%
5122   \ifcsdef{\@glo@sortingchildren@#1}%
5123   {%
5124     \@glo@addchildren{\@glo@type}{#1}%
5125   }%
5126   {}%
5127 }

\@glo@sortmacro@use Sort macro for ‘use’. (No sorting is required, as the entries are already in order
of use, so do nothing.)
5128 \newcommand*{\@glo@sortmacro@use}[1]{%

```


`\print@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.

```

5129 \newcommand*{\@print@noidx@glossary}{%
5130   \ifcsdef{@glsref@{\glo@type}}%
5131   {%
      Sort the entries:
5132     \ifcsdef{@glo@sortmacro@{\glo@sorttype}}%
5133     {%
5134       \csuse{@glo@sortmacro@{\glo@sorttype}}{\@glo@type}%
5135     }%
5136     {%
5137       \PackageError{glossaries}{Unknown sort handler ‘\@glo@sorttype’}{}%
5138     }%

```

Do the glossary heading and preamble

```

5139   \glossarysection[\glossarytoctitle]{\glossarytitle}%
5140   \glossarypreamble
5141   \begin{theglossary}%
5142   \glossaryheader
5143   \glsresetentrylist
5144   \def\@gls@currentlettergroup{}%

```

Iterate through the entries.

```

5145   \forlistcsloop{\@gls@noidx@do}{\@glsref@{\glo@type}}%

```

Finally end the glossary and do the postamble:

```

5146   \end{theglossary}%
5147   \glossarypostamble
5148 }%
5149 {%
5150   \@gls@noref@warn{\@glo@type}%
5151 }%
5152 }

```

`\glo@grabfirst`

```

5153 \def\glo@grabfirst#1#2\@nil{%
5154   \def\@gls@firsttok{#1}%
5155   \ifdefempty\@gls@firsttok
5156   {%
5157     \def\@glo@thislettergrp{0}%
5158   }%
5159   {%

```

Sanitize it:

```

5160   \@onelevel@sanitize\@gls@firsttok

```

Fetch the first letter:

```

5161 \expandafter\@glo@grabfirst\@gls@firsttok{}\}\@nil
5162 }%
5163 }

```

\@glo@grabfirst

```

5164 \def\@glo@grabfirst#1#2\@nil{%
5165 \ifdefempty\@glo@thislettergrp
5166 {%
5167 \def\@glo@thislettergrp{glssymbols}%
5168 }%
5169 {%
5170 \count@=\uccode'#1\relax
5171 \ifnum\count@=0\relax
5172 \def\@glo@thislettergrp{glssymbols}%
5173 \else
5174 \ifdefstring\@glo@sorttype{case}%
5175 {%
5176 \count@='#1\relax
5177 }%
5178 {%
5179 }%
5180 \edef\@glo@thislettergrp{\the\count@}%
5181 \fi
5182 }%
5183 }

```

\@gls@noidx@do Handler for list iteration used by \@print@noidx@glossary. The argument is the entry label. This only allows one sublevel.

```

5184 \newcommand{\@gls@noidx@do}[1]{%

```

Get this entry's location list

```

5185 \global\letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%

```

Does this entry have a parent?

```

5186 \ifglshasparent{#1}%
5187 {%

```

Has a parent.

```

5188 \gls@level=\csuse{glo@\glsdetoklabel{#1}@level}\relax
5189 \ifdefvoid{\@gls@loclist}
5190 {%
5191 \subglossentry{\gls@level}{#1}{}%
5192 }%
5193 {%
5194 \subglossentry{\gls@level}{#1}%
5195 {%
5196 \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
5197 }%
5198 }%

```

```

5199 }%
5200 {%
    Doesn't have a parent Get this entry's sort key
5201     \letcs{\@gls@sort}{glo@glsdetoklabel{#1}@sort}%
    Fetch the first letter:
5202     \expandafter\glo@grabfirst\@gls@sort{}{}\@nil
5203     \ifdefequal{\@glo@thislettergrp}{\@gls@currentlettergroup}%
5204     {}%
5205     {%
        Do the group header:
5206         \ifdefempty{\@gls@currentlettergroup}{\@gls@groupskip}%
5207         \gls@groupheading{\@glo@thislettergrp}%
5208     }%
5209     \let\@gls@currentlettergroup\@glo@thislettergrp
    Do this entry:
5210     \ifdefvoid{\@gls@loclist}
5211     {%
5212         \glossentry{#1}{}%
5213     }%
5214     {%
5215         \glossentry{#1}%
5216     }%
5217     \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
5218 }%
5219 }%
5220 }%
5221 }

```

`\glsnoidxloclist` `\glsnoidxloclist{<list cs>}`

Display location list.

```

5222 \newcommand*{\glsnoidxloclist}[1]{%
5223     \def\@gls@noidxloclist@sep{}%
5224     \def\@gls@noidxloclist@prev{}%
5225     \forlistloop{\glsnoidxloclisthandler}{#1}%
5226 }

```

`noidxloclisthandler` Handler for location list iterator.

```

5227 \newcommand*{\glsnoidxloclisthandler}[1]{%
5228     \ifdefstring{\@gls@noidxloclist@prev}{#1}%
5229     {%

```

Same as previous location so skip.

```

5230 }%
5231 {%
5232     \@gls@noidxloclist@sep

```

```

5233    #1%
5234    \def\@gls@noidxloclist@sep{\delimN}%
5235    \def\@gls@noidxloclist@prev{#1}%
5236  }%
5237 }

```

`\displayloclisthandler` Handler for location list iterator when used with `\glsdisplaynumberlist`.

```

5238 \newcommand*\@glsnoidxdisplayloclisthandler[1]{%
5239   \ifdefstring{\@gls@noidxloclist@prev}{#1}%
5240   {%
    Same as previous location so skip.
5241   }%
5242   {%
5243     \@gls@noidxloclist@sep
5244     \@gls@noidxloclist@prev
5245     \def\@gls@noidxloclist@prev{#1}%
5246   }%
5247 }

```

`\glsnoidxdisplayloc` `\glsnoidxdisplayloc{<prefix>}{<counter>}{<format>}{<location>}`

Display a location in the location list.

```

5248 \newcommand*\glsnoidxdisplayloc[4]{%
5249   \setentrycounter[#1]{#2}%
5250   \csuse{#3}{#4}%
5251 }

```

`\@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.

```

5252 \newcommand*\@gls@reference[3]{%

```

Add to label list

```

5253   \glsdoifexistsorwarn{#2}%
5254   {%
5255     \ifcsundef{\@glsref@#1}{\csgdef{\@glsref@#1}{}}{}%
5256     \ifinlistcs{#2}{\@glsref@#1}%
5257     {}%
5258     {\listcsgadd{\@glsref@#1}{#2}}%

```

Add to location list

```

5259   \ifcsundef{glo@\glsdetoklabel{#2}@loclist}%
5260   {\csgdef{glo@\glsdetoklabel{#2}@loclist}{}}{}%
5261   {}%
5262   \listcsgadd{glo@\glsdetoklabel{#2}@loclist}{#3}%

```

```

5263 }%
5264 }

```

The keys that can be used in the optional argument to `\printglossary` or `\printnoidxglossary` are as follows: The `type` key sets the glossary type.

```

5265 \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`.

```

5266 \define@key{printgloss}{title}{%
5267 \def\glossarytitle{#1}%
5268 \let\gls@dotoc@title\relax
5269 }

```

The `toctitle` sets the text used for the relevant entry in the table of contents.

```

5270 \define@key{printgloss}{toctitle}{%
5271 \def\glossarytoctitle{#1}%
5272 \let\gls@dotoc@title\relax
5273 }

```

The `style` key sets the glossary style (but only for the given glossary).

```

5274 \define@key{printgloss}{style}{%
5275 \ifcsundef{\glsstyle@#1}%
5276 {%
5277 \PackageError{glossaries}%
5278 {Glossary style ‘#1’ undefined}{}%
5279 }%
5280 {%
5281 \def\@glossarystyle{\setglossentrycompatibility
5282 \csname \glsstyle@#1\endcsname}%
5283 }%
5284 }

```

The `numberedsection` key determines if this glossary should be in a numbered section.

```

5285 \define@choicekey{printgloss}{numberedsection}[\val\nr]{%
5286 false,nolabel,autolabel,nameref}[nolabel]{%
5287 \ifcase\nr\relax
5288 \renewcommand*{\@glossarysecstar}{*}%
5289 \renewcommand*{\@glossaryseclabel}{}%
5290 \or
5291 \renewcommand*{\@glossarysecstar}{}%
5292 \renewcommand*{\@glossaryseclabel}{}%
5293 \or
5294 \renewcommand*{\@glossarysecstar}{}%
5295 \renewcommand*{\@glossaryseclabel}{\label{\glsautoprefix\@glo@type}}%
5296 \or
5297 \renewcommand*{\@glossarysecstar}{*}%
5298 \renewcommand*{\@glossaryseclabel}{%
5299 \protected@edef\@currentlabelname{\glossarytoctitle}%

```

```

5300     \label{\glsautoprefix\@glo@type}}}%
5301   \fi
5302 }

```

The `nogroupskip` key determines whether or not there should be a vertical gap between glossary groups.

```

5303 \define@choicekey{printgloss}{nogroupskip}{true,false}[true]{%
5304   \csuse{glsnogroupskip#1}}%
5305 }

```

The `nopostdot` key has the same effect as the package option of the same name.

```

5306 \define@choicekey{printgloss}{nopostdot}{true,false}[true]{%
5307   \csuse{glsnopostdot#1}}%
5308 }

```

The `entrycounter` key is the same as the package option but localised to the current glossary.

```

5309 \define@choicekey{printgloss}{entrycounter}{true,false}[true]{%
5310   \csuse{glsentrycounter#1}}%
5311   \ifglsentrycounter
5312     \ifx\@gls@counterwithin\@empty
5313       \newcounter{glossaryentry}%
5314     \else
5315       \newcounter{glossaryentry}[\@gls@counterwithin]%
5316     \fi
5317     \def\theHglossaryentry{\currentglossary.\theglossaryentry}%
5318     \renewcommand*\{glsresetentrycounter}{%
5319       \setcounter{glossaryentry}{0}}%
5320   }%
5321   \renewcommand*\{glsstepentry}[1]{%
5322     \refstepcounter{glossaryentry}%
5323     \label{glsentry-\glsdetoklabel{##1}}}%
5324   }%
5325   \renewcommand*\{glsentrycounterlabel}{\theglossaryentry.\space}%
5326   \renewcommand*\{glsentryitem}[1]{%
5327     \glsstepentry{##1}\glsentrycounterlabel
5328   }%
5329 \else
5330   \renewcommand*\{glsresetentrycounter}{}%
5331   \renewcommand*\{glsstepentry}[1]{}%
5332   \renewcommand*\{glsentrycounterlabel}{}%
5333   \renewcommand*\{glsentryitem}[1]{\glsresetsubentrycounter}
5334 \fi
5335 }

```

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.

```

5336 \define@choicekey{printgloss}{subentrycounter}{true,false}[true]{%
5337   \csuse{glssubentrycounter#1}%
5338   \ifglssubentrycounter
5339     \ifundef\c@glossarysubentry
5340     {%
5341       \ifgl Sentrycounter
5342         \newcounter{glossarysubentry}[glossaryentry]%
5343       \else
5344         \newcounter{glossarysubentry}
5345       \fi
5346     }{}%
5347   \renewcommand*{\glsteps subentry}[1]{%
5348     \edef\currentglssubentry{\glsetoklabel{##1}}%
5349     \refstepcounter{glossarysubentry}%
5350     \label{gl Sentry-\currentglssubentry}%
5351   }%
5352   \renewcommand*{\glSresetsubentrycounter}{%
5353     \setcounter{glossarysubentry}{0}%
5354   }%
5355   \renewcommand*{\glssubentryitem}[1]{%
5356     \glsteps subentry{##1}\glssubentrycounterlabel
5357   }%
5358   \renewcommand*{\glssubentrycounterlabel}{\theglossarysubentry\space}%
5359   \def\theHglossarysubentry{\currentglssubentry.\theglossarysubentry}
5360 \else
5361   \renewcommand*{\glssubentryitem}[1]{}%
5362   \renewcommand*{\glsteps subentry}[1]{}%
5363   \renewcommand*{\glSresetsubentrycounter}{}%
5364   \renewcommand*{\glssubentrycounterlabel}{}%
5365 \fi
5366 }

```

The nonumberlist key determines if this glossary should have a number list.

```

5367 \define@boolkey{printgloss}[gls]{nonumberlist}[true]{%
5368 \ifglSnonumberlist
5369   \def\glossaryentrynumbers##1{}%
5370 \else
5371   \def\glossaryentrynumbers##1{##1}%
5372 \fi}

```

The sort key sets the glossary sort handler (\printnoidxglossary only).

```

5373 \define@key{printgloss}{sort}{\@glo@assign@sortkey{#1}}

```

\no@no@assign@sortkey Issue error if used with \printglossary

```

5374 \newcommand*{\@glo@no@assign@sortkey}[1]{%
5375   \PackageError{glossaries}{‘sort’ key not permitted with
5376     \string\printglossary}%
5377   {The ‘sort’ key may only be used with \string\printnoidxglossary}%
5378 }

```

`@glo@assign@sortkey` For use with `\printnoidxglossary`

```

5379 \newcommand*{\@glo@assign@sortkey}[1]{%
5380   \def\@glo@sorttype{#1}%
5381 }

```

`\@glsnonetopages` 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 `\glsnonetopages` is place in the entry's description and 3 column tabular style glossary is used.) `\org@glossaryentrynumbers` needs to be set at the start of each glossary, in the event that `\glossaryentrynumber` is redefined.

```

5382 \newcommand*{\@glsnonetopages}{%
5383   \gdef\glossaryentrynumbers##1{%
5384     \glsresetentrylist
5385   }%
5386 }

```

`\@glsnextpages` Activate the next number list only. Global assignments required as it may not occur in the same level of grouping as the next numberlist. (For example, if `\glsnextpages` is place in the entry's description and 3 column tabular style glossary is used.) `\org@glossaryentrynumbers` needs to be set at the start of each glossary, in the event that `\glossaryentrynumber` is redefined.

```

5387 \newcommand*{\@glsnextpages}{%
5388   \gdef\glossaryentrynumbers##1{%
5389     ##1\glsresetentrylist}}

```

`\glsresetentrylist` Resets `\glossaryentrynumbers`

```

5390 \newcommand*{\glsresetentrylist}{%
5391   \global\let\glossaryentrynumbers\org@glossaryentrynumbers}

```

`\glsnonetopages` Outside of `\printglossary` this does nothing.

```

5392 \newcommand*{\glsnonetopages}{}

```

`\glsnextpages` Outside of `\printglossary` this does nothing.

```

5393 \newcommand*{\glsnextpages}{}

```

`glossaryentry` If the `entrycounter` package option has been used, define a counter to number each level 0 entry.

```

5394 \ifglstentrycounter
5395   \ifx\@gls@counterwithin\@empty
5396     \newcounter{glossaryentry}
5397   \else
5398     \newcounter{glossaryentry}[\@gls@counterwithin]
5399   \fi
5400   \def\theHglossaryentry{\currentglossary.\theglossaryentry}
5401 \fi

```


glossarysubentry If the subentrycounter package option has been used, define a counter to number each level 1 entry.

```

5402 \ifglssubentrycounter
5403   \ifglsentrycounter
5404     \newcounter{glossarysubentry}[glossaryentry]
5405   \else
5406     \newcounter{glossarysubentry}
5407   \fi
5408   \def\theHglossarysubentry{\currentglssubentry.\theglossarysubentry}
5409 \fi

```

esetsubentrycounter Resets the glossarysubentry counter.

```

5410 \ifglssubentrycounter
5411   \newcommand*{\glsresetsubentrycounter}{%
5412     \setcounter{glossarysubentry}{0}%
5413   }
5414 \else
5415   \newcommand*{\glsresetsubentrycounter}{}
5416 \fi

```

esetsubentrycounter Resets the glossareentry counter.

```

5417 \ifglsentrycounter
5418   \newcommand*{\glsresetentrycounter}{%
5419     \setcounter{glossaryentry}{0}%
5420   }
5421 \else
5422   \newcommand*{\glsresetentrycounter}{}
5423 \fi

```

\glsstepentry Advance the glossaryentry counter if in use. The argument is the label associated with the entry.

```

5424 \ifglsentrycounter
5425   \newcommand*{\glsstepentry}[1]{%
5426     \refstepcounter{glossaryentry}%
5427     \label{glsentry-\glsdetoklabel{#1}}%
5428   }
5429 \else
5430   \newcommand*{\glsstepentry}[1]{}
5431 \fi

```

\glsstepsubentry Advance the glossarysubentry counter if in use. The argument is the label associated with the subentry.

```

5432 \ifglssubentrycounter
5433   \newcommand*{\glsstepsubentry}[1]{%
5434     \edef\currentglssubentry{\glsdetoklabel{#1}}%
5435     \refstepcounter{glossarysubentry}%
5436     \label{glsentry-\currentglssubentry}%
5437   }

```

```

5438 \else
5439   \newcommand*{\glsstepsubentry}[1]{
5440 \fi

```

`\glsrefentry` Reference the entry or sub-entry counter if in use, otherwise just do `\gls`.

```

5441 \ifglssentrycounter
5442   \newcommand*{\glsrefentry}[1]{\ref{glsentry-\glsdetoklabel{#1}}}
5443 \else
5444   \ifglssubentrycounter
5445     \newcommand*{\glsrefentry}[1]{\ref{glsentry-\glsdetoklabel{#1}}}
5446   \else
5447     \newcommand*{\glsrefentry}[1]{\gls{#1}}
5448   \fi
5449 \fi

```

`glsentrycounterlabel` Defines how to display the glossaryentry counter.

```

5450 \ifglssentrycounter
5451   \newcommand*{\glsentrycounterlabel}{\theglossaryentry.\space}
5452 \else
5453   \newcommand*{\glsentrycounterlabel}{}
5454 \fi

```

`glssubentrycounterlabel` Defines how to display the glossarysubentry counter.

```

5455 \ifglssubentrycounter
5456   \newcommand*{\glssubentrycounterlabel}{\theglossarysubentry)\space}
5457 \else
5458   \newcommand*{\glssubentrycounterlabel}{}
5459 \fi

```

`\glsentryitem` Step and display glossaryentry counter, if appropriate.

```

5460 \ifglssentrycounter
5461   \newcommand*{\glsentryitem}[1]{%
5462     \glsstepentry{#1}\glsentrycounterlabel
5463   }
5464 \else
5465   \newcommand*{\glsentryitem}[1]{\glsresetsubentrycounter}
5466 \fi

```

`\glssubentryitem` Step and display glossarysubentry counter, if appropriate.

```

5467 \ifglssubentrycounter
5468   \newcommand*{\glssubentryitem}[1]{%
5469     \glsstepsubentry{#1}\glssubentrycounterlabel
5470   }
5471 \else
5472   \newcommand*{\glssubentryitem}[1]{}
5473 \fi

```

`theglossary` If the `theglossary` environment has already been defined, a warning will be issued. This environment should be redefined by glossary styles.

```

5474 \ifcsundef{theglossary}%
5475 {%
5476   \newenvironment{theglossary}{}{}%
5477 }%
5478 {%
5479   \@gls@warnontheGLOSSdefined
5480   \renewenvironment{theglossary}{}{}%
5481 }

```

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`

```

5482 \newcommand*{\glossaryheader}{}

```

`\glstarget` `\glstarget{<label>}{<name>}`

Provide user interface to `\@glstarget` to make it easier to modify the glossary style in the document.

```

5483 \newcommand*{\glstarget}[2]{\@glstarget{\glo@linkprefix#1}{#2}}

```

As from version 3.08, glossary information is now written to the external files using `\glossentry` and `\subglossentry` instead of `\glossaryentryfield` and `\glossarysubentryfield`. The default definition provides backward compatibility for glossary styles that use the old forms.

`compatibleglossentry`

`\glossentry{<label>}{<page-list>}`

```

5484 \providecommand*{\compatibleglossentry}[2]{%
5485   \toks@{#2}%
5486   \protected@edef\@do@glossentry{\noexpand\glossaryentryfield{#1}%
5487     {\noexpand\glsnamefont
5488       {\expandafter\expandonce\csname glo@#1@name\endcsname}}%
5489     {\expandafter\expandonce\csname glo@#1@desc\endcsname}%
5490     {\expandafter\expandonce\csname glo@#1@symbol\endcsname}%
5491     {\the\toks@}}%
5492   }%
5493   \@do@glossentry
5494 }

```

`\glossentryname`

```

5495 \newcommand*{\glossentryname}[1]{%
5496   \glsdoifexistsorwarn{#1}%

```

```

5497  {%
5498    \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
5499    \expandafter\glsnamefont\expandafter{\glo@name}%
5500  }%
5501 }

```

\Glossentryname

```

5502 \newcommand*{\Glossentryname}[1]{%
5503   \glsdoifexistsorwarn{#1}%
5504   {%
5505     \glsnamefont{\Glsentryname{#1}}%
5506   }%
5507 }

```

\glossentrydesc

```

5508 \newcommand*{\glossentrydesc}[1]{%
5509   \glsdoifexistsorwarn{#1}%
5510   {%
5511     \glentrydesc{#1}%
5512   }%
5513 }

```

\Glossentrydesc

```

5514 \newcommand*{\Glossentrydesc}[1]{%
5515   \glsdoifexistsorwarn{#1}%
5516   {%
5517     \Glsentrydesc{#1}%
5518   }%
5519 }

```

\glossentrysymbol

```

5520 \newcommand*{\glossentrysymbol}[1]{%
5521   \glsdoifexistsorwarn{#1}%
5522   {%
5523     \glentrysymbol{#1}%
5524   }%
5525 }

```

\Glossentrysymbol

```

5526 \newcommand*{\Glossentrysymbol}[1]{%
5527   \glsdoifexistsorwarn{#1}%
5528   {%
5529     \Glsentrysymbol{#1}%
5530   }%
5531 }

```

patiblesubglossentry \subglossentry{<level>}{<label>}{<page-list>}

```

5532 \providecommand*\compatiblesubglossentry}[3]{%
5533   \toks@{#3}%
5534   \protected@edef\@do@subglossentry{\noexpand\glossarysubentryfield{\number#1}%
5535     {#2}%
5536     {\noexpand\glsnamefont
5537       {\expandafter\expandonce\csname glo@#2@name\endcsname}}}%
5538     {\expandafter\expandonce\csname glo@#2@desc\endcsname}}%
5539     {\expandafter\expandonce\csname glo@#2@symbol\endcsname}}%
5540     {\the\toks@}%
5541   }%
5542   \@do@subglossentry
5543 }

```

sentrycompatibility

```

5544 \newcommand*\setglossentrycompatibility{%
5545   \let\glossentry\compatibleglossentry
5546   \let\subglossentry\compatiblesubglossentry
5547 }
5548 \setglossentrycompatibility

```

\glossaryentryfield

```
\glossaryentryfield{<label>}{<name>}{<description>}{<symbol>}{<page-list>}
```

This command formerly governed how each entry row should be formatted in the glossary. Now deprecated.

```

5549 \newcommand{\glossaryentryfield}[5]{%
5550   \GlossariesWarning
5551   {Deprecated use of \string\glossaryentryfield.^^J
5552     I recommend you change to \string\glossentry.^^J
5553     If you've just upgraded, try removing your gls auxiliary
5554     files^^J and recompile}%
5555   \noindent\textbf{\glstarget{#1}{#2}} #4 #3. #5\par}

```

lossarysubentryfield

```
\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.)

```

5556 \newcommand*\glossarysubentryfield}[6]{%
5557   \GlossariesWarning
5558   {Deprecated use of \string\glossarysubentryfield.^^J
5559     I recommend you change to \string\subglossentry.^^J
5560     If you've just upgraded, try removing your gls auxiliary

```

```

5561   files^^J and recompile}%
5562   \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`

```

5563 \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`

```

5564 \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 `\glsgrouptitle` and `\glsgrouplabel` so that the label is translated into the required title (and vice-versa).

`\glsgrouptitle{<label>}`

This command produces the title for the glossary group whose label is given by `<label>`. By default, the group labelled `glssymbols` produces `\glssymbolsgroupname`, the group labelled `glsnumbers` produces `\glsnumbersgroupname` and all the other groups simply produce their label. As mentioned above, the group labels are: `glssymbols`, `glsnumbers`, A, ..., Z. If you want to redefine the group titles, you will need to redefine this command. Languages other than English may produce labels that are non-expandable, so we need to check for that otherwise it will create a “missing `\endcsname` inserted” error.

```

\glsgetgrouptitle
5565 \newcommand*\glsgetgrouptitle}[1]{%
5566   \@gls@getgrouptitle{#1}{\@gls@grptitle}%
5567   \@gls@grptitle
5568 }

\@gls@getgrouptitle  Gets the group title specified by the label (first argument) and stores in the sec-
                      ond argument, which must be a control sequence.
5569 \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.
5570   \dtl@ifsingle{#1}%
5571   {%
5572     \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
5573   }%
5574   {%
5575     \ifboolexpr{test{\ifstrequal{#1}{glssymbols}}
5576                 or test{\ifstrequal{#1}{glsnumbers}}}%
5577     {%
5578       \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
5579     }%
5580     {%
5581       \def#2{#1}%
5582     }%
5583   }%
5584 }

@getothergrouptitle  Version for the no-indexing app option:
5585 \newcommand*\@gls@noidx@getgrouptitle}[2]{%
5586   \DTLifint{#1}%
5587   {\edef#2{\char#1\relax}}%
5588   {%
5589     \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
5590   }%
5591 }

```

`\glsgetgrouplabel{<title>}`

This command does the reverse to the previous command. The argument is the group title, and it produces the group label. Note that if you redefine `\glsgetgrouptitle`, you will also need to redefine `\glsgetgrouplabel`.

```

\glsgetgrouplabel
5592 \newcommand*\glsgetgrouplabel}[1]{%
5593 \ifthenelse{\equal{#1}{\glssymbolsgroupname}}{\glssymbols}{%
5594 \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`

```
5595 \newcommand*{\setentrycounter}[2][1]{%
5596   \def\@glo@counterprefix{#1}%
5597   \ifx\@glo@counterprefix\@empty
5598     \def\@glo@counterprefix{.}%
5599   \else
5600     \def\@glo@counterprefix{.#1.}%
5601   \fi
5602   \def\glsentrycounter{#2}%
5603 }
```

The current glossary style can be set using `\setglossarystyle{<style>}`.

`\setglossarystyle`

```
5604 \newcommand*{\setglossarystyle}[1]{%
5605   \ifcsundef{@glsstyle@#1}%
5606   {%
5607     \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
5608   }%
5609   {%
5610     \csname @glsstyle@#1\endcsname
5611   }%
5612 }
```

`\glossarystyle`

```
5613 \newcommand*{\glossarystyle}[1]{%
5614   \ifcsundef{@glsstyle@#1}%
5615   {%
5616     \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
5617   }%
5618   {%
5619     \GlossariesWarning
5620     {Deprecated command \string\glossarystyle.^^J
5621     I recommend you switch to \string\setglossarystyle\space unless
5622     you want to maintain backward compatibility}%
5623     \setglossentrycompatibility
5624     \csname @glsstyle@#1\endcsname

5625     \ifcsdef{@glscompstyle@#1}%
5626     {\setglossentrycompatibility\csuse{@glscompstyle@#1}}%
5627     {}%
5628   }%
5629 }
```

`\newglossarystyle` New glossary styles can be defined using:

`\newglossarystyle{<name>}{<definition>}`

The `<definition>` argument should redefine `\theglossary`, `\glossaryheader`, `\glsgroupheading`, `\glossaryentryfield` and `\glsgroupskip` (see [subsection 1.18](#) 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.

```
5630 \newcommand{\newglossarystyle}[2]{%
5631   \ifcsundef{@glsstyle@#1}%
5632   {%
5633     \expandafter\def\csname @glsstyle@#1\endcsname{#2}%
5634   }%
5635   {%
5636     \PackageError{glossaries}{Glossary style ‘#1’ is already defined}{}%
5637   }%
5638 }
```

`\renewglossarystyle` Code for this macro supplied by Marco Daniel.

```
5639 \newcommand{\renewglossarystyle}[2]{%
5640   \ifcsundef{@glsstyle@#1}%
5641   {%
5642     \PackageError{glossaries}{Glossary style ‘#1’ isn’t already defined}{}%
5643   }%
5644   {%
5645     \csdef{@glsstyle@#1}{#2}%
5646   }%
5647 }
```

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`

```
5648 \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`

```
5649 \ifcsundef{hyperlink}%
5650 {%
5651   \def\glshypernumber#1{#1}%
5652 }%
5653 {%
5654   \def\glshypernumber#1{\@glshypernumber#1\nohyperpage{}\@nil}
5655 }
```

`\@glshypernumber` This code was provided by Heiko Oberdiek to allow material to be attached to the location.

```
5656 \def\@glshypernumber#1\nohyperpage#2#3\@nil{%
5657   \ifx\#1\%
5658   \else
5659     \@delimR#1\delimR\delimR\%
5660   \fi
5661   \ifx\#2\%
5662   \else
5663     #2%
5664   \fi
5665   \ifx\#3\%
5666   \else
5667     \@glshypernumber#3\@nil
5668   \fi
5669 }
```

`\@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`

```
5670 \def\@delimR#1\delimR #2\delimR #3\%
5671 \ifx\#2\%
5672   \@delimN{#1}%
5673 \else
5674   \@gls@numberlink{#1}\delimR\@gls@numberlink{#2}%
5675 \fi}
```

`\@delimN` displays a list of individual numbers, instead of a range:

`\@delimN`

```
5676 \def\@delimN#1{\@@delimN#1\delimN \delimN\%
5677 \def\@@delimN#1\delimN #2\delimN#3\%
5678 \ifx\#3\%
5679   \@gls@numberlink{#1}%
5680 }
```

```

5680 \else
5681   \@gls@numberlink{#1}\delimN\@gls@numberlink{#2}%
5682 \fi
5683 }

```

The following code is modified from hyperref's \HyInd@pagelink where the name of the counter being used is given by \@gls@counter.

```

5684 \def\@gls@numberlink#1{%
5685 \begingroup
5686 \toks@={}%
5687 \@gls@removespaces#1 \@nil
5688 \endgroup}

5689 \def\@gls@removespaces#1 #2\@nil{%
5690 \toks@=\expandafter{\the\toks@#1}%
5691 \ifx\#2\%
5692   \edef\x{\the\toks@}%
5693   \ifx\x\empty
5694   \else

5695     \hyperlink{\glstentrycounter\@gls@counterprefix\the\toks@}%
5696               {\the\toks@}%
5697   \fi
5698 \else
5699   \@gls@ReturnAfterFi{%
5700     \@gls@removespaces#2\@nil
5701   }%
5702 \fi
5703 }
5704 \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
5705 \newcommand*\hyperrm[1]{\textrm{\glshypernumber{#1}}}

\hypersf
5706 \newcommand*\hypersf[1]{\textsf{\glshypernumber{#1}}}

\hypertt
5707 \newcommand*\hypertt[1]{\texttt{\glshypernumber{#1}}}

\hyperbf
5708 \newcommand*\hyperbf[1]{\textbf{\glshypernumber{#1}}}

\hypermd
5709 \newcommand*\hypermd[1]{\textmd{\glshypernumber{#1}}}

```

```

\hyperit
5710 \newcommand*{\hyperit}[1]{\textit{\glshypernumber{#1}}}

\hypersl
5711 \newcommand*{\hypersl}[1]{\textsl{\glshypernumber{#1}}}

\hyperup
5712 \newcommand*{\hyperup}[1]{\textup{\glshypernumber{#1}}}

\hypersc
5713 \newcommand*{\hypersc}[1]{\textsc{\glshypernumber{#1}}}

\hyperemph
5714 \newcommand*{\hyperemph}[1]{\emph{\glshypernumber{#1}}}

```

1.16 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.

```

5715 \newcommand{\oldacronym}[4][\gls@label]{%
5716   \def\gls@label{#2}%
5717   \newacronym[#4]{#1}{#2}{#3}%
5718   \ifcsundef{xspace}%
5719     {%
5720       \expandafter\edef\csname#1\endcsname{%
5721         \noexpand\@ifstar{\noexpand\Gls{#1}}{\noexpand\gls{#1}}}%
5722       }%
5723     }%
5724     {%
5725       \expandafter\edef\csname#1\endcsname{%
5726         \noexpand\@ifstar{\noexpand\Gls{#1}\noexpand\xspace}{%

```

```

5727     \noexpand\gls{#1}\noexpand\xspace}%
5728   }%
5729 }%
5730 }

```

```
\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`

```
5731 \newcommand{\newacronym}[4] [] {}
```

Set up some convenient short cuts. These need to be changed if `\newacronym` is changed (or if the description key is changed).

`\acrpluralsuffix` Plural suffix used by `\newacronym`. This just defaults to `\glspluralsuffix` but is changed to include `\textup` if the `smallcaps` option is used, so that the suffix doesn't appear in small caps as it doesn't look right. For example, `ABCS` looks as though the “s” is part of the acronym, but `ABCS` looks as though the “s” is a plural suffix. Since the entire text `abcs` is set in `\textsc`, `\textup` is needed to cancel it out.

```
5732 \newcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}
```

If `garamondx` has been loaded, need to use `\textulc` instead of `\textup`.

`\glstextup`

```
5733 \newrobustcmd*{\glstextup}[1]{\ifdef\textulc{\textulc{#1}}{\textup{#1}}}
```

The following are defined for compatibility with version 2.07 and earlier.

`\glsshortkey`

```
5734 \newcommand*{\glsshortkey}{short}
```

`\glsshortpluralkey`

```
5735 \newcommand*{\glsshortpluralkey}{shortplural}
```

`\glslongkey`

```
5736 \newcommand*{\glslongkey}{long}
```

`\glslongpluralkey`

```
5737 \newcommand*{\glslongpluralkey}{longplural}
```

`\acrfull` Full form of the acronym.

```
5738 \newrobustcmd*{\acrfull}{\@gls@hyp@opt\@ns@acrfull}
```

```
5739 \newcommand*\ns@acrfull[2][]{%
```

```
5740   \new@ifnextchar[{\@acrfull{#1}{#2}}}%
```

```
5741   {\@acrfull{#1}{#2}[]}%
```

```
5742 }
```

`\@acrfull` Low-level macro:

```
5743 \def\@acrfull#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
5744   \acrfullfmt{#1}{#2}{#3}%
```

```
5745 }
```

Using `\acrlinkfullformat` and `\acrfullformat` is now deprecated as it can cause complications with the first letter upper case variants, but the package needs to provide backward compatibility support.

`\acrfullfmt` No case change full format.

```
5746 \newcommand*{\acrfullfmt}[3]{%
```

```
5747   \acrlinkfullformat{\@acrlong}{\@acrshort}{#1}{#2}{#3}%
```

```
5748 }
```

`\acrlinkfullformat` Format for full links like `\acrfull`. Syntax: `\acrlinkfullformat{<long cs>}{<short cs>}{<options>}{<label>}{<insert>}`

```
5749 \newcommand{\acrlinkfullformat}[5]{%
```

```
5750   \acrfullformat{#1{#3}{#4}[#5]}{#2{#3}{#4}[]}%
```

```
5751 }
```

`\acrfullformat` Default full form is `<long>` (`<short>`).

```
5752 \newcommand{\acrfullformat}[2]{#1\glsspace(#2)}
```

`\glsspace` Robust space to ensure it's written to the `.glsdefs` file.

```
5753 \newrobustcmd{\glsspace}{\space}
```

Default format for full acronym

`\Acrfull`

```
5754 \newrobustcmd*{\Acrfull}{\@gls@hyp@opt\@ns@Acrfull}
```

```
5755 \newcommand*\ns@Acrfull[2][]{%
```

```
5756   \new@ifnextchar[{\@Acrfull{#1}{#2}}}%
```

```
5757   {\@Acrfull{#1}{#2}[]}%
```

```
5758 }
```

Low-level macro:

```
5759 \def\@Acrfull#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
5760 \Acrfullfmt{#1}{#2}{#3}%  
5761 }
```

`\Acrfullfmt` First letter upper case full format.

```
5762 \newcommand*\Acrfullfmt}[3]{%  
5763 \acrlinkfullformat{\@Acrlong}{\@acrshort}{#1}{#2}{#3}%  
5764 }
```

`\ACRfull`

```
5765 \newrobustcmd*\ACRfull{\@gls@hyp@opt\@ns@ACRfull}  
  
5766 \newcommand*\ns@ACRfull[2][]{%  
5767 \new@ifnextchar[\@ACRfull{#1}{#2}}%  
5768 {\@ACRfull{#1}{#2}[]}%  
5769 }
```

Low-level macro:

```
5770 \def\@ACRfull#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
5771 \ACRfullfmt{#1}{#2}{#3}%  
5772 }
```

`\ACRfullfmt` All upper case full format.

```
5773 \newcommand*\ACRfullfmt}[3]{%  
5774 \acrlinkfullformat{\@ACRlong}{\@ACRshort}{#1}{#2}{#3}%  
5775 }
```

Plural:

`\acrfullpl`

```
5776 \newrobustcmd*\acrfullpl{\@gls@hyp@opt\@ns@acrfullpl}  
  
5777 \newcommand*\ns@acrfullpl[2][]{%  
5778 \new@ifnextchar[\@acrfullpl{#1}{#2}}%  
5779 {\@acrfullpl{#1}{#2}[]}%  
5780 }
```

Low-level macro:

```
5781 \def\@acrfullpl#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
5782 \acrfullplfmt{#1}{#2}{#3}%  
5783 }
```

```

\acrfullplfmt  No case change plural full format.
5784 \newcommand*\acrfullplfmt}[3]{%
5785   \acrlinkfullformat{\@acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%
5786 }

\Acrfullpl
5787 \newrobustcmd*\Acrfullpl{\@gls@hyp@opt\ns@Acrfullpl}

5788 \newcommand*\ns@Acrfullpl[2][{}]{%
5789   \new@ifnextchar[{\@Acrfullpl{#1}{#2}}%
5790     {\@Acrfullpl{#1}{#2}[]}%
5791 }

    Low-level macro:
5792 \def\@Acrfullpl#1#2[#3]{%

    Make it easier for acronym styles to change this:
5793   \Acrfullplfmt{#1}{#2}{#3}%
5794 }

\Acrfullplfmt  First letter upper case plural full format.
5795 \newcommand*\Acrfullplfmt}[3]{%
5796   \acrlinkfullformat{\@Acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%
5797 }

\ACRfullpl
5798 \newrobustcmd*\ACRfullpl{\@gls@hyp@opt\ns@ACRfullpl}

5799 \newcommand*\ns@ACRfullpl[2][{}]{%
5800   \new@ifnextchar[{\@ACRfullpl{#1}{#2}}%
5801     {\@ACRfullpl{#1}{#2}[]}%
5802 }

    Low-level macro:
5803 \def\@ACRfullpl#1#2[#3]{%

    Make it easier for acronym styles to change this:
5804   \ACRfullplfmt{#1}{#2}{#3}%
5805 }

\ACRfullplfmt  All upper case plural full format.
5806 \newcommand*\ACRfullplfmt}[3]{%
5807   \acrlinkfullformat{\@ACRlongpl}{\@ACRshortpl}{#1}{#2}{#3}%
5808 }

```


1.17 Predefined acronym styles

`\acronymfont` This is only used with the additional acronym styles:

```
5809 \newcommand{\acronymfont}[1]{#1}
```

`\firstacronymfont` This is only used with the additional acronym styles:

```
5810 \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.

```
5811 \newcommand*{\acrnameformat}[2]{\acronymfont{#1}}
```

Define some tokens used by `\newacronym`:

`\glskeylisttok`

```
5812 \newtoks\glskeylisttok
```

`\glslabeltok`

```
5813 \newtoks\glslabeltok
```

`\glsshorttok`

```
5814 \newtoks\glsshorttok
```

`\glslongtok`

```
5815 \newtoks\glslongtok
```

`\newacronymhook` Provide a hook for `\newacronym`:

```
5816 \newcommand*{\newacronymhook}{}
```

`\setGenericNewAcronym` New improved version of setting the acronym style.

```
5817 \newcommand*{\SetGenericNewAcronym}{%
```

Change the behaviour of `\Glsentryname` to workaround expansion issues that cause a problem for `\makefirstuc`

```
5818 \let\@Gls@entryname\@Gls@acentryname
```

Change the way acronyms are defined:

```
5819 \renewcommand{\newacronym}[4][[]]{%
5820 \ifdefempty{\@glsacronymlists}%
5821 {%
5822 \def\@glo@type{\acronymtype}%
5823 \setkeys{glossentry}{##1}%
5824 \DeclareAcronymList{\@glo@type}%
5825 }%
5826 }%
5827 \glskeylisttok{##1}%
5828 \glslabeltok{##2}%
5829 \glsshorttok{##3}%
5830 \glslongtok{##4}%
```

```

5831 \newacronymhook
5832 \protected@edef\@do@newglossaryentry{%
5833 \noexpand\newglossaryentry{\the\glslabeltok}%
5834 {%
5835 type=\acronymtype,%
5836 name={\expandonce{\acronymentry{##2}}},%
5837 sort={\acronymsort{\the\glsshorttok}{\the\glslongtok}},%
5838 text={\the\glsshorttok},%
5839 short={\the\glsshorttok},%
5840 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
5841 long={\the\glslongtok},%
5842 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
5843 \GenericAcronymFields,%
5844 \the\glskeylisttok
5845 }%
5846 }%
5847 \@do@newglossaryentry
5848 }%

```

Make sure that \acrfull etc reflects the new style:

```

5849 \renewcommand*\acrfullfmt}[3]{%
5850 \glslink[##1]{##2}{\genacrfullformat{##2}{##3}}}%
5851 \renewcommand*\Acrfullfmt}[3]{%
5852 \glslink[##1]{##2}{\Genacrfullformat{##2}{##3}}}%
5853 \renewcommand*\ACRfullfmt}[3]{%
5854 \glslink[##1]{##2}{%
5855 \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}}%
5856 \renewcommand*\acrfullplfmt}[3]{%
5857 \glslink[##1]{##2}{\genplacrfullformat{##2}{##3}}}%
5858 \renewcommand*\Acrfullplfmt}[3]{%
5859 \glslink[##1]{##2}{\Genplacrfullformat{##2}{##3}}}%
5860 \renewcommand*\ACRfullplfmt}[3]{%
5861 \glslink[##1]{##2}{%
5862 \mfirstucMakeUppercase{\genplacrfullformat{##2}{##3}}}}%

```

Make sure that \glsentryfull etc reflects the new style:

```

5863 \renewcommand*\glsentryfull}[1]{\genacrfullformat{##1}{}}%
5864 \renewcommand*\Glsentryfull}[1]{\Genacrfullformat{##1}{}}%
5865 \renewcommand*\glsentryfullpl}[1]{\genplacrfullformat{##1}{}}%
5866 \renewcommand*\Glsentryfullpl}[1]{\Genplacrfullformat{##1}{}}%
5867 }

```

`\GenericAcronymFields` Fields used by \SetGenericNewAcronym that can be changed by the acronym style.

```

5868 \newcommand*\GenericAcronymFields{description={\the\glslongtok}}

```

`\acronymentry` `\acronymentry{<label>}`

Display style for the name field in the list of acronyms.

```
5869 \newcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{#1}}}
```

`\acronymsort` `\acronymsort{<short>}{<long>}`

Default sort format for acronyms.

```
5870 \newcommand*{\acronymsort}[2]{#1}
```

`\setacronymstyle` `\setacronymstyle{<style name>}`

```
5871 \newcommand*{\setacronymstyle}[1]{%
5872   \ifcsundef{@glsacr@dispstyle@#1}%
5873   {%
5874     \PackageError{glossaries}{Undefined acronym style ‘#1’}{%
5875     }%
5876   }%
5877   \ifdefempty{@glsacronymlists}%
5878   {%
5879     \DeclareAcronymList{\acronymtype}%
5880   }%
5881   {%
5882     \SetGenericNewAcronym
5883     \GlsUseAcrStyleDefs{#1}%
5884     \@for\@gls@type:=\@glsacronymlists\do{%
5885       \defglsentryfmt[\@gls@type]{\GlsUseAcrEntryDispStyle{#1}}%
5886     }%
5887   }%
5888 }
```

`\newacronymstyle` `\newacronymstyle{<style name>}{<entry format definition>}{<display definitions>}`

Defines a new acronym style called *<style name>*.

```
5889 \newcommand*{\newacronymstyle}[3]{%
5890   \ifcsdef{@glsacr@dispstyle@#1}%
5891   {%
5892     \PackageError{glossaries}{Acronym style ‘#1’ already exists}{%
5893     }%
5894   }%
5895   \csdef{@glsacr@dispstyle@#1}{#2}%
5896   \csdef{@glsacr@styledefs@#1}{#3}%
5897   }%
5898 }
```

`\renewacronymstyle` Redefines the given acronym style.

```

5899 \newcommand*\renewacronymstyle}[3]{%
5900   \ifcsdef{@glsacr@dispstyle@#1}%
5901   {%
5902     \csdef{@glsacr@dispstyle@#1}{#2}%
5903     \csdef{@glsacr@styledefs@#1}{#3}%
5904   }%
5905   {%
5906     \PackageError{glossaries}{Acronym style ‘#1’ doesn’t exist}{}%
5907   }%
5908 }

```

seAcrEntryDispStyle

```

5909 \newcommand*\GlsUseAcrEntryDispStyle}[1]{\csuse{@glsacr@dispstyle@#1}}

```

\GlsUseAcrStyleDefs

```

5910 \newcommand*\GlsUseAcrStyleDefs}[1]{\csuse{@glsacr@styledefs@#1}}

```

Predefined acronym styles:

long-short *(long)* (*short*) acronym style.

```

5911 \newacronymstyle{long-short}%
5912 {%

```

Check for long form in case this is a mixed glossary.

```

5913   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
5914 }%
5915 {%
5916   \renewcommand*\GenericAcronymFields{description={\the\glslongtok}}%
5917   \renewcommand*\genacrfullformat}[2]{%
5918     \glsentrylong{##1}##2\space
5919     (\protect\firstacronymfont{\glsentryshort{##1}})%
5920   }%
5921   \renewcommand*\Genacrfullformat}[2]{%
5922     \Glsentrylong{##1}##2\space
5923     (\protect\firstacronymfont{\glsentryshort{##1}})%
5924   }%
5925   \renewcommand*\genplacrfullformat}[2]{%
5926     \glsentrylongpl{##1}##2\space
5927     (\protect\firstacronymfont{\glsentryshortpl{##1}})%
5928   }%
5929   \renewcommand*\Genplacrfullformat}[2]{%
5930     \Glsentrylongpl{##1}##2\space
5931     (\protect\firstacronymfont{\glsentryshortpl{##1}})%
5932   }%
5933   \renewcommand*\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}
5934   \renewcommand*\acronymsort}[2]{##1}%
5935   \renewcommand*\acronymfont}[1]{##1}%
5936   \renewcommand*\firstacronymfont}[1]{\acronymfont{##1}}%
5937   \renewcommand*\acrpluralsuffix{\glspluralsuffix}%
5938 }

```

short-long *<short>* (*<long>*) acronym style.

```
5939 \newacronymstyle{short-long}%
5940 {%
    Check for long form in case this is a mixed glossary.
5941   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
5942 }%
5943 {%
5944   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
5945   \renewcommand*{\genacrfullformat}[2]{%
5946     \protect\firstacronymfont{\glsentryshort{##1}}##2\space
5947     (\glsentrylong{##1})%
5948   }%
5949   \renewcommand*{\Genacrfullformat}[2]{%
5950     \protect\firstacronymfont{\Glsentryshort{##1}}##2\space
5951     (\glsentrylong{##1})%
5952   }%
5953   \renewcommand*{\genplacrfullformat}[2]{%
5954     \protect\firstacronymfont{\glsentryshortpl{##1}}##2\space
5955     (\glsentrylongpl{##1})%
5956   }%
5957   \renewcommand*{\Genplacrfullformat}[2]{%
5958     \protect\firstacronymfont{\Glsentryshortpl{##1}}##2\space
5959     (\glsentrylongpl{##1})%
5960   }%

5961   \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
5962   \renewcommand*{\acronymsort}[2]{##1}%
5963   \renewcommand*{\acronymfont}[1]{##1}%
5964   \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
5965   \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
5966 }
```

long-sc-short *<long>* (*\textsc{<short>}*) acronym style.

```
5967 \newacronymstyle{long-sc-short}%
5968 {%
5969   \GlsUseAcrEntryDisplayStyle{long-short}%
5970 }%
5971 {%
5972   \GlsUseAcrStyleDefs{long-short}%
5973   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
5974   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
5975 }
```

long-sm-short *<long>* (*\textsmaller{<short>}*) acronym style.

```
5976 \newacronymstyle{long-sm-short}%
5977 {%
5978   \GlsUseAcrEntryDisplayStyle{long-short}%
5979 }%
5980 {%
```

```

5981 \GlsUseAcrStyleDefs{long-short}%
5982 \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
5983 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
5984 }

```

sc-short-long $\langle short \rangle$ ($\textsc{\langle long \rangle}$) acronym style.

```

5985 \newacronymstyle{sc-short-long}%
5986 {%
5987 \GlsUseAcrEntryDisplayStyle{short-long}%
5988 }%
5989 {%
5990 \GlsUseAcrStyleDefs{short-long}%
5991 \renewcommand{\acronymfont}[1]{\textsc{##1}}%
5992 \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
5993 }

```

sm-short-long $\langle short \rangle$ ($\textsmaller{\langle long \rangle}$) acronym style.

```

5994 \newacronymstyle{sm-short-long}%
5995 {%
5996 \GlsUseAcrEntryDisplayStyle{short-long}%
5997 }%
5998 {%
5999 \GlsUseAcrStyleDefs{short-long}%
6000 \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6001 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6002 }

```

long-short-desc $\langle long \rangle$ ($\langle short \rangle$) acronym style that has an accompanying description (which the user needs to supply).

```

6003 \newacronymstyle{long-short-desc}%
6004 {%
6005 \GlsUseAcrEntryDisplayStyle{long-short}%
6006 }%
6007 {%
6008 \GlsUseAcrStyleDefs{long-short}%
6009 \renewcommand*{\GenericAcronymFields}{}%
6010 \renewcommand*{\acronymsort}[2]{##2}%
6011 \renewcommand*{\acronymentry}[1]{%
6012 \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6013 }

```

long-sc-short-desc $\langle long \rangle$ ($\textsc{\langle short \rangle}$) acronym style that has an accompanying description (which the user needs to supply).

```

6014 \newacronymstyle{long-sc-short-desc}%
6015 {%
6016 \GlsUseAcrEntryDisplayStyle{long-sc-short}%
6017 }%
6018 {%

```

```

6019 \GlsUseAcrStyleDefs{long-sm-short}%
6020 \renewcommand*{\GenericAcronymFields}{}%
6021 \renewcommand*{\acronymsort}[2]{##2}%
6022 \renewcommand*{\acronymentry}[1]{%
6023     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6024 }

```

long-sm-short-desc *⟨long⟩* (\textsmaller{⟨short⟩}) acronym style that has an accompanying description (which the user needs to supply).

```

6025 \newacronymstyle{long-sm-short-desc}%
6026 {%
6027     \GlsUseAcrEntryDisplayStyle{long-sm-short}%
6028 }%
6029 {%
6030     \GlsUseAcrStyleDefs{long-sm-short}%
6031     \renewcommand*{\GenericAcronymFields}{}%
6032     \renewcommand*{\acronymsort}[2]{##2}%
6033     \renewcommand*{\acronymentry}[1]{%
6034         \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6035 }

```

short-long-desc *⟨short⟩* ({⟨long⟩}) acronym style that has an accompanying description (which the user needs to supply).

```

6036 \newacronymstyle{short-long-desc}%
6037 {%
6038     \GlsUseAcrEntryDisplayStyle{short-long}%
6039 }%
6040 {%
6041     \GlsUseAcrStyleDefs{short-long}%
6042     \renewcommand*{\GenericAcronymFields}{}%
6043     \renewcommand*{\acronymsort}[2]{##2}%
6044     \renewcommand*{\acronymentry}[1]{%
6045         \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6046 }

```

sc-short-long-desc *⟨long⟩* (\textsc{⟨short⟩}) acronym style that has an accompanying description (which the user needs to supply).

```

6047 \newacronymstyle{sc-short-long-desc}%
6048 {%
6049     \GlsUseAcrEntryDisplayStyle{sc-short-long}%
6050 }%
6051 {%
6052     \GlsUseAcrStyleDefs{sc-short-long}%
6053     \renewcommand*{\GenericAcronymFields}{}%
6054     \renewcommand*{\acronymsort}[2]{##2}%
6055     \renewcommand*{\acronymentry}[1]{%
6056         \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6057 }

```

sm-short-long-desc *<long>* (\textsmaller{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

6058 \newacronymstyle{sm-short-long-desc}%
6059 {%
6060   \GlsUseAcrEntryDispStyle{sm-short-long}%
6061 }%
6062 {%
6063   \GlsUseAcrStyleDefs{sm-short-long}%
6064   \renewcommand*{\GenericAcronymFields}{}%
6065   \renewcommand*{\acronymsort}[2]{##2}%
6066   \renewcommand*{\acronymentry}[1]{%
6067     \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6068 }
```

dua *<long>* only acronym style.

