

Documented Code For glossaries v4.18

Nicola L.C. Talbot

Dickimaw Books

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

2015-09-09

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 v4.18: 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	83
1.9	Keeping Track of How Many Times an Entry Has Been Unset	86
1.10	Loading files containing glossary entries	91
1.11	Using glossary entries in the text	92
1.11.1	Links to glossary entries	102
1.11.2	Displaying entry details without adding information to the glossary	144
1.12	Adding an entry to the glossary without generating text	153
1.13	Creating associated files	154
1.14	Writing information to associated files	170
1.15	Glossary Entry Cross-References	176
1.16	Displaying the glossary	178
1.17	Acronyms	207
1.18	Predefined acronym styles	211
1.19	Predefined Glossary Styles	243
1.20	Debugging Commands	244
1.21	Compatibility with version 2.07 and below	249
2	Prefix Support (glossaries-prefix Code)	250
3	Glossary Styles	256
3.1	Glossary hyper-navigation definitions (glossary-hypernav package)	256
3.2	In-line Style (glossary-inline.sty)	259
3.3	List Style (glossary-list.sty)	261
3.4	Glossary Styles using longtable (the glossary-long package)	264
3.5	Glossary Styles using longtable (the glossary-longragged package)	270
3.6	Glossary Styles using multicol (glossary-mcols.sty)	276
3.7	Glossary Styles using supertabular environment (glossary-super package)	280
3.8	Glossary Styles using supertabular environment (glossary-superragged package)	286
3.9	Tree Styles (glossary-tree.sty)	292
4	glossaries-compatible-207	300

5 Accessibility Support (glossaries-accsupp Code)	320
5.1 Defining Replacement Text	321
5.2 Accessing Replacement Text	324
5.3 Displaying the Glossary	340
5.4 Acronyms	341
5.5 Debugging Commands	355
6 Multi-Lingual Support	357
6.1 Polyglossia Captions	357
Glossary	358
Change History	358
Index	383

1 Main Package Code

1.1 Package Definition

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

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries}[2015/09/09 v4.18 (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 environment.

```

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

```

End of doc stuff.

```
31 \fi
```

1.2 Package Options

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

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

```

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

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

```

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

```

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

```

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

```

37 \define@choicekey{glossaries.sty}{section}{part,chapter,section,%
38 subsection,subsubsection,paragraph,subparagraph}[section]{%
39   \renewcommand*{\@@glossarysec}{#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 1.19](#).)

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 1.19](#).

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

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

\@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*{\@glo@seeautonumberlist}{%
79     \def\@glo@prefix{\glsnextpages}%
80   }%
81 }

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

nolong This option prevents from being loaded. This means that the glossary styles
that use the longtable environment will not be available. This option is pro-
vided 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 pro-
vided 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). Nat-
urally, 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). Nat-
urally, 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

```

`@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 argument entry label.)

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

```
135     \ifx\@glo@sort\@glsdefaultsort
```

```
136       \let\@glo@sort\@glo@name
```

```
137     \fi
```

```
138     \let\glsdosanitizesort\@gls@sanitizesort
```

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

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

```
141   }%
```

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

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

```
143 }
```

Set standard sort as the default:

```
144 \@gls@setupsort@standard
```

```

\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\def\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 1.10](#)).

`\glsdefaulttype`

```
205 \newcommand*{\glsdefaulttype}{main}
```

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

`\acronymtype`

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

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

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

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

```
211 \define@boolkey{glossaries.sty}[gls]{acronym}[true]{%
212   \ifglsacronym
213     \renewcommand{\@gls@do@acronymsdef}{%
214       \DeclareAcronymList{acronym}%
215       \newglossary[alg]{acronym}{acr}{acn}{\acronymname}%
216       \renewcommand*{\acronymtype}{acronym}%
```

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

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

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

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

```
233 \fi
234 }
```

`@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{}}

\DeclareNoHyperList
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 }

\GlossariesWarningNoLine   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 }

```



```

noredefwarn Suppress warning on redefinition of \printglossary
317 \@gls@declareoption{noredefwarn}{%
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 }

noidx@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 }

```

lsglsstripaccents

```

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       \glssetexpandfield{sortvalue}%
448     }%
449   \fi
450 }

```

Default setting:

```

451 \glssanitizesorttrue
452 \glssetnoexpandfield{sortvalue}%

```

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

```
453 \newcommand*{\@gls@noidx@setsanitizesort}{%
454   \gls@sanitizesortfalse
455   \gls@setexpandfield{sortvalue}%
456 }

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

sanitize

```
467 \define@key{glossaries.sty}{sanitize}[description=true,symbol=true,name=true]{%
468   \ifthenelse{\equal{#1}{none}}{%
469     {%
470       \GlossariesWarning{sanitize package option deprecated}%
471       \gls@setexpandfield{name}%
472       \gls@setexpandfield{symbol}%
473       \gls@setexpandfield{symbolplural}%
474       \gls@setexpandfield{desc}%
475       \gls@setexpandfield{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@notranslatorhook \@gls@notranslatorhook has been removed.

\@gls@usetranslator

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

```

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

```

505 \@gls@declareoption{notranslate}{%
506 \glstranslatefalse
507 \let\@gls@usetranslator\relax
508 \let\glsifusetranslator\@secondoftwo
509 }

```

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

```

510 \define@choicekey{glossaries.sty}{translate}[\val\nr]%
511 {true,false,babel}[true]%
512 {%
513 \ifcase\nr\relax
514 \glstranslatetrue
515 \renewcommand*\@gls@usetranslator{%
516 \@ifpackageloaded{polyglossia}%
517 {%
518 \let\glsifusetranslator\@secondoftwo
519 }%
520 {}%
521 \@ifpackageloaded{babel}%
522 {%
523 \IfFileExists{translator.sty}%
524 {%
525 \RequirePackage{translator}%
526 \let\glsifusetranslator\@firstoftwo
527 }%
528 {}%
529 }%

```

```

530     {}%
531   }%
532 }%
533 \or
534   \glstranslatefalse
535   \let\@gls@usetranslator\relax
536   \let\glsifusetranslator\@secondoftwo
537 \or
538   \glstranslatetrue
539   \let\@gls@usetranslator\relax
540   \let\glsifusetranslator\@secondoftwo
541 \fi
542 }

```

Set the default value:

```

543 \glstranslatefalse
544 \let\glsifusetranslator\@secondoftwo
545 \@ifpackageloaded{translator}%
546 {%
547   \glstranslatetrue
548   \let\glsifusetranslator\@firstoftwo
549 }%
550 {%
551   \@for\gls@thissty:=tracklang,babel,ngerman,polyglossia\do
552   {
553     \@ifpackageloaded{\gls@thissty}%
554     {%
555       \glstranslatetrue
556       \@endfortrue
557     }%
558   }%
559 }
560 }

```

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

```

561 \define@boolkey{glossaries.sty}[gls]{indexonlyfirst}[true]{}
562 \glsindexonlyfirstfalse

```

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

```

563 \define@boolkey{glossaries.sty}[gls]{hyperfirst}[true]{}
564 \glshyperfirsttrue

```

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

```

565 \newcommand*\@gls@setacrstyle{}

```

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

```

566 \define@boolkey{glossaries.sty}[glsacr]{footnote}[true]{}

```

```

567 \ifbool{glsacrdescription}%
568 {}%
569 {%
570   \renewcommand*{\@gls@sanitizedesc}{}%
571 }%
572 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
573 }

```

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

```

574 \define@boolkey{glossaries.sty}[glsacr]{description}[true]{%
575   \renewcommand*{\@gls@sanitizesymbol}{}%
576   \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
577 }

```

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

```

578 \define@boolkey{glossaries.sty}[glsacr]{smallcaps}[true]{%
579   \renewcommand*{\@gls@sanitizesymbol}{}%
580   \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
581 }

```

smaller Define `\newacronym` to set the short form using `\smaller` which obviously needs to be 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[gls]{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[gls]{xindy}{language}{\def\@xdy@main@language{#1}}
```

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

```
602 \ifcsundef{inputencodingname}{%
```

```
603 \def\gls@codepage{}}{%
```

```
604 \def\gls@codepage{\inputencodingname}
```

```
605 }
```

Define a key to set the code page.

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

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

```
607 \define@key{glossaries.sty}{xindy}[]{%
```

```
608 \glxindytrue
```

```
609 \setkeys[gls]{xindy}{#1}%
```

```
610 }
```

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

```
611 \@gls@declareoption{xindygloss}{%
```

```
612 \glxindytrue
```

```
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   \ifglsautomake
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 \glsautomakefalse
```

`\@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][\newglossaryentry{##2}%
684     {type={index},name={##2},description={\nopostdesc},##1}}%
685   }%
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.<n>.0` target, otherwise entries placed before the first section of a chapter will have undefined links.

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

```

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

```

`\@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 }
```

\addglossarytocaptions

Does nothing if translator hasn't been loaded.

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

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

```
777 \ifglstranslate
```

Load `tracklang`

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

Load translator if required.

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

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

```
780 \ifpackageloaded{translator}
781  {%
```

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

```
782   \ifboolexpr
783   {
784     test {\ifdefstring{\trans@languages}{English}}
785     and not
786     test {\ifdefstring{bbl@loaded}{english}}
787   }
788   {%
789   \let\glsifusetranslator\@secondoftwo
790   }%
791   {%
792     \usedictionary{glossaries-dictionary}%
793     \renewcommand*{\addglossarytocaptions}[1]{%
794       \ifcsundef{captions#1}{}%
795       {%
796         \expandafter\let\expandafter\@gls@tmp\csname captions#1\endcsname
797         \expandafter\toks@\expandafter{\@gls@tmp
798           \renewcommand*{\glossaryname}{\translate{Glossary}}}%
799         }%
800         \expandafter\edef\csname captions#1\endcsname{\the\toks@}%
801       }%
802     }%
803   }%
804 }%
805 }%
```

Check for tracked languages

```
806 \AnyTrackedLanguages
807  {%
808   \ForEachTrackedDialect{\this@dialect}{%
809     \IfTrackedLanguageFileExists{\this@dialect}%
810     {glossaries-}% prefix
811     {.ldf}%
812     {%
813       \RequireGlossariesLang{\CurrentTrackedTag}%
814     }%
815     {%
816       \PackageWarningNoLine{glossaries}%
817       {No language module detected for ‘\this@dialect’.\MessageBreak
```



```

818         Language modules need to be installed separately.\MessageBreak
819         Please check on CTAN for a bundle called\MessageBreak
820         ‘glossaries-\CurrentTrackedLanguage’ or similar}%
821     }%
822 }%
823 }%
824 {}%

```

if using translator use translator interface.

```

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

```

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

```
850 \DeclareRobustCommand*\nopostdesc{}
```

`\@nopostdesc` Suppress next description terminator.

```

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

```

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

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

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

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

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

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

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

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

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

`\istfilename`

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

`\gls@istfilebase`

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

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

`\@istfilename`

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

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

`\glscompositor`

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

`\glsSetCompositor` Sets the compositor.

```

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

```

Only use before `\makeglossaries`

```

877 \@onlypremakeg\glsSetCompositor

```

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

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

```

878 \newcommand*\@glsAlphacompositor{\glscompositor}

```

`sSetAlphaCompositor` Sets the alpha compositor.

```

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

```

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

```

892 \newcommand*\glsSetSuffiFF[1]{%
893   \renewcommand*\gls@suffiFF{#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 [type]{text}
```

Code provided by Michael Pock.

```

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

```

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

```

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

```

`\glossarypostamble`

```

916 \newcommand*{\glossarypostamble}{}

```

`\glossarysection`

The sectioning command that starts a glossary is given by `\glossarysection`. (This does not form part of the glossary style, and so should not be changed by a glossary style.) If `\phantomsection` is defined, it uses `\p@glossarysection`, otherwise it uses `\@glossarysection`.

```

917 \newcommand*{\glossarysection}[2][\@gls@title]{%
918   \def\@gls@title{#2}%
919   \ifcsundef{phantomsection}%
920     {%
921       \@glossarysection{#1}{#2}%
922     }%
923     {%
924       \p@glossarysection{#1}{#2}%
925     }%
926   \gls@glossarymark{\glossarytoctitle}%
927 }

```

`\gls@glossarymark`

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

```

928 \ifcsundef{glossarymark}%
929 {%
930   \newcommand{\gls@glossarymark}[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 \glsclearpage
975 \phantomsection
976 \ifdefempty\@glossarysecstar
977 {%
978 \csname\@glossarysec\endcsname{#2}%
979 }%
980 {%
981 \@gls@toc{#1}{\@glossarysec}%
982 \csname\@glossarysec\endcsname*{#2}%
983 }%

```

Do automatic labelling if required

```

984 \@glossaryseclabel
985 }

```

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

```

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

```

```

1003     \cleardoublepage
1004   }%
1005 }%
1006 }%
1007 {}%
1008 }

```

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

```

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

```

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

`\@gls@toc`

```

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

```

1.4 Xindy

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

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

```

`\@xdyattributelist` Comma-separated list of attributes.

```

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

```


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

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

`\@gls@ifinlist`

```
1031 \newcommand*\@gls@ifinlist}[4]{%
1032   \def\@do@ifinlist##1,#1,##2\end@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 \ifglxindy
1043   \newcommand*\@xdycounters{\@glscounter}
1044   \newcommand*\GlsAddXdyCounters[1]{%
1045     \@for\@gls@ctr:=#1\do{%
```

Check if already in list before adding.

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

Only has an effect before `\writeist`:

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

`\d@glsaddxdycounters` Counters must all be identified before adding attributes.

```
1062 \newcommand*\@disabled@glsaddxdycounters{%
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 \ifglxindy
```

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

```
1069 \newcommand*\@glsaddxdyattribute[2]{%
```

Add to xindy attribute list

```
1070   \edef\xdyattributes{\x dyattributes ^^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 \ifglxindy
1104 \newcommand*\@gls@addpredefinedattributes{%
1105 \GlsAddXdyAttribute{glsnumberformat}
1106 \GlsAddXdyAttribute{textrm}
1107 \GlsAddXdyAttribute{textsf}
1108 \GlsAddXdyAttribute{texttt}
1109 \GlsAddXdyAttribute{textbf}
1110 \GlsAddXdyAttribute{textmd}
1111 \GlsAddXdyAttribute{textit}
1112 \GlsAddXdyAttribute{textup}
1113 \GlsAddXdyAttribute{textsl}
1114 \GlsAddXdyAttribute{textsc}
1115 \GlsAddXdyAttribute{emph}
1116 \GlsAddXdyAttribute{glshypernumber}
1117 \GlsAddXdyAttribute{hyperrrm}
1118 \GlsAddXdyAttribute{hypersf}
1119 \GlsAddXdyAttribute{hypertt}
1120 \GlsAddXdyAttribute{hyperbf}
1121 \GlsAddXdyAttribute{hypermd}
1122 \GlsAddXdyAttribute{hyperit}
1123 \GlsAddXdyAttribute{hyperup}
1124 \GlsAddXdyAttribute{hypersl}
1125 \GlsAddXdyAttribute{hypersc}
1126 \GlsAddXdyAttribute{hyperemph}

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

`\@xdyuseralphabets` List of additional alphabets

```
1132 \def\@xdyuseralphabets{}
```

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

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

```

```

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

```

This code is only required for xindy:

```

1142 \ifglsxindy

```

`@xindy@locationlist` List of predefined location names.

```

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

```

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

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

```

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

appendix-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  }

arabic-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{}

xdyuserlocationnames  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 lo-
location called name. The definition must use xindy syntax. (Note that this
doesn't check to see if the location is already defined. That is left to xindy to
complain about.)
1194 \ifglsxindy
1195   \newcommand*\GlsAddXdyLocation[3][{}]{%
1196     \def\@gls@tmp{#1}%

```

```

1197 \ifx\@gls@tmp\@empty
1198 \edef\@xdyuserlocationdefs{%
1199 \@xdyuserlocationdefs ^^J%
1200 (define-location-class \string"#2\string"^^J\space\space
1201 \space(:sep \string"{} \glsopenbrace\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 "\glsopenbrace"
1209 #1
1210 :sep "\glsclosebrace\glsopenbrace" #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 \glsnoxindywarning\GlsAddXdyLocation}
1222 \fi

```

`\locationclassorder` 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.

`\LocationClassOrder`

```

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

```

```

1239 \else
1240   \newcommand*\GlsSetXdyLocationClassOrder[1]{%
1241     \glsnoxywarning\GlsSetXdyLocationClassOrder}
1242 \fi

```

`\@xdysortrules` Define sort rules

```

1243 \ifglxindy
1244   \def\@xdysortrules{}
1245 \fi

```

`\GlsAddSortRule` Add a sort rule

```

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

```

`\@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     \glsnoxywarning\GlsAddXdyStyle}
1265 \fi

```

`\GlsSetXdyStyles` Reset the list of required styles

```

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

```

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

```
1273 \newcommand*\findrootlanguage{}
```

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

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

`\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     \glsnoxywarning\GlsSetXdyLanguage}
1286 \fi
```

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

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

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

```
1288 \ifglxindy
1289   \newcommand*\GlsSetXdyCodePage[1]{%
1290     \renewcommand*\gls@codepage{#1}%
1291   }
1292   Suggested by egreg:
1293   \AtBeginDocument{%
1294     \ifx\gls@codepage\@empty
1295       \@ifpackageloaded{fontspec}{\def\gls@codepage{utf8}}{}}
1296 \fi
```



```

1297 \else
1298   \newcommand*{\GlsSetXdyCodePage}[1]{%
1299     \glsnoxindywarning\GlsSetXdyCodePage}
1300 \fi

```

`\@xdylettergroups` Store letter group definitions.

```

1301 \ifglxsindy
1302   \ifglxs@xindy@glnumbers
1303     \def\@xdylettergroups{(define-letter-group
1304       \string"glnumbers\string"^^J\space\space\space
1305       :prefixes (\string"0\string" \string"1\string"
1306       \string"2\string" \string"3\string" \string"4\string"
1307       \string"5\string" \string"6\string" \string"7\string"
1308       \string"8\string" \string"9\string")^^J\space\space\space
1309       :before \string"\@glsfirstletter\string")}
1310   \else
1311     \def\@xdylettergroups{}
1312   \fi
1313 \fi

```

`\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 }

```

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

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

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

```

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

```

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

```
\foralllgsentries[glossary list]{cmd}{code}
```

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

```

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

```

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

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

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

```

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}
```

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

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

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

```
1342 \newcommand{\ifglentryexists}[3]{%
1343   \ifcsundef{glo@\glstoklabel{#1}@name}{#3}{#2}%
1344 }
```

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

```
\ifglused{<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*\ifglused}[3]{%
1346   \ifbool{glo@\glstoklabel{#1}@flag}{#2}{#3}%
1347 }
```

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

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

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

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

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

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

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

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

Generate a warning if entry specified by $\langle label \rangle$ doesn't exist, otherwise do $\langle code \rangle$.

```

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

```

$\backslash\text{ifglshaschildren}$ $\backslash\text{ifglshaschildren}\{\langle label \rangle\}\{\langle true part \rangle\}\{\langle false part \rangle\}$

```

1365 \newcommand{\ifglshaschildren}[3]{%
1366   \glsdoifexists{#1}%
1367   {%
1368     \def\do@glshaschildren{#3}%
1369     \edef\@gls@thislabel{\glsdetoklabel{#1}}%
1370     \expandafter\for@gl@sentries\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 }

```

$\backslash\text{ifglshasparent}$ $\backslash\text{ifglshasparent}\{\langle label \rangle\}\{\langle true part \rangle\}\{\langle false part \rangle\}$

```

1385 \newcommand{\ifglshasparent}[3]{%
1386   \glsdoifexists{#1}%
1387   {%
1388     \ifcsemtyp{glo@\glsdetoklabel{#1}@parent}{#3}{#2}%
1389   }%
1390 }

```

$\backslash\text{ifglshasdesc}$ $\backslash\text{ifglshasdesc}\{\langle label \rangle\}\{\langle true part \rangle\}\{\langle false part \rangle\}$

```

1391 \newcommand*\ifglshasdesc}[3]{%
1392   \ifcsemtyp{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@glstdetoklabel{#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@glstdetoklabel{#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@glstdetoklabel{#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@glstdetoklabel{#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 \glsdoifexists{#2}%
1433 {%
1434 \letcs{\@glo@thisvalue}{glo@glsdetoklabel{#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 \ifdequal\@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}{glo@glsdetoklabel{#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 \ifdequal\@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@forbidtextext\csname @glo@type@#2@log\endcsname
```



```

1509 \expandafter\@gls@forbidtextext\csname @glotype@#2@in\endcsname
1510 \expandafter\@gls@forbidtextext\csname @glotype@#2@out\endcsname
    Store the title:
1511 \expandafter\def\csname @glotype@#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 \LaTeX , `\@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 @glotype@#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 `\langle name \rangle \langle description \rangle`.

```
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 }
```

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

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

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

```
1673 \newcommand*\@glsnodesc{%
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@startswithexpandonce#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 }

```

startswithexpandonce

```

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

```

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

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

```
1764 \let\gls@assign@field\@gls@expand@fields
```

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

```

1765 \newcommand*{\glsexpandfields}{%
1766 \let\gls@assign@field\@gls@expand@fields
1767 }

```

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

```

1768 \newcommand*{\glsnoexpandfields}{%
1769 \let\gls@assign@field\@gls@noexpand@fields
1770 }

```

```

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

docnewglossaryentry  The definition of \newglossaryentry is changed at the start of the document
                    environment.
1777 \newcommand*{\gls@defdocnewglossaryentry}{%
1778   \let\newglossaryentry\new@glossaryentry
1779 }

provideglossaryentry  Like \newglossaryentry but does nothing if the entry has already been de-
                    fined.
1780 \newrobustcmd{\provideglossaryentry}[2]{%
1781   \ifglsentryexists{#1}%
1782   {%
1783   }%
1784   \gls@defglossaryentry{#1}{#2}%
1785   }%
1786 }
1787 \@onlypreamble{\provideglossaryentry}

\new@glossaryentry  For use in document environment.
1788 \newrobustcmd{\new@glossaryentry}[2]{%
1789   \ifundef\@gls@deffile
1790   {%
1791     \global\newwrite\@gls@deffile
1792     \immediate\openout\@gls@deffile=\jobname.glsdefs
1793   }%
1794   {%
1795     \ifglsentryexists{#1}{}%
1796     {%
1797       \gls@defglossaryentry{#1}{#2}%
1798     }%
1799     \@gls@writedef{#1}%
1800 }
1801 \AtBeginDocument
1802 {
1803   \makeatletter
1804   \InputIfFileExists{\jobname.glsdefs}{%}{%}
1805   \makeatother
1806   \gls@defdocnewglossaryentry
1807 }

```

```
1808 \AtEndDocument{\ifdef\@gls@deffile{\closeout\@gls@deffile}{}}
```

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

```
1809 \newcommand*\@gls@writedef}[1]{%
1810   \immediate\write\@gls@deffile
1811   {%
1812     \string\ifglsentryexists{#1}{}\glspercentchar^^J%
1813     \expandafter\@gobble\string\{\glspercentchar^^J%
1814     \string\gls@defglossaryentry{\glsdetoklabel{#1}}\glspercentchar^^J%
1815     \expandafter\@gobble\string\{\glspercentchar%
1816   }%

   Write key value information:

1817   \@for\@gls@map:=\@gls@keymap\do
1818   {%
1819     \edef\glo@value{\expandafter\expandonce
1820       \csname glo@\glsdetoklabel{#1}@\expandafter
1821         \@secondoftwo\@gls@map\endcsname}%
1822     \@onelevel@sanitize\glo@value
1823     \immediate\write\@gls@deffile
1824     {%
1825       \expandafter\@firstoftwo\@gls@map
1826       =\expandafter\@gobble\string\{\glo@value\expandafter\@gobble\string\},%
1827       \glspercentchar%
1828     }%
1829   }%
```

Provide hook:

```
1830   \gls@writedefhook
1831   \immediate\write\@gls@deffile
1832   {%
1833     \glspercentchar^^J%
1834     \expandafter\@gobble\string\}\glspercentchar^^J%
1835     \expandafter\@gobble\string\}\glspercentchar%
1836   }%
1837 }
```

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

```
1838 \newcommand*\@gls@keymap}{%
1839   {name}{name},%
1840   {sort}{sortvalue},% unescaped sort value
1841   {type}{type},%
1842   {first}{first},%
1843   {firstplural}{firstpl},%
1844   {text}{text},%
1845   {plural}{plural},%
1846   {description}{desc},%
1847   {descriptionplural}{descplural},%
1848   {symbol}{symbol},%
```

```

1849 {symbolplural}{symbolplural},%
1850 {user1}{useri},%
1851 {user2}{userii},%
1852 {user3}{useriii},%
1853 {user4}{useriv},%
1854 {user5}{userv},%
1855 {user6}{uservi},%
1856 {long}{long},%
1857 {longplural}{longpl},%
1858 {short}{short},%
1859 {shortplural}{shortpl},%
1860 {counter}{counter},%
1861 {parent}{parent}%
1862 }

```

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

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

```
1863 \newcommand*\gls@fetchfield}[2]{%
```

Ensure user field name is fully expanded

```
1864 \edef\@gls@thisval{#2}%
```

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

```

1865 \@for\@gls@map:=\@gls@keymap\do{%
1866 \edef\@this@key{\expandafter\@firstoftwo\@gls@map}%
1867 \ifdefequal{\@this@key}{\@gls@thisval}%
1868 {%
```

Found it.

```
1869 \edef#1{\expandafter\@secondoftwo\@gls@map}%
```

Break out of loop.

```

1870 \@endfortrue
1871 }%
1872 {}%
1873 }%
1874 }

```

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

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

```
1875 \newcommand*\glsaddstoragekey{\@ifstar\@sglsaddstoragekey\@glsaddstoragekey}
```

Starred version switches on expansion for this key.

```

1876 \newcommand*\@sglsaddstoragekey}[1]{%
1877 \key@ifundefined{glossentry}{#1}%

```

```

1878  {%
1879    \expandafter\newcommand\expandafter*\expandafter
1880    {\csname gls@assign@#1@field\endcsname}[2]{%
1881      \@gls@expand@field{##1}{#1}{##2}%
1882    }%
1883  }%
1884  }%
1885  \@gls@saddstoragekey{#1}%
1886 }

```

Unstarred version doesn't override default expansion.

```

1887 \newcommand*\@gls@saddstoragekey}[3]{%

```

Check the specified key doesn't already exist.

```

1888 \key@ifundefined{glossentry}{#1}%
1889 {%

```

Set up the key.

```

1890 \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
1891 \appto@gls@keymap{,#1}{#1}}%

```

Set the default value.

```

1892 \appto@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%

```

Assignment code.

```

1893 \appto@newglossaryentryposthook{%
1894   \letcs{\@glo@tmp}{@glo@#1}%
1895   \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
1896 }%

```

Define the no-link commands.

```

1897 \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
1898 }%
1899 {%
1900 \PackageError{glossaries}{Key '#1' already exists}{}%
1901 }%
1902 }

```

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

Allow user to add their own custom keys.

```

1903 \newcommand*\@gls@saddkey{\@ifstar\@sgls@saddkey\@gls@saddkey}

```

Starred version switches on expansion for this key.

```

1904 \newcommand*\@sgls@saddkey}[1]{%
1905 \key@ifundefined{glossentry}{#1}%
1906 {%
1907   \expandafter\newcommand\expandafter*\expandafter
1908   {\csname gls@assign@#1@field\endcsname}[2]{%
1909     \@gls@expand@field{##1}{#1}{##2}%

```

```

1910     }%
1911 }%
1912 {}%
1913 \@glsaddkey{#1}%
1914 }

```

Unstarred version doesn't override default expansion.

```
1915 \newcommand*\@glsaddkey}[7]{%
```

Check the specified key doesn't already exist.

```
1916 \key@ifundefined{glossentry}{#1}%
1917 {%
```

Set up the key.

```
1918 \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
1919 \appto@gls@keymap{, #1}{#1}}%
```

Set the default value.

```
1920 \appto@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
```

Assignment code.

```
1921 \appto@newglossaryentryposthook{%
1922 \letcs{\@glo@tmp}{@glo@#1}%
1923 \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
1924 }%
```

Define the no-link commands.

```
1925 \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
1926 \newcommand*{#4}[1]{\@Gls@entry@field{##1}{#1}}%
```

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

```

1927 \ifcsdef{@gls@user@#1@}%
1928 {%
1929 \PackageError{glossaries}%
1930 {Can't define '\string#5' as helper command
1931 '\expandafter\string\csname @gls@user@#1@\endcsname' already exists}%
1932 {}%
1933 }%
1934 {%

1935 \expandafter\newcommand\expandafter*\expandafter
1936 {\csname @gls@user@#1\endcsname}[2][ ]{%
1937 \new@ifnextchar[%
1938 {\csuse{@gls@user@#1@}{##1}{##2}}%
1939 {\csuse{@gls@user@#1@}{##1}{##2}[ ]}}%
1940 \csdef{@gls@user@#1@}##1##2[##3]{%
1941 \@gls@field@link{##1}{##2}{#3{##2}##3}%
1942 }%
1943 \newrobustcmd*{#5}{%
1944 \expandafter\@gls@hyp@opt\csname @gls@user@#1\endcsname}%
1945 }%

```

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

```

1946 \ifcsdef{@Gls@user@#1@}%
1947 {%
1948   \PackageError{glossaries}%
1949   {Can't define '\string#6' as helper command
1950   '\expandafter\string\csname @Gls@user@#1@\endcsname' already exists}%
1951   }%
1952 }%
1953 {%
1954   \expandafter\newcommand\expandafter*\expandafter
1955   {\csname @Gls@user@#1@\endcsname}[2][ ]{%
1956     \new@ifnextchar[%
1957     {\csuse{@Gls@user@#1@}{##1}{##2}}%
1958     {\csuse{@Gls@user@#1@}{##1}{##2}[ ]}}%
1959   \csdef{@Gls@user@#1@}##1##2[##3]{%
1960     \@gls@field@link{##1}{##2}{#4{##2}##3}%
1961   }%
1962   \newrobustcmd*{#6}{%
1963     \expandafter\@gls@hyp@opt\csname @Gls@user@#1@\endcsname}%
1964   }%

```

Finally the all caps version:

```

1965 \ifcsdef{@GLS@user@#1@}%
1966 {%
1967   \PackageError{glossaries}%
1968   {Can't define '\string#7' as helper command
1969   '\expandafter\string\csname @GLS@user@#1@\endcsname' already exists}%
1970   }%
1971 }%
1972 {%
1973   \expandafter\newcommand\expandafter*\expandafter
1974   {\csname @GLS@user@#1@\endcsname}[2][ ]{%
1975     \new@ifnextchar[%
1976     {\csuse{@GLS@user@#1@}{##1}{##2}}%
1977     {\csuse{@GLS@user@#1@}{##1}{##2}[ ]}}%
1978   \csdef{@GLS@user@#1@}##1##2[##3]{%
1979     \@gls@field@link{##1}{##2}{\mfirstucMakeUppercase{#3{##2}##3}}%
1980   }%
1981   \newrobustcmd*{#7}{%
1982     \expandafter\@gls@hyp@opt\csname @GLS@user@#1@\endcsname}%
1983   }%
1984 }%
1985 {%
1986   \PackageError{glossaries}{Key '#1' already exists}{}%
1987 }%
1988 }

```

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

```
1989 \newcommand{\glsfieldxdef}[3]{%
1990 \glsdoifexists{#1}%
1991 {%
1992   \edef\@glo@label{\glsdetoklabel{#1}}%
1993   \ifcsdef{glo@\@glo@label @#2}%
1994     {%
1995       \expandafter\xdef\csname glo@\@glo@label @#2\endcsname{#3}%
1996     }%
1997     {%
1998       \PackageError{glossaries}{Key '#2' doesn't exist}{}%
1999     }%
2000 }%
2001 }
```

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

```
2002 \newcommand{\glsfielddedef}[3]{%
2003 \glsdoifexists{#1}%
2004 {%
2005   \edef\@glo@label{\glsdetoklabel{#1}}%
2006   \ifcsdef{glo@\@glo@label @#2}%
2007     {%
2008       \expandafter\edef\csname glo@\@glo@label @#2\endcsname{#3}%
2009     }%
2010     {%
2011       \PackageError{glossaries}{Key '#2' doesn't exist}{}%
2012     }%
2013 }%
2014 }
```

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

```
2015 \newcommand{\glsfieldgdef}[3]{%
2016 \glsdoifexists{#1}%
2017 {%
2018   \edef\@glo@label{\glsdetoklabel{#1}}%
2019   \ifcsdef{glo@\@glo@label @#2}%
2020     {%
2021       \expandafter\gdef\csname glo@\@glo@label @#2\endcsname{#3}%
2022     }%
2023     {%
```



```

2024     \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2025   }%
2026 }%
2027 }

```

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

```

2028 \newcommand{\glsfielddef}[3]{%
2029   \glsdoifexists{#1}%
2030   {%
2031     \edef\@glo@label{\glsdetoklabel{#1}}%
2032     \ifcsdef{glo@\@glo@label @#2}%
2033     {%
2034       \expandafter\def\csname glo@\@glo@label @#2\endcsname{#3}%
2035     }%
2036   }%
2037   \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2038 }%
2039 }%
2040 }

```

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

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

```

2041 \newcommand{\glsfieldfetch}[3]{%
2042   \glsdoifexists{#1}%
2043   {%
2044     \edef\@glo@label{\glsdetoklabel{#1}}%
2045     \ifcsdef{glo@\@glo@label @#2}%
2046     {%
2047       \letcs#3{glo@\@glo@label @#2}%
2048     }%
2049   }%
2050   \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2051 }%
2052 }%
2053 }

```

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

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

```

2054 \newcommand{\ifglsfieldeq}[5]{%
2055   \glsdoifexists{#1}%
2056   {%

```

```

2057 \edef\@glo@label{\glsdetoklabel{#1}}%
2058 \ifcsdef{glo@\@glo@label @#2}%
2059 {%
2060 \ifcsstring{glo@\@glo@label @#2}{#3}{#4}{#5}%
2061 }%
2062 {%
2063 \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2064 }%
2065 }%
2066 }

```

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

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

```

2067 \newcommand{\ifglsfielddefeq}[5]{%
2068 \glsdoifexists{#1}%
2069 {%
2070 \edef\@glo@label{\glsdetoklabel{#1}}%
2071 \ifcsdef{glo@\@glo@label @#2}%
2072 {%
2073 \expandafter\ifdefstrequal
2074 \csname glo@\@glo@label @#2\endcsname{#3}{#4}{#5}%
2075 }%
2076 {%
2077 \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2078 }%
2079 }%
2080 }

```

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

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

```

2081 \newcommand{\ifglsfieldcseq}[5]{%
2082 \glsdoifexists{#1}%
2083 {%
2084 \edef\@glo@label{\glsdetoklabel{#1}}%
2085 \ifcsdef{glo@\@glo@label @#2}%
2086 {%
2087 \ifcsstrequal{glo@\@glo@label @#2}{#3}{#4}{#5}%
2088 }%
2089 {%
2090 \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2091 }%
2092 }%
2093 }

```

`\glswritedefhook`

```
2094 \newcommand*\glswritedefhook{}
```

`\gls@assign@desc`

```
2095 \newcommand*\gls@assign@desc}[1]{%
2096   \gls@assign@field{#1}{desc}{\@glo@desc}%
2097   \gls@assign@field{\@glo@desc}{#1}{descplural}{\@glo@descplural}%
2098 }
```

`\longnewglossaryentry`

```
2099 \newcommand*\longnewglossaryentry}[3]{%
2100   \glsdoifnoexists{#1}%
2101   {%
2102     \bgroup
2103     \let\@org@newglossaryentryprehook\@newglossaryentryprehook
2104     \long\def\@newglossaryentryprehook{%
2105       \long\def\@glo@desc{#3\leavevmode\unskip\nopostdesc}%
2106       \@org@newglossaryentryprehook
2107     }%
2108     \renewcommand*\gls@assign@desc}[1]{%
2109       \global\cslet{glo@glstdetoklabel{#1}@desc}{\@glo@desc}%
2110       \global\cslet{glo@glstdetoklabel{#1}@descplural}{\@glo@desc}%
2111     }
2112     \gls@defglossaryentry{#1}{#2}%
2113   \egroup
2114 }
2115 }
```

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

```
2116 \@onlypreamble{\longnewglossaryentry}
```

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

```
2117 \newcommand*\longprovideglossaryentry}[3]{%
2118   \ifglstentryexists{#1}{}%
2119   {\longnewglossaryentry{#1}{#2}{#3}}%
2120 }
2121 \@onlypreamble{\longprovideglossaryentry}
```

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

Defines a new entry without checking if it already exists.

```
2122 \newcommand*\gls@defglossaryentry}[2]{%
```

Store label

```
2123   \edef\@glo@label{\glstdetoklabel{#1}}%
```

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

```
2124   \let\glslabel\@glo@label
```

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

```
2125 \let\@glo@name\@glsnname
2126 \let\@glo@desc\@glsnodesc

2127 \let\@glo@descplural\@gls@default@value
2128 \let\@glo@type\@gls@default@value
2129 \let\@glo@symbol\@gls@default@value

2130 \let\@glo@symbolplural\@gls@default@value
2131 \let\@glo@text\@gls@default@value
2132 \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.)

```
2133 \let\@glo@first\@gls@default@value
2134 \let\@glo@firstplural\@gls@default@value
```

Set the default sort:

```
2135 \let\@glo@sort\@gls@default@value
```

Set the default counter:

```
2136 \let\@glo@counter\@gls@default@value
```

```
2137 \def\@glo@see{}%
2138 \def\@glo@parent{}%
2139 \def\@glo@prefix{}%

2140 \def\@glo@useri{}%
2141 \def\@glo@userii{}%
2142 \def\@glo@useriii{}%
2143 \def\@glo@useriv{}%
2144 \def\@glo@userv{}%
2145 \def\@glo@uservi{}%

2146 \def\@glo@short{}%
2147 \def\@glo@shortpl{}%
2148 \def\@glo@long{}%
2149 \def\@glo@longpl{}
```

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

```
2150 \@newglossaryentryprehook
```

Extract key-val information from third parameter:

```
2151 \setkeys{glossentry}{#2}%
```

Check there is a default glossary.

```
2152 \ifundef\glsdefaulttype
2153 {%
2154   \PackageError{glossaries}%
2155     {No default glossary type (have you used ‘nomain’?)}%
2156     {If you use package option ‘nomain’ you must define
2157      a new glossary before you can define entries}%
2158   }%
2159   {}%
```

Assign type. This must be fully expandable

```
2160 \gls@assign@field{\glsdefaulttype}{\@glo@label}{type}{\@glo@type}%
2161 \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.

```
2162 \ifcsundef{glo@list@\@glo@type}%
2163 {%
2164   \PackageError{glossaries}%
2165     {Glossary type ‘\@glo@type’ has not been defined}%
2166     {You need to define a new glossary type, before making entries
2167      in it}%
2168   }%
2169   {}%
```

Check if it's an ignored glossary

```
2170 \ifignoredglossary\@glo@type
2171 {%
```

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

```
2172   \ifx\@glo@desc\@glsnodesc
2173     \let\@glo@desc\@empty
2174   \fi
2175   }%
2176   {%
2177   }%
2178   \protected@edef\@glo@list@{\csname glo@list@\@glo@type\endcsname}%
2179   \expandafter\xdef\csname glo@list@\@glo@type\endcsname{%
2180     \@glo@list@\@glo@label},}%
2181   }%
```

Initialise level to 0.

```
2182 \gls@level=0\relax
```

Has this entry been assigned a parent?

```
2183 \ifx\@glo@parent\@empty
```

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

```
2184 \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%
2185 \else
```

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

```
2186 \ifdefequal\@glo@label\@glo@parent%
2187 {%
2188 \PackageError{glossaries}{Entry '\@glo@label' can't be its own parent}{}%
2189 \def\@glo@parent{}}%
2190 \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}}%
2191 }%
2192 {%
```

Check the parent exists:

```
2193 \ifglentryexists{\@glo@parent}%
2194 {%
```

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

```
2195 \expandafter\xdef\csname glo@\@glo@label @parent\endcsname{%
2196 \@glo@parent}%
```

Determine level.

```
2197 \gls@level=\csname glo@\@glo@parent @level\endcsname\relax
2198 \advance\gls@level by 1\relax
```

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

```
2199 \ifx\@glo@name\@gls@name
2200 \expandafter\let\expandafter\@glo@name
2201 \csname glo@\@glo@parent @name\endcsname
```

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

```
2202 \ifx\@glo@plural\@gls@default@value
2203 \expandafter\let\expandafter\@glo@plural
2204 \csname glo@\@glo@parent @plural\endcsname
2205 \fi
2206 \fi
2207 }%
2208 {%
```

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

```
2209 \PackageError{glossaries}%
2210 {%
2211 Invalid parent '\@glo@parent'
2212 for entry '\@glo@label' - parent doesn't exist%
2213 }%
2214 {%
2215 Parent entries must be defined before their children%
2216 }%
2217 \def\@glo@parent{}}%
2218 \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}}%
2219 }%
2220 }%
2221 \fi
```

Set the level for this entry

```
2222 \expandafter\edef\csname glo@\glo@label @level\endcsname{\number\gls@level}%
```

Define commands associated with this entry:

```
2223 \gls@assign@field{\@glo@name}{\@glo@label}{sortvalue}{\@glo@sort}%
2224 \letcs\@glo@sort{glo@\@glo@label @sortvalue}%
2225 \gls@assign@field{\@glo@name}{\@glo@label}{text}{\@glo@text}%
2226 \expandafter\gls@assign@field\expandafter
2227   {\csname glo@\@glo@label @text\endcsname\glspluralsuffix}%
2228   {\@glo@label}{plural}{\@glo@plural}%
2229 \expandafter\gls@assign@field\expandafter
2230   {\csname glo@\@glo@label @text\endcsname}%
2231   {\@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.

```
2232 \ifx\@glo@first\@gls@default@value
2233   \expandafter\gls@assign@field\expandafter
2234     {\csname glo@\@glo@label @plural\endcsname}%
2235     {\@glo@label}{firstpl}{\@glo@firstplural}%
2236 \else
2237   \expandafter\gls@assign@field\expandafter
2238     {\csname glo@\@glo@label @first\endcsname\glspluralsuffix}%
2239     {\@glo@label}{firstpl}{\@glo@firstplural}%
2240 \fi

2241 \ifcsundef{@glo@type@\@glo@type @counter}%
2242 {%
2243   \def\@glo@defaultcounter{\glscounter}%
2244   }%
2245 {%
2246   \letcs\@glo@defaultcounter{@glo@type@\@glo@type @counter}%
2247   }%
2248 \gls@assign@field{\@glo@defaultcounter}{\@glo@label}{counter}{\@glo@counter}%
2249 \gls@assign@field{}{\@glo@label}{useri}{\@glo@useri}%
2250 \gls@assign@field{}{\@glo@label}{userii}{\@glo@userii}%
2251 \gls@assign@field{}{\@glo@label}{useriii}{\@glo@useriii}%
2252 \gls@assign@field{}{\@glo@label}{useriv}{\@glo@useriv}%
2253 \gls@assign@field{}{\@glo@label}{userv}{\@glo@userv}%
2254 \gls@assign@field{}{\@glo@label}{uservi}{\@glo@uservi}%
2255 \gls@assign@field{}{\@glo@label}{short}{\@glo@short}%
2256 \gls@assign@field{}{\@glo@label}{shortpl}{\@glo@shortpl}%
2257 \gls@assign@field{}{\@glo@label}{long}{\@glo@long}%
2258 \gls@assign@field{}{\@glo@label}{longpl}{\@glo@longpl}%
2259 \ifx\@glo@name\@gls@name
2260   \@gls@name
2261   \let\@glo@name\@gls@default@value
2262 \fi
2263 \gls@assign@field{}{\@glo@label}{name}{\@glo@name}%
```

Set default numberlist if not defined:

```
2264 \ifcsundef{glo@\glo@label @numberlist}%
2265 {%
2266 \csxdef{glo@\glo@label @numberlist}{%
2267 \noexpand\gls@missingnumberlist{\glo@label}}%
2268 }%
2269 {}%
```

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.

```
2270 \def\glo@@desc{\glo@first}%
2271 \ifx\glo@desc\glo@@desc
2272 \let\glo@desc\glo@first
2273 \fi
2274 \ifx\glo@desc\glsnodesc
2275 \glsnodesc
2276 \let\glo@desc\gls@default@value
2277 \fi
2278 \gls@assign@desc{\glo@label}%
```

Set the sort key for this entry:

```
2279 \gls@defsort{\glo@type}{\glo@label}%

2280 \def\glo@@symbol{\glo@text}%
2281 \ifx\glo@symbol\glo@@symbol
2282 \let\glo@symbol\glo@text
2283 \fi
2284 \gls@assign@field{\relax}{\glo@label}{symbol}{\glo@symbol}%
2285 \expandafter
2286 \gls@assign@field\expandafter
2287 {\csname glo@\glo@label @symbol\endcsname}
2288 {\glo@label}{symbolplural}{\glo@symbolplural}%
```

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

```
2289 \expandafter\xdef\csname glo@\glo@label @flagfalse\endcsname{%
2290 \noexpand\global
2291 \noexpand\let\expandafter\noexpand
2292 \csname ifglo@\glo@label @flag\endcsname\noexpand\iffalse
2293 }%
2294 \expandafter\xdef\csname glo@\glo@label @flagtrue\endcsname{%
2295 \noexpand\global
2296 \noexpand\let\expandafter\noexpand
2297 \csname ifglo@\glo@label @flag\endcsname\noexpand\iftrue
2298 }%
2299 \csname glo@\glo@label @flagfalse\endcsname
```

Sort out any cross-referencing if required.

```
2300 \ifdefined\glo@see
```



```

2301   {}%
2302   {%
2303     \protected@edef\@do@glsee{%
2304       \noexpand\@gls@fixbraces\noexpand\@glo@list\@glo@see
2305         \noexpand\@nil
2306       \noexpand\expandafter\noexpand\@glsee\noexpand\@glo@list{\@glo@label}}%
2307     \@do@glsee
2308   }%

```

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

```

2309   \ifignoredglossary\@glo@type
2310   {%
2311     \csdef{glo@\@glo@label @index}{}%
2312   }
2313   {%
2314     \do@glo@storeentry{\@glo@label}%
2315   }%

```

Define entry counters if enabled:

```

2316   \@newglossaryentry@defcounters

```

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

```

2317   \@newglossaryentryposthook
2318 }

```

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

```

2319 \newcommand*\@newglossaryentryprehook{}

```

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

```

2320 \newcommand*\@newglossaryentryposthook{}

```

`\glossaryentry@defcounters`

```

2321 \newcommand*\@newglossaryentry@defcounters{}

```

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

```

2322 \newcommand*\glsmoveentry[2]{%
2323   \edef\@glo@thislabel{\glsdetoklabel{#1}}%
2324   \edef\@glo@type{\csname glo@\@glo@thislabel @type\endcsname}%
2325   \def\@glo@list{,%}
2326   \forglseries[\@glo@type]{\@glo@label}%
2327   {%
2328     \ifdefequal\@glo@thislabel\@glo@label
2329       {\eappto\@glo@list{\@glo@label,}}%
2330     }%
2331   \cslet{glo@list\@glo@type}{\@glo@list}%
2332   \csdef{glo@\@glo@thislabel @type}{#2}%
2333 }

```

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

```
2334 \ifglxindy
2335   \newcommand*{\@glossaryentryfield}{\string\glossentry}
2336 \else
2337   \newcommand*{\@glossaryentryfield}{\string\glossentry}
2338 \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.)

```
2339 \ifglxindy
2340   \newcommand*{\@glossarysubentryfield}{%
2341     \string\subglossentry}
2342 \else
2343   \newcommand*{\@glossarysubentryfield}{%
2344     \string\subglossentry}
2345 \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.)

```
2346 \newcommand{\@glo@storeentry}[1]{%
```

Escape `makeindex/xindy` special characters in the label:

```
2347   \edef\@glo@esclabel{#1}%
2348   \@gls@checkmkidxchars\@glo@esclabel
```

Get the sort string and escape any special characters

```
2349   \protected@edef\@glo@sort{\csname glo@#1@sort\endcsname}%
2350   \@gls@checkmkidxchars\@glo@sort
```

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

```
2351   \@gls@checkmkidxchars\@glo@prefix
```

Get the parent, if one exists

```
2352   \edef\@glo@parent{\csname glo@#1@parent\endcsname}%
```

Write the information to the glossary file.

```
2353   \ifglxindy
```

Store using `xindy` syntax.

```
2354     \ifx\@glo@parent\@empty
```

```

Entry doesn't have a parent
2355     \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2356         (\string"\@glo@sort\string" %
2357         \string"\@glo@prefix@glossaryentryfield{\@glo@esclabel}\string") %
2358     }%
2359     \else

Entry has a parent
2360     \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2361         \csname glo@\@glo@parent @index\endcsname
2362         (\string"\@glo@sort\string" %
2363         \string"\@glo@prefix@glossarysubentryfield
2364         {\csname glo@#1@level\endcsname}{\@glo@esclabel}\string") %
2365     }%
2366     \fi
2367     \else

Store using makeindex syntax.
2368     \ifx\@glo@parent\@empty

Sanitize \@glo@prefix
2369     \@onelevel@sanitize\@glo@prefix

Entry doesn't have a parent
2370     \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2371         \@glo@sort\@gls@actualchar\@glo@prefix
2372         \@glossaryentryfield{\@glo@esclabel}%
2373     }%
2374     \else

Entry has a parent
2375     \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2376         \csname glo@\@glo@parent @index\endcsname\@gls@levelchar
2377         \@glo@sort\@gls@actualchar\@glo@prefix
2378         \@glossarysubentryfield
2379         {\csname glo@#1@level\endcsname}{\@glo@esclabel}%
2380     }%
2381     \fi
2382     \fi
2383 }

```

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
```

```

2384 \AtBeginDocument{%
2385   \ifpackageloaded{amsmath}%
2386   {\let\gls@ifnotmeasuring\@gls@ifnotmeasuring}%
2387   }%
2388 }
2389 \newcommand*\@gls@ifnotmeasuring[1]{%
2390   \ifmeasuring@
2391   \else
2392     #1%
2393   \fi
2394 }
2395 \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.

```

2396 \newcommand*\glsreset[1]{%
2397   \gls@ifnotmeasuring
2398   {%
2399     \glsdoifexists{#1}%
2400     {%
2401       \@glsreset{#1}%
2402     }%
2403   }%
2404 }

```

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

```

2405 \newcommand*\glslocalreset[1]{%
2406   \gls@ifnotmeasuring
2407   {%
2408     \glsdoifexists{#1}%
2409     {%
2410       \@glslocalreset{#1}%
2411     }%
2412   }%
2413 }

```

`\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.

```

2414 \newcommand*\glsunset[1]{%
2415   \gls@ifnotmeasuring
2416   {%
2417     \glsdoifexists{#1}%
2418     {%
2419       \@glsunset{#1}%
2420     }%
2421   }%
2422 }

```

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

```

2423 \newcommand*{\glslocalunset}[1]{%
2424   \gls@ifnotmeasuring
2425   {%
2426     \glsdoifexists{#1}%
2427     {%
2428       \@glslocalunset{#1}%
2429     }%
2430   }%
2431 }

```

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

```
2432 \newcommand*{\@glslocalunset}{\@@glslocalunset}
```

`\@@glslocalunset` Local unset without checks.

```

2433 \newcommand*{\@@glslocalunset}[1]{%
2434   \expandafter\let\csname ifglo@glsdetoklabel{#1}@flag\endcsname\iftrue
2435 }

```

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

```
2436 \newcommand*{\@glsunset}{\@@glsunset}
```

`\@@glsunset` Global unset without checks.

```

2437 \newcommand*{\@@glsunset}[1]{%
2438   \expandafter\global\csname glo@glsdetoklabel{#1}@flagtrue\endcsname
2439 }

```

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

```
2440 \newcommand*{\@glslocalreset}{\@@glslocalreset}
```

`\@@glslocalreset` Local reset without checks.

```

2441 \newcommand*{\@@glslocalreset}[1]{%
2442   \expandafter\let\csname ifglo@glsdetoklabel{#1}@flag\endcsname\iffalse
2443 }

```

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

```
2444 \newcommand*{\@glsreset}{\@@glsreset}
```

`\@@glsreset` Global reset without checks.

```

2445 \newcommand*{\@@glsreset}[1]{%
2446   \expandafter\global\csname glo@glsdetoklabel{#1}@flagfalse\endcsname
2447 }

```

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

`\glsresetall`

```
2448 \newcommand*\glsresetall}[1][\@glo@types]{%
2449   \forallglsentries[#1]{\@glsentry}%
2450   {%
2451     \glsreset{\@glsentry}%
2452   }%
2453 }
```

As above, but with only a local effect:

`\glslocalresetall`

```
2454 \newcommand*\glslocalresetall}[1][\@glo@types]{%
2455   \forallglsentries[#1]{\@glsentry}%
2456   {%
2457     \glslocalreset{\@glsentry}%
2458   }%
2459 }
```

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

Syntax: `\glsunsetall` [*<glossary-list>*]

`\glsunsetall`

```
2460 \newcommand*\glsunsetall}[1][\@glo@types]{%
2461   \forallglsentries[#1]{\@glsentry}%
2462   {%
2463     \glsunset{\@glsentry}%
2464   }%
2465 }
```

As above, but with only a local effect:

`\glslocalunsetall`

```
2466 \newcommand*\glslocalunsetall}[1][\@glo@types]{%
2467   \forallglsentries[#1]{\@glsentry}%
2468   {%
2469     \glslocalunset{\@glsentry}%
2470   }%
2471 }
```

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

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

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

```
2472 \newcommand*\@newglossaryentry@defcounters}{%
2473   \csdef{glo@\@glo@label @currcount}{0}%
2474   \csdef{glo@\@glo@label @prevcount}{0}%
2475 }
```

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

```
2476 \newcommand*\glsenableentrycount}{%
```

Enable new entry fields.

```
2477 \let\@newglossaryentry@defcounters\@newglossaryentry@defcounters
```

Disable \newglossaryentry in the document environment.

```
2478 \renewcommand*\gls@defdocnewglossaryentry}{%
2479   \renewcommand*\newglossaryentry[2]{%
2480     \PackageError{glossaries}{\string\newglossaryentry\space
2481     may only be used in the preamble when entry counting has
2482     been activated}{If you use \string\glsenableentrycount\space
2483     you must place all entry definitions in the preamble not in
2484     the document environment}%
2485   }%
2486 }
```

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

```
2487 \newcommand*\glsentrycurrcount}[1]{%
2488   \ifcsundef{glo@\glsdetoklabel{##1}@currcount}%
2489   {0}{\@gls@entry@field{##1}{currcount}}%
2490 }%
2491 \newcommand*\glsentryprevcount}[1]{%
2492   \ifcsundef{glo@\glsdetoklabel{##1}@prevcount}%
2493   {0}{\@gls@entry@field{##1}{prevcount}}%
2494 }
```

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

```
2495 \renewcommand*\@glsunset}[1]{%
2496   \@glsunset{##1}%
2497   \@gls@increment@currcount{##1}%
2498 }%
2499 \renewcommand*\@glslocalunset}[1]{%
2500   \@glslocalunset{##1}%
2501   \@gls@local@increment@currcount{##1}%
2502 }%
2503 \renewcommand*\@glsreset}[1]{%
2504   \@glsreset{##1}%
2505   \csgdef{glo@\glsdetoklabel{##1}@currcount}{0}%
2506 }%
2507 \renewcommand*\@glslocalreset}[1]{%
2508   \@glslocalreset{##1}%
```

```

2509 \csdef{glo\glsdetoklabel{##1}@currcount}{0}%
2510 }%

```

Alter behaviour of `\cgl's`. (Only global unset is used if previous count was one as it doesn't make sense to have a local unset here given that the previous count was global.)

```

2511 \def\@cgl's@##1##2[##3]{%
2512 \ifnum\glsentryprevcount{##2}=1\relax
2513 \cgl'sformat{##2}{##3}%
2514 \glsunset{##2}%
2515 \else
2516 \@gls@{##1}{##2}[##3]%
2517 \fi
2518 }%

```

Similarly for the analogous commands. No case change plural:

```

2519 \def\@cgl'spl@##1##2[##3]{%
2520 \ifnum\glsentryprevcount{##2}=1\relax
2521 \cgl'splformat{##2}{##3}%
2522 \glsunset{##2}%
2523 \else
2524 \@glspl@{##1}{##2}[##3]%
2525 \fi
2526 }%

```

First letter uppercase singular:

```

2527 \def\@cGls@##1##2[##3]{%
2528 \ifnum\glsentryprevcount{##2}=1\relax
2529 \cGlsformat{##2}{##3}%
2530 \glsunset{##2}%
2531 \else
2532 \@Gls@{##1}{##2}[##3]%
2533 \fi
2534 }%

```

First letter uppercase plural:

```

2535 \def\@cGlspl@##1##2[##3]{%
2536 \ifnum\glsentryprevcount{##2}=1\relax
2537 \cGlsplformat{##2}{##3}%
2538 \glsunset{##2}%
2539 \else
2540 \@Glspl@{##1}{##2}[##3]%
2541 \fi
2542 }%

```

Write information to aux file at the end of the document

```

2543 \AtEndDocument{\@gls@write@entrycounts}%

```

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

```

2544 \renewcommand*{\@gls@entry@count}[2]{%

```



```
2545 \csgdef{glo@glstetoklabel{##1}@prevcount}{##2}%
2546 }%
```

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

```
2547 \let\glsenableentrycount\relax
2548 }
2549 \@onlypreamble\glsenableentrycount
```

increment@currcount

```
2550 \newcommand*{\@gls@increment@currcount}[1]{%
2551 \csxdef{glo@glstetoklabel{##1}@currcount}{%
2552 \number\numexpr\glstentrycurrcount{##1}+1}%
2553 }
```

increment@currcount

```
2554 \newcommand*{\@gls@local@increment@currcount}[1]{%
2555 \csedef{glo@glstetoklabel{##1}@currcount}{%
2556 \number\numexpr\glstentrycurrcount{##1}+1}%
2557 }
```

s@write@entrycounts

Write the entry counts to the aux file. Use \immediate since this occurs right at the end of the document. Only write information for entries that have been used. (Some users have a file containing vast numbers of entries, many of which may not be used. There's no point writing information about the entries that haven't been used and it will only slow things down.)

```
2558 \newcommand*{\@gls@write@entrycounts}{%
2559 \immediate\write\@auxout
2560 {\string\providecommand*\string\@gls@entry@count}[2]{}%
2561 \forallglsentries{\@glstentry}{%
2562 \ifglsused{\@glstentry}%
2563 {\immediate\write\@auxout
2564 {\string\@gls@entry@count{\@glstentry}{\glstentrycurrcount{\@glstentry}}}%
2565 }%
2566 }%
2567 }
```

\@gls@entry@count

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

```
2568 \newcommand*{\@gls@entry@count}[2]{}%
```

\cgl

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

```
2569 \newrobustcmd*{\cgl}{\@gls@hyp@opt\@cgl}
```

\@cgl

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

```
2570 \newcommand*{\@cgl}[2] []{%
2571 \new@ifnextchar[{\@cgl@{##1}{##2}}{\@cgl@{##1}{##2} []}%
2572 }
```

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

```

2573 \def\@cgl@s@#1#2[#3]{%
2574 \GlossariesWarning{\string\cgl@s\space is defaulting to
2575 \string\gls\space since you haven't enabled entry counting}%
2576 \@gls@{#1}{#2}[#3]%
2577 }

```

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

```

2578 \newcommand*\cgl@sformat}[2]{%
2579 \ifglshaslong{#1}{\gl@sentrylong{#1}}{\gl@sentryfirst{#1}}#2%
2580 }

```

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

```

2581 \newrobustcmd*\cGl@s{\@gls@hyp@opt\@cGl@s}

```

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

```

2582 \newcommand*\@cGl@s}[2][ ]{%
2583 \new@ifnextchar[{\@cGl@s@{#1}{#2}}{\@cGl@s@{#1}{#2}[ ]}%
2584 }

```

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

```

2585 \def\@cGl@s@#1#2[#3]{%
2586 \GlossariesWarning{\string\cGl@s\space is defaulting to
2587 \string\Gls\space since you haven't enabled entry counting}%
2588 \@Gls@{#1}{#2}[#3]%
2589 }

```

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

```

2590 \newcommand*\cGl@sformat}[2]{%
2591 \ifglshaslong{#1}{\Gl@sentrylong{#1}}{\Gl@sentryfirst{#1}}#2%
2592 }

```

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

```

2593 \newrobustcmd*\cgl@spl{\@gls@hyp@opt\@cgl@spl}

```

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

```

2594 \newcommand*\@cgl@spl}[2][ ]{%

```

```
2595 \new@ifnextchar[{\@cglsp1@{#1}{#2}}{\@cglsp1@{#1}{#2} []}]%
2596 }
```

`\@cglsp1@` Read in the final optional argument. This defaults to same behaviour as `\glsp1` but issues a warning.

```
2597 \def\@cglsp1@#1#2[#3]{%
2598 \GlossariesWarning{\string\cglsp1\space is defaulting to
2599 \string\glsp1\space since you haven't enabled entry counting}%
2600 \@glsp1@{#1}{#2}[#3]%
2601 }
```

`\cglsp1format` Format used by `\cglsp1` if entry only used once on previous run. The first argument is the label, the second argument is the insert text.

```
2602 \newcommand*{\cglsp1format}[2]{%
2603 \ifglshaslong{#1}{\glsenrylongpl{#1}}{\glsenryfirstplural{#1}}#2%
2604 }
```

`\cGlsp1` Define command that works like `\Glsp1` but behaves differently if the entry count function is enabled. (If not enabled, it behaves the same as `\Glsp1` but issues a warning.)

```
2605 \newrobustcmd*{\cGlsp1}{\@gls@hyp@opt\@cGlsp1}
```

`\@cglsp1` Defined the un-starred form. Need to determine if there is a final optional argument

```
2606 \newcommand*{\@cGlsp1}[2] []{%
2607 \new@ifnextchar[{\@cGlsp1@{#1}{#2}}{\@cGlsp1@{#1}{#2} []}]%
2608 }
```

`\@cGlsp1@` Read in the final optional argument. This defaults to same behaviour as `\Glsp1` but issues a warning.

```
2609 \def\@cGlsp1@#1#2[#3]{%
2610 \GlossariesWarning{\string\cGlsp1\space is defaulting to
2611 \string\Glsp1\space since you haven't enabled entry counting}%
2612 \@Glsp1@{#1}{#2}[#3]%
2613 }
```

`\cGlsp1format` Format used by `\cGlsp1` if entry only used once on previous run. The first argument is the label, the second argument is the insert text.

```
2614 \newcommand*{\cGlsp1format}[2]{%
2615 \ifglshaslong{#1}{\Glsenrylongpl{#1}}{\Glsenryfirstplural{#1}}#2%
2616 }
```

1.10 Loading files containing glossary entries

Glossary entries can be defined in an external file. These external files can contain `\newglossaryentry` and `\newacronym` commands.¹

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

```
\loadglsentries[⟨type⟩]{⟨filename⟩}
```

This command will input the file using `\input`. The optional argument specifies to which glossary the entries should be assigned if they haven't used the type key. If the optional argument is not specified, the default glossary is used. Only those entries used in the document (via `\glslink`, `\gls`, `\glspl` and uppercase variants or `\glsadd` and `\glsaddall` will appear in the glossary). The mandatory argument is the filename (with or without `.tex` extension).

```
\loadglsentries
```

```
2617 \newcommand*\loadglsentries}[2][\@gls@default]{%
2618   \let\@gls@default\glsdefaulttype
2619   \def\glsdefaulttype{#1}\input{#2}%
2620   \let\glsdefaulttype\@gls@default
2621 }
```

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

```
2622 \@onlypreamble{\loadglsentries}
```

1.11 Using glossary entries in the text

Any term that has been defined using `\newglossaryentry` (or `\newacronym`) can be displayed in the text (i.e. outside of the glossary) using one of the commands defined in this section. Unless you use `\glslink`, the way the term appears in the text is determined by `\glsdisplayfirst` (if it is the first time the term has been used) or `\glsdisplay` (for subsequent use). Any formatting commands (such as `\textbf` is governed by `\glstextformat`. By default this just displays the link text “as is”.

```
\glstextformat
```

```
2623 \newcommand*\glstextformat}[1]{#1}
```

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

```
2624 \newcommand*\glsentryfmt{%
2625   \@@gls@default@entryfmt\glsdisplayfirst\glsdisplay
2626 }
```

Format that provides backwards compatibility:

```
2627 \newcommand*\@@gls@default@entryfmt}[2]{%
2628   \ifdefempty\glscustomtext
2629   {%
2630     \glsifplural
2631     {%
```

Plural form

```
2632     \glscapscase
2633     {%
```

Don't adjust case

```
2634     \ifglused\glslabel
2635     {%
```

Subsequent use

```
2636     #2{\glsentryplural{\glslabel}}%
2637     {\glsentrydescplural{\glslabel}}%
2638     {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2639     }%
2640     {%
```

First use

```
2641     #1{\glsentryfirstplural{\glslabel}}%
2642     {\glsentrydescplural{\glslabel}}%
2643     {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2644     }%
2645     }%
2646     {%
```

Make first letter upper case

```
2647     \ifglused\glslabel
2648     {%
```

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.)

```
2649     \ifbool{glscompatible-3.07}%
2650     {%
2651     \protected@edef\@glo@etext{%
2652     #2{\glsentryplural{\glslabel}}%
2653     {\glsentrydescplural{\glslabel}}%
2654     {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2655     \xmakefirstuc\@glo@etext
2656     }%
2657     {%
2658     #2{\Glsentryplural{\glslabel}}%
2659     {\glsentrydescplural{\glslabel}}%
2660     {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2661     }%
2662     }%
2663     {%
```

First use

```
2664     \ifbool{glscompatible-3.07}%
2665     {%
2666     \protected@edef\@glo@etext{%
```

```

2667         #1{\glsentryfirstplural{\glslabel}}%
2668         {\glsentrydescplural{\glslabel}}%
2669         {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2670     \xmakefirstuc\@glo@etext
2671     }%
2672     {%
2673         #1{\Glsentryfirstplural{\glslabel}}%
2674         {\glsentrydescplural{\glslabel}}%
2675         {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2676     }%
2677     }%
2678     }%
2679     {%

```

Make all upper case

```

2680     \ifglsused\glslabel
2681     {%

```

Subsequent use

```

2682         \mfirstucMakeUppercase{#2{\glsentryplural{\glslabel}}%
2683         {\glsentrydescplural{\glslabel}}%
2684         {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2685     }%
2686     {%

```

First use

```

2687         \mfirstucMakeUppercase{#1{\glsentryfirstplural{\glslabel}}%
2688         {\glsentrydescplural{\glslabel}}%
2689         {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2690     }%
2691     }%
2692     }%
2693     {%

```

Singular form

```

2694     \gls caps case
2695     {%

```

Don't adjust case

```

2696     \ifglsused\glslabel
2697     {%

```

Subsequent use

```

2698         #2{\glsentrytext{\glslabel}}%
2699         {\glsentrydesc{\glslabel}}%
2700         {\glsentrysymbol{\glslabel}}{\glsinsert}}%
2701     }%
2702     {%

```

First use

```

2703         #1{\glsentryfirst{\glslabel}}%
2704         {\glsentrydesc{\glslabel}}%

```

```

2705         {\glsentrysymbol{\glslabel}}{\glsinsert}%
2706     }%
2707 }%
2708 {%

```

Make first letter upper case

```

2709     \ifglsused\glslabel
2710     {%

```

Subsequent use

```

2711     \ifbool{glscompatible-3.07}%
2712     {%
2713         \protected@edef\@glo@etext{%
2714             #2{\glsentrytext{\glslabel}}%
2715             {\glsentrydesc{\glslabel}}%
2716             {\glsentrysymbol{\glslabel}}{\glsinsert}}%
2717         \xmakefirstuc\@glo@etext
2718     }%
2719     {%
2720         #2{\Glsentrytext{\glslabel}}%
2721         {\glsentrydesc{\glslabel}}%
2722         {\glsentrysymbol{\glslabel}}{\glsinsert}%
2723     }%
2724 }%
2725 {%

```

First use

```

2726     \ifbool{glscompatible-3.07}%
2727     {%
2728         \protected@edef\@glo@etext{%
2729             #1{\glsentryfirst{\glslabel}}%
2730             {\glsentrydesc{\glslabel}}%
2731             {\glsentrysymbol{\glslabel}}{\glsinsert}}%
2732         \xmakefirstuc\@glo@etext
2733     }%
2734     {%
2735         #1{\Glsentryfirst{\glslabel}}%
2736         {\glsentrydesc{\glslabel}}%
2737         {\glsentrysymbol{\glslabel}}{\glsinsert}%
2738     }%
2739 }%
2740 }%
2741 {%

```

Make all upper case

```

2742     \ifglsused\glslabel
2743     {%

```

Subsequent use

```

2744         \mfirstucMakeUppercase{#2{\glsentrytext{\glslabel}}%
2745         {\glsentrydesc{\glslabel}}%

```

```

2746         {\glsentrysymbol{\glslabel}}{\glsinsert}}%
2747     }%
2748     {%

```

First use

```

2749         \mfirstucMakeUppercase{#1{\glsentryfirst{\glslabel}}}%
2750         {\glsentrydesc{\glslabel}}}%
2751         {\glsentrysymbol{\glslabel}}{\glsinsert}}%
2752     }%
2753 }%
2754 }%
2755 }%
2756 {%

```

Custom text provided in \glsdisp

```

2757     \ifglsused{\glslabel}%
2758     {%

```

Subsequent use

```

2759         #2{\glscustomtext}%
2760         {\glsentrydesc{\glslabel}}}%
2761         {\glsentrysymbol{\glslabel}}{}}%
2762     }%
2763     {%

```

First use

```

2764         #1{\glscustomtext}%
2765         {\glsentrydesc{\glslabel}}}%
2766         {\glsentrysymbol{\glslabel}}{}}%
2767     }%
2768 }%
2769 }

```

`\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.

```

2770 \newcommand*{\glsgenentryfmt}{%
2771     \ifdefempty\glscustomtext
2772     {%
2773         \glsifplural
2774         {%

```

Plural form

```

2775         \glsupcase
2776         {%

```

Don't adjust case

```

2777         \ifglsused\glslabel
2778         {%

```

Subsequent use

```

2779         \glsentryplural{\glslabel}\glsinsert
2780     }%
2781     {%

```


First use

```
2782      \glstentryfirstplural{\glslabel}\glsinsert
2783      }%
2784      }%
2785      {%
```

Make first letter upper case

```
2786      \ifglsused\glslabel
2787      {%
```

Subsequent use.

```
2788      \Glstentryplural{\glslabel}\glsinsert
2789      }%
2790      {%
```

First use

```
2791      \Glstentryfirstplural{\glslabel}\glsinsert
2792      }%
2793      }%
2794      {%
```

Make all upper case

```
2795      \ifglsused\glslabel
2796      {%
```

Subsequent use

```
2797      \mfirstucMakeUppercase
2798      {\glstentryplural{\glslabel}\glsinsert}%
2799      }%
2800      {%
```

First use

```
2801      \mfirstucMakeUppercase
2802      {\glstentryfirstplural{\glslabel}\glsinsert}%
2803      }%
2804      }%
2805      }%
2806      {%
```

Singular form

```
2807      \glscapscase
2808      {%
```

Don't adjust case

```
2809      \ifglsused\glslabel
2810      {%
```

Subsequent use

```
2811      \glstentrytext{\glslabel}\glsinsert
2812      }%
2813      {%
```

First use

```
2814      \glsentryfirst{\glslabel}\glsinsert
2815      }%
2816      }%
2817      {%
```

Make first letter upper case

```
2818      \ifglsused\glslabel
2819      {%
```

Subsequent use

```
2820      \Glsentrytext{\glslabel}\glsinsert
2821      }%
2822      {%
```

First use

```
2823      \Glsentryfirst{\glslabel}\glsinsert
2824      }%
2825      }%
2826      {%
```

Make all upper case

```
2827      \ifglsused\glslabel
2828      {%
```

Subsequent use

```
2829      \mfirstucMakeUppercase{\glsentrytext{\glslabel}\glsinsert}%
2830      }%
2831      {%
```

First use

```
2832      \mfirstucMakeUppercase{\glsentryfirst{\glslabel}\glsinsert}%
2833      }%
2834      }%
2835      }%
2836      }%
2837      {%
```

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

```
2838      \glscustomtext\glsinsert
2839      }%
2840 }
```

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

```
2841 \newcommand*{\glsacronymfont}{%
2842   \ifdefempty\glsacronymfont
2843   {%
2844     \ifglsused\glslabel
2845     {%
```

Subsequent use:

2846 \glsifplural
2847 {%

Subsequent plural form:

2848 \glscapscase
2849 {%

Subsequent plural form, don't adjust case:

2850 \acronymfont{\glsentryshortpl{\glslabel}}\glsinsert
2851 }%
2852 {%

Subsequent plural form, make first letter upper case:

2853 \acronymfont{\Glsentryshortpl{\glslabel}}\glsinsert
2854 }%
2855 {%

Subsequent plural form, all caps:

2856 \mfirstucMakeUppercase
2857 {\acronymfont{\glsentryshortpl{\glslabel}}\glsinsert}%
2858 }%
2859 }%
2860 {%

Subsequent singular form

2861 \glscapscase
2862 {%

Subsequent singular form, don't adjust case:

2863 \acronymfont{\glsentryshort{\glslabel}}\glsinsert
2864 }%
2865 {%

Subsequent singular form, make first letter upper case:

2866 \acronymfont{\Glsentryshort{\glslabel}}\glsinsert
2867 }%
2868 {%

Subsequent singular form, all caps:

2869 \mfirstucMakeUppercase
2870 {\acronymfont{\glsentryshort{\glslabel}}\glsinsert}%
2871 }%
2872 }%
2873 }%
2874 {%

First use:

2875 \glsifplural
2876 {%

First use plural form:

2877 \glscapscase
2878 {%

First use plural form, don't adjust case:

```
2879      \genplacrfullformat{\glslabel}{\glsinsert}%  
2880      }%  
2881      {%
```

First use plural form, make first letter upper case:

```
2882      \Genplacrfullformat{\glslabel}{\glsinsert}%  
2883      }%  
2884      {%
```

First use plural form, all caps:

```
2885      \mfirstucMakeUppercase  
2886      {\genplacrfullformat{\glslabel}{\glsinsert}}%  
2887      }%  
2888      }%  
2889      {%
```

First use singular form

```
2890      \glsapscase  
2891      {%
```

First use singular form, don't adjust case:

```
2892      \genacrfullformat{\glslabel}{\glsinsert}%  
2893      }%  
2894      {%
```

First use singular form, make first letter upper case:

```
2895      \Genacrfullformat{\glslabel}{\glsinsert}%  
2896      }%  
2897      {%
```

First use singular form, all caps:

```
2898      \mfirstucMakeUppercase  
2899      {\genacrfullformat{\glslabel}{\glsinsert}}%  
2900      }%  
2901      }%  
2902      }%  
2903      }%  
2904      {%
```

User supplied text.

```
2905      \glscustomtext  
2906      }%  
2907 }
```

`\genacrfullformat`

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

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

```
2908 \newcommand*{\genacrfullformat}[2]{%  
2909   \glsentrylong{#1}#2\space
```

```
2910 (\protect\firstacronymfont{\glsentryshort{#1}})%
2911 }
```

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

As above but makes the first letter upper case.

```
2912 \newcommand*\Genacrfullformat}[2]{%
2913   \protected@edef\gls@text{\genacrfullformat{#1}{#2}}%
2914   \xmakefirstuc\gls@text
2915 }
```

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

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

```
2916 \newcommand*\genplacrfullformat}[2]{%
2917   \glsentrylongpl{#1}#2\space
2918   (\protect\firstacronymfont{\glsentryshortpl{#1}})%
2919 }
```

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

As above but makes the first letter upper case.

```
2920 \newcommand*\Genplacrfullformat}[2]{%
2921   \protected@edef\gls@text{\genplacrfullformat{#1}{#2}}%
2922   \xmakefirstuc\gls@text
2923 }
```

`\glsdisplayfirst` Deprecated. Kept for backward compatibility.

```
2924 \newcommand*\glsdisplayfirst}[4]{#1#4}
```

`\glsdisplay` Deprecated. Kept for backward compatibility.

```
2925 \newcommand*\glsdisplay}[4]{#1#4}
```

`\defglsdisplay` Deprecated. Kept for backward compatibility.

```
2926 \newcommand*\defglsdisplay}[2][\glsdefaulttype]{%
2927   \GlossariesWarning{\string\defglsdisplay\space is now obsolete.^^J
2928   Use \string\defglsentryfmt\space instead}%
2929   \expandafter\def\csname gls@#1@display\endcsname##1##2##3##4{#2}%
2930   \edef@gls@doentrydef{%
2931     \noexpand\defglsentryfmt[#1]{%
2932       \noexpand\ifcsdef{gls@#1@displayfirst}%
2933       {%
2934         \noexpand@@gls@default@entryfmt
2935         {\noexpand\csuse{gls@#1@displayfirst}}%

```

```

2936         {\noexpand\csuse{gls@#1@display}}%
2937     }%
2938     {%
2939         \noexpand\@gls@default@entryfmt
2940         {\noexpand\glsdisplayfirst}%
2941         {\noexpand\csuse{gls@#1@display}}%
2942     }%
2943 }%
2944 }%
2945 \@gls@doentrydef
2946 }

```

`\defglsdisplayfirst` Deprecated. Kept for backward compatibility.

```

2947 \newcommand*{\defglsdisplayfirst}[2][\glsdefaultttype]{%
2948   \GlossariesWarning{\string\defglsdisplayfirst\space is now obsolete.^^J
2949   Use \string\defglsentryfmt\space instead}%
2950   \expandafter\def\csname gls@#1@displayfirst\endcsname##1##2##3##4{#2}%
2951   \edef\@gls@doentrydef{%
2952     \noexpand\defglsentryfmt[#1]{%
2953       \noexpand\ifcsdef{gls@#1@display}%
2954       {%
2955         \noexpand\@gls@default@entryfmt
2956         {\noexpand\csuse{gls@#1@displayfirst}}%
2957         {\noexpand\csuse{gls@#1@display}}%
2958       }%
2959       {%
2960         \noexpand\@gls@default@entryfmt
2961         {\noexpand\csuse{gls@#1@displayfirst}}%
2962         {\noexpand\glsdisplay}%
2963       }%
2964     }%
2965   }%
2966   \@gls@doentrydef
2967 }

```

1.11.1 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 \LaTeX 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.

```

2968 \define@key{glslink}{counter}{%
2969   \ifcsundef{c@#1}%
2970   {%
2971     \PackageError{glossaries}%
2972     {There is no counter called ‘#1’}%
2973     {%
2974       The counter key should have the name of a valid counter
2975       as its value%
2976     }%
2977   }%
2978   {%
2979     \def\@gls@counter{#1}%
2980   }%
2981 }

```

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.

```

2982 \define@key{glslink}{format}{%
2983   \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.

```

2984 \define@boolkey{glslink}{hyper}[true]{}

```

Initialise hyper key.

```

2985 \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.

```

2986 \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.

```
2987 \newcommand*{\glslinkvar}[3]{#1}
```

`\glsifhyper` Now deprecated.

```
2988 \newcommand*{\glsifhyper}[2]{%
2989 \glslinkvar{#1}{#2}{#1}%
2990 \GlossariesWarning{\string\glsifhyper\space is deprecated. Did
2991 you mean \string\glsifhyperon\space or \string\glslinkvar?}%
2992 }
```

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

```
2993 \newcommand*{\@gls@hyp@opt}[1]{%
2994 \let\glslinkvar\@firstofthree
2995 \let\@gls@hyp@opt@cs#1\relax
2996 \ifstar{\s@gls@hyp@opt}%
2997 {\@ifnextchar+{\@firstoftwo{\p@gls@hyp@opt}}{#1}}%
2998 }
```

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

```
2999 \newcommand*{\s@gls@hyp@opt}[1] []{%
3000 \let\glslinkvar\@secondofthree
3001 \@gls@hyp@opt@cs[hyper=false,#1]}
```

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

```
3002 \newcommand*{\p@gls@hyp@opt}[1] []{%
3003 \let\glslinkvar\@thirdofthree
3004 \@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`

```
3005 \newrobustcmd*{\glslink}{%
3006 \@gls@hyp@opt\@gls@@link
3007 }
```


`\@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.

```
3008 \newcommand*\@gls@link}[3] [] {%
3009   \ifglsentryexists{#2}%
3010   {%
3011     \let\do@gls@link@checkfirsthyper\relax
3012     \@gls@link[#1]{#2}{#3}%
3013   }{%
3014     \PackageError{glossaries}{Glossary entry ‘#2’ has not been
3015     defined}{You need to define a glossary entry before you
3016     can use it.}%
```

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

```
3017   \glstextformat{#3}%
3018 }%

3019 \glspostlinkhook
3020 }
```

`\glspostlinkhook`

```
3021 \newcommand*\glspostlinkhook{}
3022 %   \end{macrocode}
3023 %\end{macro}
3024 %
3025 %
3026 %\begin{macro}{\@gls@link@checkfirsthyper}
3027 % Check for first use and switch off \glskey[glslink]{hyper} key
3028 % if hyperlink not wanted. (Should be off if first use and
3029 % hyper=false is on or if first use and both the entry is in an acronym
3030 % list and the acrfootnote setting is on.)
3031 % This assumes the glossary type is stored in \cs{glstype} and the
3032 % label is stored in \cs{glslabel}.
3033 %\changes{4.08}{2014-07-30}{new}
3034 %   \begin{macrocode}
3035 \newcommand*\@gls@link@checkfirsthyper}{%
3036   \ifglsused{\glslabel}%
3037   {%
3038   }%
3039   {%
3040     \gls@checkisacronymlist\glstype
3041     \ifglshyperfirst
3042       \ifglsisacronymlist
3043         \ifglsacrfootnote
3044           \KV@glslink@hyperfalse
3045         \fi
3046       \fi
3047     \else
3048       \KV@glslink@hyperfalse
```

```

3049   \fi
3050 }%

    Allow user to hook into this
3051   \glslinkcheckfirsthyperhook
3052 }

```

checkfirsthyperhook Allow used to hook into the `\@gls@link@checkfirsthyper` macro

```

3053 \newcommand*\glslinkcheckfirsthyperhook{}

```

`\glslinkpostsetkeys`

```

3054 \newcommand*\glslinkpostsetkeys{}

```

`\glsifhyperon` Check the value of the hyper key:

```

3055 \newcommand{\glsifhyperon}[2]{\ifKV@glslink@hyper#1\else#2\fi}

```

`\@gls@link`

```

3056 \def\@gls@link[#1]#2#3{%
    Inserting \leavevmode suggested by Donald Arseneau (avoids problem with
    tabularx).
3057   \leavevmode
3058   \edef\glslabel{\glsdetoklabel{#2}}%
    Save options in \@gls@link@opts and label in \@gls@link@label
3059   \def\@gls@link@opts{#1}%
3060   \let\@gls@link@label\glslabel
3061   \def\@glsnumberformat{\glsnumberformat}%
3062   \edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%
    If this is in one of the “nohypertypes” glossaries, suppress the hyperlink by de-
    fault
3063   \edef\glstype{\csname glo@\glslabel @type\endcsname}%
    Save original setting
3064   \let\org@ifKV@glslink@hyper\ifKV@glslink@hyper
    Switch off hyper setting if the glossary type has been identified in nohyperlist.
3065   \expandafter\DTLifinlist\expandafter
3066     {\glstype}{\@gls@nohyperlist}%
3067   {%
3068     \KV@glslink@hyperfalse
3069   }%
3070   {%
3071   }%

```

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`.

```

3072   \do@gls@link@checkfirsthyper
3073   \setkeys{glslink}{#1}%

```

Add a hook for the user to customise things after the keys have been set.

```
3074 \glslinkpostsetkeys
      Store the entry's counter in \theglsentrycounter
3075 \@gls@saveentrycounter
      Define sort key if necessary:
3076 \@gls@setsort{\glslabel}%
      (De-tok'ing done by \@do@wrglossary)
3077 \@do@wrglossary{#2}%
3078 \ifKV@glslink@hyper
3079 \glslink{\glslinkprefix\glslabel}{\glstextformat{#3}}%
3080 \else
3081 \glstextformat{#3}%
3082 \fi
      Restore original setting
3083 \let\ifKV@glslink@hyper\org@ifKV@glslink@hyper
3084 }
```

`\glslinkprefix`

```
3085 \newcommand*{\glslinkprefix}{gls:}
```

`\glsentrycounter` Set default value of entry counter

```
3086 \def\glsentrycounter{\glscounter}%
```

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

```
3087 \newcommand*{\@gls@saveentrycounter}{%
3088 \def\@gls@Hcounter{}}
```

Are we using equation counter?

```
3089 \ifthenelse{\equal{\@gls@counter}{equation}}%
3090 {
```

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

```
3091 \ifcsundef{xatlevel@}%
3092 {%
3093 \edef\theglsentrycounter{\expandafter\noexpand
3094 \csname the\@gls@counter\endcsname}%
3095 }%
3096 {%
3097 \ifx\xatlevel@\@empty
3098 \edef\theglsentrycounter{\expandafter\noexpand
3099 \csname the\@gls@counter\endcsname}%
3100 \else
3101 \savecounters@
3102 \advance\c@equation by 1\relax
3103 \edef\theglsentrycounter{\csname the\@gls@counter\endcsname}%
```

Check if hyperref version of this counter

```

3104     \ifcsundef{theH\@gls@counter}%
3105     {%
3106         \def\@gls@Hcounter{\theglentrycounter}%
3107     }%
3108     {%
3109         \def\@gls@Hcounter{\csname theH\@gls@counter\endcsname}%
3110     }%
3111     \protected@edef\theglentrycounter{\@gls@Hcounter}%
3112     \restorecounters@
3113 \fi
3114 }%
3115 }%
3116 {%

```

Not using equation counter so no special measures:

```

3117     \edef\theglentrycounter{\expandafter\noexpand
3118         \csname the\@gls@counter\endcsname}%
3119 }%

```

Check if hyperref version of this counter

```

3120 \ifx\@gls@Hcounter\@empty
3121     \ifcsundef{theH\@gls@counter}%
3122     {%
3123         \def\theglentrycounter{\theglentrycounter}%
3124     }%
3125     {%
3126         \protected@edef\theglentrycounter{\expandafter\noexpand
3127             \csname theH\@gls@counter\endcsname}%
3128     }%
3129 \fi
3130 }

```

`\@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.

```

3131 \def\@set@glo@numformat#1#2#3#4{%
3132     \expandafter\@glo@check@mkidrangechar#3\@nil
3133     \protected@edef#1{%
3134         \@glo@prefix setentrycounter[#4]{#2}%
3135         \expandafter\string\csname\@glo@suffix\endcsname
3136     }%
3137     \@gls@checkmkidchars#1%
3138 }

```

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.

```

3139 \def\@glo@check@mkidxrangechar#1#2\@nil{%
3140 \if#1(\relax
3141   \def\@glo@prefix{()}%
3142   \if\relax#2\relax
3143     \def\@glo@suffix{glsnumberformat}%
3144   \else
3145     \def\@glo@suffix{#2}%
3146   \fi
3147 \else
3148   \if#1)\relax
3149     \def\@glo@prefix{}%
3150   \if\relax#2\relax
3151     \def\@glo@suffix{glsnumberformat}%
3152   \else
3153     \def\@glo@suffix{#2}%
3154   \fi
3155 \else
3156   \def\@glo@prefix{}\def\@glo@suffix{#1#2}%
3157 \fi
3158 \fi}

```

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

```

3159 \newcommand*{\@gls@escbsdq}[1]{%
3160   \def\@gls@checkedmkidx{}%
3161   \let\gls@xdystring=#1\relax
3162   \@onelevel@sanitize\gls@xdystring
3163   \edef\do@gls@xdycheckbackslash{%
3164     \noexpand\@gls@xdycheckbackslash\gls@xdystring\noexpand\@nil
3165     \@backslashchar\@backslashchar\noexpand\null}%
3166   \do@gls@xdycheckbackslash
3167   \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdystring}%
3168   \def\@gls@checkedmkidx{}%
3169   \expandafter\@gls@xdycheckquote\gls@xdystring\@nil""\null
3170   \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdystring}%

```

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

```

3171   \@for\@gls@tmp:=\gls@protected@pagefmts\do
3172   {%
3173     \edef\@gls@sanitized@tmp{\expandafter\@gobble\string\\expandonce\@gls@tmp}%
3174     \@onelevel@sanitize\@gls@sanitized@tmp
3175     \edef\gls@dostsubst{%
3176       \noexpand\DTLsubstituteall\noexpand\gls@xdystring
3177       {\@gls@sanitized@tmp}{\expandonce\@gls@tmp}%
3178     }%
3179     \gls@dostsubst

```

```

3180 }%
Assign to required control sequence
3181 \let#1=\gls@xdystring
3182 }

```

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

\gls@checkmkidxchars

```

3183 \newcommand{\@gls@checkmkidxchars}[1]{%
3184   \ifglsxindy
3185     \@gls@escbsdq{#1}%
3186   \else
3187     \def\@gls@checkedmkidx{%
3188       \expandafter\@gls@checkquote#1\@nil""\null
3189       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3190     \def\@gls@checkedmkidx{%
3191       \expandafter\@gls@checkescquote#1\@nil""\null
3192       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3193     \def\@gls@checkedmkidx{%
3194       \expandafter\@gls@checkescactual#1\@nil"??"\null
3195       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3196     \def\@gls@checkedmkidx{%
3197       \expandafter\@gls@checkactual#1\@nil??\null
3198       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3199     \def\@gls@checkedmkidx{%
3200       \expandafter\@gls@checkbar#1\@nil||\null
3201       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3202     \def\@gls@checkedmkidx{%
3203       \expandafter\@gls@checkesbar#1\@nil\|\|\null
3204       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3205     \def\@gls@checkedmkidx{%
3206       \expandafter\@gls@checklevel#1\@nil!!\null
3207       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3208     \fi
3209 }

```

Update the control sequence and strip trailing \@nil:

\@gls@updatechecked

```

3210 \def\@gls@updatechecked#1\@nil#2{\def#2{#1}}

```

\@gls@tmpb Define temporary token

```

3211 \newtoks\@gls@tmpb

```

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

```

3212 \def\@gls@checkquote#1"#2"#3\null{%
3213   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3214   \toks@={#1}%
3215   \ifx\null#2\null

```

```

3216 \ifx\null#3\null
3217 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3218 \def\@@gls@checkquote{\relax}%
3219 \else
3220 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3221 \@gls@quotechar\@gls@quotechar\@gls@quotechar\@gls@quotechar}%
3222 \def\@@gls@checkquote{\@gls@checkquote#3\null}%
3223 \fi
3224 \else
3225 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3226 \@gls@quotechar\@gls@quotechar}%
3227 \ifx\null#3\null
3228 \def\@@gls@checkquote{\@gls@checkquote#2""\null}%
3229 \else
3230 \def\@@gls@checkquote{\@gls@checkquote#2"#3\null}%
3231 \fi
3232 \fi
3233 \@@gls@checkquote
3234 }

```

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

```

3235 \def\@gls@checkescquote#1\"#2\"#3\null{%
3236 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3237 \toks@={#1}%
3238 \ifx\null#2\null
3239 \ifx\null#3\null
3240 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3241 \def\@@gls@checkescquote{\relax}%
3242 \else
3243 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3244 \@gls@quotechar\string\"@gls@quotechar
3245 \@gls@quotechar\string\"@gls@quotechar}%
3246 \def\@@gls@checkescquote{\@gls@checkescquote#3\null}%
3247 \fi
3248 \else
3249 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3250 \@gls@quotechar\string\"@gls@quotechar}%
3251 \ifx\null#3\null
3252 \def\@@gls@checkescquote{\@gls@checkescquote#2""\null}%
3253 \else
3254 \def\@@gls@checkescquote{\@gls@checkescquote#2\"#3\null}%
3255 \fi
3256 \fi
3257 \@@gls@checkescquote
3258 }

```

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

```

3259 \def\@gls@checkescactual#1\?#2\?#3\null{%
3260 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%

```

```

3261 \toks@={#1}%
3262 \ifx\null#2\null
3263   \ifx\null#3\null
3264     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3265     \def\@@gls@checkescactual{\relax}%
3266   \else
3267     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3268     \@gls@quotechar\string"\@gls@actualchar
3269     \@gls@quotechar\string"\@gls@actualchar}%
3270     \def\@@gls@checkescactual{\@gls@checkescactual#3\null}%
3271   \fi
3272 \else
3273   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3274   \@gls@quotechar\string"\@gls@actualchar}%
3275   \ifx\null#3\null
3276     \def\@@gls@checkescactual{\@gls@checkescactual#2\?\?\null}%
3277   \else
3278     \def\@@gls@checkescactual{\@gls@checkescactual#2\?#3\null}%
3279   \fi
3280 \fi
3281 \@@gls@checkescactual
3282 }

```

\@gls@checkescbar Similarly for \||:

```

3283 \def\@gls@checkescbar#1\|#2\|#3\null{%
3284   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3285   \toks@={#1}%
3286   \ifx\null#2\null
3287     \ifx\null#3\null
3288       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3289       \def\@@gls@checkescbar{\relax}%
3290     \else
3291       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3292       \@gls@quotechar\string"\@gls@encapchar
3293       \@gls@quotechar\string"\@gls@encapchar}%
3294       \def\@@gls@checkescbar{\@gls@checkescbar#3\null}%
3295     \fi
3296   \else
3297     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3298     \@gls@quotechar\string"\@gls@encapchar}%
3299     \ifx\null#3\null
3300       \def\@@gls@checkescbar{\@gls@checkescbar#2\|\|\null}%
3301     \else
3302       \def\@@gls@checkescbar{\@gls@checkescbar#2\|#3\null}%
3303     \fi
3304   \fi
3305 \@@gls@checkescbar
3306 }

```


\@gls@checkesclevel Similarly for \!:

```
3307 \def\@gls@checkesclevel#1\!#2\!#3\null{%
3308   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3309   \toks@={#1}%
3310   \ifx\null#2\null
3311     \ifx\null#3\null
3312       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3313       \def\@@gls@checkesclevel{\relax}%
3314     \else
3315       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3316         \@gls@quotechar\string"\@gls@levelchar
3317         \@gls@quotechar\string"\@gls@levelchar}%
3318       \def\@@gls@checkesclevel{\@gls@checkesclevel#3\null}%
3319     \fi
3320   \else
3321     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3322       \@gls@quotechar\string"\@gls@levelchar}%
3323     \ifx\null#3\null
3324       \def\@@gls@checkesclevel{\@gls@checkesclevel#2\!\!\null}%
3325     \else
3326       \def\@@gls@checkesclevel{\@gls@checkesclevel#2\!#3\null}%
3327     \fi
3328   \fi
3329 \@@gls@checkesclevel
3330 }
```

\@gls@checkbar and for |:

```
3331 \def\@gls@checkbar#1|#2|#3\null{%
3332   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3333   \toks@={#1}%
3334   \ifx\null#2\null
3335     \ifx\null#3\null
3336       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3337       \def\@@gls@checkbar{\relax}%
3338     \else
3339       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3340         \@gls@quotechar\@gls@encapchar\@gls@quotechar\@gls@encapchar}%
3341       \def\@@gls@checkbar{\@gls@checkbar#3\null}%
3342     \fi
3343   \else
3344     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3345       \@gls@quotechar\@gls@encapchar}%
3346     \ifx\null#3\null
3347       \def\@@gls@checkbar{\@gls@checkbar#2||\null}%
3348     \else
3349       \def\@@gls@checkbar{\@gls@checkbar#2|#3\null}%
3350     \fi
3351   \fi
3352 \@@gls@checkbar
```

3353 }

\@gls@checklevel and for !:

```
3354 \def\@gls@checklevel#1!#2!#3\null{%
3355   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3356   \toks@={#1}%
3357   \ifx\null#2\null
3358     \ifx\null#3\null
3359       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3360       \def\@gls@checklevel{\relax}%
3361     \else
3362       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3363         \@gls@quotechar\@gls@levelchar\@gls@quotechar\@gls@levelchar}%
3364       \def\@gls@checklevel{\@gls@checklevel#3\null}%
3365     \fi
3366   \else
3367     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3368       \@gls@quotechar\@gls@levelchar}%
3369     \ifx\null#3\null
3370       \def\@gls@checklevel{\@gls@checklevel#2!\null}%
3371     \else
3372       \def\@gls@checklevel{\@gls@checklevel#2!#3\null}%
3373     \fi
3374   \fi
3375   \@gls@checklevel
3376 }
```

\@gls@checkactual and for ?:

```
3377 \def\@gls@checkactual#1?#2?#3\null{%
3378   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3379   \toks@={#1}%
3380   \ifx\null#2\null
3381     \ifx\null#3\null
3382       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3383       \def\@gls@checkactual{\relax}%
3384     \else
3385       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3386         \@gls@quotechar\@gls@actualchar\@gls@quotechar\@gls@actualchar}%
3387       \def\@gls@checkactual{\@gls@checkactual#3\null}%
3388     \fi
3389   \else
3390     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3391       \@gls@quotechar\@gls@actualchar}%
3392     \ifx\null#3\null
3393       \def\@gls@checkactual{\@gls@checkactual#2??\null}%
3394     \else
3395       \def\@gls@checkactual{\@gls@checkactual#2?#3\null}%
3396     \fi
3397   \fi
```

```

3398 \@gls@checkactual
3399 }

```

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

```

3400 \def\@gls@xdycheckquote#1"#2"#3\null{%
3401 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3402 \toks@={#1}%
3403 \ifx\null#2\null
3404 \ifx\null#3\null
3405 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3406 \def\@gls@xdycheckquote{\relax}%
3407 \else
3408 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3409 \string\}\string\}%
3410 \def\@gls@xdycheckquote{\@gls@xdycheckquote#3\null}%
3411 \fi
3412 \else
3413 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3414 \string\}%
3415 \ifx\null#3\null
3416 \def\@gls@xdycheckquote{\@gls@xdycheckquote#2""\null}%
3417 \else
3418 \def\@gls@xdycheckquote{\@gls@xdycheckquote#2"#3\null}%
3419 \fi
3420 \fi
3421 \@gls@xdycheckquote
3422 }

```

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

```

\@gls@xdycheckbackslash
3423 \edef\def\@gls@xdycheckbackslash{%
3424 \noexpand\def\noexpand\@gls@xdycheckbackslash##1\@backslashchar
3425 ##2\@backslashchar##3\noexpand\null{%
3426 \noexpand\@gls@tmpb=\noexpand\expandafter
3427 {\noexpand\@gls@checkedmkidx}%
3428 \noexpand\toks@={##1}%
3429 \noexpand\ifx\noexpand\null##2\noexpand\null
3430 \noexpand\ifx\noexpand\null##3\noexpand\null
3431 \noexpand\edef\noexpand\@gls@checkedmkidx{%
3432 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
3433 \noexpand\def\noexpand\@gls@xdycheckbackslash{\relax}%
3434 \noexpand\else
3435 \noexpand\edef\noexpand\@gls@checkedmkidx{%
3436 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
3437 \@backslashchar\@backslashchar\@backslashchar\@backslashchar}%
3438 \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3439 \noexpand\@gls@xdycheckbackslash##3\noexpand\null}%
3440 \noexpand\fi
3441 \noexpand\else

```

```

3442 \noexpand\edef\noexpand\@gls@checkedmkidx{%
3443 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
3444 \@backslashchar\@backslashchar}%
3445 \noexpand\ifx\noexpand\null##3\noexpand\null
3446 \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3447 \noexpand\@gls@xdycheckbackslash##2\@backslashchar
3448 \@backslashchar\noexpand\null}%
3449 \noexpand\else
3450 \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3451 \noexpand\@gls@xdycheckbackslash##2\@backslashchar
3452 ##3\noexpand\null}%
3453 \noexpand\fi
3454 \noexpand\fi
3455 \noexpand\@gls@xdycheckbackslash
3456 }%
3457 }

Now go ahead and define \@gls@xdycheckbackslash
3458 \def@gls@xdycheckbackslash

```

\glsdohypertarget

```

3459 \newlength@gls@tmplen
3460 \newcommand*{\glsdohypertarget}[2]{%
3461 \settoheight{\gls@tmplen}{#2}%
3462 \raisebox{\gls@tmplen}{\hypertarget{#1}}{#2}%
3463 }

```

\glsdohyperlink

```

3464 \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.

```

3465 \ifcsundef{hyperlink}%
3466 {%
3467 \let\@glslink\@secondoftwo
3468 }%
3469 {%
3470 \let\@glslink\glsdohyperlink
3471 }

```

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

```

3472 \ifcsundef{hypertarget}%
3473 {%
3474 \let\@glstarget\@secondoftwo
3475 }%
3476 {%
3477 \let\@glstarget\glsdohypertarget
3478 }

```

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

`\glsdisablehyper`

```
3479 \newcommand{\glsdisablehyper}{%
3480   \KV@glslink@hyperfalse
3481   \let\@glslink\@secondoftwo
3482   \let\@glstarget\@secondoftwo
3483 }
```

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

`\glsenablehyper`

```
3484 \newcommand{\glsenablehyper}{%
3485   \KV@glslink@hypertrue
3486   \let\@glslink\glsdohyperlink
3487   \let\@glstarget\glsdohypertarget
3488 }
```

Provide some convenience commands if not already defined:

```
3489 \providecommand{\@firstofthree}[3]{#1}
3490 \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`

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

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

`\@gls`

```
3492 \newcommand*{\@gls}[2] [] {%
3493   \new@ifnextchar[{\@gls@{#1}{#2}}{\@gls@{#1}{#2} [] }%
3494 }
```

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

```
3495 \def\@gls@#1#2[#3]{%
3496   \glsdoifexists{#2}%
3497   {%
3498     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3499     \let\glsifplural\@secondoftwo
3500     \let\glscapscase\@firstofthree
3501     \let\glscustomtext\@empty
3502     \def\glsinsert{#3}%
```

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

```
3503   \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.

```
3504   \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3505   \ifKV@gls@link@local
3506     \glslocalunset{#2}%
3507   \else
3508     \glsunset{#2}%
3509   \fi
3510 }%
```

```
3511 \glspostlinkhook
3512 }
```

`\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`

```
3513 \newrobustcmd*{\Gls}{\@gls@hyp@opt\@Gls}
```

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

```
3514 \newcommand*{\@Gls}[2][ ]{%
3515   \new@ifnextchar[{\@Gls@{#1}{#2}}{\@Gls@{#1}{#2}[ ]}%
3516 }
```

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

```
3517 \def\@Gls@#1#2[#3]{%
3518   \glsdoifexists{#2}%
3519   {%
3520     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
```

```

3521 \let\glsifplural\@secondoftwo
3522 \let\glsifcaps\@secondofthree
3523 \let\glsifcustom\@empty
3524 \def\glsinsert{#3}%

```

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

```

3525 \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.