```

6069 \newacronymstyle{dua}%
6070 {%
```

Check for long form in case this is a mixed glossary.

```

6071   \ifdefempty\glscustomtext
6072   {%
6073     \ifglshaslong{\glslabel}%
6074     {%
6075       \glsifplural
6076       {%
```

Plural form:

```

6077         \glscapscase
6078         {%
```

Plural form, don't adjust case:

```

6079         \glsentrylongpl{\glslabel}\glsinsert
6080         }%
6081         {%
```

Plural form, make first letter upper case:

```

6082         \Glsentrylongpl{\glslabel}\glsinsert
6083         }%
6084         {%
```

Plural form, all caps:

```

6085         \mfirstucMakeUppercase
6086         {\glsentrylongpl{\glslabel}\glsinsert}%
6087         }%
6088         }%
6089         {%
```

Singular form

```

6090         \glscapscase
6091         {%
```


Singular form, don't adjust case:

```
6092      \glsentrylong{\glslabel}\glsinsert
6093      }%
6094      {%
```

Subsequent singular form, make first letter upper case:

```
6095      \Glsentrylong{\glslabel}\glsinsert
6096      }%
6097      {%
```

Subsequent singular form, all caps:

```
6098      \mfirstucMakeUppercase
6099      {\glsentrylong{\glslabel}\glsinsert}%
6100      }%
6101      }%
6102      }%
6103      {%
```

Not an acronym:

```
6104      \glsgenentryfmt
6105      }%
6106      }%
6107      {\glscustomtext\glsinsert}%
6108      }%
6109      {%
6110      \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%

6111      \renewcommand*{\acrfullfmt}[3]{%
6112          \glslink[##1]{##2}{\glsentrylong{##2}##3\space
6113              (\acronymfont{\glsentryshort{##2}})}}%
6114      \renewcommand*{\Acrfullfmt}[3]{%
6115          \glslink[##1]{##2}{\Glsentrylong{##2}##3\space
6116              (\acronymfont{\glsentryshort{##2}})}}%
6117      \renewcommand*{\ACRfullfmt}[3]{%
6118          \glslink[##1]{##2}{%
6119              \mfirstucMakeUppercase{\glsentrylong{##2}##3\space
6120                  (\acronymfont{\glsentryshort{##2}})}}}%

6121      \renewcommand*{\acrfullplfmt}[3]{%
6122          \glslink[##1]{##2}{\glsentrylongpl{##2}##3\space
6123              (\acronymfont{\glsentryshortpl{##2}})}}%

6124      \renewcommand*{\Acrfullplfmt}[3]{%
6125          \glslink[##1]{##2}{\Glsentrylongpl{##2}##3\space
6126              (\acronymfont{\glsentryshortpl{##2}})}}%
6127      \renewcommand*{\ACRfullplfmt}[3]{%
6128          \glslink[##1]{##2}{%
6129              \mfirstucMakeUppercase{\glsentrylongpl{##2}##3\space
6130                  (\acronymfont{\glsentryshortpl{##2}})}}}%
6131      \renewcommand*{\glsentryfull}[1]{%
6132          \glsentrylong{##1}\space(\acronymfont{\glsentryshort{##1}})%
```

```

6133 }%
6134 \renewcommand*{\Glsentryfull}[1]{%
6135   \Glsentrylong{##1}\space(\acronymfont{\glsentryshort{##1}})%
6136 }%
6137 \renewcommand*{\glsentryfullpl}[1]{%
6138   \glsentrylongpl{##1}\space(\acronymfont{\glsentryshortpl{##1}})%
6139 }%
6140 \renewcommand*{\Glsentryfullpl}[1]{%
6141   \Glsentrylongpl{##1}\space(\acronymfont{\glsentryshortpl{##1}})%
6142 }%
6143 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6144 \renewcommand*{\acronymsort}[2]{##1}%
6145 \renewcommand*{\acronymfont}[1]{##1}%
6146 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6147 }

```

dua-desc *<long>* only acronym style with user-supplied description.

```

6148 \newacronymstyle{dua-desc}%
6149 {%
6150   \GlsUseAcrEntryDispStyle{dua}%
6151 }%
6152 {%
6153   \GlsUseAcrStyleDefs{dua}%
6154   \renewcommand*{\GenericAcronymFields}{}%
6155   \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentrylong{##1}}}%
6156   \renewcommand*{\acronymsort}[2]{##2}%
6157 }%

```

footnote *<short>*\footnote{*<long>*} acronym style.

```

6158 \newacronymstyle{footnote}%
6159 {%
6160   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6161 }%
6162 {%
6163   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6164   \glshyperfirstfalse
6165   \renewcommand*{\genacrfullformat}[2]{%
6166     \protect\firstacronymfont{\glsentryshort{##1}}##2%
6167     \protect\footnote{\glsentrylong{##1}}%
6168   }%
6169   \renewcommand*{\Genacrfullformat}[2]{%
6170     \firstacronymfont{\glsentryshort{##1}}##2%
6171     \protect\footnote{\glsentrylong{##1}}%
6172   }%
6173   \renewcommand*{\genplacrfullformat}[2]{%

```

```

6174 \protect\firstacronymfont{\glentryshortpl{##1}}##2%
6175 \protect\footnote{\glentrylongpl{##1}}%
6176 }%
6177 \renewcommand*{\Genplacrfullformat}[2]{%
6178 \protect\firstacronymfont{\Glsentryshortpl{##1}}##2%
6179 \protect\footnote{\glentrylongpl{##1}}%
6180 }%
6181 \renewcommand*{\acronymentry}[1]{\acronymfont{\glentryshort{##1}}}%
6182 \renewcommand*{\acronymsort}[2]{##1}%
6183 \renewcommand*{\acronymfont}[1]{##1}%
6184 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%

```

Don't use footnotes for \acrfull:

```

6185 \renewcommand*{\acrfullfmt}[3]{%
6186 \glslink[##1]{##2}{\acronymfont{\glentryshort{##2}}##3\space
6187 (\glentrylong{##2})}%
6188 \renewcommand*{\Acrfullfmt}[3]{%
6189 \glslink[##1]{##2}{\acronymfont{\Glsentryshort{##2}}##3\space
6190 (\glentrylong{##2})}%
6191 \renewcommand*{\ACRfullfmt}[3]{%
6192 \glslink[##1]{##2}{%
6193 \mfirstucMakeUppercase{\acronymfont{\glentryshort{##2}}##3\space
6194 (\glentrylong{##2})}}}%
6195 \renewcommand*{\acrfullplfmt}[3]{%
6196 \glslink[##1]{##2}{\acronymfont{\glentryshortpl{##2}}##3\space
6197 (\glentrylongpl{##2})}%
6198 \renewcommand*{\Acrfullplfmt}[3]{%
6199 \glslink[##1]{##2}{\acronymfont{\Glsentryshortpl{##2}}##3\space
6200 (\glentrylongpl{##2})}%
6201 \renewcommand*{\ACRfullplfmt}[3]{%
6202 \glslink[##1]{##2}{%
6203 \mfirstucMakeUppercase{\acronymfont{\glentryshortpl{##2}}##3\space
6204 (\glentrylongpl{##2})}}}%

```

Similarly for \glentryfull etc:

```

6205 \renewcommand*{\glentryfull}[1]{%
6206 \acronymfont{\glentryshort{##1}}\space(\glentrylong{##1})}%
6207 \renewcommand*{\Glsentryfull}[1]{%
6208 \acronymfont{\Glsentryshort{##1}}\space(\glentrylong{##1})}%
6209 \renewcommand*{\glentryfullpl}[1]{%
6210 \acronymfont{\glentryshortpl{##1}}\space(\glentrylongpl{##1})}%
6211 \renewcommand*{\Glsentryfullpl}[1]{%
6212 \acronymfont{\Glsentryshortpl{##1}}\space(\glentrylongpl{##1})}%
6213 }

```

footnote-sc \textsc{<short>}\footnote{<long>} acronym style.

```

6214 \newacronymstyle{footnote-sc}%
6215 {%
6216 \GlsUseAcrEntryDisplayStyle{footnote}%
6217 }%

```

```

6218 {%
6219   \GlsUseAcrStyleDefs{footnote}%
6220   \renewcommand{\acronymentry}[1]{\acronymfont{\glentryshort{##1}}}
6221   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6222   \renewcommand*\{acrpluralsuffix}{\glsupacrpluralsuffix}%
6223 }%

```

footnote-sm \textsmaller{<short>}\footnote{<long>} acronym style.

```

6224 \newacronymstyle{footnote-sm}%
6225 {%
6226   \GlsUseAcrEntryDispStyle{footnote}%
6227 }%
6228 {%
6229   \GlsUseAcrStyleDefs{footnote}%
6230   \renewcommand{\acronymentry}[1]{\acronymfont{\glentryshort{##1}}}
6231   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6232   \renewcommand*\{acrpluralsuffix}{\glacrpluralsuffix}%
6233 }%

```

footnote-desc <short>\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

6234 \newacronymstyle{footnote-desc}%
6235 {%
6236   \GlsUseAcrEntryDispStyle{footnote}%
6237 }%
6238 {%
6239   \GlsUseAcrStyleDefs{footnote}%
6240   \renewcommand*\{GenericAcronymFields}{}%
6241   \renewcommand*\{acronymsort}[2]{##2}%
6242   \renewcommand*\{acronymentry}[1]{%
6243     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6244 }

```

footnote-sc-desc \textsc{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

6245 \newacronymstyle{footnote-sc-desc}%
6246 {%
6247   \GlsUseAcrEntryDispStyle{footnote-sc}%
6248 }%
6249 {%
6250   \GlsUseAcrStyleDefs{footnote-sc}%
6251   \renewcommand*\{GenericAcronymFields}{}%
6252   \renewcommand*\{acronymsort}[2]{##2}%
6253   \renewcommand*\{acronymentry}[1]{%
6254     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6255 }

```

footnote-sm-desc \textsmaller{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

6256 \newacronymstyle{footnote-sm-desc}%
6257 {%
6258   \GlsUseAcrEntryDisplayStyle{footnote-sm}%
6259 }%
6260 {%
6261   \GlsUseAcrStyleDefs{footnote-sm}%
6262   \renewcommand*{\GenericAcronymFields}{}%
6263   \renewcommand*{\acronymsort}[2]{##2}%
6264   \renewcommand*{\acronymentry}[1]{%
6265     \glstrylong{##1}\space (\acronymfont{\glstryshort{##1}})}%
6266 }

```

fineAcronymSynonyms

```

6267 \newcommand*{\DefineAcronymSynonyms}{%

```

Short form

\acs

```

6268   \let\acs\acrshort

```

First letter uppercase short form

\Acs

```

6269   \let\Acs\Acrshort

```

Plural short form

\acsp

```

6270   \let\acsp\acrshortpl

```

First letter uppercase plural short form

\Acsp

```

6271   \let\Acsp\Acrshortpl

```

Long form

\acl

```

6272   \let\acl\acllong

```

Plural long form

\aclp

```

6273   \let\aclp\acllongpl

```

First letter upper case long form

\Acl

```

6274   \let\Acl\Aclong

```

First letter upper case plural long form

```

\Aclp
6275 \let\Aclp\Acrlongpl

Full form

\acf
6276 \let\acf\acrfull

Plural full form

\acfp
6277 \let\acfp\acrfullpl

First letter upper case full form

\Acf
6278 \let\Acf\Acrfull

First letter upper case plural full form

\Acfp
6279 \let\Acfp\Acrfullpl

Standard form

\ac
6280 \let\ac\gls

First upper case standard form

\Ac
6281 \let\Ac\Gls

Standard plural form

\acp
6282 \let\acp\glspl

Standard first letter upper case plural form

\Acp
6283 \let\Acp\Glspl
6284 }

Define synonyms if required
6285 \ifglsacrshortcuts
6286 \DefineAcronymSynonyms
6287 \fi

```

These commands for setting the style are now deprecated but are kept for backward compatibility.

AcronymDisplayStyle Sets the default acronym display style for given glossary.

```
6288 \newcommand*{\SetDefaultAcronymDisplayStyle}[1]{%
6289   \def\glsentryfmt[#1]{\glsentryfmt}%
6290 }
```

defaultNewAcronymDef Sets up the acronym definition for the default style. The information is provided by the tokens `\glslabeltok`, `\glsshorttok`, `\glslongtok` and `\glskeylisttok`.

```
6291 \newcommand*{\DefaultNewAcronymDef}{%
6292   \edef\@do@newglossaryentry{%
6293     \noexpand\newglossaryentry{\the\glslabeltok}%
6294     {%
6295       type=\acronymtype,%
6296       name={\the\glsshorttok},%
6297       sort={\the\glsshorttok},%
6298       text={\the\glsshorttok},%
6299       first={\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
6300       plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6301       firstplural={\acrfullformat{\noexpand\expandonce\noexpand\@glo@longpl}%
6302                    {\noexpand\expandonce\noexpand\@glo@shortpl}},%
6303       short={\the\glsshorttok},%
6304       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6305       long={\the\glslongtok},%
6306       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6307       description={\the\glslongtok},%
6308       descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
```

Remaining options specified by the user:

```
6309     \the\glskeylisttok
6310   }%
6311 }%
6312 \let\@org@gls@assign@firstpl\gls@assign@firstpl
6313 \let\@org@gls@assign@plural\gls@assign@plural
6314 \let\@org@gls@assign@descplural\gls@assign@descplural
6315 \def\gls@assign@firstpl##1##2{%
6316   \@@gls@expand@field{##1}{firstpl}{##2}%
6317 }%
6318 \def\gls@assign@plural##1##2{%
6319   \@@gls@expand@field{##1}{plural}{##2}%
6320 }%
6321 \def\gls@assign@descplural##1##2{%
6322   \@@gls@expand@field{##1}{descplural}{##2}%
6323 }%
6324 \@do@newglossaryentry
6325 \let\gls@assign@firstpl\@org@gls@assign@firstpl
6326 \let\gls@assign@plural\@org@gls@assign@plural
6327 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
6328 }
```

DefaultAcronymStyle Set up the default acronym style:

```
6329 \newcommand*\SetDefaultAcronymStyle{%
```

Set the display style:

```
6330 \@for\@gls@type:=\@glsacronymlists\do{%
6331 \SetDefaultAcronymDisplayStyle{\@gls@type}%
6332 }%
```

Set up the definition of `\newacronym`:

```
6333 \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).

```
6334 \ifx\@glsacronymlists\@empty
6335 \def\@glo@type{\acronymtype}%
6336 \setkeys{glossentry}{##1}%
6337 \DeclareAcronymList{\@glo@type}%
6338 \SetDefaultAcronymDisplayStyle{\@glo@type}%
6339 \fi
6340 \glskeylisttok{##1}%
6341 \glslabeltok{##2}%
6342 \glsshorttok{##3}%
6343 \glslongtok{##4}%
6344 \newacronymhook
6345 \DefaultNewAcronymDef
6346 }%
6347 \renewcommand*\acrpluralsuffix{\glsacrpluralsuffix}%
6348 }
```

\acrfootnote Used by the footnote acronym styles.

```
6349 \newcommand*\acrfootnote[3]{\acrlinkfootnote{#1}{#2}{#3}}
```

\acrlinkfootnote

```
6350 \newcommand*\acrlinkfootnote[3]{%
6351 \footnote{\glslink[#1]{#2}{#3}}%
6352 }
```

\acrnoflinkfootnote

```
6353 \newcommand*\acrnoflinkfootnote[3]{%
6354 \footnote{#3}%
6355 }
```

AcronymDisplayStyle Sets the acronym display style for given glossary for the description and footnote combination.

```
6356 \newcommand*\SetDescriptionFootnoteAcronymDisplayStyle[1]{%
6357 \def\glsentryfmt[#1]{%
```



```

6358 \ifdefempty\glscustomtext
6359 {%
6360 \ifglssused{\glslabel}%
6361 {%
6362 \acronymfont{\glsgenentryfmt}%
6363 }%
6364 {%
6365 \firstacronymfont{\glsgenentryfmt}%
6366 \ifglsshassymbol{\glslabel}%
6367 {%
6368 \expandafter\protect\expandafter\acrfootnote\expandafter
6369 {\@gls@link@opts}{\@gls@link@label}%
6370 {%
6371 \glsifplural
6372 {\glsentrysymbolplural{\glslabel}}%
6373 {\glsentrysymbol{\glslabel}}%
6374 }%
6375 }%
6376 }%
6377 }%
6378 {\glscustomtext\glsinsert}%
6379 }%
6380 }

```

otnoteNewAcronymDef

```

6381 \newcommand*{\DescriptionFootnoteNewAcronymDef}{%
6382 \edef\@do@newglossaryentry{%
6383 \noexpand\newglossaryentry{\the\glslabelltok}%
6384 {%
6385 type=\acronymtype,%
6386 name={\noexpand\acronymfont{\the\glsshorttok}},%
6387 sort={\the\glsshorttok},%
6388 first={\the\glsshorttok},%
6389 firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6390 text={\the\glsshorttok},%
6391 plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6392 short={\the\glsshorttok},%
6393 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6394 long={\the\glslongtok},%
6395 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6396 symbol={\the\glslongtok},%
6397 symbolplural={\noexpand\expandonce\noexpand\@glo@longpl},%
6398 \the\glskeylisttok
6399 }%
6400 }%
6401 \let\@org@gls@assign@firstpl\gls@assign@firstpl
6402 \let\@org@gls@assign@plural\gls@assign@plural
6403 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
6404 \def\gls@assign@firstpl##1##2{%

```

```

6405 \@@gls@expand@field{##1}{firstpl}{##2}%
6406 }%
6407 \def\gls@assign@plural##1##2{%
6408 \@@gls@expand@field{##1}{plural}{##2}%
6409 }%
6410 \def\gls@assign@symbolplural##1##2{%
6411 \@@gls@expand@field{##1}{symbolplural}{##2}%
6412 }%
6413 \do@newglossaryentry
6414 \let\gls@assign@plural\@org@gls@assign@plural
6415 \let\gls@assign@firstpl\@org@gls@assign@firstpl
6416 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
6417 }

```

footnoteAcronymStyle 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.

```

6418 \newcommand*\SetDescriptionFootnoteAcronymStyle{%
6419 \renewcommand{\newacronym}[4][\]{%
6420 \ifx\@glsacronymlists\@empty
6421 \def\@glo@type{\acronymtype}%
6422 \setkeys{glossentry}{##1}%
6423 \DeclareAcronymList{\@glo@type}%
6424 \SetDescriptionFootnoteAcronymDisplayStyle{\@glo@type}%
6425 \fi
6426 \glskeylisttok{##1}%
6427 \glslabeltok{##2}%
6428 \glsshorttok{##3}%
6429 \glslongtok{##4}%
6430 \newacronymhook
6431 \DescriptionFootnoteNewAcronymDef
6432 }%

```

If footnote package option is specified, set the first use to append the long form (stored in symbol) as a footnote.

```

6433 \@for\@gls@type:=\@glsacronymlists\do{%
6434 \SetDescriptionFootnoteAcronymDisplayStyle{\@gls@type}%
6435 }%

```

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.

```

6436 \ifglsacrsmallcaps
6437 \renewcommand*\acronymfont{[1]{\textsc{##1}}}%
6438 \renewcommand*\acrpluralsuffix{\glsupacrpluralsuffix}%
6439 \else
6440 \ifglsacrsmaller
6441 \renewcommand*\acronymfont{[1]{\textsmaller{##1}}}%

```

```

6442 \fi
6443 \fi

```

Check for package option clash

```

6444 \ifglsacrdua
6445 \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’
6446 can’t both be set}{}%
6447 \fi
6448 }%

```

AcronymDisplayStyle Sets the acronym display style for given glossary with description and dua combination.

```

6449 \newcommand*{\SetDescriptionDUAAcronymDisplayStyle}[1]{%
6450 \defglsentryfmt[#1]{\glsgenentryfmt}%
6451 }

```

ionDUANewAcronymDef

```

6452 \newcommand*{\DescriptionDUANewAcronymDef}{%
6453 \edef\@do@newglossaryentry{%
6454 \noexpand\newglossaryentry{\the\glslabeltok}%
6455 {%
6456 type=\acronymtype,%
6457 name={\the\glslongtok},%
6458 sort={\the\glslongtok},%
6459 text={\the\glslongtok},%
6460 first={\the\glslongtok},%
6461 plural={\noexpand\expandonce\noexpand\@glo@longpl},%
6462 firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
6463 short={\the\glsshorttok},%
6464 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6465 long={\the\glslongtok},%
6466 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6467 symbol={\the\glsshorttok},%
6468 symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6469 \the\glskeylisttok
6470 }%
6471 }%
6472 \let\@org@gls@assign@firstpl\gls@assign@firstpl
6473 \let\@org@gls@assign@plural\gls@assign@plural
6474 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
6475 \def\gls@assign@firstpl##1##2{%
6476 \@@gls@expand@field{##1}{firstpl}{##2}%
6477 }%
6478 \def\gls@assign@plural##1##2{%
6479 \@@gls@expand@field{##1}{plural}{##2}%
6480 }%
6481 \def\gls@assign@symbolplural##1##2{%
6482 \@@gls@expand@field{##1}{symbolplural}{##2}%
6483 }%

```

```

6484 \do@newglossaryentry
6485 \let\gls@assign@firstpl\@org@gls@assign@firstpl
6486 \let\gls@assign@plural\@org@gls@assign@plural
6487 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
6488 }

```

tionDUAAcronymStyle 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.

```

6489 \newcommand*\SetDescriptionDUAAcronymStyle{%
6490   \ifglsacrsmallcaps
6491     \PackageError{glossaries}{Option clash: 'smallcaps' and 'dua'
6492       can't both be set}{}%
6493   \else
6494     \ifglsacrsmaller
6495       \PackageError{glossaries}{Option clash: 'smaller' and 'dua'
6496         can't both be set}{}%
6497     \fi
6498   \fi
6499   \renewcommand{\newacronym}[4][[]]{%
6500     \ifx\@glsacronymlists\@empty
6501       \def\@glo@type{\acronymtype}%
6502       \setkeys{glossentry}{##1}%
6503       \DeclareAcronymList{\@glo@type}%
6504       \SetDescriptionDUAAcronymDisplayStyle{\@glo@type}%
6505     \fi
6506     \glskeylisttok{##1}%
6507     \glslabeltok{##2}%
6508     \glsshorttok{##3}%
6509     \glslongtok{##4}%
6510     \newacronymhook
6511     \DescriptionDUANewAcronymDef
6512   }%

```

Set display.

```

6513 \@for\@gls@type:=\@glsacronymlists\do{%
6514   \SetDescriptionDUAAcronymDisplayStyle{\@gls@type}%
6515 }%
6516 }%

```

AcronymDisplayStyle Sets the acronym display style for given glossary using the description setting (but not footnote or dua).

```

6517 \newcommand*\SetDescriptionAcronymDisplayStyle}[1]{%
6518   \defglsentryfmt[#1]{%
6519     \ifdefempty\glscustomtext
6520     {%
6521       \ifglsused{\glslabel}%
6522       {%

```

Move the inserted text outside of \acronymfont

```

6523     \let\gls@org@insert\glsinsert
6524     \let\glsinsert\@empty
6525     \acronymfont{\gls@org@insert
6526 }%
6527 {%
6528     \gls@entryfmt
6529     \ifgls@has@symbol{\gls@label}%
6530     {%
6531         \gls@ifplural
6532         {%
6533             \def\@glo@symbol{\gls@entrysymbolplural{\gls@label}}%
6534         }%
6535         {%
6536             \def\@glo@symbol{\gls@entrysymbol{\gls@label}}%
6537         }%
6538         \space(\protect\firstacronymfont
6539         {\gls@caps@case
6540         {\@glo@symbol}
6541         {\@glo@symbol}
6542         {\mfirstucMakeUppercase{\@glo@symbol}}})%
6543     }%
6544 }%
6545 }%
6546 }%
6547 {\gls@customtext\glsinsert}%
6548 }%
6549 }

```

ip tionNewAcronymDef

```

6550 \newcommand*{\DescriptionNewAcronymDef}{%
6551     \edef\@do@newglossaryentry{%
6552         \noexpand\newglossaryentry{\the\gls@labeltok}%
6553         {%
6554             type=\acronymtype,%
6555             name={\noexpand
6556                 \acrnameformat{\the\glsshorttok}{\the\gls@longtok}},%
6557             sort={\the\glsshorttok},%
6558             first={\the\gls@longtok},%
6559             firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
6560             text={\the\glsshorttok},%
6561             plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6562             short={\the\glsshorttok},%
6563             shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6564             long={\the\gls@longtok},%
6565             longplural={\the\gls@longtok\noexpand\acrpluralsuffix},%
6566             symbol={\noexpand\@glo@text},%
6567             symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6568             \the\gls@keylisttok}%

```

```

6569 }%
6570 \let\@org@gl@s@assign@firstpl\gl@s@assign@firstpl
6571 \let\@org@gl@s@assign@plural\gl@s@assign@plural
6572 \let\@org@gl@s@assign@symbolplural\gl@s@assign@symbolplural
6573 \def\gl@s@assign@firstpl##1##2{%
6574   \@@gl@s@expand@field{##1}{firstpl}{##2}%
6575 }%
6576 \def\gl@s@assign@plural##1##2{%
6577   \@@gl@s@expand@field{##1}{plural}{##2}%
6578 }%
6579 \def\gl@s@assign@symbolplural##1##2{%
6580   \@@gl@s@expand@field{##1}{symbolplural}{##2}%
6581 }%
6582 \do@newglossaryentry
6583 \let\gl@s@assign@firstpl\@org@gl@s@assign@firstpl
6584 \let\gl@s@assign@plural\@org@gl@s@assign@plural
6585 \let\gl@s@assign@symbolplural\@org@gl@s@assign@symbolplural
6586 }

```

DescriptionAcronymStyle 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.

```

6587 \newcommand*{\SetDescriptionAcronymStyle}{%
6588   \renewcommand{\newacronym}[4][\]{%
6589     \ifx\@gl@s@acronymlists\@empty
6590       \def\@glo@type{\acronymtype}%
6591       \setkeys{glossentry}{##1}%
6592       \DeclareAcronymList{\@glo@type}%
6593       \SetDescriptionAcronymDisplayStyle{\@glo@type}%
6594     \fi
6595     \gl@keylisttok{##1}%
6596     \gl@labeltok{##2}%
6597     \gl@shorttok{##3}%
6598     \gl@longtok{##4}%
6599     \newacronymhook
6600     \DescriptionNewAcronymDef
6601   }%

```

Set display.

```

6602 \@for\@gl@s@type:=\@gl@s@acronymlists\do{%
6603   \SetDescriptionAcronymDisplayStyle{\@gl@s@type}%
6604 }%

```

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.

```

6605 \ifgl@acrsmallcaps
6606   \renewcommand{\acronymfont}[1]{\textsc{##1}}

```

```

6607 \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6608 \else
6609 \ifglsmaller
6610 \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
6611 \fi
6612 \fi
6613 }%

```

AcronymDisplayStyle Sets the acronym display style for given glossary with footnote setting (but not description or dua).

```

6614 \newcommand*{\SetFootnoteAcronymDisplayStyle}[1]{%
6615 \defglentryfmt[#1]{%

```

```

6616 \ifdefempty\glscustomtext
6617 {%

```

Move the inserted text outside of \acronymfont

```

6618 \let\gls@org@insert\glsinsert
6619 \let\glsinsert\@empty
6620 \ifglused{\glslabel}%
6621 {%
6622 \acronymfont{\glsgenentryfmt}\gls@org@insert
6623 }%
6624 {%
6625 \firstacronymfont{\glsgenentryfmt}\gls@org@insert
6626 \ifglshaslong{\glslabel}%
6627 {%
6628 \expandafter\protect\expandafter\acrfootnote\expandafter
6629 {\@gls@link@opts}{\@gls@link@label}%
6630 {%
6631 \glsifplural
6632 {\glsentrylongpl{\glslabel}}%
6633 {\glsentrylong{\glslabel}}%
6634 }%
6635 }%
6636 }%
6637 }%
6638 }%
6639 {\glscustomtext\glsinsert}%
6640 }%
6641 }

```

FootnoteNewAcronymDef

```

6642 \newcommand*{\FootnoteNewAcronymDef}{%
6643 \edef\@do@newglossaryentry{%
6644 \noexpand\newglossaryentry{\the\glslabeltok}%
6645 {%
6646 type=\acronymtype,%
6647 name={\noexpand\acronymfont{\the\glsshorttok}},%

```

```

6648     sort={\the\glsshorttok},%
6649     text={\the\glsshorttok},%
6650     plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6651     first={\the\glsshorttok},%
6652     firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6653     short={\the\glsshorttok},%
6654     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6655     long={\the\glslongtok},%
6656     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6657     description={\the\glslongtok},%
6658     descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
6659     \the\glskeylisttok
6660 }%
6661 }%
6662 \let\@org@gls@assign@plural\gls@assign@plural
6663 \let\@org@gls@assign@firstpl\gls@assign@firstpl
6664 \let\@org@gls@assign@descplural\gls@assign@descplural
6665 \def\gls@assign@firstpl##1##2{%
6666   \@@gls@expand@field{##1}{firstpl}{##2}%
6667 }%
6668 \def\gls@assign@plural##1##2{%
6669   \@@gls@expand@field{##1}{plural}{##2}%
6670 }%
6671 \def\gls@assign@descplural##1##2{%
6672   \@@gls@expand@field{##1}{descplural}{##2}%
6673 }%
6674 \do@newglossaryentry
6675 \let\gls@assign@plural\@org@gls@assign@plural
6676 \let\gls@assign@firstpl\@org@gls@assign@firstpl
6677 \let\gls@assign@descplural\@org@gls@assign@descplural
6678 }

```

FootnoteAcronymStyle 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.

```

6679 \newcommand*\SetFootnoteAcronymStyle{%
6680   \renewcommand{\newacronym}[4][\]{%
6681     \ifx\@glsacronymlists\@empty
6682       \def\@glo@type{\acronymtype}%
6683       \setkeys{glossentry}{##1}%
6684       \DeclareAcronymList{\@glo@type}%
6685       \SetFootnoteAcronymDisplayStyle{\@glo@type}%
6686     \fi
6687     \glskeylisttok{##1}%
6688     \glslabeltok{##2}%
6689     \glsshorttok{##3}%
6690     \glslongtok{##4}%
6691     \newacronymhook
6692     \FootnoteNewAcronymDef

```



```
6693 }%
```

Set display

```
6694 \@for\@gls@type:=\@glsacronymlists\do{%
6695   \SetFootnoteAcronymDisplayStyle{\@gls@type}%
6696 }%
```

Redefine `\acronymfont` if small caps required. The plural suffix is set in an up-right font so that it remains in normal lower case, otherwise it looks as though it's part of the acronym.

```
6697 \ifglsacrsmallcaps
6698   \renewcommand*\acronymfont}[1]{\textsc{##1}}%
6699   \renewcommand*\acrpluralsuffix{\glsupacrpluralsuffix}%
6700 \else
6701   \ifglsacrsmaller
6702     \renewcommand*\acronymfont}[1]{\textsmaller{##1}}%
6703   \fi
6704 \fi
```

Check for option clash

```
6705 \ifglsacrdua
6706   \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’
6707     can’t both be set}{}%
6708 \fi
6709 }%
```

`\glsdoparenifnotempty` 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.

```
6710 \DeclareRobustCommand*\glsdoparenifnotempty}[2]{%
6711   \protected@edef\gls@tmp{#1}%
6712   \ifdefempty\gls@tmp
6713     {}%
6714   {%
6715     \ifx\gls@tmp\@gls@default@value
6716       \else
6717         \space (#2{#1})%
6718       \fi
6719     }%
6720 }
```

`AcronymDisplayStyle` Sets the acronym display style for given glossary where neither footnote nor description is required, but smallcaps or smaller specified.

```
6721 \newcommand*\SetSmallAcronymDisplayStyle}[1]{%
6722   \defglentryfmt[#1]{%
6723     \ifdefempty\glscustomtext
6724       {%
```

Move the inserted text outside of \acronymfont

```

6725 \let\gls@org@insert\glsinsert
6726 \let\glsinsert\@empty
6727 \ifglsused{\glslabel}%
6728 {%
6729 \acronymfont{\glsgetentryfmt}\gls@org@insert
6730 }%
6731 {%
6732 \glsgetentryfmt
6733 \ifglshassymbol{\glslabel}%
6734 {%
6735 \glsifplural
6736 {%
6737 \def\@glo@symbol{\glsentrysymbolplural{\glslabel}}%
6738 }%
6739 {%
6740 \def\@glo@symbol{\glsentrysymbol{\glslabel}}%
6741 }%
6742 \space
6743 (\glscapscase
6744 {\firstacronymfont{\@glo@symbol}}%
6745 {\firstacronymfont{\@glo@symbol}}%
6746 {\firstacronymfont{\mfirstucMakeUppercase{\@glo@symbol}}})%
6747 }%
6748 {}%
6749 }%
6750 }%
6751 {\glscustomtext\glsinsert}%
6752 }%
6753 }

```

\SmallNewAcronymDef

```

6754 \newcommand*{\SmallNewAcronymDef}{%
6755 \edef\@do@newglossaryentry{%
6756 \noexpand\newglossaryentry{\the\glslabeltok}%
6757 {%
6758 type=\acronymtype,%
6759 name={\noexpand\acronymfont{\the\glsshorttok}},%
6760 sort={\the\glsshorttok},%
6761 text={\the\glsshorttok},%

```

Default to the short plural.

```

6762 plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6763 first={\the\glslongtok},%

```

Default to the long plural.

```

6764 firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
6765 short={\the\glsshorttok},%
6766 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6767 long={\the\glslongtok},%

```

```

6768     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6769     description={\noexpand\@glo@first},%
6770     descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
6771     symbol={\the\glsshorttok},%

```

Default to the short plural.

```

6772     symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6773     \the\glskeylisttok
6774 }%
6775 }%
6776 \let\@org@gls@assign@firstpl\gls@assign@firstpl
6777 \let\@org@gls@assign@plural\gls@assign@plural
6778 \let\@org@gls@assign@descplural\gls@assign@descplural
6779 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
6780 \def\gls@assign@firstpl##1##2{%
6781     \@@gls@expand@field{##1}{firstpl}{##2}%
6782 }%
6783 \def\gls@assign@plural##1##2{%
6784     \@@gls@expand@field{##1}{plural}{##2}%
6785 }%
6786 \def\gls@assign@descplural##1##2{%
6787     \@@gls@expand@field{##1}{descplural}{##2}%
6788 }%
6789 \def\gls@assign@symbolplural##1##2{%
6790     \@@gls@expand@field{##1}{symbolplural}{##2}%
6791 }%
6792 \do@newglossaryentry
6793 \let\gls@assign@firstpl\@org@gls@assign@firstpl
6794 \let\gls@assign@plural\@org@gls@assign@plural
6795 \let\gls@assign@descplural\@org@gls@assign@descplural
6796 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
6797 }

```

etSmallAcronymStyle 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.

```

6798 \newcommand*\SetSmallAcronymStyle{%
6799     \renewcommand{\newacronym}[4][\]{%
6800         \ifx\@glsacronymlists\@empty
6801             \def\@glo@type{\acronymtype}%
6802             \setkeys{glossentry}{##1}%
6803             \DeclareAcronymList{\@glo@type}%
6804             \SetSmallAcronymDisplayStyle{\@glo@type}%
6805         \fi
6806         \glskeylisttok{##1}%
6807         \glslabeltok{##2}%
6808         \glsshorttok{##3}%
6809         \glslongtok{##4}%
6810         \newacronymhook
6811         \SmallNewAcronymDef
6812     }%

```

Change the display since first only contains long form.

```
6813 \@for\@gls@type:=\@glsacronymlists\do{%
6814 \SetSmallAcronymDisplayStyle{\@gls@type}%
6815 }%
```

Redefine \acronymfont if small caps required. The plural suffix is set in an up-right font so that it remains in normal lower case, otherwise it looks as though it's part of the acronym.

```
6816 \ifglsacrsmallcaps
6817 \renewcommand*{\acronymfont}[1]{\textsc{##1}}
6818 \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6819 \else
6820 \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}
6821 \fi
```

check for option clash

```
6822 \ifglsacrdua
6823 \ifglsacrsmallcaps
6824 \PackageError{glossaries}{Option clash: ‘smallcaps’ and ‘dua’
6825 can’t both be set}{}%
6826 \else
6827 \PackageError{glossaries}{Option clash: ‘smaller’ and ‘dua’
6828 can’t both be set}{}%
6829 \fi
6830 \fi
6831 }%
```

`\SetDUADisplayStyle` Sets the acronym display style for given glossary with dua setting.

```
6832 \newcommand*{\SetDUADisplayStyle}[1]{%
6833 \defglsentryfmt[##1]{\glsentryfmt}%
6834 }
```

`\DUANewAcronymDef`

```
6835 \newcommand*{\DUANewAcronymDef}{%
6836 \edef\@do@newglossaryentry{%
6837 \noexpand\newglossaryentry{\the\glslabeltok}%
6838 {%
6839 type=\acronymtype,%
6840 name={\the\glsshorttok},%
6841 text={\the\glslongtok},%
6842 first={\the\glslongtok},%
6843 plural={\noexpand\expandonce\noexpand\@glo@longpl},%
6844 firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
6845 short={\the\glsshorttok},%
6846 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6847 long={\the\glslongtok},%
6848 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6849 description={\the\glslongtok},%
6850 descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
```

```

6851     symbol={\the\glsshorttok},%
6852     symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6853     \the\glskeylisttok
6854   }%
6855 }%
6856 \let\@org@gls@assign@firstpl\gls@assign@firstpl
6857 \let\@org@gls@assign@plural\gls@assign@plural
6858 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
6859 \let\@org@gls@assign@descplural\gls@assign@descplural
6860 \def\gls@assign@firstpl##1##2{%
6861   \@@gls@expand@field{##1}{firstpl}{##2}%
6862 }%
6863 \def\gls@assign@plural##1##2{%
6864   \@@gls@expand@field{##1}{plural}{##2}%
6865 }%
6866 \def\gls@assign@symbolplural##1##2{%
6867   \@@gls@expand@field{##1}{symbolplural}{##2}%
6868 }%
6869 \def\gls@assign@descplural##1##2{%
6870   \@@gls@expand@field{##1}{descplural}{##2}%
6871 }%
6872 \do@newglossaryentry
6873 \let\gls@assign@firstpl\@org@gls@assign@firstpl
6874 \let\gls@assign@plural\@org@gls@assign@plural
6875 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
6876 \let\gls@assign@descplural\@org@gls@assign@descplural
6877 }

```

`\SetDUASyle` Always expand acronyms.

```

6878 \newcommand*\SetDUASyle{%
6879   \renewcommand{\newacronym}[4][[]]{%
6880     \ifx\@glsacronymlists\@empty
6881       \def\@glo@type{\acronymtype}%
6882       \setkeys{glossentry}{##1}%
6883       \DeclareAcronymList{\@glo@type}%
6884       \SetDUADisplayStyle{\@glo@type}%
6885     \fi
6886     \glskeylisttok{##1}%
6887     \glslabeltok{##2}%
6888     \glsshorttok{##3}%
6889     \glslongtok{##4}%
6890     \newacronymhook
6891     \DUANewAcronymDef
6892   }%
6893   \@for\@gls@type:=\@glsacronymlists\do{%
6894     \SetDUADisplayStyle{\@gls@type}%
6895   }%
6896 }

```

\SetAcronymStyle

```
6897 \newcommand*{\SetAcronymStyle}{%
6898   \SetDefaultAcronymStyle
6899   \ifglssacrdescription
6900     \ifglssacrfootnote
6901       \SetDescriptionFootnoteAcronymStyle
6902     \else
6903       \ifglssacrdua
6904         \SetDescriptionDUAAcronymStyle
6905       \else
6906         \SetDescriptionAcronymStyle
6907     \fi
6908   \fi
6909 \else
6910   \ifglssacrfootnote
6911     \SetFootnoteAcronymStyle
6912   \else
6913     \ifthenelse{\boolean{glssacrsmalldcaps}}{\OR
6914       \boolean{glssacrsmalld}}{%
6915       {%
6916         \SetSmallAcronymStyle
6917       }%
6918     {%
6919       \ifglssacrdua
6920         \SetDUASStyle
6921     \fi
6922   }%
6923 \fi
6924 \fi
6925 }
```

Set the acronym style according to the package options

6926 \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.

tCustomDisplayStyle Sets the acronym display style.

```
6927 \newcommand*{\SetCustomDisplayStyle}[1]{%
6928   \defglssentryfmt[#1]{\glsgenentryfmt}%
6929 }
```

CustomAcronymFields

```
6930 \newcommand*{\CustomAcronymFields}{%
6931   name={\the\glsshorttok},%
6932   description={\the\glslongtok},%
```

```

6933 first={\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
6934 firstplural={\acrfullformat
6935   {\noexpand\glsentrylongpl{\the\glslabeltok}}}%
6936   {\noexpand\glsentryshortpl{\the\glslabeltok}}},%

6937 text={\the\glsshorttok},%
6938 plural={\the\glsshorttok\noexpand\acrpluralsuffix}%
6939 }

```

CustomNewAcronymDef

```

6940 \newcommand*{\CustomNewAcronymDef}{%
6941   \protected@edef\@do@newglossaryentry{%
6942     \noexpand\newglossaryentry{\the\glslabeltok}%
6943     {%
6944       type=\acronymtype,%
6945       short={\the\glsshorttok},%
6946       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6947       long={\the\glslongtok},%
6948       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6949       user1={\the\glsshorttok},%
6950       user2={\the\glsshorttok\noexpand\acrpluralsuffix},%
6951       user3={\the\glslongtok},%
6952       user4={\the\glslongtok\noexpand\acrpluralsuffix},%
6953       \CustomAcronymFields,%
6954       \the\glskeylisttok
6955     }%
6956   }%
6957   \@do@newglossaryentry
6958 }

```

\SetCustomStyle

```

6959 \newcommand*{\SetCustomStyle}{%
6960   \renewcommand{\newacronym}[4][\]{%
6961     \ifx\@glsacronymlists\@empty
6962       \def\@glo@type{\acronymtype}%
6963       \setkeys{glossentry}{##1}%
6964       \DeclareAcronymList{\@glo@type}%
6965       \SetCustomDisplayStyle{\@glo@type}%
6966     \fi
6967     \glskeylisttok{##1}%
6968     \glslabeltok{##2}%
6969     \glsshorttok{##3}%
6970     \glslongtok{##4}%
6971     \newacronymhook
6972     \CustomNewAcronymDef
6973   }%

```

Set the display

```

6974   \@for\@gls@type:=\@glsacronymlists\do{%
6975     \SetCustomDisplayStyle{\@gls@type}%

```

```
6976 }%
6977 }
```

1.18 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.

```
6978 \RequirePackage{glossary-hypernav}
```

The styles that use list-like environments. These are not loaded if the `nolist` option is used:

```
6979 \@gls@loadlist
```

The styles that use the `longtable` environment. These are not loaded if the `no-long package` option is used.

```
6980 \@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.

```
6981 \@gls@loadsuper
```

The tree-like styles. These are not loaded if the `notree` package option is used.

```
6982 \@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.

```
6983 \ifx\@glossary@default@style\relax
6984 \else
6985   \setglossarystyle{\@glossary@default@style}
6986 \fi
```

1.19 Debugging Commands

`\showgloparent` `\showgloparent{<label>}`

```
6987 \newcommand*{\showgloparent}[1]{%
6988   \expandafter\show\csname glo@\glsdetoklabel{#1}@parent\endcsname
6989 }
```

`\showglolevel` `\showglolevel{<label>}`

```
6990 \newcommand*{\showglolevel}[1]{%
6991   \expandafter\show\csname glo@\glsdetoklabel{#1}@level\endcsname
6992 }
```


`\showglotext` `\showglotext{<label>}`

```
6993 \newcommand*{\showglotext}[1]{%
6994   \expandafter\show\csname glo@glstetoklabel{#1}@text\endcsname
6995 }
```

`\showgloplural` `\showgloplural{<label>}`

```
6996 \newcommand*{\showgloplural}[1]{%
6997   \expandafter\show\csname glo@glstetoklabel{#1}@plural\endcsname
6998 }
```

`\showglofirst` `\showglofirst{<label>}`

```
6999 \newcommand*{\showglofirst}[1]{%
7000   \expandafter\show\csname glo@glstetoklabel{#1}@first\endcsname
7001 }
```

`\showglofirstpl` `\showglofirstpl{<label>}`

```
7002 \newcommand*{\showglofirstpl}[1]{%
7003   \expandafter\show\csname glo@glstetoklabel{#1}@firstpl\endcsname
7004 }
```

`\showglotype` `\showglotype{<label>}`

```
7005 \newcommand*{\showglotype}[1]{%
7006   \expandafter\show\csname glo@glstetoklabel{#1}@type\endcsname
7007 }
```

`\showglocounter` `\showglocounter{<label>}`

```
7008 \newcommand*{\showglocounter}[1]{%
7009   \expandafter\show\csname glo@glstetoklabel{#1}@counter\endcsname
7010 }
```

\showglouserii \showglouserii{\label{}}

```
7011 \newcommand*{\showglouserii}[1]{%
7012   \expandafter\show\csname glo@\glsdetoklabel{#1}@userii\endcsname
7013 }
```

\showglouseriii \showglouseriii{\label{}}

```
7014 \newcommand*{\showglouseriii}[1]{%
7015   \expandafter\show\csname glo@\glsdetoklabel{#1}@useriii\endcsname
7016 }
```

\showglouseriv \showglouseriv{\label{}}

```
7017 \newcommand*{\showglouseriv}[1]{%
7018   \expandafter\show\csname glo@\glsdetoklabel{#1}@useriv\endcsname
7019 }
```

\showglouserv \showglouserv{\label{}}

```
7020 \newcommand*{\showglouserv}[1]{%
7021   \expandafter\show\csname glo@\glsdetoklabel{#1}@userv\endcsname
7022 }
```

\showglouservi \showglouservi{\label{}}

```
7023 \newcommand*{\showglouservi}[1]{%
7024   \expandafter\show\csname glo@\glsdetoklabel{#1}@uservi\endcsname
7025 }
```

\showglouservi \showglouservi{\label{}}

```
7026 \newcommand*{\showglouservi}[1]{%
7027   \expandafter\show\csname glo@\glsdetoklabel{#1}@uservi\endcsname
7028 }
```

`\showglo`name `\showglo`name{*\label*}

```
7029 \newcommand*{\showglo
```

name}[1]{%
7030 \expandafter\show\csname glo@\glsdetoklabel{#1}@name\endcsname
7031 }

`\showglo`desc `\showglo`desc{*\label*}

```
7032 \newcommand*{\showglo
```

desc}[1]{%
7033 \expandafter\show\csname glo@\glsdetoklabel{#1}@desc\endcsname
7034 }

`\showglo`descplural `\showglo`descplural{*\label*}

```
7035 \newcommand*{\showglo
```

descplural}[1]{%
7036 \expandafter\show\csname glo@\glsdetoklabel{#1}@descplural\endcsname
7037 }

`\showglo`sort `\showglo`sort{*\label*}

```
7038 \newcommand*{\showglo
```

sort}[1]{%
7039 \expandafter\show\csname glo@\glsdetoklabel{#1}@sort\endcsname
7040 }

`\showglo`symbol `\showglo`symbol{*\label*}

```
7041 \newcommand*{\showglo
```

symbol}[1]{%
7042 \expandafter\show\csname glo@\glsdetoklabel{#1}@symbol\endcsname
7043 }

`\showglo`symbolplural `\showglo`symbolplural{*\label*}

```
7044 \newcommand*{\showglo
```

symbolplural}[1]{%
7045 \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolplural\endcsname
7046 }

`\showgloshort` `\showgloshort{<label>}`

```
7047 \newcommand*{\showgloshort}[1]{%
7048   \expandafter\show\csname glo@\glsdetoklabel{#1}@short\endcsname
7049 }
```

`\showglolong` `\showglolong{<label>}`

```
7050 \newcommand*{\showglolong}[1]{%
7051   \expandafter\show\csname glo@\glsdetoklabel{#1}@long\endcsname
7052 }
```

`\showgloindex` `\showgloindex{<label>}`

```
7053 \newcommand*{\showgloindex}[1]{%
7054   \expandafter\show\csname glo@\glsdetoklabel{#1}@index\endcsname
7055 }
```

`\showgloflag` `\showgloflag{<label>}`

```
7056 \newcommand*{\showgloflag}[1]{%
7057   \expandafter\show\csname ifglo@\glsdetoklabel{#1}@flag\endcsname
7058 }
```

`\showgloloclist` `\showgloloclist{<label>}`

```
7059 \newcommand*{\showgloloclist}[1]{%
7060   \expandafter\show\csname glo@\glsdetoklabel{#1}@loclist\endcsname
7061 }
```

`\showacronymlists` `\showacronymlists`

Show list of glossaries that have been flagged as a list of acronyms.

```
7062 \newcommand*{\showacronymlists}{%
7063   \show\@glsacronymlists
7064 }
```

`\showglossaries` `\showglossaries`

Show list of defined glossaries.

```
7065 \newcommand*{\showglossaries}{%  
7066   \show\@glo@types  
7067 }
```

`\showglossaryin` `\showglossaryin{<glossary-label>}`

Show the ‘in’ extension for the given glossary.

```
7068 \newcommand*{\showglossaryin}[1]{%  
7069   \expandafter\show\csname @glotype@#1@in\endcsname  
7070 }
```

`\showglossaryout` `\showglossaryout{<glossary-label>}`

Show the ‘out’ extension for the given glossary.

```
7071 \newcommand*{\showglossaryout}[1]{%  
7072   \expandafter\show\csname @glotype@#1@out\endcsname  
7073 }
```

`\showglossarytitle` `\showglossarytitle{<glossary-label>}`

Show the title for the given glossary.

```
7074 \newcommand*{\showglossarytitle}[1]{%  
7075   \expandafter\show\csname @glotype@#1@title\endcsname  
7076 }
```

`\showglossarycounter` `\showglossarycounter{<glossary-label>}`

Show the counter for the given glossary.

```
7077 \newcommand*{\showglossarycounter}[1]{%  
7078   \expandafter\show\csname @glotype@#1@counter\endcsname  
7079 }
```

`\showglossaryentries` `\showglossaryentries{<glossary-label>}`

Show the list of entry labels for the given glossary.

```
7080 \newcommand*{\showglossaryentries}[1]{%  
7081   \expandafter\show\csname glolist@#1\endcsname  
7082 }
```

1.20 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.

```
7083 \csname ifglscompatible-2.07\endcsname
7084 \RequirePackage{glossaries-compatible-207}
7085 \fi
```

2 Prefix Support (glossaries-prefix Code)

This package provides a means of adding prefixes to your glossary entries. For example, you may want to use “a `\gls{<label>}`” on first use but use “an `\gls{<label>}`” on subsequent use.

```
7086 \NeedsTeXFormat{LaTeX2e}
7087 \ProvidesPackage{glossaries-prefix}[2014/07/30 v4.08 (NLCT)]

Pass all options to glossaries:
7088 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}

Process options:
7089 \ProcessOptions

Load glossaries:
7090 \RequirePackage{glossaries}

Add the new keys:
7091 \define@key{glossentry}{prefixfirst}{\def\@glo@entryprefixfirst{#1}}%
7092 \define@key{glossentry}{prefixfirstplural}{\def\@glo@entryprefixfirstplural{#1}}%
7093 \define@key{glossentry}{prefix}{\def\@glo@entryprefix{#1}}%
7094 \define@key{glossentry}{prefixplural}{\def\@glo@entryprefixplural{#1}}%

Add them to \@gls@keymap:
7095 \appto\@gls@keymap{,%
7096   {prefixfirst}{prefixfirst},%
7097   {prefixfirstplural}{prefixfirstplural},%
7098   {prefix}{prefix},%
7099   {prefixplural}{prefixplural}}%
7100 }
```

Set the default values:

```

7101 \appto\@newglossaryentryprehook{%
7102   \def\@glo@entryprefix{}%
7103   \def\@glo@entryprefixplural{}%
7104   \let\@glo@entryprefixfirst\@gls@default@value
7105   \let\@glo@entryprefixfirstplural\@gls@default@value
7106 }

```

Set the assignment code:

```

7107 \appto\@newglossaryentryposthook{%
7108   \gls@assign@field{\@glo@label}{prefix}{\@glo@entryprefix}%
7109   \gls@assign@field{\@glo@label}{prefixplural}{\@glo@entryprefixplural}%

```

If prefixfirst has not been supplied, make it the same as prefix.

```

7110   \expandafter\gls@assign@field\expandafter
7111     {\csname glo@\@glo@label @prefix\endcsname}{\@glo@label}{prefixfirst}%
7112     {\@glo@entryprefixfirst}%

```

If prefixfirstplural has not been supplied, make it the same as prefixplural.