```

3526 \@gls@link[#1]{#2}{\@glo@text}%

```

Indicate that this entry has now been used

```

3527 \ifKV@gls@link@local
3528 \glslocalunset{#2}%
3529 \else
3530 \glsunset{#2}%
3531 \fi
3532 }%

```

```

3533 \glspostlinkhook
3534 }

```

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

`\GLS`

```

3535 \newrobustcmd*{\GLS}{\@gls@hyp@opt\@GLS}

```

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

```

3536 \newcommand*{\@GLS}[2][{}]{%
3537 \new@ifnextchar[{\@GLS@{#1}{#2}}{\@GLS@{#1}{#2}[]}%
3538 }

```

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

```

3539 \def\@GLS@#1#2[#3]{%
3540 \glsdoifexists{#2}%
3541 {%
3542 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3543 \let\glsifplural\@secondoftwo
3544 \let\glsifcaps\@thirdofthree
3545 \let\glsifcustom\@empty
3546 \def\glsinsert{#3}%

```

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

```

3547 \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%

```

Call `\@gls@link` If footnote package option has been used and the glossary type is `\acronymstype`, suppress hyperlink for first use. Likewise if the `hyperfirst=false` package option is used.

```
3548 \gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3549 \ifKV@glslink@local
3550 \glslocalunset{#2}%
3551 \else
3552 \glsunset{#2}%
3553 \fi
3554 }%
```

```
3555 \glspostlinkhook
3556 }
```

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

`\glspl`

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

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

```
3558 \newcommand*{\@glspl}[2][ ]{%
3559 \new@ifnextchar[{\@glspl@{#1}{#2}}{\@glspl@{#1}{#2}[]}%
3560 }
```

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

```
3561 \def\@glspl@#1#2[#3]{%
3562 \glsdoifexists{#2}%
3563 {%
3564 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
3565 \let\glsifplural\@firstoftwo
3566 \let\glsifcaps\@firstofthree
3567 \let\glsifcustomtext\@empty
3568 \def\glsinsert{#3}%
```

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

```
3569 \def\@glo@text{\csname gls@\glsstyp @entryfmt\endcsname}%
```

Call `\@gls@link`. If footnote package option has been used and the glossary type is `\acronymstype`, suppress hyperlink for first use. Likewise if the `hyperfirst=false` package option is used.

```
3570 \gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3571 \ifKV@glslink@local
3572 \glslocalunset{#2}%
```



```

3573   \else
3574     \glsunset{#2}%
3575   \fi
3576 }%

3577   \glspostlinkhook
3578 }

```

`\Glspl` behaves in the same way as `\glspl`, except that the first letter of the link text is converted to uppercase (as with `\Gls`, if the first letter has an accent, it will need to be grouped).

`\Glspl`

```

3579 \newrobustcmd*{\Glspl}{\@gls@hyp@opt\@Glspl}

```

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

```

3580 \newcommand*{\@Glspl}[2][ ]{%
3581   \new@ifnextchar[{\@Glspl@{#1}{#2}}{\@Glspl@{#1}{#2}[]}%
3582 }

```

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

```

3583 \def\@Glspl@#1#2[#3]{%
3584   \glsdoifexists{#2}%
3585   {%
3586     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3587     \let\glsifplural\@firstoftwo
3588     \let\gls caps case\@secondofthree
3589     \let\gls custom text\@empty
3590     \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`.

```

3591   \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.

```

3592   \@gls@link[#1]{#2}{\@glo@text}%

```

Indicate that this entry has now been used

```

3593   \ifKV@gls@link@local
3594     \glslocalunset{#2}%
3595   \else
3596     \glsunset{#2}%
3597   \fi
3598 }%

```

```
3599 \glspostlinkhook
3600 }
```

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

\GLSp1

```
3601 \newrobustcmd*{\GLSp1}{\@gls@hyp@opt\@GLSp1}
```

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

```
3602 \newcommand*{\@GLSp1}[2][ ]{ }%
3603 \new@ifnextchar[{\@GLSp1@{#1}{#2}}{\@GLSp1@{#1}{#2}[ ]}%
3604 }
```

\@GLSp1 Read in the final optional argument:

```
3605 \def\@GLSp1@#1#2[#3]{%
3606 \glsdoifexists{#2}%
3607 {%
3608 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
3609 \let\glsifplural\@firstoftwo
3610 \let\glsifscaps\@thirdofthree
3611 \let\glsifscapscase\@thirdofthree
3612 \let\glsifscapscase\@thirdofthree
3611 \let\glsifscapscase\@thirdofthree
3612 \def\glsifscapscase\@thirdofthree
3612 \def\glsinsert{#3}%
```

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

```
3613 \def\@glo@text{\csname gls@\gls@type @entryfmt\endcsname}%
```

Call \@gls@link. If footnote package option has been used and the glossary type is \acronymtype, suppress hyperlink for first use. Likewise if the hyperfirst=false package option is used.

```
3614 \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3615 \ifKV@gls@link@local
3616 \glslocalunset{#2}%
3617 \else
3618 \glsunset{#2}%
3619 \fi
3620 }%
```

```
3621 \glspostlinkhook
3622 }
```

\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:

```
3623 \newrobustcmd*{\glsdisp}{\@gls@hyp@opt\@glsdisp}
```

Defined the un-starred form.

`\@glsdisp`

```
3624 \newcommand*{\@glsdisp}[3] [] {%
3625   \glsdoifexists{#2}{%

3626     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3627     \let\glsifplural\@secondoftwo
3628     \let\gls caps case\@firstofthree
3629     \def\gls custom text{#3}%
3630     \def\gls insert{}}%
```

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

```
3631   \def\@glo@text{\csname gls@\gls type @entryfmt\endcsname}%
```

Call `\@gls@link`. If footnote package option has been used and the glossary type is `\acronym type`, suppress hyperlink for first use. Likewise if the `hyperfirst=false` package option is used.

```
3632   \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3633   \ifKV@gls link@local
3634     \gls local unset{#2}%
3635   \else
3636     \gls unset{#2}%
3637   \fi
3638 }%

3639 \gls post link hook
3640 }
```

`\@gls@field@link`

```
3641 \newcommand{\@gls@field@link}[3] {%
3642   \glsdoifexists{#2}%
3643   {%
3644     \let\do@gls@link@checkfirsthyper\relax
3645     \@gls@link[#1]{#2}{#3}%
3646   }%

3647   \gls post link hook
3648 }
```

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

`\gls text`

```
3649 \newrobustcmd*{\gls text}{\@gls@hyp@opt\@gls text}
```

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

```
3650 \newcommand*{\@glstext}[2] [] {%
3651   \new@ifnextchar[{\@glstext@{#1}{#2}}{\@glstext@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3652 \def\@glstext@#1#2[#3] {%
3653   \@gls@field@link{#1}{#2}{\glsentrytext{#2}#3}%
3654 }
```

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

`\GLStext`

```
3655 \newrobustcmd*{\GLStext}{\@gls@hyp@opt\@GLStext}
```

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

```
3656 \newcommand*{\@GLStext}[2] [] {%
3657   \new@ifnextchar[{\@GLStext@{#1}{#2}}{\@GLStext@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3658 \def\@GLStext@#1#2[#3] {%
3659   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrytext{#2}#3}}%
3660 }
```

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

`\Glstext`

```
3661 \newrobustcmd*{\Glstext}{\@gls@hyp@opt\@Glstext}
```

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

```
3662 \newcommand*{\@Glstext}[2] [] {%
3663   \new@ifnextchar[{\@Glstext@{#1}{#2}}{\@Glstext@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3664 \def\@Glstext@#1#2[#3] {%
3665   \@gls@field@link{#1}{#2}{\Glsentrytext{#2}#3}%
3666 }
```

`\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`

```
3667 \newrobustcmd*{\glsfirst}{\@gls@hyp@opt\@glsfirst}
```

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

```
3668 \newcommand*{\@glsfirst}[2] [] {%
3669   \new@ifnextchar[{\@glsfirst@{#1}{#2}}{\@glsfirst@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3670 \def\@glsfirst@#1#2[#3]{%
3671   \@gls@field@link{#1}{#2}{\glsentryfirst{#2}#3}%
3672 }
```

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

`\Glsfirst`

```
3673 \newrobustcmd*{\Glsfirst}{\@gls@hyp@opt\@Glsfirst}
```

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

```
3674 \newcommand*{\@Glsfirst}[2] [] {%
3675   \new@ifnextchar[{\@Glsfirst@{#1}{#2}}{\@Glsfirst@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3676 \def\@Glsfirst@#1#2[#3]{%
3677   \@gls@field@link{#1}{#2}{\Glsentryfirst{#2}#3}%
3678 }
```

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

`\GLSfirst`

```
3679 \newrobustcmd*{\GLSfirst}{\@gls@hyp@opt\@GLSfirst}
```

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

```
3680 \newcommand*{\@GLSfirst}[2] [] {%
3681   \new@ifnextchar[{\@GLSfirst@{#1}{#2}}{\@GLSfirst@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3682 \def\@GLSfirst@#1#2[#3]{%
3683   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryfirst{#2}#3}}%
3684 }
```

`\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`

```
3685 \newrobustcmd*{\glsplural}{\@gls@hyp@opt\@glsplural}
```

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

```
3686 \newcommand*{\@glsplural}[2] [] {%
3687   \new@ifnextchar[{\@glsplural@{#1}{#2}}{\@glsplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3688 \def\@glsplural@#1#2[#3]{%
3689   \@gls@field@link{#1}{#2}{\glsentryplural{#2}#3}%
3690 }
```

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

`\Glsplural`

```
3691 \newrobustcmd*{\Glsplural}{\@gls@hyp@opt\@Glsplural}
```

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

```
3692 \newcommand*{\@Glsplural}[2] [] {%
```

```
3693   \new@ifnextchar[{\@Glsplural@{#1}{#2}}{\@Glsplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3694 \def\@Glsplural@#1#2[#3] {%
```

```
3695   \@gls@field@link{#1}{#2}{\Glsentryplural{#2}#3}%
```

```
3696 }
```

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

`\Glsplural`

```
3697 \newrobustcmd*{\Glsplural}{\@gls@hyp@opt\@Glsplural}
```

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

```
3698 \newcommand*{\@Glsplural}[2] [] {%
```

```
3699   \new@ifnextchar[{\@Glsplural@{#1}{#2}}{\@Glsplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3700 \def\@Glsplural@#1#2[#3] {%
```

```
3701   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentryplural{#2}#3}}%
```

```
3702 }
```

`\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`

```
3703 \newrobustcmd*{\glsfirstplural}{\@gls@hyp@opt\@glsfirstplural}
```

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

```
3704 \newcommand*{\@glsfirstplural}[2] [] {%
```

```
3705   \new@ifnextchar[{\@glsfirstplural@{#1}{#2}}{\@glsfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3706 \def\@glsfirstplural@#1#2[#3] {%
```

```
3707   \@gls@field@link{#1}{#2}{\Glsentryfirstplural{#2}#3}%
```

```
3708 }
```

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

`\Glsfirstplural`

```
3709 \newrobustcmd*{\Glsfirstplural}{\@gls@hyp@opt\@Glsfirstplural}
```

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

```
3710 \newcommand*{\@Glsfirstplural}[2] [] {%
3711   \new@ifnextchar[{\@Glsfirstplural@{#1}{#2}}{\@Glsfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3712 \def\@Glsfirstplural@#1#2[#3] {%
3713   \@gls@field@link{#1}{#2}{\Glsentryfirstplural{#2}#3}%
3714 }
```

`\GLSfirstplural` behaves like `\glsfirstplural` except that the link text is converted to uppercase.

`\GLSfirstplural`

```
3715 \newrobustcmd*{\GLSfirstplural}{\@gls@hyp@opt\@GLSfirstplural}
```

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

```
3716 \newcommand*{\@GLSfirstplural}[2] [] {%
3717   \new@ifnextchar[{\@GLSfirstplural@{#1}{#2}}{\@GLSfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3718 \def\@GLSfirstplural@#1#2[#3] {%
3719   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryfirstplural{#2}#3}}%
3720 }
```

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

`\glsname`

```
3721 \newrobustcmd*{\glsname}{\@gls@hyp@opt\@glsname}
```

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

```
3722 \newcommand*{\@glsname}[2] [] {%
3723   \new@ifnextchar[{\@glsname@{#1}{#2}}{\@glsname@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3724 \def\@glsname@#1#2[#3] {%
3725   \@gls@field@link{#1}{#2}{\glsentryname{#2}#3}%
3726 }
```

`\Glsname` behaves like `\glsname` except that the first letter is converted to uppercase.

`\Glsname`

```
3727 \newrobustcmd*{\Glsname}{\@gls@hyp@opt\@Glsname}
```

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

```
3728 \newcommand*{\@Glsname}[2] [] {%
3729   \new@ifnextchar[{\@Glsname@{#1}{#2}}{\@Glsname@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3730 \def\@Glsname@#1#2[#3]{%
3731   \@gls@field@link{#1}{#2}{\Glsentryname{#2}#3}%
3732 }
```

`\Glsname` behaves like `\glsname` except that the link text is converted to uppercase.

`\Glsname`

```
3733 \newrobustcmd*{\Glsname}{\@gls@hyp@opt\@Glsname}
```

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

```
3734 \newcommand*{\@Glsname}[2][ ]{%
3735   \new@ifnextchar[{\@Glsname@{#1}{#2}}{\@Glsname@{#1}{#2}[ ]}]}
```

Read in the final optional argument:

```
3736 \def\@Glsname@#1#2[#3]{%
3737   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryname{#2}#3}}%
3738 }
```

`\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`

```
3739 \newrobustcmd*{\glsdesc}{\@gls@hyp@opt\@glsdesc}
```

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

```
3740 \newcommand*{\@glsdesc}[2][ ]{%
3741   \new@ifnextchar[{\@glsdesc@{#1}{#2}}{\@glsdesc@{#1}{#2}[ ]}]}
```

Read in the final optional argument:

```
3742 \def\@glsdesc@#1#2[#3]{%
3743   \@gls@field@link{#1}{#2}{\glsentrydesc{#2}#3}%
3744 }
```

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

`\Glsdesc`

```
3745 \newrobustcmd*{\Glsdesc}{\@gls@hyp@opt\@Glsdesc}
```

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

```
3746 \newcommand*{\@Glsdesc}[2][ ]{%
3747   \new@ifnextchar[{\@Glsdesc@{#1}{#2}}{\@Glsdesc@{#1}{#2}[ ]}]}
```

Read in the final optional argument:

```
3748 \def\@Glsdesc@#1#2[#3]{%
3749   \@gls@field@link{#1}{#2}{\Glsentrydesc{#2}#3}%
3750 }
```


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

`\GLSdesc`

```
3751 \newrobustcmd*{\GLSdesc}{\@gls@hyp@opt\@GLSdesc}
```

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

```
3752 \newcommand*{\@GLSdesc}[2] [] {%
```

```
3753   \new@ifnextchar[{\@GLSdesc@{#1}{#2}}{\@GLSdesc@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3754 \def\@GLSdesc@#1#2[#3] {%
```

```
3755   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrydesc{#2}#3}}%
```

```
3756 }
```

`\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`

```
3757 \newrobustcmd*{\glsdescplural}{\@gls@hyp@opt\@glsdescplural}
```

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

```
3758 \newcommand*{\@glsdescplural}[2] [] {%
```

```
3759   \new@ifnextchar[{\@glsdescplural@{#1}{#2}}{\@glsdescplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3760 \def\@glsdescplural@#1#2[#3] {%
```

```
3761   \@gls@field@link{#1}{#2}{\glsentrydescplural{#2}#3}}%
```

```
3762 }
```

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

`\Glsdescplural`

```
3763 \newrobustcmd*{\Glsdescplural}{\@gls@hyp@opt\@Glsdescplural}
```

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

```
3764 \newcommand*{\@Glsdescplural}[2] [] {%
```

```
3765   \new@ifnextchar[{\@Glsdescplural@{#1}{#2}}{\@Glsdescplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3766 \def\@Glsdescplural@#1#2[#3] {%
```

```
3767   \@gls@field@link{#1}{#2}{\Glsentrydescplural{#2}#3}}%
```

```
3768 }
```

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

`\GLSdescplural`

```
3769 \newrobustcmd*{\GLSdescplural}{\@gls@hyp@opt\@GLSdescplural}
```

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

```
3770 \newcommand*{\@GLSdescplural}[2] [] {%
3771   \new@ifnextchar[{\@GLSdescplural@{#1}{#2}}{\@GLSdescplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3772 \def\@GLSdescplural@#1#2[#3] {%
3773   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrydescplural{#2}#3}}%
3774 }
```

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

`\glsymbol`

```
3775 \newrobustcmd*{\glsymbol}{\@gls@hyp@opt\@glsymbol}
```

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

```
3776 \newcommand*{\@glsymbol}[2] [] {%
3777   \new@ifnextchar[{\@glsymbol@{#1}{#2}}{\@glsymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3778 \def\@glsymbol@#1#2[#3] {%
3779   \@gls@field@link{#1}{#2}{\glsentrysymbol{#2}#3}}%
3780 }
```

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

`\Glsymbol`

```
3781 \newrobustcmd*{\Glsymbol}{\@gls@hyp@opt\@Glsymbol}
```

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

```
3782 \newcommand*{\@Glsymbol}[2] [] {%
3783   \new@ifnextchar[{\@Glsymbol@{#1}{#2}}{\@Glsymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3784 \def\@Glsymbol@#1#2[#3] {%
3785   \@gls@field@link{#1}{#2}{\Glsentrysymbol{#2}#3}}%
3786 }
```

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

`\GLSsymbol`

```
3787 \newrobustcmd*{\GLSsymbol}{\@gls@hyp@opt\@GLSsymbol}
```

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

```
3788 \newcommand*{\@GLSsymbol}[2] [] {%
3789   \new@ifnextchar[{\@GLSsymbol@{#1}{#2}}{\@GLSsymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3790 \def\@GLSsymbol@#1#2[#3]{%
3791   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrysymbol{#2}#3}}%
3792 }
```

`\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`

```
3793 \newrobustcmd*{\glssymbolplural}{\@gls@hyp@opt\@glssymbolplural}
```

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

```
3794 \newcommand*{\@glssymbolplural}[2] []{%
3795   \new@ifnextchar[{\@glssymbolplural@{#1}{#2}}{\@glssymbolplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3796 \def\@glssymbolplural@#1#2[#3]{%
3797   \@gls@field@link{#1}{#2}{\glsentrysymbolplural{#2}#3}%
3798 }
```

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

`\Glsymbolplural`

```
3799 \newrobustcmd*{\Glsymbolplural}{\@gls@hyp@opt\@Glsymbolplural}
```

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

```
3800 \newcommand*{\@Glsymbolplural}[2] []{%
3801   \new@ifnextchar[{\@Glsymbolplural@{#1}{#2}}{\@Glsymbolplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3802 \def\@Glsymbolplural@#1#2[#3]{%
3803   \@gls@field@link{#1}{#2}{\Glsentrysymbolplural{#2}#3}%
3804 }
```

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

`\GLSsymbolplural`

```
3805 \newrobustcmd*{\GLSsymbolplural}{\@gls@hyp@opt\@GLSsymbolplural}
```

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

```
3806 \newcommand*{\@GLSsymbolplural}[2] []{%
3807   \new@ifnextchar[{\@GLSsymbolplural@{#1}{#2}}{\@GLSsymbolplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3808 \def\@GLSsymbolplural@#1#2[#3]{%
3809   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrysymbolplural{#2}#3}}%
3810 }
```

`\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`

```
3811 \newrobustcmd*{\glsuseri}{\@gls@hyp@opt\@glsuseri}
```

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

```
3812 \newcommand*{\@glsuseri}[2] [] {%
```

```
3813   \new@ifnextchar[{\@glsuseri@{#1}{#2}}{\@glsuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3814 \def\@glsuseri@#1#2[#3] {%
```

```
3815   \@gls@field@link{#1}{#2}{\glsentryuseri{#2}#3}%
```

```
3816 }
```

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

`\Glsuseri`

```
3817 \newrobustcmd*{\Glsuseri}{\@gls@hyp@opt\@Glsuseri}
```

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

```
3818 \newcommand*{\@Glsuseri}[2] [] {%
```

```
3819   \new@ifnextchar[{\@Glsuseri@{#1}{#2}}{\@Glsuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3820 \def\@Glsuseri@#1#2[#3] {%
```

```
3821   \@gls@field@link{#1}{#2}{\Glsentryuseri{#2}#3}%
```

```
3822 }
```

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

`\GLSuseri`

```
3823 \newrobustcmd*{\GLSuseri}{\@gls@hyp@opt\@GLSuseri}
```

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

```
3824 \newcommand*{\@GLSuseri}[2] [] {%
```

```
3825   \new@ifnextchar[{\@GLSuseri@{#1}{#2}}{\@GLSuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3826 \def\@GLSuseri@#1#2[#3] {%
```

```
3827   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseri{#2}#3}}%
```

```
3828 }
```

`\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`

```
3829 \newrobustcmd*{\glsuserii}{\@gls@hyp@opt\@glsuserii}
```

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

```
3830 \newcommand*{\@glsuserii}[2] [] {%
3831   \new@ifnextchar[{\@glsuserii@{#1}{#2}}{\@glsuserii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3832 \def\@glsuserii@#1#2[#3] {%
3833   \@gls@field@link{#1}{#2}{\glsentryuserii{#2}#3}%
3834 }
```

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

\Glsuserii

```
3835 \newrobustcmd*{\Glsuserii}{\@gls@hyp@opt\@Glsuserii}
```

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

```
3836 \newcommand*{\@Glsuserii}[2] [] {%
3837   \new@ifnextchar[{\@Glsuserii@{#1}{#2}}{\@Glsuserii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3838 \def\@Glsuserii@#1#2[#3] {%
3839   \@gls@field@link{#1}{#2}{\Glsentryuserii{#2}#3}%
3840 }
```

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

\GLSuserii

```
3841 \newrobustcmd*{\GLSuserii}{\@gls@hyp@opt\@GLSuserii}
```

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

```
3842 \newcommand*{\@GLSuserii}[2] [] {%
3843   \new@ifnextchar[{\@GLSuserii@{#1}{#2}}{\@GLSuserii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3844 \def\@GLSuserii@#1#2[#3] {%
3845   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuserii{#2}#3}}%
3846 }
```

\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

```
3847 \newrobustcmd*{\glsuseriii}{\@gls@hyp@opt\@glsuseriii}
```

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

```
3848 \newcommand*{\@glsuseriii}[2] [] {%
3849   \new@ifnextchar[{\@glsuseriii@{#1}{#2}}{\@glsuseriii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3850 \def\@glsuseriii@#1#2[#3]{%
3851   \@gls@field@link{#1}{#2}{\glstentryuseriii{#2}#3}%
3852 }
```

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

`\Glsuseriii`

```
3853 \newrobustcmd*{\Glsuseriii}{\@gls@hyp@opt\@Glsuseriii}
```

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

```
3854 \newcommand*{\@Glsuseriii}[2] [] {%
3855   \new@ifnextchar[{\@Glsuseriii@{#1}{#2}}{\@Glsuseriii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3856 \def\@Glsuseriii@#1#2[#3]{%
3857   \@gls@field@link{#1}{#2}{\glstentryuseriii{#2}#3}%
3858 }
```

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

`\GLSuseriii`

```
3859 \newrobustcmd*{\GLSuseriii}{\@gls@hyp@opt\@GLSuseriii}
```

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

```
3860 \newcommand*{\@GLSuseriii}[2] [] {%
3861   \new@ifnextchar[{\@GLSuseriii@{#1}{#2}}{\@GLSuseriii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3862 \def\@GLSuseriii@#1#2[#3]{%
3863   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glstentryuseriii{#2}#3}%
3864 }
```

`\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`

```
3865 \newrobustcmd*{\glsuseriv}{\@gls@hyp@opt\@glsuseriv}
```

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

```
3866 \newcommand*{\@glsuseriv}[2] [] {%
3867   \new@ifnextchar[{\@glsuseriv@{#1}{#2}}{\@glsuseriv@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3868 \def\@glsuseriv@#1#2[#3]{%
3869   \@gls@field@link{#1}{#2}{\glstentryuseriv{#2}#3}%
3870 }
```

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

`\Glsuseriv`

```
3871 \newrobustcmd*{\Glsuseriv}{\@gls@hyp@opt\@Glsuseriv}
```

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

```
3872 \newcommand*{\@Glsuseriv}[2] [] {%
```

```
3873   \new@ifnextchar[{\@Glsuseriv@{#1}{#2}}{\@Glsuseriv@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3874 \def\@Glsuseriv@#1#2[#3] {%
```

```
3875   \@gls@field@link{#1}{#2}{\Glsentryuseriv{#2}#3}%
```

```
3876 }
```

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

`\GLSuseriv`

```
3877 \newrobustcmd*{\GLSuseriv}{\@gls@hyp@opt\@GLSuseriv}
```

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

```
3878 \newcommand*{\@GLSuseriv}[2] [] {%
```

```
3879   \new@ifnextchar[{\@GLSuseriv@{#1}{#2}}{\@GLSuseriv@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3880 \def\@GLSuseriv@#1#2[#3] {%
```

```
3881   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseriv{#2}#3}}%
```

```
3882 }
```

`\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`

```
3883 \newrobustcmd*{\glsuserv}{\@gls@hyp@opt\@glsuserv}
```

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

```
3884 \newcommand*{\@glsuserv}[2] [] {%
```

```
3885   \new@ifnextchar[{\@glsuserv@{#1}{#2}}{\@glsuserv@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3886 \def\@glsuserv@#1#2[#3] {%
```

```
3887   \@gls@field@link{#1}{#2}{\glsentryuserv{#2}#3}%
```

```
3888 }
```

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

`\Glsuserv`

```
3889 \newrobustcmd*{\Glsuserv}{\@gls@hyp@opt\@Glsuserv}
```

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

```
3890 \newcommand*{\@Glsuserv}[2] [] {%
3891 \new@ifnextchar [{\@Glsuserv@{#1}{#2}}{\@Glsuserv@{#1}{#2} []}] }
```

Read in the final optional argument:

```
3892 \def\@Glsuserv@#1#2[#3] {%
3893 \@gls@field@link{#1}{#2}{\Glsentryuserv{#2}#3}%
3894 }
```

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

\Glsuserv

```
3895 \newrobustcmd*{\Glsuserv}{\@gls@hyp@opt\@Glsuserv}
```

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

```
3896 \newcommand*{\@Glsuserv}[2] [] {%
3897 \new@ifnextchar [{\@Glsuserv@{#1}{#2}}{\@Glsuserv@{#1}{#2} []}] }
```

Read in the final optional argument:

```
3898 \def\@Glsuserv@#1#2[#3] {%
3899 \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuserv{#2}#3}}%
3900 }
```

\glsuservi behaves like \gls except it always uses the value given by the userv6 key and it doesn't mark the entry as used.

\glsuservi

```
3901 \newrobustcmd*{\glsuservi}{\@gls@hyp@opt\@glsuservi}
```

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

```
3902 \newcommand*{\@glsuservi}[2] [] {%
3903 \new@ifnextchar [{\@glsuservi@{#1}{#2}}{\@glsuservi@{#1}{#2} []}] }
```

Read in the final optional argument:

```
3904 \def\@glsuservi@#1#2[#3] {%
3905 \@gls@field@link{#1}{#2}{\glsentryuservi{#2}#3}%
3906 }
```

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

\Glsuservi

```
3907 \newrobustcmd*{\Glsuservi}{\@gls@hyp@opt\@Glsuservi}
```

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

```
3908 \newcommand*{\@Glsuservi}[2] [] {%
3909 \new@ifnextchar [{\@Glsuservi@{#1}{#2}}{\@Glsuservi@{#1}{#2} []}] }
```


Read in the final optional argument:

```
3910 \def\@Glsuservi@#1#2[#3]{%
3911   \@gls@field@link{#1}{#2}{\Glsentryuservi{#2}#3}%
3912 }
```

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

\Glsuservi

```
3913 \newrobustcmd*{\Glsuservi}{\@gls@hyp@opt\@Glsuservi}
```

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

```
3914 \newcommand*{\@Glsuservi}[2] [] {%
3915   \new@ifnextchar[{\@Glsuservi@{#1}{#2}}{\@Glsuservi@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3916 \def\@Glsuservi@#1#2[#3]{%
3917   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentryuservi{#2}#3}}%
3918 }
```

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

\acrshort

```
3919 \newrobustcmd*{\acrshort}{\@gls@hyp@opt\@ns@acrshort}
```

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

```
3920 \newcommand*{\@ns@acrshort}[2] [] {%
3921   \new@ifnextchar[{\@acrshort{#1}{#2}}{\@acrshort{#1}{#2} []}]%
3922 }
```

Read in the final optional argument:

```
3923 \def\@acrshort#1#2[#3]{%
3924   \glsdoifexists{#2}%
3925   {%
3926     \let\do@gls@link@checkfirsthyper\relax
3927     \let\glsifplural\@secondoftwo
3928     \let\glscapscase\@firstofthree
3929     \let\glsinsert\@empty
3930     \def\glscustomtext{%
3931       \acronymfont{\Glsentryshort{#2}}#3%
3932     }%
```

Call \@gls@link Note that \@gls@link sets \gls type.

```
3933   \@gls@link[#1]{#2}{\csname gls@\gls type @entryfmt\endcsname}%
3934 }
```

```
3935 \glspostlinkhook
3936 }
```

\Acrshort

```
3937 \newrobustcmd*{\Acrshort}{\@gls@hyp@opt\ns@Acrshort}
```

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

```
3938 \newcommand*{\ns@Acrshort}[2] [] {%
3939   \new@ifnextchar[{\@Acrshort{#1}{#2}}{\@Acrshort{#1}{#2} []}%
3940 }
```

Read in the final optional argument:

```
3941 \def\@Acrshort#1#2[#3] {%
3942   \glsdoifexists{#2}%
3943   {%
3944     \let\do@gls@link@checkfirsthyper\relax
3945     \def\glslabel{#2}%
3946     \let\glsifplural\@secondoftwo
3947     \let\glscapscase\@secondofthree
3948     \let\glsinsert\@empty
3949     \def\glscustomtext{%
3950       \acronymfont{\Glsentryshort{#2}}#3%
3951     }%
```

Call \@gls@link Note that \@gls@link sets \glsstyle.

```
3952   \@gls@link[#1]{#2}{\csname gls@\glsstyle @entryfmt\endcsname}%
3953   }%
3954   \glspostlinkhook
3955 }
```

\ACRshort

```
3956 \newrobustcmd*{\ACRshort}{\@gls@hyp@opt\ns@ACRshort}
```

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

```
3957 \newcommand*{\ns@ACRshort}[2] [] {%
3958   \new@ifnextchar[{\@ACRshort{#1}{#2}}{\@ACRshort{#1}{#2} []}%
3959 }
```

Read in the final optional argument:

```
3960 \def\@ACRshort#1#2[#3] {%
3961   \glsdoifexists{#2}%
3962   {%
3963     \let\do@gls@link@checkfirsthyper\relax
3964     \def\glslabel{#2}%
3965     \let\glsifplural\@secondoftwo
3966     \let\glscapscase\@thirdofthree
3967     \let\glsinsert\@empty
3968     \def\glscustomtext{%
3969       \mfirstucMakeUppercase{\acronymfont{\glsentryshort{#2}}#3}%
3970     }%
```

Call `\@gls@link` Note that `\@gls@link` sets `\gls@type`.

```
3971 \gls@link[#1]{#2}{\csname gls@gls@type @entryfmt\endcsname}%
3972 }%

3973 \glspostlinkhook
3974 }
```

Short plural:

`\acrshortpl`

```
3975 \newrobustcmd*{\acrshortpl}{\@gls@hyp@opt\ns@acrshortpl}
```

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

```
3976 \newcommand*{\ns@acrshortpl}[2] [] {%
3977 \new@ifnextchar[{\acrshortpl{#1}{#2}}{\acrshortpl{#1}{#2} []}%
3978 }
```

Read in the final optional argument:

```
3979 \def\@acrshortpl#1#2[#3] {%
3980 \glsdoifexists{#2}%
3981 {%
3982 \let\do@gls@link@checkfirsthyper\relax

3983 \def\glslabel{#2}%
3984 \let\glsifplural\@firstoftwo
3985 \let\gls@capscase\@firstofthree
3986 \let\glsinsert\@empty
3987 \def\gls@customtext{%
3988 \acronymfont{\glsentryshortpl{#2}}#3%
3989 }%
```

Call `\@gls@link` Note that `\@gls@link` sets `\gls@type`.

```
3990 \gls@link[#1]{#2}{\csname gls@gls@type @entryfmt\endcsname}%
3991 }%

3992 \glspostlinkhook
3993 }
```

`\Acrshortpl`

```
3994 \newrobustcmd*{\Acrshortpl}{\@gls@hyp@opt\ns@Acrshortpl}
```

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

```
3995 \newcommand*{\ns@Acrshortpl}[2] [] {%
3996 \new@ifnextchar[{\Acrshortpl{#1}{#2}}{\Acrshortpl{#1}{#2} []}%
3997 }
```

Read in the final optional argument:

```
3998 \def\@Acrshortpl#1#2[#3]{%
3999   \glsdoifexists{#2}%
4000   {%
4001     \let\do@gl@link@checkfirsthyper\relax

4002     \def\glslabel{#2}%
4003     \let\glsifplural\@firstoftwo
4004     \let\glscapscase\@secondofthree
4005     \let\glsinsert\@empty
4006     \def\glscustomtext{%
4007       \acronymfont{\Glsentryshortpl{#2}}#3%
4008     }%
```

Call `\@gl@link` Note that `\@gl@link` sets `\glstype`.

```
4009   \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
4010   }%

4011   \glspostlinkhook
4012 }
```

`\ACRshortpl`

```
4013 \newrobustcmd*{\ACRshortpl}{\@gl@hyp@opt\@ns@ACRshortpl}
```

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

```
4014 \newcommand*{\ns@ACRshortpl}[2][ ]{%
4015   \new@ifnextchar[{\@ACRshortpl{#1}{#2}}{\@ACRshortpl{#1}{#2} [ ]}%
4016 }
```

Read in the final optional argument:

```
4017 \def\@ACRshortpl#1#2[#3]{%
4018   \glsdoifexists{#2}%
4019   {%
4020     \let\do@gl@link@checkfirsthyper\relax

4021     \def\glslabel{#2}%
4022     \let\glsifplural\@firstoftwo
4023     \let\glscapscase\@thirdofthree
4024     \let\glsinsert\@empty
4025     \def\glscustomtext{%
4026       \mfirstucMakeUppercase{\acronymfont{\Glsentryshortpl{#2}}#3}%
4027     }%
```

Call `\@gl@link` Note that `\@gl@link` sets `\glstype`.

```
4028   \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
4029   }%

4030   \glspostlinkhook
4031 }
```

`\acrlong`

```
4032 \newrobustcmd*{\acrlong}{\@gls@hyp@opt\@ns@acrlong}
```

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

```
4033 \newcommand*{\ns@acrlong}[2] [] {%
```

```
4034   \new@ifnextchar[{\@acrlong{#1}{#2}}{\@acrlong{#1}{#2} []}%
```

```
4035 }
```

Read in the final optional argument:

```
4036 \def\@acrlong#1#2[#3] {%
```

```
4037   \glsdoifexists{#2}%
```

```
4038   {%
```

```
4039     \let\do@gls@link@checkfirsthyper\relax
```

```
4040     \def\glslabel{#2}%
```

```
4041     \let\glsifplural\@secondoftwo
```

```
4042     \let\glsapscase\@firstofthree
```

```
4043     \let\glsinsert\@empty
```

Bug fix v4.02 removed `\acronymfont` from `\glscustomtext` (`\acronymfont` only designed for short form).

```
4044   \def\glscustomtext{%
```

```
4045     \glsentrylong{#2}#3%
```

```
4046   }%
```

Call `\@gls@link` Note that `\@gls@link` sets `\glsstyle`.

```
4047   \@gls@link[#1]{#2}{\csname gls@glsstyle @entryfmt\endcsname}%
```

```
4048   }%
```

```
4049   \glspostlinkhook
```

```
4050 }
```

`\Acrlong`

```
4051 \newrobustcmd*{\Acrlong}{\@gls@hyp@opt\@ns@Acrlong}
```

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

```
4052 \newcommand*{\ns@Acrlong}[2] [] {%
```

```
4053   \new@ifnextchar[{\@Acrlong{#1}{#2}}{\@Acrlong{#1}{#2} []}%
```

```
4054 }
```

Read in the final optional argument:

```
4055 \def\@Acrlong#1#2[#3] {%
```

```
4056   \glsdoifexists{#2}%
```

```
4057   {%
```

```
4058     \let\do@gls@link@checkfirsthyper\relax
```

```
4059     \def\glslabel{#2}%
```

```
4060     \let\glsifplural\@secondoftwo
```

```
4061     \let\glsapscase\@secondofthree
```

```
4062     \let\glsinsert\@empty
```

Bug fix v4.02 removed `\acronymfont` from `\glscustomtext` (`\acronymfont` only designed for short form).

```
4063 \def\glscustomtext{%
4064     \Glsentrylong{#2}#3%
4065 }
```

Call `\@gls@link`. Note that `\@gls@link` sets `\glstype`.

```
4066 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4067 }%

4068 \glspostlinkhook
4069 }
```

`\ACRlong`

```
4070 \newrobustcmd*{\ACRlong}{\@gls@hyp@opt\ns@ACRlong}
```

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

```
4071 \newcommand*{\ns@ACRlong}[2] [] {%
4072     \new@ifnextchar[{\@ACRlong{#1}{#2}}{\@ACRlong{#1}{#2} []}%
4073 }
```

Read in the final optional argument:

```
4074 \def\@ACRlong#1#2[#3] {%
4075     \glsdoifexists{#2}%
4076     {%
4077         \let\do@gls@link@checkfirsthyper\relax

4078         \def\glslabel{#2}%
4079         \let\glsifplural\@secondoftwo
4080         \let\glsapscase\@thirdofthree
4081         \let\glsinsert\@empty
```

Bug fix v4.02 removed `\acronymfont` from `\glscustomtext` (`\acronymfont` only designed for short form).

```
4082 \def\glscustomtext{%
4083     \mfirstucMakeUppercase{\glsentrylong{#2}#3}%
4084 }
```

Call `\@gls@link`. Note that `\@gls@link` sets `\glstype`.

```
4085 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4086 }%

4087 \glspostlinkhook
4088 }
```

Short plural:

`\acrlongpl`

```
4089 \newrobustcmd*{\acrlongpl}{\@gls@hyp@opt\ns@acrlongpl}
```

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

```
4090 \newcommand*\ns@acrlongpl}[2] []{%
4091   \new@ifnextchar[{\@acrlongpl{#1}{#2}}{\@acrlongpl{#1}{#2} []}%
4092 }
```

Read in the final optional argument:

```
4093 \def\@acrlongpl#1#2[#3]{%
4094   \glsdoifexists{#2}%
4095   {%
4096     \let\do@gls@link@checkfirsthyper\relax

4097     \def\glslabel{#2}%
4098     \let\glsifplural\@firstoftwo
4099     \let\gls caps case\@firstofthree
4100     \let\glsinsert\@empty
```

Bug fix v4.02 removed `\acronymfont` from `\gls custom text` (`\acronymfont` only designed for short form).

```
4101   \def\gls custom text{%
4102     \glsentrylongpl{#2}#3%
4103   }%
```

Call `\@gls@link`. Note that `\@gls@link` sets `\gls type`.

```
4104   \@gls@link[#1]{#2}{\csname gls@gls type @entryfmt\endcsname}%
4105   }%

4106   \gls post link hook
4107 }
```

`\Acrlongpl`

```
4108 \newrobustcmd*\Acrlongpl{\@gls@hyp@opt\ns@Acrlongpl}
```

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

```
4109 \newcommand*\ns@Acrlongpl}[2] []{%
4110   \new@ifnextchar[{\@Acrlongpl{#1}{#2}}{\@Acrlongpl{#1}{#2} []}%
4111 }
```

Read in the final optional argument:

```
4112 \def\@Acrlongpl#1#2[#3]{%
4113   \glsdoifexists{#2}%
4114   {%
4115     \let\do@gls@link@checkfirsthyper\relax

4116     \def\glslabel{#2}%
4117     \let\glsifplural\@firstoftwo
4118     \let\gls caps case\@secondofthree
4119     \let\glsinsert\@empty
```

Bug fix v4.02 removed `\acronymfont` from `\glscustomtext` (`\acronymfont` only designed for short form).

```
4120 \def\glscustomtext{%
4121     \Glsentrylongpl{#2}#3%
4122 }
```

Call `\@gls@link`. Note that `\@gls@link` sets `\glstype`.

```
4123 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4124 }%

4125 \glspostlinkhook
4126 }
```

`\ACRlongpl`

```
4127 \newrobustcmd*{\ACRlongpl}{\@gls@hyp@opt\ns@ACRlongpl}
```

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

```
4128 \newcommand*{\ns@ACRlongpl}[2] []{%
4129     \new@ifnextchar[{\@ACRlongpl{#1}{#2}}{\@ACRlongpl{#1}{#2} []}%
4130 }
```

Read in the final optional argument:

```
4131 \def\@ACRlongpl#1#2[#3]{%
4132     \glsdoifexists{#2}%
4133     {%
4134         \let\do@gls@link@checkfirsthyper\relax

4135         \def\glslabel{#2}%
4136         \let\glsifplural\@firstoftwo
4137         \let\gls caps case\@thirdofthree
4138         \let\glsinsert\@empty
```

Bug fix v4.02 removed `\acronymfont` from `\glscustomtext` (`\acronymfont` only designed for short form).

```
4139 \def\glscustomtext{%
4140     \mfirstucMakeUppercase{\Glsentrylongpl{#2}#3}%
4141 }
```

Call `\@gls@link`. Note that `\@gls@link` sets `\glstype`.

```
4142 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4143 }%

4144 \glspostlinkhook
4145 }
```

1.11.2 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>}
```

```
4146 \newcommand*{\@gls@entry@field}[2]{%
4147   \csname glo@glsdetoklabel{#1}@#2\endcsname
4148 }
```

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

```
4149 \newcommand*{\glsletentryfield}[3]{%
4150   \letcs{#1}{glo@glsdetoklabel{#2}@#3}%
4151 }
```

`\@Gls@entry@field` Generic first letter uppercase version.

```
\@Gls@entry@field{<label>}{<field>}
```

```
4152 \newcommand*{\@Gls@entry@field}[2]{%
4153   \letcs@glo@text{glo@glsdetoklabel{#1}@#2}%
4154   \ifdef@glo@text
4155     {%
4156       \xmakefirstuc@glo@text}%
4157     }%
4158     {%
4159       \PackageError{glossaries}{Either glossary entry
4160         ‘glsdetoklabel{#1}’ doesn’t exist or the field ‘#2’
4161         doesn’t exist}{Check you have correctly spelt the entry
4162         label and the field name}%
4163     }%
4164 }
```

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`

```
4165 \newcommand*{\glsentryname}[1]{\@gls@entry@field{#1}{name}}
```

`\Glsentryname`

```
4166 \newrobustcmd*{\Glsentryname}[1]{%
4167   \@Gls@entryname{#1}%
4168 }
```

`\@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:

```
4169 \newcommand*{\@Gls@entryname}[1]{%
4170   \@Gls@entry@field{#1}{name}%
4171 }
```

`\@Gls@acrentryname` Now the behaviour when `\setacronymstyle` is used:

```
4172 \newcommand*{\@Gls@acrentryname}[1]{%
4173   \ifglshaslong{#1}%
4174   {%
4175     \letcs\@glo@text{glo\@glsdetoklabel{#1}@name}%
4176     \expandafter\@gls@getbody\@glo@text{}\@nil
4177     \expandafter\ifx\@gls@body\glsentrylong\relax
4178     \expandafter\Glsentrylong\@gls@rest
4179   }else
4180     \expandafter\ifx\@gls@body\glsentryshort\relax
4181     \expandafter\Glsentryshort\@gls@rest
4182   }else
4183     \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.

```
4184     {%
4185       \let\glsentryshort\Glsentryshort
4186       \@glo@text
4187     }%
4188   }else
4189     \xmakefirstuc{\@glo@text}%
4190   \fi
4191 \fi
4192 \fi
4193 }%
4194 {%
```

Not an acronym

```
4195   \@Gls@entry@field{#1}{name}%
4196 }%
4197 }
```

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`

```
4198 \newcommand*{\glsentrydesc}[1]{\@gls@entry@field{#1}{desc}}
```

`\Glsentrydesc`

```
4199 \newrobustcmd*{\Glsentrydesc}[1]{%
4200   \@Gls@entry@field{#1}{desc}%
4201 }
```

Plural form:

`\glsentrydescplural`

```
4202 \newcommand*{\glsentrydescplural}[1]{%
4203   \@gls@entry@field{#1}{descplural}%
4204 }
```

`\Glsentrydescplural`

```
4205 \newrobustcmd*{\Glsentrydescplural}[1]{%
4206   \@Gls@entry@field{#1}{descplural}%
4207 }
```

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`

```
4208 \newcommand*{\glsentrytext}[1]{\@gls@entry@field{#1}{text}}
```

`\Glsentrytext`

```
4209 \newrobustcmd*{\Glsentrytext}[1]{%
4210   \@Gls@entry@field{#1}{text}%
4211 }
```

Get the plural form:

`\glsentryplural`

```
4212 \newcommand*{\glsentryplural}[1]{%
4213   \@gls@entry@field{#1}{plural}%
4214 }
```

`\Glsentryplural`

```
4215 \newrobustcmd*{\Glsentryplural}[1]{%
4216   \@Gls@entry@field{#1}{plural}%
4217 }
```

Get the symbol associated with this entry. The argument is the label associated with the entry.

`\glsentrysymbol`

```
4218 \newcommand*{\glsentrysymbol}[1]{%
4219   \@gls@entry@field{#1}{symbol}%
4220 }
```

`\Glsentrysymbol`

```
4221 \newrobustcmd*{\Glsentrysymbol}[1]{%
4222   \@Gls@entry@field{#1}{symbol}%
4223 }
```

Plural form:

`lgsentrysymbolplural`

```
4224 \newcommand*{\lgsentrysymbolplural}[1]{%
4225   \@Gls@entry@field{#1}{symbolplural}%
4226 }
```

`lgsentrysymbolplural`

```
4227 \newrobustcmd*{\Glsentrysymbolplural}[1]{%
4228   \@Gls@entry@field{#1}{symbolplural}%
4229 }
```

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).

`\glentryfirst`

```
4230 \newcommand*{\glentryfirst}[1]{%
4231   \@Gls@entry@field{#1}{first}%
4232 }
```

`\Glsentryfirst`

```
4233 \newrobustcmd*{\Glsentryfirst}[1]{%
4234   \@Gls@entry@field{#1}{first}%
4235 }
```

Get the plural form (as specified by the firstplural key when the entry was defined).

`glentryfirstplural`

```
4236 \newcommand*{\glentryfirstplural}[1]{%
4237   \@Gls@entry@field{#1}{firstpl}%
4238 }
```

`Glsentryfirstplural`

```
4239 \newrobustcmd*{\Glsentryfirstplural}[1]{%
4240   \@Gls@entry@field{#1}{firstpl}%
4241 }
```

Display the glossary type with which this entry is associated (as specified by the type key used when the entry was defined)

`\glentrytype`

```
4242 \newcommand*{\glentrytype}[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`

```
4243 \newcommand*{\glsentrysort}[1]{%
4244   \@gls@entry@field{#1}{sort}%
4245 }
```

`\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.

```
4246 \newcommand*{\glsentryuseri}[1]{%
4247   \@gls@entry@field{#1}{useri}%
4248 }
```

`\Glsentryuseri`

```
4249 \newrobustcmd*{\Glsentryuseri}[1]{%
4250   \@Gls@entry@field{#1}{useri}%
4251 }
```

`\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.

```
4252 \newcommand*{\glsentryuserii}[1]{%
4253   \@gls@entry@field{#1}{userii}%
4254 }
```

`\Glsentryuserii`

```
4255 \newrobustcmd*{\Glsentryuserii}[1]{%
4256   \@Gls@entry@field{#1}{userii}%
4257 }
```

`\glsentryuseriii` Get the third user key (as specified by the user3 when the entry was defined).
The argument is the label associated with the entry.

```
4258 \newcommand*{\glsentryuseriii}[1]{%
4259   \@gls@entry@field{#1}{useriii}%
4260 }
```

`\Glsentryuseriii`

```
4261 \newrobustcmd*{\Glsentryuseriii}[1]{%
4262   \@Gls@entry@field{#1}{useriii}%
4263 }
```

`\glsentryuseriv` Get the fourth user key (as specified by the user4 when the entry was defined).
The argument is the label associated with the entry.

```
4264 \newcommand*{\glsentryuseriv}[1]{%
4265   \@gls@entry@field{#1}{useriv}%
4266 }
```

```

\Glsentryuseriv
4267 \newrobustcmd*{\Glsentryuseriv}[1]{%
4268   \@Gls@entry@field{#1}{useriv}%
4269 }

\glsentryuseriv  Get the fifth user key (as specified by the user5 when the entry was defined).
                  The argument is the label associated with the entry.
4270 \newcommand*{\glsentryuseriv}[1]{%
4271   \@Gls@entry@field{#1}{useriv}%
4272 }

\Glsentryuseriv
4273 \newrobustcmd*{\Glsentryuseriv}[1]{%
4274   \@Gls@entry@field{#1}{useriv}%
4275 }

\glsentryuserivi  Get the sixth user key (as specified by the user6 when the entry was defined).
                   The argument is the label associated with the entry.
4276 \newcommand*{\glsentryuserivi}[1]{%
4277   \@Gls@entry@field{#1}{userivi}%
4278 }

\Glsentryuserivi
4279 \newrobustcmd*{\Glsentryuserivi}[1]{%
4280   \@Gls@entry@field{#1}{userivi}%
4281 }

\glsentryshort  Get the short key (as specified by the short the entry was defined). The argu-
                 ment is the label associated with the entry.
4282 \newcommand*{\glsentryshort}[1]{\@Gls@entry@field{#1}{short}}

\Glsentryshort
4283 \newrobustcmd*{\Glsentryshort}[1]{%
4284   \@Gls@entry@field{#1}{short}%
4285 }

\glsentryshortpl  Get the short plural key (as specified by the shortplural the entry was defined).
                   The argument is the label associated with the entry.
4286 \newcommand*{\glsentryshortpl}[1]{\@Gls@entry@field{#1}{shortpl}}

\Glsentryshortpl
4287 \newrobustcmd*{\Glsentryshortpl}[1]{%
4288   \@Gls@entry@field{#1}{shortpl}%
4289 }

\glsentrylong  Get the long key (as specified by the long the entry was defined). The argument
                is the label associated with the entry.
4290 \newcommand*{\glsentrylong}[1]{\@Gls@entry@field{#1}{long}}

```

`\Glsentrylong`

```
4291 \newrobustcmd*{\Glsentrylong}[1]{%
4292   \@Gls@entry@field{#1}{long}%
4293 }
```

`\glsentrylongpl` Get the long plural key (as specified by the longplural the entry was defined).
The argument is the label associated with the entry.

```
4294 \newcommand*{\glsentrylongpl}[1]{\@Gls@entry@field{#1}{longpl}}
```

`\Glsentrylongpl`

```
4295 \newrobustcmd*{\Glsentrylongpl}[1]{%
4296   \@Gls@entry@field{#1}{longpl}%
4297 }
```

Short cut macros to access full form:

`\glsentryfull`

```
4298 \newcommand*{\glsentryfull}[1]{%
4299   \acrfullformat{\glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%
4300 }
```

`\Glsentryfull`

```
4301 \newrobustcmd*{\Glsentryfull}[1]{%
4302   \acrfullformat{\Glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%
4303 }
```

`\glsentryfullpl`

```
4304 \newcommand*{\glsentryfullpl}[1]{%
4305   \acrfullformat{\glsentrylongpl{#1}}{\acronymfont{\glsentryshortpl{#1}}}%
4306 }
```

`\Glsentryfullpl`

```
4307 \newrobustcmd*{\Glsentryfullpl}[1]{%
4308   \acrfullformat{\Glsentrylongpl{#1}}{\acronymfont{\glsentryshortpl{#1}}}%
4309 }
```

`\glsentrynumberlist` Displays the number list as is.

```
4310 \newcommand*{\glsentrynumberlist}[1]{%
4311   \glsdoifexists{#1}%
4312   {%
4313     \@Gls@entry@field{#1}{numberlist}%
4314   }%
4315 }
```

`\glsdisplaynumberlist` Formats the number list for the given entry label. Doesn't work with hyperref.

```
4316 \@ifpackageloaded{hyperref} {%
4317   \newcommand*{\glsdisplaynumberlist}[1]{%
4318     \GlossariesWarning
```

```

4319  {%
4320    \string\glsdisplaynumberlist\space
4321    doesn't work with hyperref.^^JUsing
4322    \string\glsentrynumberlist\space instead%
4323  }%
4324  \glsentrynumberlist{#1}%
4325 }%
4326 }%
4327 {%
4328  \newcommand*{\glsdisplaynumberlist}[1]{%
4329    \glsdoifexists{#1}%
4330    {%
4331      \bgroup
4332        \edef\@glo@label{\glsdetoklabel{#1}}%
4333        \let\@org@glsnumberformat\glsnumberformat
4334        \def\glsnumberformat##1{##1}%
4335        \protected@edef\the@numberlist{%
4336          \csname glo@\@glo@label @numberlist\endcsname}%
4337        \def\@gls@numlist@sep{}%
4338        \def\@gls@numlist@nextsep{}%
4339        \def\@gls@numlist@lastsep{}%
4340        \def\@gls@thislist{}%
4341        \def\@gls@donext@def{}%
4342        \renewcommand\do[1]{%
4343          \protected@edef\@gls@thislist{%
4344            \@gls@thislist
4345            \noexpand\@gls@numlist@sep
4346            ##1%
4347          }%
4348          \let\@gls@numlist@sep\@gls@numlist@nextsep
4349          \def\@gls@numlist@nextsep{\glsnumlistsep}%
4350          \@gls@donext@def
4351          \def\@gls@donext@def{%
4352            \def\@gls@numlist@lastsep{\glsnumlistlastsep}%
4353          }%
4354        }%
4355        \expandafter \glsnumlistparser \expandafter{\the@numberlist}%
4356        \let\@gls@numlist@sep\@gls@numlist@lastsep
4357        \@gls@thislist
4358      \egroup
4359    }%
4360  }
4361 }

```

\glsnumlistsep

```
4362 \newcommand*{\glsnumlistsep}{, }
```

\glsnumlistlastsep

```
4363 \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.

```
4364 \newcommand*{\glshyperlink}[2][\glsentrytext{\@glo@label}]{%
4365 \def\@glo@label{#2}%
4366 \@glslink{\glo@linkprefix\glsdetoklabel{#2}}{#1}}
```

1.12 Adding an entry to the glossary without generating text

The following keys are provided for `\glsadd` and `\glsaddall`:

```
4367 \define@key{glossadd}{counter}{\def\@gls@counter{#1}}
4368 \define@key{glossadd}{format}{\def\@glsnumberformat{#1}}
```

This key is only used by `\glsaddall`:

```
4369 \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`

```
4370 \newrobustcmd*{\glsadd}[2][ ]{%
  Need to move to horizontal mode if not already in it, but only if not in preamble.
4371 \@gls@adjustmode
4372 \glsdoifexists{#2}%
4373 {%
4374 \def\@glsnumberformat{glsnumberformat}%
4375 \edef\@gls@counter{\csname glo@%glsdetoklabel{#2}@counter\endcsname}%
4376 \setkeys{glossadd}{#1}%
```

Store the entry's counter in `\theglentrycounter`

```
4377 \@gls@saveentrycounter
```

This should use `\@do@wrglossary` rather than `\do@wrglossary` since the whole point of `\glsadd` is to add a line to the glossary.

```
4378 \@do@wrglossary{#2}%
4379 }%
4380 }
```

`\@gls@adjustmode`

```
4381 \newcommand*{\@gls@adjustmode}{}
4382 \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`

```
4383 \newrobustcmd*{\glsaddall}[1] [] {%
4384   \edef\@glo@type{\@glo@types}%
4385   \setkeys{glossadd}{#1}%
4386   \forallglsentries[\@glo@type]{\@glo@entry}{%
4387     \glsadd[#1]{\@glo@entry}%
4388   }%
4389 }
```

`\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.

```
4390 \newrobustcmd*{\glsaddallunused}[1] [\@glo@types] {%
4391   \forallglsentries[#1]{\@glo@entry}%
4392   {%
4393     \ifglsused{\@glo@entry}{\glsadd[format=glsignore]{\@glo@entry}}%
4394   }%
4395 }
```

`\glsignore`

```
4396 \newcommand*{\glsignore}[1] {}
```

1.13 Creating associated files

The `\writeist` command creates the associated customized `.ist` `makeindex` style file. While defining this command, some characters have their catcodes temporarily changed to ensure they get written to the `.ist` file correctly. The `makeindex` actual character (usually `@`) is redefined to be a `?`, to allow internal commands to be written to the glossary file output file.

The special characters are stored in `\@gls@actualchar`, `\@gls@encapchar`, `\@gls@levelchar` and `\@gls@quotechar` to make them easier to use later, but don't change these values, because the characters are encoded in the command definitions that are used to escape the special characters (which means that the user no longer needs to worry about `makeindex` special characters).

The symbols and numbers label for group headings are hardwired into the `.ist` file as `glsymbols` and `glsnumbers`, the group titles can be translated (so that `\glsymbolsgroupname` replaces `glsymbols` and `\glsnumbersgroupname`

replaces `glsnumbers`) using the command `\glsgetgrouptitle` which is defined in `.` This is done to prevent any problem characters in `\glssymbolsgroupname` and `\glsnumbersgroupname` from breaking hyperlinks.

`\glsopenbrace` Define `\glsopenbrace` to make it easier to write an opening brace to a file.

```
4397 \edef\glsopenbrace{\expandafter\@gobble\string\{}
```

`\glsclosebrace` Define `\glsclosebrace` to make it easier to write an opening brace to a file.

```
4398 \edef\glsclosebrace{\expandafter\@gobble\string\}}
```

`\glsbackslash` Define `\glsbackslash` to make it easier to write a backslash to a file.

```
4399 \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.

```
4400 \edef\glsquote#1{\string"#1\string"}
```

`\glspercentchar` Define `\glspercentchar` to make it easier to write a percent character to a file.

```
4401 \edef\glspercentchar{\expandafter\@gobble\string\%}
```

`\glstildechar` Define `\glstildechar` to make it easier to write a tilde character to a file.

```
4402 \edef\glstildechar{\string~}
```

`\@glsfirstletter` Define the first letter to come after the digits 0,...,9. Only required for `xindy`.

```
4403 \ifglsxindy
4404   \newcommand*{\@glsfirstletter}{A}
4405 \fi
```

`stLetterAfterDigits` Sets the first letter to come after the digits 0,...,9.

```
4406 \ifglsxindy
4407   \newcommand*{\GlsSetXdyFirstLetterAfterDigits}[1]{%
4408     \renewcommand*{\@glsfirstletter}{#1}}
4409 \else
4410   \newcommand*{\GlsSetXdyFirstLetterAfterDigits}[1]{%
4411     \glsnoxindywarning\GlsSetXdyFirstLetterAfterDigits}
4412 \fi
```

`\@glsminrange` Define the minimum number of successive location references to merge into a range.

```
4413 \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`.

```
4414 \ifglsxindy
4415   \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
4416     \renewcommand*{\@glsminrange}{#1}}
```

```

4417 \else
4418   \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
4419     \glsnoxindywarning\GlsSetXdyMinRangeLength}
4420 \fi

```

`\writeist`

```

4421 \ifglxindy
    Code to use if xindy is required.
4422   \def\writeist{%
    Define write register if not already defined
4423     \ifundef{\glswrite}{\newwrite\glswrite}{}%
    Update attributes list
4424     \@gls@addpredefinedattributes
    Open the file.
4425     \openout\glswrite=\istfilename
    Write header comment at the start of the file
4426     \write\glswrite{;; xindy style file created by the glossaries
4427       package}%
4428     \write\glswrite{;; for document '\jobname' on
4429       \the\year-\the\month-\the\day}%
    Specify the required styles
4430     \write\glswrite{^^J; required styles^^J}
4431     \@for\@xdystyle:=\@xdyrequiredstyles\do{%
4432       \ifx\@xdystyle\@empty
4433       \else
4434         \protected@write\glswrite{{(require
4435           \string"\@xdystyle.xdy\string")}}%
4436       \fi
4437     }%
    List the allowed attributes (possible values used by the format key)
4438     \write\glswrite{^^J%
4439       ; list of allowed attributes (number formats)^^J}%
4440     \write\glswrite{(define-attributes ((\@xdyattributes)))}%
    Define any additional alphabets
4441     \write\glswrite{^^J; user defined alphabets^^J}%
4442     \write\glswrite{\@xdyuseralphabets}%
    Define location classes.
4443     \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.
4444     \@for\@gls@classI:=\@gls@xdy@locationlist\do{%

```

Case were $\langle Hprefix \rangle$ is empty:

```
4445 \protected@write\glswrite{}{(define-location-class
4446 \string"@gls@classI\string"^^J\space\space\space
4447 (
4448 :sep "{-{"
4449 \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
4450 :sep "}"
4451 )
4452 ^^J\space\space\space
4453 :min-range-length \glsminrange^^J%
4454 )
4455 }%
```

Nested iteration over all classes:

```
4456 {%
4457 \@for\@gls@classII:=\@gls@xdy@locationlist\do{%
4458 \protected@write\glswrite{}{(define-location-class
4459 \string"@gls@classII-\@gls@classI\string"
4460 ^^J\space\space\space
4461 (
4462 :sep "{"
4463 \csname @gls@xdy@Lclass@\@gls@classII\endcsname\space
4464 :sep "-{"
4465 \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
4466 :sep "}"
4467 )
4468 ^^J\space\space\space
4469 :min-range-length \glsminrange^^J%
4470 )
4471 }%
4472 }%
4473 }%
4474 }%
```

User defined location classes (needs checking for new location format).

```
4475 \write\glswrite{^^J; user defined location classes}%
4476 \write\glswrite{\@xdyuserlocationdefs}%
```

Cross-reference class. (The unverified option is used as the cross-references are supplied using the list of labels along with the optional argument for $\backslash\text{glsseeformat}$ which xindy won't recognise.)

```
4477 \write\glswrite{^^J; define cross-reference class^^J}%
4478 \write\glswrite{(define-crossref-class \string"see\string"
4479 :unverified )}%
```

Define how cross-references should be displayed. This adds an empty set of braces after the cross-referencing information allowing for the final argument of $\backslash\text{glsseeformat}$ which gets ignored. (When using `makeindex` this final argument contains the location information which is not required.)

```
4480 \write\glswrite{(markup-crossref-list
```

```

4481         :class \string"see\string"^^J\space\space\space
4482         :open \string"\string\glsseeformat\string"
4483         :close \string"{}\string")}%

```

List the order to sort the classes.

```

4484     \write\glswrite{^^J; define the order of the location classes}%
4485     \write\glswrite{(define-location-class-order
4486         (\@xdylocationclassorder))}%

```

Specify what to write to the start and end of the glossary file.

```

4487     \write\glswrite{^^J; define the glossary markup^^J}%

4488     \write\glswrite{(markup-index^^J\space\space\space
4489         :open \string"\string
4490         \glossarysection[\string\glossarytoctitle]{\string
4491         \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)

```

4492     \@for\@this@ctr:=\@xdycounters\do{%
4493         {%
4494             \@for\@this@attr:=\@xdyattributelist\do{%
4495                 \protected\write\glswrite-{}{\string\providecommand*%
4496                     \expandafter\string
4497                     \csname glsX\@this@ctr X\@this@attr\endcsname[2]%
4498                 {%
4499                     \string\setentrycounter
4500                     [\expandafter\@gobble\string\#1]{\@this@ctr}%
4501                     \expandafter\string
4502                     \csname\@this@attr\endcsname
4503                     {\expandafter\@gobble\string\#2}%
4504                 }%
4505             }%
4506         }%
4507     }%
4508 }%

```

Add the end part of the open tag and the rest of the markup-index information:

```

4509     \write\glswrite{%
4510         \string\begin
4511         {theglossary}\string\glossaryheader\glstildechar n\string" ^^J\space
4512         \space\space:close \string"\glspercentchar\glstildechar n\string
4513         \end{theglossary}\string\glossarypostamble
4514         \glstildechar n\string" ^^J\space\space\space
4515         :tree)}}%

```

Specify what to put between letter groups

```

4516     \write\glswrite{(markup-letter-group-list
4517         :sep \string"\string\glsgroupskip\glstildechar n\string")}%

```

Specify what to put between entries

```

4518     \write\glswrite{(markup-indexentry

```

```

4519         :open \string"\string\relax \string\glsresetentrylist
4520         \glstildechar n\string")}]%

```

Specify how to format entries

```

4521     \write\glswrite{(markup-locclass-list :open
4522         \string"\glsopenbrace\string\glossaryentrynumbers
4523         \glsopenbrace\string\relax\space \string"^^J\space\space\space
4524         :sep \string", \string"
4525         :close \string"\glsclosebrace\glsclosebrace\string")}]%

```

Specify how to separate location numbers

```

4526     \write\glswrite{(markup-locref-list
4527         :sep \string"\string\delimN\space\string")}]%

```

Specify how to indicate location ranges

```

4528     \write\glswrite{(markup-range
4529         :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.

```

4530     \@onelevel@sanitize\gls@suffiXF
4531     \@onelevel@sanitize\gls@suffiFF

4532     \ifx\gls@suffiXF\@empty
4533     \else
4534         \write\glswrite{(markup-range
4535             :close "\gls@suffiXF" :length 1 :ignore-end)}%
4536     \fi
4537     \ifx\gls@suffiFF\@empty
4538     \else
4539         \write\glswrite{(markup-range
4540             :close "\gls@suffiFF" :length 2 :ignore-end)}%
4541     \fi

```

Specify how to format locations.

```

4542     \write\glswrite{^^J; define format to use for locations^^J}%
4543     \write\glswrite{\@xdylocref}%

```

Specify how to separate letter groups.

```

4544     \write\glswrite{^^J; define letter group list format^^J}%
4545     \write\glswrite{(markup-letter-group-list
4546         :sep \string"\string\glsgroupskip\glstildechar n\string")}]%

```

Define letter group headings.

```

4547     \write\glswrite{^^J; letter group headings^^J}%
4548     \write\glswrite{(markup-letter-group
4549         :open-head \string"\string\glsgroupheading
4550         \glsopenbrace\string"^^J\space\space\space
4551         :close-head \string"\glsclosebrace\string")}]%

```

Define additional letter groups.

```

4552     \write\glswrite{^^J; additional letter groups^^J}%
4553     \write\glswrite{\@xdylettergroups}%

```

Define additional sort rules

```
4554 \write\glswrite{^^J; additional sort rules^^J}
4555 \write\glswrite{\@xdysortrules}%
```

Close the style file

```
4556 \closeout\glswrite
```

Suppress any further calls.

```
4557 \let\writeist\relax
4558 }
4559 \else
```

Code to use if makeindex is required.

```
4560 \edef\@gls@actualchar{\string?}
4561 \edef\@gls@encapchar{\string|}
4562 \edef\@gls@levelchar{\string!}
4563 \edef\@gls@quotechar{\string"}
4564 \def\writeist{\relax
4565 \ifundef{\glswrite}{\newwrite\glswrite}{}\relax
4566 \openout\glswrite=\istfilename
4567 \write\glswrite{\glspercentchar\space makeindex style file
4568 created by the glossaries package}
4569 \write\glswrite{\glspercentchar\space for document
4570 '\jobname' on \the\year-\the\month-\the\day}
4571 \write\glswrite{actual '@gls@actualchar'}
4572 \write\glswrite{encap '@gls@encapchar'}
4573 \write\glswrite{level '@gls@levelchar'}
4574 \write\glswrite{quote '@gls@quotechar'}
4575 \write\glswrite{keyword \string"\string\glossaryentry\string"}
4576 \write\glswrite{preamble \string"\string\glossarysection[\string
4577 \glossarytoctitle]{\string\glossarytitle}\string
4578 \glossarypreamble\string\n\string\begin{theglossary}\string
4579 \glossaryheader\string\n\string"}
4580 \write\glswrite{postamble \string"\string%\string\n\string
4581 \end{theglossary}\string\glossarypostamble\string\n
4582 \string"}
4583 \write\glswrite{group_skip \string"\string\glsgroupskip\string\n
4584 \string"}
4585 \write\glswrite{item_0 \string"\string%\string\n\string"}
4586 \write\glswrite{item_1 \string"\string%\string\n\string"}
4587 \write\glswrite{item_2 \string"\string%\string\n\string"}
4588 \write\glswrite{item_01 \string"\string%\string\n\string"}
4589 \write\glswrite{item_x1
4590 \string"\string\relax \string\glsresetentrylist\string\n
4591 \string"}
4592 \write\glswrite{item_12 \string"\string%\string\n\string"}
4593 \write\glswrite{item_x2
4594 \string"\string\relax \string\glsresetentrylist\string\n
4595 \string"}

4596 \write\glswrite{delim_0 \string"\string{\string
```



```

4597     \glossaryentrynumbers\string\{\string\relax \string}
4598 \write\glswrite{delim_1 \string}\string\{\string
4599     \glossaryentrynumbers\string\{\string\relax \string}
4600 \write\glswrite{delim_2 \string}\string\{\string
4601     \glossaryentrynumbers\string\{\string\relax \string}
4602 \write\glswrite{delim_t \string}\string\}\string\}\string}
4603 \write\glswrite{delim_n \string}\string\delimN \string}
4604 \write\glswrite{delim_r \string}\string\delimR \string}
4605 \write\glswrite{headings_flag 1}
4606 \write\glswrite{heading_prefix
4607     \string}\string\glsgroupheading\string\{\string}
4608 \write\glswrite{heading_suffix
4609     \string}\string\}\string\relax
4610     \string\glsresetentrylist \string}
4611 \write\glswrite{symhead_positive \string}glsymbols\string}
4612 \write\glswrite{numhead_positive \string}glsnumbers\string}
4613 \write\glswrite{page_compositor \string}glscompositor\string}
4614 \@gls@escbsdq\gls@suffixF
4615 \@gls@escbsdq\gls@suffixFF
4616 \ifx\gls@suffixF\@empty
4617 \else
4618     \write\glswrite{suffix_2p \string}\gls@suffixF\string}
4619 \fi
4620 \ifx\gls@suffixFF\@empty
4621 \else
4622     \write\glswrite{suffix_3p \string}\gls@suffixFF\string}
4623 \fi
4624 \closeout\glswrite
4625 \let\writeist\relax
4626 }
4627 \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`

```

4628 \newcommand{\noist}{%
    Update attributes list
4629 \@gls@addpredefinedattributes
4630 \let\writeist\relax
4631 }

```

`\@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 $\text{T}_{\text{E}}\text{X}$ is trying to write to a non-existent file). The relevant glossary must be defined prior to using `\@makeglossary`.

`\@makeglossary`

```

4632 \newcommand*\@makeglossary}[1]{%
4633   \ifglossaryexists{#1}%
4634   {%
      Only create a new write if savewrites=false otherwise create a token to collect
      the information.
4635     \ifglssavewrites
4636       \expandafter\newtoks\csname glo@#1@filetok\endcsname
4637     \else
4638       \expandafter\newwrite\csname glo@#1@file\endcsname
4639       \expandafter\@glsopenfile\csname glo@#1@file\endcsname{#1}%
4640     \fi
4641     \@gls@renewglossary
4642     \writeist
4643   }%
4644   {%
4645     \PackageError{glossaries}%
4646     {Glossary type ‘#1’ not defined}%
4647     {New glossaries must be defined before using \string\makeglossary}%
4648   }%
4649 }

```

`\@glsopenfile` Open write file associated with the given glossary.

```

4650 \newcommand*\@glsopenfile}[2]{%
4651   \immediate\openout#1=\jobname.\csname @glotype@#2@out\endcsname
4652   \PackageInfo{glossaries}{Writing glossary file
4653     \jobname.\csname @glotype@#2@out\endcsname}%
4654 }

```

`\@closegls`

```

4655 \newcommand*\@closegls}[1]{%
4656   \closeout\csname glo@#1@file\endcsname
4657 }
4658 % \end{macrocode}
4659 %\end{macro}
4660 %
4661 %\begin{macro}\@gls@automake}
4662 %\changes{4.08}{2014-07-30}{new}
4663 % \begin{macrocode}
4664 \ifglxsindy
4665 \newcommand*\@gls@automake}[1]{%
4666   \ifglossaryexists{#1}
4667   {%
4668     \@closegls{#1}%

```

```

4669 \ifdefstring{\glsorder}{letter}%
4670 {\def\@gls@order{-M ord/letorder }}}%
4671 {\let\@gls@order\@empty}%
4672 \ifcsundef{@xdy@#1@language}%
4673 {\let\@gls@langmod\@xdy@main@language}%
4674 {\letcs\@gls@langmod{@xdy@#1@language}}}%
4675 \edef\@gls@dothiswrite{\noexpand\write18{xindy
4676 -I xindy
4677 \@gls@order
4678 -L \@gls@langmod\space
4679 -M \gls@istfilebase\space
4680 -C \gls@codepage\space
4681 -t \jobname.\csuse{@glotype@#1@log}
4682 -o \jobname.\csuse{@glotype@#1@in}
4683 \jobname.\csuse{@glotype@#1@out}}}%
4684 }%
4685 \@gls@dothiswrite
4686 }%
4687 {%
4688 \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
4689 }%
4690 }
4691 \else
4692 \newcommand*{\@gls@automake}[1]{%
4693 \ifglossaryexists{#1}
4694 {%
4695 \@closegls{#1}%
4696 \ifdefstring{\glsorder}{letter}%
4697 {\def\@gls@order{-l }}}%
4698 {\let\@gls@order\@empty}%
4699 \edef\@gls@dothiswrite{\noexpand\write18{makeindex \@gls@order
4700 -s \istfilename\space
4701 -t \jobname.\csuse{@glotype@#1@log}
4702 -o \jobname.\csuse{@glotype@#1@in}
4703 \jobname.\csuse{@glotype@#1@out}}}%
4704 }%
4705 \@gls@dothiswrite
4706 }%
4707 {%
4708 \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
4709 }%
4710 }
4711 \fi

```

`\warn@nomakeglossaries` Issue warning that `\makeglossaries` hasn't been used.

```
4712 \newcommand*{\@warn@nomakeglossaries}{}

```

Only use this if warning if `\printglossary` has been used without `\makeglossaries`

```
4713 \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`

```

4714 \newcommand*\makeglossaries{%
    Define the write used for style file also used for all other output files if
    savewrites=true.
4715 \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.
4716 \protected@write\@auxout{}{\string\providecommand\string\@glsorder[1]{}%
4717 \protected@write\@auxout{}{\string\providecommand\string\@istfilename[1]{}%
    Write the name of the style file to the aux file (needed by makeglossaries)
4718 \protected@write\@auxout{}{\string\@istfilename{\istfilename}}%
4719 \protected@write\@auxout{}{\string\@glsorder{\glsorder}}
    Iterate through each glossary type and activate it.
4720 \@for\@glo@type:=\@glo@types\do{%
4721 \ifthenelse{equal{\@glo@type}{}}{}}{%
4722 \@makeglossary{\@glo@type}}%
4723 }%
    New glossaries must be created before \makeglossaries so disable \newglossary.
4724 \renewcommand*\newglossary[4] []{%
4725 \PackageError{glossaries}{New glossaries
4726 must be created before \string\makeglossaries}{You need
4727 to move \string\makeglossaries\space after all your
4728 \string\newglossary\space commands}}%
    Any subsequence instances of this command should have no effect
4729 \let\@makeglossary\relax
4730 \let\makeglossary\relax
4731 \let\makeglossaries\relax
    Disable all commands that have no effect after \makeglossaries
4732 \@disable@onlypremakeg
    Allow see key:
4733 \let\gls@checkseeallowed\relax
    Suppress warning about no \makeglossaries
4734 \let\warn@nomakeglossaries\relax
    Activate warning about missing \printglossary
4735 \def\warn@noprintglossary{%
4736 \GlossariesWarningNoLine{No \string\printglossary\space

```

```

4737     or \string\printglossaries\space
4738     found.^^J(Remove \string\makeglossaries\space if you don't want
4739     any glossaries.)^^JThis document will not have a glossary}%
4740 }%

```

Declare list parser for \glsdisplaynumberlist

```

4741 \ifglssavenumberlist
4742   \edef\@gls@dodolistparser{\noexpand\DeclareListParser
4743     {\noexpand\glsnumlistparser}{\delimN}}}%
4744   \@gls@dodolistparser
4745 \fi

```

Prevent user from also using \makenoidxglossaries

```

4746 \let\makenoidxglossaries\@no@makeglossaries

```

Prohibit sort key in printgloss family:

```

4747 \renewcommand*{\@printgloss@setsort}{%
4748   \let\@glo@assign@sortkey\@glo@no@assign@sortkey
4749 }%

```

Check the automake setting:

```

4750 \ifglsautomake
4751   \renewcommand*{\@gls@doautomake}{%
4752     \@for\@gls@type:=\@glo@types\do{%
4753       \ifdefempty{\@gls@type}{}%
4754       {\@gls@automake{\@gls@type}}}%
4755   }%
4756 }%
4757 \fi
4758 }

```

Must occur in the preamble:

```

4759 \@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`

```

4760 \let\makeglossary\makeglossaries

```

If `\makeglossaries` hasn't been used, issue a warning. Also issue a warning if neither `\printglossaries` nor `\printglossary` have been used.

```

4761 \AtEndDocument{%
4762   \warn@nomakeglossaries
4763   \warn@noprintglossary
4764 }

```

makenoidxglossaries Analogous to \makeglossaries this activates the commands needed for \printnoidxglossary

```
4765 \newcommand*\makenoidxglossaries}{%
```

Redefine empty glossary warning:

```
4766 \renewcommand{\@gls@noref@warn}[1]{%
4767 \GlossariesWarning{Empty glossary for
4768 \string\printnoidxglossary[type={##1}].
4769 Rerun may be required (or you may have forgotten to use
4770 commands like \string\gls).}%
4771 }%
```

Don't escape makeindex/xindy characters

```
4772 \let\@gls@checkmkidxchars\@gobble
```

Write glossary information to aux instead of glossary files

```
4773 \let\@do@wrglossary\gls@noidxglossary
```

Switch on group headings that use the character code:

```
4774 \let\@gls@getgrouptitle\@gls@noidx@getgrouptitle
```

Allow see key:

```
4775 \let\gls@checkseeallowed\relax
```

Redefine cross-referencing macro:

```
4776 \renewcommand{\@do@seeglossary}[2]{%
4777 \edef\@gls@label{\glsdetoklabel{##1}}%
4778 \protected@write\@auxout{}{%
4779 \string\@gls@reference
4780 {\csname glo@\@gls@label @type\endcsname}%
4781 {\@gls@label}%
4782 {%
4783 \string\glsseeformat##2}%
4784 }%
4785 }%
4786 }%
```

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.

```
4787 \AtBeginDocument
4788 {%
4789 \write\@auxout{\string\providecommand\string\@gls@reference[3]{}}%
4790 }%
```

Change warning about no glossares

```
4791 \def\warn@noprntglossary{%
4792 \GlossariesWarningNoLine{No \string\printnoidxglossary\space
4793 or \string\printnoidxglossaries ^^J
4794 found. (Remove \string\makenoidxglossaries\space if you
4795 don't want any glossaries.)^^JThis document will not have a glossary}%
4796 }%
```

```

Suppress warning about no \makeglossaries
4797 \let\warn@nomakeglossaries\relax

Prevent user from also using \makeglossaries
4798 \let\makeglossaries\@no@makeglossaries

Allow sort key in printgloss family:
4799 \renewcommand*\@printgloss@setsort}{%
4800 \let\@glo@assign@sortkey\@glo@assign@sortkey

Initialise default sort order:
4801 \def\@glo@sorttype{\@glo@default@sorttype}%
4802 }%

All entries must be defined in the preamble:
4803 \renewcommand*\new@glossaryentry[2]{%
4804 \PackageError{glossaries}{Glossary entries must be
4805 defined in the preamble^^Jwhen you use
4806 \string\makenoidxglossaries}%
4807 {Either move your definitions to the preamble or use
4808 \string\makeglossaries}%
4809 }%

Redefine \glsentrynumberlist
4810 \renewcommand*\glsentrynumberlist}[1]{%
4811 \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
4812 \ifdef\@gls@loclist
4813 {%
4814 \glsnoidxloclist{\@gls@loclist}%
4815 }%
4816 {%
4817 \ifglsentryexists{##1}%
4818 {%
4819 \GlossariesWarning{Missing location list for ‘##1’. Either
4820 a rerun is required or you haven’t referenced the entry.}%
4821 }%
4822 {%
4823 \PackageError{glossaries}{Glossary entry ‘##1’ has not been
4824 defined.}{}%
4825 }%
4826 }%
4827 }%

Redefine \glsdisplaynumberlist
4828 \renewcommand*\glsdisplaynumberlist}[1]{%
4829 \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
4830 \ifdef\@gls@loclist
4831 {%
4832 \def\@gls@noidxloclist@sep{%
4833 \def\@gls@noidxloclist@sep{%
4834 \def\@gls@noidxloclist@sep{%
4835 \glsnumlistsep

```

```

4836     }%
4837     \def\@gls@noidxloclist@finalsep{\glsnumlistlastsep}%
4838     }%
4839     }%
4840     \def\@gls@noidxloclist@finalsep{}%
4841     \def\@gls@noidxloclist@prev{}%
4842     \forlistloop{\glsnoidxdisplayloclisthandler}{\@gls@loclist}%
4843     \@gls@noidxloclist@finalsep
4844     \@gls@noidxloclist@prev
4845     }%
4846     {%
4847     ??\ifglsentryexists{##1}%
4848     {%
4849     \GlossariesWarning{Missing location list for ‘##1’. Either
4850     a rerun is required or you haven’t referenced the entry.}%
4851     }%
4852     {%
4853     \PackageError{glossaries}{Glossary entry ‘##1’ has not been
4854     defined.}{}%
4855     }%
4856     }%
4857     }%

```

Provide a generic way of iterating through the number list:

```

4858 \renewcommand*\glsnumberlistloop}[3]{%
4859 \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
4860 \let\@gls@org@glsnoidxdisplayloc\glsnoidxdisplayloc
4861 \let\@gls@org@glsseeformat\glsseeformat
4862 \let\glsnoidxdisplayloc##2\relax
4863 \let\glsseeformat##3\relax
4864 \ifdef\@gls@loclist
4865 {%
4866 \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
4867 }%
4868 {%
4869 \ifglsentryexists{##1}%
4870 {%
4871 \GlossariesWarning{Missing location list for ‘##1’. Either
4872 a rerun is required or you haven’t referenced the entry.}%
4873 }%
4874 {%
4875 \PackageError{glossaries}{Glossary entry ‘##1’ has not been
4876 defined.}{}%
4877 }%
4878 }%
4879 \let\glsnoidxdisplayloc\@gls@org@glsnoidxdisplayloc
4880 \let\glsseeformat\@gls@org@glsseeformat
4881 }%

```

Modify sanitize sort function


```

4882 \let\@gls@sanitizesort\@gls@noidx@sanitizesort
4883 \let\@gls@nosanitizesort\@gls@noidx@nosanitizesort
4884 \@gls@noidx@setsanitizesort
4885 }

```

Preamble-only command:

```
4886 \onlypreamble{\makenoidxglossaries}
```

```
\glsnumberlistloop \glsnumberlistloop{<label>}{<handler>}
```

```

4887 \newcommand*{\glsnumberlistloop}[2]{%
4888   \PackageError{glossaries}{\string\glsnumberlistloop\space
4889     only works with \string\makenoidxglossaries}{}%
4890 }

```

numberlistloophandler Handler macro for `\glsnumberlistloop`. (The argument should be in the form `\glsnoidxdisplayloc{<prefix>}{<counter>}{<format>}{<n>}`)

```

4891 \newcommand*{\glsnoidxnumberlistloophandler}[1]{%
4892   #1%
4893 }

```

\@no@makeglossaries Can't use both `\makeglossaries` and `\makenoidxglossaries`

```

4894 \newcommand*{\@no@makeglossaries}{%
4895   \PackageError{glossaries}{You can't use both
4896     \string\makeglossaries\space and \string\makenoidxglossaries}%
4897   {Either use one or other (or none) of those commands but not both
4898     together.}%
4899 }

```

\@gls@noref@warn Warning when no instances of `\@gls@reference` found.

```

4900 \newcommand{\@gls@noref@warn}[1]{%
4901   \GlossariesWarning{\string\makenoidxglossaries\space
4902     is required to make \string\printnoidxglossary[type={#1}] work}%
4903 }

```

\gls@noidxglossary Write the glossary information to the aux file:

```

4904 \newcommand*{\gls@noidxglossary}{%
4905   \protected@write\@auxout{}{%
4906     \string\@gls@reference
4907     {\csname glo@\@gls@label @type\endcsname}%
4908     {\@gls@label}%
4909     {\string\glsnoidxdisplayloc
4910       {\@glo@counterprefix}%
4911       {\@gls@counter}%
4912       {\@glsnumberformat}%
4913       {\@glslocref}%
4914     }%
4915   }%
4916 }

```

1.14 Writing information to associated files

`\istfile` Deprecated.

```
4917 \def\istfile{\glswrite}
```

At the end of the document, the files should be created if `savewrites=true`.

```
4918 \AtEndDocument{%
```

```
4919   \glswritefiles
```

```
4920 }
```

`\@glswritefiles` Only write the files if `savewrites=true`

```
4921 \newcommand*{\@glswritefiles}{%
```

Iterate through all the glossaries

```
4922   \forallglossaries{\@glo@type}{%
```

Check for empty glossaries (patch provided by Patrick Häcker)

```
4923     \ifcsundef{glo@\@glo@type @filetok}%
```

```
4924     {%
```

```
4925       \def\gls@tmp{}
```

```
4926     }%
```

```
4927     {%
```

```
4928       \edef\gls@tmp{\expandafter\the
```

```
4929         \csname glo@\@glo@type @filetok\endcsname}%
```

```
4930     }%
```

```
4931     \ifx\gls@tmp\@empty
```

```
4932       \ifx\@glo@type\glsdefaulttype
```

```
4933         \GlossariesWarningNoLine{Glossary ‘\@glo@type’ has no
```

```
4934           entries.^^JRemember to use package option ‘nomain’ if
```

```
4935 you
```

```
4936           don’t want to^^Juse the main glossary}%
```

```
4937       \else
```

```
4938         \GlossariesWarningNoLine{Glossary ‘\@glo@type’ has no
```

```
4939           entries}%
```

```
4940       \fi
```

```
4941     \else
```

```
4942       \@glsopenfile{\glswrite}{\@glo@type}%
```

```
4943       \immediate\write\glswrite{%
```

```
4944         \expandafter\the
```

```
4945         \csname glo@\@glo@type @filetok\endcsname}%
```

```
4946       \immediate\closeout\glswrite
```

```
4947     \fi
```

```
4948   }%
```

```
4949 }
```

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`

```
4950 \if@gls@docloaded
4951 \else
4952   \renewcommand*{\glossary}[1][main]{\gls@glossary{#1}}
4953 \fi
```

The associated number should be stored in `\theglsentrycounter` before using `\gls@glossary`.

`\gls@glossary`

```
4954 \newcommand*{\gls@glossary}[1]{%
4955   \@gls@glossary{#1}%
4956 }
```

`\@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.

```
4957 \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`

```
4958 \newcommand{\@gls@renewglossary}{%
4959   \gdef\@gls@glossary##1{\@bsphack\beginngroup\gls@wrglossary{##1}}%
4960   \let\@gls@renewglossary\@empty
4961 }
```

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 `\gls@link`).

`\gls@wrglossary`

```
4962 \newcommand*{\gls@wrglossary}[2]{%
4963   \ifglssavewrites
4964     \protected@edef\@gls@tmp{\the\csname glo@#1@filetok\endcsname#2}%
4965     \expandafter\global\expandafter\csname glo@#1@filetok\endcsname
4966       \expandafter{\@gls@tmp^^J}%
4967   \else
```

```

4968 \ifcsdef{glo@#1@file}%
4969 {%
4970 \expandafter\protected@write\csname glo@#1@file\endcsname{%
4971 \gls@disablepagerefexpansion}{#2}%
4972 }%
4973 {%
4974 \ifignoredglossary{#1}{}%
4975 {%
4976 \GlossariesWarning{No file defined for glossary '#1'}%
4977 }%
4978 }%
4979 \fi
4980 \endgroup\@esphack
4981 }

```

`\@do@wrglossary`

```

4982 \newcommand*\@do@wrglossary}[1]{%
4983 \glswriteentry{#1}{\@do@wrglossary{#1}}%
4984 }

```

`\glswriteentry` Provide a user level command so the user can customize whether or not a line should be added to the glossary. The arguments are the label and the code that writes to the glossary file.

```

4985 \newcommand*\glswriteentry}[2]{%
4986 \ifglsindexonlyfirst
4987 \ifglsused{#1}{#2}%
4988 \else
4989 #2%
4990 \fi
4991 }

```

`\protected@pagefmts` List of page formats to be protected against expansion.

```

4992 \newcommand{\gls@protected@pagefmts}{%
4993 \gls@numberpage,\gls@alphpage,\gls@Alphpage,\gls@romanpage,\gls@Romanpage%
4994 }

```

`\gls@disablepagerefexpansion`

```

4995 \newcommand*\gls@disablepagerefexpansion{%
4996 \@for\@gls@this:=\gls@protected@pagefmts\do
4997 {%
4998 \expandafter\let\@gls@this\relax
4999 }%
5000 }

```

`\gls@alphpage`

```

5001 \newcommand*\gls@alphpage{\@alph\c@page}

```

`\gls@Alphpage`

```

5002 \newcommand*\gls@Alphpage{\@Alph\c@page}

```

`\gls@numberpage`
 5003 `\newcommand*{\gls@numberpage}{\number\c@page}`

`\gls@romanpage`
 5004 `\newcommand*{\gls@romanpage}{\romannumeral\c@page}`

`\gls@Romanpage`
 5005 `\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).

```
5006 \newcommand*{\glsaddprotectedpagefmt}[1]{%
5007   \eappto\gls@protected@pagefmts{,\expandonce{\csname gls#1page\endcsname}}}%
5008   \csedef{gls#1page}{\expandonce{\csname#1\endcsname}\noexpand\c@page}%
5009   \eappto\@wrglossarynumberhook{%
5010     \noexpand\let\expandonce{\csname org@gls#1\endcsname}%
5011     \expandonce{\csname#1\endcsname}%
5012     \noexpand\def\expandonce{\csname#1\endcsname}{%
5013       \noexpand\@wrglossary@pageformat
5014       \expandonce{\csname gls#1page\endcsname}%
5015       \expandonce{\csname org@gls#1\endcsname}%
5016     }%
5017   }%
5018 }
```

`\@do@wrglossarynumberhook`

Hook used by `\@do@wrglossary`

```
5019 \newcommand*\@wrglossarynumberhook{}
```

`\@do@wrglossary@pageformat`

```
5020 \newcommand{\@wrglossary@pageformat}[3]{%
5021   \ifx#3\c@page #1\else #2#3\fi
5022 }
```

`\@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.

```
5023 \newcommand*{\@do@wrglossary}[1]{%
5024   \begingroup
```

First a bit of hackery to prevent premature expansion of `\c@page`. Store original definitions:

```
5025   \let\orgthe\the
5026   \let\orgnumber\number
5027   \let\orgromannumeral\romannumeral
```

```

5028 \let\orgalph\@alph
5029 \let\orgAlph\@Alph
5030 \let\orgRoman\@Roman

```

Redefine:

```

5031 \def\the##1{%
5032   \ifx##1\c@page \gls@numberpage\else\orgthe##1\fi}%
5033 \def\number##1{%
5034   \ifx##1\c@page \gls@numberpage\else\orgnumber##1\fi}%
5035 \def\romannumeral##1{%
5036   \ifx##1\c@page \gls@romanpage\else\orgromannumeral##1\fi}%
5037 \def\@Roman##1{%
5038   \ifx##1\c@page \gls@Romanpage\else\orgRoman##1\fi}%
5039 \def\@alph##1{%
5040   \ifx##1\c@page \gls@alphpage\else\orgalph##1\fi}%
5041 \def\@Alph##1{%
5042   \ifx##1\c@page \gls@Alphpage\else\orgAlph##1\fi}%

```

Add hook to allow for other number formats:

```
5043 \@wrglossarynumberhook
```

Prevent expansion:

```
5044 \gls@disablepagerefexpansion
```

Now store location in \glslocref:

```

5045 \protected@xdef\@glslocref{\theglsentrycounter}%
5046 \endgroup

```

Escape any special characters

```
5047 \@gls@checkmkidxchars\@glslocref
```

Check if the hyper-location is the same as the location and set the hyper prefix.

```

5048 \expandafter\ifx\theHglSentrycounter\theglsentrycounter\relax
5049 \def\@glo@counterprefix{%
5050 \else
5051 \protected@edef\@glsHlocref{\theHglSentrycounter}%
5052 \@gls@checkmkidxchars\@glsHlocref
5053 \edef\@do@gls@getcounterprefix{\noexpand\@gls@getcounterprefix
5054   {\@glslocref}{\@glsHlocref}%
5055 }%
5056 \@do@gls@getcounterprefix
5057 \fi

```

De-tok label if required

```
5058 \edef\@gls@label{\glsdetoklabel{#1}}%
```

Write the information to file:

```

5059 \@do@@wrglossary
5060 }

```

```
\@do@@wrglossary
```

```
5061 \newcommand*{\@do@@wrglossary}{%
```

Determine whether to use xindy or makeindex syntax

```
5062 \ifglxindy
```

Need to determine if the formatting information starts with a (or) indicating a range.