```

7113   \expandafter\gls@assign@field\expandafter
7114     {\csname glo@\@glo@label @prefixplural\endcsname}{\@glo@label}%
7115     {prefixfirstplural}{\@glo@entryprefixfirstplural}%
7116 }

```

Define commands to access these fields:

glsentryprefixfirst

```

7117 \newcommand*\{glsentryprefixfirst}[1]{\csuse{glo@#1@prefixfirst}}

```

ryprefixfirstplural

```

7118 \newcommand*\{glsentryprefixfirstplural}[1]{\csuse{glo@#1@prefixfirstplural}}

```

\glsentryprefix

```

7119 \newcommand*\{glsentryprefix}[1]{\csuse{glo@#1@prefix}}

```

lsentryprefixplural

```

7120 \newcommand*\{glsentryprefixplural}[1]{\csuse{glo@#1@prefixplural}}

```

Now for the initial upper case variants:

Glsentryprefixfirst

```

7121 \newrobustcmd*\{Glsentryprefixfirst}[1]{%
7122   \protected@edef\@glo@text{\csname glo@#1@prefixfirst\endcsname}%
7123   \xmakefirstuc\@glo@text
7124 }

```

ryprefixfirstplural

```

7125 \newrobustcmd*\{Glsentryprefixfirstplural}[1]{%
7126   \protected@edef\@glo@text{\csname glo@#1@prefixfirstplural\endcsname}%
7127   \xmakefirstuc\@glo@text
7128 }

```

\Glsentryprefix

```
7129 \newrobustcmd*{\Glsentryprefix}[1]{%
7130   \protected@edef\@glo@text{\csname glo@#1@prefix\endcsname}%
7131   \xmakefirstuc\@glo@text
7132 }
```

\Glsentryprefixplural

```
7133 \newrobustcmd*{\Glsentryprefixplural}[1]{%
7134   \protected@edef\@glo@text{\csname glo@#1@prefixplural\endcsname}%
7135   \xmakefirstuc\@glo@text
7136 }
```

Define commands to determine if the prefix keys have been set:

\ifglshasprefix

```
7137 \newcommand*{\ifglshasprefix}[3]{%
7138   \ifcempty{glo@#1@prefix}%
7139   {#3}%
7140   {#2}%
7141 }
```

\ifglshasprefixplural

```
7142 \newcommand*{\ifglshasprefixplural}[3]{%
7143   \ifcempty{glo@#1@prefixplural}%
7144   {#3}%
7145   {#2}%
7146 }
```

\ifglshasprefixfirst

```
7147 \newcommand*{\ifglshasprefixfirst}[3]{%
7148   \ifcempty{glo@#1@prefixfirst}%
7149   {#3}%
7150   {#2}%
7151 }
```

\ifglshasprefixfirstplural

```
7152 \newcommand*{\ifglshasprefixfirstplural}[3]{%
7153   \ifcempty{glo@#1@prefixfirstplural}%
7154   {#3}%
7155   {#2}%
7156 }
```

Define commands that insert the prefix before commands like \gls:

\pgls

```
7157 \newrobustcmd{\pgls}{\@gls@hyp@opt\@pgls}
```


\@pgls Unstarred version.

```
7158 \newcommand*\@pgls}[2] [] {%
7159   \new@ifnextchar [%
7160     {\@pgls@{#1}{#2}}%
7161     {\@pgls@{#1}{#2} []}%
7162 }
```

\@pgls@ Read in the final optional argument:

```
7163 \def\@pgls@#1#2[#3] {%
7164   \glsdoifexists{#2}%
7165   {%
7166     \ifglsused{#2}%
7167     {%
7168       \glsentryprefix{#2}%
7169     }%
7170     {%
7171       \glsentryprefixfirst{#2}%
7172     }%
7173     \@gls@{#1}{#2}[#3]%
7174   }%
7175 }
```

Similarly for the plural version:

\pglsp1

```
7176 \newrobustcmd{\pglsp1}{\@gls@hyp@opt\@pglsp1}
```

\@pglsp1 Unstarred version.

```
7177 \newcommand*\@pglsp1}[2] [] {%
7178   \new@ifnextchar [%
7179     {\@pglsp1@{#1}{#2}}%
7180     {\@pglsp1@{#1}{#2} []}%
7181 }
```

\@pglsp1@ Read in the final optional argument:

```
7182 \def\@pglsp1@#1#2[#3] {%
7183   \glsdoifexists{#2}%
7184   {%
7185     \ifglsused{#2}%
7186     {%
7187       \glsentryprefixplural{#2}%
7188     }%
7189     {%
7190       \glsentryprefixfirstplural{#2}%
7191     }%
7192     \@glspl@{#1}{#2}[#3]%
7193   }%
7194 }
```

Now for the first letter upper case versions:

\Pgl's

```
7195 \newrobustcmd{\Pgl's}{\@gls@hyp@opt\@Pgl's}
```

\@Pgl's Unstarred version.

```
7196 \newcommand*{\@Pgl's}[2][ ]{%
7197   \new@ifnextchar[%
7198     {\@Pgl's@{#1}{#2}}%
7199     {\@Pgl's@{#1}{#2}[ ]}%
7200 }
```

\@Pgl's@ Read in the final optional argument:

```
7201 \def\@Pgl's@#1#2[#3]{%
7202   \glsdoifexists{#2}%
7203   {%
7204     \ifglsused{#2}%
7205     {%
7206       \ifgls hasprefix{#2}%
7207       {%
7208         \Glsentryprefix{#2}%
7209         \@gls@{#1}{#2}[#3]%
7210       }%
7211       {\@Gls@{#1}{#2}[#3]}%
7212     }%
7213   }%
7214   \ifgls hasprefixfirst{#2}%
7215   {%
7216     \Glsentryprefixfirst{#2}%
7217     \@gls@{#1}{#2}[#3]%
7218   }%
7219   {\@Gls@{#1}{#2}[#3]}%
7220 }%
7221 }%
7222 }
```

Similarly for the plural version:

\Pgl'spl

```
7223 \newrobustcmd{\Pgl'spl}{\@gls@hyp@opt\@Pgl'spl}
```

\@Pgl'spl Unstarred version.

```
7224 \newcommand*{\@Pgl'spl}[2][ ]{%
7225   \new@ifnextchar[%
7226     {\@Pgl'spl@{#1}{#2}}%
7227     {\@Pgl'spl@{#1}{#2}[ ]}%
7228 }
```

\@Pglsp1@ Read in the final optional argument:

```
7229 \def\@Pglsp1@#1#2[#3]{%
7230   \glsdoifexists{#2}%
7231   {%
7232     \ifglsused{#2}%
7233     {%
7234       \ifglshasprefixplural{#2}%
7235       {%
7236         \Glsentryprefixplural{#2}%
7237         \@glsp1@{#1}{#2}[#3]%
7238       }%
7239       {\@Glspl@{#1}{#2}[#3]}%
7240     }%
7241     {%
7242       \ifglshasprefixfirstplural{#2}%
7243       {%
7244         \Glsentryprefixfirstplural{#2}%
7245         \@glsp1@{#1}{#2}[#3]%
7246       }%
7247       {\@Glspl@{#1}{#2}[#3]}%
7248     }%
7249   }%
7250 }
```

Finally the all upper case versions:

\PGLS

```
7251 \newrobustcmd{\PGLS}{\@gls@hyp@opt\@PGLS}
```

\@PGLS Unstarred version.

```
7252 \newcommand*{\@PGLS}[2][ ]{%
7253   \new@ifnextchar[%
7254   {\@PGLS@{#1}{#2}}%
7255   {\@PGLS@{#1}{#2}[ ]}%
7256 }
```

\@PGLS@ Read in the final optional argument:

```
7257 \def\@PGLS@#1#2[#3]{%
7258   \glsdoifexists{#2}%
7259   {%
7260     \ifglsused{#2}%
7261     {%
7262       \mfirstucMakeUppercase{\glsentryprefix{#2}}%
7263     }%
7264     {%
7265       \mfirstucMakeUppercase{\glsentryprefixfirst{#2}}%
7266     }%
7267     \@GLS@{#1}{#2}[#3]%
7268   }
```

```

7268 }%
7269 }

```

Plural version:

`\PGLSp1`

```

7270 \newrobustcmd{\PGLSp1}{\@gls@hyp@opt\PGLSp1}

```

`\@PGLSp1` Unstarred version.

```

7271 \newcommand*{\@PGLSp1}[2][{}]{%
7272   \new@ifnextchar[%
7273     {\@PGLSp1@{#1}{#2}}%
7274     {\@PGLSp1@{#1}{#2}[]}%
7275 }

```

`\@PGLSp1@` Read in the final optional argument:

```

7276 \def\@PGLSp1@#1#2[#3]{%
7277   \glsdoifexists{#2}%
7278   {%
7279     \ifglsused{#2}%
7280     {%
7281       \mfirstucMakeUppercase{\glsentryprefixplural{#2}}%
7282     }%
7283     {%
7284       \mfirstucMakeUppercase{\glsentryprefixfirstplural{#2}}%
7285     }%
7286     \@GLSp1@{#1}{#2}[#3]%
7287   }%
7288 }

```

3 Mfirstuc Documented Code

```

7289 \NeedsTeXFormat{LaTeX2e}
7290 \ProvidesPackage{mfirstuc}[2015/02/03 v1.10 (NLCT)]

```

Requires etoolbox:

```

7291 \RequirePackage{etoolbox}

```

`\makefirstuc` Syntax:

`\makefirstuc{<text>}`

Makes the first letter uppercase, but will skip initial control sequences if they are followed by a group and make the first thing in the group uppercase, unless the group is empty. Thus `\makefirstuc{abc}` will produce: `Abc`, `\makefirstuc{\ae bc}` will produce: `Æbc`, but `\makefirstuc{\emph{abc}}` will produce `Abc`. This is required by `\Gls` and `\Glspl`.

```

7292 \newif\if@glscs

```

```

7293 \newtoks\@glsmfirst
7294 \newtoks\@glsmrest
7295 \newrobustcmd*{\makefirstuc}[1]{%
7296   \def\@gls@argi{#1}%
7297   \ifx\@gls@argi\@empty
       If the argument is empty, do nothing.
7298   \else

7299     \def\@gls@tmp{\ #1}%
7300     \@onelevel@sanitize\@gls@tmp
7301     \expandafter\@gls@checkcs\@gls@tmp\relax\relax
7302     \if@glscs
7303       \@gls@getbody #1{}\@nil
7304       \ifx\@gls@rest\@empty
7305         \glsmakefirstuc{#1}%
7306       \else
7307         \expandafter\@gls@split\@gls@rest\@nil
7308         \ifx\@gls@first\@empty
7309           \glsmakefirstuc{#1}%
7310         \else
7311           \expandafter\@glsmfirst\expandafter{\@gls@first}%
7312           \expandafter\@glsmrest\expandafter{\@gls@rest}%
7313           \edef\@gls@domfirstuc{\noexpand\@gls@body
7314             {\noexpand\glsmakefirstuc\the\@glsmfirst}%
7315             \the\@glsmrest}%
7316           \@gls@domfirstuc
7317         \fi
7318       \fi
7319     \else
7320       \glsmakefirstuc{#1}%
7321     \fi
7322   \fi
7323 }

       Put first argument in \@gls@first and second argument in \@gls@rest:
7324 \def\@gls@split#1#2\@nil{%
7325   \def\@gls@first{#1}\def\@gls@rest{#2}%
7326 }

7327 \def\@gls@checkcs#1 #2#3\relax{%
7328   \def\@gls@argi{#1}\def\@gls@argii{#2}%
7329   \ifx\@gls@argi\@gls@argii
7330     \@glscstrue
7331   \else
7332     \@glscsfalse
7333   \fi
7334 }

```

\@gls@makefirstuc Make first thing upper case:

```

7335 \def\@gls@makefirstuc#1{\mfirstucMakeUppercase #1}

```

irstucMakeUppercase Allow user to replace \MakeUppercase with another case changing command.

```
7336 \newcommand*{\mfirstucMakeUppercase}{\MakeUppercase}
```

\glsmakefirstuc Provide a user command to make it easier to customise.

```
7337 \newcommand*{\glsmakefirstuc}[1]{\@gls@makefirstuc{#1}}
```

Get the first grouped argument and store in \@gls@body.

```
7338 \def\@gls@getbody#1#\def\@gls@body{#1}\@gls@gobbletonil}
```

Scoup up everything to \@nil and store in \@gls@rest:

```
7339 \def\@gls@gobbletonil#1\@nil{\def\@gls@rest{#1}}
```

\xmakefirstuc Expand argument once before applying \makefirstuc (added v1.01).

```
7340 \newcommand*{\xmakefirstuc}[1]{%
```

```
7341 \expandafter\makefirstuc\expandafter{#1}}
```

\emakefirstuc Fully expand argument before applying \makefirstuc

```
7342 \DeclareRobustCommand*{\emakefirstuc}[1]{%
```

```
7343 \protected@edef\@MFU@caparg{#1}%
```

```
7344 \expandafter\makefirstuc\expandafter{\@MFU@caparg}%
```

```
7345 }
```

\capitalisewords Capitalise each word in the argument. Words are considered to be separated by plain spaces (i.e. non-breakable spaces won't be considered a word break).

```
7346 \newrobustcmd*{\capitalisewords}[1]{%
```

```
7347 \def\gls@add@space{}%
```

```
7348 \let\@mfu@domakefirstuc\makefirstuc
```

```
7349 \let\@mfu@checkword\@gobble
```

```
7350 \mfu@capitalisewords#1 \@nil\mfu@endcap
```

```
7351 }
```

```
7352 \def\mfu@capitalisewords#1 #2\mfu@endcap{%
```

```
7353 \def\mfu@cap@first{#1}%
```

```
7354 \def\mfu@cap@second{#2}%
```

```
7355 \gls@add@space
```

```
7356 \@mfu@checkword{#1}%
```

```
7357 \@mfu@domakefirstuc{#1}%
```

```
7358 \def\gls@add@space{ }%
```

```
7359 \ifx\mfu@cap@second\@nnil
```

```
7360 \let\next\mfu@cap\mfu@noop
```

```
7361 \else
```

```
7362 \let\next\mfu@cap\mfu@capitalisewords
```

```
7363 \let\@mfu@checkword\mfu@checkword
```

```
7364 \fi
```

```
7365 \next\mfu@cap#2\mfu@endcap
```

```
7366 }
```

```
7367 \def\mfu@noop#1\mfu@endcap{}
```

`\mfu@checkword` Check if word should be capitalised.

```
7368 \newcommand*\mfu@checkword[1]{%
7369   \ifinlist{#1}{\@mfu@nocaplist}%
7370   {%
7371     \let\@mfu@domakefirstuc\@firstofone
7372   }%
7373   {%
7374     \let\@mfu@domakefirstuc\makefirstuc
7375   }%
7376 }
```

`\@mfu@nocaplist` List of words that shouldn't be capitalised.

```
7377 \newcommand*\@mfu@nocaplist{}
```

`\MFUnocap` Provide the user with a means to add a word to the list.

```
7378 \newcommand*\MFUnocap[1]{\listadd{\@mfu@nocaplist}{#1}}
```

`\gMFUnocap` Global version.

```
7379 \newcommand*\gMFUnocap[1]{\listgadd{\@mfu@nocaplist}{#1}}
```

`\MFUclear` Clear the list

```
7380 \newcommand*\MFUclear{\renewcommand*\@mfu@nocaplist{}}
```

`\xcapitalisewords` Short-cut command:

```
7381 \newcommand*\xcapitalisewords[1]{%
7382   \expandafter\capitalisewords\expandafter{#1}%
7383 }
```

`\ecapitalisewords` Fully expand argument before applying `\capitalisewords`

```
7384 \DeclareRobustCommand*\ecapitalisewords[1]{%
7385   \protected@edef\@MFU@caparg{#1}%
7386   \expandafter\capitalisewords\expandafter{\@MFU@caparg}%
7387 }
```

4 Mfirstuc-english Documented Code

```
7388 \NeedsTeXFormat{LaTeX2e}
7389 \ProvidesPackage{mfirstuc-english}[2014/07/30 v1.0 (NLCT)]
```

Load mfirstuc if not already loaded:

```
7390 \RequirePackage{mfirstuc}
```

Add no-cap words. (List isn't a complete list.)

```
7391 \MFUnocap{a}
7392 \MFUnocap{an}
7393 \MFUnocap{and}
7394 \MFUnocap{but}
7395 \MFUnocap{for}
```

```

7396 \MFUnocap{in}
7397 \MFUnocap{of}
7398 \MFUnocap{or}
7399 \MFUnocap{no}
7400 \MFUnocap{nor}
7401 \MFUnocap{so}
7402 \MFUnocap{some}
7403 \MFUnocap{the}
7404 \MFUnocap{with}
7405 \MFUnocap{yet}

```

5 Glossary Styles

5.1 Glossary hyper-navigation definitions (glossary-hypernav package)

Package Definition:

```

7406 \ProvidesPackage{glossary-hypernav}[2013/11/14 v4.0 (NLCT)]

```

The commands defined in this package are provided to help navigate around the groups within a glossary (see [subsection 1.15.](#)) `\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`

```

7407 \newcommand*{\glsnavhyperlink}[3][\@glo@type]{%
7408   \edef\gls@grplabel{#2}\protected@edef\gls@grptitle{#3}%
7409   \@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`

```

7410 \newcommand*{\glsnavhypertarget}[3][\@glo@type]{%
  Add this group to the aux file for re-run check.
7411   \protected@write\@auxout{}{\string\@gls@hypergroup{#1}{#2}}%
  Add the target.
7412   \@glstarget{glsn:#1@#2}{#3}%
  Check list of know groups to determine if a re-run is required.

```



```

7413 \expandafter\let
7414 \expandafter\@gls@list\csname @gls@hypergroup\list@#1\endcsname
    Iterate through list and terminate loop if this group is found.
7415 \@for\@gls@elem:=\@gls@list\do{%
7416 \ifthenelse{\equal{\@gls@elem}{#2}}{\@endfortrue}{}}%
    Check if list terminated prematurely.
7417 \if@endfor
7418 \else
    This group was not included in the list, so issue a warning.
7419 \GlossariesWarningNoLine{Navigation panel
7420 for glossary type '#1'~Jmissing group '#2'}%
7421 \gdef\gls@hypergroup\prerun{%
7422 \GlossariesWarningNoLine{Navigation panel
7423 has changed. Rerun LaTeX}}%
7424 \fi
7425 }

```

`\gls@hypergroup\prerun` Give a warning at the end if re-run required

```

7426 \let\gls@hypergroup\relax
7427 \AtEndDocument{\gls@hypergroup\prerun}

```

`\@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.

```

7428 \newcommand*{\@gls@hypergroup}[2]{%
7429 \@ifundefined{\@gls@hypergroup\list@#1}{%
7430 \expandafter\xdef\csname @gls@hypergroup\list@#1\endcsname{#2}%
7431 }{%
7432 \expandafter\let\expandafter\@gls@tmp
7433 \csname @gls@hypergroup\list@#1\endcsname
7434 \expandafter\xdef\csname @gls@hypergroup\list@#1\endcsname{%
7435 \@gls@tmp,#2}%
7436 }%
7437 }

```

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`

```

7438 \newcommand*{\glsnavigation}{%
7439 \def\@gls@between{}%

```

```

7440 \ifundefined{@gls@hypergrouplist@{@glo@type}}{%
7441   \def@gls@list{%
7442 }{%
7443   \expandafter\let\expandafter@gls@list
7444     \csname @gls@hypergrouplist@{@glo@type}\endcsname
7445 }%
7446 \@for@gls@tmp:=\@gls@list\do{%
7447   \gls@between

7448   \@gls@getgrouptitle{\@gls@tmp}{\@gls@grptitle}%
7449   \glsnavhyperlink{\@gls@tmp}{\@gls@grptitle}%
7450   \let@gls@between\glshypernavsep%
7451 }%
7452 }

```

`\glshypernavsep` Separator for the hyper navigation bar.

```

7453 \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`

```

7454 \newcommand*{\glssymbolnav}{%
7455 \glsnavhyperlink{glssymbols}{\glsgetgrouptitle{glssymbols}}%
7456 \glshypernavsep
7457 \glsnavhyperlink{glsnumbers}{\glsgetgrouptitle{glsnumbers}}%
7458 \glshypernavsep
7459 }

```

5.2 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.

```

7460 \ProvidesPackage{glossary-inline}[2013/11/14 v4.0 (NLCT)]

```

`inline` Define the inline style.

```

7461 \newglossarystyle{inline}{%

```

Start of glossary sets up first empty separator between entries. (This is then changed by `\glossentry`)

```

7462   \renewenvironment{theglossary}%
7463     {%
7464       \def@gls@inlinesep{%
7465       \def@gls@inlinesubsep{%
7466       \def@gls@inlinepostchild{%
7467     }%
7468     {\glspostinline}%

```

No header:

```
7469 \renewcommand*{\glossaryheader}{}%
```

No group headings (if heading is required, add `\glsinlinedopostchild` to start definition in case heading follows a child entry):

```
7470 \renewcommand*{\glsgroupheading}[1]{}%
```

Just display separator followed by name and description:

```
7471 \renewcommand{\glossentry}[2]{%
7472   \glsinlinedopostchild
7473   \gls@inlinesep
7474   \glsentryitem{##1}%
7475   \glsinlinenameformat{##1}{%
7476     \glossentryname{##1}%
7477   }%
7478   \ifglshasdescsuppressed{##1}%
7479   {%
7480     \glsinlineemptydescformat
7481     {%
7482       \glossentrysymbol{##1}%
7483     }%
7484     {%
7485       ##2%
7486     }%
7487   }%
7488   {%
7489     \ifglshasdesc{##1}%
7490     {\glsinlinedescformat{\glossentrydesc{##1}}{\glossentrysymbol{##1}}{##2}}%
7491     {\glsinlineemptydescformat{\glossentrysymbol{##1}}{##2}}%
7492   }%
7493   \ifglshaschildren{##1}%
7494   {%
7495     \glsresetsubentrycounter
7496     \glsinlineparentchildseparator
7497     \def\gls@inlinesubsep{}%
7498     \def\gls@inlinepostchild{\glsinlinepostchild}%
7499   }%
7500   {%
7501     \def\gls@inlinesep{\glsinlineseparator}%
7502   }%
```

Sub-entries display description:

```
7503 \renewcommand{\subglossentry}[3]{%
7504   \gls@inlinesubsep%
7505   \glsinlinesubnameformat{##2}{%
7506     \glossentryname{##2}%
7507   }%
7508   \glssubentryitem{##2}%
7509   \glsinlinesubdescformat{\glossentrydesc{##2}}{\glossentrysymbol{##2}}{##3}%
7510   \def\gls@inlinesubsep{\glsinlinesubseparator}%
7511 }
```

Nothing special between groups:

```
7511 \renewcommand*{\glsgroupskip}{}%
7512 }
```

`\glsinlinedopostchild`

```
7513 \newcommand*{\glsinlinedopostchild}{%
7514     \gls@inlinepostchild
7515     \def\gls@inlinepostchild{}%
7516 }
```

`\glsinlineseparator` Separator to use between entries.

```
7517 \newcommand*{\glsinlineseparator}{;\space}
```

`\glsinlinesubseparator` Separator to use between sub-entries.

```
7518 \newcommand*{\glsinlinesubseparator}{,\space}
```

`\glsinlineparentchildseparator` Separator to use between parent and children.

```
7519 \newcommand*{\glsinlineparentchildseparator}{:\space}
```

`\glsinlinepostchild` Hook to use between child and next entry

```
7520 \newcommand*{\glsinlinepostchild}{}%
```

`\glspostinline` Terminator for inline glossary.

```
7521 \newcommand*{\glspostinline}{\glspostdescription\space}
```

`\glsinlinenameformat` Formats the name of the entry (first argument label, second argument name):

```
7522 \newcommand*{\glsinlinenameformat}[2]{\glstarget{#1}{#2}}
```

`\glsinlinedescformat` Formats the entry's description, symbol and location list:

```
7523 \newcommand*{\glsinlinedescformat}[3]{\space#1}
```

`\glsinlineemptydescformat` Formats the entry's symbol and location list when the description is empty:

```
7524 \newcommand*{\glsinlineemptydescformat}[2]{}%
```

`\glsinlinesubnameformat` Formats the name of the subentry (first argument label, second argument name):

```
7525 \newcommand*{\glsinlinesubnameformat}[2]{\glstarget{#1}{}}
```

`\glsinlinesubdescformat` Formats the subentry's description, symbol and location list:

```
7526 \newcommand*{\glsinlinesubdescformat}[3]{#1}
```

5.3 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.

```
7527 \ProvidesPackage{glossary-list}[2015/02/03 v4.13 (NLCT)]
```

`\indexspace` The are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```
7528 \providecommand{\indexspace}{%
7529   \par \vskip 10\p@ \p@ \@plus 5\p@ \@minus 3\p@ \relax
7530 }
```

`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.

```
7531 \newglossarystyle{list}{%
```

Use description environment:

```
7532   \renewenvironment{theglossary}%
7533     {\begin{description}}{\end{description}}%
```

No header at the start of the environment:

```
7534   \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
7535   \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries start a new item in the list:

```
7536   \renewcommand*{\glossentry}[2]{%
7537     \item[\glssentryitem{##1}%
7538       \glstarget{##1}{\glossentryname{##1}}]
7539     \glossentrydesc{##1}\glspostdescription\space ##2}%
```

Sub-entries continue on the same line:

```
7540   \renewcommand*{\subglossentry}[3]{%
7541     \glssubentryitem{##2}%
7542     \glstarget{##2}{\strut}%
7543     \glossentrydesc{##2}\glspostdescription\space ##3.}%
7544 %   \end{macrocode}
```

7545 % Add vertical space between groups:

```
7546 %\changes{3.03}{2012/09/21}{added check for glsnogroupskip}
```

```
7547 %   \begin{macrocode}
```

```
7548   \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
```

```
7549 }
```

`listgroup` The listgroup style is like the list style, but the glossary groups have headings.

```
7550 \newglossarystyle{listgroup}{%
```

Base it on the list style:

```
7551 \setglossarystyle{list}%
```

Each group has a heading:

```
7552 \renewcommand*{\glsgroupheading}[1]{\item[\glsgetgrouptitle{##1}]}
```

listhypergroup The listhypergroup style is like the listgroup style, but has a set of links to the groups at the start of the glossary.

```
7553 \newglossarystyle{listhypergroup}{%
```

Base it on the list style:

```
7554 \setglossarystyle{list}%
```

Add navigation links at the start of the environment:

```
7555 \renewcommand*{\glossaryheader}{%
```

```
7556 \item[\glsnavigation]]%
```

Each group has a heading with a hypertarget:

```
7557 \renewcommand*{\glsgroupheading}[1]{%
```

```
7558 \item[\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}]}
```

altlist The altlist glossary style is like the list style, but places the description on a new line. Sub-entries follow in separate paragraphs without the sub-entry name. This style does not use the entry's symbol.

```
7559 \newglossarystyle{altlist}{%
```

Base it on the list style:

```
7560 \setglossarystyle{list}%
```

Main (level 0) entries start a new item in the list with a line break after the entry name:

```
7561 \renewcommand*{\glossentry}[2]{%
```

```
7562 \item[\glsentryitem{##1}%
```

```
7563 \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.

```
7564 \mbox{} \par \nobreak \@afterheading
```

```
7565 \glossentrydesc{##1} \glspostdescription \space ##2}%
```

Sub-entries start a new paragraph:

```
7566 \renewcommand{\subglossentry}[3]{%
```

```
7567 \par
```

```
7568 \glssubentryitem{##2}%
```

```
7569 \glstarget{##2}{\strut} \glossentrydesc{##2} \glspostdescription \space ##3}%
```

```
7570 }
```

altlistgroup The altlistgroup glossary style is like the altlist style, but the glossary groups have headings.

```
7571 \newglossarystyle{altlistgroup}{%
```

Base it on the altlist style:

```
7572 \setglossarystyle{altlist}%
```

Each group has a heading:

```
7573 \renewcommand*{\glsgroupheading}[1]{\item[\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.

```
7574 \newglossarystyle{altlisthypergroup}{%
```

Base it on the altlist style:

```
7575 \setglossarystyle{altlist}%
```

Add navigation links at the start of the environment:

```
7576 \renewcommand*{\glossaryheader}{%
```

```
7577 \item[\glsnavigation]}%
```

Each group has a heading with a hypertarget:

```
7578 \renewcommand*{\glsgroupheading}[1]{%
```

```
7579 \item[\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.

```
7580 \newglossarystyle{listdotted}{%
```

Base it on the list style:

```
7581 \setglossarystyle{list}%
```

Each main (level 0) entry starts a new item:

```
7582 \renewcommand*{\glossentry}[2]{%
```

```
7583 \item[]\makebox[\glslistdottedwidth][l]{%
```

```
7584 \glsentryitem{##1}%
```

```
7585 \glstarget{##1}{\glossentryname{##1}}%
```

```
7586 \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##1}}%
```

Sub entries have the same format as main entries:

```
7587 \renewcommand*{\subglossentry}[3]{%
```

```
7588 \item[]\makebox[\glslistdottedwidth][l]{%
```

```
7589 \glssubentryitem{##2}%
```

```
7590 \glstarget{##2}{\glossentryname{##2}}%
```

```
7591 \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##2}}%
```

```
7592 }
```

`\glslistdottedwidth`

```
7593 \newlength\glslistdottedwidth
```

```
7594 \setlength{\glslistdottedwidth}{.5\hsize}
```

`sublistdotted` This style is similar to the `glostylelistdotted` style, except that the main entries just have the name displayed.

```

7595 \newglossarystyle{sublistdotted}{%
    Base it on the listdotted style:
7596   \setglossarystyle{listdotted}%
    Main (level 0) entries just display the name:
7597   \renewcommand*{\glossentry}[2]{%
7598     \item[\glentryitem{##1}\glstarget{##1}{\glossentryname{##1}}]}%
7599 }
```

5.4 Glossary Styles using longtable (the glossary-long package)

The glossary styles defined in the package used the `longtable` environment in the glossary.

```

7600 \ProvidesPackage{glossary-long}[2013/11/14 v4.0 (NLCT)]
```

Requires the package:

```

7601 \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`.)

```

7602 \@ifundefined{glsdescwidth}{%
7603   \newlength{glsdescwidth}
7604   \setlength{glsdescwidth}{0.6\hsize}
7605 }{}
```

`\glspagelistwidth` This is a length that governs the width of the page list column.

```

7606 \@ifundefined{glspagelistwidth}{%
7607   \newlength{glspagelistwidth}
7608   \setlength{glspagelistwidth}{0.1\hsize}
7609 }{}
```

`long` The long glossary style command which uses the `longtable` environment:

```

7610 \newglossarystyle{long}{%
    Use longtable with two columns:
7611   \renewenvironment{theglossary}%
7612     {\begin{longtable}\lp{glsdescwidth}}%
7613     {\end{longtable}}%
    Do nothing at the start of the environment:
7614   \renewcommand*{\glossaryheader}{}%
    No heading between groups:
7615   \renewcommand*{\glsgroupheading}[1]{}
```


Main (level 0) entries displayed in a row:

```
7616 \renewcommand{\glossentry}[2]{%
7617   \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
7618   \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline
7619 }%
```

Sub entries displayed on the following row without the name:

```
7620 \renewcommand{\subglossentry}[3]{%
7621   &
7622   \glssubentryitem{##2}%
7623   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
7624   ##3\tabularnewline
7625 }%
```

Blank row between groups:

```
7626 \renewcommand*{\glsgroupskip}{\ifglsgnোগroupskip\else &
7627 \tabularnewline\fi}%
7628 }
```

longborder The longborder style is like the above, but with horizontal and vertical lines:

```
7629 \newglossarystyle{longborder}{%
```

Base it on the glostylelong style:

```
7630 \setglossarystyle{long}%
```

Use longtable with two columns with vertical lines between each column:

```
7631 \renewenvironment{theglossary}{%
7632   \begin{longtable}{|l|p{\glsgdescwidth}|}{\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
7633 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
7634 }
```

longheader The longheader style is like the long style but with a header:

```
7635 \newglossarystyle{longheader}{%
```

Base it on the glostylelong style:

```
7636 \setglossarystyle{long}%
```

Set the table's header:

```
7637 \renewcommand*{\glossaryheader}{%
7638   \bfseries \entryname & \bfseries \descriptionname\tabularnewline\endhead}%
7639 }
```

longheaderborder The longheaderborder style is like the long style but with a header and border:

```
7640 \newglossarystyle{longheaderborder}{%
```

Base it on the glostylelongborder style:

```
7641 \setglossarystyle{longborder}%
```

Set the table's header and add horizontal line to table's foot:

```

7642 \renewcommand*{\glossaryheader}{%
7643 \hline\bfseries \entryname & \bfseries
7644 \descriptionname\tabularnewline\hline
7645 \endhead
7646 \hline\endfoot}%
7647 }

```

long3col The long3col style is like long but with 3 columns

```

7648 \newglossarystyle{long3col}{%

```

Use a longtable with 3 columns:

```

7649 \renewenvironment{theglossary}%
7650 {\begin{longtable}{lp{\glsgdescwidth}p{\glspagelistwidth}}}%
7651 {\end{longtable}}%

```

No table header:

```

7652 \renewcommand*{\glossaryheader}{}%

```

No headings between groups:

```

7653 \renewcommand*{\glsgroupheading}[1]{}%

```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```

7654 \renewcommand{\glossentry}[2]{%
7655 \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
7656 \glossentrydesc{##1} & ##2\tabularnewline
7657 }%

```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```

7658 \renewcommand{\subglossentry}[3]{%
7659 &
7660 \glssubentryitem{##2}%
7661 \glstarget{##2}{\strut}\glossentrydesc{##2} &
7662 ##3\tabularnewline
7663 }%

```

Blank row between groups:

```

7664 \renewcommand*{\glsgroupskip}{%
7665 \ifglsgnোগroupskip\else & &\tabularnewline\fi}%
7666 }

```

long3colborder The long3colborder style is like the long3col style but with a border:

```

7667 \newglossarystyle{long3colborder}{%

```

Base it on the glostylelong3col style:

```

7668 \setglossarystyle{long3col}%

```

Use a longtable with 3 columns with vertical lines around them:

```

7669 \renewenvironment{theglossary}%
7670 {\begin{longtable}{|l|p{\glsgdescwidth}|p{\glspagelistwidth}|}%
7671 {\end{longtable}}%

```

Place horizontal lines at the head and foot of the table:

```
7672 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
7673 }
```

`long3colheader` The `long3colheader` style is like `long3col` but with a header row:

```
7674 \newglossarystyle{long3colheader}{%
```

Base it on the `glostylelong3col` style:

```
7675 \setglossarystyle{long3col}%
```

Set the table's header:

```
7676 \renewcommand*{\glossaryheader}{%
7677 \bfseries\entryname&\bfseries\descriptionname&
7678 \bfseries\pagelistname\tabularnewline\endhead}%
7679 }
```

`long3colheaderborder` The `long3colheaderborder` style is like the above but with a border

```
7680 \newglossarystyle{long3colheaderborder}{%
```

Base it on the `glostylelong3colborder` style:

```
7681 \setglossarystyle{long3colborder}%
```

Set the table's header and add horizontal line at table's foot:

```
7682 \renewcommand*{\glossaryheader}{%
7683 \hline
7684 \bfseries\entryname&\bfseries\descriptionname&
7685 \bfseries\pagelistname\tabularnewline\hline\endhead
7686 \hline\endfoot}%
7687 }
```

`long4col` The `long4col` style has four columns where the third column contains the value of the associated symbol key.

```
7688 \newglossarystyle{long4col}{%
```

Use a `longtable` with 4 columns:

```
7689 \renewenvironment{theglossary}%
7690 {\begin{longtable}{llll}}%
7691 {\end{longtable}}%
```

No table header:

```
7692 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
7693 \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):

```
7694 \renewcommand{\glossentry}[2]{%
7695 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
7696 \glossentrydesc{##1} &
7697 \glossentrysymbol{##1} &
7698 ##2\tabularnewline
7699 }%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```

7700 \renewcommand{\subglossentry}[3]{%
7701     &
7702     \glssubentryitem{##2}%
7703     \glstarget{##2}{\strut}\glossentrydesc{##2} &
7704     \glossentrysymbol{##2} & ##3\tabularnewline
7705 }%
```

Blank row between groups:

```

7706 \renewcommand*{\glsgroupskip}{%
7707     \ifglsgnogroupskip\else & & &\tabularnewline\fi}%
7708 }
```

long4colheader The long4colheader style is like long4col but with a header row.

```

7709 \newglossarystyle{long4colheader}{%
```

Base it on the glostylelong4col style:

```

7710 \setglossarystyle{long4col}%
```

Table has a header:

```

7711 \renewcommand*{\glossaryheader}{%
7712     \bfseries\entryname&\bfseries\descriptionname&
7713     \bfseries \symbolname&
7714     \bfseries\pagelistname\tabularnewline\endhead}%
7715 }
```

long4colborder The long4colborder style is like long4col but with a border.

```

7716 \newglossarystyle{long4colborder}{%
```

Base it on the glostylelong4col style:

```

7717 \setglossarystyle{long4col}%
```

Use a longtable with 4 columns surrounded by vertical lines:

```

7718 \renewenvironment{theglossary}%
7719     {\begin{longtable}{|l|l|l|l|}}%
7720     {\end{longtable}}%
```

Add horizontal lines to the head and foot of the table:

```

7721 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
7722 }
```

long4colheaderborder The long4colheaderborder style is like the above but with a border.

```

7723 \newglossarystyle{long4colheaderborder}{%
```

Base it on the glostylelong4col style:

```

7724 \setglossarystyle{long4col}%
```

Use a longtable with 4 columns surrounded by vertical lines:

```

7725 \renewenvironment{theglossary}%
7726     {\begin{longtable}{|l|l|l|l|}}%
7727     {\end{longtable}}%
```

Add table header and horizontal line at the table's foot:

```

7728 \renewcommand*{\glossaryheader}{%
7729 \hline\bfseries\entryname&\bfseries\descriptionname&
7730 \bfseries \symbolname&
7731 \bfseries\pagelistname\tabularnewline\hline\endhead
7732 \hline\endfoot}%
7733 }
```

altlong4col The altlong4col style is like the long4col style but can have multiline descriptions and page lists.

```
7734 \newglossarystyle{altlong4col}{%
```

Base it on the glostylelong4col style:

```
7735 \setglossarystyle{long4col}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```

7736 \renewenvironment{theglossary}%
7737 {\begin{longtable}{lp{\glsdescwidth}lp{\glspagelistwidth}}}%
7738 {\end{longtable}}%
7739 }
```

altlong4colheader The altlong4colheader style is like altlong4col but with a header row.

```
7740 \newglossarystyle{altlong4colheader}{%
```

Base it on the glostylelong4colheader style:

```
7741 \setglossarystyle{long4colheader}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```

7742 \renewenvironment{theglossary}%
7743 {\begin{longtable}{lp{\glsdescwidth}lp{\glspagelistwidth}}}%
7744 {\end{longtable}}%
7745 }
```

altlong4colborder The altlong4colborder style is like altlong4col but with a border.

```
7746 \newglossarystyle{altlong4colborder}{%
```

Base it on the glostylelong4colborder style:

```
7747 \setglossarystyle{long4colborder}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```

7748 \renewenvironment{theglossary}%
7749 {\begin{longtable}{|lp{\glsdescwidth}|lp{\glspagelistwidth}|}}%
7750 {\end{longtable}}%
7751 }
```

altlong4colheaderborder The altlong4colheaderborder style is like the above but with a header as well as a border.

```
7752 \newglossarystyle{altlong4colheaderborder}{%
```

Base it on the `glostylelong4colheaderborder` style:

```
7753 \setglossarystyle{long4colheaderborder}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
7754 \renewenvironment{theglossary}%  
7755   {\begin{longtable}{|l|p{\glsdescwidth}|l|p{\glspagelistwidth}|}%  
7756   {\end{longtable}}%  
7757 }
```

5.5 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.

```
7758 \ProvidesPackage{glossary-longragged}[2014/07/30 v4.08 (NLCT)]
```

Requires the package:

```
7759 \RequirePackage{array}
```

Requires the package:

```
7760 \RequirePackage{longtable}
```

`\glsdescwidth` This is a length that governs the width of the description column. This may have already been defined.

```
7761 \@ifundefined{glsdescwidth}{%  
7762   \newlength\glsdescwidth  
7763   \setlength{\glsdescwidth}{0.6\hsize}  
7764 }{}
```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```
7765 \@ifundefined{glspagelistwidth}{%  
7766   \newlength\glspagelistwidth  
7767   \setlength{\glspagelistwidth}{0.1\hsize}  
7768 }{}
```

`longragged` The `longragged` glossary style is like the `long` but uses ragged right formatting for the description column.

```
7769 \newglossarystyle{longragged}{%
```

Use `longtable` with two columns:

```
7770 \renewenvironment{theglossary}%  
7771   {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}}}%  
7772   {\end{longtable}}%
```

Do nothing at the start of the environment:

```
7773 \renewcommand*{\glossaryheader}{}%
```

No heading between groups:

```
7774 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries displayed in a row:

```
7775 \renewcommand{\glossentry}[2]{%
7776   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
7777   \glossentrydesc{##1}\glspostdescription\space ##2%
7778   \tabularnewline
7779 }%
```

Sub entries displayed on the following row without the name:

```
7780 \renewcommand{\subglossentry}[3]{%
7781   &
7782   \glssubentryitem{##2}%
7783   \glstarget{##2}{\strut}\glossentrydesc{##2}%
7784   \glspostdescription\space ##3%
7785   \tabularnewline
7786 }%
```

Blank row between groups:

```
7787 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else & \tabularnewline\fi}%
7788 }
```

longraggedborder The longraggedborder style is like the above, but with horizontal and vertical lines:

```
7789 \newglossarystyle{longraggedborder}{%
```

Base it on the glostylelongragged style:

```
7790 \setglossarystyle{longragged}%
```

Use longtable with two columns with vertical lines between each column:

```
7791 \renewenvironment{theglossary}{%
7792   \begin{longtable}{|l|>{\raggedright}p{\glsdescwidth}|}%
7793   {\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
7794 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
7795 }
```

longraggedheader The longraggedheader style is like the longragged style but with a header:

```
7796 \newglossarystyle{longraggedheader}{%
```

Base it on the glostylelongragged style:

```
7797 \setglossarystyle{longragged}%
```

Set the table's header:

```
7798 \renewcommand*{\glossaryheader}{%
7799   \bfseries \entryname & \bfseries \descriptionname
7800   \tabularnewline\endhead}%
7801 }
```

raggedheaderborder The longraggedheaderborder style is like the longragged style but with a header and border:

```
7802 \newglossarystyle{longraggedheaderborder}{%
```

Base it on the glostylelongraggedborder style:

```
7803 \setglossarystyle{longraggedborder}{%
```

Set the table's header and add horizontal line to table's foot:

```
7804 \renewcommand*{\glossaryheader}{%
7805 \hline\bfseries \entryname & \bfseries \descriptionname
7806 \tabularnewline\hline
7807 \endhead
7808 \hline\endfoot}%
7809 }
```

longragged3col The longragged3col style is like longragged but with 3 columns

```
7810 \newglossarystyle{longragged3col}{%
```

Use a longtable with 3 columns:

```
7811 \renewenvironment{theglossary}{%
7812 {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}%
7813 >{\raggedright}p{\glspagelistwidth}}}%
7814 {\end{longtable}}%
```

No table header:

```
7815 \renewcommand*{\glossaryheader}{}%
```

No headings between groups:

```
7816 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
7817 \renewcommand{\glossentry}[2]{%
7818 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
7819 \glossentrydesc{##1} & ##2\tabularnewline
7820 }%
```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```
7821 \renewcommand{\subglossentry}[3]{%
7822 &
7823 \glssubentryitem{##2}%
7824 \glstarget{##2}{\strut}\glossentrydesc{##2} &
7825 ##3\tabularnewline
7826 }%
```

Blank row between groups:

```
7827 \renewcommand*{\glsgroupskip}{%
7828 \ifglsnogroupskip\else & &\tabularnewline\fi}%
7829 }
```


`longragged3colborder` The `longragged3colborder` style is like the `longragged3col` style but with a border:

```
7830 \newglossarystyle{longragged3colborder}{%
```

Base it on the `glostylelongragged3col` style:

```
7831 \setglossarystyle{longragged3col}{%
```

Use a `longtable` with 3 columns with vertical lines around them:

```
7832 \renewenvironment{theglossary}{%
```

```
7833 {\begin{longtable}{|l|>{\raggedright}p{\glsdescwidth}}|}%
```

```
7834 >{\raggedright}p{\glspagelistwidth}}|}%
```

```
7835 {\end{longtable}}}%
```

Place horizontal lines at the head and foot of the table:

```
7836 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
```

```
7837 }
```

`longragged3colheader` The `longragged3colheader` style is like `longragged3col` but with a header row:

```
7838 \newglossarystyle{longragged3colheader}{%
```

Base it on the `glostylelongragged3col` style:

```
7839 \setglossarystyle{longragged3col}{%
```

Set the table's header:

```
7840 \renewcommand*{\glossaryheader}{%
```

```
7841 \bfseries\entryname&\bfseries\descriptionname&
```

```
7842 \bfseries\pagelistname\tabularnewline\endhead}%
```

```
7843 }
```

`longragged3colheaderborder` The `longragged3colheaderborder` style is like the above but with a border

```
7844 \newglossarystyle{longragged3colheaderborder}{%
```

Base it on the `glostylelongragged3colborder` style:

```
7845 \setglossarystyle{longragged3colborder}{%
```

Set the table's header and add horizontal line at table's foot:

```
7846 \renewcommand*{\glossaryheader}{%
```

```
7847 \hline
```

```
7848 \bfseries\entryname&\bfseries\descriptionname&
```

```
7849 \bfseries\pagelistname\tabularnewline\hline\endhead
```

```
7850 \hline\endfoot}%
```

```
7851 }
```

`altlongragged4col` The `altlongragged4col` style is like the `altlong4col` style defined in the package, except that ragged right formatting is used for the description and page list columns.

```
7852 \newglossarystyle{altlongragged4col}{%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
7853 \renewenvironment{theglossary}{%
```

```

7854    {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}l%
7855        >{\raggedright}p{\glspagelistwidth}}}%
7856    {\end{longtable}}}%

```

No table header:

```

7857    \renewcommand*{\glossaryheader}{}%

```

No group headings:

```

7858    \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):

```

7859    \renewcommand{\glossentry}[2]{%
7860        \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
7861        \glossentrydesc{##1} & \glossentrysymbol{##1} &
7862        ##2\tabularnewline
7863    }%

```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```

7864    \renewcommand{\subglossentry}[3]{%
7865        &
7866        \glssubentryitem{##2}%
7867        \glstarget{##2}{\strut}\glossentrydesc{##2} &
7868        \glossentrysymbol{##2} & ##3\tabularnewline
7869    }%

```

Blank row between groups:

```

7870    \renewcommand*{\glsgroupskip}{%
7871        \ifglsgroupskip\else & & \tabularnewline\fi}%
7872 }

```

`ongragged4colheader` The `altlongragged4colheader` style is like `altlongragged4col` but with a header row.

```

7873 \newglossarystyle{altlongragged4colheader}{%

```

Base it on the `glostylealtlongragged4col` style:

```

7874    \setglossarystyle{altlongragged4col}%

```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```

7875    \renewenvironment{theglossary}%
7876        {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}l%
7877            >{\raggedright}p{\glspagelistwidth}}}%
7878        {\end{longtable}}%

```

Table has a header:

```

7879    \renewcommand*{\glossaryheader}{%
7880        \bfseries\entryname&\bfseries\descriptionname&
7881        \bfseries \symbolname&
7882        \bfseries\pagelistname\tabularnewline\endhead}%
7883 }

```

`longragged4colborder` The `altlongragged4colborder` style is like `altlongragged4col` but with a border.

```

7884 \newglossarystyle{altlongragged4colborder}{%
    Base it on the glostylealtlongragged4col style:
7885   \setglossarystyle{altlongragged4col}%

    Use a longtable with 4 columns where the second and last columns may have
    multiple lines in each row:
7886   \renewenvironment{theglossary}%
7887     {\begin{longtable}{|l|>{\raggedright}p{\glsdescwidth}|l|}%
7888      >{\raggedright}p{\glsPagelistwidth}|}%
7889     {\end{longtable}}%

    Add horizontal lines to the head and foot of the table:
7890   \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
7891 }
```

`longragged4colheaderborder` The `altlongragged4colheaderborder` style is like the above but with a header as well as a border.

```

7892 \newglossarystyle{altlongragged4colheaderborder}{%
    Base it on the glostylealtlongragged4col style:
7893   \setglossarystyle{altlongragged4col}%

    Use a longtable with 4 columns where the second and last columns may have
    multiple lines in each row:
7894   \renewenvironment{theglossary}%
7895     {\begin{longtable}{|l|>{\raggedright}p{\glsdescwidth}|l|}%
7896      >{\raggedright}p{\glsPagelistwidth}|}%
7897     {\end{longtable}}%

    Add table header and horizontal line at the table's foot:
7898   \renewcommand*{\glossaryheader}{%
7899     \hline\bfseries\entryname&\bfseries\descriptionname&
7900     \bfseries \symbolname&
7901     \bfseries\pagelistname\tabularnewline\hline\endhead
7902     \hline\endfoot}%
7903 }
```

5.6 Glossary Styles using multicol (glossary-mcols.sty)

The style file defines glossary styles that use the `multicol` package. These use the tree-like glossary styles in a `multicol` environment.

```

7904 \ProvidesPackage{glossary-mcols}[2015/02/03 v4.13 (NLCT)]
```

Required packages:

```

7905 \RequirePackage{multicol}
7906 \RequirePackage{glossary-tree}
```

`\indexspace` The are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```
7907 \providecommand{\indexspace}{%
7908   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
7909 }
```

`\glsmcols` Define macro in which to store the number of columns. (Defaults to 2.)

```
7910 \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.)

```
7911 \newglossarystyle{mcolindex}{%
7912   \setglossarystyle{index}%
7913   \renewenvironment{theglossary}%
7914     {%
7915       \begin{multicols}{\glsmcols}
7916       \setlength{\parindent}{0pt}%
7917       \setlength{\parskip}{0pt plus 0.3pt}%
7918       \let\item\@idxitem%
7919       {\end{multicols}}%
7920 }
```

`mcolindexgroup` As `mcolindex` but has headings:

```
7921 \newglossarystyle{mcolindexgroup}{%
7922   \setglossarystyle{mcolindex}%
7923   \renewcommand*\glsgroupheading[1]{%
7924     \item\textbf{\glsgroupheading{##1}}\indexspace}%
7925 }
```

`mcolindexhypergroup` The `mcolindexhypergroup` style is like the `mcolindexgroup` style but has hyper navigation.

```
7926 \newglossarystyle{mcolindexhypergroup}{%
```

Base it on the `glostylemcolindex` style:

```
7927   \setglossarystyle{mcolindex}%
```

Put navigation links to the groups at the start of the glossary:

```
7928   \renewcommand*\glossaryheader{%
7929     \item\textbf{\glsnavigation}\indexspace}%
```

Add a heading for each group (with a target). The group's title is in bold followed by a vertical gap.

```
7930   \renewcommand*\glsgroupheading[1]{%
7931     \item\textbf{\glsnavhypertarget{##1}}{\glsgroupheading{##1}}}%
7932   \indexspace}%
7933 }
```

mcoltree Multi-column index style. Same as the tree, but puts the glossary in multiple columns.

```

7934 \newglossarystyle{mcoltree}{%
7935   \setglossarystyle{tree}%
7936   \renewenvironment{theglossary}%
7937   {%
7938     \begin{multicols}{\glsmcols}
7939     \setlength{\parindent}{0pt}%
7940     \setlength{\parskip}{0pt plus 0.3pt}%
7941   }%
7942   {\end{multicols}}%
7943 }
```

mcoltreegroup Like the mcoltree style but the glossary groups have headings.

```

7944 \newglossarystyle{mcoltreegroup}{%
  Base it on the glostylemcoltree style:
7945   \setglossarystyle{mcoltree}%
  Each group has a heading (in bold) followed by a vertical gap):
7946   \renewcommand{\glsgroupheading}[1]{\par
7947     \noindent\textbf{\glsgrouptitle{##1}}\par\indexspace}%
7948 }
```

mcoltreehypergroup The mcoltreehypergroup style is like the treegroup style, but has a set of links to the groups at the start of the glossary.

```

7949 \newglossarystyle{mcoltreehypergroup}{%
  Base it on the glostylemcoltree style:
7950   \setglossarystyle{mcoltree}%
  Put navigation links to the groups at the start of the theglossary environment:
7951   \renewcommand*\glossaryheader{%
7952     \par\noindent\textbf{\glsnavigation}\par\indexspace}%
  Each group has a heading (in bold with a target) followed by a vertical gap):
7953   \renewcommand*\glsgroupheading[1]{%
7954     \par\noindent
7955     \textbf{\glsnavigationhypertarget{##1}{\glsgrouptitle{##1}}}\par
7956     \indexspace}%
7957 }
```

mcoltreename Multi-column index style. Same as the treename, but puts the glossary in multiple columns.

```

7958 \newglossarystyle{mcoltreename}{%
7959   \setglossarystyle{treename}%
7960   \renewenvironment{theglossary}%
7961   {%
```

```

7962     \begin{multicols}{\glsmcols}
7963     \setlength{\parindent}{0pt}%
7964     \setlength{\parskip}{0pt plus 0.3pt}%
7965 }%
7966 {\end{multicols}}}%
7967 }

```

mcoltreenonamegroup Like the mcoltreenoname style but the glossary groups have headings.

```

7968 \newglossarystyle{mcoltreenonamegroup}{%
    Base it on the glostylemcoltreenoname style:
7969   \setglossarystyle{mcoltreenoname}%
    Give each group a heading:
7970   \renewcommand{\glsgroupheading}[1]{\par
7971     \noindent\textbf{\glsgrouptitle{##1}}\par\indexspace}%
7972 }

```

treenonamehypergroup The mcoltreenonamehypergroup style is like the mcoltreenonamegroup style, but has a set of links to the groups at the start of the glossary.

```

7973 \newglossarystyle{mcoltreenonamehypergroup}{%
    Base it on the glostylemcoltreenoname style:
7974   \setglossarystyle{mcoltreenoname}%
    Put navigation links to the groups at the start of the theglossary environment:
7975   \renewcommand*{\glossaryheader}{%
7976     \par\noindent\textbf{\glsnavigation}\par\indexspace}%
    Each group has a heading (in bold with a target) followed by a vertical gap):
7977   \renewcommand*{\glsgroupheading}[1]{%
7978     \par\noindent
7979     \textbf{\glsnavigationhypertarget{##1}{\glsgrouptitle{##1}}}\par
7980     \indexspace}%
7981 }

```

mcolalmtree Multi-column index style. Same as the almtree, but puts the glossary in multiple columns.

```

7982 \newglossarystyle{mcolalmtree}{%
7983   \setglossarystyle{almtree}%
7984   \renewenvironment{theglossary}%
7985   {%
7986     \begin{multicols}{\glsmcols}
7987     \def\@gls@prevlevel{-1}%
7988     \mbox{}\par
7989   }%
7990   {\par\end{multicols}}}%
7991 }

```

`mcolalattoreegroup` Like the `mcolalattoree` style but the glossary groups have headings.

```
7992 \newglossarystyle{mcolalattoreegroup}{%
    Base it on the glostylemcolalattoree style:
7993   \setglossarystyle{mcolalattoree}%
    Give each group a heading.
7994   \renewcommand{\glsgroupheading}[1]{\par
7995     \def\@gls@prevlevel{-1}%
7996     \hangindent0pt\relax
7997     \parindent0pt\relax
7998     \textbf{\glsgrouptitle{##1}}\par\indexspace}%
7999 }
```

`colalattoreehypergroup` The `mcolalattoreehypergroup` style is like the `mcolalattoreegroup` style, but has a set of links to the groups at the start of the glossary.

```
8000 \newglossarystyle{mcolalattoreehypergroup}{%
    Base it on the glostylemcolalattoree style:
8001   \setglossarystyle{mcolalattoree}%
    Put the navigation links in the header
8002   \renewcommand*{\glossaryheader}{%
8003     \par
8004     \def\@gls@prevlevel{-1}%
8005     \hangindent0pt\relax
8006     \parindent0pt\relax
8007     \textbf{\glsnavigation}\par\indexspace}%
    Put a hypertarget at the start of each group
8008   \renewcommand*{\glsgroupheading}[1]{%
8009     \par
8010     \def\@gls@prevlevel{-1}%
8011     \hangindent0pt\relax
8012     \parindent0pt\relax
8013     \textbf{\glsnavigationhypertarget{##1}{\glsgrouptitle{##1}}}\par
8014     \indexspace}}
```

5.7 Glossary Styles using supertabular environment (glossary-super package)

The glossary styles defined in the package use the supertabular environment.

```
8015 \ProvidesPackage{glossary-super}[2013/11/14 v4.0 (NLCT)]
```

Requires the package:

```
8016 \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.