```
5063 \expandafter\@glo@check@mkidxrangear\@glsnumberformat\@nil
5064 \def\@glo@range{}%
5065 \expandafter\if\@glo@prefix(\relax
5066 \def\@glo@range{:open-range}%
5067 \else
5068 \expandafter\if\@glo@prefix)\relax
5069 \def\@glo@range{:close-range}%
5070 \fi
5071 \fi
```

Write to the glossary file using xindy syntax.

```
5072 \gls@glossary{\csname glo@\@gls@label @type\endcsname}{%
5073 (indexentry :tkey (\csname glo@\@gls@label @index\endcsname)
5074 :locref \string"{\@glo@counterprefix}{\@glslocref}\string" %
5075 :attr \string"\@gls@counter\@glo@suffix\string"
5076 \@glo@range
5077 )
5078 }%
5079 \else
```

Convert the format information into the format required for makeindex

```
5080 \@set@glo@numformat{\@glo@numfmt}{\@gls@counter}{\@glsnumberformat}%
5081 {\@glo@counterprefix}%
```

Write to the glossary file using makeindex syntax.

```
5082 \gls@glossary{\csname glo@\@gls@label @type\endcsname}{%
5083 \string\glossaryentry{\csname glo@\@gls@label @index\endcsname
5084 \@gls@encapchar\@glo@numfmt}{\@glslocref}}%
5085 \fi
5086 }
```

`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.)

```
5087 \newcommand*\@gls@getcounterprefix[2]{%
5088 \edef\@gls@thisloc{#1}\edef\@gls@thisHloc{#2}%
5089 \ifx\@gls@thisloc\@gls@thisHloc
5090 \def\@glo@counterprefix{}%
5091 \else
5092 \def\@gls@get@counterprefix##1.#1##2\end@getprefix{%
5093 \def\@glo@tmp{##2}%
5094 \ifx\@glo@tmp\@empty
```

```

5095     \def\@glo@counterprefix{}%
5096     \else
5097     \def\@glo@counterprefix{##1}%
5098     \fi
5099     }%
5100     \@gls@get@counterprefix#2.#1\end@getprefix

```

Warn if no prefix can be formed.

```

5101     \ifx\@glo@counterprefix\@empty
5102     \GlossariesWarning{Hyper target ‘#2’ can’t be formed by
5103     prefixing location ‘#1’. You need to modify the
5104     definition of \string\theH\@gls@counter otherwise you
5105     will get the warning: “name{\@gls@counter.#1}’ has been
5106     referenced but does not exist”}%
5107     \fi
5108     \fi
5109 }

```

1.15 Glossary Entry Cross-References

`\do@seeglossary` Write the glossary entry with a cross reference. The first argument is the entry’s label, the second must be in the form [*<tag>*]{*<list>*}, where *<tag>* is a tag such as “see” and *<list>* is a list of labels.

```

5110 \newcommand{\do@seeglossary}[2]{%
5111 \def\@gls@xref{#2}%
5112 \@onelevel@sanitize\@gls@xref
5113 \@gls@checkmkidxchars\@gls@xref
5114 \ifglsxindy
5115   \gls@glossary{\csname glo@#1@type\endcsname}{%
5116     (indexentry
5117       :tkey (\csname glo@#1@index\endcsname)
5118       :xref (\string"\@gls@xref\string")
5119       :attr \string"see\string"
5120     )
5121   }%
5122 \else
5123   \gls@glossary{\csname glo@#1@type\endcsname}{%
5124     \string\glossaryentry{\csname glo@#1@index\endcsname
5125     \@gls@encapchar glsseeformat\@gls@xref}{Z}}%
5126 \fi
5127 }

```

`\@gls@fixbraces` If no optional argument is specified, list needs to be enclosed in a set of braces.

```

5128 \def\@gls@fixbraces#1#2#3\@nil{%
5129   \ifx#2[\relax
5130     \@gls@fixbraces#1#2#3\@end@fixbraces
5131   \else
5132     \def#1{{#2#3}}%
5133   \fi

```


5134 }

`\@gls@fixbraces`

```
5135 \def\@gls@fixbraces#1[#2]#3\@end@fixbraces{%  
5136   \def#1{[#2]{#3}}%  
5137 }
```

`\glssee` `\glssee{<label>}{<cross-ref list>}`

```
5138 \DeclareRobustCommand*\glssee}[3][\seename]{%  
5139   \@do@seeglossary{#2}{#1}{#3}}  
5140 \newcommand*\@glssee}[3][\seename]{%  
5141   \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.

```
5142 \DeclareRobustCommand*\glsseeformat}[3][\seename]{%  
5143   \emph{#1} \glsseelist{#2}}
```

`\glsseelist` `\glsseelist{<list>}` formats list of entry labels.

```
5144 \DeclareRobustCommand*\glsseelist}[1]{%
```

If there is only one item in the list, set the last separator to do nothing.

```
5145   \let\@gls@dolast\relax
```

Don't display separator on the first iteration of the loop

```
5146   \let\@gls@donext\relax
```

Iterate through the labels

```
5147   \@for\@gls@thislabel:=#1\do{%
```

Check if on last iteration of loop

```
5148     \ifx\@xfor@nextelement\@nnil
```

```
5149       \@gls@dolast
```

```
5150     \else
```

```
5151       \@gls@donext
```

```
5152     \fi
```

Display the entry for this label. (Expanding label as it's a temporary control sequence that's used elsewhere.)

```
5153     \expandafter\glsseeitem\expandafter{\@gls@thislabel}%
```

Update separators

```
5154     \let\@gls@dolast\glsseelastsep
```

```
5155     \let\@gls@donext\glsseesep
```

```
5156   }%
```

```
5157 }
```

`\glsseelastsep` Separator to use between penultimate and ultimate entries in a cross-referencing list.

```
5158 \newcommand*\glsseelastsep}{\space\andname\space}
```

`\glsseesep` Separator to use between entires in a cross-referencing list.

```
5159 \newcommand*\glsseesep}{, }
```

`\glsseeitem` `\glsseeitem{<label>}` formats individual entry in a cross-referencing list.

```
5160 \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.)

```
5161 \newcommand*\glsseeitemformat}[1]{\glsentrytext{#1}}
```

1.16 Displaying the glossary

An individual glossary is displayed in the text using `\printglossary[<key-val list>]`. If the type key is omitted, the default glossary is displayed. The optional argument can be used to specify an alternative glossary, and can also be used to set the style, title and entry in the table of contents. Available keys are defined below.

`\gls@save@numberlist` Provide command to store number list.

```
5162 \newcommand*\gls@save@numberlist}[1]{%
5163   \ifglssavenumberlist
5164     \toks@{#1}%
5165     \edef\@do@writeaux@info{%
5166       \noexpand\csgdef{glo@\glscurrententrylabel @numberlist}{\the\toks@}%
5167     }%
5168     \@onelevel@sanitize\@do@writeaux@info
5169     \protected@write\@auxout{}\@do@writeaux@info%
5170   \fi
5171 }
```

`\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.)

```
5172 \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.

```
5173 \ifcsundef{printglossary}{}%
5174 {%
```

If `\printglossary` is already defined, issue a warning and undefine it.

```
5175   \@gls@warnonglossdefined
5176   \undef\printglossary
5177 }
```

`\printglossary` has an optional argument. The default value is to set the glossary type to the main glossary.

```
5178 \newcommand*\printglossary}[1][type=\glsdefaulttype]{%
5179   \@printglossary{#1}{\@print@glossary}}%
5180 }
```

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`

```
5181 \newcommand*\printglossaries{%
5182   \forallglossaries{\@glo@type}{\printglossary[type=\@glo@type]}%
5183 }
```

`\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.

```
5184 \newcommand*\printnoidxglossary}[1][type=\glsdefaulttype]{%
5185   \@printglossary{#1}{\@print@noidx@glossary}}%
5186 }
```

`\printnoidxglossaries` Analogous to `\printglossaries`

```
5187 \newcommand*\printnoidxglossaries{%
5188   \forallglossaries{\@glo@type}{\printnoidxglossary[type=\@glo@type]}%
5189 }
```

`\@printgloss@setsort` Initialise to do nothing.

```
5190 \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.

```
5191 \newcommand{\@printglossary}[2]{%
  Set up defaults.
5192   \def\@glo@type{\glsdefaulttype}%
5193   \def\glossarytitle{\csname @glo@type @title\endcsname}%
5194   \def\glossarytoctitle{\glossarytitle}%
5195   \let\org@glossarytitle\glossarytitle
5196   \def\@glossarystyle{}%
5197   \def\gls@dotoc@title{\glssettoctitle{\@glo@type}}%
```

Store current value of `\glossaryentrynumbers`. (This may be changed via the optional argument)

```
5198 \let\org@glossaryentrynumbers\glossaryentrynumbers
```

Localise the effects of the optional argument

```
5199 \bgroup
```

Activate or deactivate sort key:

```
5200 \@printgloss@setsort
```

Determine settings specified in the optional argument.

```
5201 \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)

```
5202 \ifx\glossarytitle\org@glossarytitle
5203 \else
5204 \expandafter\let\csname @glo@type @title\endcsname
5205 \glossarytitle
5206 \fi
```

Allow a high-level user command to indicate the current glossary

```
5207 \let\currentglossary\@glo@type
```

Enable individual number lists to be suppressed.

```
5208 \let\org@glossaryentrynumbers\glossaryentrynumbers
5209 \let\glsnonextpages\@glsnonextpages
```

Enable individual number list to be activated:

```
5210 \let\glsnextpages\@glsnextpages
```

Enable suppression of description terminators.

```
5211 \let\nopostdesc\@nopostdesc
```

Set up the entry for the TOC

```
5212 \gls@dotoc@title
```

Set the glossary style

```
5213 \@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):

```
5214 \let\gls@org@glossaryentryfield\glossentry
5215 \let\gls@org@glossarysubentryfield\subglossentry
5216 \renewcommand{\glossentry}[1]{%
5217 \xdef\glscurrententrylabel{\glsdetoklabel{##1}}%
5218 \gls@org@glossaryentryfield{##1}%
5219 }%
5220 \renewcommand{\subglossentry}[2]{%
5221 \xdef\glscurrententrylabel{\glsdetoklabel{##2}}%
5222 \gls@org@glossarysubentryfield{##1}{##2}%
5223 }%
```

Now do the handler macro that deals with the actual glossary:

```
5224   #2%
      End the current scope
5225   \egroup
      Reset \glossaryentrynumbers
5226   \global\let\glossaryentrynumbers\@org@glossaryentrynumbers
      Suppress warning about no \printglossary
5227   \global\let\warn@noprintglossary\relax
5228 }
```

\@print@glossary Internal workings of \printglossary dealing with reading the external file.

```
5229 \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.)

```
5230   \makeatletter
```

Input the glossary file, if it exists.

```
5231   \@input@{\jobname.\csname @glo@type@\@glo@type @in\endcsname}%
```

If the glossary file doesn't exist, do \null. (This ensures that the page is shipped out and all write commands are done.) This might produce an empty page, but at this point the document isn't complete, so it shouldn't matter.

```
5232   \IfFileExists{\jobname.\csname @glo@type@\@glo@type @in\endcsname}%
```

```
5233   {}%
```

```
5234   {\null}%
```

If xindy is being used, need to write the language dependent information to the .aux file for makeglossaries.

```
5235   \ifglxindy
```

```
5236     \ifcsundef{@xdy@\@glo@type @language}%
```

```
5237     {%
```

```
5238       \edef\@do@auxoutstuff{%
```

```
5239         \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.

```
5240           \noexpand\immediate\noexpand\write\@auxout{%
```

```
5241             \string\providecommand\string\@xdy@language[2]{}%}
```

```
5242           \noexpand\immediate\noexpand\write\@auxout{%
```

```
5243             \string\@xdy@language{\@glo@type}{\@xdy@main@language}}%
```

```
5244           }%
```

```
5245         }%
```

```
5246       }%
```

```
5247     {%
```

```
5248       \edef\@do@auxoutstuff{%
```

```

5249     \noexpand\AtEndDocument{%
5250         \noexpand\immediate\noexpand\write\@auxout{%
5251             \string\providecommand\string\xdylanguage[2]{}%
5252         \noexpand\immediate\noexpand\write\@auxout{%
5253             \string\xdylanguage{\@glo@type}{\csname @xdy@\@glo@type
5254                 @language\endcsname}}%
5255     }%
5256 }%
5257 }%
5258 \do@auxoutstuff
5259 \edef\do@auxoutstuff{%
5260     \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.

```

5261     \noexpand\immediate\noexpand\write\@auxout{%
5262         \string\providecommand\string\@gls@codepage[2]{}%
5263     \noexpand\immediate\noexpand\write\@auxout{%
5264         \string\@gls@codepage{\@glo@type}{\gls@codepage}}%
5265     }%
5266 }%
5267 \do@auxoutstuff
5268 \fi

```

Activate warning if `\makeglossaries` hasn't been used.

```

5269 \renewcommand*\@warn@nomakeglossaries{%
5270     \GlossariesWarningNoLine{\string\makeglossaries\space
5271     hasn't been used,^^Jthe glossaries will not be updated}%
5272 }%
5273 }

```

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`

```

5274 \newcommand*\@glo@sortentries[2]{%
5275     \def\@glo@sortinglist{}%
5276     \def\@glo@sortinghandler{#1}%
5277     \edef\@glo@type{#2}%
5278     \forlistcslloop{\@glo@do@sortentries}{\@glsref@#2}%
5279     \csdef{\@glsref@#2}{}%
5280     \@for\@this@label:=\@glo@sortinglist\do{%

```

Has this entry already been added?

```
5281 \xifinlistcs{\@this@label}{@glsref@#2}%
5282 {}%
5283 {%
5284 \listcsxadd{@glsref@#2}{\@this@label}%
5285 }%
5286 \ifcsdef{@glo@sortingchildren@\@this@label}%
5287 {%
5288 \@glo@addchildren{#2}{\@this@label}%
5289 }%
5290 {}%
5291 }%
5292 }
```

`\@glo@addchildren` `\@glo@addchildren{<type>}{<parent>}`

```
5293 \newcommand*{\@glo@addchildren}[2]{%
```

Scope to allow nesting.

```
5294 \bgroup
5295 \letcs{\@glo@childlist}{@glo@sortingchildren@#2}%
5296 \@for\@this@childlabel:=\@glo@childlist\do
5297 {%
```

Check this label hasn't already been added.

```
5298 \xifinlistcs{\@this@childlabel}{@glsref@#1}%
5299 {}%
5300 {%
5301 \listcsxadd{@glsref@#1}{\@this@childlabel}%
5302 }%
```

Does this child have children?

```
5303 \ifcsdef{@glo@sortingchildren@\@this@childlabel}%
5304 {%
5305 \@glo@addchildren{#1}{\@this@childlabel}%
5306 }%
5307 {%
5308 }%
5309 }%
5310 \egroup
5311 }
```

`@glo@do@sortentries`

```
5312 \newcommand*{\@glo@do@sortentries}[1]{%
5313 \ifglshasparent{#1}%
5314 {%
```

This entry has a parent, so add it to the child list

```
5315 \edef\@glo@parent{\csuse{glo@glstetoklabel{#1}@parent}}%
```

```

5316 \ifcsundef{@glo@sortingchildren@\@glo@parent}%
5317 {%
5318 \csdef{@glo@sortingchildren@\@glo@parent}{}%
5319 }%
5320 {}%
5321 \expandafter\@glo@sortedinsert
5322 \csname @glo@sortingchildren@\@glo@parent\endcsname{#1}%

```

Has the parent been added?

```

5323 \xifinlistcs{\@glo@parent}{\@glsref@\@glo@type}%
5324 {%

```

Yes, it has so do nothing.

```

5325 }%
5326 {}%

```

No, it hasn't so add it now.

```

5327 \expandafter\@glo@do@sortentries\expandafter{\@glo@parent}%
5328 }%
5329 }%
5330 {}%
5331 \@glo@sortedinsert{\@glo@sortinglist}{#1}%
5332 }%
5333 }

```

```
\@glo@sortedinsert \@glo@sortedinsert{<list>}{<entry label>}
```

Insert into list.

```

5334 \newcommand*{\@glo@sortedinsert}[2]{%
5335 \dtl@insertinto{#2}{#1}{\@glo@sortinghandler}%
5336 }%

```

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`

```

5337 \newcommand*{\@glo@sorthandler@word}[2]{%
5338 \letcs\@gls@sort@A{glo\@glsdetoklabel{#1}@sort}%
5339 \letcs\@gls@sort@B{glo\@glsdetoklabel{#2}@sort}%
5340 \edef\@glo@do@compare{%
5341 \noexpand\dtlwordindexcompare{\noexpand\dtl@sortresult}%
5342 {\expandonce\@gls@sort@B}%
5343 {\expandonce\@gls@sort@A}%
5344 }%
5345 \@glo@do@compare
5346 }

```


@sorthandler@letter

```
5347 \newcommand*{\@glo@sorthandler@letter}[2]{%
5348   \letcs\@gls@sort@A{glo\glsdetoklabel{#1}@sort}%
5349   \letcs\@gls@sort@B{glo\glsdetoklabel{#2}@sort}%
5350   \edef\glo@do@compare{%
5351     \noexpand\dtlletterindexcompare{\noexpand\dtl@sortresult}%
5352     {\expandonce\@gls@sort@B}%
5353     {\expandonce\@gls@sort@A}%
5354   }%
5355   \glo@do@compare
5356 }
```

lo@sorthandler@case Case-sensitive sort.

```
5357 \newcommand*{\@glo@sorthandler@case}[2]{%
5358   \letcs\@gls@sort@A{glo\glsdetoklabel{#1}@sort}%
5359   \letcs\@gls@sort@B{glo\glsdetoklabel{#2}@sort}%
5360   \edef\glo@do@compare{%
5361     \noexpand\dtlcompare{\noexpand\dtl@sortresult}%
5362     {\expandonce\@gls@sort@B}%
5363     {\expandonce\@gls@sort@A}%
5364   }%
5365   \glo@do@compare
5366 }
```

@sorthandler@nocase Case-insensitive sort.

```
5367 \newcommand*{\@glo@sorthandler@nocase}[2]{%
5368   \letcs\@gls@sort@A{glo\glsdetoklabel{#1}@sort}%
5369   \letcs\@gls@sort@B{glo\glsdetoklabel{#2}@sort}%
5370   \edef\glo@do@compare{%
5371     \noexpand\dtlicompare{\noexpand\dtl@sortresult}%
5372     {\expandonce\@gls@sort@B}%
5373     {\expandonce\@gls@sort@A}%
5374   }%
5375   \glo@do@compare
5376 }
```

@glo@sortmacro@word Sort macro for ‘word’

```
5377 \newcommand*{\@glo@sortmacro@word}[1]{%
5378   \ifdefstring{\@glo@default@sorttype}{standard}%
5379   {%
5380     \@glo@sortentries{\@glo@sorthandler@word}{#1}%
5381   }%
5382   {%
5383     \PackageError{glossaries}{Conflicting sort options:^^J
5384       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5385       \string\printnoidxglossary[sort=word]}{}%
5386   }%
5387 }
```

lo@sortmacro@letter Sort macro for 'letter'

```
5388 \newcommand*\@glo@sortmacro@letter}[1]{%
5389 \ifdefstring{\@glo@default@sorttype}{standard}%
5390 {%
5391 \@glo@sortentries{\@glo@sorthandler@letter}{#1}%
5392 }%
5393 {%
5394 \PackageError{glossaries}{Conflicting sort options:^^J
5395 \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5396 \string\printnoidxglossary[sort=letter]}{}}%
5397 }%
5398 }
```

@sortmacro@standard Sort macro for 'standard'. (Use either 'word' or 'letter' order.)

```
5399 \newcommand*\@glo@sortmacro@standard}[1]{%
5400 \ifdefstring{\@glo@default@sorttype}{standard}%
5401 {%
5402 \ifcsdef{@glo@sorthandler@\glsorder}%
5403 {%
5404 \@glo@sortentries{\csuse{@glo@sorthandler@\glsorder}}{#1}%
5405 }%
5406 {%
5407 \PackageError{glossaries}{Unknown sort handler '\glsorder'}{}}%
5408 }%
5409 }%
5410 {%
5411 \PackageError{glossaries}{Conflicting sort options:^^J
5412 \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5413 \string\printnoidxglossary[sort=standard]}{}}%
5414 }%
5415 }
```

@glo@sortmacro@case Sort macro for 'case'

```
5416 \newcommand*\@glo@sortmacro@case}[1]{%
5417 \ifdefstring{\@glo@default@sorttype}{standard}%
5418 {%
5419 \@glo@sortentries{\@glo@sorthandler@case}{#1}%
5420 }%
5421 {%
5422 \PackageError{glossaries}{Conflicting sort options:^^J
5423 \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5424 \string\printnoidxglossary[sort=case]}{}}%
5425 }%
5426 }
```

lo@sortmacro@nocase Sort macro for 'nocase'

```
5427 \newcommand*\@glo@sortmacro@nocase}[1]{%
5428 \ifdefstring{\@glo@default@sorttype}{standard}%
5429 {%
```

```

5430   \@glo@sortentries{\@glo@sorthandler@nocase}{#1}%
5431 }%
5432 {%
5433   \PackageError{glossaries}{Conflicting sort options:^^J
5434   \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5435   \string\printnoidxglossary[sort=nocase]}{}%
5436 }%
5437 }

```

`\@glo@sortmacro@def` Sort macro for ‘def’. The order of definition is given in `\glo@list@<type>`.

```

5438 \newcommand*\@glo@sortmacro@def}[1]{%
5439   \def\@glo@sortinglist{}%
5440   \for@sentries[#1]{\@gls@thislabel}%
5441   {%
5442     \xifinlistcs{\@gls@thislabel}{\@glsref@#1}%
5443     {%
5444       \listead{\@glo@sortinglist}{\@gls@thislabel}%
5445     }%
5446     {%
5447       }%
5448     }%
5449     \cslet{\@glsref@#1}{\@glo@sortinglist}%
5450 }

```

Hasn't been referenced.

`\@glo@sortmacro@def@do` This won't include parent entries that haven't been referenced.

```

5451 \newcommand*\@glo@sortmacro@def@do}[1]{%
5452   \ifinlistcs{#1}{\@glsref@\@glo@type}%
5453   {}%
5454   {%
5455     \listcsadd{\@glsref@\@glo@type}{#1}%
5456   }%
5457   \ifcsdef{\@glo@sortingchildren@#1}%
5458   {%
5459     \@glo@addchildren{\@glo@type}{#1}%
5460   }%
5461   {}%
5462 }

```

`\@glo@sortmacro@use` Sort macro for ‘use’. (No sorting is required, as the entries are already in order of use, so do nothing.)

```

5463 \newcommand*\@glo@sortmacro@use}[1]{%

```

`\printnoidx@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.

```
5464 \newcommand*{\@print@noidx@glossary}{%
5465   \ifcsdef{@glsref@\@glo@type}%
5466   {%
```

Sort the entries:

```
5467   \ifcsdef{@glo@sortmacro@\@glo@sorttype}%
5468   {%
5469     \csuse{@glo@sortmacro@\@glo@sorttype}{\@glo@type}%
5470   }%
5471   {%
5472     \PackageError{glossaries}{Unknown sort handler ‘\@glo@sorttype’}{}%
5473   }%
```

Do the glossary heading and preamble

```
5474   \glossarysection[\glossarytoctitle]{\glossarytitle}%
5475   \glossarypreamble
5476   \begin{theglossary}%
5477   \glossaryheader
5478   \glsresetentrylist
5479   \def\@gls@currentlettergroup{}%
```

Iterate through the entries.

```
5480   \forlistcsloop{\@gls@noidx@do}{@glsref@\@glo@type}%
```

Finally end the glossary and do the postamble:

```
5481   \end{theglossary}%
5482   \glossarypostamble
5483 }%
5484 {%
5485   \@gls@noref@warn{\@glo@type}%
5486 }%
5487 }
```

\glo@grabfirst

```
5488 \def\glo@grabfirst#1#2\@nil{%
5489   \def\@gls@firsttok{#1}%
5490   \ifdefempty\@gls@firsttok
5491   {%
5492     \def\@glo@thislettergrp{0}%
5493   }%
5494   {%
```

Sanitize it:

```
5495   \@onelevel@sanitize\@gls@firsttok
```

Fetch the first letter:

```
5496   \expandafter\@glo@grabfirst\@gls@firsttok{}{}\@nil
5497 }%
5498 }
```

`\@glo@grabfirst`

```
5499 \def\@glo@grabfirst#1#2\@nil{%
5500   \ifdefempty\@glo@thislettergrp
5501   {%
5502     \def\@glo@thislettergrp{glssymbols}%
5503   }%
5504   {%
5505     \count@=\ucode‘#1\relax
5506     \ifnum\count@=0\relax
5507       \def\@glo@thislettergrp{glssymbols}%
5508     \else
5509       \ifdefstring\@glo@sorttype{case}%
5510       {%
5511         \count@=‘#1\relax
5512       }%
5513     }%
5514   }%
5515   \edef\@glo@thislettergrp{\the\count@}%
5516   \fi
5517 }%
5518 }
```

`\@gls@noidx@do` Handler for list iteration used by `\@print@noidx@glossary`. The argument is the entry label. This only allows one sublevel.

```
5519 \newcommand{\@gls@noidx@do}[1]{%
```

Get this entry's location list

```
5520   \global\letcs{\@gls@loclist}{glo@glstdetoklabel{#1}@loclist}%
```

Does this entry have a parent?

```
5521   \ifglshasparent{#1}%
5522   {%
```

Has a parent.

```
5523     \gls@level=\csuse{glo@glstdetoklabel{#1}@level}\relax
5524     \ifdefvoid{\@gls@loclist}
5525     {%
5526       \subglossentry{\gls@level}{#1}{}%
5527     }%
5528     {%
5529       \subglossentry{\gls@level}{#1}%
5530     }%
5531     \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
5532     }%
5533   }%
5534 }%
5535   {%
```

Doesn't have a parent Get this entry's sort key

```
5536   \letcs{\@gls@sort}{glo@glstdetoklabel{#1}@sort}%
```

Fetch the first letter:

```
5537 \expandafter\glo@grabfirst\@gls@sort{}{} \@nil
5538 \ifdefequal{\@glo@thislettergrp}{\@gls@currentlettergroup}%
5539 {}%
5540 {}%
```

Do the group header:

```
5541 \ifdefempty{\@gls@currentlettergroup}{\@gls@groupskip}%
5542 \gls@groupheading{\@glo@thislettergrp}%
5543 }%
5544 \let\@gls@currentlettergroup\@glo@thislettergrp
```

Do this entry:

```
5545 \ifdefvoid{\@gls@loclist}
5546 {}%
5547 \glossentry{#1}{}%
5548 }%
5549 {}%
5550 \glossentry{#1}%
5551 {}%
5552 \glossaryentrynumbers{\@gls@noidxloclist{\@gls@loclist}}%
5553 }%
5554 }%
5555 }%
5556 }
```

`\glsnoidxloclist` `\glsnoidxloclist{<list cs>}`

Display location list.

```
5557 \newcommand*{\glsnoidxloclist}[1]{}%
5558 \def\@gls@noidxloclist@sep{}%
5559 \def\@gls@noidxloclist@prev{}%
5560 \forlistloop{\glsnoidxloclisthandler}{#1}%
5561 }
```

`noidxloclisthandler` Handler for location list iterator.

```
5562 \newcommand*{\glsnoidxloclisthandler}[1]{}%
5563 \ifdefstring{\@gls@noidxloclist@prev}{#1}%
5564 {}%
```

Same as previous location so skip.

```
5565 }%
5566 {}%
5567 \@gls@noidxloclist@sep
5568 #1%
5569 \def\@gls@noidxloclist@sep{\delimN}%
5570 \def\@gls@noidxloclist@prev{#1}%
5571 }%
5572 }
```

`\displayloclisthandler` Handler for location list iterator when used with `\glsdisplaynumberlist`.

```
5573 \newcommand*\glsnoidxdisplayloclisthandler[1]{%
5574   \ifdefstring{\@gls@noidxloclist@prev}{#1}%
5575   {%
      Same as previous location so skip.
5576   }%
5577   {%
5578     \@gls@noidxloclist@sep
5579     \@gls@noidxloclist@prev
5580     \def\@gls@noidxloclist@prev{#1}%
5581   }%
5582 }
```

`\glsnoidxdisplayloc` `\glsnoidxdisplayloc{<prefix>}{<counter>}{<format>}{<location>}`

Display a location in the location list.

```
5583 \newcommand*\glsnoidxdisplayloc[4]{%
5584   \setentrycounter[#1]{#2}%
5585   \csuse{#3}{#4}%
5586 }
```

`\@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.

```
5587 \newcommand*\@gls@reference[3]{%
      Add to label list
5588   \glsdoifexistsorwarn{#2}%
5589   {%
5590     \ifcsundef{@glsref@#1}{\csgdef{@glsref@#1}{}}{}%
5591     \ifinlistcs{#2}{@glsref@#1}%
5592     {}%
5593     {\listcsgadd{@glsref@#1}{#2}}%
      Add to location list
5594     \ifcsundef{glo@glstdetoklabel{#2}@loclist}%
5595     {\csgdef{glo@glstdetoklabel{#2}@loclist}{}}%
5596     {}%
5597     \listcsgadd{glo@glstdetoklabel{#2}@loclist}{#3}%
5598   }%
5599 }
```

The keys that can be used in the optional argument to `\printglossary` or `\printnoidxglossary` are as follows: The type key sets the glossary type.

```
5600 \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`.

```
5601 \define@key{printgloss}{title}{%
5602 \def\glossarytitle{#1}%
5603 \let\gls@dotoc\relax
5604 }
```

The toctitle sets the text used for the relevant entry in the table of contents.

```
5605 \define@key{printgloss}{toctitle}{%
5606 \def\glossarytoctitle{#1}%
5607 \let\gls@dotoc\relax
5608 }
```

The style key sets the glossary style (but only for the given glossary).

```
5609 \define@key{printgloss}{style}{%
5610 \ifcsundef{@glsstyle@#1}%
5611 {%
5612 \PackageError{glossaries}%
5613 {Glossary style ‘#1’ undefined}{}%
5614 }%
5615 {%
5616 \def\@glossarystyle{\setglossentrycompatibility
5617 \csname @glsstyle@#1\endcsname}%
5618 }%
5619 }
```

The numberedsection key determines if this glossary should be in a numbered section.

```
5620 \define@choicekey{printgloss}{numberedsection}[\val\nr]{%
5621 false,nolabel,autolabel,nameref}[nolabel]{%
5622 \ifcase\nr\relax
5623 \renewcommand*{\@glossarysecstar}{*}%
5624 \renewcommand*{\@glossaryseclabel}{}%
5625 \or
5626 \renewcommand*{\@glossarysecstar}{}%
5627 \renewcommand*{\@glossaryseclabel}{}%
5628 \or
5629 \renewcommand*{\@glossarysecstar}{}%
5630 \renewcommand*{\@glossaryseclabel}{\label{\glsautoprefix\@glo@type}}%
5631 \or
5632 \renewcommand*{\@glossarysecstar}{*}%
5633 \renewcommand*{\@glossaryseclabel}{%
5634 \protected@edef\@currentlabelname{\glossarytoctitle}%
5635 \label{\glsautoprefix\@glo@type}}%
5636 \fi
5637 }
```

The nogroupskip key determines whether or not there should be a vertical gap between glossary groups.

```
5638 \define@choicekey{printgloss}{nogroupskip}{true,false}[true]{%
```



```
5639 \csuse{glsnogroupskip#1}%
5640 }
```

The nopostdot key has the same effect as the package option of the same name.

```
5641 \define@choicekey{printgloss}{nopostdot}{true,false}[true]{%
5642 \csuse{glsnopostdot#1}%
5643 }
```

The entrycounter key is the same as the package option but localised to the current glossary.

```
5644 \define@choicekey{printgloss}{entrycounter}{true,false}[true]{%
5645 \csuse{glsentrycounter#1}%
5646 \ifglsentrycounter
5647 \ifx\@gls@counterwithin\@empty
5648 \newcounter{glossaryentry}%
5649 \else
5650 \newcounter{glossaryentry}[\@gls@counterwithin]%
5651 \fi
5652 \def\theHglossaryentry{\currentglossary.\theglossaryentry}%
5653 \renewcommand*{\glsresetentrycounter}{%
5654 \setcounter{glossaryentry}{0}%
5655 }%
5656 \renewcommand*{\glsstepentry}[1]{%
5657 \refstepcounter{glossaryentry}%
5658 \label{glsentry-\glsdetoklabel{##1}}%
5659 }%
5660 \renewcommand*{\glsentrycounterlabel}{\theglossaryentry.\space}%
5661 \renewcommand*{\glsentryitem}[1]{%
5662 \glsstepentry{##1}\glsentrycounterlabel
5663 }%
5664 \else
5665 \renewcommand*{\glsresetentrycounter}{}%
5666 \renewcommand*{\glsstepentry}[1]{%
5667 \renewcommand*{\glsentrycounterlabel}{}%
5668 \renewcommand*{\glsentryitem}[1]{\glsresetsubentrycounter}
5669 \fi
5670 }
```

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.

```
5671 \define@choicekey{printgloss}{subentrycounter}{true,false}[true]{%
5672 \csuse{glssubentrycounter#1}%
5673 \ifglssubentrycounter
5674 \ifundef\c@glossarysubentry
5675 {%
5676 \ifglsentrycounter
5677 \newcounter{glossarysubentry}[glossaryentry]%

```

```

5678     \else
5679         \newcounter{glossarysubentry}
5680     \fi
5681 }{}%
5682 \renewcommand*{\glsstepsubentry}[1]{%
5683     \edef\currentglssubentry{\glsdetoklabel{##1}}%
5684     \refstepcounter{glossarysubentry}%
5685     \label{glsentry-\currentglssubentry}%
5686 }%
5687 \renewcommand*{\glsresetsubentrycounter}{%
5688     \setcounter{glossarysubentry}{0}%
5689 }%
5690 \renewcommand*{\glssubentryitem}[1]{%
5691     \glsstepsubentry{##1}\glssubentrycounterlabel
5692 }%
5693 \renewcommand*{\glssubentrycounterlabel}{\theglossarysubentry}\space}%
5694 \def\theHglossarysubentry{\currentglssubentry.\theglossarysubentry}
5695 \else
5696     \renewcommand*{\glssubentryitem}[1]{}%
5697     \renewcommand*{\glsstepsubentry}[1]{}%
5698     \renewcommand*{\glsresetsubentrycounter}{}%
5699     \renewcommand*{\glssubentrycounterlabel}{}%
5700 \fi
5701 }

```

The nonumberlist key determines if this glossary should have a number list.

```

5702 \define@boolkey{printgloss}[gls]{nonumberlist}[true]{}%
5703 \ifglsnonumberlist
5704     \def\glossaryentrynumbers##1{}%
5705 \else
5706     \def\glossaryentrynumbers##1{##1}%
5707 \fi}

```

The sort key sets the glossary sort handler (`\printnoidxglossary` only).

```

5708 \define@key{printgloss}{sort}{\@glo@assign@sortkey{#1}}

```

`\@glo@no@assign@sortkey` Issue error if used with `\printglossary`

```

5709 \newcommand*{\@glo@no@assign@sortkey}[1]{%
5710     \PackageError{glossaries}{‘sort’ key not permitted with
5711     \string\printglossary}%
5712     {The ‘sort’ key may only be used with \string\printnoidxglossary}%
5713 }

```

`\@glo@assign@sortkey` For use with `\printnoidxglossary`

```

5714 \newcommand*{\@glo@assign@sortkey}[1]{%
5715     \def\@glo@sorttype{#1}%
5716 }

```

`\@glsnonextpages` Suppresses the next number list only. Global assignments required as it may not occur in the same level of grouping as the next numberlist. (For example, if

`\glsnonextpages` is 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.

```
5717 \newcommand*\@glsnonextpages}{%
5718   \gdef\glossaryentrynumbers##1{%
5719     \glsresetentrylist
5720   }%
5721 }
```

`\@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.

```
5722 \newcommand*\@glsnextpages}{%
5723   \gdef\glossaryentrynumbers##1{%
5724     ##1\glsresetentrylist}}
```

`\glsresetentrylist` Resets `\glossaryentrynumbers`

```
5725 \newcommand*\glsresetentrylist}{%
5726   \global\let\glossaryentrynumbers\org@glossaryentrynumbers}
```

`\glsnonextpages` Outside of `\printglossary` this does nothing.

```
5727 \newcommand*\glsnonextpages}{}
```

`\glsnextpages` Outside of `\printglossary` this does nothing.

```
5728 \newcommand*\glsnextpages}{}
```

`glossaryentry` If the `entrycounter` package option has been used, define a counter to number each level 0 entry.

```
5729 \ifglentrycounter
5730   \ifx\@gls@counterwithin\@empty
5731     \newcounter{glossaryentry}
5732   \else
5733     \newcounter{glossaryentry}[\@gls@counterwithin]
5734   \fi
5735   \def\theHglossaryentry{\currentglossary.\theglossaryentry}
5736 \fi
```

`glossarysubentry` If the `subentrycounter` package option has been used, define a counter to number each level 1 entry.

```
5737 \ifglsubentrycounter
5738   \ifglentrycounter
5739     \newcounter{glossarysubentry}[glossaryentry]
5740   \else
5741     \newcounter{glossarysubentry}
5742   \fi
```

```

5743 \def\theHglossarysubentry{\currentglssubentry.\theglossarysubentry}
5744 \fi

```

`\glsresetsubentrycounter` Resets the glossarysubentry counter.

```

5745 \ifglssubentrycounter
5746 \newcommand*\glsresetsubentrycounter{%
5747 \setcounter{glossarysubentry}{0}%
5748 }
5749 \else
5750 \newcommand*\glsresetsubentrycounter{}
5751 \fi

```

`\glsresetentrycounter` Resets the glossentry counter.

```

5752 \ifglentrycounter
5753 \newcommand*\glsresetentrycounter{%
5754 \setcounter{glossaryentry}{0}%
5755 }
5756 \else
5757 \newcommand*\glsresetentrycounter{}
5758 \fi

```

`\glsstepentry` Advance the glossaryentry counter if in use. The argument is the label associated with the entry.

```

5759 \ifglentrycounter
5760 \newcommand*\glsstepentry}[1]{%
5761 \refstepcounter{glossaryentry}%
5762 \label{glentry-\glsdetoklabel{#1}}%
5763 }
5764 \else
5765 \newcommand*\glsstepentry}[1]{}
5766 \fi

```

`\glsstepsubentry` Advance the glossarysubentry counter if in use. The argument is the label associated with the subentry.

```

5767 \ifglssubentrycounter
5768 \newcommand*\glsstepsubentry}[1]{%
5769 \edef\currentglssubentry{\glsdetoklabel{#1}}%
5770 \refstepcounter{glossarysubentry}%
5771 \label{glentry-\currentglssubentry}%
5772 }
5773 \else
5774 \newcommand*\glsstepsubentry}[1]{}
5775 \fi

```

`\glsrefentry` Reference the entry or sub-entry counter if in use, otherwise just do `\gls`.

```

5776 \ifglentrycounter
5777 \newcommand*\glsrefentry}[1]{\ref{glentry-\glsdetoklabel{#1}}}
5778 \else

```

```

5779 \ifglssubentrycounter
5780   \newcommand*{\glsrefentry}[1]{\ref{glsentry-\glsdetoklabel{#1}}}
5781 \else
5782   \newcommand*{\glsrefentry}[1]{\gls{#1}}
5783 \fi
5784 \fi

```

`glsentrycounterlabel` Defines how to display the glossaryentry counter.

```

5785 \ifglssentrycounter
5786   \newcommand*{\glsentrycounterlabel}{\theglossaryentry.\space}
5787 \else
5788   \newcommand*{\glsentrycounterlabel}{}
5789 \fi

```

`glssubentrycounterlabel` Defines how to display the glossarysubentry counter.

```

5790 \ifglssubentrycounter
5791   \newcommand*{\glssubentrycounterlabel}{\theglossarysubentry)\space}
5792 \else
5793   \newcommand*{\glssubentrycounterlabel}{}
5794 \fi

```

`glsentryitem` Step and display glossaryentry counter, if appropriate.

```

5795 \ifglssentrycounter
5796   \newcommand*{\glsentryitem}[1]{%
5797     \glsstepentry{#1}\glsentrycounterlabel
5798   }
5799 \else
5800   \newcommand*{\glsentryitem}[1]{\glsresetsubentrycounter}
5801 \fi

```

`glssubentryitem` Step and display glossarysubentry counter, if appropriate.

```

5802 \ifglssubentrycounter
5803   \newcommand*{\glssubentryitem}[1]{%
5804     \glsstepsubentry{#1}\glssubentrycounterlabel
5805   }
5806 \else
5807   \newcommand*{\glssubentryitem}[1]{}
5808 \fi

```

`theglossary` If the `theglossary` environment has already been defined, a warning will be issued. This environment should be redefined by glossary styles.

```

5809 \ifcsundef{theglossary}%
5810 {%
5811   \newenvironment{theglossary}{}{}%
5812 }%
5813 {%
5814   \@gls@warnontheglossdefined
5815   \renewenvironment{theglossary}{}{}%
5816 }

```

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`

```
5817 \newcommand*\glossaryheader{}
```

`\glstarget` `\glstarget{<label>}{<name>}`

Provide user interface to `\@glstarget` to make it easier to modify the glossary style in the document.

```
5818 \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>}
```

```
5819 \providecommand*\compatibleglossentry}[2]{%
5820   \toks@{#2}%
5821   \protected@edef\@do@glossentry{\noexpand\glossaryentryfield{#1}%
5822     {\noexpand\glsnamefont
5823       {\expandafter\expandonce\csname glo@#1@name\endcsname}}%
5824     {\expandafter\expandonce\csname glo@#1@desc\endcsname}%
5825     {\expandafter\expandonce\csname glo@#1@symbol\endcsname}%
5826     {\the\toks@}}%
5827   }%
5828   \@do@glossentry
5829 }
```

`\glossentryname`

```
5830 \newcommand*\glossentryname}[1]{%
5831   \glsdoifexistsorwarn{#1}%
5832   {%
5833     \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
5834     \expandafter\glsnamefont\expandafter{\glo@name}%
5835   }%
5836 }
```

`\Glossentryname`

```
5837 \newcommand*\Glossentryname}[1]{%
```

```

5838 \glsdoifexistsorwarn{#1}%
5839 {%
5840     \glsnamefont{\Glsentryname{#1}}%
5841 }%
5842 }

```

`\glossentrydesc`

```

5843 \newcommand*{\glossentrydesc}[1]{%
5844     \glsdoifexistsorwarn{#1}%
5845     {%
5846         \glsentrydesc{#1}%
5847     }%
5848 }

```

`\Glossentrydesc`

```

5849 \newcommand*{\Glossentrydesc}[1]{%
5850     \glsdoifexistsorwarn{#1}%
5851     {%
5852         \Glsentrydesc{#1}%
5853     }%
5854 }

```

`\glossentrysymbol`

```

5855 \newcommand*{\glossentrysymbol}[1]{%
5856     \glsdoifexistsorwarn{#1}%
5857     {%
5858         \glsentrysymbol{#1}%
5859     }%
5860 }

```

`\Glossentrysymbol`

```

5861 \newcommand*{\Glossentrysymbol}[1]{%
5862     \glsdoifexistsorwarn{#1}%
5863     {%
5864         \Glsentrysymbol{#1}%
5865     }%
5866 }

```

`compatiblesubglossentry`

```
\subglossentry{<level>}{<label>}{<page-list>}
```

```

5867 \providecommand*{\compatiblesubglossentry}[3]{%
5868     \toks@{#3}%
5869     \protected@edef\@do@subglossentry{\noexpand\glossarysubentryfield{\number#1}%
5870     {#2}%
5871     {\noexpand\glsnamefont
5872     {\expandafter\expandonce\csname glo@#2@name\endcsname}}%
5873     {\expandafter\expandonce\csname glo@#2@desc\endcsname}%

```

```

5874   {\expandafter\expandonce\csname glo@#2@symbol\endcsname}%
5875   {\the\toks@}%
5876   }%
5877   \@do@subglossentry
5878 }

```

sentrycompatibility

```

5879 \newcommand*\setglossentrycompatibility{%
5880   \let\glossentry\compatibleglossentry
5881   \let\subglossentry\compatiblesubglossentry
5882 }
5883 \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.

```

5884 \newcommand{\glossaryentryfield}[5]{%
5885   \GlossariesWarning
5886   {Deprecated use of \string\glossaryentryfield.^^J
5887     I recommend you change to \string\glossentry.^^J
5888     If you've just upgraded, try removing your gls auxiliary
5889     files^^J and recompile}%
5890   \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.)

```

5891 \newcommand*\glossarysubentryfield}[6]{%
5892   \GlossariesWarning
5893   {Deprecated use of \string\glossarysubentryfield.^^J
5894     I recommend you change to \string\subglossentry.^^J
5895     If you've just upgraded, try removing your gls auxiliary
5896     files^^J and recompile}%
5897   \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`

```
5898 \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: `glsymbols`, `glsnumbers`, `A`, ..., `Z`. Glossary styles must redefine this command. (In between groups, `\glsgroupheading` comes immediately after `\glsgroupskip`.)

`\glsgroupheading`

```
5899 \newcommand*{\glsgroupheading}[1]{}
```

It is possible to “trick” `makeindex` into treating entries as though they belong to the same group, even if the terms don’t start with the same letter, by modifying the sort key. For example, all entries belonging to one group could be defined so that the sort key starts with an `a`, while entries belonging to another group could be defined so that the sort key starts with a `b`, and so on. If you want each group to have a heading, you would then need to modify the translation control sequences `\glsgetgrouptitle` and `\glsgetgrouplabel` so that the label is translated into the required title (and vice-versa).

```
\glsgetgrouptitle{<label>}
```

This command produces the title for the glossary group whose label is given by `<label>`. By default, the group labelled `glsymbols` produces `\glsymbolsgroupname`, the group labelled `glsnumbers` produces `\glsnumbersgroupname` and all the other groups simply produce their label. As mentioned above, the group labels are: `glsymbols`, `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`

```
5900 \newcommand*{\glsgetgrouptitle}[1]{%
5901   \@gls@getgrouptitle{#1}{\@gls@grptitle}%
5902   \@gls@grptitle
5903 }
```

`\@gls@getgrouptitle` Gets the group title specified by the label (first argument) and stores in the second argument, which must be a control sequence.

```

5904 \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.
5905 \dtl@ifsingle{#1}%
5906 {%
5907   \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
5908 }%
5909 {%
5910   \ifboolexpr{test{\ifstrequal{#1}{glsymbols}}
5911               or test{\ifstrequal{#1}{glsnumbers}}}%
5912   {%
5913     \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
5914   }%
5915   {%
5916     \def#2{#1}%
5917   }%
5918 }%
5919 }

```

`@getothergrouptitle` Version for the no-indexing app option:

```

5920 \newcommand*\@gls@noidx@getgrouptitle}[2]{%
5921   \DTLifint{#1}%
5922   {\edef#2{\char#1\relax}}%
5923   {%
5924     \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
5925   }%
5926 }

```

`\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`

```

5927 \newcommand*\glsgetgrouplabel}[1]{%
5928 \ifthenelse{equal{#1}{\glsymbolsgroupname}}{glsymbols}{%
5929 \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`

```

5930 \newcommand*\setentrycounter}[2][ ]{%
5931   \def\@glo@counterprefix{#1}%
5932   \ifx\@glo@counterprefix\@empty

```

```

5933   \def\@glo@counterprefix{.}%
5934   \else
5935     \def\@glo@counterprefix{.#1.}%
5936   \fi
5937   \def\glseentrycounter{#2}%
5938 }

```

The current glossary style can be set using `\setglossarystyle{<style>}`.

`\setglossarystyle`

```

5939 \newcommand*\setglossarystyle[1]{%
5940   \ifcsundef{@glsstyle@#1}%
5941   {%
5942     \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
5943   }%
5944   {%
5945     \csname @glsstyle@#1\endcsname
5946   }%
5947 }

```

`\glossarystyle`

```

5948 \newcommand*\glossarystyle[1]{%
5949   \ifcsundef{@glsstyle@#1}%
5950   {%
5951     \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
5952   }%
5953   {%
5954     \GlossariesWarning
5955     {Deprecated command \string\glossarystyle.^^J
5956     I recommend you switch to \string\setglossarystyle\space unless
5957     you want to maintain backward compatibility}%
5958     \setglossentrycompatibility
5959     \csname @glsstyle@#1\endcsname

5960   \ifcsdef{@glscompstyle@#1}%
5961     {\setglossentrycompatibility\csuse{@glscompstyle@#1}}%
5962     {}%
5963   }%
5964 }

```

`\newglossarystyle` New glossary styles can be defined using:

```
\newglossarystyle{<name>}{<definition>}
```

The *<definition>* argument should redefine `theglossary`, `\glossaryheader`, `\glsgroupheading`, `\glossaryentryfield` and `\glsgroupskip` (see [subsection 1.19](#) for the definitions of predefined styles). Glossary styles should not redefine `\glossary preamble` and `\glossary postamble`, as the user should be able to switch between styles without affecting the pre- and postambles.

```

5965 \newcommand{\newglossarystyle}[2]{%
5966   \ifcsundef{@glsstyle@#1}%
5967   {%
5968     \expandafter\def\csname @glsstyle@#1\endcsname{#2}%
5969   }%
5970   {%
5971     \PackageError{glossaries}{Glossary style ‘#1’ is already defined}{}%
5972   }%
5973 }

```

`\renewglossarystyle` Code for this macro supplied by Marco Daniel.

```

5974 \newcommand{\renewglossarystyle}[2]{%
5975   \ifcsundef{@glsstyle@#1}%
5976   {%
5977     \PackageError{glossaries}{Glossary style ‘#1’ isn’t already defined}{}%
5978   }%
5979   {%
5980     \csdef{@glsstyle@#1}{#2}%
5981   }%
5982 }

```

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`

```

5983 \newcommand*{\glsnamefont}[1]{#1}

```

Each glossary entry has an associated number list (usually page numbers) that indicate where in the document the entry has been used. The format for these number lists can be changed using the format key in commands like `\glslink`. The default format is given by `\glsnumber`. This takes a single argument which may be a single number, a number range or a number list. The number ranges are delimited with `\delimR`, the number lists are delimited with `\delimN`.

If the document doesn’t have hyperlinks, the numbers can be displayed just as they are, but if the document supports hyperlinks, the numbers should link to the relevant location. This means extracting the individual numbers from the list or ranges. The package does this with the `\hyperpage` command, but this is encoded for comma and dash delimiters and only for the page counter, but this code needs to be more general. So I have adapted the code used in the package.

`\glsnumber`

```

5984 \ifcsundef{hyperlink}%

```

```

5985 {%
5986 \def\glshypernumber#1{#1}%
5987 }%
5988 {%
5989 \def\glshypernumber#1{\@glshypernumber#1\nohyperpage{}}\@nil}
5990 }

```

`\@glshypernumber` This code was provided by Heiko Oberdiek to allow material to be attached to the location.

```

5991 \def\@glshypernumber#1\nohyperpage#2#3\@nil{%
5992 \ifx\#1\%
5993 \else
5994 \@delimR#1\delimR\delimR\%
5995 \fi
5996 \ifx\#2\%
5997 \else
5998 #2%
5999 \fi
6000 \ifx\#3\%
6001 \else
6002 \@glshypernumber#3\@nil
6003 \fi
6004 }

```

`\@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`

```

6005 \def\@delimR#1\delimR #2\delimR #3\%
6006 \ifx\#2\%
6007 \@delimN{#1}%
6008 \else
6009 \@gls@numberlink{#1}\delimR\@gls@numberlink{#2}%
6010 \fi}

```

`\@delimN` displays a list of individual numbers, instead of a range:

`\@delimN`

```

6011 \def\@delimN#1{\@delimN#1\delimN \delimN\}
6012 \def\@delimN#1\delimN #2\delimN#3\%
6013 \ifx\#3\%
6014 \@gls@numberlink{#1}%
6015 \else
6016 \@gls@numberlink{#1}\delimN\@gls@numberlink{#2}%
6017 \fi
6018 }

```

The following code is modified from hyperref's `\HyInd@pagelink` where the name of the counter being used is given by `\@gls@counter`.

```

6019 \def\@gls@numberlink#1{%
6020 \begingroup
6021 \toks@={}%
6022 \@gls@removespaces#1 \@nil
6023 \endgroup}

6024 \def\@gls@removespaces#1 #2\@nil{%
6025 \toks@=\expandafter{\the\toks@#1}%
6026 \ifx\#2\%
6027 \edef\x{\the\toks@}%
6028 \ifx\x\empty
6029 \else

6030 \hyperlink{\glsentrycounter\@glo@counterprefix\the\toks@}%
6031 {\the\toks@}%
6032 \fi
6033 \else
6034 \@gls@ReturnAfterFi{%
6035 \@gls@removespaces#2\@nil
6036 }%
6037 \fi
6038 }
6039 \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
6040 \newcommand*\hyperrm[1]{\textrm{\glsnumber{#1}}}

\hypersf
6041 \newcommand*\hypersf[1]{\textsf{\glsnumber{#1}}}

\hypertt
6042 \newcommand*\hypertt[1]{\texttt{\glsnumber{#1}}}

\hyperbf
6043 \newcommand*\hyperbf[1]{\textbf{\glsnumber{#1}}}

\hypermd
6044 \newcommand*\hypermd[1]{\textmd{\glsnumber{#1}}}

\hyperit
6045 \newcommand*\hyperit[1]{\textit{\glsnumber{#1}}}

\hypersl
6046 \newcommand*\hypersl[1]{\textsl{\glsnumber{#1}}}

```

```
\hyperup
6047 \newcommand*\hyperup}[1]{\textup{\glsnumber{#1}}}
```

```
\hypersc
6048 \newcommand*\hypersc}[1]{\textsc{\glsnumber{#1}}}
```

```
\hyperemph
6049 \newcommand*\hyperemph}[1]{\emph{\glsnumber{#1}}}
```

1.17 Acronyms

```
\oldacronym \oldacronym[⟨label⟩]{⟨abbrv⟩}{⟨long⟩}{⟨key-val list⟩}
```

This emulates the way the old package defined acronyms. It is equivalent to `\newacronym[⟨key-val list⟩]{⟨label⟩}{⟨abbrv⟩}{⟨long⟩}` and it additionally defines the command `\⟨label⟩` which is equivalent to `\gls{⟨label⟩}` (thus `⟨label⟩` must only contain alphabetical characters). If `⟨label⟩` is omitted, `⟨abbrv⟩` is used. This only emulates the syntax of the old package. The way the acronyms appear in the list of acronyms is determined by the definition of `\newacronym` and the glossary style.

Note that `\⟨label⟩` can't have an optional argument if the package is loaded. If hasn't been loaded then you can do `\⟨label⟩[⟨insert⟩]` but you can't do `\⟨label⟩[⟨key-val list⟩]`. For example if you define the acronym `svm`, then you can do `\svm['s]` but you can't do `\svm[format=textbf]`. If the package is loaded, `\svm['s]` will appear as `svm ['s]` which is unlikely to be the desired result. In this case, you will need to use `\gls` explicitly, e.g. `\gls{svm}['s]`. Note that it is up to the user to load if desired.

```
6050 \newcommand{\oldacronym}[4][\gls@label]{%
6051   \def\gls@label{#2}%
6052   \newacronym[#4]{#1}{#2}{#3}%
6053   \ifcsundef{xspace}%
6054     {%
6055       \expandafter\edef\csname#1\endcsname{%
6056         \noexpand\@ifstar{\noexpand\Gls{#1}}{\noexpand\gls{#1}}}%
6057     }%
6058   }%
6059   {%
6060     \expandafter\edef\csname#1\endcsname{%
6061       \noexpand\@ifstar{\noexpand\Gls{#1}\noexpand\xspace}{%
6062         \noexpand\gls{#1}\noexpand\xspace}%
6063     }%
6064   }%
6065 }
```

```
\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`

```
6066 \newcommand{\newacronym}[4] [] {}
```

Set up some convenient short cuts. These need to be changed if `\newacronym` is changed (or if the description key is changed).

`\acrpluralsuffix`

Plural suffix used by `\newacronym`. This just defaults to `\glspluralsuffix` but is changed to include `\textup` if the `smallcaps` option is used, so that the suffix doesn't appear in small caps as it doesn't look right. For example, `ABCS` looks as though the "s" is part of the acronym, but `ABCs` looks as though the "s" is a plural suffix. Since the entire text `abcs` is set in `\textsc`, `\textup` is need to cancel it out.

```
6067 \newcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}
```

If `garamondx` has been loaded, need to use `\textulc` instead of `\textup`.

`\glstextup`

```
6068 \newrobustcmd*{\glstextup}[1]{\ifdef\textulc{\textulc{#1}}{\textup{#1}}}
```

The following are defined for compatibility with version 2.07 and earlier.

`\glsshortkey`

```
6069 \newcommand*{\glsshortkey}{short}
```

`\glsshortpluralkey`

```
6070 \newcommand*{\glsshortpluralkey}{shortplural}
```

`\glslongkey`

```
6071 \newcommand*{\glslongkey}{long}
```

`\glslongpluralkey`

```
6072 \newcommand*{\glslongpluralkey}{longplural}
```

`\acrfull` Full form of the acronym.

```
6073 \newrobustcmd*{\acrfull}{\@gls@hyp@opt\ns@acrfull}
```

```
6074 \newcommand*\ns@acrfull[2] [] {%
```

```
6075 \new@ifnextchar[{\@acrfull{#1}{#2}}%
```

```
6076 \@acrfull{#1}{#2} []}%
```

```
6077 }
```


`\@acrfull` Low-level macro:

```
6078 \def\@acrfull#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6079 \acrfullfmt{#1}{#2}{#3}%
6080 }
```

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.

```
6081 \newcommand*\acrfullfmt}[3]{%
6082 \acrlinkfullformat{\@acrlong}{\@acrshort}{#1}{#2}{#3}%
6083 }
```

`\acrlinkfullformat` Format for full links like `\acrfull`. Syntax: `\acrlinkfullformat{<long cs>}{<short cs>}{<options>}{<label>}{<insert>}`

```
6084 \newcommand\acrlinkfullformat}[5]{%
6085 \acrfullformat{#1{#3}{#4}[#5]}{#2{#3}{#4}[]}%
6086 }
```

`\acrfullformat` Default full form is `<long>` (`<short>`).

```
6087 \newcommand\acrfullformat}[2]{#1\glsspace(#2)}
```

`\glsspace` Robust space to ensure it's written to the `.glsdefs` file.

```
6088 \newrobustcmd\glsspace{\space}
```

Default format for full acronym

`\Acrfull`

```
6089 \newrobustcmd*\Acrfull{\@gls@hyp@opt\ns@Acrfull}
```

```
6090 \newcommand*\ns@Acrfull[2][]{%
```

```
6091 \new@ifnextchar[{\@Acrfull{#1}{#2}}%
```

```
6092 \@Acrfull{#1}{#2}[]}%
6093 }
```

Low-level macro:

```
6094 \def\@Acrfull#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6095 \Acrfullfmt{#1}{#2}{#3}%
6096 }
```

`\Acrfullfmt` First letter upper case full format.

```
6097 \newcommand*\Acrfullfmt}[3]{%
```

```
6098 \acrlinkfullformat{\@Acrlong}{\@acrshort}{#1}{#2}{#3}%
6099 }
```

`\ACRfull`

```
6100 \newrobustcmd*{\ACRfull}{\@gls@hyp@opt\ns@ACRfull}
```

```
6101 \newcommand*\ns@ACRfull[2] [] {%  
6102   \new@ifnextchar[{\@ACRfull{#1}{#2}}%  
6103     {\@ACRfull{#1}{#2} []}%  
6104 }
```

Low-level macro:

```
6105 \def\@ACRfull#1#2[#3] {%
```

Make it easier for acronym styles to change this:

```
6106   \ACRfullfmt{#1}{#2}{#3}%  
6107 }
```

`\ACRfullfmt` All upper case full format.

```
6108 \newcommand*\@ACRfullfmt[3] {%  
6109   \acrlinkfullformat{\@ACRlong}{\@ACRshort}{#1}{#2}{#3}%  
6110 }
```

Plural:

`\acrfullpl`

```
6111 \newrobustcmd*{\acrfullpl}{\@gls@hyp@opt\ns@acrfullpl}
```

```
6112 \newcommand*\ns@acrfullpl[2] [] {%  
6113   \new@ifnextchar[{\@acrfullpl{#1}{#2}}%  
6114     {\@acrfullpl{#1}{#2} []}%  
6115 }
```

Low-level macro:

```
6116 \def\@acrfullpl#1#2[#3] {%
```

Make it easier for acronym styles to change this:

```
6117   \acrfullplfmt{#1}{#2}{#3}%  
6118 }
```

`\acrfullplfmt` No case change plural full format.

```
6119 \newcommand*\@acrfullplfmt[3] {%  
6120   \acrlinkfullformat{\@acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%  
6121 }
```

`\Acrfullpl`

```
6122 \newrobustcmd*{\Acrfullpl}{\@gls@hyp@opt\ns@Acrfullpl}
```

```
6123 \newcommand*\ns@Acrfullpl[2] [] {%  
6124   \new@ifnextchar[{\@Acrfullpl{#1}{#2}}%  
6125     {\@Acrfullpl{#1}{#2} []}%  
6126 }
```

Low-level macro:

```
6127 \def\@Acrfullpl#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6128 \Acrfullplfmt{#1}{#2}{#3}%  
6129 }
```

`\Acrfullplfmt` First letter upper case plural full format.

```
6130 \newcommand*\Acrfullplfmt}[3]{%  
6131 \acrlinkfullformat{\@Acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%  
6132 }
```

`\ACRfullpl`

```
6133 \newrobustcmd*\ACRfullpl{\@gls@hyp@opt\ns@ACRfullpl}  
  
6134 \newcommand*\ns@ACRfullpl[2][ ]{%  
6135 \new@ifnextchar[{\@ACRfullpl{#1}{#2}}%  
6136 {\@ACRfullpl{#1}{#2}[ ]}%  
6137 }
```

Low-level macro:

```
6138 \def\@ACRfullpl#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6139 \ACRfullplfmt{#1}{#2}{#3}%  
6140 }
```

`\ACRfullplfmt` All upper case plural full format.

```
6141 \newcommand*\ACRfullplfmt}[3]{%  
6142 \acrlinkfullformat{\@ACRlongpl}{\@ACRshortpl}{#1}{#2}{#3}%  
6143 }
```

1.18 Predefined acronym styles

`\acronymfont` This is only used with the additional acronym styles:

```
6144 \newcommand{\acronymfont}[1]{#1}
```

`\firstacronymfont` This is only used with the additional acronym styles:

```
6145 \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.

```
6146 \newcommand*\acrnameformat}[2]{\acronymfont{#1}}
```

Define some tokens used by `\newacronym`:

`\glskeylisttok`

```
6147 \newtoks\glskeylisttok
```

`\glslabeltok`

6148 `\newtoks\glslabeltok`

`\glsshorttok`

6149 `\newtoks\glsshorttok`

`\glslongtok`

6150 `\newtoks\glslongtok`

`\newacronymhook` Provide a hook for `\newacronym`:

6151 `\newcommand*\newacronymhook{}`

`etGenericNewAcronym` New improved version of setting the acronym style.

6152 `\newcommand*\SetGenericNewAcronym{%`

Change the behaviour of `\Glsentryname` to workaround expansion issues that cause a problem for `\makefirstuc`

6153 `\let\@Gls@entryname\@Gls@acentryname`

Change the way acronyms are defined:

6154 `\renewcommand{\newacronym}[4][[]]{%`

6155 `\ifdefempty{\@glsacronymlists}%`

6156 `{%`

6157 `\def\@glo@type{\acronymtype}%`

6158 `\setkeys{glossentry}{##1}%`

6159 `\DeclareAcronymList{\@glo@type}%`

6160 `}%`

6161 `{}%`

6162 `\glskeylisttok{##1}%`

6163 `\glslabeltok{##2}%`

6164 `\glsshorttok{##3}%`

6165 `\glslongtok{##4}%`

6166 `\newacronymhook`

6167 `\protected@edef\@do@newglossaryentry{%`

6168 `\noexpand\newglossaryentry{\the\glslabeltok}%`

6169 `{%`

6170 `type=\acronymtype,%`

6171 `name={\expandonce{\acronymentry{##2}}},%`

6172 `sort={\acronymssort{\the\glsshorttok}{\the\glslongtok}},%`

6173 `text={\the\glsshorttok},%`

6174 `short={\the\glsshorttok},%`

6175 `shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%`

6176 `long={\the\glslongtok},%`

6177 `longplural={\the\glslongtok\noexpand\acrpluralsuffix},%`

6178 `\GenericAcronymFields,%`

6179 `\the\glskeylisttok`

6180 `}%`

6181 `}%`

6182 `\@do@newglossaryentry`

6183 `}%`

Make sure that `\acrfull` etc reflects the new style:

```

6184 \renewcommand*\acrfullfmt}[3]{%
6185   \glslink[##1]{##2}{\genacrfullformat{##2}{##3}}}%
6186 \renewcommand*\acrfullfmt}[3]{%
6187   \glslink[##1]{##2}{\Genacrfullformat{##2}{##3}}}%
6188 \renewcommand*\ACRfullfmt}[3]{%
6189   \glslink[##1]{##2}{%
6190     \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}}}%
6191 \renewcommand*\acrfullplfmt}[3]{%
6192   \glslink[##1]{##2}{\genplacrfullformat{##2}{##3}}}%
6193 \renewcommand*\Acrfullplfmt}[3]{%
6194   \glslink[##1]{##2}{\Genplacrfullformat{##2}{##3}}}%
6195 \renewcommand*\ACRfullplfmt}[3]{%
6196   \glslink[##1]{##2}{%
6197     \mfirstucMakeUppercase{\genplacrfullformat{##2}{##3}}}}}%

```

Make sure that `\glentryfull` etc reflects the new style:

```

6198 \renewcommand*\glentryfull}[1]{\genacrfullformat{##1}{}}}%
6199 \renewcommand*\Glsentryfull}[1]{\Genacrfullformat{##1}{}}}%
6200 \renewcommand*\glentryfullpl}[1]{\genplacrfullformat{##1}{}}}%
6201 \renewcommand*\Glsentryfullpl}[1]{\Genplacrfullformat{##1}{}}}%
6202 }

```

`\GenericAcronymFields` Fields used by `\SetGenericNewAcronym` that can be changed by the acronym style.

```

6203 \newcommand*\GenericAcronymFields}{description={\the\glslongtok}}

```

`\acronymentry` `\acronymentry{<label>}`

Display style for the name field in the list of acronyms.

```

6204 \newcommand*\acronymentry}[1]{\acronymfont{\glentryshort{#1}}}

```

`\acronymsort` `\acronymsort{<short>}{<long>}`

Default sort format for acronyms.

```

6205 \newcommand*\acronymsort}[2]{#1}

```

`\setacronymstyle` `\setacronymstyle{<style name>}`

```

6206 \newcommand*\setacronymstyle}[1]{%
6207   \ifcsundef{@glsacr@dispstyle@#1}
6208   {%
6209     \PackageError{glossaries}{Undefined acronym style ‘#1’}{%
6210     }%

```

```

6211  {%
6212    \ifdefempty{\@glsacronymlists}%
6213    {%
6214      \DeclareAcronymList{\acronymtype}%
6215    }%
6216  }%
6217  \SetGenericNewAcronym
6218  \GlsUseAcrStyleDefs{#1}%
6219  \@for\@gls@type:=\@glsacronymlists\do{%
6220    \defglsentryfmt[\@gls@type]{\GlsUseAcrEntryDispStyle{#1}}%
6221  }%
6222 }%
6223 }

```

`\newacronymstyle` `\newacronymstyle{<style name>}{<entry format definition>}{<display definitions>}`

Defines a new acronym style called *<style name>*.

```

6224 \newcommand*\newacronymstyle}[3]{%
6225   \ifcsdef{@glsacr@dispstyle@#1}%
6226   {%
6227     \PackageError{glossaries}{Acronym style ‘#1’ already exists}{}%
6228   }%
6229   {%
6230     \csdef{@glsacr@dispstyle@#1}{#2}%
6231     \csdef{@glsacr@styledefs@#1}{#3}%
6232   }%
6233 }

```

`\renewacronymstyle` Redefines the given acronym style.

```

6234 \newcommand*\renewacronymstyle}[3]{%
6235   \ifcsdef{@glsacr@dispstyle@#1}%
6236   {%
6237     \csdef{@glsacr@dispstyle@#1}{#2}%
6238     \csdef{@glsacr@styledefs@#1}{#3}%
6239   }%
6240   {%
6241     \PackageError{glossaries}{Acronym style ‘#1’ doesn’t exist}{}%
6242   }%
6243 }

```

`\useAcrEntryDispStyle`

```

6244 \newcommand*\GlsUseAcrEntryDispStyle[1]{\csuse{@glsacr@dispstyle@#1}}

```

`\GlsUseAcrStyleDefs`

```

6245 \newcommand*\GlsUseAcrStyleDefs[1]{\csuse{@glsacr@styledefs@#1}}

```

Predefined acronym styles:

long-short *<long>* (*<short>*) acronym style.

```
6246 \newacronymstyle{long-short}%
6247 {%
    Check for long form in case this is a mixed glossary.
6248 \ifglshaslong{\glslabel}{\glsngenacfmt}{\glsngenentryfmt}%
6249 }%
6250 {%
6251 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6252 \renewcommand*{\genacrfullformat}[2]{%
6253 \glsentrylong{##1}##2\space
6254 (\protect\firstacronymfont{\glsentryshort{##1}})%
6255 }%
6256 \renewcommand*{\Genacrfullformat}[2]{%
6257 \Glsentrylong{##1}##2\space
6258 (\protect\firstacronymfont{\glsentryshort{##1}})%
6259 }%
6260 \renewcommand*{\genplacrfullformat}[2]{%
6261 \glsentrylongpl{##1}##2\space
6262 (\protect\firstacronymfont{\glsentryshortpl{##1}})%
6263 }%
6264 \renewcommand*{\Genplacrfullformat}[2]{%
6265 \Glsentrylongpl{##1}##2\space
6266 (\protect\firstacronymfont{\glsentryshortpl{##1}})%
6267 }%
6268 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6269 \renewcommand*{\acronymsort}[2]{##1}%
6270 \renewcommand*{\acronymfont}[1]{##1}%
6271 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
6272 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6273 }
```

long-sp-short Similar to the previous style but allows the space between the long and short form to be customized.

```
6274 \newacronymstyle{long-sp-short}%
6275 {%
    Check for long form in case this is a mixed glossary.
6276 \ifglshaslong{\glslabel}{\glsngenacfmt}{\glsngenentryfmt}%
6277 }%
6278 {%
6279 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6280 \renewcommand*{\genacrfullformat}[2]{%
6281 \glsentrylong{##1}##2\glsacspace{##1}%
6282 (\protect\firstacronymfont{\glsentryshort{##1}})%
6283 }%
6284 \renewcommand*{\Genacrfullformat}[2]{%
6285 \Glsentrylong{##1}##2\glsacspace{##1}%
6286 (\protect\firstacronymfont{\glsentryshort{##1}})%
6287 }%
```

```

6288 \renewcommand*\genplacrfullformat}[2]{%
6289   \glsentrylongpl{##1}##2\glsacspace{##1}%
6290   (\protect\firstacronymfont{\glsentryshortpl{##1}})%
6291 }%
6292 \renewcommand*\Genplacrfullformat}[2]{%
6293   \Glsentrylongpl{##1}##2\glsacspace{##1}%
6294   (\protect\firstacronymfont{\glsentryshortpl{##1}})%
6295 }%
6296 \renewcommand*\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6297 \renewcommand*\acronymsort}[2]{##1}%
6298 \renewcommand*\acronymfont}[1]{##1}%
6299 \renewcommand*\firstacronymfont}[1]{\acronymfont{##1}}%
6300 \renewcommand*\acrpluralsuffix{\glspluralsuffix}%
6301 }

```

`\glsacspace` Space between long and short form for the above style. This uses a non-breakable space if the short form is less than 3em, otherwise it uses a regular space.

```

6302 \newcommand*\glsacspace}[1]{%
6303   \settowidth{\dimen@}{(\firstacronymfont{\glsentryshort{##1}})%
6304   \ifdim\dimen@<3em~\else\space\fi
6305 }

```

`short-long` *(short)* (*long*) acronym style.

```

6306 \newacronymstyle{short-long}%
6307 {%

```

Check for long form in case this is a mixed glossary.

```

6308   \ifglshaslong{\glslabel}{\glsacspace}{\glsacspace}%
6309 }%
6310 {%
6311   \renewcommand*\GenericAcronymFields{description={\the\glslongtok}}%
6312   \renewcommand*\genacrfullformat}[2]{%
6313     \protect\firstacronymfont{\glsentryshort{##1}}##2\space
6314     (\glsentrylong{##1})%
6315   }%
6316   \renewcommand*\Genacrfullformat}[2]{%
6317     \protect\firstacronymfont{\Glsentryshort{##1}}##2\space
6318     (\glsentrylong{##1})%
6319   }%
6320   \renewcommand*\genplacrfullformat}[2]{%
6321     \protect\firstacronymfont{\glsentryshortpl{##1}}##2\space
6322     (\glsentrylongpl{##1})%
6323   }%
6324   \renewcommand*\Genplacrfullformat}[2]{%
6325     \protect\firstacronymfont{\Glsentryshortpl{##1}}##2\space
6326     (\glsentrylongpl{##1})%
6327   }%
6328   \renewcommand*\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%

```



```

6329 \renewcommand*\acronymsort}[2]{##1}%
6330 \renewcommand*\acronymfont}[1]{##1}%
6331 \renewcommand*\firstacronymfont}[1]{\acronymfont{##1}}%
6332 \renewcommand*\acrpluralsuffix{\glspluralsuffix}%
6333 }

```

long-sc-short *<long>* (`\textsc{<short>}`) acronym style.

```

6334 \newacronymstyle{long-sc-short}%
6335 {%
6336 \GlsUseAcrEntryDispStyle{long-short}%
6337 }%
6338 {%
6339 \GlsUseAcrStyleDefs{long-short}%
6340 \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6341 \renewcommand*\acrpluralsuffix{\glsupacrpluralsuffix}%
6342 }

```

long-sm-short *<long>* (`\textsmaller{<short>}`) acronym style.

```

6343 \newacronymstyle{long-sm-short}%
6344 {%
6345 \GlsUseAcrEntryDispStyle{long-short}%
6346 }%
6347 {%
6348 \GlsUseAcrStyleDefs{long-short}%
6349 \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6350 \renewcommand*\acrpluralsuffix{\glsacrpluralsuffix}%
6351 }

```

sc-short-long *<short>* (`\textsc{<long>}`) acronym style.

```

6352 \newacronymstyle{sc-short-long}%
6353 {%
6354 \GlsUseAcrEntryDispStyle{short-long}%
6355 }%
6356 {%
6357 \GlsUseAcrStyleDefs{short-long}%
6358 \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6359 \renewcommand*\acrpluralsuffix{\glsupacrpluralsuffix}%
6360 }

```

sm-short-long *<short>* (`\textsmaller{<long>}`) acronym style.

```

6361 \newacronymstyle{sm-short-long}%
6362 {%
6363 \GlsUseAcrEntryDispStyle{short-long}%
6364 }%
6365 {%
6366 \GlsUseAcrStyleDefs{short-long}%
6367 \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6368 \renewcommand*\acrpluralsuffix{\glsacrpluralsuffix}%
6369 }

```

long-short-desc *⟨long⟩* (*{⟨short⟩}*) acronym style that has an accompanying description (which the user needs to supply).

```
6370 \newacronymstyle{long-short-desc}%
6371 {%
6372   \GlsUseAcrEntryDispStyle{long-short}%
6373 }%
6374 {%
6375   \GlsUseAcrStyleDefs{long-short}%
6376   \renewcommand*{\GenericAcronymFields}{}%
6377   \renewcommand*{\acronymsort}[2]{##2}%
6378   \renewcommand*{\acronymentry}[1]{%
6379     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})%
6380 }
```

long-sp-short-desc *⟨long⟩* (*{⟨short⟩}*) acronym style that has an accompanying description (which the user needs to supply). The space between the long and short form is given by `\glsacspace`.

```
6381 \newacronymstyle{long-sp-short-desc}%
6382 {%
6383   \GlsUseAcrEntryDispStyle{long-sp-short}%
6384 }%
6385 {%
6386   \GlsUseAcrStyleDefs{long-sp-short}%
6387   \renewcommand*{\GenericAcronymFields}{}%
6388   \renewcommand*{\acronymsort}[2]{##2}%
6389   \renewcommand*{\acronymentry}[1]{%
6390     \glentrylong{##1}\glsacspace{##1}(\acronymfont{\glentryshort{##1}})%
6391 }
```

long-sc-short-desc *⟨long⟩* (`\textsc{⟨short⟩}`) acronym style that has an accompanying description (which the user needs to supply).

```
6392 \newacronymstyle{long-sc-short-desc}%
6393 {%
6394   \GlsUseAcrEntryDispStyle{long-sc-short}%
6395 }%
6396 {%
6397   \GlsUseAcrStyleDefs{long-sc-short}%
6398   \renewcommand*{\GenericAcronymFields}{}%
6399   \renewcommand*{\acronymsort}[2]{##2}%
6400   \renewcommand*{\acronymentry}[1]{%
6401     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})%
6402 }
```

long-sm-short-desc *⟨long⟩* (`\textsmaller{⟨short⟩}`) acronym style that has an accompanying description (which the user needs to supply).

```
6403 \newacronymstyle{long-sm-short-desc}%
6404 {%
6405   \GlsUseAcrEntryDispStyle{long-sm-short}%
```

```

6406 }%
6407 {%
6408   \GlsUseAcrStyleDefs{long-sm-short}%
6409   \renewcommand*\GenericAcronymFields{}%
6410   \renewcommand*\acronymsort}[2]{##2}%
6411   \renewcommand*\acronymentry}[1]{%
6412     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6413 }

```

short-long-desc *<short>* (*{<long>}*) acronym style that has an accompanying description (which the user needs to supply).

```

6414 \newacronymstyle{short-long-desc}%
6415 {%
6416   \GlsUseAcrEntryDispStyle{short-long}%
6417 }%
6418 {%
6419   \GlsUseAcrStyleDefs{short-long}%
6420   \renewcommand*\GenericAcronymFields{}%
6421   \renewcommand*\acronymsort}[2]{##2}%
6422   \renewcommand*\acronymentry}[1]{%
6423     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6424 }

```

sc-short-long-desc *<long>* (`\textsc{<short>}`) acronym style that has an accompanying description (which the user needs to supply).

```

6425 \newacronymstyle{sc-short-long-desc}%
6426 {%
6427   \GlsUseAcrEntryDispStyle{sc-short-long}%
6428 }%
6429 {%
6430   \GlsUseAcrStyleDefs{sc-short-long}%
6431   \renewcommand*\GenericAcronymFields{}%
6432   \renewcommand*\acronymsort}[2]{##2}%
6433   \renewcommand*\acronymentry}[1]{%
6434     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6435 }

```

sm-short-long-desc *<long>* (`\textsmaller{<short>}`) acronym style that has an accompanying description (which the user needs to supply).

```

6436 \newacronymstyle{sm-short-long-desc}%
6437 {%
6438   \GlsUseAcrEntryDispStyle{sm-short-long}%
6439 }%
6440 {%
6441   \GlsUseAcrStyleDefs{sm-short-long}%
6442   \renewcommand*\GenericAcronymFields{}%
6443   \renewcommand*\acronymsort}[2]{##2}%
6444   \renewcommand*\acronymentry}[1]{%
6445     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%

```

6446 }

dua *<long>* only acronym style.

```
6447 \newacronymstyle{dua}%  
6448 {%
```

Check for long form in case this is a mixed glossary.

```
6449 \ifdefempty\glscustomtext  
6450 {%  
6451 \ifglshaslong{\glslabel}%  
6452 {%  
6453 \glsifplural  
6454 {%
```

Plural form:

```
6455 \glscapscase  
6456 {%
```

Plural form, don't adjust case:

```
6457 \glsentrylongpl{\glslabel}\glsinsert  
6458 }%  
6459 {%
```

Plural form, make first letter upper case:

```
6460 \Glsentrylongpl{\glslabel}\glsinsert  
6461 }%  
6462 {%
```

Plural form, all caps:

```
6463 \mfirstucMakeUppercase  
6464 {\glsentrylongpl{\glslabel}\glsinsert}%  
6465 }%  
6466 }%  
6467 {%
```

Singular form

```
6468 \glscapscase  
6469 {%
```

Singular form, don't adjust case:

```
6470 \glsentrylong{\glslabel}\glsinsert  
6471 }%  
6472 {%
```

Subsequent singular form, make first letter upper case:

```
6473 \Glsentrylong{\glslabel}\glsinsert  
6474 }%  
6475 {%
```

Subsequent singular form, all caps:

```
6476 \mfirstucMakeUppercase  
6477 {\glsentrylong{\glslabel}\glsinsert}%  
6478 }%
```

```

6479     }%
6480     }%
6481     {%

    Not an acronym:
6482     \glsgenentryfmt
6483     }%
6484     }%
6485     {\glscustomtext\glsinsert}%
6486 }%
6487 {%
6488 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%

6489 \renewcommand*{\acrfullfmt}[3]{%
6490     \glslink[##1]{##2}{\glsentrylong{##2}##3\space
6491         (\acronymfont{\glsentryshort{##2}})}}%
6492 \renewcommand*{\Acrfullfmt}[3]{%
6493     \glslink[##1]{##2}{\Glsentrylong{##2}##3\space
6494         (\acronymfont{\glsentryshort{##2}})}}%
6495 \renewcommand*{\ACRfullfmt}[3]{%
6496     \glslink[##1]{##2}{%
6497         \mfirstucMakeUppercase{\glsentrylong{##2}##3\space
6498             (\acronymfont{\glsentryshort{##2}})}}}%

6499 \renewcommand*{\acrfullplfmt}[3]{%
6500     \glslink[##1]{##2}{\glsentrylongpl{##2}##3\space
6501         (\acronymfont{\glsentryshortpl{##2}})}}%

6502 \renewcommand*{\Acrfullplfmt}[3]{%
6503     \glslink[##1]{##2}{\Glsentrylongpl{##2}##3\space
6504         (\acronymfont{\glsentryshortpl{##2}})}}%
6505 \renewcommand*{\ACRfullplfmt}[3]{%
6506     \glslink[##1]{##2}{%
6507         \mfirstucMakeUppercase{\glsentrylongpl{##2}##3\space
6508             (\acronymfont{\glsentryshortpl{##2}})}}}%
6509 \renewcommand*{\glsentryfull}[1]{%
6510     \glsentrylong{##1}\space(\acronymfont{\glsentryshort{##1}})%
6511 }%
6512 \renewcommand*{\Glsentryfull}[1]{%
6513     \Glsentrylong{##1}\space(\acronymfont{\glsentryshort{##1}})%
6514 }%
6515 \renewcommand*{\glsentryfullpl}[1]{%
6516     \glsentrylongpl{##1}\space(\acronymfont{\glsentryshortpl{##1}})%
6517 }%
6518 \renewcommand*{\Glsentryfullpl}[1]{%
6519     \Glsentrylongpl{##1}\space(\acronymfont{\glsentryshortpl{##1}})%
6520 }%
6521 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6522 \renewcommand*{\acronymsort}[2]{##1}%
6523 \renewcommand*{\acronymfont}[1]{##1}%

```

```
6524 \renewcommand*\acrpluralsuffix{\glsacrpluralsuffix}%
6525 }
```

dua-desc *<long>* only acronym style with user-supplied description.

```
6526 \newacronymstyle{dua-desc}%
6527 {%
6528 \GlsUseAcrEntryDispStyle{dua}%
6529 }%
6530 {%
6531 \GlsUseAcrStyleDefs{dua}%
6532 \renewcommand*\GenericAcronymFields{}}%

6533 \renewcommand*\acronymentry}[1]{\acronymfont{\glsentrylong{##1}}}%
6534 \renewcommand*\acronymsort}[2]{##2}%
6535 }%
```

footnote *<short>*\footnote{*<long>*} acronym style.

```
6536 \newacronymstyle{footnote}%
6537 {%
```

Check for long form in case this is a mixed glossary.

```
6538 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6539 }%
6540 {%
6541 \renewcommand*\GenericAcronymFields{description={\the\glslongtok}}%
```

Need to ensure hyperlinks are switched off on first use:

```
6542 \glshyperfirstfalse
6543 \renewcommand*\genacrfullformat}[2]{%
6544 \protect\firstacronymfont{\glsentryshort{##1}}##2%
6545 \protect\footnote{\glsentrylong{##1}}%
6546 }%
6547 \renewcommand*\Genacrfullformat}[2]{%
6548 \firstacronymfont{\Glsentryshort{##1}}##2%
6549 \protect\footnote{\glsentrylong{##1}}%
6550 }%
6551 \renewcommand*\genplacrfullformat}[2]{%
6552 \protect\firstacronymfont{\glsentryshortpl{##1}}##2%
6553 \protect\footnote{\glsentrylongpl{##1}}%
6554 }%
6555 \renewcommand*\Genplacrfullformat}[2]{%
6556 \protect\firstacronymfont{\Glsentryshortpl{##1}}##2%
6557 \protect\footnote{\glsentrylongpl{##1}}%
6558 }%
6559 \renewcommand*\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6560 \renewcommand*\acronymsort}[2]{##1}%
6561 \renewcommand*\acronymfont}[1]{##1}%
6562 \renewcommand*\acrpluralsuffix{\glsacrpluralsuffix}%
```

Don't use footnotes for \acrfull:

```
6563 \renewcommand*\acrfullfmt}[3]{%
```

```

6564 \glslink[##1]{##2}{\acronymfont{\glsentryshort{##2}}##3\space
6565 (\glsentrylong{##2})}%
6566 \renewcommand*{\Acrfullfmt}[3]{%
6567 \glslink[##1]{##2}{\acronymfont{\Glsentryshort{##2}}##3\space
6568 (\glsentrylong{##2})}%
6569 \renewcommand*{\ACRfullfmt}[3]{%
6570 \glslink[##1]{##2}{%
6571 \mfirstucMakeUppercase{\acronymfont{\glsentryshort{##2}}##3\space
6572 (\glsentrylong{##2})}}}%
6573 \renewcommand*{\acrfullplfmt}[3]{%
6574 \glslink[##1]{##2}{\acronymfont{\glsentryshortpl{##2}}##3\space
6575 (\glsentrylongpl{##2})}}}%
6576 \renewcommand*{\Acrfullplfmt}[3]{%
6577 \glslink[##1]{##2}{\acronymfont{\Glsentryshortpl{##2}}##3\space
6578 (\glsentrylongpl{##2})}}}%
6579 \renewcommand*{\ACRfullplfmt}[3]{%
6580 \glslink[##1]{##2}{%
6581 \mfirstucMakeUppercase{\acronymfont{\glsentryshortpl{##2}}##3\space
6582 (\glsentrylongpl{##2})}}}%

```

Similarly for \glsentryfull etc:

```

6583 \renewcommand*{\glsentryfull}[1]{%
6584 \acronymfont{\glsentryshort{##1}}\space(\glsentrylong{##1})}%
6585 \renewcommand*{\Glsentryfull}[1]{%
6586 \acronymfont{\Glsentryshort{##1}}\space(\glsentrylong{##1})}%
6587 \renewcommand*{\glsentryfullpl}[1]{%
6588 \acronymfont{\glsentryshortpl{##1}}\space(\glsentrylongpl{##1})}%
6589 \renewcommand*{\Glsentryfullpl}[1]{%
6590 \acronymfont{\Glsentryshortpl{##1}}\space(\glsentrylongpl{##1})}%
6591 }

```

footnote-sc \textsc{<short>}\footnote{<long>} acronym style.

```

6592 \newacronymstyle{footnote-sc}%
6593 {%
6594 \GlsUseAcrEntryDispStyle{footnote}%
6595 }%
6596 {%
6597 \GlsUseAcrStyleDefs{footnote}%
6598 \renewcommand{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}
6599 \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6600 \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6601 }%

```

footnote-sm \textsmaller{<short>}\footnote{<long>} acronym style.

```

6602 \newacronymstyle{footnote-sm}%
6603 {%
6604 \GlsUseAcrEntryDispStyle{footnote}%
6605 }%
6606 {%
6607 \GlsUseAcrStyleDefs{footnote}%

```

```

6608 \renewcommand{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}
6609 \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6610 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6611 }%

```

footnote-desc *<short>*\footnote{*<long>*} acronym style that has an accompanying description (which the user needs to supply).

```

6612 \newacronymstyle{footnote-desc}%
6613 {%
6614 \GlsUseAcrEntryDispStyle{footnote}%
6615 }%
6616 {%
6617 \GlsUseAcrStyleDefs{footnote}%
6618 \renewcommand*{\GenericAcronymFields}{}%
6619 \renewcommand*{\acronymsort}[2]{##2}%
6620 \renewcommand*{\acronymentry}[1]{%
6621 \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6622 }

```

footnote-sc-desc \textsc{*<short>*}\footnote{*<long>*} acronym style that has an accompanying description (which the user needs to supply).

```

6623 \newacronymstyle{footnote-sc-desc}%
6624 {%
6625 \GlsUseAcrEntryDispStyle{footnote-sc}%
6626 }%
6627 {%
6628 \GlsUseAcrStyleDefs{footnote-sc}%
6629 \renewcommand*{\GenericAcronymFields}{}%
6630 \renewcommand*{\acronymsort}[2]{##2}%
6631 \renewcommand*{\acronymentry}[1]{%
6632 \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6633 }

```

footnote-sm-desc \textsmaller{*<short>*}\footnote{*<long>*} acronym style that has an accompanying description (which the user needs to supply).

```

6634 \newacronymstyle{footnote-sm-desc}%
6635 {%
6636 \GlsUseAcrEntryDispStyle{footnote-sm}%
6637 }%
6638 {%
6639 \GlsUseAcrStyleDefs{footnote-sm}%
6640 \renewcommand*{\GenericAcronymFields}{}%
6641 \renewcommand*{\acronymsort}[2]{##2}%
6642 \renewcommand*{\acronymentry}[1]{%
6643 \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6644 }

```

fineAcronymSynonyms

```

6645 \newcommand*{\DefineAcronymSynonyms}{%

```


Short form

`\acs`

6646 `\let\acs\acrshort`

First letter uppercase short form

`\Acs`

6647 `\let\Acs\Acrshort`

Plural short form

`\acsp`

6648 `\let\acsp\acrshortpl`

First letter uppercase plural short form

`\Acsp`

6649 `\let\Acsp\Acrshortpl`

Long form

`\acl`

6650 `\let\acl\aclong`

Plural long form

`\aclp`

6651 `\let\aclp\aclongpl`

First letter upper case long form

`\Acl`

6652 `\let\Acl\Aclong`

First letter upper case plural long form

`\Aclp`

6653 `\let\Aclp\Aclongpl`

Full form

`\acf`

6654 `\let\acf\acrfull`

Plural full form

`\acfp`

6655 `\let\acfp\acrfullpl`

First letter upper case full form

`\Acf`

6656 `\let\Acf\Acrfull`

First letter upper case plural full form

`\Acfp`

```
6657 \let\Acfp\Acrfullpl
```

Standard form

`\ac`

```
6658 \let\ac\gls
```

First upper case standard form

`\Ac`

```
6659 \let\Ac\Gls
```

Standard plural form

`\acp`

```
6660 \let\acp\glspl
```

Standard first letter upper case plural form

`\Acp`

```
6661 \let\Acp\Glspl
```

```
6662 }
```

Define synonyms if required

```
6663 \ifglsacrshortcuts
```

```
6664 \DefineAcronymSynonyms
```

```
6665 \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.

```
6666 \newcommand*\SetDefaultAcronymDisplayStyle}[1]{%
```

```
6667 \defglsentryfmt[#1]{\glsentryfmt}%
```

```
6668 }
```

`DefaultNewAcronymDef` Sets up the acronym definition for the default style. The information is provided by the tokens `\glslabeltok`, `\glsshorttok`, `\gslongtok` and `\glskeylisttok`.