```
8017 \ifundefined{glsdescwidth}{%
```

```

8018 \newlength\glsdescwidth
8019 \setlength{\glsdescwidth}{0.6\hsize}
8020 }{}

```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined if has been loaded.

```

8021 \@ifundefined{glspagelistwidth}{%
8022 \newlength\glspagelistwidth
8023 \setlength{\glspagelistwidth}{0.1\hsize}
8024 }{}

```

`super` The super glossary style uses the supertabular environment (it uses lengths defined in the package.)

```
8025 \newglossarystyle{super}{%
```

Put the glossary in a supertabular environment with two columns and no head or tail:

```

8026 \renewenvironment{theglossary}%
8027 {\tablehead{}\tabletail{}}%
8028 \begin{supertabular}{lp{\glsdescwidth}}}%
8029 {\end{supertabular}}%

```

Do nothing at the start of the table:

```
8030 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8031 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```

8032 \renewcommand{\glossentry}[2]{%
8033 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8034 \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline
8035 }%

```

Sub entries put in a row (no name, description and page list in second column):

```

8036 \renewcommand{\subglossentry}[3]{%
8037 &
8038 \glssubentryitem{##2}%
8039 \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
8040 ##3\tabularnewline
8041 }%

```

Blank row between groups:

```

8042 \renewcommand*{\glsgroupskip}{%
8043 \ifglsnogroupskip\else & \tabularnewline\fi}%
8044 }

```

`superborder` The superborder style is like the above, but with horizontal and vertical lines:

```
8045 \newglossarystyle{superborder}{%
```


Base it on the `glostylesuper` style:

```
8046 \setglossarystyle{super}%
```

Put the glossary in a `supertabular` environment with two columns and a horizontal line in the head and tail:

```
8047 \renewenvironment{theglossary}%  
8048 {\tablehead{\hline}\tabletail{\hline}%  
8049 \begin{supertabular}{|l|p{\glsdescwidth}|}%  
8050 {\end{supertabular}}%  
8051 }
```

`superheader` The `superheader` style is like the `super` style, but with a header:

```
8052 \newglossarystyle{superheader}{%
```

Base it on the `glostylesuper` style:

```
8053 \setglossarystyle{super}%
```

Put the glossary in a `supertabular` environment with two columns, a header and no tail:

```
8054 \renewenvironment{theglossary}%  
8055 {\tablehead{\bfseries \entryname &  
8056 \bfseries \descriptionname \tabularnewline}%  
8057 \tabletail{}}%  
8058 \begin{supertabular}{lp{\glsdescwidth}}%  
8059 {\end{supertabular}}%  
8060 }
```

`superheaderborder` The `superheaderborder` style is like the `super` style but with a header and border:

```
8061 \newglossarystyle{superheaderborder}{%
```

Base it on the `glostylesuper` style:

```
8062 \setglossarystyle{super}%
```

Put the glossary in a `supertabular` environment with two columns, a header and horizontal lines above and below the table:

```
8063 \renewenvironment{theglossary}%  
8064 {\tablehead{\hline\bfseries \entryname &  
8065 \bfseries \descriptionname \tabularnewline\hline}%  
8066 \tabletail{\hline}  
8067 \begin{supertabular}{|l|p{\glsdescwidth}|}%  
8068 {\end{supertabular}}%  
8069 }
```

`super3col` The `super3col` style is like the `super` style, but with 3 columns:

```
8070 \newglossarystyle{super3col}{%
```

Put the glossary in a `supertabular` environment with three columns and no head or tail:

```
8071 \renewenvironment{theglossary}%  
8072 {\tablehead{} \tabletail{}}%  
8073 \begin{supertabular}{lp{\glsdescwidth}p{\glspagelistwidth}}%  
8074 {\end{supertabular}}%
```

Do nothing at the start of the table:

```
8075 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8076 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
8077 \renewcommand{\glossentry}[2]{%
8078   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8079   \glossentrydesc{##1} & ##2\tabularnewline
8080 }
```

Sub entries on a row (no name, description in second column, page list in last column):

```
8081 \renewcommand{\subglossentry}[3]{%
8082   &
8083   \glssubentryitem{##2}%
8084   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8085   ##3\tabularnewline
8086 }
```

Blank row between groups:

```
8087 \renewcommand*{\glsgroupskip}{}%
8088 \ifglsnogroupskip\else & &\tabularnewline\fi}%
8089 }
```

super3colborder The `super3colborder` style is like the `super3col` style, but with a border:

```
8090 \newglossarystyle{super3colborder}{}%
```

Base it on the `glostylesuper3col` style:

```
8091 \setglossarystyle{super3col}%
```

Put the glossary in a `supertabular` environment with three columns and a horizontal line in the head and tail:

```
8092 \renewenvironment{theglossary}%
8093   {\tablehead{\hline}\tabletail{\hline}}%
8094   \begin{supertabular}{|l|p{\glsdescwidth}|p{\glspagelistwidth}|}%
8095   {\end{supertabular}}%
8096 }
```

super3colheader The `super3colheader` style is like the `super3col` style but with a header row:

```
8097 \newglossarystyle{super3colheader}{}%
```

Base it on the `glostylesuper3col` style:

```
8098 \setglossarystyle{super3col}%
```

Put the glossary in a `supertabular` environment with three columns, a header and no tail:

```
8099 \renewenvironment{theglossary}%
8100   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
```

```

8101      \bfseries\pagelistname\tabularnewline\tabletail{}}%
8102      \begin{supertabular}{lp{\glsgdescwidth}p{\glspagelistwidth}}}%
8103      {\end{supertabular}}}%
8104 }

```

`per3colheaderborder` The `super3colheaderborder` style is like the `super3col` style but with a header and border:

```

8105 \newglossarystyle{super3colheaderborder}{%
      Base it on the glostylesuper3colborder style:
8106   \setglossarystyle{super3colborder}%
      Put the glossary in a supertabular environment with three columns, a header
      with horizontal lines and a horizontal line in the tail:
8107   \renewenvironment{theglossary}%
8108     {\tablehead{\hline
8109       \bfseries\entryname&\bfseries\descriptionname&
8110       \bfseries\pagelistname\tabularnewline\hline}%
8111     \tabletail{\hline}%
8112     \begin{supertabular}{|l|p{\glsgdescwidth}|p{\glspagelistwidth}|}%
8113     {\end{supertabular}}}%
8114 }

```

`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.

```

8115 \newglossarystyle{super4col}{%
      Put the glossary in a supertabular environment with four columns and no head
      or tail:
8116   \renewenvironment{theglossary}%
8117     {\tablehead{}\tabletail{}}%
8118     \begin{supertabular}{|l|l|l|l|}%
8119     \end{supertabular}}%

```

Do nothing at the start of the table:

```

8120   \renewcommand*{\glossaryheader}{}%

```

No group headings:

```

8121   \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:

```

8122   \renewcommand{\glossentry}[2]{%
8123     \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8124     \glossentrydesc{##1} &
8125     \glossentrysymbol{##1} & ##3\tabularnewline
8126   }%

```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```

8127   \renewcommand{\subglossentry}[3]{%

```

```

8128      &
8129      \glssubentryitem{##2}%
8130      \glstarget{##2}{\strut}\glossentrydesc{##2} &
8131      \glossentrysymbol{##2} & ##3\tabularnewline
8132    }%

```

Blank row between groups:

```

8133  \renewcommand*{\glsgroupskip}{%
8134    \ifglsgnোগroupskip\else & & \tabularnewline\fi}%
8135 }

```

super4colheader The super4colheader style is like the super4col but with a header row.

```

8136 \newglossarystyle{super4colheader}{%
  Base it on the glostylesuper4col style:
8137  \setglossarystyle{super4col}%
  Put the glossary in a supertabular environment with four columns, a header and
  no tail:
8138  \renewenvironment{theglossary}%
8139    {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8140      \bfseries\symbolname &
8141      \bfseries\pagelistname\tabularnewline}%
8142    \tabletail{}}%
8143    \begin{supertabular}{|l|l|l|l|}%
8144    {\end{supertabular}}%
8145 }

```

super4colborder The super4colborder style is like the super4col but with a border.

```

8146 \newglossarystyle{super4colborder}{%
  Base it on the glostylesuper4col style:
8147  \setglossarystyle{super4col}%
  Put the glossary in a supertabular environment with four columns and a hori-
  zontal line in the head and tail:
8148  \renewenvironment{theglossary}%
8149    {\tablehead{\hline}\tabletail{\hline}%
8150    \begin{supertabular}{|l|l|l|l|}%
8151    {\end{supertabular}}%
8152 }

```

super4colheaderborder The super4colheaderborder style is like the super4col but with a header and border.

```

8153 \newglossarystyle{super4colheaderborder}{%
  Base it on the glostylesuper4col style:
8154  \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:

```

8155 \renewenvironment{theglossary}%
8156   {\tablehead{\hline\bfseries\entryname&\bfseries\descriptionname&
8157     \bfseries\symbolname &
8158     \bfseries\pagelistname\tabularnewline\hline}%
8159   \tabletail{\hline}%
8160   \begin{supertabular}{|l|l|l|l|}%
8161   {\end{supertabular}}}%
8162 }
```

altsuper4col The altsuper4col glossary style is like super4col but has provision for multiline descriptions.

```

8163 \newglossarystyle{altsuper4col}{%
  Base it on the glostylesuper4col style:
8164   \setglossarystyle{super4col}%
  Put the glossary in a supertabular environment with four columns and no head
  or tail:
8165   \renewenvironment{theglossary}%
8166   {\tablehead{}\tabletail{}}%
8167   \begin{supertabular}{lp{\glsgdescwidth}lp{\glspagelistwidth}}}%
8168   {\end{supertabular}}}%
8169 }
```

altsuper4colheader The altsuper4colheader style is like the altsuper4col but with a header row.

```

8170 \newglossarystyle{altsuper4colheader}{%
  Base it on the glostylesuper4colheader style:
8171   \setglossarystyle{super4colheader}%
  Put the glossary in a supertabular environment with four columns, a header and
  no tail:
8172   \renewenvironment{theglossary}%
8173   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8174     \bfseries\symbolname &
8175     \bfseries\pagelistname\tabularnewline}\tabletail{}}%
8176   \begin{supertabular}{lp{\glsgdescwidth}lp{\glspagelistwidth}}}%
8177   {\end{supertabular}}}%
8178 }
```

altsuper4colborder The altsuper4colborder style is like the altsuper4col but with a border.

```

8179 \newglossarystyle{altsuper4colborder}{%
  Base it on the glostylesuper4colborder style:
8180   \setglossarystyle{super4colborder}%
  Put the glossary in a supertabular environment with four columns and a hori-
  zontal line in the head and tail:
```

```

8181 \renewenvironment{theglossary}%
8182   {\tablehead{\hline}\tabletail{\hline}%
8183    \begin{supertabular}%
8184     {\lllp{\glsdescwidth}\lllp{\glspagelistwidth}}}%
8185   {\end{supertabular}}%
8186 }

```

`per4colheaderborder` The `altsuper4colheaderborder` style is like the `altsuper4col` but with a header and border.

```

8187 \newglossarystyle{altsuper4colheaderborder}{%

```

Base it on the `glostylessuper4colheaderborder` style:

```

8188 \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:

```

8189 \renewenvironment{theglossary}%
8190   {\tablehead{\hline
8191     \bfseries\entryname &
8192     \bfseries\descriptionname &
8193     \bfseries\symbolname &
8194     \bfseries\pagelistname\tabularnewline\hline}%
8195   \tabletail{\hline}%
8196   \begin{supertabular}%
8197     {\lllp{\glsdescwidth}\lllp{\glspagelistwidth}}}%
8198   {\end{supertabular}}%
8199 }

```

5.8 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.

```

8200 \ProvidesPackage{glossary-superragged}[2013/11/14 v4.0 (NLCT)]

```

Requires the package:

```

8201 \RequirePackage{array}

```

Requires the package:

```

8202 \RequirePackage{supertabular}

```

`\glsdescwidth` This is a length that governs the width of the description column. This may already have been defined.

```

8203 \@ifundefined{glsdescwidth}{%
8204   \newlength\glsdescwidth
8205   \setlength{\glsdescwidth}{0.6\hsize}
8206 }{}

```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```
8207 \@ifundefined{glspagelistwidth}{%
8208   \newlength{glspagelistwidth}
8209   \setlength{glspagelistwidth}{0.1\hsize}
8210 }{}
```

`superragged` The `superragged` glossary style uses the `supertabular` environment.

```
8211 \newglossarystyle{superragged}{%
```

Put the glossary in a `supertabular` environment with two columns and no head or tail:

```
8212   \renewenvironment{theglossary}%
8213     {\tablehead{}\tabletail{}}%
8214     \begin{supertabular}{1>{\raggedright}p{glstdescwidth}}}%
8215     {\end{supertabular}}%
```

Do nothing at the start of the table:

```
8216   \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8217   \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```
8218   \renewcommand{\glossentry}[2]{%
8219     \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8220     \glossentrydesc{##1}\glspostdescription\space ##2%
8221     \tabularnewline
8222   }%
```

Sub entries put in a row (no name, description and page list in second column):

```
8223   \renewcommand{\subglossentry}[3]{%
8224     &
8225     \glssubentryitem{##2}%
8226     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
8227     ##3%
8228     \tabularnewline
8229   }%
```

Blank row between groups:

```
8230   \renewcommand*{\glsgroupskip}{\ifglsgnোগroupskip\else & \tabularnewline\fi}%
8231 }
```

`superraggedborder` The `superraggedborder` style is like the above, but with horizontal and vertical lines:

```
8232 \newglossarystyle{superraggedborder}{%
```

Base it on the `glostylessuperragged` style:

```
8233   \setglossarystyle{superragged}%
```

Put the glossary in a supertabular environment with two columns and a horizontal line in the head and tail:

```
8234 \renewenvironment{theglossary}%
8235   {\tablehead{\hline}\tabletail{\hline}%
8236    \begin{supertabular}{|l|>{\raggedright}p{\glsgdescwidth}|}}%
8237   {\end{supertabular}}%
8238 }
```

superraggedheader The superraggedheader style is like the super style, but with a header:

```
8239 \newglossarystyle{superraggedheader}{%
```

Base it on the glostylesuperragged style:

```
8240 \setglossarystyle{superragged}%
```

Put the glossary in a supertabular environment with two columns, a header and no tail:

```
8241 \renewenvironment{theglossary}%
8242   {\tablehead{\bfseries \entryname & \bfseries \descriptionname
8243    \tabularnewline}%
8244    \tabletail{}}%
8245   \begin{supertabular}{|l>{\raggedright}p{\glsgdescwidth}|}}%
8246   {\end{supertabular}}%
8247 }
```

superraggedheaderborder The superraggedheaderborder style is like the superragged style but with a header and border:

```
8248 \newglossarystyle{superraggedheaderborder}{%
```

Base it on the glostylesuper style:

```
8249 \setglossarystyle{superragged}%
```

Put the glossary in a supertabular environment with two columns, a header and horizontal lines above and below the table:

```
8250 \renewenvironment{theglossary}%
8251   {\tablehead{\hline\bfseries \entryname &
8252    \bfseries \descriptionname\tabularnewline\hline}%
8253    \tabletail{\hline}
8254    \begin{supertabular}{|l|>{\raggedright}p{\glsgdescwidth}|}}%
8255   {\end{supertabular}}%
8256 }
```

superragged3col The superragged3col style is like the superragged style, but with 3 columns:

```
8257 \newglossarystyle{superragged3col}{%
```

Put the glossary in a supertabular environment with three columns and no head or tail:

```
8258 \renewenvironment{theglossary}%
8259   {\tablehead{}\tabletail{}}%
8260   \begin{supertabular}{|l>{\raggedright}p{\glsgdescwidth}%
8261    >{\raggedright}p{\glspagelistwidth}|}}%
8262   {\end{supertabular}}%
```


Do nothing at the start of the table:

```
8263 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8264 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
8265 \renewcommand{\glossentry}[2]{%
8266   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8267   \glossentrydesc{##1} &
8268   ##2\tabularnewline
8269 }%
```

Sub entries on a row (no name, description in second column, page list in last column):

```
8270 \renewcommand{\subglossentry}[3]{%
8271   &
8272   \glssubentryitem{##2}%
8273   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8274   ##3\tabularnewline
8275 }%
```

Blank row between groups:

```
8276 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else & &\tabularnewline\fi}%
8277 }
```

superragged3colborder The `superragged3colborder` style is like the `superragged3col` style, but with a border:

```
8278 \newglossarystyle{superragged3colborder}{}%
```

Base it on the `glostylesuperragged3col` style:

```
8279 \setglossarystyle{superragged3col}%
```

Put the glossary in a `supertabular` environment with three columns and a horizontal line in the head and tail:

```
8280 \renewenvironment{theglossary}%
8281   {\tablehead{\hline}\tabletail{\hline}}%
8282   \begin{supertabular}{|l|>{\raggedright}p{\glsdescwidth}|%
8283     >{\raggedright}p{\glspagerlistwidth}|}%
8284   {\end{supertabular}}%
8285 }
```

superragged3colheader The `superragged3colheader` style is like the `superragged3col` style but with a header row:

```
8286 \newglossarystyle{superragged3colheader}{}%
```

Base it on the `glostylesuperragged3col` style:

```
8287 \setglossarystyle{superragged3col}%
```

Put the glossary in a supertabular environment with three columns, a header and no tail:

```
8288 \renewenvironment{theglossary}%
8289   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8290     \bfseries\pagelistname\tabularnewline}\tabletail{}}%
8291   \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}%
8292     >{\raggedright}p{\glspagelistwidth}}}%
8293   {\end{supertabular}}%
8294 }
```

ght3colheaderborder The superragged3colheaderborder style is like the superragged3col style but with a header and border:

```
8295 \newglossarystyle{superragged3colheaderborder}{%
```

Base it on the glostylesuperragged3colborder style:

```
8296 \setglossarystyle{superragged3colborder}%
```

Put the glossary in a supertabular environment with three columns, a header with horizontal lines and a horizontal line in the tail:

```
8297 \renewenvironment{theglossary}%
8298   {\tablehead{\hline
8299     \bfseries\entryname&\bfseries\descriptionname&
8300     \bfseries\pagelistname\tabularnewline\hline}%
8301   \tabletail{\hline}%
8302   \begin{supertabular}{|l|>{\raggedright}p{\glsgdescwidth}|%
8303     >{\raggedright}p{\glspagelistwidth}|}%
8304   {\end{supertabular}}%
8305 }
```

altsuperragged4col The altsuperragged4col glossary style is like altsuper4col style in the package but uses ragged right formatting in the description and page list columns.

```
8306 \newglossarystyle{altsuperragged4col}{%
```

Put the glossary in a supertabular environment with four columns and no head or tail:

```
8307 \renewenvironment{theglossary}%
8308   {\tablehead{}\tabletail{}}%
8309   \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}l%
8310     >{\raggedright}p{\glspagelistwidth}}}%
8311   {\end{supertabular}}%
```

Do nothing at the start of the table:

```
8312 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8313 \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:

```
8314 \renewcommand{\glossentry}[2]{}%
```

```

8315 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8316 \glossentrydesc{##1} &
8317 \glossentrysymbol{##1} & ##2\tabularnewline
8318 }%

```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```

8319 \renewcommand{\subglossentry}[3]{%
8320 &
8321 \glssubentryitem{##2}%
8322 \glstarget{##2}{\strut}\glossentrydesc{##2} &
8323 \glossentrysymbol{##2} & ##3\tabularnewline
8324 }%

```

Blank row between groups:

```

8325 \renewcommand*{\glsgroupskip}{\ifglsgnogroupskip\else & & \tabularnewline\fi}%
8326 }

```

altperragged4colheader The `altperragged4colheader` style is like the `altperragged4col` style but with a header row.

```

8327 \newglossarystyle{altperragged4colheader}{%

```

Base it on the `glostylealtperragged4col` style:

```

8328 \setglossarystyle{altperragged4col}%

```

Put the glossary in a supertabular environment with four columns, a header and no tail:

```

8329 \renewenvironment{theglossary}%
8330 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8331 \bfseries\symbolname &
8332 \bfseries\pagelistname\tabularnewline}\tabletail{}}%
8333 \begin{supertabular}{1>{\raggedright}p{\glsdescwidth}1%
8334 >{\raggedright}p{\glspagelistwidth}}}%
8335 {\end{supertabular}}%
8336 }

```

altperragged4colborder The `altperragged4colborder` style is like the `altperragged4col` style but with a border.

```

8337 \newglossarystyle{altperragged4colborder}{%

```

Base it on the `glostylealtperragged4col` style:

```

8338 \setglossarystyle{altperragged4col}%

```

Put the glossary in a supertabular environment with four columns and a horizontal line in the head and tail:

```

8339 \renewenvironment{theglossary}%
8340 {\tablehead{\hline}\tabletail{\hline}%
8341 \begin{supertabular}%
8342 {\l|>{\raggedright}p{\glsdescwidth}l|}%
8343 >{\raggedright}p{\glspagelistwidth}l}%
8344 {\end{supertabular}}%
8345 }

```

`ged4colheaderborder` The `altsuperragged4colheaderborder` style is like the `altsuperragged4col` style but with a header and border.

```
8346 \newglossarystyle{altsuperragged4colheaderborder}{%
```

Base it on the `glostylealtsuperragged4col` style:

```
8347 \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:

```
8348 \renewenvironment{theglossary}{%
8349   {\tablehead{\hline
8350     \bfseries\entryname &
8351     \bfseries\descriptionname &
8352     \bfseries\symbolname &
8353     \bfseries\pagelistname\tabularnewline\hline}%
8354   \tabletail{\hline}%
8355   \begin{supertabular}%
8356     {|l|>{\raggedright}p{\glstdescwidth}|l|}%
8357     >{\raggedright}p{\glspagelistwidth}|}%
8358   {\end{supertabular}}}%
8359 }
```

5.9 Tree Styles (`glossary-tree.sty`)

The style file defines glossary styles that have a tree-like structure. These are designed for hierarchical glossaries.

```
8360 \ProvidesPackage{glossary-tree}[2015/02/03 v4.13 (NLCT)]
```

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```
8361 \providecommand{\indexspace}{%
8362   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
8363 }
```

`\glstreenamefmt` Format used to display the name in the tree styles. (This may be counteracted by `\glstnamefont`.) This command is also used to format the group headings.

```
8364 \newcommand*{\glstreenamefmt}[1]{\textbf{#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.

```
8365 \newglossarystyle{index}{%
```

Set the paragraph indentation and skip and define `\item` to be the same as that used by the `index`:

```
8366 \renewenvironment{theglossary}{%
8367   {\setlength{\parindent}{0pt}}%
```

```

8368     \setlength{\parskip}{0pt plus 0.3pt}%
8369     \let\item\@idxitem}%

8370     {\par}%

```

Do nothing at the start of the environment:

```
8371 \renewcommand*{\glossaryheader}{}%
```

No group headers:

```
8372 \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.

```

8373 \renewcommand*{\glossentry}[2]{%
8374     \item\glstentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
8375     \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
8376     \space \glossentrydesc{##1}\glspostdescription\space ##2%
8377 }%

```

Sub entries: level 1 entries use `\subitem`, levels greater than 1 use `\subsubitem`. The level (##1) shouldn't be 0, as that's catered by `\glossentry`, but for completeness, if the level is 0, `\item` is used. The name is put in bold, followed by the symbol in brackets (if it exists), the description and the page list.

```

8378 \renewcommand{\subglossentry}[3]{%
8379     \ifcase##1\relax
8380     % level 0
8381     \item
8382     \or
8383     % level 1
8384     \subitem
8385     \glssubentryitem{##2}%
8386     \else
8387     % all other levels
8388     \subsubitem
8389     \fi
8390     \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
8391     \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
8392     \space\glossentrydesc{##2}\glspostdescription\space ##3%
8393 }%

```

Vertical gap between groups is the same as that used by indices:

```
8394 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}}
```

indexgroup The `indexgroup` style is like the `index` style but has headings.

```
8395 \newglossarystyle{indexgroup}{%
```

Base it on the `glostyleindex` style:

```
8396 \setglossarystyle{index}%
```

Add a heading for each group. This puts the group's title in bold followed by a vertical gap.

```

8397 \renewcommand*{\glsgroupheading}[1]{%
8398 \item\glstreenamefmt{\glsgrouptitle{##1}}\indexspace}%
8399 }

```

indexhypergroup The indexhypergroup style is like the indexgroup style but has hyper navigation.

```
8400 \newglossarystyle{indexhypergroup}{%
```

Base it on the glostyleindex style:

```
8401 \setglossarystyle{index}%
```

Put navigation links to the groups at the start of the glossary:

```
8402 \renewcommand*{\glossaryheader}{%
```

```
8403 \item\glstreenamefmt{\glsnavigation}\indexspace}%
```

Add a heading for each group (with a target). The group's title is in bold followed by a vertical gap.

```
8404 \renewcommand*{\glsgroupheading}[1]{%
```

```
8405 \item\glstreenamefmt{\glshypertarget{##1}}{\glsgrouptitle{##1}}}%
```

```
8406 \indexspace}%
```

```
8407 }
```

tree The tree glossary style is similar in style to the index style, but can have arbitrary levels.

```
8408 \newglossarystyle{tree}{%
```

Set the paragraph indentation and skip:

```
8409 \renewenvironment{theglossary}%
```

```
8410 {\setlength{\parindent}{0pt}%
```

```
8411 \setlength{\parskip}{0pt plus 0.3pt}}%
```

```
8412 {}%
```

Do nothing at the start of the theglossary environment:

```
8413 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8414 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries: name in bold, followed by symbol in brackets (if it exists), the description and the page list:

```
8415 \renewcommand{\glossentry}[2]{%
```

```
8416 \hangindent0pt\relax
```

```
8417 \parindent0pt\relax
```

```
8418 \glstryitem{##1}\glstreenamefmt{\glstarget{##1}}{\glossentryname{##1}}}%
```

```
8419 \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
```

```
8420 \space\glossentrydesc{##1}\glspostdescription\space##2\par
```

```
8421 }%
```

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.

```
8422 \renewcommand{\subglossentry}[3]{%
```

```

8423 \hangindent##1\glstreeindent\relax
8424 \parindent##1\glstreeindent\relax
8425 \ifnum##1=1\relax
8426 \glssubentryitem{##2}%
8427 \fi
8428 \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
8429 \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
8430 \space\glossentrydesc{##2}\glspostdescription\space ##3\par
8431 }%

```

Vertical gap between groups is the same as that used by indices:

```

8432 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}

```

treegroup Like the tree style but the glossary groups have headings.

```

8433 \newglossarystyle{treegroup}{%

```

Base it on the glostyletree style:

```

8434 \setglossarystyle{tree}%

```

Each group has a heading (in bold) followed by a vertical gap):

```

8435 \renewcommand{\glsgroupheading}[1]{\par
8436 \noindent\glstreenamefmt{\glsgrouptitle{##1}}\par\indexspace}%
8437 }

```

treehypergroup The treehypergroup style is like the treegroup style, but has a set of links to the groups at the start of the glossary.

```

8438 \newglossarystyle{treehypergroup}{%

```

Base it on the glostyletree style:

```

8439 \setglossarystyle{tree}%

```

Put navigation links to the groups at the start of the theglossary environment:

```

8440 \renewcommand*{\glossaryheader}{%
8441 \par\noindent\glstreenamefmt{\glsnavigation}\par\indexspace}%

```

Each group has a heading (in bold with a target) followed by a vertical gap):

```

8442 \renewcommand*{\glsgroupheading}[1]{%
8443 \par\noindent
8444 \glstreenamefmt{\glsnavigationtarget{##1}{\glsgrouptitle{##1}}}\par
8445 \indexspace}%
8446 }

```

\glstreeindent Length governing left indent for each level of the tree style.

```

8447 \newlength\glstreeindent
8448 \setlength{\glstreeindent}{10pt}

```

treenoname The treenoname glossary style is like the tree style, but doesn't print the name or symbol for sub-levels.

```

8449 \newglossarystyle{treenoname}{%

```

Set the paragraph indentation and skip:

```
8450 \renewenvironment{theglossary}%  
8451   {\setlength{\parindent}{0pt}}%  
8452   \setlength{\parskip}{0pt plus 0.3pt}}%  
8453   {}%
```

No header:

```
8454 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8455 \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.

```
8456 \renewcommand{\glossentry}[2]{%  
8457   \hangindent0pt\relax  
8458   \parindent0pt\relax  
8459   \glstryitem{##1}\glstreenamfmt{\glstarget{##1}{\glossentryname{##1}}}%  
8460   \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%  
8461   \space\glossentrydesc{##1}\glspostdescription\space##2\par  
8462 }%
```

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.

```
8463 \renewcommand{\subglossentry}[3]{%  
8464   \hangindent##1\glstreeindent\relax  
8465   \parindent##1\glstreeindent\relax  
8466   \ifnum##1=1\relax  
8467     \glssubentryitem{##2}%  
8468   \fi  
8469   \glstarget{##2}{\strut}%  
8470   \glossentrydesc{##2}\glspostdescription\space##3\par  
8471 }%
```

Vertical gap between groups is the same as that used by indices:

```
8472 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%  
8473 }
```

`treenonamegroup` Like the `treenoname` style but the glossary groups have headings.

```
8474 \newglossarystyle{treenonamegroup}{%
```

Base it on the `glostyletreenoname` style:

```
8475 \setglossarystyle{treenoname}%
```

Give each group a heading:

```
8476 \renewcommand{\glsgroupheading}[1]{\par  
8477   \noindent\glstreenamfmt{\glsgrouptitle{##1}}\par\indexspace}%  
8478 }
```

`treenonamehypergroup` The `treenonamehypergroup` style is like the `treenonamegroup` style, but has a set of links to the groups at the start of the glossary.

```
8479 \newglossarystyle{treenonamehypergroup}{%
```


Base it on the `glostytreeoname` style:

```
8480 \setglossarystyle{treeoname}%
```

Put navigation links to the groups at the start of the `theglossary` environment:

```
8481 \renewcommand*{\glossaryheader}{%
8482   \par\noindent\glstreenamfmt{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
8483 \renewcommand*{\glsgroupheading}[1]{%
8484   \par\noindent
8485   \glstreenamfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8486   \indexspace}%
8487 }
```

`\glssetwidest` `\glssetwidest[level]{text}` sets the widest text for the given level. It is used by the `alttree` glossary styles to determine the indentation of each level.

```
8488 \newcommand*{\glssetwidest}[2][0]{%
8489   \expandafter\def\csname @glswidestname\romannumeral#1\endcsname{%
8490     #2}%
8491 }
```

`\@glswidestname` Initialise `\@glswidestname`.

```
8492 \newcommand*{\@glswidestname}{}%
```

`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`.

```
8493 \newglossarystyle{alttree}{%
```

Redefine the `theglossary` environment.

```
8494 \renewenvironment{theglossary}%
8495   {\def\@gls@prevlevel{-1}%
8496    \mbox{}\par}%
8497   {\par}%
```

Set the header and group headers to nothing.

```
8498 \renewcommand*{\glossaryheader}{}%
8499 \renewcommand*{\glsgroupheading}[1]{}%
```

Redefine the way that the level 0 entries are displayed.

```
8500 \renewcommand{\glosseentry}[2]{%
8501   \ifnum\@gls@prevlevel=0\relax
8502   \else
```

Find out how big the indentation should be by measuring the widest entry.

```
8503     \settowidth{\glstreeindent}{\glstreenamfmt{\@glswidestname\space}}%
8504     \fi
```

Set the hangindent and paragraph indent.

```
8505     \hangindent\glstreeindent
8506     \parindent\glstreeindent
```

Put the name to the left of the paragraph block.

```
8507 \makebox[0pt][r]{\makebox[\glstreeindent][l]{%
8508 \glstryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}}}%
```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
8509 \ifglshassymbol{##1}{(\glossentrysymbol{##1})\space}{}%
```

Do the description followed by the description terminator and location list.

```
8510 \glossentrydesc{##1}\glspostdescription \space ##2\par
```

Set the previous level to 0.

```
8511 \def\@gls@prevlevel{0}%
8512 }%
```

Redefine the way sub-entries are displayed.

```
8513 \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.

```
8514 \ifnum##1=1\relax
8515 \glssubentryitem{##2}%
8516 \fi
```

If the level hasn't changed, keep the same settings, otherwise adjust `\glstreeindent` accordingly.

```
8517 \ifnum\@gls@prevlevel=##1\relax
8518 \else
```

Compute the widest entry for this level, or for level 0 if not defined for this level.

Store in `\gls@tmplen`

```
8519 \@ifundefined{@glswidestname\romannumeral##1}{%
8520 \settowidth{\gls@tmplen}{\glstreenamefmt{\@glswidestname\space}}}%
8521 \settowidth{\gls@tmplen}{\glstreenamefmt{%
8522 \csname @glswidestname\romannumeral##1\endcsname\space}}}%
```

Determine if going up or down a level

```
8523 \ifnum\@gls@prevlevel<##1\relax
```

Depth has increased, so add the width of the widest entry to `\glstreeindent`.

```
8524 \setlength\glstreeindent\gls@tmplen
8525 \addtolength\glstreeindent\parindent
8526 \parindent\glstreeindent
8527 \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`.

```
8528 \@ifundefined{@glswidestname\romannumeral\@gls@prevlevel}{%
8529 \settowidth{\glstreeindent}{\glstreenamefmt{%
8530 \@glswidestname\space}}}%
8531 \settowidth{\glstreeindent}{\glstreenamefmt{%
8532 \csname @glswidestname\romannumeral\@gls@prevlevel
8533 \endcsname\space}}}%
```

Subtract this length from the previous level's paragraph indent and set to
`\glstreeindent`.

```
8534      \addtolength\parindent{-\glstreeindent}%
8535      \setlength\glstreeindent\parindent
8536      \fi
8537      \fi
```

Set the hanging indentation.

```
8538      \hangindent\glstreeindent
```

Put the name to the left of the paragraph block

```
8539      \makebox[0pt][r]{\makebox[\glstemplen][l]{%
8540      \glstreenamfmt{\glstarget{##2}{\glossentryname{##2}}}}}%
```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
8541      \ifglshassymbol{##2}{(\glossentrysymbol{##2})\space}{}%
```

Do the description followed by the description terminator and location list.

```
8542      \glossentrydesc{##2}\glspostdescription\space ##3\par
```

Set the previous level macro to the current level.

```
8543      \def\@gls@prevlevel{##1}%
8544      }%
```

Vertical gap between groups is the same as that used by indices:

```
8545      \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
8546 }
```

almtreegroup Like the almtree style but the glossary groups have headings.

```
8547 \newglossarystyle{almtreegroup}{%
```

Base it on the glostylealmtree style:

```
8548      \setglossarystyle{almtree}%
```

Give each group a heading.

```
8549      \renewcommand{\glsgroupheading}[1]{\par
8550      \def\@gls@prevlevel{-1}%
8551      \hangindent0pt\relax
8552      \parindent0pt\relax
8553      \glstreenamfmt{\glsgetgrouptitle{##1}}\par\indexspace}%
8554 }
```

almtreehypergroup The almtreehypergroup style is like the almtreegroup style, but has a set of links to the groups at the start of the glossary.

```
8555 \newglossarystyle{almtreehypergroup}{%
```

Base it on the glostylealmtree style:

```
8556      \setglossarystyle{almtree}%
```

Put the navigation links in the header

```
8557      \renewcommand*{\glossaryheader}{%
8558      \par
```

```

8559 \def\@gls@prevlevel{-1}%
8560 \hangindent0pt\relax
8561 \parindent0pt\relax
8562 \glstreenamefmt{\glsnavigation}\par\indexspace}%

Put a hypertarget at the start of each group
8563 \renewcommand*\@gls@groupheading[1]{%
8564 \par
8565 \def\@gls@prevlevel{-1}%
8566 \hangindent0pt\relax
8567 \parindent0pt\relax
8568 \glstreenamefmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8569 \indexspace}}

```

6 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.

```

8570 \NeedsTeXFormat{LaTeX2e}
8571 \ProvidesPackage{glossaries-compatible-207}[2011/04/02 v1.0 (NLCT)]

```

`\GlsAddXdyAttribute` Adds an attribute in old format.

```

8572 \ifglsxindy
8573 \renewcommand*\GlsAddXdyAttribute[1]{%
8574 \edef\@xdyattributes{\@xdyattributes ^^J \string"#1\string"}%
8575 \expandafter\toks@\expandafter{\@xdylocref}%
8576 \edef\@xdylocref{\the\toks@ ^^J%
8577 (markup-locref
8578 :open \string"\string~n\string\setentrycounter
8579 {\noexpand\glscounter}%
8580 \expandafter\string\csname#1\endcsname
8581 \expandafter\@gobble\string\{\string" ^^J
8582 :close \string"\expandafter\@gobble\string\}\string" ^^J
8583 :attr \string"#1\string"))}

```

Only has an effect before `\writeist`:

```

8584 \fi

```

`\GlsAddXdyCounters`

```

8585 \renewcommand*\GlsAddXdyCounters[1]{%
8586 \GlossariesWarning{\string\GlsAddXdyCounters\space not available
8587 in compatibility mode.}%
8588 }

```

Add predefined attributes

```

8589 \GlsAddXdyAttribute{glsnumberformat}
8590 \GlsAddXdyAttribute{textrm}

```

```

8591 \GlsAddXdyAttribute{textsf}
8592 \GlsAddXdyAttribute{texttt}
8593 \GlsAddXdyAttribute{textbf}
8594 \GlsAddXdyAttribute{textmd}
8595 \GlsAddXdyAttribute{textit}
8596 \GlsAddXdyAttribute{textup}
8597 \GlsAddXdyAttribute{textsl}
8598 \GlsAddXdyAttribute{textsc}
8599 \GlsAddXdyAttribute{emph}
8600 \GlsAddXdyAttribute{glshypernumber}
8601 \GlsAddXdyAttribute{hyperrrm}
8602 \GlsAddXdyAttribute{hypersf}
8603 \GlsAddXdyAttribute{hypertt}
8604 \GlsAddXdyAttribute{hyperbf}
8605 \GlsAddXdyAttribute{hypermd}
8606 \GlsAddXdyAttribute{hyperit}
8607 \GlsAddXdyAttribute{hyperup}
8608 \GlsAddXdyAttribute{hypersl}
8609 \GlsAddXdyAttribute{hypersc}
8610 \GlsAddXdyAttribute{hyperemph}

```

\GlsAddXdyLocation Restore v2.07 definition:

```

8611 \ifglxsindy
8612 \renewcommand*{\GlsAddXdyLocation}[2]{%
8613 \edef\@xdyuserlocationdefs{%
8614 \@xdyuserlocationdefs ^^J%
8615 (define-location-class \string"#1\string"^^J\space\space
8616 \space(#2))
8617 }%
8618 \edef\@xdyuserlocationnames{%
8619 \@xdyuserlocationnames^^J\space\space\space
8620 \string"#1\string"}%
8621 }
8622 \fi

```

\@do@wrglossary

```

8623 \renewcommand{\@do@wrglossary}[1]{%

```

Determine whether to use xindy or makeindex syntax

```

8624 \ifglxsindy

```

Need to determine if the formatting information starts with a (or) indicating a range.

```

8625 \expandafter\@glo@check@mkidxrangechar\@glsnumberformat\@nil
8626 \def\@glo@range{}%
8627 \expandafter\if\@glo@prefix(\relax
8628 \def\@glo@range{:open-range}%
8629 \else
8630 \expandafter\if\@glo@prefix)\relax
8631 \def\@glo@range{:close-range}%

```

8632 \fi

8633 \fi

Get the location and escape any special characters

8634 \protected@edef\@glslocref{\theglsentrycounter}%

8635 \@gls@checkmkidxchars\@glslocref

Write to the glossary file using xindy syntax.

8636 \glossary[\csname glo@#1@type\endcsname]{%

8637 (indexentry :tkey (\csname glo@#1@index\endcsname)

8638 :locref \string"\@glslocref\string" %

8639 :attr \string"\@glo@suffix\string" \@glo@range

8640)

8641 }%

8642 \else

Convert the format information into the format required for makeindex

8643 \@set@glo@numformat\@glo@numfmt\@gls@counter\@glsnumberformat

Write to the glossary file using makeindex syntax.

8644 \glossary[\csname glo@#1@type\endcsname]{%

8645 \string\glossaryentry{\csname glo@#1@index\endcsname

8646 \@gls@encapchar\@glo@numfmt}{\theglsentrycounter}}%

8647 \fi

8648 }

\@set@glo@numformat Only had 3 arguments in v2.07

8649 \def\@set@glo@numformat#1#2#3{%

8650 \expandafter\@glo@check@mkidxrangechar#3\@nil

8651 \protected@edef#1{%

8652 \@glo@prefix setentrycounter[] {#2}%

8653 \expandafter\string\csname\@glo@suffix\endcsname

8654 }%

8655 \@gls@checkmkidxchars#1%

8656 }

\writeist Redefine \writeist back to the way it was in v2.07, but change \istfile to
\glswrite.

8657 \ifglsxindy

8658 \def\writeist{%

8659 \openout\glswrite=\istfilename

8660 \write\glswrite{;; xindy style file created by the glossaries

8661 package in compatible-2.07 mode}%

8662 \write\glswrite{;; for document '\jobname' on

8663 \the\year-\the\month-\the\day}%

8664 \write\glswrite{^^J; required styles^^J}

8665 \@for\@xdystyle:=\@xdyrequiredstyles\do{%

8666 \ifx\@xdystyle\@empty

8667 \else

8668 \protected@write\glswrite{{(require

8669 \string"\@xdystyle.xdy\string")}}%

```

8670     \fi
8671 }%
8672 \write\glswrite{^^J%
8673     ; list of allowed attributes (number formats)^^J}%
8674 \write\glswrite{(define-attributes ((\@xdyattributes)))}%
8675 \write\glswrite{^^J; user defined alphabets^^J}%
8676 \write\glswrite{\@xdyuseralphabets}%
8677 \write\glswrite{^^J; location class definitions^^J}%
8678 \protected@edef\@gls@roman{\@roman{0}\string"
8679     \string"roman-numbers-lowercase\string" :sep \string"}}%
8680 \@onelevel@sanitize\@gls@roman
8681 \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
8682     :sep \string"}%
8683 \@onelevel@sanitize\@tmp
8684 \ifx\@tmp\@gls@roman
8685     \write\glswrite{(define-location-class
8686         \string"roman-page-numbers\string"^^J\space\space\space
8687         (\string"roman-numbers-lowercase\string")
8688         :min-range-length \@glsminrange)}}%
8689 \else
8690     \write\glswrite{(define-location-class
8691         \string"roman-page-numbers\string"^^J\space\space\space
8692         (:sep "\@gls@roman")
8693         :min-range-length \@glsminrange)}}%
8694 \fi
8695 \write\glswrite{(define-location-class
8696     \string"Roman-page-numbers\string"^^J\space\space\space
8697     (\string"roman-numbers-uppercase\string")
8698     :min-range-length \@glsminrange)}}%
8699 \write\glswrite{(define-location-class
8700     \string"arabic-page-numbers\string"^^J\space\space\space
8701     (\string"arabic-numbers\string")
8702     :min-range-length \@glsminrange)}}%
8703 \write\glswrite{(define-location-class
8704     \string"alpha-page-numbers\string"^^J\space\space\space
8705     (\string"alpha\string")
8706     :min-range-length \@glsminrange)}}%
8707 \write\glswrite{(define-location-class
8708     \string"Alpha-page-numbers\string"^^J\space\space\space
8709     (\string"ALPHA\string")
8710     :min-range-length \@glsminrange)}}%
8711 \write\glswrite{(define-location-class
8712     \string"Appendix-page-numbers\string"^^J\space\space\space
8713     (\string"ALPHA\string"
8714     :sep \string"\@glsAlphacompositor\string"
8715     \string"arabic-numbers\string")
8716     :min-range-length \@glsminrange)}}%
8717 \write\glswrite{(define-location-class
8718     \string"arabic-section-numbers\string"^^J\space\space\space

```

```

8719      (\string"arabic-numbers\string"
8720       :sep \string"\glscompositor\string"
8721       \string"arabic-numbers\string")
8722       :min-range-length \@glsminrange))}%
8723 \write\glswrite{^^J; user defined location classes}%
8724 \write\glswrite{\@xdyuserlocationdefs}%
8725 \write\glswrite{^^J; define cross-reference class^^J}%
8726 \write\glswrite{(define-crossref-class \string"see\string"
8727   :unverified )}%
8728 \write\glswrite{(markup-crossref-list
8729   :class \string"see\string"^^J\space\space\space
8730   :open \string"\string\glsseeformat\string"
8731   :close \string"{}\string"))}%
8732 \write\glswrite{^^J; define the order of the location classes}%
8733 \write\glswrite{(define-location-class-order
8734   (\@xdylocationclassorder))}%
8735 \write\glswrite{^^J; define the glossary markup^^J}%
8736 \write\glswrite{(markup-index^^J\space\space\space
8737   :open \string"\string
8738     \glossarysection[\string\glossarytoctitle]{\string
8739     \glossarytitle}\string\glossarypreamble\string~\n\string\begin
8740     {theglossary}\string\glossaryheader\string~\n\string" ^^J\space
8741     \space\space:close \string"\expandafter\@gobble
8742     \string\%\string~\n\string
8743     \end{theglossary}\string\glossarypostamble
8744     \string~\n\string" ^^J\space\space\space
8745     :tree)}}%
8746 \write\glswrite{(markup-letter-group-list
8747   :sep \string"\string\glsgroupskip\string~\n\string"))}%
8748 \write\glswrite{(markup-indexentry
8749   :open \string"\string\relax \string\glsresetentrylist
8750     \string~\n\string"))}%
8751 \write\glswrite{(markup-locclass-list :open
8752   \string"\glsopenbrace\string\glossaryentrynumbers
8753   \glsopenbrace\string\relax\space \string"^^J\space\space\space
8754   :sep \string", \string"
8755   :close \string"\glsclosebrace\glsclosebrace\string"))}%
8756 \write\glswrite{(markup-locref-list
8757   :sep \string"\string\delimN\space\string"))}%
8758 \write\glswrite{(markup-range
8759   :sep \string"\string\delimR\space\string"))}%
8760 \@onelevel@sanitize\gls@suffiF
8761 \@onelevel@sanitize\gls@suffiFF
8762 \ifx\gls@suffiF\@empty
8763 \else
8764   \write\glswrite{(markup-range
8765     :close "\gls@suffiF" :length 1 :ignore-end)}}%
8766 \fi
8767 \ifx\gls@suffiFF\@empty

```



```

8768 \else
8769 \write\glswrite{(markup-range
8770 :close "\gls@suffixFF" :length 2 :ignore-end)}}%
8771 \fi
8772 \write\glswrite{^^J; define format to use for locations^^J}%
8773 \write\glswrite{\@xdylocref}%
8774 \write\glswrite{^^J; define letter group list format^^J}%
8775 \write\glswrite{(markup-letter-group-list
8776 :sep \string\string\glsgroupskip\string~n\string)}}%
8777 \write\glswrite{^^J; letter group headings^^J}%
8778 \write\glswrite{(markup-letter-group
8779 :open-head \string\string\glsgroupheading
8780 \glsopenbrace\string^^J\space\space\space
8781 :close-head \string\glsclosebrace\string)}}%
8782 \write\glswrite{^^J; additional letter groups^^J}%
8783 \write\glswrite{\@xdylettergroups}%
8784 \write\glswrite{^^J; additional sort rules^^J}%
8785 \write\glswrite{\@xdysortrules}%
8786 \noist}
8787 \else
8788 \edef\@gls@actualchar{\string?}
8789 \edef\@gls@encapchar{\string|}
8790 \edef\@gls@levelchar{\string!}
8791 \edef\@gls@quotechar{\string"}
8792 \def\writeist{\relax
8793 \openout\glswrite=\istfilename
8794 \write\glswrite{\expandafter\@gobble\string\% makeindex style file
8795 created by the glossaries package}
8796 \write\glswrite{\expandafter\@gobble\string\% for document
8797 '\jobname' on \the\year-\the\month-\the\day}
8798 \write\glswrite{actual '\@gls@actualchar'}
8799 \write\glswrite{encap '\@gls@encapchar'}
8800 \write\glswrite{level '\@gls@levelchar'}
8801 \write\glswrite{quote '\@gls@quotechar'}
8802 \write\glswrite{keyword \string\string\glossaryentry\string"}
8803 \write\glswrite{preamble \string\string\glossarysection[\string
8804 \glossarytoctitle]{\string\glossarytitle}\string
8805 \glossarypreamble\string\n\string\begin{theglossary}\string
8806 \glossaryheader\string\n\string"}
8807 \write\glswrite{postamble \string\string\%\string\n\string
8808 \end{theglossary}\string\glossarypostamble\string\n
8809 \string"}
8810 \write\glswrite{group_skip \string\string\glsgroupskip\string\n
8811 \string"}
8812 \write\glswrite{item_0 \string\string\%\string\n\string"}
8813 \write\glswrite{item_1 \string\string\%\string\n\string"}
8814 \write\glswrite{item_2 \string\string\%\string\n\string"}
8815 \write\glswrite{item_01 \string\string\%\string\n\string"}
8816 \write\glswrite{item_x1