```
6669 \newcommand*\DefaultNewAcronymDef){%
```

```
6670 \edef\do@newglossaryentry{%
```

```
6671 \noexpand\newglossaryentry{\the\glslabeltok}%
```

```
6672 {%
```

```
6673 type=\acronymtype,%
```

```
6674 name={\the\glsshorttok},%
```

```
6675 sort={\the\glsshorttok},%
```

```
6676 text={\the\glsshorttok},%
```

```

6677 first={\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
6678 plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6679 firstplural={\acrfullformat{\noexpand\expandonce\noexpand\@glo@longpl}%
6680             {\noexpand\expandonce\noexpand\@glo@shortpl}},%
6681 short={\the\glsshorttok},%
6682 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6683 long={\the\glslongtok},%
6684 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6685 description={\the\glslongtok},%
6686 descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%

```

Remaining options specified by the user:

```

6687     \the\glskeylisttok
6688   }%
6689 }%
6690 \let\@org@gls@assign@firstpl\gls@assign@firstpl
6691 \let\@org@gls@assign@plural\gls@assign@plural
6692 \let\@org@gls@assign@descplural\gls@assign@descplural
6693 \def\gls@assign@firstpl##1##2{%
6694   \@@gls@expand@field{##1}{firstpl}{##2}%
6695 }%
6696 \def\gls@assign@plural##1##2{%
6697   \@@gls@expand@field{##1}{plural}{##2}%
6698 }%
6699 \def\gls@assign@descplural##1##2{%
6700   \@@gls@expand@field{##1}{descplural}{##2}%
6701 }%
6702 \do@newglossaryentry
6703 \let\gls@assign@firstpl\@org@gls@assign@firstpl
6704 \let\gls@assign@plural\@org@gls@assign@plural
6705 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
6706 }

```

DefaultAcronymStyle Set up the default acronym style:

```

6707 \newcommand*\SetDefaultAcronymStyle}{%

```

Set the display style:

```

6708   \@for\@gls@type:=\@glsacronymlists\do{%
6709     \SetDefaultAcronymDisplayStyle{\@gls@type}%
6710   }%

```

Set up the definition of `\newacronym`:

```

6711 \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).

```

6712   \ifx\@glsacronymlists\@empty
6713     \def\@glo@type{\acronymtype}%
6714     \setkeys{glossentry}{##1}%
6715     \DeclareAcronymList{\@glo@type}%

```

```

6716     \SetDefaultAcronymDisplayStyle{\@glo@type}%
6717     \fi
6718     \glskeylisttok{##1}%
6719     \glslabeltok{##2}%
6720     \glsshorttok{##3}%
6721     \glslongtok{##4}%
6722     \newacronymhook
6723     \DefaultNewAcronymDef
6724 }%
6725 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6726 }

```

`\acrfootnote` Used by the footnote acronym styles.

```
6727 \newcommand*{\acrfootnote}[3]{\acrlinkfootnote{#1}{#2}{#3}}
```

`\acrlinkfootnote`

```

6728 \newcommand*{\acrlinkfootnote}[3]{%
6729   \footnote{\glslink[#1]{#2}{#3}}%
6730 }

```

`\acrnofootnote`

```

6731 \newcommand*{\acrnofootnote}[3]{%
6732   \footnote{#3}%
6733 }

```

`AcronymDisplayStyle` Sets the acronym display style for given glossary for the description and footnote combination.

```

6734 \newcommand*{\SetDescriptionFootnoteAcronymDisplayStyle}[1]{%
6735   \def\glsentryfmt[#1]{%

6736     \ifdefempty\glscustomtext
6737     {%
6738       \ifglsused{\glslabel}%
6739       {%
6740         \acronymfont{\glsentryfmt}%
6741       }%
6742     {%
6743       \firstacronymfont{\glsentryfmt}%
6744       \ifglsymbol{\glslabel}%
6745       {%
6746         \expandafter\protect\expandafter\acrfootnote\expandafter
6747         {\@gls@link@opts}{\@gls@link@label}%
6748       }%
6749       \glsifplural
6750       {\glsentrysymbolplural{\glslabel}}%
6751       {\glsentrysymbol{\glslabel}}%
6752     }%
6753   }%
6754 }%

```

```

6755 }%
6756 {\glscustomtext\glsinsert}%
6757 }%
6758 }

```

otnoteNewAcronymDef

```

6759 \newcommand*{\DescriptionFootnoteNewAcronymDef}{%
6760 \edef\@do@newglossaryentry{%
6761 \noexpand\newglossaryentry{\the\glslabeltok}%
6762 {%
6763 type=\acronymtype,%
6764 name={\noexpand\acronymfont{\the\glsshorttok}},%
6765 sort={\the\glsshorttok},%
6766 first={\the\glsshorttok},%
6767 firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6768 text={\the\glsshorttok},%
6769 plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6770 short={\the\glsshorttok},%
6771 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6772 long={\the\glslongtok},%
6773 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6774 symbol={\the\glslongtok},%
6775 symbolplural={\noexpand\expandonce\noexpand\@glo@longpl},%
6776 \the\glskeylisttok
6777 }%
6778 }%
6779 \let\@org@gls@assign@firstpl\gls@assign@firstpl
6780 \let\@org@gls@assign@plural\gls@assign@plural
6781 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
6782 \def\gls@assign@firstpl##1##2{%
6783 \@@gls@expand@field{##1}{firstpl}{##2}%
6784 }%
6785 \def\gls@assign@plural##1##2{%
6786 \@@gls@expand@field{##1}{plural}{##2}%
6787 }%
6788 \def\gls@assign@symbolplural##1##2{%
6789 \@@gls@expand@field{##1}{symbolplural}{##2}%
6790 }%
6791 \@do@newglossaryentry
6792 \let\gls@assign@plural\@org@gls@assign@plural
6793 \let\gls@assign@firstpl\@org@gls@assign@firstpl
6794 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
6795 }

```

ootnoteAcronymStyle

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.

```

6796 \newcommand*{\SetDescriptionFootnoteAcronymStyle}{%

```

```

6797 \renewcommand{\newacronym}[4][\]{%
6798   \ifx\@glsacronymlists\@empty
6799     \def\@glo@type{\acronymtype}%
6800     \setkeys{glossentry}{##1}%
6801     \DeclareAcronymList{\@glo@type}%
6802     \SetDescriptionFootnoteAcronymDisplayStyle{\@glo@type}%
6803   \fi
6804   \glskeylisttok{##1}%
6805   \glslabeltok{##2}%
6806   \glsshorttok{##3}%
6807   \glslongtok{##4}%
6808   \newacronymhook
6809   \DescriptionFootnoteNewAcronymDef
6810 }%

```

If footnote package option is specified, set the first use to append the long form (stored in symbol) as a footnote.

```

6811 \@for\@gls@type:=\@glsacronymlists\do{%
6812   \SetDescriptionFootnoteAcronymDisplayStyle{\@gls@type}%
6813 }%

```

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.

```

6814 \ifglsacrsmallcaps
6815   \renewcommand*\acronymfont}[1]{\textsc{##1}}%
6816   \renewcommand*\acrpluralsuffix{\glsupacrpluralsuffix}%
6817 \else
6818   \ifglsacrsmaller
6819     \renewcommand*\acronymfont}[1]{\textsmaller{##1}}%
6820   \fi
6821 \fi

```

Check for package option clash

```

6822 \ifglsacrdua
6823   \PackageError{glossaries}{Option clash: 'footnote' and 'dua'
6824     can't both be set}{}%
6825 \fi
6826 }%

```

`AcronymDisplayStyle` Sets the acronym display style for given glossary with description and dua combination.

```

6827 \newcommand*\SetDescriptionDUAAcronymDisplayStyle}[1]{%
6828   \def\glsentryfmt[#1]{\glsentryfmt}%
6829 }

```

`ionDUANewAcronymDef`

```

6830 \newcommand*\DescriptionDUANewAcronymDef}{%
6831   \edef\@do@newglossaryentry{%

```

```

6832 \noexpand\newglossaryentry{\the\glslabeltok}%
6833 {%
6834   type=\acronymtype,%
6835   name={\the\glslongtok},%
6836   sort={\the\glslongtok},
6837   text={\the\glslongtok},%
6838   first={\the\glslongtok},%
6839   plural={\noexpand\expandonce\noexpand\@glo@longpl},%
6840   firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
6841   short={\the\glsshorttok},%
6842   shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6843   long={\the\glslongtok},%
6844   longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6845   symbol={\the\glsshorttok},%
6846   symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6847   \the\glskeylisttok
6848 }%
6849 }%
6850 \let\@org@gls@assign@firstpl\gls@assign@firstpl
6851 \let\@org@gls@assign@plural\gls@assign@plural
6852 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
6853 \def\gls@assign@firstpl##1##2{%
6854   \@@gls@expand@field{##1}{firstpl}{##2}%
6855 }%
6856 \def\gls@assign@plural##1##2{%
6857   \@@gls@expand@field{##1}{plural}{##2}%
6858 }%
6859 \def\gls@assign@symbolplural##1##2{%
6860   \@@gls@expand@field{##1}{symbolplural}{##2}%
6861 }%
6862 \do@newglossaryentry
6863 \let\gls@assign@firstpl\@org@gls@assign@firstpl
6864 \let\gls@assign@plural\@org@gls@assign@plural
6865 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
6866 }

```

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.

```

6867 \newcommand*\SetDescriptionDUAAcronymStyle{%
6868   \ifglsacrsmallcaps
6869     \PackageError{glossaries}{Option clash: 'smallcaps' and 'dua'
6870     can't both be set}{}%
6871   \else
6872     \ifglsacrsmaller
6873       \PackageError{glossaries}{Option clash: 'smaller' and 'dua'
6874       can't both be set}{}%
6875     \fi
6876   \fi

```

```

6877 \renewcommand{\newacronym}[4] []{%
6878   \ifx\@glsacronymlists\@empty
6879     \def\@glo@type{\acronymtype}%
6880     \setkeys{glossentry}{##1}%
6881     \DeclareAcronymList{\@glo@type}%
6882     \SetDescriptionDUAAcronymDisplayStyle{\@glo@type}%
6883     \fi
6884     \glskeylisttok{##1}%
6885     \glslabeltok{##2}%
6886     \glsshorttok{##3}%
6887     \glslongtok{##4}%
6888     \newacronymhook
6889     \DescriptionDUANewAcronymDef
6890 }%

```

Set display.

```

6891 \@for\@gls@type:=\@glsacronymlists\do{%
6892   \SetDescriptionDUAAcronymDisplayStyle{\@gls@type}%
6893 }%
6894 }%

```

AcronymDisplayStyle Sets the acronym display style for given glossary using the description setting (but not footnote or dua).

```

6895 \newcommand*\SetDescriptionAcronymDisplayStyle}[1]{%
6896   \def\glsentryfmt[#1]{%

6897     \ifdefempty\glscustomtext
6898     {%
6899       \ifglsused{\glslabel}%
6900       {%

```

Move the inserted text outside of \acronymfont

```

6901         \let\gls@org@insert\glsinsert
6902         \let\glsinsert\@empty
6903         \acronymfont{\glsgenentryfmt}\gls@org@insert
6904       }%
6905     {%
6906       \glsgenentryfmt
6907       \ifglshassymbol{\glslabel}%
6908       {%
6909         \glsifplural
6910         {%
6911           \def\@glo@symbol{\glsentrysymbolplural{\glslabel}}%
6912         }%
6913         {%
6914           \def\@glo@symbol{\glsentrysymbol{\glslabel}}%
6915         }%
6916         \space\protect\firstacronymfont
6917         {\gls@symbol}
6918       }%

```



```

6919         {\@glo@symbol}
6920         {\mfirstucMakeUppercase{\@glo@symbol}}})%
6921     }%
6922     {}%
6923 }%
6924 }%
6925 {\glscustomtext\glsinsert}%
6926 }%
6927 }

```

ptionNewAcronymDef

```

6928 \newcommand*{\DescriptionNewAcronymDef}{%
6929   \edef\@do@newglossaryentry{%
6930     \noexpand\newglossaryentry{\the\glslabeltok}%
6931     {%
6932       type=\acronymtype,%
6933       name={\noexpand
6934         \acrnameformat{\the\glsshorttok}{\the\glslongtok}},%
6935       sort={\the\glsshorttok},%
6936       first={\the\glslongtok},%
6937       firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
6938       text={\the\glsshorttok},%
6939       plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6940       short={\the\glsshorttok},%
6941       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6942       long={\the\glslongtok},%
6943       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6944       symbol={\noexpand\@glo@text},%
6945       symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6946       \the\glskeylisttok}%
6947   }%
6948   \let\@org@gls@assign@firstpl\gls@assign@firstpl
6949   \let\@org@gls@assign@plural\gls@assign@plural
6950   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
6951   \def\gls@assign@firstpl##1##2{%
6952     \@@gls@expand@field{##1}{firstpl}{##2}%
6953   }%
6954   \def\gls@assign@plural##1##2{%
6955     \@@gls@expand@field{##1}{plural}{##2}%
6956   }%
6957   \def\gls@assign@symbolplural##1##2{%
6958     \@@gls@expand@field{##1}{symbolplural}{##2}%
6959   }%
6960   \@do@newglossaryentry
6961   \let\gls@assign@firstpl\@org@gls@assign@firstpl
6962   \let\gls@assign@plural\@org@gls@assign@plural
6963   \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
6964 }

```

riptionAcronymStyle 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.

```

6965 \newcommand*\SetDescriptionAcronymStyle}{%
6966   \renewcommand{\newacronym}[4][]{%
6967     \ifx\@glsacronymlists\@empty
6968       \def\@glo@type{\acronymtype}%
6969       \setkeys{glossentry}{##1}%
6970       \DeclareAcronymList{\@glo@type}%
6971       \SetDescriptionAcronymDisplayStyle{\@glo@type}%
6972     \fi
6973     \glskeylisttok{##1}%
6974     \glslabeltok{##2}%
6975     \glsshorttok{##3}%
6976     \glslongtok{##4}%
6977     \newacronymhook
6978     \DescriptionNewAcronymDef
6979   }%

```

Set display.

```

6980   \@for\@gls@type:=\@glsacronymlists\do{%
6981     \SetDescriptionAcronymDisplayStyle{\@gls@type}%
6982   }%

```

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.

```

6983   \ifglsacrsmallcaps
6984     \renewcommand{\acronymfont}[1]{\textsc{##1}}
6985     \renewcommand*\{acrpluralsuffix}{\glsupacrpluralsuffix}%
6986   \else
6987     \ifglsacrsmaller
6988       \renewcommand*\{acronymfont}[1]{\textsmaller{##1}}%
6989     \fi
6990   \fi
6991 }%

```

`AcronymDisplayStyle` Sets the acronym display style for given glossary with footnote setting (but not description or dua).

```

6992 \newcommand*\SetFootnoteAcronymDisplayStyle}[1]{%
6993   \defglsentryfmt[#1]{%
6994     \ifdefempty\glscustomtext
6995     {%

```

Move the inserted text outside of `\acronymfont`

```

6996     \let\gls@org@insert\glsinsert
6997     \let\glsinsert\@empty
6998     \ifglsused{\glslabel}%
6999     {%

```

```

7000     \acronymfont{\glsgenentryfmt}\gls@org@insert
7001   }%
7002   {%
7003     \firstacronymfont{\glsgenentryfmt}\gls@org@insert
7004     \ifglshaslong{\glslabel}%
7005     {%
7006       \expandafter\protect\expandafter\acrfootnote\expandafter
7007       {\@gls@link@opts}{\@gls@link@label}%
7008       {%
7009         \glsifplural
7010         {\glsentrylongpl{\glslabel}}%
7011         {\glsentrylong{\glslabel}}%
7012       }%
7013     }%

7014   }%
7015 }%
7016 }%
7017 {\glscustomtext\glsinsert}%
7018 }%
7019 }

```

otnoteNewAcronymDef

```

7020 \newcommand*{\FootnoteNewAcronymDef}{%
7021   \edef\@do@newglossaryentry{%
7022     \noexpand\newglossaryentry{\the\glslabeltok}%
7023     {%
7024       type=\acronymtype,%
7025       name={\noexpand\acronymfont{\the\glsshorttok}},%
7026       sort={\the\glsshorttok},%
7027       text={\the\glsshorttok},%
7028       plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7029       first={\the\glsshorttok},%
7030       firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7031       short={\the\glsshorttok},%
7032       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7033       long={\the\glslongtok},%
7034       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7035       description={\the\glslongtok},%
7036       descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7037       \the\glskeylisttok
7038     }%
7039   }%
7040   \let\@org@gls@assign@plural\gls@assign@plural
7041   \let\@org@gls@assign@firstpl\gls@assign@firstpl
7042   \let\@org@gls@assign@descplural\gls@assign@descplural
7043   \def\gls@assign@firstpl##1##2{%
7044     \@gls@expand@field{##1}{firstpl}{##2}%
7045   }%
7046   \def\gls@assign@plural##1##2{%

```

```

7047 \@@gls@expand@field{##1}{plural}{##2}%
7048 }%
7049 \def\gls@assign@descplural##1##2{%
7050 \@@gls@expand@field{##1}{descplural}{##2}%
7051 }%
7052 \do@newglossaryentry
7053 \let\gls@assign@plural\@org@gls@assign@plural
7054 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7055 \let\gls@assign@descplural\@org@gls@assign@descplural
7056 }

```

`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.

```

7057 \newcommand*\SetFootnoteAcronymStyle{%
7058 \renewcommand{\newacronym}[4][[]]{%
7059 \ifx\@glsacronymlists\@empty
7060 \def\@glo@type{\acronymtype}%
7061 \setkeys{glossentry}{##1}%
7062 \DeclareAcronymList{\@glo@type}%
7063 \SetFootnoteAcronymDisplayStyle{\@glo@type}%
7064 \fi
7065 \glskeylisttok{##1}%
7066 \glslabeltok{##2}%
7067 \glsshorttok{##3}%
7068 \glslongtok{##4}%
7069 \newacronymhook
7070 \FootnoteNewAcronymDef
7071 }%

```

Set display

```

7072 \@for\@gls@type:=\@glsacronymlists\do{%
7073 \SetFootnoteAcronymDisplayStyle{\@gls@type}%
7074 }%

```

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.

```

7075 \ifglsacrsmallcaps
7076 \renewcommand*\acronymfont[1]{\textsc{##1}}%
7077 \renewcommand*\acrpluralsuffix{\glsupacrpluralsuffix}%
7078 \else
7079 \ifglsacrsmaller
7080 \renewcommand*\acronymfont[1]{\textsmaller{##1}}%
7081 \fi
7082 \fi

```

Check for option clash

```

7083 \ifglsacrdua
7084 \PackageError{glossaries}{Option clash: 'footnote' and 'dua'

```

```

7085     can't both be set}{}%
7086   \fi
7087 }%

```

`\dsdoparenifnotempty` 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.

```

7088 \DeclareRobustCommand*\glsdsdoparenifnotempty}[2]{%
7089   \protected@edef\gls@tmp{#1}%
7090   \ifdefempty\gls@tmp
7091   }{%
7092   {%
7093     \ifx\gls@tmp@\gls@default@value
7094     \else
7095     \space (#2{#1})%
7096     \fi
7097   }%
7098 }

```

`AcronymDisplayStyle` Sets the acronym display style for given glossary where neither footnote nor description is required, but smallcaps or smaller specified.

```

7099 \newcommand*\SetSmallAcronymDisplayStyle}[1]{%
7100   \def\glsentryfmt[#1]{%

7101   \ifdefempty\glscustomtext
7102   {%

```

Move the inserted text outside of `\acronymfont`

```

7103     \let\gls@org@insert\glsinsert
7104     \let\glsinsert\@empty
7105     \ifglsused{\glslabel}%
7106     {%
7107     \acronymfont{\glsentryfmt}\gls@org@insert
7108     }%
7109     {%
7110     \glsentryfmt
7111     \ifglsymbol{\glslabel}%
7112     {%
7113     \glsifplural
7114     {%
7115     \def\@glo@symbol{\glsentrysymbolplural{\glslabel}}%
7116     }%
7117     {%
7118     \def\@glo@symbol{\glsentrysymbol{\glslabel}}%
7119     }%
7120     \space
7121     (\glsupcase
7122     {\firstacronymfont{\@glo@symbol}}%
7123     {\firstacronymfont{\@glo@symbol}}%

```

```

7124         {\firstacronymfont{\mfirstucMakeUppercase{\@glo@symbol}}})%
7125     }%
7126     {}%
7127     }%
7128 }%
7129 {\glscustomtext\glsinsert}%
7130 }%
7131 }

```

\SmallNewAcronymDef

```

7132 \newcommand*{\SmallNewAcronymDef}{%
7133   \edef\@do@newglossaryentry{%
7134     \noexpand\newglossaryentry{\the\glslabeltok}%
7135     {%
7136       type=\acronymtype,%
7137       name={\noexpand\acronymfont{\the\glsshorttok}},%
7138       sort={\the\glsshorttok},%
7139       text={\the\glsshorttok},%
7140       plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7141       first={\the\glslongtok},%
7142       firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7143       short={\the\glsshorttok},%
7144       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7145       long={\the\glslongtok},%
7146       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7147       description={\noexpand\@glo@first},%
7148       descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7149       symbol={\the\glsshorttok},%
7150       symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7151       \the\glskeylisttok
7152     }%
7153   }%
7154   \let\@org@gls@assign@firstpl\gls@assign@firstpl
7155   \let\@org@gls@assign@plural\gls@assign@plural
7156   \let\@org@gls@assign@descplural\gls@assign@descplural
7157   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7158   \def\gls@assign@firstpl##1##2{%
7159     \@@gls@expand@field{##1}{firstpl}{##2}%
7160   }%
7161   \def\gls@assign@plural##1##2{%
7162     \@@gls@expand@field{##1}{plural}{##2}%
7163   }%
7164   \def\gls@assign@descplural##1##2{%
7165     \@@gls@expand@field{##1}{descplural}{##2}%

```

```

7166 }%
7167 \def\gls@assign@symbolplural##1##2{%
7168   \@gls@expand@field{##1}{symbolplural}{##2}%
7169 }%
7170 \@do@newglossaryentry
7171 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7172 \let\gls@assign@plural\@org@gls@assign@plural
7173 \let\gls@assign@descplural\@org@gls@assign@descplural
7174 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7175 }

```

`\SetSmallAcronymStyle` 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.

```

7176 \newcommand*\SetSmallAcronymStyle{%
7177   \renewcommand{\newacronym}[4][]{%
7178     \ifx\@glsacronymlists\@empty
7179       \def\@glo@type{\acronymtype}%
7180       \setkeys{glossentry}{##1}%
7181       \DeclareAcronymList{\@glo@type}%
7182       \SetSmallAcronymDisplayStyle{\@glo@type}%
7183     \fi
7184     \glskeylisttok{##1}%
7185     \glslabeltok{##2}%
7186     \glsshorttok{##3}%
7187     \glslongtok{##4}%
7188     \newacronymhook
7189     \SmallNewAcronymDef
7190   }%

```

Change the display since first only contains long form.

```

7191 \@for\@gls@type:=\@glsacronymlists\do{%
7192   \SetSmallAcronymDisplayStyle{\@gls@type}%
7193 }%

```

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.

```

7194 \ifglsacrsmallcaps
7195   \renewcommand*\acronymfont}[1]{\textsc{##1}}
7196   \renewcommand*\acrpluralsuffix{\glsupacrpluralsuffix}%
7197 \else
7198   \renewcommand*\acronymfont}[1]{\textsmaller{##1}}
7199 \fi

```

check for option clash

```

7200 \ifglsacrdua
7201   \ifglsacrsmallcaps
7202     \PackageError{glossaries}{Option clash: 'smallcaps' and 'dua'
7203       can't both be set}{}%
7204   \else

```

```

7205     \PackageError{glossaries}{Option clash: ‘smaller’ and ‘dua’
7206     can’t both be set}{}%
7207     \fi
7208     \fi
7209 }%

```

`\SetDUADisplayStyle` Sets the acronym display style for given glossary with dua setting.

```

7210 \newcommand*\SetDUADisplayStyle}[1]{%
7211   \def\glsentryfmt[#1]{\glsentryfmt}%
7212 }

```

`\DUANewAcronymDef`

```

7213 \newcommand*\DUANewAcronymDef}{%
7214   \edef\@do@newglossaryentry{%
7215     \noexpand\newglossaryentry{\the\glslabeltok}%
7216     {%
7217       type=\acronymtype,%
7218       name={\the\glsshorttok},%
7219       text={\the\glslongtok},%
7220       first={\the\glslongtok},%
7221       plural={\noexpand\expandonce\noexpand\@glo@longpl},%
7222       firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7223       short={\the\glsshorttok},%
7224       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7225       long={\the\glslongtok},%
7226       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7227       description={\the\glslongtok},%
7228       descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7229       symbol={\the\glsshorttok},%
7230       symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7231       \the\glskeylisttok
7232     }%
7233   }%
7234   \let\@org@gls@assign@firstpl\gls@assign@firstpl
7235   \let\@org@gls@assign@plural\gls@assign@plural
7236   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7237   \let\@org@gls@assign@descplural\gls@assign@descplural
7238   \def\gls@assign@firstpl##1##2{%
7239     \@@gls@expand@field{##1}{firstpl}{##2}%
7240   }%
7241   \def\gls@assign@plural##1##2{%
7242     \@@gls@expand@field{##1}{plural}{##2}%
7243   }%
7244   \def\gls@assign@symbolplural##1##2{%
7245     \@@gls@expand@field{##1}{symbolplural}{##2}%
7246   }%
7247   \def\gls@assign@descplural##1##2{%
7248     \@@gls@expand@field{##1}{descplural}{##2}%
7249   }%

```



```

7250 \do@newglossaryentry
7251 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7252 \let\gls@assign@plural\@org@gls@assign@plural
7253 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7254 \let\gls@assign@descplural\@org@gls@assign@descplural
7255 }

```

`\SetDUASStyle` Always expand acronyms.

```

7256 \newcommand*{\SetDUASStyle}{%
7257   \renewcommand{\newacronym}[4][[]]{%
7258     \ifx\@glsacronymlists\@empty
7259       \def\@glo@type{\acronymtype}%
7260       \setkeys{glossentry}{##1}%
7261       \DeclareAcronymList{\@glo@type}%
7262       \SetDUADisplayStyle{\@glo@type}%
7263       \fi
7264       \glskeylisttok{##1}%
7265       \glslabeltok{##2}%
7266       \glsshorttok{##3}%
7267       \glslongtok{##4}%
7268       \newacronymhook
7269       \DUANewAcronymDef
7270   }%

```

Set the display

```

7271   \@for\@gls@type:=\@glsacronymlists\do{%
7272     \SetDUADisplayStyle{\@gls@type}%
7273   }%
7274 }

```

`\SetAcronymStyle`

```

7275 \newcommand*{\SetAcronymStyle}{%
7276   \SetDefaultAcronymStyle
7277   \ifglsacrdescription
7278     \ifglsacrfootnote
7279       \SetDescriptionFootnoteAcronymStyle
7280     \else
7281       \ifglsacrdua
7282         \SetDescriptionDUAAcronymStyle
7283       \else
7284         \SetDescriptionAcronymStyle
7285       \fi
7286     \fi
7287   \else
7288     \ifglsacrfootnote
7289       \SetFootnoteAcronymStyle
7290     \else
7291       \ifthenelse{\boolean{glsacrsmalldescription}\OR
7292         \boolean{glsacrsmaller}}{
7293         {

```

```

7294     \SetSmallAcronymStyle
7295   }%
7296   {%
7297     \ifglsacrdua
7298     \SetDUASStyle
7299     \fi
7300   }%
7301   \fi
7302   \fi
7303 }

```

Set the acronym style according to the package options

```
7304 \SetAcronymStyle
```

Allow user to define their own custom acronyms. (For compatibility with versions before v3.0, the short form is stored in the user1 key, the plural short form is stored in the user2 key, the long form is stored in the user3 key and the plural long form is stored in the user4 key.) Defaults to displaying only the acronym with the long form as the description.

`\SetCustomDisplayStyle` Sets the acronym display style.

```

7305 \newcommand*\SetCustomDisplayStyle[1]{%
7306   \defglsentryfmt[#1]{\glsgenentryfmt}%
7307 }

```

`\CustomAcronymFields`

```

7308 \newcommand*\CustomAcronymFields{%
7309   name={\the\glsshorttok},%
7310   description={\the\glslongtok},%
7311   first={\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
7312   firstplural={\acrfullformat
7313     {\noexpand\glsentrylongpl{\the\glslabeltok}}%
7314     {\noexpand\glsentryshortpl{\the\glslabeltok}}},%
7315   text={\the\glsshorttok},%
7316   plural={\the\glsshorttok\noexpand\acrpluralsuffix}%
7317 }

```

`\CustomNewAcronymDef`

```

7318 \newcommand*\CustomNewAcronymDef{%
7319   \protected@edef\@do@newglossaryentry{%
7320     \noexpand\newglossaryentry{\the\glslabeltok}%
7321     {%
7322       type=\acronymtype,%
7323       short={\the\glsshorttok},%
7324       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7325       long={\the\glslongtok},%
7326       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7327       user1={\the\glsshorttok},%

```

```

7328     user2={\the\glsshorttok\noexpand\acrpluralsuffix},%
7329     user3={\the\glslongtok},%
7330     user4={\the\glslongtok\noexpand\acrpluralsuffix},%
7331     \CustomAcronymFields,%
7332     \the\glskeylisttok
7333   }%
7334 }%
7335 \@do@newglossaryentry
7336 }

```

\SetCustomStyle

```

7337 \newcommand*\SetCustomStyle{%
7338   \renewcommand{\newacronym}[4][\%
7339     \ifx\@glsacronymlists\@empty
7340       \def\@glo@type{\acronymtype}%
7341       \setkeys{glossentry}{##1}%
7342       \DeclareAcronymList{\@glo@type}%
7343       \SetCustomDisplayStyle{\@glo@type}%
7344     \fi
7345     \glskeylisttok{##1}%
7346     \glslabeltok{##2}%
7347     \glsshorttok{##3}%
7348     \glslongtok{##4}%
7349     \newacronymhook
7350     \CustomNewAcronymDef
7351   }%

```

Set the display

```

7352   \@for\@gls@type:=\@glsacronymlists\do{%
7353     \SetCustomDisplayStyle{\@gls@type}%
7354   }%
7355 }

```

1.19 Predefined Glossary Styles

The glossaries bundle comes with some predefined glossary styles. These need to be loaded now for the style option to use them.

First, the glossary hyper-navigation commands need to be loaded.

```
7356 \RequirePackage{glossary-hypernav}
```

The styles that use list-like environments. These are not loaded if the `nolist` option is used:

```
7357 \@gls@loadlist
```

The styles that use the `longtable` environment. These are not loaded if the `no-long package` option is used.

```
7358 \@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.

```
7359 \@gls@loadsuper
```

The tree-like styles. These are not loaded if the `notree` package option is used.

```
7360 \@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.

```
7361 \ifx\@glossary@default@style\relax
```

```
7362 \else
```

```
7363   \setglossarystyle{\@glossary@default@style}
```

```
7364 \fi
```

1.20 Debugging Commands

```
\showgloparent  \showgloparent{<label>}
```

```
7365 \newcommand*\showgloparent}[1]{%
```

```
7366   \expandafter\show\csname glo@\glsdetoklabel{#1}@parent\endcsname
```

```
7367 }
```

```
\showglolevel  \showglolevel{<label>}
```

```
7368 \newcommand*\showglolevel}[1]{%
```

```
7369   \expandafter\show\csname glo@\glsdetoklabel{#1}@level\endcsname
```

```
7370 }
```

```
\showglotext  \showglotext{<label>}
```

```
7371 \newcommand*\showglotext}[1]{%
```

```
7372   \expandafter\show\csname glo@\glsdetoklabel{#1}@text\endcsname
```

```
7373 }
```

```
\showgloplural  \showgloplural{<label>}
```

```
7374 \newcommand*\showgloplural}[1]{%
```

```
7375   \expandafter\show\csname glo@\glsdetoklabel{#1}@plural\endcsname
```

```
7376 }
```

`\showglofirst` `\showglofirst{<label>}`

```
7377 \newcommand*{\showglofirst}[1]{%
7378   \expandafter\show\csname glo@glstetoklabel{#1}@first\endcsname
7379 }
```

`\showglofirstpl` `\showglofirstpl{<label>}`

```
7380 \newcommand*{\showglofirstpl}[1]{%
7381   \expandafter\show\csname glo@glstetoklabel{#1}@firstpl\endcsname
7382 }
```

`\showglotype` `\showglotype{<label>}`

```
7383 \newcommand*{\showglotype}[1]{%
7384   \expandafter\show\csname glo@glstetoklabel{#1}@type\endcsname
7385 }
```

`\showglocounter` `\showglocounter{<label>}`

```
7386 \newcommand*{\showglocounter}[1]{%
7387   \expandafter\show\csname glo@glstetoklabel{#1}@counter\endcsname
7388 }
```

`\showglouserii` `\showglouserii{<label>}`

```
7389 \newcommand*{\showglouserii}[1]{%
7390   \expandafter\show\csname glo@glstetoklabel{#1}@userii\endcsname
7391 }
```

`\showglouserii` `\showglouserii{<label>}`

```
7392 \newcommand*{\showglouserii}[1]{%
7393   \expandafter\show\csname glo@glstetoklabel{#1}@userii\endcsname
7394 }
```

`\showglouseriii` `\showglouseriii{<label>}`

```
7395 \newcommand*{\showglouseriii}[1]{%  
7396   \expandafter\show\csname glo@glstdetoklabel{#1}@useriii\endcsname  
7397 }
```

`\showglouseriv` `\showglouseriv{<label>}`

```
7398 \newcommand*{\showglouseriv}[1]{%  
7399   \expandafter\show\csname glo@glstdetoklabel{#1}@useriv\endcsname  
7400 }
```

`\showglouserv` `\showglouserv{<label>}`

```
7401 \newcommand*{\showglouserv}[1]{%  
7402   \expandafter\show\csname glo@glstdetoklabel{#1}@userv\endcsname  
7403 }
```

`\showglouservi` `\showglouservi{<label>}`

```
7404 \newcommand*{\showglouservi}[1]{%  
7405   \expandafter\show\csname glo@glstdetoklabel{#1}@uservi\endcsname  
7406 }
```

`\showgloname` `\showgloname{<label>}`

```
7407 \newcommand*{\showgloname}[1]{%  
7408   \expandafter\show\csname glo@glstdetoklabel{#1}@name\endcsname  
7409 }
```

`\showglodesc` `\showglodesc{<label>}`

```
7410 \newcommand*{\showglodesc}[1]{%  
7411   \expandafter\show\csname glo@glstdetoklabel{#1}@desc\endcsname  
7412 }
```

`\showglodescplural` `\showglodescplural{<label>}`

```
7413 \newcommand*{\showglodescplural}[1]{%
7414   \expandafter\show\csname glo@glsdetoklabel{#1}@descplural\endcsname
7415 }
```

`\showglosort` `\showglosort{<label>}`

```
7416 \newcommand*{\showglosort}[1]{%
7417   \expandafter\show\csname glo@glsdetoklabel{#1}@sort\endcsname
7418 }
```

`\showglosymbol` `\showglosymbol{<label>}`

```
7419 \newcommand*{\showglosymbol}[1]{%
7420   \expandafter\show\csname glo@glsdetoklabel{#1}@symbol\endcsname
7421 }
```

`\showglosymbolplural` `\showglosymbolplural{<label>}`

```
7422 \newcommand*{\showglosymbolplural}[1]{%
7423   \expandafter\show\csname glo@glsdetoklabel{#1}@symbolplural\endcsname
7424 }
```

`\showgloshort` `\showgloshort{<label>}`

```
7425 \newcommand*{\showgloshort}[1]{%
7426   \expandafter\show\csname glo@glsdetoklabel{#1}@short\endcsname
7427 }
```

`\showglolong` `\showglolong{<label>}`

```
7428 \newcommand*{\showglolong}[1]{%
7429   \expandafter\show\csname glo@glsdetoklabel{#1}@long\endcsname
7430 }
```

`\showgloindex` `\showgloindex{<label>}`

```
7431 \newcommand*{\showgloindex}[1]{%
7432   \expandafter\show\csname glo@glstetoklabel{#1}@index\endcsname
7433 }
```

`\showgloflag` `\showgloflag{<label>}`

```
7434 \newcommand*{\showgloflag}[1]{%
7435   \expandafter\show\csname ifglo@glstetoklabel{#1}@flag\endcsname
7436 }
```

`\showgloclist` `\showgloclist{<label>}`

```
7437 \newcommand*{\showgloclist}[1]{%
7438   \expandafter\show\csname glo@glstetoklabel{#1}@loclist\endcsname
7439 }
```

`\showglofield` `\showglofield{<label>}{<field>}`

```
7440 \newcommand*{\showglofield}[2]{%
7441   \csshow{glo@glstetoklabel{#1}@#2}%
7442 }
```

`\showacronymlists` `\showacronymlists`

Show list of glossaries that have been flagged as a list of acronyms.

```
7443 \newcommand*{\showacronymlists}{%
7444   \show\@glsacronymlists
7445 }
```

`\showglossaries` `\showglossaries`

Show list of defined glossaries.

```
7446 \newcommand*{\showglossaries}{%
7447   \show\@glo@types
7448 }
```


`\showglossaryin` `\showglossaryin{<glossary-label>}`

Show the ‘in’ extension for the given glossary.

```
7449 \newcommand*{\showglossaryin}[1]{%
7450   \expandafter\show\csname @glotype@#1@in\endcsname
7451 }
```

`\showglossaryout` `\showglossaryout{<glossary-label>}`

Show the ‘out’ extension for the given glossary.

```
7452 \newcommand*{\showglossaryout}[1]{%
7453   \expandafter\show\csname @glotype@#1@out\endcsname
7454 }
```

`\showglossarytitle` `\showglossarytitle{<glossary-label>}`

Show the title for the given glossary.

```
7455 \newcommand*{\showglossarytitle}[1]{%
7456   \expandafter\show\csname @glotype@#1@title\endcsname
7457 }
```

`\showglossarycounter` `\showglossarycounter{<glossary-label>}`

Show the counter for the given glossary.

```
7458 \newcommand*{\showglossarycounter}[1]{%
7459   \expandafter\show\csname @glotype@#1@counter\endcsname
7460 }
```

`\showglossaryentries` `\showglossaryentries{<glossary-label>}`

Show the list of entry labels for the given glossary.

```
7461 \newcommand*{\showglossaryentries}[1]{%
7462   \expandafter\show\csname glolist@#1\endcsname
7463 }
```

1.21 Compatibility with version 2.07 and below

In order to fix some bugs in v3.0, it was necessary to change the way information is written to the `glo` file, which also meant a change in the format of the Kindy 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.

```
7464 \csname ifglscompatible-2.07\endcsname
7465 \RequirePackage{glossaries-compatible-207}
7466 \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.

```
7467 \NeedsTeXFormat{LaTeX2e}
7468 \ProvidesPackage{glossaries-prefix}[2015/09/09 v4.18 (NLCT)]

  Pass all options to glossaries:
7469 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}

  Process options:
7470 \ProcessOptions

  Load glossaries:
7471 \RequirePackage{glossaries}

  Add the new keys:
7472 \define@key{glossentry}{prefixfirst}{\def\@glo@entryprefixfirst{#1}}%
7473 \define@key{glossentry}{prefixfirstplural}{\def\@glo@entryprefixfirstplural{#1}}%
7474 \define@key{glossentry}{prefix}{\def\@glo@entryprefix{#1}}%
7475 \define@key{glossentry}{prefixplural}{\def\@glo@entryprefixplural{#1}}%

  Add them to \@gls@keymap:
7476 \appto\@gls@keymap{,%
7477   {prefixfirst}{prefixfirst},%
7478   {prefixfirstplural}{prefixfirstplural},%
7479   {prefix}{prefix},%
7480   {prefixplural}{prefixplural}}%
7481 }

  Set the default values:
7482 \appto\@newglossaryentryprehook{%
7483   \def\@glo@entryprefix{}%
7484   \def\@glo@entryprefixplural{}%
```

```

7485 \let\@glo@entryprefixfirst\@gls@default@value
7486 \let\@glo@entryprefixfirstplural\@gls@default@value
7487 }

```

Set the assignment code:

```

7488 \appto\@newglossaryentryposthook{%
7489 \gls@assign@field}{\@glo@label}{prefix}{\@glo@entryprefix}%
7490 \gls@assign@field}{\@glo@label}{prefixplural}{\@glo@entryprefixplural}%

```

If prefixfirst has not been supplied, make it the same as prefix.

```

7491 \expandafter\gls@assign@field\expandafter
7492 {\csname glo@\@glo@label @prefix\endcsname}{\@glo@label}{prefixfirst}%
7493 {\@glo@entryprefixfirst}%

```

If prefixfirstplural has not been supplied, make it the same as prefixplural.

```

7494 \expandafter\gls@assign@field\expandafter
7495 {\csname glo@\@glo@label @prefixplural\endcsname}{\@glo@label}%
7496 {prefixfirstplural}{\@glo@entryprefixfirstplural}%
7497 }

```

Define commands to access these fields:

glsentryprefixfirst

```

7498 \newcommand*\glsentryprefixfirst[1]{\csuse{glo@#1@prefixfirst}}

```

ryprefixfirstplural

```

7499 \newcommand*\glsentryprefixfirstplural[1]{\csuse{glo@#1@prefixfirstplural}}

```

\glsentryprefix

```

7500 \newcommand*\glsentryprefix[1]{\csuse{glo@#1@prefix}}

```

lsentryprefixplural

```

7501 \newcommand*\glsentryprefixplural[1]{\csuse{glo@#1@prefixplural}}

```

Now for the initial upper case variants:

Glsentryprefixfirst

```

7502 \newrobustcmd*\Glsentryprefixfirst[1]{%
7503 \protected@edef\@glo@text{\csname glo@#1@prefixfirst\endcsname}%
7504 \xmakefirstuc\@glo@text
7505 }

```

ryprefixfirstplural

```

7506 \newrobustcmd*\Glsentryprefixfirstplural[1]{%
7507 \protected@edef\@glo@text{\csname glo@#1@prefixfirstplural\endcsname}%
7508 \xmakefirstuc\@glo@text
7509 }

```

`\Glsentryprefix`

```
7510 \newrobustcmd*{\Glsentryprefix}[1]{%
7511   \protected@edef\@glo@text{\csname glo@#1@prefix\endcsname}%
7512   \xmakefirstuc\@glo@text
7513 }
```

`lentryprefixplural`

```
7514 \newrobustcmd*{\Glsentryprefixplural}[1]{%
7515   \protected@edef\@glo@text{\csname glo@#1@prefixplural\endcsname}%
7516   \xmakefirstuc\@glo@text
7517 }
```

Define commands to determine if the prefix keys have been set:

`\ifglshasprefix`

```
7518 \newcommand*{\ifglshasprefix}[3]{%
7519   \ifcempty{glo@#1@prefix}%
7520   {#3}%
7521   {#2}%
7522 }
```

`ifglshasprefixplural`

```
7523 \newcommand*{\ifglshasprefixplural}[3]{%
7524   \ifcempty{glo@#1@prefixplural}%
7525   {#3}%
7526   {#2}%
7527 }
```

`ifglshasprefixfirst`

```
7528 \newcommand*{\ifglshasprefixfirst}[3]{%
7529   \ifcempty{glo@#1@prefixfirst}%
7530   {#3}%
7531   {#2}%
7532 }
```

`asprefixfirstplural`

```
7533 \newcommand*{\ifglshasprefixfirstplural}[3]{%
7534   \ifcempty{glo@#1@prefixfirstplural}%
7535   {#3}%
7536   {#2}%
7537 }
```

Define commands that insert the prefix before commands like `\gls`:

`\pgls`

```
7538 \newrobustcmd{\pgls}{\@gls@hyp@opt\@pgls}
```

\@pgls Unstarred version.

```
7539 \newcommand*\@pgls}[2] [] {%
7540   \new@ifnextchar [%
7541     {\@pgls@{#1}{#2}}%
7542     {\@pgls@{#1}{#2} []}%
7543 }
```

\@pgls@ Read in the final optional argument:

```
7544 \def\@pgls@#1#2[#3] {%
7545   \glsdoifexists{#2}%
7546   {%
7547     \ifglsused{#2}%
7548     {%
7549       \glsentryprefix{#2}%
7550     }%
7551     {%
7552       \glsentryprefixfirst{#2}%
7553     }%
7554     \@gls@{#1}{#2}[#3]%
7555   }%
7556 }
```

Similarly for the plural version:

\pglspl

```
7557 \newrobustcmd{\pglspl}{\@gls@hyp@opt\@pglspl}
```

\@pglspl Unstarred version.

```
7558 \newcommand*\@pglspl}[2] [] {%
7559   \new@ifnextchar [%
7560     {\@pglspl@{#1}{#2}}%
7561     {\@pglspl@{#1}{#2} []}%
7562 }
```

\@pglspl@ Read in the final optional argument:

```
7563 \def\@pglspl@#1#2[#3] {%
7564   \glsdoifexists{#2}%
7565   {%
7566     \ifglsused{#2}%
7567     {%
7568       \glsentryprefixplural{#2}%
7569     }%
7570     {%
7571       \glsentryprefixfirstplural{#2}%
7572     }%
7573     \@glspl@{#1}{#2}[#3]%
7574   }%
7575 }
```

Now for the first letter upper case versions:

```
\PglS
7576 \newrobustcmd{\PglS}{\@gls@hyp@opt\@PglS}
```

\@PglS Unstarred version.

```
7577 \newcommand*{\@PglS}[2] [] {%
7578   \new@ifnextchar[%
7579     {\@PglS@{#1}{#2}}%
7580     {\@PglS@{#1}{#2} []}%
7581 }
```

\@PglS@ Read in the final optional argument:

```
7582 \def\@PglS@#1#2[#3] {%
7583   \glsdoifexists{#2}%
7584   {%
7585     \ifglsused{#2}%
7586     {%
7587       \ifglsHasprefix{#2}%
7588       {%
7589         \Glsentryprefix{#2}%
7590         \@gls@{#1}{#2}[#3]%
7591       }%
7592       {\@Gls@{#1}{#2}[#3]}%
7593     }%
7594     {%
7595       \ifglsHasprefixfirst{#2}%
7596       {%
7597         \Glsentryprefixfirst{#2}%
7598         \@gls@{#1}{#2}[#3]%
7599       }%
7600       {\@Gls@{#1}{#2}[#3]}%
7601     }%
7602   }%
7603 }
```

Similarly for the plural version:

```
\PglSpl
7604 \newrobustcmd{\PglSpl}{\@gls@hyp@opt\@PglSpl}
```

\@PglSpl Unstarred version.

```
7605 \newcommand*{\@PglSpl}[2] [] {%
7606   \new@ifnextchar[%
7607     {\@PglSpl@{#1}{#2}}%
7608     {\@PglSpl@{#1}{#2} []}%
7609 }
```

\@Pglsp1@ Read in the final optional argument:

```
7610 \def\@Pglsp1@#1#2[#3]{%
7611   \glsdoifexists{#2}%
7612   {%
7613     \ifglsused{#2}%
7614     {%
7615       \ifglshasprefixplural{#2}%
7616       {%
7617         \Glsentryprefixplural{#2}%
7618         \@glspl@{#1}{#2}[#3]%
7619       }%
7620       {\@Glspl@{#1}{#2}[#3]}%
7621     }%
7622     {%
7623       \ifglshasprefixfirstplural{#2}%
7624       {%
7625         \Glsentryprefixfirstplural{#2}%
7626         \@glspl@{#1}{#2}[#3]%
7627       }%
7628       {\@Glspl@{#1}{#2}[#3]}%
7629     }%
7630   }%
7631 }
```

Finally the all upper case versions:

\PGLS

```
7632 \newrobustcmd{\PGLS}{\@gls@hyp@opt\@PGLS}
```

\@PGLS Unstarred version.

```
7633 \newcommand*{\@PGLS}[2][ ]{%
7634   \new@ifnextchar[%
7635     {\@PGLS@{#1}{#2}}%
7636     {\@PGLS@{#1}{#2}[]}%
7637 }
```

\@PGLS@ Read in the final optional argument:

```
7638 \def\@PGLS@#1#2[#3]{%
7639   \glsdoifexists{#2}%
7640   {%
7641     \ifglsused{#2}%
7642     {%
7643       \mfirstucMakeUppercase{\glsentryprefix{#2}}%
7644     }%
7645     {%
7646       \mfirstucMakeUppercase{\glsentryprefixfirst{#2}}%
7647     }%
7648     \@GLS@{#1}{#2}[#3]%
7649   }%
7650 }
```

```
7649 }%
7650 }
```

Plural version:

```
\PGLSp1
7651 \newrobustcmd{\PGLSp1}{\@gls@hyp@opt\PGLSp1}
```

\@PGLSp1 Unstarred version.

```
7652 \newcommand*{\@PGLSp1}[2][{}]{%
7653 \new@ifnextchar[%
7654 {\@PGLSp1@{#1}{#2}}%
7655 {\@PGLSp1@{#1}{#2}[]}%
7656 }
```

\@PGLSp1@ Read in the final optional argument:

```
7657 \def\@PGLSp1@#1#2[#3]{%
7658 \glsdoifexists{#2}%
7659 {%
7660 \ifglsused{#2}%
7661 {%
7662 \mfirstucMakeUppercase{\glsentryprefixplural{#2}}%
7663 }%
7664 {%
7665 \mfirstucMakeUppercase{\glsentryprefixfirstplural{#2}}%
7666 }%
7667 \@GGLSp1@{#1}{#2}[#3]%
7668 }%
7669 }
```

3 Glossary Styles

3.1 Glossary hyper-navigation definitions (glossary-hypernav package)

Package Definition:

```
7670 \ProvidesPackage{glossary-hypernav}[2015/09/09 v4.18 (NLCT)]
```

The commands defined in this package are provided to help navigate around the groups within a glossary (see [subsection 1.16.](#)) `\printglossary` (and `\printglossaries`) set `\@glo@type` to the label of the current glossary. This is used to create a unique hypertarget in the event of multiple glossaries.

```
\glsnavhyperlink[⟨type⟩]{⟨label⟩}{⟨text⟩}
```

This command makes `⟨text⟩` a hyperlink to the glossary group whose label is given by `⟨label⟩` for the glossary given by `⟨type⟩`.

`\glsnavhyperlink`

```
7671 \newcommand*{\glsnavhyperlink}[3][\@glo@type]{%
7672   \edef\gls@grplabel{#2}\protected@edef\gls@grptitle{#3}%
7673   \@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`

```
7674 \newcommand*{\glsnavhypertarget}[3][\@glo@type]{%
  Add this group to the aux file for re-run check.
7675   \protected@write\@auxout{}{\string\@gls@hypergroup{#1}{#2}}%
  Add the target.
7676   \@glstarget{glsn:#1@#2}{#3}%
  Check list of know groups to determine if a re-run is required.
7677   \expandafter\let
7678   \expandafter\@gls@list\csname @gls@hypergroup@list@#1\endcsname
  Iterate through list and terminate loop if this group is found.
7679   \@for\@gls@elem:=\@gls@list\do{%
7680     \ifthenelse{\equal{\@gls@elem}{#2}}{\@endfortrue}{}%
  Check if list terminated prematurely.
7681   \if@endfor
7682   \else
  This group was not included in the list, so issue a warning.
7683   \GlossariesWarningNoLine{Navigation panel
7684     for glossary type ‘#1’^^missing group ‘#2’}%
7685   \gdef\gls@hypergroup@rerun{%
7686     \GlossariesWarningNoLine{Navigation panel
7687       has changed. Rerun LaTeX}}%
7688   \fi
7689 }
```

`\gls@hypergroup@rerun` Give a warning at the end if re-run required

```
7690 \let\gls@hypergroup@rerun\relax
7691 \AtEndDocument{\gls@hypergroup@rerun}
```

`\@gls@hypergroup` This adds to (or creates) the command `\@gls@hypergroup@list@<glossary type>` which lists all groups for a given glossary, so that the navigation bar only contains those groups that are present. However it requires at least 2 runs to ensure the information is up-to-date.

```
7692 \newcommand*{\@gls@hypergroup}[2]{%
```

```

7693 \@ifundefined{@gls@hypergrouplist@#1}{%
7694   \expandafter\xdef\csname @gls@hypergrouplist@#1\endcsname{#2}%
7695 }{%
7696   \expandafter\let\expandafter\@gls@tmp
7697     \csname @gls@hypergrouplist@#1\endcsname
7698   \expandafter\xdef\csname @gls@hypergrouplist@#1\endcsname{%
7699     \@gls@tmp,#2}%
7700 }%
7701 }

```

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`

```

7702 \newcommand*{\glsnavigation}{%
7703 \def\@gls@between{}%
7704 \@ifundefined{@gls@hypergrouplist@\@glo@type}{%
7705   \def\@gls@list{}%
7706 }{%
7707   \expandafter\let\expandafter\@gls@list
7708     \csname @gls@hypergrouplist@\@glo@type\endcsname
7709 }%
7710 \@for\@gls@tmp:=\@gls@list\do{%
7711   \@gls@between

7712   \@gls@getgrouptitle{\@gls@tmp}{\@gls@grptitle}%
7713   \glsnavhyperlink{\@gls@tmp}{\@gls@grptitle}%
7714   \let\@gls@between\glshypernavsep%
7715 }%
7716 }

```

`\glshypernavsep` Separator for the hyper navigation bar.

```

7717 \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`

```

7718 \newcommand*{\glssymbolnav}{%
7719 \glsnavhyperlink{glssymbols}{\glsgetgrouptitle{glssymbols}}%
7720 \glshypernavsep
7721 \glsnavhyperlink{glsnumbers}{\glsgetgrouptitle{glsnumbers}}%
7722 \glshypernavsep
7723 }

```

3.2 In-line Style (glossary-inline.sty)

This defines an in-line style where the entries are comma-separated with just the name and description displayed.

```
7724 \ProvidesPackage{glossary-inline}[2015/09/09 v4.18 (NLCT)]
```

`inline` Define the inline style.

```
7725 \newglossarystyle{inline}{%
```

Start of glossary sets up first empty separator between entries. (This is then changed by `\glossentry`)

```
7726 \renewenvironment{theglossary}{%
```

```
7727 {%
```

```
7728 \def\gls@inlinesep{}}%
```

```
7729 \def\gls@inlinesubsep{}}%
```

```
7730 \def\gls@inlinepostchild{}}%
```

```
7731 }%
```

```
7732 {\glspostinline}}%
```

No header:

```
7733 \renewcommand*{\glossaryheader}{}%
```

No group headings (if heading is required, add `\glsinlinedopostchild` to start definition in case heading follows a child entry):

```
7734 \renewcommand*{\glsgroupheading}[1]{}%
```

Just display separator followed by name and description:

```
7735 \renewcommand{\glossentry}[2]{%
```

```
7736 \glsinlinedopostchild
```

```
7737 \gls@inlinesep
```

```
7738 \glsentryitem{##1}}%
```

```
7739 \glsinlinenameformat{##1}{%
```

```
7740 \glossentryname{##1}}%
```

```
7741 }%
```

```
7742 \ifglshasdescsuppressed{##1}}%
```

```
7743 {%
```

```
7744 \glsinlineemptydescformat
```

```
7745 {%
```

```
7746 \glossentrysymbol{##1}}%
```

```
7747 }%
```

```
7748 {%
```

```
7749 ##2}}%
```

```
7750 }%
```

```
7751 }%
```

```
7752 {%
```

```
7753 \ifglshasdesc{##1}}%
```

```
7754 {\glsinlinedescformat{\glossentrydesc{##1}}{\glossentrysymbol{##1}}{##2}}%
```

```
7755 {\glsinlineemptydescformat{\glossentrysymbol{##1}}{##2}}%
```

```
7756 }%
```

```
7757 \ifglshaschildren{##1}}%
```

```

7758   {%
7759     \glsresetsubentrycounter
7760     \glsinlineparentchildseparator
7761     \def\gls@inlinesubsep{}%
7762     \def\gls@inlinepostchild{\glsinlinepostchild}%
7763   }%
7764   {}%
7765   \def\gls@inlinesep{\glsinlineseparator}%
7766 }%

```

Sub-entries display description:

```

7767 \renewcommand{\subglossentry}[3]{%
7768   \gls@inlinesubsep%
7769   \glsinlinesubnameformat{##2}%
7770   \glossentryname{##2}%
7771   \glssubentryitem{##2}%
7772   \glsinlinesubdescformat{\glossentrydesc{##2}}{\glossentrysymbol{##2}}{##3}%
7773   \def\gls@inlinesubsep{\glsinlinesubseparator}%
7774 }%

```

Nothing special between groups:

```

7775 \renewcommand*{\glsgroupskip}{}%
7776 }

```

`\glsinlinedopostchild`

```

7777 \newcommand*{\glsinlinedopostchild}{%
7778   \gls@inlinepostchild
7779   \def\gls@inlinepostchild{}%
7780 }

```

`\glsinlineseparator` Separator to use between entries.