```

```

8817     \string"\string\relax \string\glsresetentrylist\string\n
8818     \string"}
8819     \write\glswrite{item_12 \string"\string%\string\n\string"}
8820     \write\glswrite{item_x2
8821     \string"\string\relax \string\glsresetentrylist\string\n
8822     \string"}
8823     \write\glswrite{delim_0 \string"\string\{\string
8824     \glossaryentrynumbers\string\{\string\relax \string"}
8825     \write\glswrite{delim_1 \string"\string\{\string
8826     \glossaryentrynumbers\string\{\string\relax \string"}
8827     \write\glswrite{delim_2 \string"\string\{\string
8828     \glossaryentrynumbers\string\{\string\relax \string"}
8829     \write\glswrite{delim_t \string"\string\}\string\}\string"}
8830     \write\glswrite{delim_n \string"\string\delimN \string"}
8831     \write\glswrite{delim_r \string"\string\delimR \string"}
8832     \write\glswrite{headings_flag 1}
8833     \write\glswrite{heading_prefix
8834     \string"\string\glsgroupheading\string\{\string"}
8835     \write\glswrite{heading_suffix
8836     \string"\string\}\string\relax
8837     \string\glsresetentrylist \string"}
8838     \write\glswrite{symhead_positive \string"glssymbols\string"}
8839     \write\glswrite{numhead_positive \string"glssymbols\string"}
8840     \write\glswrite{page_compositor \string"glscpositor\string"}
8841     \@gls@escbsdq\gls@suffixF
8842     \@gls@escbsdq\gls@suffixFF
8843     \ifx\gls@suffixF\@empty
8844     \else
8845     \write\glswrite{suffix_2p \string"\gls@suffixF\string"}
8846     \fi
8847     \ifx\gls@suffixFF\@empty
8848     \else
8849     \write\glswrite{suffix_3p \string"\gls@suffixFF\string"}
8850     \fi
8851     \noist
8852   }
8853 \fi

```

\noist

```
8854 \renewcommand*{\noist}{\let\writeist\relax}
```

Compatibility macros.

```

8855 \NeedsTeXFormat{LaTeX2e}
8856 \ProvidesPackage{glossaries-compatible-307}[2013/11/14 v4.0 (NLCT)]

```

Compatibility macros for predefined glossary styles:

`compatglossarystyle` Defines a compatibility glossary style.

```

8857 \newcommand{\compatglossarystyle}[2]{%
8858   \ifcsundef{@glscompstyle#1}%

```

```

8859  {%
8860    \csdef{@glscompstyle@#1}{#2}%
8861  }%
8862  {%
8863    \PackageError{glossaries}{Glossary compatibility style ‘#1’ is already defined}{}%
8864  }%
8865 }

```

Backward compatible inline style.

```

8866 \compatglossarystyle{inline}{%
8867   \renewcommand{\glossaryentryfield}[5]{%
8868     \glsinlinedopostchild
8869     \gls@inlinesep
8870     \def\glo@desc{##3}%
8871     \def\@no@post@desc{\nopostdesc}%
8872     \glsentryitem{##1}\glsinlinenameformat{##1}{##2}%
8873     \ifx\glo@desc\@no@post@desc
8874       \glsinlineemptydescformat{##4}{##5}%
8875     \else
8876       \ifstrempy{##3}%
8877         {\glsinlineemptydescformat{##4}{##5}}%
8878         {\glsinlinedescformat{##3}{##4}{##5}}%
8879     \fi
8880     \ifglshaschildren{##1}%
8881     {%
8882       \glsresetsubentrycounter
8883       \glsinlineparentchildseparator
8884       \def\gls@inlinesubsep{}%
8885       \def\gls@inlinepostchild{\glsinlinepostchild}%
8886     }%
8887   }%
8888   \def\gls@inlinesep{\glsinlineseparator}%
8889 }%

```

Sub-entries display description:

```

8890 \renewcommand{\glossarysubentryfield}[6]{%
8891   \gls@inlinesubsep%
8892   \glsinlinesubnameformat{##2}{##3}%
8893   \glssubentryitem{##2}\glsinlinesubdescformat{##4}{##5}{##6}%
8894   \def\gls@inlinesubsep{\glsinlinesubseparator}%
8895 }%
8896 }

```

Backward compatible list style.

```

8897 \compatglossarystyle{list}{%
8898   \renewcommand*{\glossaryentryfield}[5]{%
8899     \item[\glsentryitem{##1}\glstarget{##1}{##2}]
8900     ##3\glspostdescription\space ##5}%

```

Sub-entries continue on the same line:

```

8901 \renewcommand*{\glossarysubentryfield}[6]{%

```

```

8902 \glssubentryitem{##2}%
8903 \glstarget{##2}{\strut}##4\glspostdescription\space ##6.}%
8904 }

```

Backward compatible listgroup style.

```

8905 \compatglossarystyle{listgroup}{%
8906 \csuse{@glscmpstyle@list}%
8907 }%

```

Backward compatible listhypergroup style.

```

8908 \compatglossarystyle{listhypergroup}{%
8909 \csuse{@glscmpstyle@list}%
8910 }%

```

Backward compatible altlist style.

```

8911 \compatglossarystyle{altlist}{%
8912 \renewcommand*{\glossaryentryfield}[5]{%
8913 \item[\glssubentryitem{##1}\glstarget{##1}{##2}]%
8914 \mbox{\par\nobreak\@afterheading
8915 ##3\glspostdescription\space ##5}%
8916 \renewcommand{\glossarysubentryfield}[6]{%
8917 \par
8918 \glssubentryitem{##2}%
8919 \glstarget{##2}{\strut}##4\glspostdescription\space ##6}%
8920 }%

```

Backward compatible altlistgroup style.

```

8921 \compatglossarystyle{altlistgroup}{%
8922 \csuse{@glscmpstyle@altlist}%
8923 }%

```

Backward compatible altlisthypergroup style.

```

8924 \compatglossarystyle{altlisthypergroup}{%
8925 \csuse{@glscmpstyle@altlist}%
8926 }%

```

Backward compatible listdotted style.

```

8927 \compatglossarystyle{listdotted}{%
8928 \renewcommand*{\glossaryentryfield}[5]{%
8929 \item[\makebox[\glslistdottedwidth][l]{%
8930 \glssubentryitem{##1}\glstarget{##1}{##2}%
8931 \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##3}%
8932 \renewcommand*{\glossarysubentryfield}[6]{%
8933 \item[\makebox[\glslistdottedwidth][l]{%
8934 \glssubentryitem{##2}%
8935 \glstarget{##2}{##3}%
8936 \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##4}%
8937 }%

```

Backward compatible sublistdotted style.

```

8938 \compatglossarystyle{sublistdotted}{%
8939 \csuse{@glscmpstyle@listdotted}%

```

```

8940 \renewcommand*{\glossaryentryfield}[5]{%
8941 \item[\glentryitem{##1}\glstarget{##1}{##2}]}%
8942 }%

```

Backward compatible long style.

```

8943 \compatglossarystyle{long}{%
8944 \renewcommand*{\glossaryentryfield}[5]{%
8945 \glentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
8946 \renewcommand*{\glossarysubentryfield}[6]{%
8947 &
8948 \glssubentryitem{##2}%
8949 \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
8950 }%

```

Backward compatible longborder style.

```

8951 \compatglossarystyle{longborder}{%
8952 \csuse{@glscmpstyle@long}%
8953 }%

```

Backward compatible longheader style.

```

8954 \compatglossarystyle{longheader}{%
8955 \csuse{@glscmpstyle@long}%
8956 }%

```

Backward compatible longheaderborder style.

```

8957 \compatglossarystyle{longheaderborder}{%
8958 \csuse{@glscmpstyle@long}%
8959 }%

```

Backward compatible long3col style.

```

8960 \compatglossarystyle{long3col}{%
8961 \renewcommand*{\glossaryentryfield}[5]{%
8962 \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\\}%
8963 \renewcommand*{\glossarysubentryfield}[6]{%
8964 &
8965 \glssubentryitem{##2}%
8966 \glstarget{##2}{\strut}##4 & ##6\\}%
8967 }%

```

Backward compatible long3colborder style.

```

8968 \compatglossarystyle{long3colborder}{%
8969 \csuse{@glscmpstyle@long3col}%
8970 }%

```

Backward compatible long3colheader style.

```

8971 \compatglossarystyle{long3colheader}{%
8972 \csuse{@glscmpstyle@long3col}%
8973 }%

```

Backward compatible long3colheaderborder style.

```

8974 \compatglossarystyle{long3colheaderborder}{%
8975 \csuse{@glscmpstyle@long3col}%
8976 }%

```

Backward compatible long4col style.

```
8977 \compatglossarystyle{long4col}{%
8978   \renewcommand*{\glossaryentryfield}[5]{%
8979     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%
8980   \renewcommand*{\glossarysubentryfield}[6]{%
8981     &
8982     \glssubentryitem{##2}%
8983     \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%
8984 }%
```

Backward compatible long4colheader style.

```
8985 \compatglossarystyle{long4colheader}{%
8986   \csuse{@glscmpstyle@long4col}%
8987 }%
```

Backward compatible long4colborder style.

```
8988 \compatglossarystyle{long4colborder}{%
8989   \csuse{@glscmpstyle@long4col}%
8990 }%
```

Backward compatible long4colheaderborder style.

```
8991 \compatglossarystyle{long4colheaderborder}{%
8992   \csuse{@glscmpstyle@long4col}%
8993 }%
```

Backward compatible altlong4col style.

```
8994 \compatglossarystyle{altlong4col}{%
8995   \csuse{@glscmpstyle@long4col}%
8996 }%
```

Backward compatible altlong4colheader style.

```
8997 \compatglossarystyle{altlong4colheader}{%
8998   \csuse{@glscmpstyle@long4col}%
8999 }%
```

Backward compatible altlong4colborder style.

```
9000 \compatglossarystyle{altlong4colborder}{%
9001   \csuse{@glscmpstyle@long4col}%
9002 }%
```

Backward compatible altlong4colheaderborder style.

```
9003 \compatglossarystyle{altlong4colheaderborder}{%
9004   \csuse{@glscmpstyle@long4col}%
9005 }%
```

Backward compatible long style.

```
9006 \compatglossarystyle{longragged}{%
9007   \renewcommand*{\glossaryentryfield}[5]{%
9008     \glentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
9009     \tabularnewline}%
9010   \renewcommand*{\glossarysubentryfield}[6]{%
9011     &
```

```

9012     \glssubentryitem{##2}%
9013     \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
9014     \tabularnewline}%
9015 }%

```

Backward compatible longraggedborder style.

```

9016 \compatglossarystyle{longraggedborder}{%
9017 \csuse{@glscmpstyle@longragged}%
9018 }%

```

Backward compatible longraggedheader style.

```

9019 \compatglossarystyle{longraggedheader}{%
9020 \csuse{@glscmpstyle@longragged}%
9021 }%

```

Backward compatible longraggedheaderborder style.

```

9022 \compatglossarystyle{longraggedheaderborder}{%
9023 \csuse{@glscmpstyle@longragged}%
9024 }%

```

Backward compatible longragged3col style.

```

9025 \compatglossarystyle{longragged3col}{%
9026 \renewcommand*{\glossaryentryfield}[5]{%
9027     \glssubentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%
9028 \renewcommand*{\glossarysubentryfield}[6]{%
9029     &
9030     \glssubentryitem{##2}%
9031     \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%
9032 }%

```

Backward compatible longragged3colborder style.

```

9033 \compatglossarystyle{longragged3colborder}{%
9034 \csuse{@glscmpstyle@longragged3col}%
9035 }%

```

Backward compatible longragged3colheader style.

```

9036 \compatglossarystyle{longragged3colheader}{%
9037 \csuse{@glscmpstyle@longragged3col}%
9038 }%

```

Backward compatible longragged3colheaderborder style.

```

9039 \compatglossarystyle{longragged3colheaderborder}{%
9040 \csuse{@glscmpstyle@longragged3col}%
9041 }%

```

Backward compatible altlongragged4col style.

```

9042 \compatglossarystyle{altlongragged4col}{%
9043 \renewcommand*{\glossaryentryfield}[5]{%
9044     \glssubentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%
9045 \renewcommand*{\glossarysubentryfield}[6]{%
9046     &
9047     \glssubentryitem{##2}%

```

```

9048 \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%
9049 }%

```

Backward compatible altlongragged4colheader style.

```

9050 \compatglossarystyle{altlongragged4colheader}{%
9051 \csuse{@glscmpstyle@altlong4col}%
9052 }%

```

Backward compatible altlongragged4colborder style.

```

9053 \compatglossarystyle{altlongragged4colborder}{%
9054 \csuse{@glscmpstyle@altlong4col}%
9055 }%

```

Backward compatible altlongragged4colheaderborder style.

```

9056 \compatglossarystyle{altlongragged4colheaderborder}{%
9057 \csuse{@glscmpstyle@altlong4col}%
9058 }%

```

Backward compatible index style.

```

9059 \compatglossarystyle{index}{%
9060 \renewcommand*{\glossaryentryfield}[5]{%
9061 \item\glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
9062 \ifx\relax##4\relax
9063 \else
9064 \space(##4)%
9065 \fi
9066 \space ##3\glspostdescription \space ##5}%
9067 \renewcommand*{\glossarysubentryfield}[6]{%
9068 \ifcase##1\relax
9069 % level 0
9070 \item
9071 \or
9072 % level 1
9073 \subitem
9074 \glssubentryitem{##2}%
9075 \else
9076 % all other levels
9077 \subsubitem
9078 \fi
9079 \textbf{\glstarget{##2}{##3}}%
9080 \ifx\relax##5\relax
9081 \else
9082 \space(##5)%
9083 \fi
9084 \space##4\glspostdescription\space ##6}%
9085 }%

```

Backward compatible indexgroup style.

```

9086 \compatglossarystyle{indexgroup}{%
9087 \csuse{@glscmpstyle@index}%
9088 }%

```


Backward compatible indexhypergroup style.

```
9089 \compatglossarystyle{indexhypergroup}{%
9090 \csuse{@glscmpstyle@index}%
9091 }%
```

Backward compatible tree style.

```
9092 \compatglossarystyle{tree}{%
9093 \renewcommand{\glossaryentryfield}[5]{%
9094 \hangindent0pt\relax
9095 \parindent0pt\relax
9096 \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
9097 \ifx\relax##4\relax
9098 \else
9099 \space{##4}%
9100 \fi
9101 \space ##3\glspostdescription \space ##5\par}%
9102 \renewcommand{\glossarysubentryfield}[6]{%
9103 \hangindent##1\glstreeindent\relax
9104 \parindent##1\glstreeindent\relax
9105 \ifnum##1=1\relax
9106 \glssubentryitem{##2}%
9107 \fi
9108 \textbf{\glstarget{##2}{##3}}%
9109 \ifx\relax##5\relax
9110 \else
9111 \space{##5}%
9112 \fi
9113 \space##4\glspostdescription\space ##6\par}%
9114 }%
```

Backward compatible treegroup style.

```
9115 \compatglossarystyle{treegroup}{%
9116 \csuse{@glscmpstyle@tree}%
9117 }%
```

Backward compatible treehypergroup style.

```
9118 \compatglossarystyle{treehypergroup}{%
9119 \csuse{@glscmpstyle@tree}%
9120 }%
```

Backward compatible treenoname style.

```
9121 \compatglossarystyle{treenoname}{%
9122 \renewcommand{\glossaryentryfield}[5]{%
9123 \hangindent0pt\relax
9124 \parindent0pt\relax
9125 \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
9126 \ifx\relax##4\relax
9127 \else
9128 \space{##4}%
9129 \fi
9130 \space ##3\glspostdescription \space ##5\par}%
9131 }
```

```

9131 \renewcommand{\glossarysubentryfield}[6]{%
9132   \hangindent##1\glstreeindent\relax
9133   \parindent##1\glstreeindent\relax
9134   \ifnum##1=1\relax
9135     \glssubentryitem{##2}%
9136   \fi
9137   \glstarget{##2}{\strut}%
9138   ##4\glspostdescription\space ##6\par}%
9139 }%

```

Backward compatible treenonamegroup style.

```

9140 \compatglossarystyle{treenonamegroup}{%
9141   \csuse{@glscmpstyle@treenoname}%
9142 }%

```

Backward compatible treenonamehypergroup style.

```

9143 \compatglossarystyle{treenonamehypergroup}{%
9144   \csuse{@glscmpstyle@treenoname}%
9145 }%

```

Backward compatible alttree style.

```

9146 \compatglossarystyle{alttree}{%
9147   \renewcommand{\glossaryentryfield}[5]{%
9148     \ifnum \@gls@prevlevel=0\relax
9149     \else
9150       \settowidth{\glstreeindent}{\textbf{\@glswidestname\space}}%
9151       \hangindent\glstreeindent
9152       \parindent\glstreeindent
9153     \fi
9154     \makebox[0pt][r]{\makebox[\glstreeindent][l]{%
9155       \glssubentryitem{##1}\textbf{\glstarget{##1}{##2}}}%
9156     \ifx\relax##4\relax
9157     \else
9158       (##4)\space
9159     \fi
9160     ##3\glspostdescription\space ##5\par
9161     \def\@gls@prevlevel{0}%
9162   }%
9163   \renewcommand{\glossarysubentryfield}[6]{%
9164     \ifnum##1=1\relax
9165       \glssubentryitem{##2}%
9166     \fi
9167     \ifnum \@gls@prevlevel=##1\relax
9168     \else
9169       \@ifundefined{@glswidestname\romannumeral##1}{%
9170         \settowidth{\gls@tmplen}{\textbf{\@glswidestname\space}}{%
9171         \settowidth{\gls@tmplen}{\textbf{%
9172           \csname @glswidestname\romannumeral##1\endcsname\space}}}%
9173       \ifnum \@gls@prevlevel<##1\relax
9174         \setlength\glstreeindent\gls@tmplen
9175         \addtolength\glstreeindent\parindent

```

```

9176         \parindent\glstreeindent
9177     \else
9178         \@ifundefined{@glswidestname\romannumeral\@gls@prevlevel}{%
9179             \settowidth{\glstreeindent}{\textbf{%
9180                 \@glswidestname\space}}}{%
9181                 \settowidth{\glstreeindent}{\textbf{%
9182                     \csname @glswidestname\romannumeral\@gls@prevlevel
9183                         \endcsname\space}}}{%
9184                     \addtolength\parindent{-\glstreeindent}}%
9185                 \setlength\glstreeindent\parindent
9186             \fi
9187         \fi
9188         \hangindent\glstreeindent
9189         \makebox[0pt][r]{\makebox[\@gls@tmplen][l]{%
9190             \textbf{\glstarget{##2}{##3}}}}%
9191         \ifx##5\relax\relax
9192         \else
9193             (##5)\space
9194         \fi
9195         ##4\glspostdescription\space ##6\par
9196         \def\@gls@prevlevel{##1}%
9197     }%
9198 }%

```

Backward compatible alttreegroup style.

```

9199 \compatglossarystyle{alttreegroup}{%
9200 \csuse{@glscompstyle@alttree}%
9201 }%

```

Backward compatible alttreehypergroup style.

```

9202 \compatglossarystyle{alttreehypergroup}{%
9203 \csuse{@glscompstyle@alttree}%
9204 }%

```

Backward compatible mcolindex style.

```

9205 \compatglossarystyle{mcolindex}{%
9206 \csuse{@glscompstyle@index}%
9207 }%

```

Backward compatible mcolindexgroup style.

```

9208 \compatglossarystyle{mcolindexgroup}{%
9209 \csuse{@glscompstyle@index}%
9210 }%

```

Backward compatible mcolindexhypergroup style.

```

9211 \compatglossarystyle{mcolindexhypergroup}{%
9212 \csuse{@glscompstyle@index}%
9213 }%

```

Backward compatible mcoltree style.

```

9214 \compatglossarystyle{mcoltree}{%
9215 \csuse{@glscompstyle@tree}%

```

9216 }%

Backward compatible mcoltreegroup style.

9217 \compatglossarystyle{mcolindextreegroup}{%

9218 \csuse{@glscompstyle@tree}%

9219 }%

Backward compatible mcoltreehypergroup style.

9220 \compatglossarystyle{mcolindextreehypergroup}{%

9221 \csuse{@glscompstyle@tree}%

9222 }%

Backward compatible mcoltreenoname style.

9223 \compatglossarystyle{mcoltreenoname}{%

9224 \csuse{@glscompstyle@tree}%

9225 }%

Backward compatible mcoltreenonamegroup style.

9226 \compatglossarystyle{mcoltreenonamegroup}{%

9227 \csuse{@glscompstyle@tree}%

9228 }%

Backward compatible mcoltreenonamehypergroup style.

9229 \compatglossarystyle{mcoltreenonamehypergroup}{%

9230 \csuse{@glscompstyle@tree}%

9231 }%

Backward compatible mcolalmtree style.

9232 \compatglossarystyle{mcolalmtree}{%

9233 \csuse{@glscompstyle@almtree}%

9234 }%

Backward compatible mcolalmtreegroup style.

9235 \compatglossarystyle{mcolalmtreegroup}{%

9236 \csuse{@glscompstyle@almtree}%

9237 }%

Backward compatible mcolalmtreehypergroup style.

9238 \compatglossarystyle{mcolalmtreehypergroup}{%

9239 \csuse{@glscompstyle@almtree}%

9240 }%

Backward compatible superragged style.

9241 \compatglossarystyle{superragged}{%

9242 \renewcommand*{\glossaryentryfield}[5]{%

9243 \glsentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5%

9244 \tabularnewline}%

9245 \renewcommand*{\glossarysubentryfield}[6]{%

9246 &

9247 \glssubentryitem{##2}%

9248 \glstarget{##2}{\strut}##4\glspostdescription\space ##6%

9249 \tabularnewline}%

9250 }%

Backward compatible superraggedborder style.

```
9251 \compatglossarystyle{superraggedborder}{%  
9252 \csuse{@glscmpstyle@superragged}%  
9253 }%
```

Backward compatible superraggedheader style.

```
9254 \compatglossarystyle{superraggedheader}{%  
9255 \csuse{@glscmpstyle@superragged}%  
9256 }%
```

Backward compatible superraggedheaderborder style.

```
9257 \compatglossarystyle{superraggedheaderborder}{%  
9258 \csuse{@glscmpstyle@superragged}%  
9259 }%
```

Backward compatible superragged3col style.

```
9260 \compatglossarystyle{superragged3col}{%  
9261 \renewcommand*{\glossaryentryfield}[5]{%  
9262 \glstentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%  
9263 \renewcommand*{\glossarysubentryfield}[6]{%  
9264 &  
9265 \glssubentryitem{##2}%  
9266 \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%  
9267 }%
```

Backward compatible superragged3colborder style.

```
9268 \compatglossarystyle{superragged3colborder}{%  
9269 \csuse{@glscmpstyle@superragged3col}%  
9270 }%
```

Backward compatible superragged3colheader style.

```
9271 \compatglossarystyle{superragged3colheader}{%  
9272 \csuse{@glscmpstyle@superragged3col}%  
9273 }%
```

Backward compatible superragged3colheaderborder style.

```
9274 \compatglossarystyle{superragged3colheaderborder}{%  
9275 \csuse{@glscmpstyle@superragged3col}%  
9276 }%
```

Backward compatible altsuperragged4col style.

```
9277 \compatglossarystyle{altsuperragged4col}{%  
9278 \renewcommand*{\glossaryentryfield}[5]{%  
9279 \glstentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%  
9280 \renewcommand*{\glossarysubentryfield}[6]{%  
9281 &  
9282 \glssubentryitem{##2}%  
9283 \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%  
9284 }%
```

Backward compatible altsuperragged4colheader style.

```
9285 \compatglossarystyle{altsuperragged4colheader}{%
```

```

9286 \csuse{@glscompstyle@altsuperragged4col}%
9287 }%

```

Backward compatible altsuperragged4colborder style.

```

9288 \compatglossarystyle{altsuperragged4colborder}{%
9289 \csuse{@glscompstyle@altsuperragged4col}%
9290 }%

```

Backward compatible altsuperragged4colheaderborder style.

```

9291 \compatglossarystyle{altsuperragged4colheaderborder}{%
9292 \csuse{@glscompstyle@altsuperragged4col}%
9293 }%

```

Backward compatible super style.

```

9294 \compatglossarystyle{super}{%
9295 \renewcommand*{\glossaryentryfield}[5]{%
9296 \glstarget{##1}{\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
9297 \renewcommand*{\glossarysubentryfield}[6]{%
9298 &
9299 \glssubentryitem{##2}%
9300 \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
9301 }%

```

Backward compatible superborder style.

```

9302 \compatglossarystyle{superborder}{%
9303 \csuse{@glscompstyle@super}%
9304 }%

```

Backward compatible superheader style.

```

9305 \compatglossarystyle{superheader}{%
9306 \csuse{@glscompstyle@super}%
9307 }%

```

Backward compatible superheaderborder style.

```

9308 \compatglossarystyle{superheaderborder}{%
9309 \csuse{@glscompstyle@super}%
9310 }%

```

Backward compatible super3col style.

```

9311 \compatglossarystyle{super3col}{%
9312 \renewcommand*{\glossaryentryfield}[5]{%
9313 \glstarget{##1}{\glstarget{##1}{##2} & ##3 & ##5\\}%
9314 \renewcommand*{\glossarysubentryfield}[6]{%
9315 &
9316 \glssubentryitem{##2}%
9317 \glstarget{##2}{\strut}##4 & ##6\\}%
9318 }%

```

Backward compatible super3colborder style.

```

9319 \compatglossarystyle{super3colborder}{%
9320 \csuse{@glscompstyle@super3col}%
9321 }%

```

Backward compatible super3colheader style.

```
9322 \compatglossarystyle{super3colheader}{%  
9323 \csuse{@glscmpstyle@super3col}%  
9324 }%
```

Backward compatible super3colheaderborder style.

```
9325 \compatglossarystyle{super3colheaderborder}{%  
9326 \csuse{@glscmpstyle@super3col}%  
9327 }%
```

Backward compatible super4col style.

```
9328 \compatglossarystyle{super4col}{%  
9329 \renewcommand*{\glossaryentryfield}[5]{%  
9330 \glstryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%  
9331 \renewcommand*{\glossarysubentryfield}[6]{%  
9332 &  
9333 \glssubentryitem{##2}%  
9334 \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%  
9335 }%
```

Backward compatible super4colheader style.

```
9336 \compatglossarystyle{super4colheader}{%  
9337 \csuse{@glscmpstyle@super4col}%  
9338 }%
```

Backward compatible super4colborder style.

```
9339 \compatglossarystyle{super4colborder}{%  
9340 \csuse{@glscmpstyle@super4col}%  
9341 }%
```

Backward compatible super4colheaderborder style.

```
9342 \compatglossarystyle{super4colheaderborder}{%  
9343 \csuse{@glscmpstyle@super4col}%  
9344 }%
```

Backward compatible altsuper4col style.

```
9345 \compatglossarystyle{altsuper4col}{%  
9346 \csuse{@glscmpstyle@super4col}%  
9347 }%
```

Backward compatible altsuper4colheader style.

```
9348 \compatglossarystyle{altsuper4colheader}{%  
9349 \csuse{@glscmpstyle@super4col}%  
9350 }%
```

Backward compatible altsuper4colborder style.

```
9351 \compatglossarystyle{altsuper4colborder}{%  
9352 \csuse{@glscmpstyle@super4col}%  
9353 }%
```

Backward compatible altsuper4colheaderborder style.

```
9354 \compatglossarystyle{altsuper4colheaderborder}{%  
9355 \csuse{@glscmpstyle@super4col}%  
9356 }%
```

7 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.

```
9357 \NeedsTeXFormat{LaTeX2e}
```

Package version number now in line with main glossaries package number but will only be updated when glossaries-accsupp.sty is modified.

```
9358 \ProvidesPackage{glossaries-accsupp}[2014/07/30 v4.08 (NLCT)]
```

```
9359 Experimental glossaries accessibility
```

Pass all options to glossaries:

```
9360 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
9361 \ProcessOptions
```

compatibleglossentry Override style compatibility macros:

```
9362 \def\compatibleglossentry#1#2{%
9363   \toks@{#2}%
9364   \protected@edef\@do@glossentry{%
9365     \noexpand\accsuppglossaryentryfield{#1}%
9366     {\noexpand\glsnamefont
9367       {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@name\endcsname}}}%
9368     {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@desc\endcsname}}%
9369     {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@symbol\endcsname}}%
9370     {\the\toks@}%
9371   }%
9372   \@do@glossentry
9373 }
```

compatiblesubglossentry

```
9374 \def\compatiblesubglossentry#1#2#3{%
9375   \toks@{#3}%
9376   \protected@edef\@do@subglossentry{%
9377     \noexpand\accsuppglossarysubentryfield{\number#1}%
9378     {#2}%
9379     {\noexpand\glsnamefont
9380       {\expandafter\expandonce\csname glo@\glsdetoklabel{#2}@name\endcsname}}}%
9381     {\expandafter\expandonce\csname glo@\glsdetoklabel{#2}@desc\endcsname}}%
9382     {\expandafter\expandonce\csname glo@\glsdetoklabel{#2}@symbol\endcsname}}%
9383     {\the\toks@}%
9384   }%
9385   \@do@subglossentry
9386 }
```

Required packages:

```
9387 \RequirePackage{glossaries}
```

```
9388 \RequirePackage{accsupp}
```


7.1 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:

```
9389 \define@key{glossentry}{access}{%
9390   \def\@glo@access{#1}%
9391 }
```

textaccess The replacement text corresponding to the text key:

```
9392 \define@key{glossentry}{textaccess}{%
9393   \def\@glo@textaccess{#1}%
9394 }
```

firstaccess The replacement text corresponding to the first key:

```
9395 \define@key{glossentry}{firstaccess}{%
9396   \def\@glo@firstaccess{#1}%
9397 }
```

pluralaccess The replacement text corresponding to the plural key:

```
9398 \define@key{glossentry}{pluralaccess}{%
9399   \def\@glo@pluralaccess{#1}%
9400 }
```

firstpluralaccess The replacement text corresponding to the firstplural key:

```
9401 \define@key{glossentry}{firstpluralaccess}{%
9402   \def\@glo@firstpluralaccess{#1}%
9403 }
```

symbolaccess The replacement text corresponding to the symbol key:

```
9404 \define@key{glossentry}{symbolaccess}{%
9405   \def\@glo@symbolaccess{#1}%
9406 }
```

symbolpluralaccess The replacement text corresponding to the symbolplural key:

```
9407 \define@key{glossentry}{symbolpluralaccess}{%
9408   \def\@glo@symbolpluralaccess{#1}%
9409 }
```

descriptionaccess The replacement text corresponding to the description key:

```
9410 \define@key{glossentry}{descriptionaccess}{%
9411   \def\@glo@descaccess{#1}%
9412 }
```

descriptionpluralaccess The replacement text corresponding to the descriptionplural key:

```
9413 \define@key{glossentry}{descriptionpluralaccess}{%
9414   \def\@glo@descpluralaccess{#1}%
9415 }
```

shortaccess The replacement text corresponding to the short key:

```
9416 \define@key{glossentry}{shortaccess}{%
9417   \def\@glo@shortaccess{#1}%
9418 }
```

shortpluralaccess The replacement text corresponding to the shortplural key:

```
9419 \define@key{glossentry}{shortpluralaccess}{%
9420   \def\@glo@shortpluralaccess{#1}%
9421 }
```

longaccess The replacement text corresponding to the long key:

```
9422 \define@key{glossentry}{longaccess}{%
9423   \def\@glo@longaccess{#1}%
9424 }
```

longpluralaccess The replacement text corresponding to the longplural key:

```
9425 \define@key{glossentry}{longpluralaccess}{%
9426   \def\@glo@longpluralaccess{#1}%
9427 }
```

There are no equivalent keys for the user1...user6 keys. The replacement text would have to be explicitly put in the value, e.g., user1={\glsaccsupp{inches}{in}}.

Append these new keys to \@gls@keymap:

```
9428 \appto\@gls@keymap{,%
9429   {access}{access},%
9430   {textaccess}{textaccess},%
9431   {firstaccess}{firstaccess},%
9432   {pluralaccess}{pluralaccess},%
9433   {firstpluralaccess}{firstpluralaccess},%
9434   {symbolaccess}{symbolaccess},%
9435   {symbolpluralaccess}{symbolpluralaccess},%
9436   {descaccess}{descaccess},%
9437   {descpluralaccess}{descpluralaccess},%
9438   {shortaccess}{shortaccess},%
9439   {shortpluralaccess}{shortpluralaccess},%
9440   {longaccess}{longaccess},%
9441   {longpluralaccess}{longpluralaccess}%
9442 }
```

\@gls@noaccess Indicates that no replacement text has been provided.

```
9443 \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):

```

9444 \let\@gls@oldnewglossaryentryprehook\@newglossaryentryprehook
9445 \renewcommand*{\@newglossaryentryprehook}{%
9446   \@gls@oldnewglossaryentryprehook
9447   \def\@glo@access{\@glo@symbol}%

```

Initialise the other keys:

```

9448   \def\@glo@textaccess{\@glo@access}%
9449   \def\@glo@firstaccess{\@glo@access}%
9450   \def\@glo@pluralaccess{\@glo@textaccess}%
9451   \def\@glo@firstpluralaccess{\@glo@pluralaccess}%
9452   \def\@glo@symbolaccess{\relax}%
9453   \def\@glo@symbolpluralaccess{\@glo@symbolaccess}%
9454   \def\@glo@descaccess{\relax}%
9455   \def\@glo@descpluralaccess{\@glo@descaccess}%
9456   \def\@glo@shortaccess{\relax}%
9457   \def\@glo@shortpluralaccess{\@glo@shortaccess}%
9458   \def\@glo@longaccess{\relax}%
9459   \def\@glo@longpluralaccess{\@glo@longaccess}%
9460 }

```

Add to the end hook:

```

9461 \let\@gls@oldnewglossaryentryposthook\@newglossaryentryposthook
9462 \renewcommand*{\@newglossaryentryposthook}{%
9463   \@gls@oldnewglossaryentryposthook

```

Store the access information:

```

9464   \expandafter
9465     \protected@xdef\csname glo@\@glo@label @access\endcsname{%
9466       \@glo@access}%
9467   \expandafter
9468     \protected@xdef\csname glo@\@glo@label @textaccess\endcsname{%
9469       \@glo@textaccess}%
9470   \expandafter
9471     \protected@xdef\csname glo@\@glo@label @firstaccess\endcsname{%
9472       \@glo@firstaccess}%
9473   \expandafter
9474     \protected@xdef\csname glo@\@glo@label @pluralaccess\endcsname{%
9475       \@glo@pluralaccess}%
9476   \expandafter
9477     \protected@xdef\csname glo@\@glo@label @firstpluralaccess\endcsname{%
9478       \@glo@firstpluralaccess}%
9479   \expandafter
9480     \protected@xdef\csname glo@\@glo@label @symbolaccess\endcsname{%
9481       \@glo@symbolaccess}%
9482   \expandafter
9483     \protected@xdef\csname glo@\@glo@label @symbolpluralaccess\endcsname{%
9484       \@glo@symbolpluralaccess}%
9485   \expandafter

```

```

9486 \protected@xdef\csname glo@\@glo@label @descaccess\endcsname{%
9487 \@glo@descaccess}%
9488 \expandafter
9489 \protected@xdef\csname glo@\@glo@label @descpluralaccess\endcsname{%
9490 \@glo@descpluralaccess}%
9491 \expandafter
9492 \protected@xdef\csname glo@\@glo@label @shortaccess\endcsname{%
9493 \@glo@shortaccess}%
9494 \expandafter
9495 \protected@xdef\csname glo@\@glo@label @shortpluralaccess\endcsname{%
9496 \@glo@shortpluralaccess}%
9497 \expandafter
9498 \protected@xdef\csname glo@\@glo@label @longaccess\endcsname{%
9499 \@glo@longaccess}%
9500 \expandafter
9501 \protected@xdef\csname glo@\@glo@label @longpluralaccess\endcsname{%
9502 \@glo@longpluralaccess}%
9503 }

```

7.2 Accessing Replacement Text

`\glsentryaccess` Get the value of the access key for the entry with the given label:

```

9504 \newcommand*{\glsentryaccess}[1]{%
9505 \@gls@entry@field{#1}{access}%
9506 }

```

`\glsentrytextaccess` Get the value of the textaccess key for the entry with the given label:

```

9507 \newcommand*{\glsentrytextaccess}[1]{%
9508 \@gls@entry@field{#1}{textaccess}%
9509 }

```

`\glsentryfirstaccess` Get the value of the firstaccess key for the entry with the given label:

```

9510 \newcommand*{\glsentryfirstaccess}[1]{%
9511 \@gls@entry@field{#1}{firstaccess}%
9512 }

```

`\glsentrypluralaccess` Get the value of the pluralaccess key for the entry with the given label:

```

9513 \newcommand*{\glsentrypluralaccess}[1]{%
9514 \@gls@entry@field{#1}{pluralaccess}%
9515 }

```

`\glsentryfirstpluralaccess` Get the value of the firstpluralaccess key for the entry with the given label:

```

9516 \newcommand*{\glsentryfirstpluralaccess}[1]{%
9517 \csname glo@#1@firstpluralaccess\endcsname
9518 }

```

`\glsentrysymbolaccess` Get the value of the symbolaccess key for the entry with the given label:

```

9519 \newcommand*{\glsentrysymbolaccess}[1]{%

```

```

9520 \@gls@entry@field{#1}{symbolaccess}%
9521 }

symbolpluralaccess Get the value of the symbolpluralaccess key for the entry with the given label:
9522 \newcommand*{\glsentrysymbolpluralaccess}[1]{%
9523 \@gls@entry@field{#1}{symbolpluralaccess}%
9524 }

\glsentrydescaccess Get the value of the descriptionaccess key for the entry with the given label:
9525 \newcommand*{\glsentrydescaccess}[1]{%
9526 \@gls@entry@field{#1}{descaccess}%
9527 }

entrydescpluralaccess Get the value of the descriptionpluralaccess key for the entry with the given label:
9528 \newcommand*{\glsentrydescpluralaccess}[1]{%
9529 \@gls@entry@field{#1}{descaccess}%
9530 }

\glsentryshortaccess Get the value of the shortaccess key for the entry with the given label:
9531 \newcommand*{\glsentryshortaccess}[1]{%
9532 \@gls@entry@field{#1}{shortaccess}%
9533 }

entryshortpluralaccess Get the value of the shortpluralaccess key for the entry with the given label:
9534 \newcommand*{\glsentryshortpluralaccess}[1]{%
9535 \@gls@entry@field{#1}{shortpluralaccess}%
9536 }

\glsentrylongaccess Get the value of the longaccess key for the entry with the given label:
9537 \newcommand*{\glsentrylongaccess}[1]{%
9538 \@gls@entry@field{#1}{longaccess}%
9539 }

entrylongpluralaccess Get the value of the longpluralaccess key for the entry with the given label:
9540 \newcommand*{\glsentrylongpluralaccess}[1]{%
9541 \@gls@entry@field{#1}{longpluralaccess}%
9542 }

\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.)
9543 \newcommand*{\glsaccsupp}[2]{%
9544 \BeginAccSupp{ActualText=#1}#2\EndAccSupp{}%
9545 }

```

`\xglsaccsupp` Fully expands replacement text before calling `\glsaccsupp`

```

9546 \newcommand*\xglsaccsupp}[2]{%
9547   \protected@edef\xgls@replacementtext{#1}%
9548   \expandafter\glsaccsupp\expandafter{\xgls@replacementtext}{#2}%
9549 }

```

`\@gls@access@display`

```

9550 \newcommand*\@gls@access@display}[2]{%
9551   \protected@edef\@glo@access{#2}%
9552   \ifx\@glo@access\xgls@noaccess
9553     #1%
9554   \else
9555     \xglsaccsupp{\@glo@access}{#1}%
9556   \fi
9557 }

```

`\@glsname@access@display` Displays the first argument with the accessibility text for the entry with the label given by the second argument (if set).

```

9558 \DeclareRobustCommand*\@glsname@access@display}[2]{%
9559   \@gls@access@display{#1}{\glsentryaccess{#2}}%
9560 }

```

`\@gls@text@access@display` As above but for the `\textaccess` replacement text.

```

9561 \DeclareRobustCommand*\@gls@text@access@display}[2]{%
9562   \@gls@access@display{#1}{\glsentrytextaccess{#2}}%
9563 }

```

`\@gls@plural@access@display` As above but for the `\pluralaccess` replacement text.

```

9564 \DeclareRobustCommand*\@gls@plural@access@display}[2]{%
9565   \@gls@access@display{#1}{\glsentrypluralaccess{#2}}%
9566 }

```

`\@gls@first@access@display` As above but for the `\firstaccess` replacement text.

```

9567 \DeclareRobustCommand*\@gls@first@access@display}[2]{%
9568   \@gls@access@display{#1}{\glsentryfirstaccess{#2}}%
9569 }

```

`\@gls@first@plural@access@display` As above but for the `\firstpluralaccess` replacement text.

```

9570 \DeclareRobustCommand*\@gls@first@plural@access@display}[2]{%
9571   \@gls@access@display{#1}{\glsentryfirstpluralaccess{#2}}%
9572 }

```

`\@gls@symbol@access@display` As above but for the `\symbolaccess` replacement text.

```

9573 \DeclareRobustCommand*\@gls@symbol@access@display}[2]{%
9574   \@gls@access@display{#1}{\glsentrysymbolaccess{#2}}%
9575 }

```

pluralaccessdisplay As above but for the symbolpluralaccess replacement text.

```

9576 \DeclareRobustCommand*\glssymbolpluralaccessdisplay}[2]{%
9577   \@gls@access@display{#1}{\glsentrysymbolpluralaccess{#2}}}%
9578 }

```

descriptionaccessdisplay As above but for the descriptionaccess replacement text.

```

9579 \DeclareRobustCommand*\glsdescriptionaccessdisplay}[2]{%
9580   \@gls@access@display{#1}{\glsentrydescaccess{#2}}}%
9581 }

```

descriptionpluralaccessdisplay As above but for the descriptionpluralaccess replacement text.

```

9582 \DeclareRobustCommand*\glsdescriptionpluralaccessdisplay}[2]{%
9583   \@gls@access@display{#1}{\glsentrydescpluralaccess{#2}}}%
9584 }

```

shortaccessdisplay As above but for the shortaccess replacement text.

```

9585 \DeclareRobustCommand*\glsshortaccessdisplay}[2]{%
9586   \@gls@access@display{#1}{\glsentryshortaccess{#2}}}%
9587 }

```

shortpluralaccessdisplay As above but for the shortpluralaccess replacement text.

```

9588 \DeclareRobustCommand*\glsshortpluralaccessdisplay}[2]{%
9589   \@gls@access@display{#1}{\glsentryshortpluralaccess{#2}}}%
9590 }

```

longaccessdisplay As above but for the longaccess replacement text.

```

9591 \DeclareRobustCommand*\glslongaccessdisplay}[2]{%
9592   \@gls@access@display{#1}{\glsentrylongaccess{#2}}}%
9593 }

```

longpluralaccessdisplay As above but for the longpluralaccess replacement text.

```

9594 \DeclareRobustCommand*\glslongpluralaccessdisplay}[2]{%
9595   \@gls@access@display{#1}{\glsentrylongpluralaccess{#2}}}%
9596 }

```

\glsaccessdisplay Gets the replacement text corresponding to the named key given by the first argument and calls the appropriate command defined above.

```

9597 \DeclareRobustCommand*\glsaccessdisplay}[3]{%
9598   \@ifundefined{gls#1accessdisplay}%
9599   {%
9600     \PackageError{glossaries-accsupp}{No accessibility support
9601       for key ‘#1’}{}%
9602   }%
9603   {%
9604     \csname gls#1accessdisplay\endcsname{#2}{#3}%
9605   }%
9606 }

```

ls@default@entryfmt Redefine the default entry format to use accessibility information

```
9607 \renewcommand*{\@@gls@default@entryfmt}[2]{%
9608   \ifdefempty\glscustomtext
9609   {%
9610     \glsifplural
9611     {%
```

Plural form

```
9612     \glscapscase
9613     {%
```

Don't adjust case

```
9614     \ifglsused\glslabel
9615     {%
```

Subsequent use

```
9616         #2{\glspluralaccessdisplay
9617             {\glsentryplural{\glslabel}}{\glslabel}}%
9618         {\glsdescriptionpluralaccessdisplay
9619             {\glsentrydescplural{\glslabel}}{\glslabel}}%
9620         {\glsymbolpluralaccessdisplay
9621             {\glsentrysymbolplural{\glslabel}}{\glslabel}}
9622         {\glsinsert}}%
9623     }%
9624     {%
```

First use

```
9625         #1{\glsfirstpluralaccessdisplay
9626             {\glsentryfirstplural{\glslabel}}{\glslabel}}%
9627         {\glsdescriptionpluralaccessdisplay
9628             {\glsentrydescplural{\glslabel}}{\glslabel}}%
9629         {\glsymbolpluralaccessdisplay
9630             {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
9631         {\glsinsert}}%
9632     }%
9633     }%
9634     {%
```

Make first letter upper case

```
9635     \ifglsused\glslabel
9636     {%
```

Subsequent use.

```
9637         #2{\glspluralaccessdisplay
9638             {\Glsentryplural{\glslabel}}{\glslabel}}%
9639         {\glsdescriptionpluralaccessdisplay
9640             {\glsentrydescplural{\glslabel}}{\glslabel}}%
9641         {\glsymbolpluralaccessdisplay
9642             {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
9643         {\glsinsert}}%
9644     }%
9645     {%
```


First use

```
9646      #1{\glsfirstpluralaccessdisplay
9647          {\Glsentryfirstplural{\glslabel}}{\glslabel}}%
9648          {\glsdescriptionpluralaccessdisplay
9649              {\Glsentrydescplural{\glslabel}}{\glslabel}}%
9650          {\glssymbolpluralaccessdisplay
9651              {\Glsentrysymbolplural{\glslabel}}{\glslabel}}%
9652          {\glsinsert}}%
9653      }%
9654  }%
9655  {%
```

Make all upper case

```
9656      \ifglsused\glslabel
9657      {%
```

Subsequent use

```
9658      \MakeUppercase{%
9659          #2{\glspluralaccessdisplay
9660              {\Glsentryplural{\glslabel}}{\glslabel}}%
9661              {\glsdescriptionpluralaccessdisplay
9662                  {\Glsentrydescplural{\glslabel}}{\glslabel}}%
9663                  {\glssymbolpluralaccessdisplay
9664                      {\Glsentrysymbolplural{\glslabel}}{\glslabel}}%
9665                      {\glsinsert}}}%
9666      }%
9667  {%
```

First use

```
9668      \MakeUppercase{%
9669          #1{\glsfirstpluralaccessdisplay
9670              {\Glsentryfirstplural{\glslabel}}{\glslabel}}%
9671              {\glsdescriptionpluralaccessdisplay
9672                  {\Glsentrydescplural{\glslabel}}{\glslabel}}%
9673                  {\glssymbolpluralaccessdisplay
9674                      {\Glsentrysymbolplural{\glslabel}}{\glslabel}}%
9675                      {\glsinsert}}}%
9676      }%
9677  }%
9678  }%
9679  {%
```

Singular form

```
9680      \glscapscase
9681      {%
```

Don't adjust case

```
9682      \ifglsused\glslabel
9683      {%
```

Subsequent use

```

9684      #2{\glstextaccessdisplay
9685          {\glentrytext{\glslabel}}{\glslabel}}%
9686      {\glsdescriptionaccessdisplay
9687          {\glentrydesc{\glslabel}}{\glslabel}}%
9688      {\glssymbolaccessdisplay
9689          {\glentrysymbol{\glslabel}}{\glslabel}}%
9690      {\glsinsert}%
9691      }%
9692      {%

```

First use

```

9693      #1{\glsfirstaccessdisplay
9694          {\glentryfirst{\glslabel}}{\glslabel}}%
9695      {\glsdescriptionaccessdisplay
9696          {\glentrydesc{\glslabel}}{\glslabel}}%
9697      {\glssymbolaccessdisplay
9698          {\glentrysymbol{\glslabel}}{\glslabel}}%
9699      {\glsinsert}%
9700      }%
9701      }%
9702      {%

```

Make first letter upper case

```

9703      \ifglsused\glslabel
9704      {%

```

Subsequent use

```

9705      #2{\glstextaccessdisplay
9706          {\Glsentrytext{\glslabel}}{\glslabel}}%
9707      {\glsdescriptionaccessdisplay
9708          {\glentrydesc{\glslabel}}{\glslabel}}%
9709      {\glssymbolaccessdisplay
9710          {\glentrysymbol{\glslabel}}{\glslabel}}%
9711      {\glsinsert}%
9712      }%
9713      {%

```

First use

```

9714      #1{\glsfirstaccessdisplay
9715          {\Glsentryfirst{\glslabel}}{\glslabel}}%
9716      {\glsdescriptionaccessdisplay
9717          {\glentrydesc{\glslabel}}{\glslabel}}%
9718      {\glssymbolaccessdisplay
9719          {\glentrysymbol{\glslabel}}{\glslabel}}%
9720      {\glsinsert}%
9721      }%
9722      }%
9723      {%

```

Make all upper case

```

9724      \ifglsused\glslabel
9725      {%

```

Subsequent use

```

9726      \MakeUppercase{%
9727          #2{\glstextaccessdisplay
9728              {\glentrytext{\glslabel}}{\glslabel}}%
9729              {\glsdescriptionaccessdisplay
9730                  {\glentrydesc{\glslabel}}{\glslabel}}%
9731                  {\glssymbolaccessdisplay
9732                      {\glentrysymbol{\glslabel}}{\glslabel}}%
9733                      {\glinsert}}}%
9734      }%
9735      {%

```

First use

```

9736      \MakeUppercase{%
9737          #1{\glfirstaccessdisplay
9738              {\glentryfirst{\glslabel}}{\glslabel}}%
9739              {\glsdescriptionaccessdisplay
9740                  {\glentrydesc{\glslabel}}{\glslabel}}%
9741                  {\glssymbolaccessdisplay
9742                      {\glentrysymbol{\glslabel}}{\glslabel}}%
9743                      {\glinsert}}}%
9744      }%
9745      }%
9746      }%
9747      }%
9748      {%

```

Custom text provided in \glldisp

```

9749      \ifglused{\glslabel}%
9750      {%

```

Subsequent use

```

9751      #2{\glscustomtext}%
9752      {\glsdescriptionaccessdisplay
9753          {\glentrydesc{\glslabel}}{\glslabel}}%
9754          {\glssymbolaccessdisplay
9755              {\glentrysymbol{\glslabel}}{\glslabel}}%
9756              {\glinsert}}%
9757      }%
9758      {%

```

First use

```

9759      #1{\glscustomtext}%
9760      {\glsdescriptionaccessdisplay
9761          {\glentrydesc{\glslabel}}{\glslabel}}%
9762          {\glssymbolaccessdisplay
9763              {\glentrysymbol{\glslabel}}{\glslabel}}%
9764              {\glinsert}}%
9765      }%
9766      }%
9767      }

```

`\glsgenentryfmt` Redefine to use accessibility information.

```
9768 \renewcommand*{\glsgenentryfmt}{%
9769   \ifdefempty\glscustomtext
9770   {%
9771     \glssifplural
9772     {%
```

Plural form

```
9773     \glscapscase
9774     {%
```

Don't adjust case

```
9775     \ifglssused\glslabel
9776     {%
```

Subsequent use

```
9777     \glspluralaccessdisplay
9778     {\glssentryplural{\glslabel}}{\glslabel}%
9779     \glssinsert
9780     }%
9781     {%
```

First use

```
9782     \glssfirstpluralaccessdisplay
9783     {\glssentryfirstplural{\glslabel}}{\glslabel}%
9784     \glssinsert
9785     }%
9786     }%
9787     {%
```

Make first letter upper case

```
9788     \ifglssused\glslabel
9789     {%
```

Subsequent use.

```
9790     \glspluralaccessdisplay
9791     {\Glsentryplural{\glslabel}}{\glslabel}%
9792     \glssinsert
9793     }%
9794     {%
```

First use

```
9795     \glssfirstpluralaccessdisplay
9796     {\Glsentryfirstplural{\glslabel}}{\glslabel}%
9797     \glssinsert
9798     }%
9799     }%
9800     {%
```

Make all upper case

```
9801     \ifglssused\glslabel
9802     {%
```

Subsequent use

```

9803      \glspluralaccessdisplay
9804      {\mfirstucMakeUppercase{\glsentryplural{\glslabel}}}%
9805      {\glslabel}%
9806      \mfirstucMakeUppercase{\glsinsert}%
9807  }%
9808  {%

```

First use

```

9809      \glsfirstpluralaccessdisplay
9810      {\mfirstucMakeUppercase{\glsentryfirstplural{\glslabel}}}%
9811      {\glslabel}%
9812      \mfirstucMakeUppercase{\glsinsert}%
9813  }%
9814  }%
9815  }%
9816  {%

```

Singular form

```

9817      \glscapscase
9818      {%

```

Don't adjust case

```

9819      \ifglsused\glslabel
9820      {%

```

Subsequent use

```

9821      \glstextaccessdisplay{\glsentrytext{\glslabel}}{\glslabel}%
9822      \glsinsert
9823  }%
9824  {%

```

First use

```

9825      \glsfirstaccessdisplay{\glsentryfirst{\glslabel}}{\glslabel}%
9826      \glsinsert
9827  }%
9828  }%
9829  {%

```

Make first letter upper case

```

9830      \ifglsused\glslabel
9831      {%

```

Subsequent use

```

9832      \glstextaccessdisplay{\Glsentrytext{\glslabel}}{\glslabel}%
9833      \glsinsert
9834  }%
9835  {%

```

First use

```

9836      \glsfirstaccessdisplay{\Glsentryfirst{\glslabel}}{\glslabel}%
9837      \glsinsert

```

```

9838     }%
9839     }%
9840     {%

```

Make all upper case

```

9841     \ifglused\glslabel
9842     {%

```

Subsequent use

```

9843     \glstextaccessdisplay
9844     {\mfirstucMakeUppercase{\glentrytext{\glslabel}}}{\glslabel}%
9845     \mfirstucMakeUppercase{\glsinsert}%
9846     }%
9847     {%

```

First use

```

9848     \glsfirstaccessdisplay
9849     {\mfirstucMakeUppercase{\glentryfirst{\glslabel}}}{\glslabel}%
9850     \mfirstucMakeUppercase{\glsinsert}%
9851     }%
9852     }%
9853     }%
9854     }%
9855     {%

```

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.

```

9856     \glscustomtext\glsinsert
9857     }%
9858 }

```

\glsгенacfmt Redefine to include accessibility information.

```

9859 \renewcommand*{\glsгенacfmt}{%
9860     \ifdefempty\glscustomtext
9861     {%
9862         \ifglused\glslabel
9863         {%

```

Subsequent use:

```

9864     \glsifplural
9865     {%

```

Subsequent plural form:

```

9866     \glscapscase
9867     {%

```

Subsequent plural form, don't adjust case:

```

9868     \acronymfont
9869     {\glsshortpluralaccessdisplay
9870     {\glentryshortpl{\glslabel}}{\glslabel}}%
9871     \glsinsert

```

9872 }%
 9873 {%

Subsequent plural form, make first letter upper case:

9874 \acronymfont
 9875 {\glsshortpluralaccessdisplay
 9876 {\Glsentryshortpl{\glslabel}}{\glslabel}}%
 9877 \glsinsert
 9878 }%
 9879 {%

Subsequent plural form, all caps:

9880 \mfirstucMakeUppercase
 9881 {\acronymfont
 9882 {\glsshortpluralaccessdisplay
 9883 {\Glsentryshortpl{\glslabel}}{\glslabel}}%
 9884 \glsinsert}%
 9885 }%
 9886 }%
 9887 {%

Subsequent singular form

9888 \glscapscase
 9889 {%

Subsequent singular form, don't adjust case:

9890 \acronymfont
 9891 {\glsshortaccessdisplay{\Glsentryshort{\glslabel}}{\glslabel}}%
 9892 \glsinsert
 9893 }%
 9894 {%

Subsequent singular form, make first letter upper case:

9895 \acronymfont
 9896 {\glsshortaccessdisplay{\Glsentryshort{\glslabel}}{\glslabel}}%
 9897 \glsinsert
 9898 }%
 9899 {%

Subsequent singular form, all caps:

9900 \mfirstucMakeUppercase
 9901 {\acronymfont{%
 9902 \glsshortaccessdisplay{\Glsentryshort{\glslabel}}{\glslabel}}%
 9903 \glsinsert}%
 9904 }%
 9905 }%
 9906 }%
 9907 {%

First use:

9908 \glsifplural
 9909 {%

First use plural form:

```
9910      \glscapscase
9911      {%
```

First use plural form, don't adjust case:

```
9912      \genplacrfullformat{\glslabel}{\glsinsert}%
9913      }%
9914      {%
```

First use plural form, make first letter upper case:

```
9915      \Genplacrfullformat{\glslabel}{\glsinsert}%
9916      }%
9917      {%
```

First use plural form, all caps:

```
9918      \mfirstucMakeUppercase
9919      {\genplacrfullformat{\glslabel}{\glsinsert}}%
9920      }%
9921      }%
9922      {%
```

First use singular form

```
9923      \glscapscase
9924      {%
```

First use singular form, don't adjust case:

```
9925      \genacrfullformat{\glslabel}{\glsinsert}%
9926      }%
9927      {%
```

First use singular form, make first letter upper case:

```
9928      \Genacrfullformat{\glslabel}{\glsinsert}%
9929      }%
9930      {%
```

First use singular form, all caps:

```
9931      \mfirstucMakeUppercase
9932      {\genacrfullformat{\glslabel}{\glsinsert}}%
9933      }%
9934      }%
9935      }%
9936      }%
9937      {%
```

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.

```
9938      \glscustomtext
9939      }%
9940 }
```

`\genacrfullformat` Redefine to include accessibility information.

```
9941 \renewcommand*{\genacrfullformat}[2]{%
```



```

9942 \glslongaccessdisplay{\glsentrylong{#1}}{#1}#2\space
9943 (\glsshortaccessdisplay{\protect\firstacronymfont{\glsentryshort{#1}}}{#1})%
9944 }

```

\Genacrfullformat Redefine to include accessibility information.

```

9945 \renewcommand*{\Genacrfullformat}[2]{%
9946 \glslongaccessdisplay{\Glsentrylong{#1}}{#1}#2\space
9947 (\glsshortaccessdisplay{\protect\firstacronymfont{\Glsentryshort{#1}}}{#1})%
9948 }

```

\genplacrfullformat Redefine to include accessibility information.

```

9949 \renewcommand*{\genplacrfullformat}[2]{%
9950 \glslongpluralaccessdisplay{\glsentrylongpl{#1}}{#1}#2\space
9951 (\glsshortpluralaccessdisplay
9952 {\protect\firstacronymfont{\glsentryshortpl{#1}}}{#1})%
9953 }

```

\Genplacrfullformat Redefine to include accessibility information.

```

9954 \renewcommand*{\Genplacrfullformat}[2]{%
9955 \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}#2\space
9956 (\glsshortpluralaccessdisplay
9957 {\protect\firstacronymfont{\glsentryshortpl{#1}}}{#1})%
9958 }

```

\@acrshort

```

9959 \def\@acrshort#1#2[#3]{%
9960 \glsdoifexists{#2}%
9961 {%
9962 \let\do@gls@link@checkfirsthyper\relax

9963 \let\glsifplural\@secondoftwo
9964 \let\glsifcaps\@firstofthree
9965 \let\glsinsert\@empty
9966 \def\glscustomtext{%
9967 \acronymfont{\glsshortaccessdisplay{\glsentryshort{#2}}{#2}}#3%
9968 }%

```

Call \@gls@link

```

9969 \@gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
9970 }%
9971 }

```

\@Acrshort

```

9972 \def\@Acrshort#1#2[#3]{%
9973 \glsdoifexists{#2}%
9974 {%
9975 \let\do@gls@link@checkfirsthyper\relax

```

```

9976 \let\glsifplural\@secondoftwo
9977 \let\glscapscase\@secondofthree
9978 \let\glsinsert\@empty
9979 \def\glscustomtext{%
9980 \acronymfont{\glsshortaccessdisplay{\Glsentryshort{#2}}{#2}}#3%
9981 }%

```

Call \@gls@link

```

9982 \@gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
9983 }%
9984 }

```

\@ACRshort

```

9985 \def\@ACRshort#1#2[#3]{%
9986 \glsdoifexists{#2}%
9987 {%
9988 \let\do@gls@link@checkfirsthyper\relax

9989 \let\glsifplural\@secondoftwo
9990 \let\glscapscase\@thirdofthree
9991 \let\glsinsert\@empty
9992 \def\glscustomtext{%
9993 \acronymfont{\glsshortaccessdisplay
9994 {\MakeUppercase{\Glsentryshort{#2}}}{#2}}#3%
9995 }%

```

Call \@gls@link

```

9996 \@gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
9997 }%
9998 }

```

\@acrlong

```

9999 \def\@acrlong#1#2[#3]{%
10000 \glsdoifexists{#2}%
10001 {%
10002 \let\do@gls@link@checkfirsthyper\relax

10003 \let\glsifplural\@secondoftwo
10004 \let\glscapscase\@firstofthree
10005 \let\glsinsert\@empty
10006 \def\glscustomtext{%
10007 \acronymfont{\glslongaccessdisplay{\Glsentrylong{#2}}{#2}}#3%
10008 }%

```

Call \@gls@link

```

10009 \@gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
10010 }%
10011 }

```

\@Acrlong

```

10012 \def\@Acrlong#1#2[#3]{%
10013   \glsdoifexists{#2}%
10014   {%
10015     \let\do@gl@link@checkfirsthyper\relax

10016     \let\glsifplural\@secondoftwo
10017     \let\glscapscase\@firstofthree
10018     \let\glsinsert\@empty
10019     \def\glscustomtext{%
10020       \acronymfont{\glslongaccessdisplay{\Glsentrylong{#2}}{#2}}#3%
10021     }%

```

Call \@gl@link

```

10022   \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
10023   }%
10024 }

```

\@ACRlong

```

10025 \def\@ACRlong#1#2[#3]{%
10026   \glsdoifexists{#2}%
10027   {%
10028     \let\do@gl@link@checkfirsthyper\relax

10029     \let\glsifplural\@secondoftwo
10030     \let\glscapscase\@firstofthree
10031     \let\glsinsert\@empty
10032     \def\glscustomtext{%
10033       \acronymfont{\glslongaccessdisplay{%
10034         \MakeUppercase{\Glsentrylong{#2}}}{#2}}#3}%
10035     }%

```

Call \@gl@link

```

10036   \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
10037   }%
10038 }

```

7.3 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.

```

10039 \renewcommand*\glossentryname}[1]{%
10040   \glsdoifexists{#1}%
10041   {%

```

```

10042 \glsnamefont{\glsnameaccessdisplay{\glsentryname{#1}}{#1}}%
10043 }%
10044 }

10045 \renewcommand*{\glossentryname}[1]{%
10046 \glsdoifexists{#1}%
10047 {%
10048 \glsnamefont{\glsnameaccessdisplay{\Glsentryname{#1}}{#1}}%
10049 }%
10050 }

10051 \renewcommand*{\glossentrydesc}[1]{%
10052 \glsdoifexists{#1}%
10053 {%
10054 \glsdescriptionaccessdisplay{\glsentrydesc{#1}}{#1}%
10055 }%
10056 }

10057 \renewcommand*{\Glossentrydesc}[1]{%
10058 \glsdoifexists{#1}%
10059 {%
10060 \glsdescriptionaccessdisplay{\Glsentrydesc{#1}}{#1}%
10061 }%
10062 }

10063 \renewcommand*{\glossentrysymbol}[1]{%
10064 \glsdoifexists{#1}%
10065 {%
10066 \glssymbolaccessdisplay{\glsentrysymbol{#1}}{#1}%
10067 }%
10068 }

10069 \renewcommand*{\Glossentrysymbol}[1]{%
10070 \glsdoifexists{#1}%
10071 {%
10072 \glssymbolaccessdisplay{\Glsentrysymbol{#1}}{#1}%
10073 }%
10074 }

```

pglossaryentryfield

```

10075 \newcommand*{\accsuppglossaryentryfield}[5]{%
10076 \glossaryentryfield{#1}%
10077 {\glsnameaccessdisplay{#2}{#1}}%
10078 {\glsdescriptionaccessdisplay{#3}{#1}}%
10079 {\glssymbolaccessdisplay{#4}{#1}}{#5}%
10080 }

```

glossarysubentryfield

```

10081 \newcommand*{\accsuppglossarysubentryfield}[6]{%
10082 \glossarysubentryfield{#1}{#2}%
10083 {\glsnameaccessdisplay{#3}{#2}}%
10084 {\glsdescriptionaccessdisplay{#4}{#2}}%

```

```

10085 {\glssymbolaccessdisplay{#5}{#2}}{#6}%
10086 }

```

7.4 Acronyms

Redefine acronym styles provided by glossaries:

long-short *<long>* (*<short>*) acronym style.

```

10087 \renewacronymstyle{long-short}%
10088 {%

```

Check for long form in case this is a mixed glossary.

```

10089 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
10090 }%
10091 {%
10092 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
10093 \renewcommand*{\genacrfullformat}[2]{%
10094 \glslongaccessdisplay{\glsentrylong{##1}}{##1}##2\space
10095 (\glsshortaccessdisplay
10096 {\protect\firstacronymfont{\glsentryshort{##1}}}{##1})%
10097 }%
10098 \renewcommand*{\Genacrfullformat}[2]{%
10099 \glslongaccessdisplay{\Glsentrylong{##1}}{##1}##2\space
10100 (\glsshortaccessdisplay
10101 {\protect\firstacronymfont{\glsentryshort{##1}}}{##1})%
10102 }%
10103 \renewcommand*{\genplacrfullformat}[2]{%
10104 \glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}##2\space
10105 (\glsshortpluralaccessdisplay
10106 {\protect\firstacronymfont{\glsentryshortpl{##1}}}{##1})%
10107 }%
10108 \renewcommand*{\Genplacrfullformat}[2]{%
10109 \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}##2\space
10110 (\glsshortpluralaccessdisplay
10111 {\protect\firstacronymfont{\glsentryshortpl{##1}}}{##1})%
10112 }%
10113 \renewcommand*{\acronymentry}[1]{%
10114 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}
10115 \renewcommand*{\acronymsort}[2]{##1}%
10116 \renewcommand*{\acronymfont}[1]{##1}%
10117 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
10118 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
10119 }

```

short-long *<short>* (*<long>*) acronym style.

```

10120 \renewacronymstyle{short-long}%
10121 {%

```

Check for long form in case this is a mixed glossary.

```

10122 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%

```

```

10123 }%
10124 {%
10125   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
10126   \renewcommand*{\genacrfullformat}[2]{%
10127     \glsshortaccessdisplay
10128     {\protect\firstacronymfont{\glsentryshort{##1}}}{##1}##2\space
10129     (\glslongaccessdisplay{\glsentrylong{##1}}{##1})%
10130   }%
10131   \renewcommand*{\Genacrfullformat}[2]{%
10132     \glsshortaccessdisplay
10133     {\protect\firstacronymfont{\Glsentryshort{##1}}}{##1}##2\space
10134     (\glslongaccessdisplay{\glsentrylong{##1}}{##1})%
10135   }%
10136   \renewcommand*{\genplacrfullformat}[2]{%
10137     \glsshortpluralaccessdisplay
10138     {\protect\firstacronymfont{\glsentryshortpl{##1}}}{##1}##2\space
10139     (\glslongpluralaccessdisplay
10140     {\glsentrylongpl{##1}}{##1})%
10141   }%
10142   \renewcommand*{\Genplacrfullformat}[2]{%
10143     \glsshortpluralaccessdisplay
10144     {\protect\firstacronymfont{\Glsentryshortpl{##1}}}{##1}##2\space
10145     (\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1})%
10146   }%
10147   \renewcommand*{\acronymentry}[1]{%
10148     \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}%
10149   \renewcommand*{\acronymsort}[2]{##1}%
10150   \renewcommand*{\acronymfont}[1]{##1}%
10151   \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
10152   \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
10153 }

```

long-short-desc *<long>* (*<short>*) acronym style that has an accompanying description (which the user needs to supply).

```

10154 \renewacronymstyle{long-short-desc}%
10155 {%
10156   \GlsUseAcrEntryDisplayStyle{long-short}%
10157 }%
10158 {%
10159   \GlsUseAcrStyleDefs{long-short}%
10160   \renewcommand*{\GenericAcronymFields}{}%
10161   \renewcommand*{\acronymsort}[2]{##2}%
10162   \renewcommand*{\acronymentry}[1]{%
10163     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10164     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})%
10165 }

```

long-sc-short-desc *<long>* (*\textsc{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

10166 \renewacronymstyle{long-sc-short-desc}%
10167 {%
10168   \GlsUseAcrEntryDispStyle{long-sc-short}%
10169 }%
10170 {%
10171   \GlsUseAcrStyleDefs{long-sc-short}%
10172   \renewcommand*{\GenericAcronymFields}{}%
10173   \renewcommand*{\acronymsort}[2]{##2}%
10174   \renewcommand*{\acronymentry}[1]{%
10175     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10176     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10177 }

```

long-sm-short-desc *⟨long⟩* (\textsmaller{*⟨short⟩*}) acronym style that has an accompanying description (which the user needs to supply).

```

10178 \renewacronymstyle{long-sm-short-desc}%
10179 {%
10180   \GlsUseAcrEntryDispStyle{long-sm-short}%
10181 }%
10182 {%
10183   \GlsUseAcrStyleDefs{long-sm-short}%
10184   \renewcommand*{\GenericAcronymFields}{}%
10185   \renewcommand*{\acronymsort}[2]{##2}%
10186   \renewcommand*{\acronymentry}[1]{%
10187     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10188     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10189 }

```

short-long-desc *⟨short⟩* ({*⟨long⟩*}) acronym style that has an accompanying description (which the user needs to supply).

```

10190 \renewacronymstyle{short-long-desc}%
10191 {%
10192   \GlsUseAcrEntryDispStyle{short-long}%
10193 }%
10194 {%
10195   \GlsUseAcrStyleDefs{short-long}%
10196   \renewcommand*{\GenericAcronymFields}{}%
10197   \renewcommand*{\acronymsort}[2]{##2}%
10198   \renewcommand*{\acronymentry}[1]{%
10199     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10200     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10201 }

```

sc-short-long-desc *⟨long⟩* (\textsc{*⟨short⟩*}) acronym style that has an accompanying description (which the user needs to supply).

```

10202 \renewacronymstyle{sc-short-long-desc}%
10203 {%
10204   \GlsUseAcrEntryDispStyle{sc-short-long}%
10205 }%

```

```

10206 {%
10207   \GlsUseAcrStyleDefs{sc-short-long}%
10208   \renewcommand*{\GenericAcronymFields}{}%
10209   \renewcommand*{\acronymsort}[2]{##2}%
10210   \renewcommand*{\acronymentry}[1]{%
10211     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10212     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10213 }

```

sm-short-long-desc *(long)* (\textsmaller{\i(short)}) acronym style that has an accompanying description (which the user needs to supply).

```

10214 \renewacronymstyle{sm-short-long-desc}%
10215 {%
10216   \GlsUseAcrEntryDispStyle{sm-short-long}%
10217 }%
10218 {%
10219   \GlsUseAcrStyleDefs{sm-short-long}%
10220   \renewcommand*{\GenericAcronymFields}{}%
10221   \renewcommand*{\acronymsort}[2]{##2}%
10222   \renewcommand*{\acronymentry}[1]{%
10223     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10224     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10225 }

```

dua *(long)* only acronym style.

```

10226 \renewacronymstyle{dua}%
10227 {%

```

Check for long form in case this is a mixed glossary.

```

10228   \ifdefempty\glscustomtext
10229   {%
10230     \ifglshaslong{\glslabel}%
10231     {%
10232       \glsifplural
10233       {%

```

Plural form:

```

10234         \glscapscase
10235         {%

```

Plural form, don't adjust case:

```

10236         \glslongpluralaccessdisplay{\glsentrylongpl{\glslabel}}{\glslabel}%
10237         \glsinsert
10238       }%
10239     }%

```

Plural form, make first letter upper case:

```

10240         \glslongpluralaccessdisplay{\Glsentrylongpl{\glslabel}}{\glslabel}%
10241         \glsinsert
10242       }%
10243     }%

```


Plural form, all caps:

```

10244         \glslongpluralaccessdisplay
10245         {\mfirstucMakeUppercase{\glsentrylongpl{\glslabel}}}{\glslabel}%
10246         \mfirstucMakeUppercase{\glsinsert}%
10247     }%
10248 }%
10249 {%

```

Singular form

```

10250         \glscapscase
10251     {%

```

Singular form, don't adjust case:

```

10252         \glslongaccessdisplay{\glsentrylong{\glslabel}}{\glslabel}\glsinsert
10253     }%
10254 {%

```

Subsequent singular form, make first letter upper case:

```

10255         \glslongaccessdisplay{\Glsentrylong{\glslabel}}{\glslabel}\glsinsert
10256     }%
10257 {%

```

Subsequent singular form, all caps:

```

10258         \glslongaccessdisplay
10259         {\mfirstucMakeUppercase
10260         {\glsentrylong{\glslabel}\glsinsert}}{\glslabel}%
10261         \mfirstucMakeUppercase{\glsinsert}%
10262     }%
10263 }%
10264 }%
10265 {%

```

Not an acronym:

```

10266         \glsgenentryfmt
10267     }%
10268 }%
10269 {\glscustomtext\glsinsert}%
10270 }%
10271 {%
10272 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
10273 \renewcommand*{\acrfullfmt}[3]{%
10274     \glslink[##1]{##2}{%
10275         \glslongaccessdisplay{\glsentrylong{##2}}{##2}##3\space
10276         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2})}%
10277 \renewcommand*{\Acrfullfmt}[3]{%
10278     \glslink[##1]{##2}{%
10279         \glslongaccessdisplay{\Glsentrylong{##2}}{##2}##3\space
10280         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2})}%
10281 \renewcommand*{\ACRfullfmt}[3]{%
10282     \glslink[##1]{##2}{%
10283         \glslongaccessdisplay

```

```

10284         {\mfirstucMakeUppercase{\glentrylong{##2}}{##2}##3\space
10285         (\glsshortaccessdisplay{\acronymfont{\glentryshort{##2}}{##2}})}%
10286 \renewcommand*{\acrfullplfmt}[3]{%
10287     \glslink[##1]{##2}{%
10288         \glslongpluralaccessdisplay
10289         {\glentrylongpl{##2}}{##2}##3\space
10290         (\glsshortpluralaccessdisplay
10291         {\acronymfont{\glentryshortpl{##2}}{##2}})}%
10292 \renewcommand*{\Acrfullplfmt}[3]{%
10293     \glslink[##1]{##2}{%
10294         \glslongpluralaccessdisplay
10295         {\Glsentrylongpl{##2}}{##2}##3\space
10296         (\glsshortpluralaccessdisplay
10297         {\acronymfont{\glentryshortpl{##2}}{##2}})}%
10298 \renewcommand*{\ACRfullplfmt}[3]{%
10299     \glslink[##1]{##2}{%
10300         \glslongpluralaccessdisplay
10301         {\mfirstucMakeUppercase{\glentrylongpl{##2}}{##2}##3\space
10302         (\glsshortpluralaccessdisplay
10303         {\acronymfont{\glentryshortpl{##2}}{##2}})}%
10304 \renewcommand*{\glentryfull}[1]{%
10305     \glslongaccessdisplay{\glentrylong{##1}}\space
10306     (\glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}{##1}})%
10307 }%
10308 \renewcommand*{\Glsentryfull}[1]{%
10309     \glslongaccessdisplay{\Glsentrylong{##1}}{##1}\space
10310     (\glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}{##1}})%
10311 }%
10312 \renewcommand*{\glentryfullpl}[1]{%
10313     \glslongpluralaccessdisplay{\glentrylongpl{##1}}{##1}\space
10314     (\glsshortpluralaccessdisplay{\acronymfont{\glentryshortpl{##1}}{##1}})%
10315 }%
10316 \renewcommand*{\Glsentryfullpl}[1]{%
10317     \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}\space
10318     (\glsshortpluralaccessdisplay{\acronymfont{\glentryshortpl{##1}}{##1}})%
10319 }%
10320 \renewcommand*{\acronymentry}[1]{%
10321     \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}{##1}}%
10322 \renewcommand*{\acronymsort}[2]{##1}%
10323 \renewcommand*{\acronymfont}[1]{##1}%
10324 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
10325 }

```

dua-desc <long> only acronym style with user-supplied description.

```

10326 \renewacronymstyle{dua-desc}%
10327 {%
10328     \GlsUseAcrEntryDispStyle{dua}%
10329 }%
10330 {%

```

```

10331 \GlsUseAcrStyleDefs{dua}%
10332 \renewcommand*{\GenericAcronymFields}{}%
10333 \renewcommand*{\acronymentry}[1]{%
10334     \glslongaccessdisplay{\acronymfont{\glsentrylong{##1}}}{##1}}%
10335 \renewcommand*{\acronymsort}[2]{##2}%
10336 }%

```

footnote *<short>*\footnote{*<long>*} acronym style.

```

10337 \renewacronymstyle{footnote}%
10338 {%

```

Check for long form in case this is a mixed glossary.

```

10339 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
10340 }%
10341 {%
10342 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%

```

Need to ensure hyperlinks are switched off on first use:

```

10343 \glshyperfirstfalse
10344 \renewcommand*{\genacrfullformat}[2]{%
10345     \glsshortaccessdisplay
10346     {\protect\firstacronymfont{\glsentryshort{##1}}}{##1}##2%
10347     \protect\footnote{\glslongaccessdisplay{\glsentrylong{##1}}}{##1}}%
10348 }%
10349 \renewcommand*{\Genacrfullformat}[2]{%
10350     \glsshortaccessdisplay
10351     {\firstacronymfont{\Glsentryshort{##1}}}{##1}##2%
10352     \protect\footnote{\glslongaccessdisplay{\glsentrylong{##1}}}{##1}}%
10353 }%
10354 \renewcommand*{\genplacrfullformat}[2]{%
10355     \glsshortpluralaccessdisplay
10356     {\protect\firstacronymfont{\glsentryshortpl{##1}}}{##1}##2%
10357     \protect\footnote{\glslongpluralaccessdisplay{\glsentrylongpl{##1}}}{##1}}%
10358 }%
10359 \renewcommand*{\Genplacrfullformat}[2]{%
10360     \glsshortpluralaccessdisplay
10361     {\protect\firstacronymfont{\Glsentryshortpl{##1}}}{##1}##2%
10362     \protect\footnote{\glslongpluralaccessdisplay{\glsentrylongpl{##1}}}{##1}}%
10363 }%
10364 \renewcommand*{\acronymentry}[1]{%
10365     \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}%
10366 \renewcommand*{\acronymsort}[2]{##1}%
10367 \renewcommand*{\acronymfont}[1]{##1}%
10368 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%

```

Don't use footnotes for \acrfull:

```

10369 \renewcommand*{\acrfullfmt}[3]{%
10370     \glslink{##1}{##2}{%
10371         \glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}##3\space
10372         (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}%

```

```

10373 \renewcommand*{\Acrfullfmt}[3]{%
10374 \glslink[##1]{##2}{%
10375 \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##2}}}{##2}##3\space
10376 (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}%
10377 \renewcommand*{\ACRfullfmt}[3]{%
10378 \glslink[##1]{##2}{%
10379 \glsshortaccessdisplay
10380 {\mfirstucMakeUppercase
10381 {\acronymfont{\glsentryshort{##2}}}{##2}##3\space
10382 (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}}%
10383 \renewcommand*{\acrfullplfmt}[3]{%
10384 \glslink[##1]{##2}{%
10385 \glsshortpluralaccessdisplay
10386 {\acronymfont{\glsentryshortpl{##2}}}{##2}##3\space
10387 (\glslongpluralaccessdisplay{\glsentrylongpl{##2}}{##2})}}}%
10388 \renewcommand*{\Acrfullplfmt}[3]{%
10389 \glslink[##1]{##2}{%
10390 \glsshortpluralaccessdisplay
10391 {\acronymfont{\Glsentryshortpl{##2}}}{##2}##3\space
10392 (\glslongpluralaccessdisplay{\glsentrylongpl{##2}}{##2})}}}%
10393 \renewcommand*{\ACRfullplfmt}[3]{%
10394 \glslink[##1]{##2}{%
10395 \glsshortpluralaccessdisplay
10396 {\mfirstucMakeUppercase
10397 {\acronymfont{\glsentryshortpl{##2}}}{##2}##3\space
10398 (\glslongpluralaccessdisplay{\glsentrylongpl{##2}}{##2})}}}%

```

Similarly for \glsentryfull etc:

```

10399 \renewcommand*{\glsentryfull}[1]{%
10400 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}\space
10401 (\glslongaccessdisplay{\glsentrylong{##1}}{##1})}%
10402 \renewcommand*{\Glsentryfull}[1]{%
10403 \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##1}}}{##1}\space
10404 (\glslongaccessdisplay{\glsentrylong{##1}}{##1})}%
10405 \renewcommand*{\glsentryfullpl}[1]{%
10406 \glsshortpluralaccessdisplay
10407 {\acronymfont{\glsentryshortpl{##1}}}{##1}\space
10408 (\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1})}%
10409 \renewcommand*{\Glsentryfullpl}[1]{%
10410 \glsshortpluralaccessdisplay
10411 {\acronymfont{\Glsentryshortpl{##1}}}{##1}\space
10412 (\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1})}%
10413 }

```

footnote-sc \textsc{<short>}\footnote{<long>} acronym style.

```

10414 \renewacronymstyle{footnote-sc}%
10415 {%
10416 \GlsUseAcrEntryDisplayStyle{footnote}%
10417 }%
10418 {%

```

```

10419 \GlsUseAcrStyleDefs{footnote}%
10420 \renewcommand{\acronymentry}[1]{%
10421     \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}
10422 \renewcommand{\acronymfont}[1]{\textsc{##1}}%
10423 \renewcommand*{\acrpluralsuffix}{\glstextup{\glspluralsuffix}}%
10424 }%

```

footnote-sm \textsmaller{<short>}\footnote{<long>} acronym style.

```

10425 \renewacronymstyle{footnote-sm}%
10426 {%
10427     \GlsUseAcrEntryDisplayStyle{footnote}%
10428 }%
10429 {%
10430     \GlsUseAcrStyleDefs{footnote}%
10431     \renewcommand{\acronymentry}[1]{%
10432         \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}
10433     \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
10434     \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
10435 }%

```

footnote-desc <short>\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

10436 \renewacronymstyle{footnote-desc}%
10437 {%
10438     \GlsUseAcrEntryDisplayStyle{footnote}%
10439 }%
10440 {%
10441     \GlsUseAcrStyleDefs{footnote}%
10442     \renewcommand*{\GenericAcronymFields}{}%
10443     \renewcommand*{\acronymsort}[2]{##2}%
10444     \renewcommand*{\acronymentry}[1]{%
10445         \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10446         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}})%
10447 }

```

footnote-sc-desc \textsc{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

10448 \renewacronymstyle{footnote-sc-desc}%
10449 {%
10450     \GlsUseAcrEntryDisplayStyle{footnote-sc}%
10451 }%
10452 {%
10453     \GlsUseAcrStyleDefs{footnote-sc}%
10454     \renewcommand*{\GenericAcronymFields}{}%
10455     \renewcommand*{\acronymsort}[2]{##2}%
10456     \renewcommand*{\acronymentry}[1]{%
10457         \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10458         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}})%
10459 }

```

footnote-sm-desc \textsmaller{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

10460 \renewacronymstyle{footnote-sm-desc}%
10461 {%
10462   \GlsUseAcrEntryDispStyle{footnote-sm}%
10463 }%
10464 {%
10465   \GlsUseAcrStyleDefs{footnote-sm}%
10466   \renewcommand*{\GenericAcronymFields}{}%
10467   \renewcommand*{\acronymsort}[2]{##2}%
10468   \renewcommand*{\acronymentry}[1]{%
10469     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10470     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10471 }
```

Use \newacronymhook to modify the key list to set the access text to the long version by default.

```

10472 \renewcommand*{\newacronymhook}{%
10473   \edef\@gls@keylist{shortaccess=\the\glslongtok,%
10474     \the\glskeylisttok}%
10475   \expandafter\glskeylisttok\expandafter{\@gls@keylist}%
10476 }
```

defaultNewAcronymDef Modify default style to use access text:

```

10477 \renewcommand*{\DefaultNewAcronymDef}{%
10478   \edef\@do@newglossaryentry{%
10479     \noexpand\newglossaryentry{\the\glslabeltok}%
10480     {%
10481       type=\acronymtype,%
10482       name={\the\glsshorttok},%
10483       description={\the\glslongtok},%
10484       descriptionaccess=\relax,%
10485       text={\the\glsshorttok},%
10486       access={\noexpand\@glo@textaccess},%
10487       sort={\the\glsshorttok},%
10488       short={\the\glsshorttok},%
10489       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10490       shortaccess={\the\glslongtok},%
10491       long={\the\glslongtok},%
10492       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10493       descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10494       first={\noexpand\glslongaccessdisplay
10495         {\the\glslongtok}{\the\glslabeltok}\space
10496         (\noexpand\glsshortaccessdisplay
10497           {\the\glsshorttok}{\the\glslabeltok})},%
10498       plural={\the\glsshorttok\acrpluralsuffix},%
10499       firstplural={\noexpand\glslongpluralaccessdisplay
10500         {\noexpand\@glo@longpl}{\the\glslabeltok}\space
10501         (\noexpand\glsshortpluralaccessdisplay
```

```

10502         {\noexpand\@glo@shortpl}\the\glslabeltok}}},%
10503     firstaccess=\relax,
10504     firstpluralaccess=\relax,
10505     textaccess={\noexpand\@glo@shortaccess},%
10506     \the\glskeylisttok
10507 }%
10508 }%
10509 \let\@org@gls@assign@firstpl\gls@assign@firstpl
10510 \let\@org@gls@assign@plural\gls@assign@plural
10511 \let\@org@gls@assign@descplural\gls@assign@descplural
10512 \def\gls@assign@firstpl##1##2{%
10513     \@gls@expand@field{##1}{firstpl}{##2}%
10514 }%
10515 \def\gls@assign@plural##1##2{%
10516     \@gls@expand@field{##1}{plural}{##2}%
10517 }%
10518 \def\gls@assign@descplural##1##2{%
10519     \@gls@expand@field{##1}{descplural}{##2}%
10520 }%
10521 \@do@newglossaryentry
10522 \let\gls@assign@firstpl\@org@gls@assign@firstpl
10523 \let\gls@assign@plural\@org@gls@assign@plural
10524 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
10525 }

```

otnoteNewAcronymDef

```

10526 \renewcommand*{\DescriptionFootnoteNewAcronymDef}{%
10527     \edef\@do@newglossaryentry{%
10528         \noexpand\newglossaryentry{\the\glslabeltok}%
10529         {%
10530             type=\acronymtype,%
10531             name={\noexpand\acronymfont{\the\glsshorttok}},%
10532             sort={\the\glsshorttok},%
10533             text={\the\glsshorttok},%
10534             short={\the\glsshorttok},%
10535             shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10536             shortaccess={\the\glslongtok},%
10537             long={\the\glslongtok},%
10538             longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10539             access={\noexpand\@glo@textaccess},%
10540             plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10541             symbol={\the\glslongtok},%
10542             symbolplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10543             firstpluralaccess=\relax,
10544             textaccess={\noexpand\@glo@shortaccess},%
10545             \the\glskeylisttok
10546         }%
10547     }%
10548     \let\@org@gls@assign@firstpl\gls@assign@firstpl

```

```

10549 \let\@org@gls@assign@plural\gls@assign@plural
10550 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
10551 \def\gls@assign@firstpl##1##2{%
10552   \@@gls@expand@field{##1}{firstpl}{##2}%
10553 }%
10554 \def\gls@assign@plural##1##2{%
10555   \@@gls@expand@field{##1}{plural}{##2}%
10556 }%
10557 \def\gls@assign@symbolplural##1##2{%
10558   \@@gls@expand@field{##1}{symbolplural}{##2}%
10559 }%
10560 \do@newglossaryentry
10561 \let\gls@assign@plural\@org@gls@assign@plural
10562 \let\gls@assign@firstpl\@org@gls@assign@firstpl
10563 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
10564 }

```

ptionNewAcronymDef

```

10565 \renewcommand*{\DescriptionNewAcronymDef}{%
10566   \edef\@do@newglossaryentry{%
10567     \noexpand\newglossaryentry{\the\glslabeltok}%
10568     {%
10569       type=\acronymtype,%
10570       name={\noexpand
10571         \acrnameformat{\the\glsshorttok}{\the\glslongtok}},%
10572       access={\noexpand\@glo@textaccess},%
10573       sort={\the\glsshorttok},%
10574       short={\the\glsshorttok},%
10575       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10576       shortaccess={\the\glslongtok},%
10577       long={\the\glslongtok},%
10578       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10579       first={\the\glslongtok},%
10580       firstaccess=\relax,
10581       firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10582       text={\the\glsshorttok},%
10583       textaccess={\the\glslongtok},%
10584       plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10585       symbol={\noexpand\@glo@text},%
10586       symbolaccess={\noexpand\@glo@textaccess},%
10587       symbolplural={\noexpand\@glo@plural},%
10588       firstpluralaccess=\relax,
10589       textaccess={\noexpand\@glo@shortaccess},%
10590       \the\glskeylisttok}%
10591   }%
10592   \let\@org@gls@assign@firstpl\gls@assign@firstpl
10593   \let\@org@gls@assign@plural\gls@assign@plural
10594   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
10595   \def\gls@assign@firstpl##1##2{%

```



```

10596 \@@gls@expand@field{##1}{firstpl}{##2}%
10597 }%
10598 \def\gls@assign@plural##1##2{%
10599 \@@gls@expand@field{##1}{plural}{##2}%
10600 }%
10601 \def\gls@assign@symbolplural##1##2{%
10602 \@@gls@expand@field{##1}{symbolplural}{##2}%
10603 }%
10604 \do@newglossaryentry
10605 \let\gls@assign@firstpl\@org@gls@assign@firstpl
10606 \let\gls@assign@plural\@org@gls@assign@plural
10607 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
10608 }

```

otnoteNewAcronymDef

```

10609 \renewcommand*{\FootnoteNewAcronymDef}{%
10610 \edef\@do@newglossaryentry{%
10611 \noexpand\newglossaryentry{\the\glslabeltok}%
10612 {%
10613 type=\acronymtype,%
10614 name={\noexpand\acronymfont{\the\glsshorttok}},%
10615 sort={\the\glsshorttok},%
10616 text={\the\glsshorttok},%
10617 textaccess={\the\glslongtok},%
10618 access={\noexpand\@glo@textaccess},%
10619 plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10620 short={\the\glsshorttok},%
10621 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10622 long={\the\glslongtok},%
10623 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10624 description={\the\glslongtok},%
10625 descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10626 \the\glskeylisttok
10627 }%
10628 }%
10629 \let\@org@gls@assign@plural\gls@assign@plural
10630 \let\@org@gls@assign@firstpl\gls@assign@firstpl
10631 \let\@org@gls@assign@descplural\gls@assign@descplural
10632 \def\gls@assign@firstpl##1##2{%
10633 \@@gls@expand@field{##1}{firstpl}{##2}%
10634 }%
10635 \def\gls@assign@plural##1##2{%
10636 \@@gls@expand@field{##1}{plural}{##2}%
10637 }%
10638 \def\gls@assign@descplural##1##2{%
10639 \@@gls@expand@field{##1}{descplural}{##2}%
10640 }%
10641 \do@newglossaryentry
10642 \let\gls@assign@plural\@org@gls@assign@plural

```

```

10643 \let\gls@assign@firstpl\@org@gls@assign@firstpl
10644 \let\gls@assign@descplural\@org@gls@assign@descplural
10645 }

```

\SmallNewAcronymDef

```

10646 \renewcommand*{\SmallNewAcronymDef}{%
10647 \edef\@do@newglossaryentry{%
10648 \noexpand\newglossaryentry{\the\glslabeltok}%
10649 {%
10650 type=\acronymtype,%
10651 name={\noexpand\acronymfont{\the\glsshorttok}},%
10652 access={\noexpand\@glo@symbolaccess},%
10653 sort={\the\glsshorttok},%
10654 short={\the\glsshorttok},%
10655 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10656 shortaccess={\the\glslongtok},%
10657 long={\the\glslongtok},%
10658 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10659 text={\noexpand\@glo@short},%
10660 textaccess={\noexpand\@glo@shortaccess},%
10661 plural={\noexpand\@glo@shortpl},%
10662 first={\the\glslongtok},%
10663 firstaccess=\relax,
10664 firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10665 description={\noexpand\@glo@first},%
10666 descriptionplural={\noexpand\@glo@firstplural},%
10667 symbol={\the\glsshorttok},%
10668 symbolaccess={\the\glslongtok},%
10669 symbolplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10670 \the\glskeylisttok
10671 }%
10672 }%
10673 \let\@org@gls@assign@firstpl\gls@assign@firstpl
10674 \let\@org@gls@assign@plural\gls@assign@plural
10675 \let\@org@gls@assign@descplural\gls@assign@descplural
10676 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
10677 \def\gls@assign@firstpl##1##2{%
10678 \@@gls@expand@field{##1}{firstpl}{##2}%
10679 }%
10680 \def\gls@assign@plural##1##2{%
10681 \@@gls@expand@field{##1}{plural}{##2}%
10682 }%
10683 \def\gls@assign@descplural##1##2{%
10684 \@@gls@expand@field{##1}{descplural}{##2}%
10685 }%
10686 \def\gls@assign@symbolplural##1##2{%
10687 \@@gls@expand@field{##1}{symbolplural}{##2}%
10688 }%
10689 \@do@newglossaryentry

```

```

10690 \let\gls@assign@firstpl\@org@gls@assign@firstpl
10691 \let\gls@assign@plural\@org@gls@assign@plural
10692 \let\gls@assign@descplural\@org@gls@assign@descplural
10693 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
10694 }

```

The following are kept for compatibility with versions before 3.0:

```

\glsshortaccesskey
10695 \newcommand*{\glsshortaccesskey}{\glsshortkey access}%

hortpluralaccesskey
10696 \newcommand*{\glsshortpluralaccesskey}{\glsshortpluralkey access}%

\glslongaccesskey
10697 \newcommand*{\glslongaccesskey}{\glslongkey access}%

longpluralaccesskey
10698 \newcommand*{\glslongpluralaccesskey}{\glslongpluralkey access}%

```

7.5 Debugging Commands

```

\showglonameaccess
10699 \newcommand*{\showglonameaccess}[1]{%
10700 \expandafter\show\csname glo@\glsdetoklabel{#1}@textaccess\endcsname
10701 }

\showglotextaccess
10702 \newcommand*{\showglotextaccess}[1]{%
10703 \expandafter\show\csname glo@\glsdetoklabel{#1}@textaccess\endcsname
10704 }

showglopluralaccess
10705 \newcommand*{\showglopluralaccess}[1]{%
10706 \expandafter\show\csname glo@\glsdetoklabel{#1}@pluralaccess\endcsname
10707 }

\showglofirstaccess
10708 \newcommand*{\showglofirstaccess}[1]{%
10709 \expandafter\show\csname glo@\glsdetoklabel{#1}@firstaccess\endcsname
10710 }

lofirstpluralaccess
10711 \newcommand*{\showglofirstpluralaccess}[1]{%
10712 \expandafter\show\csname glo@\glsdetoklabel{#1}@firstpluralaccess\endcsname
10713 }

```

showglosymbolaccess

```
10714 \newcommand*{\showglosymbolaccess}[1]{%
10715   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolaccess\endcsname
10716 }
```

symbolpluralaccess

```
10717 \newcommand*{\showglosymbolpluralaccess}[1]{%
10718   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolpluralaccess\endcsname
10719 }
```

\showglodescaccess

```
10720 \newcommand*{\showglodescaccess}[1]{%
10721   \expandafter\show\csname glo@\glsdetoklabel{#1}@descaccess\endcsname
10722 }
```

glodescpluralaccess

```
10723 \newcommand*{\showglodescpluralaccess}[1]{%
10724   \expandafter\show\csname glo@\glsdetoklabel{#1}@descpluralaccess\endcsname
10725 }
```

\showgloshortaccess

```
10726 \newcommand*{\showgloshortaccess}[1]{%
10727   \expandafter\show\csname glo@\glsdetoklabel{#1}@shortaccess\endcsname
10728 }
```

loshortpluralaccess

```
10729 \newcommand*{\showgloshortpluralaccess}[1]{%
10730   \expandafter\show\csname glo@\glsdetoklabel{#1}@shortpluralaccess\endcsname
10731 }
```

\showglolongaccess

```
10732 \newcommand*{\showglolongaccess}[1]{%
10733   \expandafter\show\csname glo@\glsdetoklabel{#1}@longaccess\endcsname
10734 }
```

glolongpluralaccess

```
10735 \newcommand*{\showglolongpluralaccess}[1]{%
10736   \expandafter\show\csname glo@\glsdetoklabel{#1}@longpluralaccess\endcsname
10737 }
```

8 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.

```
10738 \NeedsTeXFormat{LaTeX2e}
10739 \ProvidesPackage{glossaries-babel}[2014/11/22 v4.12 (NLCT)]
```

Load tracklang to obtain language settings.

```
10740 \RequirePackage{tracklang}
10741 \let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
10742 \AnyTrackedLanguages
10743 {%
10744   \ForEachTrackedDialect{\this@dialect}{%
10745     \IfTrackedLanguageFileExists{\this@dialect}%
10746     {glossaries-}% prefix
10747     {.ldf}%
10748     {%
10749       \RequireGlossariesLang{\CurrentTrackedTag}%
10750     }%
10751     {%
10752       \PackageWarningNoLine{glossaries}%
10753       {No language module detected for ‘\this@dialect’.\MessageBreak
10754       Language modules need to be installed separately.\MessageBreak
10755       Please check on CTAN for a bundle called\MessageBreak
10756       ‘glossaries-\CurrentTrackedLanguage’ or similar}%
10757     }%
10758   }%
10759 }%
10760 {}%
```

8.1 Polyglossia Captions

Language support has now been split off into independent language modules.

```
10761 \NeedsTeXFormat{LaTeX2e}
10762 \ProvidesPackage{glossaries-polyglossia}[2014/11/22 v4.12 (NLCT)]
```

Load tracklang to obtain language settings.

```
10763 \RequirePackage{tracklang}
10764 \let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
10765 \AnyTrackedLanguages
10766 {%
10767   \ForEachTrackedDialect{\this@dialect}{%
10768     \IfTrackedLanguageFileExists{\this@dialect}%
10769     {glossaries-}% prefix
10770     {.ldf}%
10771     {%
10772       \RequireGlossariesLang{\CurrentTrackedTag}%
10773     }%
10774     {%
10775       \PackageWarningNoLine{glossaries}%
10776       {No language module detected for ‘\this@dialect’.\MessageBreak
10777       Language modules need to be installed separately.\MessageBreak
10778       Please check on CTAN for a bundle called\MessageBreak
```

```

10779      'glossaries-\CurrentTrackedLanguage' or similar}%
10780      }%
10781      }%
10782      }%
10783      {}%

```

Glossary

`makeindex` An indexing application. [10](#), [25](#), [26](#), [160](#)

`xindy` An flexible indexing application with multilingual support written in Perl. [10](#), [25](#), [26](#), [160](#)

Change History

??		<code>\theglsentrycounter</code> setting the page number too soon 96
	<code>super</code> : fixed typo in <code>\subglossentry</code>	<code>\glsadd</code> : fixed bug caused by
	<code>(\glossentrydesc)</code> 272	<code>\theglsentrycounter</code> setting the page number too soon
1.01	General: Added range facility in format key 98 142
	<code>\writeist</code> : Added spaces after <code>\delimN</code> and <code>\delimR</code> in ist file 145	1.08
		General: Added babel support ... 31
1.03	<code>\makefirstuc</code> : changed 'protected@edef' to 'def' 245	<code>\capitalisewords</code> : made robust 246
		<code>listgroup</code> : changed <code>listgroup</code> style to use <code>\glsgetgrouptitle</code> 253
1.04	General: Added <code>\glstextformat</code> 82	<code>altlistgroup</code> : changed <code>altlistgroup</code> style to use <code>\glsgetgrouptitle</code> 254
1.05	<code>\glossarysection</code> : added <code>\@mkboth</code> to <code>\glossarysection</code> 37	<code>\makefirstuc</code> : made robust ... 244
	<code>\gls@defglossaryentry</code> : Changed the default value of the sort key to just the value of the name key 72	1.09
	<code>\glsmakefirstuc</code> : new 246	<code>\@mfu@nocaplist</code> : new 247
1.06	General: now requires <code>etoolbox</code> . 244	<code>\capitalisewords</code> : added check for words that shouldn't be capitalised 246
	<code>\capitalisewords</code> : new 246	<code>\gMfUnocap</code> : new 247
	<code>\xcapitalisewords</code> : new 247	<code>\mfu@checkword</code> : new 247
1.07	<code>\@gls@link</code> : fixed bug caused by	<code>\MFUclear</code> : new 247
		1.1
		<code>\@glossarysection</code> : numbered sections and auto label added 39
		<code>\@gls@tmpb</code> : changed <code>\toksdef</code> to <code>\newtoks</code> 100

\@gls@toc: numberline added .. 40	\SetFootnoteAcronymStyle:
\@p@glossarysection: num-	Added \protect before
bered sections and auto label	\footnote and \glslink . 225
added 39	symbolplural: new 61
General: amsgen now loaded	1.13
(\new@ifnextchar needed) .. 4	General: fixed bug that ignored 3rd
translate: translate option	parameter 113–121
added 22	\ACRfullpl: new 200
\setglossarysection: new ... 38	\Acrfullpl: new 200
numberedsection: numbered-	\acrfullpl: new 199
section package option added . 6	\acrpluralsuffix: New 197
numberline: numberline option	\gls@defglossaryentry:
added 5	Changed default first value .. 72
1.10	Changed default firstplural
\ecapitalisewords: new 247	value 72
\emakefirstuc: new 246	Removed restriction on only
1.12	using \newglossaryentry in
\@GLSpl: now uses \glsentrydescplural	the preamble 77
and \glsentrysymbolplural	\newacronym: Removed re-
instead of \glsentrydesc	striction on only using
and \glsentrysymbol 112	\newacronym in the preamble 197
\@GLspl@: now uses \glsentrydescplural	
and \glsentrysymbolplural	\@gls@hypergroup: new 249
instead of \glsentrydesc	General: added nonnumberlist key
and \glsentrysymbol 111	to \printglossary 183
\@glspl@: now uses \glsentrydescplural	added numberedsection key to
and \glsentrysymbolplural	\printglossary 181
instead of \glsentrydesc	\firstacronymfont: new 201
and \glsentrysymbol 110	\glsautoprefix: new 6
General: added check for	\glsnavhyperlink: changed
\hypertarget separate to	'edef to'protected@edef ... 248
\hyperlink (memoir de-	\glsnavhypertarget: added
finies \hyperlink but not	write to aux file 248
\hypertarget) 106	\glsnavigation: changed to
descriptionplural: new 60	only use labels for groups that
\gls@defglossaryentry:	are present 249
Changed default first plural to	1.15
be first key with s appended	\@gls@link: added \glslabel . 96
(was text key with s appended) 72	\gls@defglossaryentry: check
descriptionplural support	for \@glo@first in descrip-
added 72	tion 76
symbolplural support added .. 72	check for \@glo@text in sym-
\Glsentrydescplural: New .. 136	bol 76
\glsentrydescplural: New .. 136	\gls@hypergroup: new . 249
\Glsentrysymbolplural: New 137	\glsnavhypertarget: added
\glsentrysymbolplural: New 137	check if rerun required 248
\SetDescriptionFootnoteAcronymStyle	\glssettoctitle: new 30
Added \protect before	\printglossary: changed the
\footnote and \glslink . 218	way the TOC title is set 168

1.16			\gls@defglossaryentry: added nonumberlist key 72 added parent key 72 added see key 72 Stored main part of entry format when entry is defined 77
	\@GLS@: Test glossary type is \acronymtype in addition to checking if footnote option has been used 110		\gls@suffixF: new 35
	\@GLSpl: Test glossary type is \acronymtype in addition to checking if footnote option has been used 112		\gls@suffixFF: new 36
	\@GLs@: Test glossary type is \acronymtype in addition to checking if footnote option has been used 109		\gls@wrglossary: modified to allow for xindy support 161
	\@GLspl@: Test glossary type is \acronymtype in addition to checking if footnote option has been used 111		\glshyperlink: new 142
	\@GLs@: Test glossary type is \acronymtype in addition to checking if footnote option has been used 108		\glshypernumber: modified to allow material to be attached to location 194
	\@GLsdisp: Test glossary type is \acronymtype in addition to checking if footnote option has been used 113		\glsnavhyperlink: replaced 'hy- perlink to '@glslink 248
	\@GLspl@: Test glossary type is \acronymtype in addition to checking if footnote option has been used 110		\glsnavhypertarget: replaced 'hypertarget to '@glstarget . 248
	\@glstarget: raised the hyper- target so the target text doesn't scroll off the top of the page 106		\glssee: new 166
	\gls@defglossaryentry: Changed def to let 72		\glsseeformat: new 166
1.17			\glsSetSuffixF: new 35
	\@@do@wrglossary: new 163		\glsSetSuffixFF: new 36
	\@do@seeglossary: new 165		\ifglsxindy: new 25
	\@glo@storeentry: new 78		\istfilename: added xindy sup- port 34
	\@gls@glossary: changed defi- nition to use \index instead of \@index 160		\newglossarystyle: made \newglossarystyle long . 193
	\@glsdefaultplural: new 63		\nopostdesc: new 33
	\@glsdefaultsort: new 64		nonumberlist: new 62
	\@glshypernumber: new 194		\printglossary: added check to determine if \printglossary is already defined 168
	\@glsnoname: new 63		added print language to aux file 168
	\@glsnonextpages: new 184		order: order package option added 25
	General: added xindy support ... 25		\writeist: added xindy support 145
	parent: new 62	1.18	
	see: new 61		\@gls@loadlist: new 8
			\@gls@loadlong: new 8
			\@gls@loadsuper: new 8
			\@gls@loadtree: new 8
			\gls@defglossaryentry: Changed default value of sort to \@glsdefaultsort 72
			moved sort sanitization to \newglossaryentry 76
			\glstarget: new 187
			\oldacronym: new 196
			nolist: new 8

nolong: new	8	\glossarysection: changed	
sort: moved sanitization to		\@mkboth to \glossarymark	37
\newglossaryentry	60	\glsglossarymark: New	38
nostyles: new	8		2.03
nosuper: new	8	\@GLS@: Added check for hyper-	
notree: new	8	first	110
1.19		\@GLSpl: Added check for hyper-	
\glsclearpage: new	40	first	112
\glsdisp: new	112	\@Gls@: Added check for hyper-	
\SetDescriptionAcronymStyle:		first	109
changed \acronymfont to		\@Glspl@: Added check for hyper-	
use \textsmaller instead of		first	111
\smaller	222	\@gls@: Added check for hyper-	
\SetDescriptionFootnoteAcronymStyle:		first	108
changed \acronymfont to		\@gls@link: new	95
use \textsmaller instead of		\@gls@link: added \leavevmode	
\smaller	218	96
\SetFootnoteAcronymStyle:		Moved entry existence check to	
changed \acronymfont to		avoid duplicate code	96
use \textsmaller instead of		\@glsdisp: Added check for hy-	
\smaller	225	perfirst	113
\SetSmallAcronymStyle:		\@Glspl@: Added check for hyper-	
changed \acronymfont to		first	110
use \textsmaller instead of		\glsglossarymark: Added check	
\smaller	228	to see if it's already defined ..	38
2.01		hyperfirst: new	23
\@gls@link: moved \@do@wrglossary			2.04
before term is displayed to pre-		\@GLS@: Changed test to check if	
vent unwanted whatsit	97	glossary type has been identi-	
\foralllglossaries: replaced		fied as a list of acronyms ...	110
\ifthenelse with \ifx	49	\@GLSpl: Changed test to check if	
\forallgsentries: replaced		glossary type has been identi-	
\ifthenelse with \ifx	50	fied as a list of acronyms ...	112
\glsdefmain: new	12	\@Gls@: Changed test to check if	
\glsdescwidth: changed		glossary type has been identi-	
\linewidth to \hsize . 256,271		fied as a list of acronyms ...	109
\glslistdottedwidth: changed		\@Glspl@: Changed test to check	
\linewidth to \hsize	255	if glossary type has been iden-	
\glspagelistwidth: changed		tified as a list of acronyms ..	111
\linewidth to \hsize . 256,272		\@glossaryentryfield: new ..	78
nomain: added nomain package		\@glossarysubentryfield:	
option	13	new	78
\writeist: removed item_02 - no		\@gls@: Changed test to check if	
such makeindex key	149	glossary type has been identi-	
2.02		fied as a list of acronyms ...	108
\@printglossary: suppressed		\@glsacronymlists: new	14
warning globally rather than		\@glsdisp: Changed test to check	
locally	170	if glossary type has been iden-	
		tified as a list of acronyms ..	113

\@glsp1@: Changed test to check if glossary type has been identified as a list of acronyms ..	110	2.05	\@glstdisp: Added closing brace. Patch provided by Sergiu Dotenco	113
\@newglossaryentryposthook: new	77		Removed spurious brace. Patch provided by Sergiu Dotenco	113
\@newglossaryentryprehook: new	77		\writeist: Added \string before opening and closing braces. Patch provided by Segiu Dotenco	150
acronymlists:new	15	2.06	\altnewglossary:new	57
\DeclareAcronymList:new ...	14		\CustomAcronymFields:new .	230
\DefineAcronymSynonyms:new	213		\CustomNewAcronymDef:new .	231
\gls@defglossaryentry: added user1-6 keys	73		\SetCustomDisplayStyle:new	230
\glsadd: fixed bug that ignored counter	142		\SetCustomStyle:new	231
\Glsentryuseri:new	138	2.07		
\glsentryuseri:new	138		General: glssadd format key stored in \@glsnumberformat (was mistakenly stored in \@glo@format)	142
\Glsentryuserii:new	138	3.0	\@@do@wrglossary: added check for hyper location prefix ...	163
\glsentryuserii:new	138		modified to use new format ..	163
\Glsentryuseriii:new	139		\@@glossarysec: replaced \@ifundefined with \@ifcsundef	6
\glsentryuseriii:new	139		\@do@seeglossary: Sanitize and escape cross-referencing information	165
\Glsentryuseriv:new	139		\@gls@counterwithin:new ...	10
\glsentryuseriv:new	139		\@gls@ifinlist:new	41
\Glsentryuseriv:new	139		\@gls@link: added \@gls@saveentrycounter	97
\Glsentryuseriv:new	139		added \@gls@setsort	97
\ns@newglossary: added check to determine if \@gls@<type>@display and \@gls@<type>@displayfirst have been defined.	57		\@gls@saveentrycounter:new	97
\SetAcronymLists:new	15		\@gls@setupsort@def:new ...	11
\SetDefaultAcronymDisplayStyle:new	215		\@gls@setupsort@standard:new	10
\SetDefaultAcronymStyle:new	215		\@gls@setupsort@use:new ...	11
\SetDescriptionAcronymDisplayStyle:new	220		\@gls@xdy@locationlist:new	44
\SetDescriptionDUAAcronymDisplayStyle:new	219		\@gls@link: replaced \@ifundefined with \@ifcsundef	106
\SetDescriptionFootnoteAcronymDisplayStyle:new	216		\@gls@nextpages:new	184
\SetDUADisplayStyle:new ..	228		\@makeglossary: Added check for savewrites	151
\SetFootnoteAcronymDisplayStyle:new	223		\@print@glossary: replaced \@ifundefined with	
\SetSmallAcronymDisplayStyle:new	225			

\ifcsundef	170	\glsadd:added \@gls@saveentrycounter	143
\@printglossary: added		\GlsAddXdyCounters:new	41
\currentglossary	169	\glsentrycounterlabel:new	186
added \glsnextpages	169	\glsentryitem:new	186
make toctitle default to title ..	169	\Glsentrylong:new	140
\@set@glo@numformat: added		\glsentrylong:new	140
4th argument	98	\Glsentrylongpl:new	140
\@xdyattributelist:new	41	\glsentrylongpl:new	140
General: added prefix to hyperlink	195	\Glsentryshort:new	140
etoolbox now loaded	4	\glsentryshort:new	139
replaced \@ifundefined with		\Glsentryshortpl:new	140
\ifcsundef	29, 32, 93, 181	\glsentryshortpl:new	140
\acrfootnote:new	216	\glsgetgroup title: replaced \@ifundefined with	
\ACRfull: added starred version	199	\ifcsundef	191
\Acrfull: added starred version	198	\gls glossary mark: replaced \@ifundefined with	
\acrfull: added starred version	198	\ifcsundef	38
\ACRfullpl: added starred version	200	\gls hyperlink: changed default from \glsentryname to	
\Acrfullpl: added starred version	200	\glsentrytext	142
\acrfullpl: added starred version	199	\gls hyper number: replaced \@ifundefined with	
\acrlinkfootnote:new	216	\ifcsundef	194
\acrnolinkfootnote:new ...	216	\gls number format: replaced \@ifundefined with	
savewrites:new	26	\ifcsundef	36
see: added \@glo@seeautonumberlist	61	\gls ref entry:new	186
seeautonumberlist:new	8	\gls resetsubentrycounter: new	185
\glossarysection: replaced \@ifundefined with		\gls see item: hyperlink uses \glsseeitemformat instead of \glsentryname	167
\ifcsundef	37	\gls see item format:new	167
\glossarystyle: replaced \@ifundefined with		\glssortnumberfmt:new	11
\ifcsundef	192	\glsstepentry:new	185
\gls@codepage: replaced \@ifundefined with		\glsstepsubentry:new	185
\ifcsundef	25	\glssubentrycounterlabel: new	186
\gls@def glossary entry: added \@gls@defsort	76	\glssubentryitem:new	186
added short and long keys	73	the glossary: replaced \@ifundefined with \ifcsundef	186
replaced \@ifundefined with \ifcsundef	73	short:new	63
\gls@docclearpage: replaced \@ifundefined with		shortplural:new	63
\ifcsundef	39	\if glossary exists: replaced \@ifundefined with	
\gls@wrglossary: modified to take into account savewrites	161	\ifcsundef	50

<code>\ifglentryexists:</code>	re-	<code>\showglossaries:new</code>	237
placed <code>\@ifundefined</code> with		<code>\showglossarycounter:new</code>	237
<code>\ifcsundef</code>	51	<code>\showglossaryentries:new</code>	237
<code>\istfile:deprecated</code>	159	<code>\showglossaryin:new</code>	237
<code>glossaryentry:new</code>	184	<code>\showglossaryout:new</code>	237
<code>glossarysubentry:new</code>	185	<code>\showglossarytitle:new</code>	237
<code>\newglossaryentry:</code>	replaced	<code>\showglosymbol:new</code>	235
<code>\DeclareRobustCommand</code>		<code>\showglosymbolplural:new</code>	235
with <code>\newrobustcmd</code>	66	<code>\showglotext:new</code>	233
<code>\newglossarystyle:</code>	re-	<code>\showglotype:new</code>	233
placed <code>\@ifundefined</code> with		<code>\showglouseri:new</code>	234
<code>\ifcsundef</code>	193	<code>\showglouserii:new</code>	234
<code>\ns@newglossary:</code>	added	<code>\showglouseriii:new</code>	234
<code>\@gls@defsortcount</code>	57	<code>\showglouseriv:new</code>	234
replaced <code>\@ifundefined</code> with		<code>\showglouserv:new</code>	234
<code>\ifcsundef</code>	57	<code>\showglouservi:new</code>	234
<code>entrycounter:new</code>	9	<code>subentrycounter:new</code>	10
<code>entrycounterwithin:new</code>	9	<code>\writeist:</code>	added xindy-only
<code>\oldacronym:replaced\@ifundefined</code>		macro definitions to glossary	
with <code>\ifcsundef</code>	196	open tag	147
<code>compatible-2.07: compatible-</code>		modified to support new for-	
<code>2.07 option added</code>	27	mat	145
<code>long:new</code>	63		
<code>longplural:new</code>	63	<code>\@glswritefiles:</code>	added check
<code>nonumberlist:now boolean</code>	62	for empty glossaries	159
<code>sort:new</code>	10	General: made robust	109
<code>counter:replaced\@ifundefined</code>		<code>\ACRfull:</code>	made robust 199
with <code>\ifcsundef</code>	61	<code>\Acrfull:</code>	made robust 198
<code>\printglossary:</code>	replaced	<code>\acrfull:</code>	made robust 198
<code>\@ifundefined</code>	with	<code>\acrfullformat:</code>	removed
<code>\ifcsundef</code>	168	<code>\acronymfont:</code>	as it should al-
<code>\SetDescriptionFootnoteAcronymDisplayStyle:</code>	be set in the second ar-		
expanded options link op-	gument.	<code>\ACRfullpl:</code>	made robust 200
tions	216	<code>\Acrfullpl:</code>	made robust 200
<code>\setentrycounter:</code>	added op-	<code>\acrfullpl:</code>	made robust 199
tional argument	192	<code>\ACRlong:</code>	made robust 131
<code>\showacronymlists:new</code>	236	<code>\Acrlong:</code>	made robust 131
<code>\showglocounter:new</code>	233	<code>\acrlong:</code>	made robust 130
<code>\showglodesc:new</code>	235	<code>\ACRlongpl:</code>	made robust 133
<code>\showglodescplural:new</code>	235	<code>\Acrlongpl:</code>	made robust 133
<code>\showglofirst:new</code>	233	<code>\acrlongpl:</code>	made robust 132
<code>\showglofirstpl:new</code>	233	<code>\ACRshort:</code>	made robust 128
<code>\showgloflag:new</code>	236	<code>\Acrshort:</code>	made robust 127
<code>\showgloindex:new</code>	236	<code>\acrshort:</code>	made robust 127
<code>\showglolevel:new</code>	232	<code>\ACRshortpl:</code>	made robust 130
<code>\showgloname:new</code>	235	<code>\Acrshortpl:</code>	made robust 129
<code>\showgloparent:new</code>	232	<code>\acrshortpl:</code>	made robust 128
<code>\showgloplural:new</code>	233	<code>\Gls:</code>	made robust 108
<code>\showglosort:new</code>	235		

\glsadd: made robust	142	3.02	
\glsaddall: made robust	143		\@do@wrglossary: changed
\GLSdesc: made robust	118		\@glslocref to \theglsentrycounter
\Glsdesc: made robust	118		164
\glsdesc: made robust	118		\@do@wrglossary: changed
\GLSdescplural: made robust	119		\@do@wrglossary to test for
\Glsdescplural: made robust	119		indexonlyfirst option; put old
\glsdescplural: made robust	119		\@do@wrglossary code into
\glsfirst: made robust	114		\@do@wrglossary 161
\GLSfirstplural: made robust	117		\@gls@missingnumberlist:
\Glsfirstplural: made robust	116		new 64
\glsfirstplural: made robust	116		\@glswritefiles: added check
\glslink: made robust	95		for existence of token in case
\GLSname: made robust	117		\makeglossaries has been
\Glsname: made robust	117		omitted 159
\glsname: made robust	117		\@printglossary: add a way to
\GLSpl: made robust	112		fetch current entry label 169
\Glspl: made robust	111		savenumberlist: new 7
\glspl: made robust	110		ucmark: new 9
\GLSplural: made robust	116		\gls@defglossaryentry: added
\GLSsymbol: made robust	120		numberlist element 76
\Glsymbol: made robust	120		\gls@save@numberlist: new 167
\glsymbol: made robust	120		\gls@wrglossary: added check
\GLSsymbolplural: made robust			for glossary file defined 161
	121		\glsdisplaynumberlist: new 141
\Glsymbolplural: made robust			\glsentrycounter: set default
	121		value 97
\glsymbolplural: made robust			\Glsentryfull: fixed bug (re-
	120		placed \glsentryshortpl
\Glstext: made robust	114		with \glsentryshort) 140
\glstext: made robust	113		\glsentryfullpl: fixed bug (re-
\GLSuseri: made robust	122		placed \glsentryshort with
\Glsuseri: made robust	122		\glsentryshortpl) 140
\glsuseri: made robust	121		\glsentrynumberlist: new 141
\GLSuserii: made robust	123		\glsmoveentry: new 77
\Glsuserii: made robust	123		\glsnumlistlastsep: new 142
\glsuserii: made robust	122		\glsnumlistsep: new 142
\GLSuseriii: made robust	124		\glsresetsubentrycounter:
\Glsuseriii: made robust	123		new 185
\glsuseriii: made robust	123		\ifglshaschildren: new 52
\GLSuseriv: made robust	125		\ifglshasparent: new 52
\Glsuseriv: made robust	124		\makeglossaries: added list
\glsuseriv: made robust	124		parser 154
\GLSuserv: made robust	126		indexonlyfirst: new 23
\Glsuserv: made robust	125		\renewglossarystyle: new 193
\glsuserv: made robust	125		\showglossaryentries: fixed
\GLSuservi: made robust	126		misspelt command 237
\Glsuservi: made robust	126		\SmallNewAcronymDef: fixed
\glsuservi: made robust	126		broken short and long plural 226

3.03		
\@gls@sanitizesort:new	18	superragged: added check for glsnogroupskip 279
\@gls@setupsort@standard: used \@gls@sanitizesort .	10	superragged3col: added check for glsnogroupskip 281
\@printglossary: allow title to override default toctitle	169	3.04
General: allow title to set toctitle	181	\@@do@wrglossary: changed \theglsentrycounter back to \@glslocref 164
\glsinlinedescformat:new .	252	\@@do@wrglossary: modified to compensate for possible incor- rect page number 163
\glsinlineemptydescformat: new	252	\@gls@escbsdq: unsani- tize \gls@numberpage, \gls@alphpage, \gls@Alphpage and \gls@romanpage 99
\glsinlinenameformat:new .	252	\@print@glossary: Moved aux write to end of document to prevent unwanted whatsit oc- curring here. 170
\glsinlinepostchild:new ..	252	General: Added check for doc package 4
\glsinlinesubdescformat: new	252	added datatool-base as a re- quired package 4
\glsinlinesubnameformat: new	252	added local key 93
\glspostinline: replaced “.” with \glspostdescription	252	\gls@Alphpage:new 162
altlongragged4col: added check for glsnogroupskip ..	266	\gls@alphpage:new 162
altsuperragged4col: added check for glsnogroupskip ..	283	\gls@disablepagerefexpansion: new 161
alttree: added check for glsnogroupskip 291		\gls@numberpage:new 162
index: added check for glsnogroupskip 285		\gls@protected@pagefmts: new 161
nogroupskip:new 9		\gls@romanpage:new 162
long: added check for glsnogroupskip 257		\glsdefmain: added check for doc package 12
long3col: added check for glsnogroupskip 258		\glsorg@endtheglossary:new . 5
long4col: added check for glsnogroupskip 260		\glsorg@theglossary:new 5
longragged: added check for glsnogroupskip 263		altlist: replaced \newline with paragraph break 254
longragged3col: added check for glsnogroupskip 264		\PrintChanges:new 5
nopostdot:new 9		3.05
tree: added check for glsnogroupskip 287		\@@do@wrglossary: add Roman case. Fixed bugs in the else statements 163
treenoname: added check for glsnogroupskip 288		\@gls@link: added check for “no- hypertypes” 96
super: added check for glsnogroupskip 272		\@gls@nohyperlist:new 16
super3col: added check for glsnogroupskip 274		mcolalttree: replaced ‘2’ with \glsmcols 270
super4col: added check for glsnogroupskip 276		

mcolindex: replaced '2' with \glsmcols 268	removed definition of \@glossarysubentryfield 331
mcoltree: replaced '2' with \glsmcols 269	\compatibleglossentry:new 187
mcoltreename: replaced '2' with \glsmcols 270	\compatiblesubglossentry: new 188
\gls@protected@pagefmts: added Roman to list 161	\glossaryentryfield: depre- cated 189
\gls@Romanpage:new 162	\Glossentrydesc:new 188
\GlsDeclareNoHyperList:new 16	\glossentrydesc:new 188
\glsgetgrouplabel: fixed bug (typo in \equal) 191	\Glossentryname:new 188
\nopostdesc: made robust 33	\glossentryname:new 187
nohypertypes:new 16	\Glossentrysymbol:new 188
3.06	\glossentrysymbol:new 188
\@xdy@main@language: Changed back to using \language name 25	\gls@assign@desc@field:new 18
\findrootlanguage: Obsoleted 48	\gls@assign@descplural@field: new 18
3.07	\gls@assign@field:new 66
\@gls@link: fixed bug that failed to find entry in list 96	\gls@ifnotmeasuring:new ... 80
\glossarypreamble: modified to work with \setglossarypreamble 37	\glsaddallunused:new 143
\gls@doclearpage: added check for openright 39	\glsexpandfields:new 66
\glspostdescription: Added spacefactor code 9	\glsnoexpandfields:new 66
\GlsSetXdyCodePage: Added check for fontspec 49	\glssee: made robust 166
\SetDescriptionAcronymDisplayStyle: now using \glsdoparenifnotempty 220	\glsseeformat: made robust .. 166
\setglossarypreamble:new .. 37	\glsseeitem: made robust 167
3.08a	\glsseelist: made robust 166
\@glo@storeentry: no longer need to check for special char- acters in any of the fields other than sort 78	\ifglsdessuppressed:new .. 53
updated for \glossentry 79	\ifglshasdesc:new 53
\@glossaryentryfield:switched to \glossentry 78	\ifglshassymbol:new 53
\@glossarysubentryfield: switched to \subglossentry 78	list: updated list style to use \glossentry and \subglossentry 253
General: added nogroupskip key to \printglossary 182	listdotted: updated listdotted style to use \glossentry and \subglossentry 255
removed definition of \@glossaryentryfield .. 331	altlist: updated altlist style to use \glossentry and \subglossentry 254
	altlongragged4col: updated to use \glossentry and \subglossentry 266
	alttree: updated to use \glossentry and \subglossentry 289
	index: added paragraph break at end of environment 285
	updated to use \glossentry and \subglossentry 285

change to using \glentryfmt
style commands 109

removed \MakeUppercase
(now moved to \glentryfmt)
..... 110

\@GLSp1: add \glslabel,
\glrifplural, \glscapscase,
\glscustomtext and
\glinsert 112

change to using \glentryfmt
style commands 112

removed \MakeUppercase
as now dealt with in
\glentryfmt 112

\@GLs@: add \glrifplural,
\glscapscase, \glscustomtext
and \glinsert 109

change to using \glentryfmt
style commands 109

removed \makefirstuc (now
dealt with in \glentryfmt) 109

\@GLspl@: add \glrifplural,
\glscapscase, \glscustomtext
and \glinsert 111

change to using \glentryfmt
style commands 111

removed \makefirstuc (now
dealt with in \glentryfmt) 111

\@acrlong: added \glslabel,
\glrifplural, \glscapscase,
\glinsert and \glscustomtext
..... 330

\@acrshort: added \glslabel,
\glrifplural, \glscapscase,
\glinsert and \glscustomtext
..... 329

\@gls@: add \glslabel,
\glrifplural, \glscapscase,
\glscustomtext and
\glinsert 108

change to using \glentryfmt
style commands 108

\@gls@noexpand@fields: Fixed
bug expand replaced with
noexpand 64

\@glsdisp: add \glslabel,
\glrifplural, \glscapscase,
\glscustomtext and
\glinsert 113

change to using \glentryfmt
style commands 113

\@glspl@: add \glslabel,
\glrifplural, \glscapscase,
\glscustomtext and
\glinsert 110

change to using \glentryfmt
style commands 110

General: added \glslabel,
\glrifplural, \glscapscase,
\glinsert and \glscustomtext
..... 127–134

changed to just use \Glsentrydescplural
..... 119

changed to just use \glentrydescplural
..... 119

changed to just use \Glsentrydesc
..... 118

changed to just use \glentrydesc
..... 118, 119

changed to just use \Glsentryfirstplural
..... 116

changed to just use \glentryfirstplural
..... 116, 117

changed to just use \Glsentryfirst
..... 115

changed to just use \glentryfirst
..... 114, 115

changed to just use \Glsentryname
..... 117

changed to just use \glentryname
..... 117, 118

changed to just use \Glsentryplural
..... 116

changed to just use \glentryplural
..... 115, 116

changed to just use \Glsentrysymbolplural
..... 121

changed to just use \glentrysymbolplural
..... 121

changed to just use \Glsentrysymbol
..... 120

changed to just use \glentrysymbol
..... 120

Changed to just use
\Glsentrytext 114

changed to just use \glentrytext
..... 113

changed to just use \Glsentryuseriii 124	\glsnavigation: switched to using \@gls@getgrouptitle 250
changed to just use \Glsentryuseriii 123, 124	\ifglshasdesc: replaced \ifdefempty with \ifcsemtyp 53
changed to just use \Glsentryuserii 123	\ifglshaslong: new 53
changed to just use \Glsentryuserii 122, 123	\ifglshasshort: new 53
changed to just use \Glsentryuseriv 125	\ifglshassymbol: replaced \ifdefempty with \ifcsemtyp 53
changed to just use \Glsentryuseriv 124, 125	\ifglused: replaced \ifthenelse with \ifbool 51
changed to just use \Glsentryuseri 122	\longnewglossaryentry: new . 71
changed to just use \Glsentryuseri 122	\ns@newglossary: replaced \glsdisplay and \glsdisplayfirst with \glsentryfmt 57
changed to just use \Glsentryuservi 126	compatible-3.07: cnew 27
changed to just use \Glsentryuservi 126, 127	\SetCustomDisplayStyle: up- dated to use \defglentryfmt 230
changed to just use \Glsentryuserv 125	\SetDefaultAcronymDisplayStyle: changed to use \defglentryfmt 215
changed to just use \Glsentryuserv 125, 126	\SetDescriptionAcronymDisplayStyle: updated to use \defglentryfmt 220
Now requires textcase 4	\SetDescriptionDUAAcronymDisplayStyle: updated to use \defglentryfmt 219
acronymlists: replaced \@addtoacronymlists with \DeclareAcronymList 15	\SetDescriptionFootnoteAcronymDisplayStyle: updated to use \defglentryfmt 216
\defgldisplay: obsoleted 92	\SetDUADisplayStyle: updated to use \defglentryfmt .. 228
\defgldisplayfirst: obso- leted 92	\SetFootnoteAcronymDisplayStyle: updated to use \defglentryfmt 223
\defglentryfmt: new 55	\SetSmallAcronymDisplayStyle: updated to use \defglentryfmt 225
\forglentries: replaced \ifx with \ifdefempty 50	\setupglossaries: new 28
\gls@assign@desc: new 71	\showglolong: new 236
\gls@defglossaryentry: Fixed default counter if none sup- plied 75	\showgloshort: new 236
\gls@doentryfmt: new 55	numbers: new 27
\glsdisplay: obsoleted 92	symbols: new 27
\glsdisplayfirst: obsoleted .. 91	
\glsgenentryfmt: new 86	
\glsgetgrouptitle: Added check in case non-Latin alpha- bet in use 191	3.12a
\glsglossarymark: replaced \MakeUppercase with \mfirstucMakeUppercase . 38	\gls@defglossaryentry: added \glslabel 72
	\glsaddkey: new 69

3.13a		\glsetnoexpandfield: new .. 17
\@gls@assign@symbol@field:	altsuper4colheader: switched	
changed to use \glsetnoexpandfield	to \tabularnewline	277
..... 18	altsuper4colheaderborder:	
\@gls@assign@symbolplural@field:	switched to \tabularnewline	
changed to use \glsetnoexpandfield	278
..... 18	long: switched to \tabularnewline	
\@gls@link: removed \relax .. 97	257
\@gls@notranslatorhook: new 21	long3col: switched to \tabularnewline	
\@gls@setupsort@standard:	258
moved \@gls@santizesort	long3colheader: switched to	
to \glsprestandardsort .. 10	\tabularnewline	259
ucmark: added check for memoir . 9	long3colheaderborder: switched	
see: added \gls@checkseeallowed	to \tabularnewline	259
..... 61	long4col: switched to \tabularnewline	
\glossarysection: changed	259
\glossarymark to \glsglossarymark	long4colheader: switched to	
..... 37	\tabularnewline	260
\glossarystyle: fixed bug	longheader: switched to	
caused by using \ifdef in-	\tabularnewline	257
stead of \ifcsdef	longheaderborder: switched to	
192	\tabularnewline	258
\gls@assign@desc@field:	\SetFootnoteAcronymDisplayStyle:	
changed to use \glsetnoexpandfield	fixed missing argument bug	223
..... 18	super: switched to \tabularnewline	
\gls@assign@descplural@field:	272
changed to use \glsetnoexpandfield	super3col: switched to	
..... 18	\tabularnewline	274
\gls@assign@name@field:	super3colheader: switched to	
changed to use \glsetnoexpandfield	\tabularnewline	274
..... 18	super4col: switched to	
\gls@assign@type@field:	\tabularnewline	275
changed to use \glsetexpandfield	super4colheader: switched to	
..... 17	\tabularnewline	276
\gls@checkseeallowed: new .. 61	super4colheaderborder:	
\glsaddallunused: set default to	switched to \tabularnewline	
\@glo@types	277
143	superheader: switched to	
\Glsentryfull: changed to use	\tabularnewline	273
\acrfullformat	superheaderborder: switched to	
140	\tabularnewline	273
\Glsentryfullpl: changed to		
use \acrfullformat		
141		
\Glsentryfullpl: changed to	3.14a	
use \acrfullformat	\@glswritefiles: renamed	
140	\glswritefiles to \@glswritefiles	
\glsglossarymark: renamed	and used "savewrites" option	
\glossarymark to \glsglossarymark	to set \glswritefiles	159
to avoid conflict with memoir 38	General: new	238
\glsprestandardsort: new ... 10	acronyms: new	14
\glsetexpandfield: new 17		

\gls@defglossaryentry: added check for existence of default glossary 73 set the default for firstplural to be the value of plural 75	\glsentryfull: bug fix: added missing\acronymfont 140
xindygloss:new 26	\Glsentryfullpl: bug fix: added missing\acronymfont 141
\longprovideglossaryentry: new 72	\glsentryfullpl: bug fix: added missing\acronymfont 140
compatible-2.07: added check for 2.07 before setting 3.07 compatibility 27	\glsgenacfmt:new 89
notranslate:new 22	\GlsUseAcrEntryDisplayStyle: new 204
\provideglossaryentry:new . 66	\GlsUseAcrStyleDefs:new .. 204
4.0	short-long:new 205
\gls@defglossaryentry: added check for first key 75	short-long-desc:new 207
4.01	xindynoglsnumbers:new 26
General: fixed non-value options so that they can be passed to document class 7	sm-short-long:new 206
\CustomAcronymFields: in- serted missing comma 231	sm-short-long-desc:new ... 208
4.02	\makeglossaries: made pream- ble only 154
\@acrfull: now using \acrfullfmt 198	index:new 28
\@gls@indexdef:new 28	\newacronymstyle:new 203
\@gls@numbersdef:new 27	long-sc-short:new 205
\@gls@symbolsdef:new 27	long-sc-short-desc:new ... 206
General: Removed \acronymfont 131–134	long-short:new 204
\ACRfullfmt:new 199	long-short-desc:new 206
\Acrfullfmt:new 199	long-sm-short:new 205
\acrfullfmt:new 198	long-sm-short-desc:new ... 207
\ACRfullplfmt:new 200	footnote:new 210
\Acrfullplfmt:new 200	footnote-desc:new 212
\acrfullplfmt:new 200	footnote-sc:new 211
\acronymentry:new 202	footnote-sc-desc:new 212
sanitize: fixed bug that caused an error here 21	footnote-sm:new 212
sc-short-long:new 206	footnote-sm-desc:new 212
sc-short-long-desc:new ... 207	\setacronymstyle:new 203
\Genacrfullformat:new 91	\SetDescriptionAcronymDisplayStyle: Moved check for empty cus- tom text to prevent unwanted parenthetical material 220
\genacrfullformat:new 91	\SetDescriptionFootnoteAcronymDisplayStyle: Moved check for empty cus- tom text to prevent unwanted parenthetical material 217
\GenericAcronymFields:new 202	\SetFootnoteAcronymDisplayStyle: Moved check for empty cus- tom text to prevent unwanted parenthetical material 223
\Genplacrfullformat:new ... 91	\SetGenericNewAcronym:new 201
\genplacrfullformat:new ... 91	\SetSmallAcronymDisplayStyle: Moved check for empty cus- tom text to prevent unwanted parenthetical material 225
\Glsentryfull: bug fix: added missing\acronymfont 140	

dua: new	208	removed \glslabel (defined in	
dua-desc: new	210	\@gls@link)	127
numberedsection:	added	sc-short-long-desc: redefined	
nameref option	6	to use accessibility informa-	
4.03		tion	335
\@do@wrglossary:	added	\compatibleglossentry: added	
\glsdetoklabel	164	\glsdetoklabel	312
\@ACRlong: removed \glslabel		\compatiblesubglossentry:	
(defined in \@gls@link) ...	331	added \glsdetoklabel ...	312
\@ACRshort: removed \glslabel		\Genacrfullformat: redefined	
(defined in \@gls@link) ...	330	to use accessibility informa-	
\@ACrlong: removed \glslabel		tion	329
(defined in \@gls@link) ...	331	\genacrfullformat: redefined	
\@ACRshort: removed \glslabel		to use accessibility informa-	
(defined in \@gls@link) ...	330	tion	328
\@GLS@: removed \glslabel (de-		\Genplacrfullformat: rede-	
defined in \@gls@link)	109	defined to use accessibility in-	
\@GLSpl: removed \glslabel		formation	329
(defined in \@gls@link) ...	112	\genplacrfullformat: rede-	
\@GLS@: removed \glslabel (de-		defined to use accessibility in-	
defined in \@gls@link)	109	formation	329
\@GLS@entry@field: new	134	\glossentryname: added	
\@GLSpl@: removed \glslabel		\glsdetoklabel	187
(defined in \@gls@link) ...	111	\gls@defglossaryentry: added	
\@acrlong: removed \glslabel		\glsdetoklabel	72
(defined in \@gls@link) ...	330	replaced #1 with \@gls@label	73
\@acrshort: removed \glslabel		replaced \ifthenelse with	
(defined in \@gls@link) ...	329	\ifdefequal	74
\@gls@: removed \glslabel (de-		\glsadd: added \glsdetoklabel	
defined in \@gls@link)	108	142
\@gls@access@display: new .	318	\glsaddkey: switched to using	
\@gls@entry@field: new	134	\@gls@field@link	69
\@gls@fetchfield: new	68	\glsdetoklabel: new	51
\@gls@field@link: new	113	\glsdisplaynumberlist: added	
\@gls@link: added \glsdetoklabel		\glsdetoklabel	141
.....	96	\glsdoifexistsorwarn: new ..	52
moved \@gls@link@opts		\glsentryaccess: switched to	
and \@gls@link@label to		using \@gls@entry@field .	316
\@gls@link	96	\glsentrydescaccess: switched	
\@gls@writedef: added		to using \@gls@entry@field	317
\glsdetoklabel	67	\glsentrydescpluralaccess:	
\@glsdisp: removed \glslabel		switched to using \@gls@entry@field	
(defined in \@gls@link) ...	113	317
\@glspl@: removed \glslabel		\glsentryfirstaccess: switched	
(defined in \@gls@link) ...	110	to using \@gls@entry@field	316
\@printglossary: added		\glsentryfirstplural: added	
\glsdetoklabel	169	\glsdetoklabel	138
General: changed default to		\glsentrylongaccess: switched	
\@empty instead of \relax ..	26	to using \@gls@entry@field	317

<code>\glsentrylongpluralaccess:</code>	short-long-desc: redefined to
switched to using <code>\@gls@entry@field</code>	use accessibility information 335
..... 317	<code>\ifglsgdescsuppressed:</code> added
<code>\glsentrypluralaccess:</code>	<code>\glsdetoklabel</code> 53
switched to using <code>\@gls@entry@field</code>	fixed typo 53
..... 316	<code>\ifglsgentryexists:</code> added
<code>\glsentryshortaccess:switched</code>	<code>\glsdetoklabel</code> 51
to using <code>\@gls@entry@field</code> 317	<code>\ifglsgshaschildren:</code> added
<code>\glsentryshortpluralaccess:</code>	<code>\glsdetoklabel</code> 52
switched to using <code>\@gls@entry@field</code>	<code>\ifglsgshasdesc:added\glsdetoklabel</code>
..... 317 53
<code>\glsentrysymbolaccess:</code>	<code>\ifglsgshasfield:new</code> 54
switched to using <code>\@gls@entry@field</code>	<code>\ifglsgshaslong:added\glsdetoklabel</code>
..... 316 53
<code>\glsentrysymbolpluralaccess:</code>	<code>\ifglsgshasparent:</code> added
switched to using <code>\@gls@entry@field</code>	<code>\glsdetoklabel</code> 52
..... 317	<code>\ifglsgshasshort:</code> added
<code>\glsentrytextaccess:switched</code>	<code>\glsdetoklabel</code> 53
to using <code>\@gls@entry@field</code> 316	<code>\ifglsgshassymbol:</code> added
<code>\glsgenacfmt:</code> redefined to use	<code>\glsdetoklabel</code> 53
accessibility information ... 326	replaced <code>\ifcseempty</code> with
<code>\glsgenentryfmt:</code> redefined to	<code>\ifdefempty</code> and replaced
use accessibility information 324	<code>\ifx</code> with <code>\ifdefequal</code> 53
<code>\glshyperlink:added\glsdetoklabel</code>	<code>\ifglsgused:added\glsdetoklabel</code>
..... 142 51
<code>\glslocalreset:</code> added	<code>sm-short-long-desc:</code> redefined
<code>\glsdetoklabel</code> 80	to use accessibility informa-
<code>\glslocalunset:</code> added	tion 336
<code>\glsdetoklabel</code> 81	<code>long-sc-short-desc:</code> redefined
<code>\glsmoveentry:added\glsdetoklabel</code>	to use accessibility informa-
..... 77	tion 334
replaced <code>\ifthenelse</code> with	<code>long-short:</code> redefined to use ac-
<code>\ifdefequal</code> 78	cessibility information 333
<code>\glsrefentry:added\glsdetoklabel</code>	<code>long-short-desc:</code> redefined to
..... 186	use accessibility information 334
<code>\glsreset:added\glsdetoklabel</code>	<code>long-sm-short-desc:</code> redefined
..... 80	to use accessibility informa-
<code>\glsseelist:added\expandafter</code>	tion 335
commands 166	<code>footnote:</code> redefined to use acces-
<code>\glsstepentry:added\glsdetoklabel</code>	sibility information 339
..... 185	<code>footnote-desc:</code> redefined to use
<code>\glsstepsubentry:</code> added	accessibility information ... 341
<code>\glsdetoklabel</code> 185	<code>footnote-sc:</code> redefined to use ac-
<code>\glsunset:added\glsdetoklabel</code>	cessibility information 340
..... 80	<code>footnote-sc-desc:</code> redefined to
<code>short-long:</code> commented spuri-	use accessibility information 341
ous EOL 205	<code>footnote-sm:</code> redefined to use ac-
redefined to use accessibility in-	cessibility information 341
formation 333	

footnote-sm-desc: redefined to use accessibility information	342	\showglosort:added\glsdetoklabel	235
\renewacronymstyle:new	203	\showglosymbol:	added
\showglocounter:	added	\glsdetoklabel	235
\showglodesc:added\glsdetoklabel	235	\showglosymbolaccess:	added
\showglodescaccess:	added	\glsdetoklabel	348
\glsdetoklabel	348	\showglosymbolplural:	added
\showglodescplural:	added	\glsdetoklabel	235
\glsdetoklabel	235	\showglosymbolpluralaccess:	added\glsdetoklabel
\showglodescpluralaccess:	added\glsdetoklabel	348	
\showglofirst:added\glsdetoklabel	233	\showglotext:added\glsdetoklabel	233
\showglofirstaccess:	added	\showglotextaccess:	added
\glsdetoklabel	347	\glsdetoklabel	347
\showglofirstpl:	added	\showglotype:added\glsdetoklabel	233
\glsdetoklabel	233	\showglouserii:added\glsdetoklabel	234
\showglofirstpluralaccess:	added\glsdetoklabel	234	
347		\showglouseriii:	added
\showgloflag:added\glsdetoklabel	236	\glsdetoklabel	234
\showgloindex:added\glsdetoklabel	236	\showglouseriv:	added
\showglolevel:added\glsdetoklabel	232	\glsdetoklabel	234
\showglolevel:	232	\showglouserv:added\glsdetoklabel	234
\showglolong:added\glsdetoklabel	236	\showglouservi:	added
\showglolongaccess:	added	\glsdetoklabel	234
\glsdetoklabel	348	dua: fixed bug in \acrfullfmt	209
\showglolongpluralaccess:	added\glsdetoklabel	fixed bug in \Acrfullplfmt	209
348		fixed bug in \acrfullplfmt	209
\showgloname:added\glsdetoklabel	235	redefined to use accessibility in-	
\showglonameaccess:	added	formation	336
\glsdetoklabel	347	dua-desc: commented spurious	
\showgloparent:	added	EOL	210
\glsdetoklabel	232	redefined to use accessibility in-	
\showgloplural:	added	formation	338
\glsdetoklabel	233	4.04	
\showglopluralaccess:	added	\@@gls@noidx@nosanitizesort:	
\glsdetoklabel	347	new	19
\showgloshort:added\glsdetoklabel	236	\@@gls@noidx@sanitizesort:	
\showgloshortaccess:	added	new	18
\glsdetoklabel	348	\@@gls@nosanitizesort:new	18
\showgloshortpluralaccess:	added\glsdetoklabel	\@@gls@sanitizesort:new	18
348		\@glo@addchildren:new	172
\@glo@do@sortentries:new	173	\@glo@grabfirst:new	178
\@glo@sortedinsert:new	173	\@glo@sortentries:new	171

\@glo@sorthandler@case: new	174	\glsnoidxdisplaylocclisthandler:	
\@glo@sorthandler@letter:		new	180
new	174	\glsnoidxlocclist: new	179
\@glo@sorthandler@nocase:		\glsnoidxlocclisthandler:	
new	174	new	179
\@glo@sorthandler@word: new	173	\glsnoidxstripaccents: new	19
\@glo@sortmacro@case: new	175	alttree: moved hangindent and	
\@glo@sortmacro@def: new	176	parindent assignments out-	
\@glo@sortmacro@def@do: new	176	side level test	289
\@glo@sortmacro@letter: new	175	\makeglossaries: Moved def-	
\@glo@sortmacro@nocase: new	176	inition of \glswrite to	
\@glo@sortmacro@standard:		\makeglossaries	153
new	175	\makenoidxglossaries: new	155
\@glo@sortmacro@use: new	176	\printglossary: changed to use	
\@glo@sortmacro@word: new	174	new \@printglossary ...	168
\@gls@getcounterprefix:		\printnoidxglossaries: new	168
added warning if no prefix can		\printnoidxglossary: new	168
be formed	165	\showglocllist: new	236
\@gls@getothergrouptitle:		\warn@noprintglossary: Acti-	
new	191	vate warning in \makeglossaries	
\@gls@noidx@do: new	178	167
\@gls@noref@warn: new	158	\writeist: checked for definition	
\@gls@reference: new	180	of \glswrite	145, 149
\@gls@warnonglossdefined:		4.06	
new	17	\@GLS@: added \glsifhyper	109
\@gls@warnontheGLOSSdefined:		\@GLSpl: added \glsifhyper	112
new	17	\@Gls@: added \glsifhyper	109
\@no@makeglossaries: new	158	\@Glspl@: added \glsifhyper	111
\@print@glossary: new	170	\@gls@: added \glsifhyper	108
\@print@noidx@glossary: new	177	\@gls@numbersdef: added hook	
\@printgloss@setsort: new	168	to set toc title	28
\@printglossary: new	168	\@gls@symbolsdef: added hook	
General: added sort key to print-		to set toc title	27
gloss group	183	\@glsdisp: added \glsifhyper	113
\compatibleglossentry:		\@glspl@: added \glsifhyper	110
changed \newcommand to		General: added \glsifhyper	
\def as is may or may not be		127–134
defined	312	acronym: added hook to set toc ti-	
\compatiblesubglossentry:		tle	13
changed \newcommand to		acronyms: added hook to set toc	
\def as is may or may not be		title	14
defined	312	\glsdefmain: added hook to set	
\defglsdisplayfirst: fixed un-		toc title	13
wanted space	92	4.07	
\glo@grabfirst: new	177	\@glossarysection: added op-	
\gls@defglossaryentry: re-		tional argument when using	
placed \ifx with \ifdefvoid	77	unstarred version	39
\glsnoidxdisplayloc: new	180		

\@gls@noidx@do: added \global	moved check for first use to
in case it's used in a tabular-	\@gls@link 113
like style 178	\@glspl@: moved \glsifhyper 110
\Acrfullplfmt: fixed no case	moved check for first use to
change bug 200	\@gls@link 110
\glsletentryfield: new 134	\@ignored@glossaries: new .. 59
4.08	General: added entrycounter op-
\@ACRlong: added \do@gls@link@checkfirsthyper	tion to printgloss family . 182
..... 331	added nopostdot option to
\@ACRshort: added \do@gls@link@checkfirsthyper	printgloss family 182
..... 330	added subentrycounter option
\@Acrlong: added \do@gls@link@checkfirsthyper	to printgloss family 182
..... 331	explicitly initialise hyper key .. 93
\@Acrshort: added \do@gls@link@checkfirsthyper	moved \glsifhyper ... 127-134
..... 329	removed \@sACRlongpl 133
\@GLS@: moved \glsifhyper .. 109	removed \@sAcrlongpl 133
moved check for first use to	removed \@sacrlongpl 132
\@gls@link 109	removed \@sACRlong 131
\@GLSpl: moved \glsifhyper . 112	removed \@sAcrlong 131
moved check for first use to	removed \@sacrlong 130
\@gls@link 112	removed \@sACRshortpl ... 130
\@Gls@: moved \glsifhyper .. 109	removed \@sAcrshortpl ... 129
moved check for first use to	removed \@sacrshortpl ... 129
\@gls@link 109	removed \@sACRshort 128
\@GLspl@: moved \glsifhyper 111	removed \@sAcrshort 127
moved check for first use to	removed \@sacrshort 127
\@gls@link 111	removed \@sgls@link 95
\@acrlong: added \do@gls@link@checkfirsthyper	removed \@sGLSdescplural 119
..... 330	removed \@sGlsdescplural 119
\@acrshort: added \do@gls@link@checkfirsthyper	removed \@sglsdescplural 119
..... 329	removed \@sGLSdesc 118
\@closegls: new 152	removed \@sGlsdesc 118
\@gls@: moved \glsifhyper .. 108	removed \@sglsdesc 118
moved check for first use to	removed \@sglsdisp 112
\@gls@link 108	removed \@sGLSfirstplural 117
\@gls@doautomake: new 26	removed \@sGlsfirstplural 116
\@gls@field@link: added as-	removed \@sglsfirstplural 116
signment of \do@gls@link@checkfirsthyper	removed \@sGLSfirst 115
..... 113	removed \@sGlsfirst 115
\@gls@forbidtext: new 55	removed \@sglsfirst 114
\@gls@hyp@opt: new 94	removed \@sGLSname 118
\@gls@link: removed redun-	removed \@sGlsname 117
dancy 96	removed \@sglsname 117
renamed \gls@type to	removed \@sGLSplural 116
\glstype 96	removed \@sGlsplural 115
\@gls@link@checkfirsthyper:	removed \@sglsplural 115
new 95	removed \@sGLSpl 112
\@glsdisp: moved \glsifhyper 113	removed \@sGlspl 111
	removed \@sglspl 110

removed \sGLSsymbolplural	121	switched to using \@gls@hyp@opt	198
removed \sGlsymbolplural	121	\ACRfullpl: removed \s@ACRfullpl	200
removed \sglssymbolplural	121	switched to using \@gls@hyp@opt	200
removed \sGLSsymbol	120	\Acrfullpl: removed \s@Acrfullpl	200
removed \sGlsymbol	120	switched to using \@gls@hyp@opt	200
removed \sglssymbol	120	\acrfullpl: removed \s@acrfullpl	199
removed \sGLStext 114		switched to using \@gls@hyp@opt	199
removed \sGlstext 114		\ACRlong: switched to using \@gls@hyp@opt	131
removed \sglstext 113		\Acrlong: switched to using \@gls@hyp@opt	131
removed \sGLSuseriii ... 124		\acrlong: switched to using \@gls@hyp@opt	130
removed \sGlsuseriii ... 124		\ACRlongpl: switched to using \@gls@hyp@opt	133
removed \sglsuseriii ... 123		\Acrlongpl: switched to using \@gls@hyp@opt	133
removed \sGLSuserii 123		\acrlongpl: switched to using \@gls@hyp@opt	132
removed \sGlsuserii 123		\ACRshort: switched to using \@gls@hyp@opt	128
removed \sglsuserii 122		\Acrshort: switched to using \@gls@hyp@opt	127
removed \sGLSuseriv 125		\acrshort: switched to using \@gls@hyp@opt	127
removed \sGlsuseriv 124		\ACRshortpl: switched to using \@gls@hyp@opt	130
removed \sglsuseriv 124		\Acrshortpl: switched to using \@gls@hyp@opt	129
removed \sGLSuseri 122		\acrshortpl: switched to using \@gls@hyp@opt	128
removed \sGlsuseri 122		\forallacronyms: new	50
removed \sglsuseri 121		\GLS: switched to using \@gls@hyp@opt	109
removed \sGLSuservi 127		\Gls: switched to using \@gls@hyp@opt	108
removed \sGlsuservi 126		\gls: switched to using \@gls@hyp@opt	107
removed \sglsuservi 126		\gls@defglossaryentry: added check for ignored glossary ...	73
removed \sGLSuserv 126		\gls@istfilebase: new	34
removed \sGlsuserv 125			
removed \sglsuserv 125			
removed \sGLS 109			
removed \sGls 108			
removed \sgls 107			
removed \@thirdofthree (de- fined in kernel)	107		
removed sPGLS 243			
removed sPgls 242			
removed spgls 240			
removed sPGLSpl 244			
removed sPglspl 242			
removed spglspl 241			
\ACRfull: removed \s@ACRfull	199		
switched to using \@gls@hyp@opt	199		
\Acrfull: removed \s@Acrfull	198		
switched to using \@gls@hyp@opt	198		
\acrfull: removed \s@acrfull	198		

\glsaddkey: removed \@sGLS@user@<key>	\Glsname: switched to using
..... 71	\@gls@hyp@opt 117
removed \@sGLS@user@<key> . 70	\glsname: switched to using
removed \@sgls@user@<key> . 70	\@gls@hyp@opt 117
switched to using \@gls@hyp@opt	\GLSpl: switched to using
..... 70, 71	\@gls@hyp@opt 112
\GLSdesc: switched to using	\Glspl: switched to using
\@gls@hyp@opt 118	\@gls@hyp@opt 111
\Glsdesc: switched to using	\glspl: switched to using
\@gls@hyp@opt 118	\@gls@hyp@opt 110
\glsdesc: switched to using	\GLSplural: switched to using
\@gls@hyp@opt 118	\@gls@hyp@opt 116
\GLSdescplural: switched to using	\Glsplural: switched to using
\@gls@hyp@opt 119	\@gls@hyp@opt 115
\Glsdescplural: switched to using	\glsplural: switched to using
\@gls@hyp@opt 119	\@gls@hyp@opt 115
\glsdescplural: switched to using	\glsspace: new 198
\@gls@hyp@opt 119	\GLSsymbol: switched to using
\glsdisablehyper: added	\@gls@hyp@opt 120
\KV@glslink@hyperfalse to	\Glsymbol: switched to using
definition 107	\@gls@hyp@opt 120
\glsdisp: switched to using	\glssymbol: switched to using
\@gls@hyp@opt 112	\@gls@hyp@opt 120
\glsdohyperlink: new 106	\GLSsymbolplural: switched to
\glsdohypertarget: new 106	using \@gls@hyp@opt 121
\glsenablehyper: added	\Glsymbolplural: switched to
\KV@glslink@hypertrue to	using \@gls@hyp@opt 121
definition 107	\glssymbolplural: switched to
\GLSfirst: switched to using	using \@gls@hyp@opt 120
\@gls@hyp@opt 115	\GLStext: switched to using
\Glsfirst: switched to using	\@gls@hyp@opt 114
\@gls@hyp@opt 114	\Glstext: switched to using
\glsfirst: switched to using	\@gls@hyp@opt 114
\@gls@hyp@opt 114	\glstext: switched to using
\GLSfirstplural: switched to	\@gls@hyp@opt 113
using \@gls@hyp@opt 117	\glstreenamfmt: new 284
\Glsfirstplural: switched to	\GLSuseri: switched to using
using \@gls@hyp@opt 116	\@gls@hyp@opt 122
\glsfirstplural: switched to	\Glsuseri: switched to using
using \@gls@hyp@opt 116	\@gls@hyp@opt 122
\glsifhyper: deprecated 94	\glsuseri: switched to using
\glslink: switched to using	\@gls@hyp@opt 121
\@gls@hyp@opt 95	\GLSuserii: switched to using
\glslinkcheckfirsthyperhook:	\@gls@hyp@opt 123
new 96	\Glsuserii: switched to using
\glslinkvar: new 94	\@gls@hyp@opt 123
\GLSname: switched to using	\glsuserii: switched to using
\@gls@hyp@opt 117	\@gls@hyp@opt 122

\GLSuseriii: switched to using \@gls@hyp@opt 124		\s@newglossary: new 56
\Glsuseriii: switched to using \@gls@hyp@opt 123		automake: new 26
\glsuseriii: switched to using \@gls@hyp@opt 123	4.09	\glsaddkey: fixed bug in user commands 70
\GLSuseriv: switched to using \@gls@hyp@opt 125	4.10	
\Glsuseriv: switched to using \@gls@hyp@opt 124		\@Gls@acrentryname: new ... 135
\glsuseriv: switched to using \@gls@hyp@opt 124		\@Gls@entryname: new 135
\GLSuserv: switched to using \@gls@hyp@opt 126		\@gls@glossary: Renamed \@glossary to \@gls@glossary 160
\Glsuserv: switched to using \@gls@hyp@opt 125		\glspercentchar: new 144
\glsuserv: switched to using \@gls@hyp@opt 125		\glstildechar: new 144
\GLSserv: switched to using \@gls@hyp@opt 126		almtree: moved space after sym- bol 290, 291
\Glserv: switched to using \@gls@hyp@opt 125	4.11	
\glserv: switched to using \@gls@hyp@opt 125		\@@do@wrglossary: added hook 163
\GLSservi: switched to using \@gls@hyp@opt 126		sanitize: none option 21
\Glservi: switched to using \@gls@hyp@opt 126		\gls@wrglossary: renamed from \@wrglossary to \gls@wrglossary 161
\glservi: switched to using \@gls@hyp@opt 126		\glsaddprotectedpagefmt: new 162
\ifignoredglossary: new 59		\glsbackslash: new 144
altnongragged4col: fixed bug that displayed description in- stead of symbol 266	4.12	
\newglossary: added starred ver- sion 56		\@gls@addpredefinedattributes: Added glsignore attribute ... 43
\newignoredglossary: new ... 58		\@gls@adjustmode: new 143
\ns@newglossary: added \@glotype@<name>@log ... 57		\@gls@notranslatorhook: re- moved 21
new 56		\@gls@toc: added \protect to \numberline 40
\p@gls@hyp@opt: new 94		\@gls@usetranslator: new ... 22
\PGLS: changed to use \@gls@hyp@opt 243		\glsacrpluralsuffix: new ... 31
\Pgls: changed to use \@gls@hyp@opt 242		\glsadd: added check for vertical mode 142
\pgls: changed to use \@gls@hyp@opt 240		\glsaddallunused: replaced @gobble with glsignore 143
\PGLSpl: changed to use \@gls@hyp@opt 244		\glsifusedtranslatordict: new 22
\Pglspl: changed to use \@gls@hyp@opt 242		\glsignore: new 143
\pglspl: changed to use \@gls@hyp@opt 241		\glsupacrpluralsuffix: new . 31
\s@gls@hyp@opt: new 94	4.13	\ProvidesGlossariesLang: new 31
		\RequireGlossariesLang: new 31
		\indexspace: new ... 253, 268, 284

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	
\@do@wrglossary	164
\@do@wrglossary	163
\@glo@assign@sortkey	184
\@glossarysec	6
\@glossaryseclabel	6
\@glossarysecstar	6
\@gls@default@entryfmt	320
\@gls@expand@field	65
\@gls@fixbraces	166
\@gls@noidx@nosanitizesort .	19
\@gls@noidx@sanitizesort ...	18
\@gls@nosanitizesort	18
\@gls@sanitizesort	18
\@ACRlong	331
\@ACRshort	330
\@Acrlong	331
\@Acrshort	329
\@GLS@	109
\@GLSpl	112
\@Gls@	108
\@Gls@acrentryname	135
\@Gls@entry@field	134
\@Gls@entryname	135
\@Glspl@	111
\@PGLS	243
\@PGLS@	243
\@PGLSpl	244
\@PGLSpl@	244
\@PglS	242
\@PglS@	242
\@PglSpl	242
\@PglSpl@	243
\@acrfull	198
\@acrlong	330
\@acrshort	329
\@addtoacronymlists	14
\@closegls	152
\@delimN	194
\@delimR	194
\@disable@onlypremakeg	30
\@disable@premakecs	30
\@disabled@glsaddxdycounters	42
\@do@seeglossary	165
\@do@wrglossary	161, 293
\@glo@addchildren	172
\@glo@default@sorttype	10
\@glo@do@sortentries	173
\@glo@grabfirst	178
\@glo@no@assign@sortkey	183
\@glo@seeautonumberlist	8
\@glo@sortedinsert	173
\@glo@sortentries	171
\@glo@sorthandler@case	174
\@glo@sorthandler@letter ...	174
\@glo@sorthandler@nocase ...	174
\@glo@sorthandler@word	173
\@glo@sortmacro@case	175
\@glo@sortmacro@def	176
\@glo@sortmacro@def@do	176
\@glo@sortmacro@letter	175
\@glo@sortmacro@nocase	176
\@glo@sortmacro@standard ...	175
\@glo@sortmacro@use	176
\@glo@sortmacro@word	174
\@glo@storeentry	78
\@glo@types	55
\@glossary@default@style	7
\@glossaryentryfield	78
\@glossarysection	39
\@glossarysubentryfield	78
\@gls	107
\@gls@	108
\@gls@@link	95
\@gls@access@display	318
\@gls@addpredefinedattributes	43
\@gls@adjustmode	143
\@gls@assign@symbol@field ...	18
\@gls@assign@symbolplural@field	18
\@gls@checkactual	104
\@gls@checkbar	103
\@gls@checkescactual	101
\@gls@checkescbar	102
\@gls@checkesclevel	103

<code>\@gls@checkescquote</code>	101	<code>\@gls@sanitizename</code>	18
<code>\@gls@checklevel</code>	104	<code>\@gls@sanitizesort</code>	18
<code>\@gls@checkmkidxchars</code>	100	<code>\@gls@sanitizesymbol</code>	18
<code>\@gls@checkquote</code>	100	<code>\@gls@saveentrycounter</code>	97
<code>\@gls@codepage</code>	48	<code>\@gls@setacrstyle</code>	24
<code>\@gls@counterwithin</code>	10	<code>\@gls@setcounter</code>	58
<code>\@gls@declareoption</code>	7	<code>\@gls@setupsort@def</code>	11
<code>\@gls@default@value</code>	60	<code>\@gls@setupsort@standard</code>	10
<code>\@gls@do@acronymsdef</code>	14	<code>\@gls@setupsort@use</code>	11
<code>\@gls@doautomake</code>	26	<code>\@gls@startswithexpandonce</code>	65
<code>\@gls@entry@field</code>	134	<code>\@gls@symbolsdef</code>	27
<code>\@gls@escbsdq</code>	99	<code>\@gls@tmpb</code>	100
<code>\@gls@expand@fields</code>	65	<code>\@gls@toc</code>	40
<code>\@gls@fetchfield</code>	68	<code>\@gls@updatechecked</code>	100
<code>\@gls@field@link</code>	113	<code>\@gls@usetranslator</code>	22
<code>\@gls@fixbraces</code>	166	<code>\@gls@warnonglossdefined</code>	17
<code>\@gls@forbidtexext</code>	55	<code>\@gls@warnonthehglossdefined</code>	17
<code>\@gls@getcounter</code>	58	<code>\@gls@writedef</code>	67
<code>\@gls@getcounterprefix</code>	164	<code>\@gls@xdy@Lclass@Alpha-page-numbers</code>	45
<code>\@gls@getgrouptitle</code>	191	<code>\@gls@xdy@Lclass@Appendix-page-numbers</code>	45
<code>\@gls@getothergrouptitle</code>	191	<code>\@gls@xdy@Lclass@Roman-page-numbers</code>	45
<code>\@gls@glossary</code>	160	<code>\@gls@xdy@Lclass@alpha-page-numbers</code>	45
<code>\@gls@hyp@opt</code>	94	<code>\@gls@xdy@Lclass@arabic-page-numbers</code>	45
<code>\@gls@hypergroup</code>	249	<code>\@gls@xdy@Lclass@arabic-section-numbers</code>	45
<code>\@gls@ifinlist</code>	41	<code>\@gls@xdy@Lclass@roman-page-numbers</code>	44
<code>\@gls@indexdef</code>	28	<code>\@gls@xdy@locationlist</code>	44
<code>\@gls@keymap</code>	68, 238	<code>\@gls@xdy@checkbackslash</code>	105
<code>\@gls@link</code>	96	<code>\@gls@xdy@checkquote</code>	105
<code>\@gls@link@checkfirsthyper</code>	95	<code>\@gls@Alpha compositor</code>	35, 45
<code>\@gls@loadlist</code>	8	<code>\@gls@acronymlists</code>	14
<code>\@gls@loadlong</code>	8	<code>\@gls@defaultplural</code>	63
<code>\@gls@loadsuper</code>	8	<code>\@gls@defaultsort</code>	64
<code>\@gls@loadtree</code>	8	<code>\@gls@disp</code>	112
<code>\@gls@makefirsttuc</code>	245	<code>\@gls@firstletter</code>	144
<code>\@gls@missingnumberlist</code>	64	<code>\@gls@hypernumber</code>	194
<code>\@gls@noaccess</code>	314	<code>\@gls@link</code>	106
<code>\@gls@noexpand@field</code>	64	<code>\@gls@minrange</code>	145
<code>\@gls@noexpand@fields</code>	64	<code>\@gls@nextpages</code>	184
<code>\@gls@nohyperlist</code>	16	<code>\@gls@nodesc</code>	63
<code>\@gls@noidx@do</code>	178	<code>\@gls@noname</code>	63
<code>\@gls@noidx@setsanitizesort</code>	21	<code>\@gls@nonextpages</code>	184
<code>\@gls@noref@warn</code>	158	<code>\@gls@openfile</code>	151
<code>\@gls@notranslatorhook</code>	21		
<code>\@gls@numbersdef</code>	27		
<code>\@gls@onlypremakeg</code>	29		
<code>\@gls@provide@newglossary</code>	55		
<code>\@gls@reference</code>	180		
<code>\@gls@renewglossary</code>	161		
<code>\@gls@sanitizedesc</code>	17		

\@glsorder	25
\@glspl@	110
\@glstarget	106
\@glswidestname	289
\@glswritefiles	159
\@ignored@glossaries	59
\@istfilename	34
\@makeglossary	151
\@mfu@nocaplist	247
\@newglossary	58
\@newglossaryentryposthook ..	77
\@newglossaryentryprehook ...	77
\@no@makeglossaries	158
\@no@post@desc	34
\@nopostdesc	34
\@onlypremakeg	30
\@p@glossarysection	39
\@pgls	241
\@pgls@	241
\@pglspl	241
\@pglspl@	241
\@print@glossary	170
\@print@noidx@glossary	177
\@printgloss@setsort	168
\@printglossary	168
\@set@glo@numformat	98, 294
\@wrglossary@pageformat	162
\@wrglossarynumberhook	162
\@xdy@main@language	25
\@xdy@attributelist	41
\@xdy@attributes	40
\@xdy@language	48
\@xdylettergroups	49
\@xdylocationclassorder	46
\@xdylocref	41
\@xdyrequiredstyles	47
\@xdysortrules	47
\@xdyuseralphabets	44
\@xdyuserlocationdefs	45
\@xdyuserlocationnames	45

A

\Ac	214
\ac	214
access (key)	313
accsupp package	312
\accsuppglossaryentryfield .	332
\accsuppglossarysubentryfield	
.....	332

\Acf	214
\acf	214
\Acfp	214
\acfp	214
\Acl	213
\acl	213
\Aclp	214
\aclp	213
\Acp	214
\acp	214
\acrfootnote	216
\ACRfull	199
\Acrfull	198
\acrfull	198, 202, 211, 339
\ACRfullfmt	199
\Acrfullfmt	199
\acrfullfmt	198
\acrfullformat	89, 198
\ACRfullpl	200
\Acrfullpl	200
\acrfullpl	199
\ACRfullplfmt	200
\Acrfullplfmt	200
\acrfullplfmt	200
\acrlinkfootnote	216
\acrlinkfullformat	198
\ACRlong	131
\Acrlong	131
\acrlong	130
\ACRlongpl	133
\Acrlongpl	133
\acrlongpl	132
\acrnameformat	201, 222
\acrlinkfootnote	216
acronym (option)	13
acronym styles:	
dua	208, 336
dua-desc	210, 338
footnote	210, 339
footnote-desc	212, 341
footnote-sc	211, 340
footnote-sc-desc	212, 341
footnote-sm	212, 341
footnote-sm-desc	212, 342
long-sc-short	205
long-sc-short-desc ..	206, 334
long-short	204, 333
long-short-desc	206, 334
long-sm-short	205

long-sm-short-desc ..	207, 335	altsuperragged4colborder	
sc-short-long	206	(style)	283
sc-short-long-desc ..	207, 335	altsuperragged4colheader	
short-long	205, 333	(style)	283
short-long-desc	207, 335	altsuperragged4colheaderborder	
sm-short-long	206	(style)	284
sm-short-long-desc ..	208, 336	alttree (style)	289
\acronymentry	202	alttreegroup (style)	291
\acronymfont		alttreehypergroup (style)	291
.....	89, 201, 218, 222, 225, 228	amsgen package	4, 93
acronymlists (option)	15	amsmath package	80
\acronymname	30	\andname	31
acronyms (option)	14	array package	262, 278
\acronymsort	203	article class	164
\acronymtype	13, 197	automake (option)	26
\acrpluralsuffix	197		
\ACRshort	128	B	
\Acrshort	127	babel package	22, 30, 32, 48
\acrshort	127		
\ACRshortpl	130	C	
\Acrshortpl	129	\capitalisewords	246
\acrshortpl	128	\compatglossarystyle	298
\Acs	213	compatible-2.07 (option)	27
\acs	213	compatible-3.07 (option)	27
\Acsp	213	\compatibleglossentry ..	187, 312
\acsp	213	\compatiblesubglossentry	188, 312
\addglossarytocaptions	32	counter (key)	61
\addto	32	counter (option)	16
align (environment)	80, 97	\CustomAcronymFields	230
altlist (style)	254	\CustomNewAcronymDef	231
altlistgroup (style)	254		
altlisthypergroup (style)	255	D	
altlong4col (style)	261	datatool package	173
altlong4colborder (style)	261	\DeclareAcronymList	14
altlong4colheader (style)	261	\DefaultNewAcronymDef ..	215, 342
altlong4colheaderborder (style)	261	\defentryfmt	93
altlongragged4col (style)	265	\defglstdisplay	92
altlongragged4colborder (style)	267	\defglstdisplayfirst	92
altlongragged4colheader (style)	266	\defglssentry	57
altlongragged4colheaderborder		\defglssentryfmt	55, 59, 61, 83
(style)	267	\DefineAcronymSynonyms	213
\altnewglossary	57	\delimN	36, 193
altsuper4col (style)	277	\delimR	36, 193
altsuper4colborder (style)	277	description (environment)	253
altsuper4colheader (style)	277	description (key)	59
altsuper4colheaderborder		description (option)	24
(style)	278	descriptionaccess (key)	313
altsuperragged4col (style)	282	\DescriptionDUANewAcronymDef	219
		\DescriptionFootnoteNewAcronymDef	
		217, 343

`\descriptionname` 31
`\DescriptionNewAcronymDef` ..
 221, 344
`descriptionplural (key)` 60
`descriptionpluralaccess (key)` 314
`\detokenize` 51
`doc package` 4, 5, 12
`dua (acrstyle)` 208, 336
`dua (option)` 24
`dua-desc (acrstyle)` 210, 338
`\DUANewAcronymDef` 228

E

`\ecapitalisewords` 247
`\emakefirsttuc` 246
`entrycounter (option)` 9
`entrycounterwithin (option)` 9
`\entryname` 31
 environments:
 `align` 80, 97
 `description` 253
 `longtable` 8, 232, 256–267
 `multicols` 268
 `supertabular` ... 8, 232, 271–284
 `theglossary` 5, 17, 36, 37, 186,
 187, 193, 269, 270, 286, 287, 289
 `theindex` 284
`equation (counter)` 97, 98
`etoolbox package` 4, 244

F

file types
 `.aux` 170
 `.glo` 78
 `.ist` 144, 150, 151
 `.toc` 40
 `.xdy` 34
 `glo` 238
`\findrootlanguage` 48
`first (key)` 60
`firstaccess (key)` 313
`\firstacronymfont` 89, 201
`firstplural (key)` 60
`firstpluralaccess (key)` 313
`footnote (acrstyle)` 210, 339
`footnote (option)` 24
`footnote-desc (acrstyle)` .. 212, 341
`footnote-sc (acrstyle)` 211, 340
`footnote-sc-desc (acrstyle)` 212, 341
`footnote-sm (acrstyle)` 212, 341

`footnote-sm-desc (acrstyle)` 212, 342
`\FootnoteNewAcronymDef` . 223, 345
`\forallallacronyms` 50
`\forallallglossaries` 49
`\forallallglsentries` 50
`\forallrglsentries` 50

G

`garamondx package` 197
`\Genacrfullformat` 91, 329
`\genacrfullformat` 91, 328
`\GenericAcronymFields` 202
`\Genplacrfullformat` 91, 329
`\genplacrfullformat` 91, 329
`\glo@grabfirst` 177
`\glolinkprefix` 97
`glossareentry (counter)` 185
`glossaries package` 28,
 48, 145, 232, 238, 253, 292, 312
`glossaries-accsupp package` ... 78, 312
`\GlossariesWarning` 16
`\GlossariesWarningNoLine` 16
`\glossary` 56, 151, 160, 192
 glossary counters:
 `glossaryentry` 184
 `glossarysubentry` 185
 glossary keys:
 `access` 313
 `counter` 61
 `description` 59
 `descriptionaccess` 313
 `descriptionplural` 60
 `descriptionpluralaccess` . 314
 `first` 60
 `firstaccess` 313
 `firstplural` 60
 `firstpluralaccess` 313
 `long` 63
 `longaccess` 314
 `longplural` 63
 `longpluralaccess` 314
 `name` 59
 `nonumberlist` 62
 `parent` 62
 `plural` 60
 `pluralaccess` 313
 `see` 61
 `short` 63
 `shortaccess` 314

shortplural	63	altsuper4colheader	277, 311
shortpluralaccess	314	altsuper4colheader	277
sort	60	altsuper4colheaderborder	
symbol	61	altsuper4colheaderborder	278, 311
symbolaccess	313	altsuper4colheaderborder	278
symbolplural	61	altsuperragged4col	282–284, 309
symbolpluralaccess	313	altsuperragged4col	282
text	60	altsuperragged4colborder	
textaccess	313	altsuperragged4colborder	283, 310
type	61	altsuperragged4colborder	283
user1	62	altsuperragged4colheader	
user2	62	altsuperragged4colheader	283, 309
user3	62	altsuperragged4colheader	283
user4	62	altsuperragged4colheaderborder	
user5	62	altsuperragged4colheaderborder	284, 310
user6	63	altsuperragged4colheaderborder	
glossary package	1, 196	altsuperragged4colheaderborder	284
glossary styles:		alttree	270, 289, 291, 306
altlist	254, 255, 300	alttree	289
altlist	254	alttreegroup	291, 307
altlistgroup	254, 255, 300	alttreegroup	291
altlistgroup	254	alttreehypergroup	291, 307
altlisthypergroup	255, 300	alttreehypergroup	291
altlisthypergroup	255	index	268, 284–286, 304
altlong4col	261, 265, 302	index	284
altlong4col	261	indexgroup	285, 286, 304
altlong4colborder	261, 302	indexgroup	285
altlong4colborder	261	indexhypergroup	286, 305
altlong4colheader	261, 302	indexhypergroup	286
altlong4colheader	261	inline	299
altlong4colheaderborder		inline	250
	261, 302	list	7, 253–255, 299
altlong4colheaderborder	261	list	253
altlongragged4col	265–267, 303	listdotted	255, 256, 300
altlongragged4col	265	listdotted	255
altlongragged4colborder		listgroup	253, 254, 300
	267, 304	listgroup	253
altlongragged4colborder	267	listhypergroup	254, 300
altlongragged4colheader		listhypergroup	254
	266, 304	long	256–258, 262, 301, 302
altlongragged4colheader	266	long	256
altlongragged4colheaderborder		long3col	258, 259, 301
	267, 304	long3col	258
altlongragged4colheaderborder	267	long3colborder	258, 301
		long3colborder	258
altsuper4col	277, 278, 282, 311	long3colheader	259, 301
altsuper4col	277	long3colheader	259
altsuper4colborder	277, 311	long3colheaderborder	259, 301
altsuper4colborder	277	long3colheaderborder	259

long4col	259–261, 302	mcoltreehypergroup	269
long4col	259	mcoltreenoname	270, 308
long4colborder	260, 302	mcoltreenoname	269
long4colborder	260	mcoltreenonamegroup .	270, 308
long4colheader	260, 302	mcoltreenonamegroup	270
long4colheader	260	mcoltreenonamehypergroup	
long4colheaderborder	260, 302	270, 308
long4colheaderborder	260	mcoltreenonamehypergroup	270
longborder	257, 301	sublistdotted	300
longborder	257	sublistdotted	256
longheader	257, 301	super	272, 273, 280, 310
longheader	257	super	272
longheaderborder	257, 301	super3col	273–275, 310
longheaderborder	257	super3col	273
longragged	262–264	super3colborder	274, 310
longragged	262	super3colborder	274
longragged3col ...	264, 265, 303	super3colheader	274, 311
longragged3col	264	super3colheader	274
longragged3colborder	265, 303	super3colheaderborder	275, 311
longragged3colborder	265	super3colheaderborder ...	275
longragged3colheader	265, 303	super4col	275–277, 311
longragged3colheader	265	super4col	275
longragged3colheaderborder		super4colborder	276, 311
.....	265, 303	super4colborder	276
longragged3colheaderborder		super4colheader	276, 311
.....	265	super4colheader	276
longraggedborder	263, 303	super4colheaderborder	276, 311
longraggedborder	263	super4colheaderborder ...	276
longraggedheader	263, 303	superborder	272, 310
longraggedheader	263	superborder	272
longraggedheaderborder	264, 303	superheader	273, 310
longraggedheaderborder ..	264	superheader	273
mcotalttree	271, 308	superheaderborder ...	273, 310
mcotalttree	270	superheaderborder	273
mcotalttreegroup	271, 308	superragged	279, 280, 308
mcotalttreegroup	271	superragged	279
mcotalttreehypergroup	271, 308	superragged3col ..	280–282, 309
mcotalttreehypergroup ...	271	superragged3col	280
mcolindex	268, 307	superragged3colborder	281, 309
mcolindex	268	superragged3colborder ...	281
mcolindexgroup	268, 307	superragged3colheader	281, 309
mcolindexgroup	268	superragged3colheader ...	281
mcolindexhypergroup .	268, 307	superragged3colheaderborder	
mcolindexhypergroup	268	282, 309
mcoltree	269, 307	superraggedborder ...	279, 309
mcoltree	269	superraggedborder	279
mcoltreegroup	308	superraggedheader ...	280, 309
mcoltreegroup	269	superraggedheader	280
mcoltreehypergroup ..	269, 308		

superraggedheaderborder .	\Glossentrysymbol 188
. 280, 309	\glossentrysymbol 188, 331
superraggedheaderborder . 280	\GLS 109
superraggedright3colheaderborder	\Gls 108, 111, 244
. 282	\gls 4, 61, 82, 93,
tree 269, 286, 287, 289, 305	107, 109, 110, 113–126, 186, 240
tree 286	\gls@Alphpage 162
treegroup 269, 287, 305	\gls@alphpage 162
treegroup 287	\gls@assign@desc 71
treehypergroup 287, 305	\gls@assign@desc@field 18
treehypergroup 287	\gls@assign@descplural@field 18
treenoname . . . 269, 287, 288, 305	\gls@assign@field 66
treenoname 287	\gls@assign@name@field 18
treenonamegroup 288, 306	\gls@assign@type@field 17
treenonamegroup 288	\gls@checkisacronymlist 15
treenonamehypergroup 288, 306	\gls@checkseeallowed 61
treenonamehypergroup 288	\gls@codepage 25
glossary-hypernav package 144	\gls@defglossaryentry 72
glossary-list package 7, 8, 253	\gls@disablepagerefexpansion 161
glossary-long package	\gls@docclearpage 39
. 8, 256, 265, 271, 272	\gls@doentryfmt 55
glossary-longragged package 262	\gls@glossary 160
glossary-mcols package 267	\gls@hypergroupprerun 249
glossary-super package	\gls@ifnotmeasuring 80
. 8, 256, 271, 278, 282	\gls@istfilebase 34
glossary-superragged package 278	\gls@level 64
glossary-tree package 8, 284	\gls@noidxglossary 159
glossaryentry (counter) . 9, 185, 186	\gls@numberpage 162
glossaryentry (counter) 184	\gls@protected@pagefmts 161
\glossaryentryfield 189, 193	\gls@Romanpage 162
\glossaryentrynumber 184	\gls@romanpage 162
\glossaryentrynumbers	\gls@save@numberlist 167
. 7, 36, 169, 170	\gls@suffixF 35
\glossaryheader 187, 193	\gls@suffixFF 36
\glossarymark 38	\gls@wrglossary 161
\glossaryname 30, 32	\glsaccessdisplay 319
\glossarypostamble 37, 193	\glsaccsupp 317
\glossarypreamble 37, 193	\glsacrpluralsuffix 31
\glossarysection 6, 37, 56	\glsadd 82, 142, 192
\glossarystyle 192, 232	\glsadd options
glossarysubentry (counter)	counter 142
. 10, 185, 186	format 142, 193
glossarysubentry (counter) . . . 185	\glsaddall 51, 82, 143
\glossarysubentryfield 189	\glsaddall options
\glossentry 61, 187	types 142, 143
\Glossentrydesc 188	\glsaddallunused 143
\glossentrydesc 188, 331	\glsaddkey 69
\Glossentryname 188	\GlsAddLetterGroup 49
\glossentryname 187, 331	\glsaddprotectedpagefmt 162

<code>\GlsAddSortRule</code>	47	<code>\glstentryfirstaccess</code>	316
<code>\GlsAddXdyAlphabet</code>	44	<code>\Glstentryfirstplural</code>	138
<code>\GlsAddXdyAttribute</code>	42, 292	<code>\glstentryfirstplural</code>	138
<code>\GlsAddXdyCounters</code>	41, 292	<code>\glstentryfirstpluralaccess</code> ..	316
<code>\GlsAddXdyLocation</code>	46, 293	<code>\glstentryfmt</code>	59, 61, 83
<code>\GlsAddXdyStyle</code>	47	<code>\Glstentryfull</code>	140
<code>\glstautoprefix</code>	6	<code>\glstentryfull</code> ...	140, 202, 211, 340
<code>\glstbackslash</code>	144	<code>\Glstentryfullpl</code>	141
<code>\glstclearpage</code>	40	<code>\glstentryfullpl</code>	140
<code>\glstclosebrace</code>	144	<code>\glstentryitem</code>	186
<code>\glstcompositor</code>	35, 45	<code>\Glstentrylong</code>	140
<code>\glstcounter</code>	16, 57	<code>\glstentrylong</code>	140
<code>\GlsDeclareNoHyperList</code>	16	<code>\glstentrylongaccess</code>	317
<code>\glstdefaulttype</code>	13	<code>\Glstentrylongpl</code>	140
<code>\glstdefmain</code>	12	<code>\glstentrylongpl</code>	140
<code>\GLSdesc</code>	118	<code>\glstentrylongpluralaccess</code> ..	317
<code>\Glsdesc</code>	118	<code>\Glstentryname</code>	135
<code>\glstdesc</code>	118	<code>\glstentryname</code>	135, 167
<code>\GLSdescplural</code>	119	<code>\glstentrynumberlist</code>	141, 156
<code>\Glsdescplural</code>	119	<code>\Glstentryplural</code>	137
<code>\glstdescplural</code>	119	<code>\glstentryplural</code>	137
<code>\glstdescriptionaccessdisplay</code>	319	<code>\glstentrypluralaccess</code>	316
<code>\glstdescriptionpluralaccessdisplay</code>	319	<code>\Glstentryprefix</code>	240
<code>\glstdescwidth</code> ...	256, 262, 271, 278	<code>\glstentryprefix</code>	239
<code>\glstdetoklabel</code>	51	<code>\Glstentryprefixfirst</code>	239
<code>\glstdisablehyper</code>	107	<code>\glstentryprefixfirst</code>	239
<code>\glstdisp</code>	112	<code>\Glstentryprefixfirstplural</code> .	239
<code>\glstdisplay</code>	82, 92, 107	<code>\glstentryprefixfirstplural</code> .	239
<code>\glstdisplayfirst</code>	82, 91, 107	<code>\glstentryprefixplural</code>	240
<code>\glstdisplaynumberlist</code>	141, 157, 180	<code>\glstentryprefixplural</code>	239
<code>\glstdohyperlink</code>	106	<code>\Glstentryshort</code>	140
<code>\glstdohypertarget</code>	106	<code>\glstentryshort</code>	139
<code>\glstdoifexists</code>	51	<code>\glstentryshortaccess</code>	317
<code>\glstdoifexistsorwarn</code>	52	<code>\Glstentryshortpl</code>	140
<code>\glstdoifnoexists</code>	51	<code>\glstentryshortpl</code>	140
<code>\glstdoparenifnotempty</code>	225	<code>\glstentryshortpluralaccess</code> .	317
<code>\glstenablehyper</code>	107	<code>\glstentrysort</code>	138
<code>\glstentryaccess</code>	316	<code>\Glstentrysymbol</code>	137
<code>\glstentrycounter</code>	97	<code>\glstentrysymbol</code>	137
<code>\glstentrycounterlabel</code>	186	<code>\glstentrysymbolaccess</code>	316
<code>\Glstentrydesc</code>	136	<code>\Glstentrysymbolplural</code>	137
<code>\glstentrydesc</code>	136	<code>\glstentrysymbolplural</code>	137
<code>\glstentrydescaccess</code>	317	<code>\glstentrysymbolpluralaccess</code>	317
<code>\Glstentrydescplural</code>	136	<code>\Glstentrytext</code>	136
<code>\glstentrydescplural</code>	136	<code>\glstentrytext</code>	50, 136, 167
<code>\glstentrydescplural</code>	136	<code>\glstentrytextaccess</code>	316
<code>\glstentrydescpluralaccess</code> ..	317	<code>\glstentrytype</code>	138
<code>\Glstentryfirst</code>	137	<code>\Glstentryuseri</code>	138
<code>\glstentryfirst</code>	137	<code>\glstentryuseri</code>	138

<code>\Glsentryuserii</code>	138	<code>\glslink options</code>	
<code>\glsentryuserii</code>	138	counter	93, 107, 238
<code>\Glsentryuseriii</code>	139	format	93, 107, 193
<code>\glsentryuseriii</code>	139	hyper	93, 95, 107
<code>\Glsentryuseriv</code>	139	local	93
<code>\glsentryuseriv</code>	139	<code>\glslinkcheckfirsthyperhook</code>	96
<code>\Glsentryuserv</code>	139	<code>\glslinkvar</code>	94
<code>\glsentryuserv</code>	139	<code>\glslistdottedwidth</code>	255
<code>\Glsentryuservi</code>	139	<code>\glslocalreset</code>	80
<code>\glsentryuservi</code>	139	<code>\glslocalresetall</code>	81
<code>\glsexpandfields</code>	66	<code>\glslocalunset</code>	81
<code>\GLSfirst</code>	115	<code>\glslocalunsetall</code>	82
<code>\Glsfirst</code>	114, 115	<code>\glslongaccessdisplay</code>	319
<code>\glsfirst</code>	114	<code>\glslongaccesskey</code>	347
<code>\glsfirstaccessdisplay</code>	318	<code>\glslongkey</code>	197
<code>\GLSfirstplural</code>	117	<code>\glslongpluralaccessdisplay</code>	319
<code>\Glsfirstplural</code>	116	<code>\glslongpluralaccesskey</code>	347
<code>\glsfirstplural</code>	116, 117	<code>\glslongpluralkey</code>	198
<code>\glsfirstpluralaccessdisplay</code>	318	<code>\glslongtok</code>	201
<code>\glsgenacfmt</code>	89, 326	<code>\glsmakefirstuc</code>	246
<code>\glsgenentryfmt</code>	86, 324	<code>\glsmcols</code>	268
<code>\glsgetgrouplabel</code>	191	<code>\glsmoveentry</code>	77
<code>\glsgetgrouptitle</code>	144, 191	<code>\GLSname</code>	117
<code>\glsglossarymark</code>	9, 38	<code>\Glsname</code>	117
<code>\glsgroupheading</code>	190, 193	<code>\glsname</code>	117
<code>\glsgroupskip</code>	190, 193, 253	<code>\glsnameaccessdisplay</code>	318
<code>\glshyperlink</code>	142	<code>\glsnamefont</code>	193, 284
<code>\glshypernavsep</code>	250	<code>\glsnavhyperlink</code>	248
<code>\glshypernumber</code>	36, 194	<code>\glsnavhypertarget</code>	248
<code>\glsifhyper</code>	94	<code>\glsnavigation</code>	249
<code>\glsifhyperon</code>	94	<code>\glsnextpages</code>	184
<code>\glsIfListOfAcronyms</code>	15	<code>\glsnoexpandfields</code>	66
<code>\glsifusedtranslatordict</code>	22	<code>\glsnoidxdisplayloc</code>	180
<code>\glsignore</code>	143	<code>\glsnoidxdisplaylocclishandler</code>	180
<code>\glsinlinedescformat</code>	252	180
<code>\glsinlinedopostchild</code>	251, 252	<code>\glsnoidxloclist</code>	179
<code>\glsinlineemptydescformat</code>	252	<code>\glsnoidxloclisthandler</code>	179
<code>\glsinlinenameformat</code>	252	<code>\glsnoidxnumberlistloophandler</code>	158
<code>\glsinlinenonparentchildseparator</code>	252	158
<code>\glsinlinedopostchild</code>	252	<code>\glsnoidxstripaccents</code>	19
<code>\glsinlineseparator</code>	252	<code>\glsnonextpages</code>	184
<code>\glsinlinesubdescformat</code>	252	<code>\glsnoxindywarning</code>	40
<code>\glsinlinesubnameformat</code>	252	<code>\glsnumberformat</code>	36
<code>\glsinlinesubseparator</code>	252	<code>\glsnumberlistloop</code>	158
<code>\glskeylisttok</code>	201	<code>\glsnumbersgroupname</code>	31, 144, 190
<code>\glslabeltok</code>	201	<code>\glsnumlistlastsep</code>	142
<code>\glsletentryfield</code>	134	<code>\glsnumlistsep</code>	142
<code>\glslink</code>	82, 95, 107, 142, 192, 193	<code>\glsopenbrace</code>	144
		<code>\glsorder</code>	24

<code>\glsorg@endtheglossary</code>	5	<code>\glsshortpluralkey</code>	197
<code>\glsorg@theglossary</code>	5	<code>\glsshorttok</code>	201
<code>\glspagelistwidth</code>	256, 262, 272, 279	<code>\glssortnumberfmt</code>	11
<code>\glspar</code>	34	<code>\glsspace</code>	198
<code>\glsperscentchar</code>	144	<code>\glssstepentry</code>	185
<code>\GLSpl</code>	112	<code>\glssstepsubentry</code>	185
<code>\Glspl</code>	111, 244	<code>\glssubentrycounterlabel</code>	186
<code>\glspl</code>	82, 110, 111	<code>\glssubentryitem</code>	186
<code>\GLSplural</code>	116	<code>\GLSsymbol</code>	120
<code>\Glsplural</code>	115	<code>\Glsymbol</code>	120
<code>\glsplural</code>	115, 116	<code>\glsymbol</code>	120
<code>\glspluralaccessdisplay</code>	318	<code>\glsymbolaccessdisplay</code>	318
<code>\glspluralsuffix</code>	31, 60	<code>\glsymbolnav</code>	250
<code>\glspostdescription</code>	9	<code>\GLSsymbolplural</code>	121
<code>\glspostinline</code>	252	<code>\Glsymbolplural</code>	121
<code>\glsprestandardsort</code>	10	<code>\glsymbolplural</code>	120, 121
<code>\glsquote</code>	144	<code>\glsymbolpluralaccessdisplay</code>	319
<code>\glsrefentry</code>	186	<code>\glssymbolsgroupname</code>	31, 144, 190
<code>\glsreset</code>	80	<code>\glstarget</code>	187
<code>\glsresetall</code>	81	<code>\GLStext</code>	114
<code>\glsresetentrylist</code>	184	<code>\Glstext</code>	114
<code>\glsresetsubentrycounter</code>	185	<code>\glstext</code>	113
<code>\glssee</code>	166	<code>\glstextaccessdisplay</code>	318
<code>\glsseeformat</code>	147, 166	<code>\glstextformat</code>	82
<code>\glsseeitem</code>	167	<code>\glstextup</code>	197
<code>\glsseeitemformat</code>	167	<code>\glstildechar</code>	144
<code>\glsseelastsep</code>	167	<code>\glstreeindent</code>	286–288
<code>\glsseelist</code>	166	<code>\glstreenamefmt</code>	284
<code>\glsseesep</code>	167	<code>\glsunset</code>	80
<code>\glsSetAlphaCompositor</code>	35	<code>\glsunsetall</code>	81
<code>\glsSetCompositor</code>	35	<code>\glsupacrpluralsuffix</code>	31
<code>\glssetexpandfield</code>	17	<code>\GlsUseAcrEntryDispStyle</code>	204
<code>\glssetnoexpandfield</code>	17	<code>\GlsUseAcrStyleDefs</code>	204
<code>\glsSetSuffixF</code>	35	<code>\GLSuseri</code>	122
<code>\glsSetSuffixFF</code>	36	<code>\Glsuseri</code>	122
<code>\glssettoctitle</code>	30	<code>\glsuseri</code>	121, 122
<code>\glssetwidest</code>	289	<code>\GLSuserii</code>	123
<code>\GlsSetXdyCodePage</code>	48	<code>\Glsuserii</code>	123
<code>\GlsSetXdyFirstLetterAfterDigits</code>	144	<code>\glsuserii</code>	122, 123
<code>\GlsSetXdyLanguage</code>	48	<code>\GLSuseriii</code>	124
<code>\GlsSetXdyLocationClassOrder</code>	47	<code>\Glsuseriii</code>	123
<code>\GlsSetXdyMinRangeLength</code>	145	<code>\glsuseriii</code>	123, 124
<code>\GlsSetXdyStyles</code>	48	<code>\GLSuseriv</code>	125
<code>\glsshortaccessdisplay</code>	319	<code>\Glsuseriv</code>	124
<code>\glsshortaccesskey</code>	347	<code>\glsuseriv</code>	124, 125
<code>\glsshortkey</code>	197	<code>\GLSuserv</code>	126
<code>\glsshortpluralaccessdisplay</code>	319	<code>\Glsuserv</code>	125
<code>\glsshortpluralaccesskey</code>	347	<code>\glsuserv</code>	125, 126

<code>\GLSuservi</code>	126
<code>\Glsuservi</code>	126
<code>\glsuservi</code>	126
<code>\glswrite</code>	154
<code>\glswritedefhook</code>	71
<code>\gMFUnocap</code>	247

H

<code>\hyperbf</code>	195
<code>\hyperemph</code>	196
<code>hyperfirst (option)</code>	23
<code>\hyperit</code>	196
<code>\hyperlink</code>	106
<code>\hypermd</code>	195
<code>\hyperpage</code>	194
<code>hyperref package</code> ...	164, 168, 194, 238
<code>\hyperrm</code>	195
<code>\hypersc</code>	196
<code>\hypersf</code>	195
<code>\hypersl</code>	196
<code>\hypertarget</code>	106
<code>\hypertt</code>	195
<code>\hyperup</code>	196

I

<code>\if@gls@docloaded</code>	5
<code>\if@gls@isacronymlist</code>	15
<code>\ifglossaryexists</code>	50
<code>\ifglshdescsuppressed</code>	53
<code>\ifglshentryexists</code>	51
<code>\ifglshaschildren</code>	52
<code>\ifglshasdesc</code>	53
<code>\ifglshasfield</code>	54
<code>\ifglshaslong</code>	53
<code>\ifglshasparent</code>	52
<code>\ifglshasprefix</code>	240
<code>\ifglshasprefixfirst</code>	240
<code>\ifglshasprefixfirstplural</code> ..	240
<code>\ifglshasprefixplural</code>	240
<code>\ifglshasshort</code>	53
<code>\ifglshassymbol</code>	53
<code>\ifglstranslate</code>	21
<code>\ifglused</code>	51, 80
<code>\ifglxindy</code>	25
<code>\ifignoredglossary</code>	59
<code>index (option)</code>	28
<code>index (style)</code>	284
<code>indexgroup (style)</code>	285
<code>indexhypergroup (style)</code>	286
<code>indexonlyfirst (option)</code>	23

<code>\indexspace</code>	253, 268, 284
<code>inline (style)</code>	250
<code>\inputencodingname</code>	25
<code>\istfile</code>	159
<code>\istfilename</code>	34
<code>\item</code>	193, 253, 284, 285

L

<code>link text</code>	82
<code>list (style)</code>	253
<code>listdotted (style)</code>	255
<code>listgroup (style)</code>	253
<code>listhypergroup (style)</code>	254
<code>\loadglsentries</code>	13, 82
<code>long (key)</code>	63
<code>long (style)</code>	256
<code>long-sc-short (acrstyle)</code>	205
<code>long-sc-short-desc (acrstyle)</code> ..	206, 334
<code>long-short (acrstyle)</code>	204, 333
<code>long-short-desc (acrstyle)</code> ..	206, 334
<code>long-sm-short (acrstyle)</code>	205
<code>long-sm-short-desc (acrstyle)</code> ..	207, 335
<code>long3col (style)</code>	258
<code>long3colborder (style)</code>	258
<code>long3colheader (style)</code>	259
<code>long3colheaderborder (style)</code> ..	259
<code>long4col (style)</code>	259
<code>long4colborder (style)</code>	260
<code>long4colheader (style)</code>	260
<code>long4colheaderborder (style)</code> ..	260
<code>longaccess (key)</code>	314
<code>longborder (style)</code>	257
<code>longheader (style)</code>	257
<code>longheaderborder (style)</code>	257
<code>\longnewglossaryentry</code>	60, 71
<code>longplural (key)</code>	63
<code>longpluralaccess (key)</code>	314
<code>\longprovideglossaryentry</code> ...	72
<code>longragged (style)</code>	262
<code>longragged3col (style)</code>	264
<code>longragged3colborder (style)</code> ..	265
<code>longragged3colheader (style)</code> ..	265
<code>longragged3colheaderborder</code> (style)	265
<code>longraggedborder (style)</code>	263
<code>longraggedheader (style)</code>	263
<code>longraggedheaderborder (style)</code>	264

longtable (environment)
 8, 232, 256–267
 longtable package 256, 262

M

\makefirstuc 244
 makeglossaries
 24, 25, 34, 48, 56, 153, 170
 \makeglossaries 26,
 29, 30, 34, 35, 55, 57, 153, 155
 \makeglossary 155
 makeindex **350**
 makeindex 10, 25, 26, 31, 34–36, 56,
 58, 60, 79, 98, 101, 144, 147,
 149, 151, 160, 164, 190, 293, 294
 delim_n 36
 delim_r 36
 page_compositor 35
 special characters 100, 144
 makeindex (option) 25
 \makenoidxglossaries 21, 155
 mcolalttree (style) 270
 mcolalttreegroup (style) 271
 mcolalttreehypergroup (style) . 271
 mcolindex (style) 268
 mcolindexgroup (style) 268
 mcolindexhypergroup (style) ... 268
 mcoltree (style) 269
 mcoltreegroup (style) 269
 mcoltreehypergroup (style) 269
 mcoltreenoname (style) 269
 mcoltreenonamegroup (style) ... 270
 mcoltreenonamehypergroup
 (style) 270
 memoir class 160
 mfirstuc package 1, 247
 \mfirstucMakeUppercase 246
 \mfu@checkword 247
 \MFUclear 247
 \MFUnocap 247
 multicol package 267
 multicol (environment) 268

N

name (key) 59
 \new@glossaryentry 66
 \newacronym 24, 63, 82, 196, 197
 \newacronymhook 201
 \newacronymstyle 203
 \newglossary 16, 56, 58, 151, 153, 181

\newglossaryentry .. 60, 66, 82, 197
 \newglossaryentry options
 access 315, 316
 counter 61
 description 24, 59, 60,
 63, 66, 72, 118, 136, 197, 224, 313
 descriptionaccess 317, 319
 descriptionplural 119, 314
 descriptionpluralaccess .. 317, 319
 first 60, 75,
 107, 114, 137, 222, 227, 228, 313
 firstaccess 316, 318
 firstplural 60, 116, 138, 313
 firstpluralaccess 316, 318
 format 146
 long 89, 140, 314
 longaccess 317, 319
 longplural 140, 314
 longpluralaccess 317, 319
 name 59,
 60, 63, 66, 72, 117, 135, 167, 313
 nonumberlist 62
 parent 62, 66
 plural 60, 75, 115, 313
 pluralaccess 316, 318
 prefix 239
 prefixfirst 239
 prefixfirstplural 239
 prefixplural 239
 see 8, 61, 154, 155
 short 89, 139, 314
 shortaccess 317, 319
 shortplural 140, 314
 shortpluralaccess 317, 319
 sort 60, 138, 190
 symbol 59, 61, 120, 218,
 220, 222, 227, 259, 275, 313, 315
 symbolaccess 316, 318
 symbolplural 120, 313
 symbolpluralaccess 317, 319
 text 60, 107, 113, 136, 218, 222, 313
 textaccess 316, 318
 type 13, 61, 82, 138
 user1 121, 138, 314
 user2 122, 138
 user3 123, 139
 user4 124, 139
 user5 125, 139
 user6 126, 139, 314

<code>\newglossarystyle</code>	192
<code>\newignoredglossary</code>	58
<code>nogroupskip</code> (option)	9
<code>nohypertypes</code> (option)	16
<code>\noist</code>	151, 238, 298
<code>nolist</code> (option)	8
<code>nolong</code> (option)	8
<code>nomain</code> (option)	13
<code>nonumberlist</code> (key)	62
<code>nonumberlist</code> (option)	7
<code>\nopostdesc</code>	33
<code>nopostdot</code> (option)	9
<code>noredefwarn</code> (option)	17
<code>nostyles</code> (option)	8
<code>nosuper</code> (option)	8
<code>notranslate</code> (option)	22
<code>notree</code> (option)	8
<code>nowarn</code> (option)	16
<code>\ns@newglossary</code>	56
<code>numberedsection</code> (option)	6
<code>numberline</code> (option)	5
<code>numbers</code> (option)	27

O

<code>\oldacronym</code>	196
<code>order</code> (option)	25

P

<code>\p@glshyp@opt</code>	94
package options:	
<code>acronym</code>	13, 14, 30, 168, 197
<code>true</code>	14
<code>acronym</code>	13
<code>acronymlists</code>	15
<code>acronyms</code>	14
<code>automake</code>	26
<code>compatible-2.07</code>	27
<code>compatible-3.07</code>	27
<code>counter</code>	16
<code>counter</code>	16
<code>description</code>	222, 223
<code>description</code>	24
<code>dua</code>	220, 222, 223
<code>dua</code>	24
<code>entrycounter</code>	182, 184
<code>true</code>	9
<code>entrycounter</code>	9
<code>entrycounterwithin</code>	9
<code>footnote</code> 108–113, 218, 220, 222, 224	
<code>footnote</code>	24

<code>hyperfirst</code>	
<code>false</code>	108–113
<code>hyperfirst</code>	23
<code>index</code>	28
<code>indexonlyfirst</code>	357
<code>indexonlyfirst</code>	23
<code>makeindex</code>	147, 238
<code>makeindex</code>	25
<code>nogroupskip</code>	9
<code>nohypertypes</code>	16
<code>nolist</code>	232
<code>nolist</code>	8
<code>nolong</code>	232, 256
<code>nolong</code>	8
<code>nomain</code>	12, 13
<code>nomain</code>	13
<code>nonumberlist</code>	7
<code>nonumberlist</code>	7
<code>nopostdot</code>	9
<code>noredefwarn</code>	17
<code>nostyles</code>	8
<code>nosuper</code>	232
<code>nosuper</code>	8
<code>notranslate</code>	22
<code>notree</code>	232
<code>notree</code>	8
<code>nowarn</code>	16
<code>numberedsection</code>	6
<code>numberline</code>	5
<code>numberline</code>	5
<code>numbers</code>	27
<code>order</code>	25
<code>sanitize</code>	20, 59, 135, 136
<code>sanitize</code>	21
<code>sanitizesort</code>	20
<code>savenumberlist</code>	7
<code>savewrites</code>	26, 354, 355
<code>false</code>	151
<code>true</code>	153, 159
<code>savewrites</code>	26
<code>section</code>	6, 38
<code>section</code>	6
<code>seeautonumberlist</code>	8
<code>shotcuts</code>	24
<code>smallcaps</code>	24
<code>smaller</code>	24
<code>sort</code>	
<code>def</code>	10, 11
<code>standard</code>	10

use	10, 11	\printnoidxglossary	168, 181
sort	10	\printnoidxglossary options	
style	7, 232	sort	183
style	7	\provideglossaryentry	66
subentrycounter	182, 185	\ProvidesGlossariesLang	31
subentrycounter	10		
symbols	27	R	
toc	5	\renewacronymstyle	203
true	5	\renewglossarystyle	193
toc	5	\RequireGlossariesLang	31
translate	22	\roman	44
false	22		
translate	22	S	
translator	21	\s@gl@s@hyp@opt	94
ucmark	9	\s@newglossary	56
xindy	25, 26, 147, 238	sanitize (option)	21
xindy	25	sanitizesort (option)	20
xindygloss	26	savenumberlist (option)	7
xindynoglsnumbers	26	savewrites (option)	26
\pagelistname	31	sc-short-long (acrstyle)	206
parent (key)	62	sc-short-long-desc (acrstyle) ..	
\PGLS	243	207, 335
\Pgls	242	\scantokens	50
\pgls	240	\section	50
\PGLSpl	244	section (option)	6
\Pglspl	242	see (key)	61
\pglspl	241	seeautonumberlist (option)	8
\phantomsection	37–39	\seename	31
plural (key)	60	\SetAcronymLists	15
pluralaccess (key)	313	\SetAcronymStyle	14, 230
polyglossia package	22, 32	\setacronymstyle	203
\printacronyms	14	\SetCustomDisplayStyle	230
\PrintChanges	5	\SetCustomStyle	231
\printglossaries		\SetDefaultAcronymDisplayStyle	
. 12, 36, 55, 58, 155, 167, 168, 248		215
\printglossary	36, 37,	\SetDefaultAcronymStyle	215
56, 58, 155, 167, 168, 170, 181, 248		\SetDescriptionAcronymDisplayStyle	
\printglossary options		220
entrycounter	182	\SetDescriptionAcronymStyle	222
nogroupskip	182	\SetDescriptionDUAAcronymDisplayStyle	
nonumberlist	183	219
nopostdot	182	\SetDescriptionDUAAcronymStyle	
numberedsection	181	220
style	181	\SetDescriptionFootnoteAcronymDisplayStyle	
subentrycounter	182	216
title	181	\SetDescriptionFootnoteAcronymStyle	
toctitle	181	218
type	13, 167, 181	\SetDUADisplayStyle	228
\printnoidxglossaries	168	\SetDUASStyle	229
		\setentrycounter	192

\SetFootnoteAcronymDisplayStyle	223	\showglossaryout	237
\SetFootnoteAcronymStyle ...	224	\showglossarytitle	237
\SetGenericNewAcronym	201	\showglosymbol	235
\setglossarypreamble	37	\showglosymbolaccess	348
\setglossarysection	38	\showglosymbolplural	235
\setglossarystyle	192	\showglosymbolpluralaccess .	348
\setglossentrycompatibility	189	\showglotext	233
\SetSmallAcronymDisplayStyle	225	\showglotextaccess	347
\SetSmallAcronymStyle	227	\showglotype	233
\setStyleFile	34	\showglouserii	234
\setupglossaries	28	\showglouseriii	234
short (key)	63	\showglouseriii	234
short-long (acrstyle)	205, 333	\showglouseriv	234
short-long-desc (acrstyle) .	207, 335	\showglouserv	234
shortaccess (key)	314	\showglouservi	234
shortplural (key)	63	sm-short-long (acrstyle)	206
shortpluralaccess (key)	314	sm-short-long-desc (acrstyle) ..	208, 336
shotcuts (option)	24	smallcaps (option)	24
\showacronymlists	236	smaller (option)	24
\showglocounter	233	\SmallNewAcronymDef	226, 346
\showglodesc	235	sort (key)	60
\showglodescaccess	348	sort (option)	10
\showglodescplural	235	style (option)	7
\showglodescpluralaccess ...	348	subentrycounter (option)	10
\showglofirst	233	\subglossentry	187
\showglofirstaccess	347	\subitem	285
\showglofirstpl	233	sublistdotted (style)	256
\showglofirstpluralaccess ..	347	\subsubitem	285
\showgloflag	236	super (style)	272
\showgloindex	236	super3col (style)	273
\showglolevel	232	super3colborder (style)	274
\showgloloclist	236	super3colheader (style)	274
\showglolong	236	super3colheaderborder (style) .	275
\showglolongaccess	348	super4col (style)	275
\showglolongpluralaccess ...	348	super4colborder (style)	276
\showgloname	235	super4colheader (style)	276
\showglonameaccess	347	super4colheaderborder (style) .	276
\showgloparent	232	superborder (style)	272
\showgloplural	233	superheader (style)	273
\showglopluralaccess	347	superheaderborder (style)	273
\showgloshort	236	superragged (style)	279
\showgloshortaccess	348	superragged3col (style)	280
\showgloshortpluralaccess ..	348	superragged3colborder (style) .	281
\showglosort	235	superragged3colheader (style) .	281
\showglossaries	237	superraggedborder (style)	279
\showglossarycounter	237	superraggedheader (style)	280
\showglossaryentries	237	superraggedheaderborder (style)	280
\showglossaryin	237		

superraggedright3colheaderborder (style)	282	treenoname (style)	287
supertabular (environment) ...		treenonamegroup (style)	288
.....	8, 232, 271–284	treenonamehypergroup (style) ..	288
supertabular package ..	8, 232, 271, 278	type (key)	61
symbol (key)	61		
symbolaccess (key)	313	U	
\symbolname	31	ucmark (option)	9
symbolplural (key)	61	user1 (key)	62
symbolpluralaccess (key)	313	user2 (key)	62
symbols (option)	27	user3 (key)	62
		user4 (key)	62
T		user5 (key)	62
text (key)	60	user6 (key)	63
textaccess (key)	313		
textcase package	4	W	
\theequation	165	\warn@nomakeglossaries	153
theglossary (environment)		\warn@noprintglossary	167
.....	5, 17, 36, 37, 186,	\writeist 34, 41, 43, 46, 145, 292, 294	
	187, 193, 269, 270, 286, 287, 289		
\theHequation	165	X	
theindex (environment)	284	\xcapitalisewords	247
toc (option)	5	\xglsacccsupp	318
tracklang package	32, 349	xindy	350
\translate	32	xindy ... 10, 25, 26, 34, 35, 40, 44,	
translate (option)	22	46–49, 79, 105, 144, 145, 147,	
translator package		160, 164, 170, 190, 238, 293, 294	
... 13, 14, 22, 27, 28, 32, 33, 168		xindy (option)	25
tree (style)	286	xindygloss (option)	26
treegroup (style)	287	xindynoglsnumbers (option)	26
treehypergroup (style)	287	\xmakefirsttuc	246
		xspace package	4, 196