```

7781 \newcommand*{\glsinlineseparator}{;\space}

```

`\glsinlinesubseparator` Separator to use between sub-entries.

```

7782 \newcommand*{\glsinlinesubseparator}{,\space}

```

`\glsinlineparentchildseparator` Separator to use between parent and children.

```

7783 \newcommand*{\glsinlineparentchildseparator}{:\space}

```

`\glsinlinepostchild` Hook to use between child and next entry

```

7784 \newcommand*{\glsinlinepostchild}{}

```

`\glspostinline` Terminator for inline glossary.

```

7785 \newcommand*{\glspostinline}{\glspostdescription\space}

```

`\glsinlinenameformat` Formats the name of the entry (first argument label, second argument name):

```

7786 \newcommand*{\glsinlinenameformat}[2]{\glstarget{#1}{#2}}

```

`\glsinlinedescformat` Formats the entry's description, symbol and location list:

```

7787 \newcommand*{\glsinlinedescformat}[3]{\space#1}

```

`\lineemptydescformat` Formats the entry's symbol and location list when the description is empty:

```
7788 \newcommand*{\glsinlineemptydescformat}[2]{}
```

`\inlinesubnameformat` Formats the name of the subentry (first argument label, second argument name):

```
7789 \newcommand*{\glsinlinesubnameformat}[2]{\glsstarget{#1}{}}
```

`\inlinesubdescformat` Formats the subentry's description, symbol and location list:

```
7790 \newcommand*{\glsinlinesubdescformat}[3]{#1}
```

3.3 List Style (glossary-list.sty)

The style file defines glossary styles that use the description environment. Note that since the entry name is placed in the optional argument to the `\item` command, it will appear in a bold font by default.

```
7791 \ProvidesPackage{glossary-list}[2015/09/09 v4.18 (NLCT)]
```

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```
7792 \providecommand{\indexspace}{%
```

```
7793   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
```

```
7794 }
```

`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.

```
7795 \newglossarystyle{list}{%
```

Use description environment:

```
7796   \renewenvironment{theglossary}{%
```

```
7797     {\begin{description}}{\end{description}}%
```

No header at the start of the environment:

```
7798   \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
7799   \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries start a new item in the list:

```
7800   \renewcommand*{\glossentry}[2]{%
```

```
7801     \item[\glsentryitem{##1}]%
```

```
7802       \glsstarget{##1}{\glossentryname{##1}}
```

```
7803       \glossentrydesc{##1}\glspostdescription\space ##2}%
```

Sub-entries continue on the same line:

```
7804 \renewcommand*{\subglossentry}[3]{%
7805   \glssubentryitem{##2}%
7806   \glstarget{##2}{\strut}%
7807   \glossentrydesc{##2}\glspostdescription\space ##3.}%
7808 % \end{macrocode}
7809 % Add vertical space between groups:
7810 %\changes{3.03}{2012/09/21}{added check for glsnogroupskip}
7811 % \begin{macrocode}
7812 \renewcommand*{\glsgroupskip}{\ifglsgroupskip\else\indexspace\fi}%
7813 }
```

`listgroup` The `listgroup` style is like the `list` style, but the glossary groups have headings.

```
7814 \newglossarystyle{listgroup}{%
  Base it on the list style:
7815 \setglossarystyle{list}%
  Each group has a heading:
7816 \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.

```
7817 \newglossarystyle{listhypergroup}{%
  Base it on the list style:
7818 \setglossarystyle{list}%
  Add navigation links at the start of the environment:
7819 \renewcommand*{\glossaryheader}{%
7820 \item[\glsnavigation]}%
  Each group has a heading with a hypertext:
7821 \renewcommand*{\glsgroupheading}[1]{%
7822 \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.

```
7823 \newglossarystyle{altlist}{%
  Base it on the list style:
7824 \setglossarystyle{list}%
  Main (level 0) entries start a new item in the list with a line break after the entry name:
7825 \renewcommand*{\glossentry}[2]{%
7826 \item[\glsentryitem{##1}%
7827 \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.

```
7828     \mbox{}\par\nobreak\@afterheading
7829     \glossentrydesc{##1}\glspostdescription\space ##2}%
```

Sub-entries start a new paragraph:

```
7830 \renewcommand{\subglossentry}[3]{%
7831   \par
7832   \glssubentryitem{##2}%
7833   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space ##3}%
7834 }
```

`altlistgroup` The `altlistgroup` glossary style is like the `altlist` style, but the glossary groups have headings.

```
7835 \newglossarystyle{altlistgroup}{%
  Base it on the altlist style:
7836   \setglossarystyle{altlist}%
  Each group has a heading:
7837   \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.

```
7838 \newglossarystyle{altlisthypergroup}{%
  Base it on the altlist style:
7839   \setglossarystyle{altlist}%
  Add navigation links at the start of the environment:
7840   \renewcommand*{\glossaryheader}{%
7841     \item[\glsnavigation]}%
  Each group has a heading with a hypertext:
7842   \renewcommand*{\glsgroupheading}[1]{%
7843     \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.

```
7844 \newglossarystyle{listdotted}{%
  Base it on the list style:
7845   \setglossarystyle{list}%
```

Each main (level 0) entry starts a new item:

```
7846 \renewcommand*{\glossentry}[2]{%
7847   \item[]\makebox[\glslistdottedwidth][l]{%
7848     \glstryitem{##1}%
7849     \glstarget{##1}{\glossentryname{##1}}%
7850     \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##1}}%
```

Sub entries have the same format as main entries:

```
7851 \renewcommand*{\subglossentry}[3]{%
7852   \item[]\makebox[\glslistdottedwidth][l]{%
7853     \glssubentryitem{##2}%
7854     \glstarget{##2}{\glossentryname{##2}}%
7855     \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##2}}%
7856 }
```

`\glslistdottedwidth`

```
7857 \newlength\glslistdottedwidth
7858 \setlength{\glslistdottedwidth}{.5\hsize}
```

`sublistdotted` This style is similar to the `glostylelistdotted` style, except that the main entries just have the name displayed.

```
7859 \newglossarystyle{sublistdotted}{%
```

Base it on the `listdotted` style:

```
7860 \setglossarystyle{listdotted}%,
```

Main (level 0) entries just display the name:

```
7861 \renewcommand*{\glossentry}[2]{%
7862   \item[\glstryitem{##1}\glstarget{##1}{\glossentryname{##1}}]}%
7863 }
```

3.4 Glossary Styles using `longtable` (the `glossary-long` package)

The glossary styles defined in the package used the `longtable` environment in the glossary.

```
7864 \ProvidesPackage{glossary-long}[2015/09/09 v4.18 (NLCT)]
```

Requires the package:

```
7865 \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`.)

```
7866 \@ifundefined{glsdescwidth}{%
7867   \newlength\glsdescwidth
7868   \setlength{\glsdescwidth}{0.6\hsize}
7869 }{}
```


`\glspagelistwidth` This is a length that governs the width of the page list column.

```
7870 \@ifundefined{glspagelistwidth}{%
7871   \newlength{glspagelistwidth
7872   \setlength{glspagelistwidth}{0.1\hsize}
7873 }
```

`long` The long glossary style command which uses the `longtable` environment:

```
7874 \newglossarystyle{long}{%
```

Use `longtable` with two columns:

```
7875   \renewenvironment{theglossary}{%
7876     \begin{longtable}[lp{glstdescwidth}]{%
7877     \end{longtable}}%
```

Do nothing at the start of the environment:

```
7878   \renewcommand*{\glossaryheader}{}%
```

No heading between groups:

```
7879   \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries displayed in a row:

```
7880   \renewcommand{\glossentry}[2]{%
7881     \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
7882     \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline
7883   }%
```

Sub entries displayed on the following row without the name:

```
7884   \renewcommand{\subglossentry}[3]{%
7885     &
7886     \glssubentryitem{##2}%
7887     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
7888     ##3\tabularnewline
7889   }%
```

Blank row between groups:

```
7890   \renewcommand*{\glsgroupskip}{\ifglsgnোগroupskip\else &
7891   \tabularnewline\fi}%
7892 }
```

`longborder` The `longborder` style is like the above, but with horizontal and vertical lines:

```
7893 \newglossarystyle{longborder}{%
```

Base it on the `glostylelong` style:

```
7894   \setglossarystyle{long}%
```

Use `longtable` with two columns with vertical lines between each column:

```
7895   \renewenvironment{theglossary}{%
7896     \begin{longtable}[|l|p{glstdescwidth}|]{\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
7897   \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
7898 }
```

`longheader` The `longheader` style is like the `long` style but with a header:

```
7899 \newglossarystyle{longheader}{%
    Base it on the glostylelong style:
7900 \setglossarystyle{long}%
    Set the table's header:
7901 \renewcommand*{\glossaryheader}{%
7902 \bfseries \entryname & \bfseries \descriptionname\tabularnewline\endhead}%
7903 }
```

`longheaderborder` The `longheaderborder` style is like the `long` style but with a header and border:

```
7904 \newglossarystyle{longheaderborder}{%
    Base it on the glostylelongborder style:
7905 \setglossarystyle{longborder}%
    Set the table's header and add horizontal line to table's foot:
7906 \renewcommand*{\glossaryheader}{%
7907 \hline\bfseries \entryname & \bfseries
7908 \descriptionname\tabularnewline\hline
7909 \endhead
7910 \hline\endfoot}%
7911 }
```

`long3col` The `long3col` style is like `long` but with 3 columns

```
7912 \newglossarystyle{long3col}{%
    Use a longtable with 3 columns:
7913 \renewenvironment{theglossary}{%
7914 {\begin{longtable}{lp{\glsdescwidth}p{\glspagerlistwidth}}}%
7915 {\end{longtable}}%
    No table header:
7916 \renewcommand*{\glossaryheader}{}%
    No headings between groups:
7917 \renewcommand*{\glsgroupheading}[1]{}%
    Main (level 0) entries on a row (name in first column, description in second
    column, page list in last column):
7918 \renewcommand{\glossentry}[2]{%
7919 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
7920 \glossentrydesc{##1} & ##2\tabularnewline
7921 }%
    Sub-entries on a separate row (no name, description in second column, page
    list in third column):
7922 \renewcommand{\subglossentry}[3]{%
7923 &
7924 \glssubentryitem{##2}%
7925 \glstarget{##2}{\strut}\glossentrydesc{##2} &
```

```
7926     ##3\tabularnewline
7927   }%
```

Blank row between groups:

```
7928   \renewcommand*{\glsgroupskip}{%
7929     \ifglsnogroupskip\else & &\tabularnewline\fi}%
7930 }
```

`long3colborder` The `long3colborder` style is like the `long3col` style but with a border:

```
7931 \newglossarystyle{long3colborder}{%
```

Base it on the `glostylelong3col` style:

```
7932   \setglossarystyle{long3col}%
```

Use a `longtable` with 3 columns with vertical lines around them:

```
7933   \renewenvironment{theglossary}{%
7934     {\begin{longtable}{|l|p{\glsdescwidth}|p{\glspagelistwidth}|}}%
7935     {\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
7936   \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
7937 }
```

`long3colheader` The `long3colheader` style is like `long3col` but with a header row:

```
7938 \newglossarystyle{long3colheader}{%
```

Base it on the `glostylelong3col` style:

```
7939   \setglossarystyle{long3col}%
```

Set the table's header:

```
7940   \renewcommand*{\glossaryheader}{%
7941     \bfseries\entryname&\bfseries\descriptionname&
7942     \bfseries\pagelistname\tabularnewline\endhead}%
7943 }
```

`long3colheaderborder` The `long3colheaderborder` style is like the above but with a border

```
7944 \newglossarystyle{long3colheaderborder}{%
```

Base it on the `glostylelong3colborder` style:

```
7945   \setglossarystyle{long3colborder}%
```

Set the table's header and add horizontal line at table's foot:

```
7946   \renewcommand*{\glossaryheader}{%
7947     \hline
7948     \bfseries\entryname&\bfseries\descriptionname&
7949     \bfseries\pagelistname\tabularnewline\hline\endhead
7950     \hline\endfoot}%
7951 }
```

`long4col` The `long4col` style has four columns where the third column contains the value of the associated symbol key.

```
7952 \newglossarystyle{long4col}{%
```

Use a longtable with 4 columns:

```
7953 \renewenvironment{theglossary}%  
7954   {\begin{longtable}{llll}}%  
7955   {\end{longtable}}%
```

No table header:

```
7956 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
7957 \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):

```
7958 \renewcommand{\glossentry}[2]{%  
7959   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &  
7960   \glossentrydesc{##1} &  
7961   \glossentrysymbol{##1} &  
7962   ##2\tabularnewline  
7963 }%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
7964 \renewcommand{\subglossentry}[3]{%  
7965   &  
7966   \glssubentryitem{##2}%  
7967   \glstarget{##2}{\strut}\glossentrydesc{##2} &  
7968   \glossentrysymbol{##2} & ##3\tabularnewline  
7969 }%
```

Blank row between groups:

```
7970 \renewcommand*{\glsgroupskip}{%  
7971   \ifglsgnogroupskip\else & & \tabularnewline\fi}%  
7972 }
```

`long4colheader` The `long4colheader` style is like `long4col` but with a header row.

```
7973 \newglossarystyle{long4colheader}{%
```

Base it on the `glostylelong4col` style:

```
7974 \setglossarystyle{long4col}%
```

Table has a header:

```
7975 \renewcommand*{\glossaryheader}{%  
7976   \bfseries\entryname&\bfseries\descriptionname&  
7977   \bfseries \symbolname&  
7978   \bfseries\pagelistname\tabularnewline\endhead}%  
7979 }
```

`long4colborder` The `long4colborder` style is like `long4col` but with a border.

```
7980 \newglossarystyle{long4colborder}{%
```

Base it on the `glostylelong4col` style:

```
7981 \setglossarystyle{long4col}%
```

Use a longtable with 4 columns surrounded by vertical lines:

```
7982 \renewenvironment{theglossary}%  
7983   {\begin{longtable}{|l|l|l|l|}}%  
7984   {\end{longtable}}%
```

Add horizontal lines to the head and foot of the table:

```
7985 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%  
7986 }
```

`long4colheaderborder` The `long4colheaderborder` style is like the above but with a border.

```
7987 \newglossarystyle{long4colheaderborder}{%
```

Base it on the `glostylelong4col` style:

```
7988 \setglossarystyle{long4col}%
```

Use a longtable with 4 columns surrounded by vertical lines:

```
7989 \renewenvironment{theglossary}%  
7990   {\begin{longtable}{|l|l|l|l|}}%  
7991   {\end{longtable}}%
```

Add table header and horizontal line at the table's foot:

```
7992 \renewcommand*{\glossaryheader}{%  
7993   \hline\bfseries\entryname&\bfseries\descriptionname&  
7994   \bfseries \symbolname&  
7995   \bfseries\pagelistname\tabularnewline\hline\endhead  
7996   \hline\endfoot}%  
7997 }
```

`altlong4col` The `altlong4col` style is like the `long4col` style but can have multiline descriptions and page lists.

```
7998 \newglossarystyle{altlong4col}{%
```

Base it on the `glostylelong4col` style:

```
7999 \setglossarystyle{long4col}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8000 \renewenvironment{theglossary}%  
8001   {\begin{longtable}{lp{\glsdescwidth}lp{\glspagelistwidth}}%  
8002   {\end{longtable}}%  
8003 }
```

`altlong4colheader` The `altlong4colheader` style is like `altlong4col` but with a header row.

```
8004 \newglossarystyle{altlong4colheader}{%
```

Base it on the `glostylelong4colheader` style:

```
8005 \setglossarystyle{long4colheader}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8006 \renewenvironment{theglossary}%
```

```

8007   {\begin{longtable}{lp{\glsdescwidth}lp{\glspagelistwidth}}}%
8008   {\end{longtable}}%
8009 }

```

`altlong4colborder` The `altlong4colborder` style is like `altlong4col` but with a border.

```

8010 \newglossarystyle{altlong4colborder}{%
      Base it on the glostylelong4colborder style:
8011   \setglossarystyle{long4colborder}%
      Use a longtable with 4 columns where the second and last columns may have
      multiple lines in each row:
8012   \renewenvironment{theglossary}%
8013     {\begin{longtable}{|l|p{\glsdescwidth}|l|p{\glspagelistwidth}|}}%
8014     {\end{longtable}}%
8015 }

```

`long4colheaderborder` The `altlong4colheaderborder` style is like the above but with a header as well as a border.

```

8016 \newglossarystyle{altlong4colheaderborder}{%
      Base it on the glostylelong4colheaderborder style:
8017   \setglossarystyle{long4colheaderborder}%
      Use a longtable with 4 columns where the second and last columns may have
      multiple lines in each row:
8018   \renewenvironment{theglossary}%
8019     {\begin{longtable}{|l|p{\glsdescwidth}|l|p{\glspagelistwidth}|}}%
8020     {\end{longtable}}%
8021 }

```

3.5 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.

```

8022 \ProvidesPackage{glossary-longragged}[2015/09/09 v4.18 (NLCT)]

```

Requires the package:

```

8023 \RequirePackage{array}

```

Requires the package:

```

8024 \RequirePackage{longtable}

```

`\glsdescwidth` This is a length that governs the width of the description column. This may have already been defined.

```

8025 \@ifundefined{glsdescwidth}{%
8026   \newlength{glsdescwidth}
8027   \setlength{glsdescwidth}{0.6\hsize}
8028 }{}

```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```
8029 \@ifundefined{glspagelistwidth}{%
8030   \newlength{glspagelistwidth
8031   \setlength{glspagelistwidth}{0.1\hsize}
8032 }{}
```

`longragged` The `longragged` glossary style is like the `long` but uses ragged right formatting for the description column.

```
8033 \newglossarystyle{longragged}{%
```

Use `longtable` with two columns:

```
8034   \renewenvironment{theglossary}{%
8035     \begin{longtable}{1>{\raggedright}p{\glsdescwidth}}%
8036     {\end{longtable}}%
```

Do nothing at the start of the environment:

```
8037   \renewcommand*{\glossaryheader}{}%
```

No heading between groups:

```
8038   \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries displayed in a row:

```
8039   \renewcommand{\glossentry}[2] {%
8040     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8041     \glossentrydesc{##1}\glspostdescription\space ##2%
8042     \tabularnewline
8043   }%
```

Sub entries displayed on the following row without the name:

```
8044   \renewcommand{\subglossentry}[3] {%
8045     &
8046     \glssubentryitem{##2}%
8047     \glstarget{##2}{\strut}\glossentrydesc{##2}%
8048     \glspostdescription\space ##3%
8049     \tabularnewline
8050   }%
```

Blank row between groups:

```
8051   \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else & \tabularnewline\fi}%
8052 }
```

`longraggedborder` The `longraggedborder` style is like the above, but with horizontal and vertical lines:

```
8053 \newglossarystyle{longraggedborder}{%
```

Base it on the `glostylelongragged` style:

```
8054   \setglossarystyle{longragged}%,
```

Use `longtable` with two columns with vertical lines between each column:

```
8055   \renewenvironment{theglossary}{%
8056     \begin{longtable}{|1|>{\raggedright}p{\glsdescwidth}|}%
8057     {\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
8058 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8059 }
```

`longraggedheader` The `longraggedheader` style is like the `longragged` style but with a header:

```
8060 \newglossarystyle{longraggedheader}{%
```

Base it on the `glostylelongragged` style:

```
8061 \setglossarystyle{longragged}{%
```

Set the table's header:

```
8062 \renewcommand*{\glossaryheader}{%
8063 \bfseries \entryname & \bfseries \descriptionname
8064 \tabularnewline\endhead}%
8065 }
```

`longraggedheaderborder` The `longraggedheaderborder` style is like the `longragged` style but with a header and border:

```
8066 \newglossarystyle{longraggedheaderborder}{%
```

Base it on the `glostylelongraggedborder` style:

```
8067 \setglossarystyle{longraggedborder}{%
```

Set the table's header and add horizontal line to table's foot:

```
8068 \renewcommand*{\glossaryheader}{%
8069 \hline\bfseries \entryname & \bfseries \descriptionname
8070 \tabularnewline\hline
8071 \endhead
8072 \hline\endfoot}%
8073 }
```

`longragged3col` The `longragged3col` style is like `longragged` but with 3 columns

```
8074 \newglossarystyle{longragged3col}{%
```

Use a `longtable` with 3 columns:

```
8075 \renewenvironment{theglossary}{%
8076 {\begin{longtable}{1>{\raggedright}p{\glsdescwidth}%
8077 >{\raggedright}p{\glspagelistwidth}}}%
8078 {\end{longtable}}%
```

No table header:

```
8079 \renewcommand*{\glossaryheader}{}%
```

No headings between groups:

```
8080 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
8081 \renewcommand{\glossentry}[2]{%
8082 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8083 \glossentrydesc{##1} & ##2\tabularnewline
8084 }%
```


Sub-entries on a separate row (no name, description in second column, page list in third column):

```
8085 \renewcommand{\subglossentry}[3]{%
8086     &
8087     \glssubentryitem{##2}%
8088     \glstarget{##2}{\strut}\glossentrydesc{##2} &
8089     ##3\tabularnewline
8090 }%
```

Blank row between groups:

```
8091 \renewcommand*{\glsgroupskip}{%
8092     \ifglsnogroupskip\else & &\tabularnewline\fi}%
8093 }
```

`longragged3colborder` The `longragged3colborder` style is like the `longragged3col` style but with a border:

```
8094 \newglossarystyle{longragged3colborder}{%
```

Base it on the `glostylelongragged3col` style:

```
8095 \setglossarystyle{longragged3col}{%
```

Use a `longtable` with 3 columns with vertical lines around them:

```
8096 \renewenvironment{theglossary}{%
8097     {\begin{longtable}{|l|>{\raggedright}p{\glsdescwidth}|%
8098     >{\raggedright}p{\glspagelistwidth}|}%
8099     {\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
8100 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8101 }
```

`longragged3colheader` The `longragged3colheader` style is like `longragged3col` but with a header row:

```
8102 \newglossarystyle{longragged3colheader}{%
```

Base it on the `glostylelongragged3col` style:

```
8103 \setglossarystyle{longragged3col}{%
```

Set the table's header:

```
8104 \renewcommand*{\glossaryheader}{%
8105     \bfseries\entryname&\bfseries\descriptionname&
8106     \bfseries\pagelistname\tabularnewline\endhead}%
8107 }
```

`longragged3colheaderborder` The `longragged3colheaderborder` style is like the above but with a border

```
8108 \newglossarystyle{longragged3colheaderborder}{%
```

Base it on the `glostylelongragged3colborder` style:

```
8109 \setglossarystyle{longragged3colborder}{%
```

Set the table's header and add horizontal line at table's foot:

```
8110 \renewcommand*{\glossaryheader}{%
8111 \hline
8112 \bfseries\entryname&\bfseries\descriptionname&
8113 \bfseries\pagelistname\tabularnewline\hline\endhead
8114 \hline\endfoot}%
8115 }
```

`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.

```
8116 \newglossarystyle{altlongragged4col}{%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8117 \renewenvironment{theglossary}%
8118 {\begin{longtable}[1>{\raggedright}p{\glsdescwidth}1%
8119 >{\raggedright}p{\glspagelistwidth}}}%
8120 {\end{longtable}}%
```

No table header:

```
8121 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8122 \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):

```
8123 \renewcommand{\glossentry}[2]{%
8124 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8125 \glossentrydesc{##1} & \glossentrysymbol{##1} &
8126 ##2\tabularnewline
8127 }%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
8128 \renewcommand{\subglossentry}[3]{%
8129 &
8130 \glssubentryitem{##2}%
8131 \glstarget{##2}{\strut}\glossentrydesc{##2} &
8132 \glossentrysymbol{##2} & ##3\tabularnewline
8133 }%
```

Blank row between groups:

```
8134 \renewcommand*{\glsgroupskip}{%
8135 \ifglsnogroupskip\else & & \tabularnewline\fi}%
8136 }
```

`ongragged4colheader` The `altlongragged4colheader` style is like `altlongragged4col` but with a header row.

```
8137 \newglossarystyle{altlongragged4colheader}{%
```

Base it on the `glostylealtlongragged4col` style:

```
8138 \setglossarystyle{altlongragged4col}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8139 \renewenvironment{theglossary}%  
8140   {\begin{longtable}{1>{\raggedright}p{\glsdescwidth}1%  
8141     >{\raggedright}p{\glspagelistwidth}}}%  
8142   {\end{longtable}}%
```

Table has a header:

```
8143 \renewcommand*{\glossaryheader}{%  
8144   \bfseries\entryname&\bfseries\descriptionname&  
8145   \bfseries \symbolname&  
8146   \bfseries\pagelistname\tabularnewline\endhead}%  
8147 }
```

`altlongragged4colborder` The `altlongragged4colborder` style is like `altlongragged4col` but with a border.

```
8148 \newglossarystyle{altlongragged4colborder}{%
```

Base it on the `glostylealtlongragged4col` style:

```
8149 \setglossarystyle{altlongragged4col}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8150 \renewenvironment{theglossary}%  
8151   {\begin{longtable}{|1|>{\raggedright}p{\glsdescwidth}|1|%  
8152     >{\raggedright}p{\glspagelistwidth}|}%  
8153   {\end{longtable}}%
```

Add horizontal lines to the head and foot of the table:

```
8154 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%  
8155 }
```

`altlongragged4colheaderborder` The `altlongragged4colheaderborder` style is like the above but with a header as well as a border.

```
8156 \newglossarystyle{altlongragged4colheaderborder}{%
```

Base it on the `glostylealtlongragged4col` style:

```
8157 \setglossarystyle{altlongragged4col}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8158 \renewenvironment{theglossary}%  
8159   {\begin{longtable}{|1|>{\raggedright}p{\glsdescwidth}|1|%  
8160     >{\raggedright}p{\glspagelistwidth}|}%  
8161   {\end{longtable}}%
```

Add table header and horizontal line at the table's foot:

```
8162 \renewcommand*{\glossaryheader}{%  
8163   \hline\bfseries\entryname&\bfseries\descriptionname&
```

```

8164 \bfseries \symbolname&
8165 \bfseries\pagelistname\tabularnewline\hline\endhead
8166 \hline\endfoot}%
8167 }

```

3.6 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.

```
8168 \ProvidesPackage{glossary-mcols}[2015/09/09 v4.18 (NLCT)]
```

Required packages:

```

8169 \RequirePackage{multicol}
8170 \RequirePackage{glossary-tree}

```

`\indexspace` The are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```

8171 \providecommand{\indexspace}{%
8172 \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
8173 }

```

`\glsmcols` Define macro in which to store the number of columns. (Defaults to 2.)

```
8174 \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.)

```

8175 \newglossarystyle{mcolindex}{%
8176 \setglossarystyle{index}%
8177 \renewenvironment{theglossary}%
8178 {%
8179 \begin{multicols}{\glsmcols}
8180 \setlength{\parindent}{0pt}%
8181 \setlength{\parskip}{0pt plus 0.3pt}%
8182 \let\item\@idxitem}%
8183 {\end{multicols}}%
8184 }

```

`mcolindexgroup` As `mcolindex` but has headings:

```

8185 \newglossarystyle{mcolindexgroup}{%
8186 \setglossarystyle{mcolindex}%
8187 \renewcommand*\glsgroupheading[1]{%
8188 \item\textbf{\glsgroupheading{##1}}\indexspace}%
8189 }

```

`mcolindexhypergroup` The `mcolindexhypergroup` style is like the `mcolindexgroup` style but has hyper navigation.

```
8190 \newglossarystyle{mcolindexhypergroup}{%
```

Base it on the `glostylemcolindex` style:

```
8191 \setglossarystyle{mcolindex}%
```

Put navigation links to the groups at the start of the glossary:

```
8192 \renewcommand*{\glossaryheader}{%
8193   \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.

```
8194 \renewcommand*{\glsgroupheading}[1]{%
8195   \item\textbf{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
8196   \indexspace}%
8197 }
```

`mcoltree` Multi-column index style. Same as the `tree`, but puts the glossary in multiple columns.

```
8198 \newglossarystyle{mcoltree}{%
8199   \setglossarystyle{tree}%
8200   \renewenvironment{theglossary}%
8201     {%
8202       \begin{multicols}{\glsncols}
8203       \setlength{\parindent}{0pt}%
8204       \setlength{\parskip}{0pt plus 0.3pt}%
8205     }%
8206     {\end{multicols}}%
8207 }
```

`mcoltreegroup` Like the `mcoltree` style but the glossary groups have headings.

```
8208 \newglossarystyle{mcoltreegroup}{%
      Base it on the glostylemcoltree style:
8209   \setglossarystyle{mcoltree}%
      Each group has a heading (in bold) followed by a vertical gap):
8210   \renewcommand{\glsgroupheading}[1]{\par
8211     \noindent\textbf{\glsgetgrouptitle{##1}}\par\indexspace}%
8212 }
```

`mcoltreehypergroup` The `mcoltreehypergroup` style is like the `treegroup` style, but has a set of links to the groups at the start of the glossary.

```
8213 \newglossarystyle{mcoltreehypergroup}{%
      Base it on the glostylemcoltree style:
8214   \setglossarystyle{mcoltree}%
      Put navigation links to the groups at the start of the theglossary environment:
8215   \renewcommand*{\glossaryheader}{%
8216     \par\noindent\textbf{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
8217 \renewcommand*{\glsgroupheading}[1]{%
8218   \par\noindent
8219   \textbf{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8220   \indexspace}%
8221 }
```

`mcoltreename` Multi-column index style. Same as the `treename`, but puts the glossary in multiple columns.

```
8222 \newglossarystyle{mcoltreename}{%
8223   \setglossarystyle{treename}%
8224   \renewenvironment{theglossary}%
8225   {%
8226     \begin{multicols}{\glsncols}
8227     \setlength{\parindent}{0pt}%
8228     \setlength{\parskip}{0pt plus 0.3pt}%
8229   }%
8230   {\end{multicols}}%
8231 }
```

`mcoltreenamegroup` Like the `mcoltreename` style but the glossary groups have headings.

```
8232 \newglossarystyle{mcoltreenamegroup}{%
8233   Base it on the glostylemcoltreename style:
8234   \setglossarystyle{mcoltreename}%
8235   Give each group a heading:
8236   \renewcommand{\glsgroupheading}[1]{\par
8237     \noindent\textbf{\glsgetgrouptitle{##1}}\par\indexspace}%
8238 }
```

`treenamehypergroup` The `mcoltreenamehypergroup` style is like the `mcoltreenamegroup` style, but has a set of links to the groups at the start of the glossary.

```
8237 \newglossarystyle{mcoltreenamehypergroup}{%
8238   Base it on the glostylemcoltreename style:
8239   \setglossarystyle{mcoltreename}%
8240   Put navigation links to the groups at the start of the theglossary environment:
8241   \renewcommand*{\glossaryheader}{%
8242     \par\noindent\textbf{\glsnavigation}\par\indexspace}%
8243   Each group has a heading (in bold with a target) followed by a vertical gap):
8244   \renewcommand*{\glsgroupheading}[1]{%
8245     \par\noindent
8246     \textbf{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8247     \indexspace}%
8248 }
```

`mcolalmtree` Multi-column index style. Same as the `almtree`, but puts the glossary in multiple columns.

```
8246 \newglossarystyle{mcolalmtree}{%
8247   \setglossarystyle{almtree}%
8248   \renewenvironment{theglossary}%
8249   {%

8250     \begin{multicols}{\glsmcols}
8251     \def\@gls@prevlevel{-1}%
8252     \mbox{\par
8253   }%
8254   {\par\end{multicols}}%
8255 }
```

`mcolalmtreegroup` Like the `mcolalmtree` style but the glossary groups have headings.

```
8256 \newglossarystyle{mcolalmtreegroup}{%
      Base it on the glostylemcolalmtree style:
8257   \setglossarystyle{mcolalmtree}%
      Give each group a heading.
8258   \renewcommand{\glsgroupheading}[1]{\par
8259     \def\@gls@prevlevel{-1}%
8260     \hangindent0pt\relax
8261     \parindent0pt\relax
8262     \textbf{\glsgetgrouptitle{##1}}\par\indexspace}%
8263 }
```

`colalmtreehypergroup` The `mcolalmtreehypergroup` style is like the `mcolalmtreegroup` style, but has a set of links to the groups at the start of the glossary.

```
8264 \newglossarystyle{mcolalmtreehypergroup}{%
      Base it on the glostylemcolalmtree style:
8265   \setglossarystyle{mcolalmtree}%
      Put the navigation links in the header
8266   \renewcommand*\glossaryheader{%
8267     \par
8268     \def\@gls@prevlevel{-1}%
8269     \hangindent0pt\relax
8270     \parindent0pt\relax
8271     \textbf{\glsnavigation}\par\indexspace}%
      Put a hypertarget at the start of each group
8272   \renewcommand*\glsgroupheading[1]{%
8273     \par
8274     \def\@gls@prevlevel{-1}%
8275     \hangindent0pt\relax
8276     \parindent0pt\relax
8277     \textbf{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8278     \indexspace}}
```

3.7 Glossary Styles using supertabular environment (glossary-super package)

The glossary styles defined in the package use the supertabular environment.

```
8279 \ProvidesPackage{glossary-super}[2015/09/09 v4.18 (NLCT)]
```

Requires the package:

```
8280 \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.

```
8281 \@ifundefined{glsdescwidth}{%
8282   \newlength{glsdescwidth}
8283   \setlength{glsdescwidth}{0.6\hsize}
8284 }{}
```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined if has been loaded.

```
8285 \@ifundefined{glspagelistwidth}{%
8286   \newlength{glspagelistwidth}
8287   \setlength{glspagelistwidth}{0.1\hsize}
8288 }{}
```

`super` The super glossary style uses the supertabular environment (it uses lengths defined in the package.)

```
8289 \newglossarystyle{super}{%
```

Put the glossary in a supertabular environment with two columns and no head or tail:

```
8290   \renewenvironment{theglossary}%
8291     {\tablehead{ }\tabletail{ }}%
8292     \begin{supertabular}{lp{glsdescwidth}}%
8293     {\end{supertabular}}%
```

Do nothing at the start of the table:

```
8294   \renewcommand*{glossaryheader}{ }%
```

No group headings:

```
8295   \renewcommand*{glsgroupheading}[1]{ }
```

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```
8296   \renewcommand{glossentry}[2]{%
8297     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8298     \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline
8299   }%
```

Sub entries put in a row (no name, description and page list in second column):

```
8300   \renewcommand{subglossentry}[3]{%
```



```

8301     &
8302     \glssubentryitem{##2}%
8303     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
8304     ##3\tabularnewline
8305 }%

```

Blank row between groups:

```

8306 \renewcommand*{\glsgroupskip}{%
8307   \ifglsgnogroupskip\else & \tabularnewline\fi}%
8308 }

```

superborder The superborder style is like the above, but with horizontal and vertical lines:

```
8309 \newglossarystyle{superborder}{%
```

Base it on the glostylesuper style:

```
8310 \setglossarystyle{super}%
```

Put the glossary in a supertabular environment with two columns and a horizontal line in the head and tail:

```

8311 \renewenvironment{theglossary}%
8312   {\tablehead{\hline}\tabletail{\hline}%
8313    \begin{supertabular}{|l|p{\glsdescwidth}|}%
8314   \end{supertabular}}%
8315 }

```

superheader The superheader style is like the super style, but with a header:

```
8316 \newglossarystyle{superheader}{%
```

Base it on the glostylesuper style:

```
8317 \setglossarystyle{super}%
```

Put the glossary in a supertabular environment with two columns, a header and no tail:

```

8318 \renewenvironment{theglossary}%
8319   {\tablehead{\bfseries \entryname &
8320    \bfseries\descriptionname\tabularnewline}%
8321    \tabletail{}}%
8322   \begin{supertabular}{lp{\glsdescwidth}}%
8323   \end{supertabular}}%
8324 }

```

superheaderborder The superheaderborder style is like the super style but with a header and border:

```
8325 \newglossarystyle{superheaderborder}{%
```

Base it on the glostylesuper style:

```
8326 \setglossarystyle{super}%
```

Put the glossary in a supertabular environment with two columns, a header and horizontal lines above and below the table:

```

8327 \renewenvironment{theglossary}%
8328   {\tablehead{\hline\bfseries \entryname &

```

```

8329     \bfseries \descriptionname\tabularnewline\hline}%
8330     \tabletail{\hline}
8331     \begin{supertabular}{|l|p{\glsdescwidth}|}%
8332     {\end{supertabular}}%
8333 }

```

`super3col` The `super3col` style is like the `super` style, but with 3 columns:

```

8334 \newglossarystyle{super3col}{%
    Put the glossary in a supertabular environment with three columns and no head
    or tail:
8335     \renewenvironment{theglossary}%
8336     {\tablehead{}\tabletail}%
8337     \begin{supertabular}{lp{\glsdescwidth}p{\glspagerlistwidth}}%
8338     {\end{supertabular}}%
    Do nothing at the start of the table:
8339     \renewcommand*{\glossaryheader}{}%
    No group headings:
8340     \renewcommand*{\glsgroupheading}[1]{}%
    Main (level 0) entries on a row (name in first column, description in second
    column, page list in last column):
8341     \renewcommand{\glossentry}[2]{%
8342         \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8343         \glossentrydesc{##1} & ##2\tabularnewline
8344     }%
    Sub entries on a row (no name, description in second column, page list in last
    column):
8345     \renewcommand{\subglossentry}[3]{%
8346         &
8347         \glsesubentryitem{##2}%
8348         \glstarget{##2}{\strut}\glossentrydesc{##2} &
8349         ##3\tabularnewline
8350     }%
    Blank row between groups:
8351     \renewcommand*{\glsgroupskip}{%
8352         \ifglsnogroupskip\else & &\tabularnewline\fi}%
8353 }

```

`super3colborder` The `super3colborder` style is like the `super3col` style, but with a border:

```

8354 \newglossarystyle{super3colborder}{%
    Base it on the glostylesuper3col style:
8355     \setglossarystyle{super3col}%
    Put the glossary in a supertabular environment with three columns and a hori-
    zontal line in the head and tail:
8356     \renewenvironment{theglossary}%

```

```

8357   {\tablehead{\hline}\tabletail{\hline}}%
8358   \begin{supertabular}{|l|p{\glstdescwidth}|p{\glspagelistwidth}|}%
8359   {\end{supertabular}}%
8360 }

```

super3colheader The `super3colheader` style is like the `super3col` style but with a header row:

```

8361 \newglossarystyle{super3colheader}{%
      Base it on the glostylesuper3col style:
8362   \setglossarystyle{super3col}%
      Put the glossary in a supertabular environment with three columns, a header
      and no tail:
8363   \renewenvironment{theglossary}{%
8364     {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8365       \bfseries\pagelistname\tabularnewline}\tabletail{}}%
8366     \begin{supertabular}{|lp{\glstdescwidth}p{\glspagelistwidth}|}%
8367     {\end{supertabular}}%
8368 }

```

super3colheaderborder The `super3colheaderborder` style is like the `super3col` style but with a header and border:

```

8369 \newglossarystyle{super3colheaderborder}{%
      Base it on the glostylesuper3colborder style:
8370   \setglossarystyle{super3colborder}%
      Put the glossary in a supertabular environment with three columns, a header
      with horizontal lines and a horizontal line in the tail:
8371   \renewenvironment{theglossary}{%
8372     {\tablehead{\hline
8373       \bfseries\entryname&\bfseries\descriptionname&
8374       \bfseries\pagelistname\tabularnewline\hline}%
8375     \tabletail{\hline}}%
8376     \begin{supertabular}{|l|p{\glstdescwidth}|p{\glspagelistwidth}|}%
8377     {\end{supertabular}}%
8378 }

```

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.

```

8379 \newglossarystyle{super4col}{%
      Put the glossary in a supertabular environment with four columns and no head
      or tail:
8380   \renewenvironment{theglossary}{%
8381     {\tablehead{} \tabletail{}}%
8382     \begin{supertabular}{|l|l|l|l|}%
8383     \end{supertabular}}%
      Do nothing at the start of the table:
8384   \renewcommand*{\glossaryheader}{}%

```

No group headings:

```
8385 \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:

```
8386 \renewcommand{\glossentry}[2]{%
8387   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8388   \glossentrydesc{##1} &
8389   \glossentrysymbol{##1} & ##3\tabularnewline
8390 }%
```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```
8391 \renewcommand{\subglossentry}[3]{%
8392   &
8393   \glssubentryitem{##2}%
8394   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8395   \glossentrysymbol{##2} & ##3\tabularnewline
8396 }%
```

Blank row between groups:

```
8397 \renewcommand*\glsgroupskip}{%
8398   \ifglsnogroupskip\else & & \tabularnewline\fi}%
8399 }
```

`super4colheader` The `super4colheader` style is like the `super4col` but with a header row.

```
8400 \newglossarystyle{super4colheader}{%
```

Base it on the `glostylesuper4col` style:

```
8401 \setglossarystyle{super4col}%
```

Put the glossary in a `supertabular` environment with four columns, a header and no tail:

```
8402 \renewenvironment{theglossary}{%
8403   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8404     \bfseries\symbolname &
8405     \bfseries\pagelistname\tablearnewline}}%
8406   \tabletail{}}%
8407   \begin{supertabular}{l111}}%
8408   {\end{supertabular}}%
8409 }
```

`super4colborder` The `super4colborder` style is like the `super4col` but with a border.

```
8410 \newglossarystyle{super4colborder}{%
```

Base it on the `glostylesuper4col` style:

```
8411 \setglossarystyle{super4col}%
```

Put the glossary in a `supertabular` environment with four columns and a horizontal line in the head and tail:

```
8412 \renewenvironment{theglossary}{%
```

```

8413   {\tablehead{\hline}\tabletail{\hline}}%
8414   \begin{supertabular}{|l|l|l|l|}%
8415   {\end{supertabular}}%
8416 }

```

`super4colheaderborder` The `super4colheaderborder` style is like the `super4col` but with a header and border.

```

8417 \newglossarystyle{super4colheaderborder}{%
      Base it on the glostylesuper4col style:
8418   \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:
8419   \renewenvironment{theglossary}%
8420     {\tablehead{\hline\bfseries\entryname&\bfseries\descriptionname&
8421               \bfseries\symbolname &
8422               \bfseries\pagelistname\tabularnewline\hline}}%
8423     \tabletail{\hline}}%
8424     \begin{supertabular}{|l|l|l|l|}%
8425     {\end{supertabular}}%
8426 }

```

`altsuper4col` The `altsuper4col` glossary style is like `super4col` but has provision for multiline descriptions.

```

8427 \newglossarystyle{altsuper4col}{%
      Base it on the glostylesuper4col style:
8428   \setglossarystyle{super4col}%
      Put the glossary in a supertabular environment with four columns and no head
      or tail:
8429   \renewenvironment{theglossary}%
8430     {\tablehead{}\tabletail{}}%
8431     \begin{supertabular}{lp{\glstdescwidth}lp{\glspagelistwidth}}}%
8432     {\end{supertabular}}%
8433 }

```

`altsuper4colheader` The `altsuper4colheader` style is like the `altsuper4col` but with a header row.

```

8434 \newglossarystyle{altsuper4colheader}{%
      Base it on the glostylesuper4colheader style:
8435   \setglossarystyle{super4colheader}%
      Put the glossary in a supertabular environment with four columns, a header and
      no tail:
8436   \renewenvironment{theglossary}%
8437     {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8438               \bfseries\symbolname &
8439               \bfseries\pagelistname\tabularnewline}\tabletail{}}%

```

```

8440     \begin{supertabular}{lp{\glsdescwidth}lp{\glspagelistwidth}}}%
8441     {\end{supertabular}}}%
8442 }

```

`altsuper4colborder` The `altsuper4colborder` style is like the `altsuper4col` but with a border.

```

8443 \newglossarystyle{altsuper4colborder}{%
      Base it on the glostylesuper4colborder style:
8444   \setglossarystyle{super4colborder}%
      Put the glossary in a supertabular environment with four columns and a horizontal line in the head and tail:
8445   \renewenvironment{theglossary}%
8446     {\tablehead{\hline}\tabletail{\hline}}%
8447     \begin{supertabular}%
8448       {|lp{\glsdescwidth}|lp{\glspagelistwidth}|}%
8449     {\end{supertabular}}}%
8450 }

```

`altsuper4colheaderborder` The `altsuper4colheaderborder` style is like the `altsuper4col` but with a header and border.

```

8451 \newglossarystyle{altsuper4colheaderborder}{%
      Base it on the glostylesuper4colheaderborder style:
8452   \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:
8453   \renewenvironment{theglossary}%
8454     {\tablehead{\hline
8455       \bfseries\entryname &
8456       \bfseries\descriptionname &
8457       \bfseries\symbolname &
8458       \bfseries\pagelistname\tabularnewline\hline}%
8459     \tabletail{\hline}}%
8460     \begin{supertabular}%
8461       {|lp{\glsdescwidth}|lp{\glspagelistwidth}|}%
8462     {\end{supertabular}}}%
8463 }

```

3.8 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.

```

8464 \ProvidesPackage{glossary-superragged}[2015/09/09 v4.18 (NLCT)]

```

Requires the package:

```

8465 \RequirePackage{array}

```

Requires the package:

```
8466 \RequirePackage{supertabular}
```

`\glsdescwidth` This is a length that governs the width of the description column. This may already have been defined.

```
8467 \@ifundefined{glsdescwidth}{%
8468   \newlength{glsdescwidth}
8469   \setlength{glsdescwidth}{0.6\hsize}
8470 }{}
```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```
8471 \@ifundefined{glspagelistwidth}{%
8472   \newlength{glspagelistwidth}
8473   \setlength{glspagelistwidth}{0.1\hsize}
8474 }{}
```

`superragged` The superragged glossary style uses the supertabular environment.

```
8475 \newglossarystyle{superragged}{%
```

Put the glossary in a supertabular environment with two columns and no head or tail:

```
8476   \renewenvironment{theglossary}%
8477     {\tablehead{} \tabletail{}}%
8478     \begin{supertabular}{1>{\raggedright}p{glsdescwidth}}%
8479     {\end{supertabular}}%
```

Do nothing at the start of the table:

```
8480   \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8481   \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```
8482   \renewcommand{\glossentry}[2]{%
8483     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8484     \glossentrydesc{##1}\glspostdescription\space ##2%
8485     \tabularnewline
8486   }%
```

Sub entries put in a row (no name, description and page list in second column):

```
8487   \renewcommand{\subglossentry}[3]{%
8488     &
8489     \glssubentryitem{##2}%
8490     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
8491     ##3%
8492     \tabularnewline
8493   }%
```

Blank row between groups:

```
8494 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else & \tabularnewline\fi}%
8495 }
```

superraggedborder The superraggedborder style is like the above, but with horizontal and vertical lines:

```
8496 \newglossarystyle{superraggedborder}{%
```

Base it on the glostylesuperragged style:

```
8497 \setglossarystyle{superragged}%
```

Put the glossary in a supertabular environment with two columns and a horizontal line in the head and tail:

```
8498 \renewenvironment{theglossary}%
8499 {\tablehead{\hline}\tabletail{\hline}%
8500 \begin{supertabular}{|l|>{\raggedright}p{\glsdescwidth}|}}%
8501 {\end{supertabular}}%
8502 }
```

superraggedheader The superraggedheader style is like the super style, but with a header:

```
8503 \newglossarystyle{superraggedheader}{%
```

Base it on the glostylesuperragged style:

```
8504 \setglossarystyle{superragged}%
```

Put the glossary in a supertabular environment with two columns, a header and no tail:

```
8505 \renewenvironment{theglossary}%
8506 {\tablehead{\bfseries \entryname & \bfseries \descriptionname
8507 \tabularnewline}%
8508 \tabletail{}}%
8509 \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}}}%
8510 {\end{supertabular}}%
8511 }
```

superraggedheaderborder The superraggedheaderborder style is like the superragged style but with a header and border:

```
8512 \newglossarystyle{superraggedheaderborder}{%
```

Base it on the glostylesuper style:

```
8513 \setglossarystyle{superragged}%
```

Put the glossary in a supertabular environment with two columns, a header and horizontal lines above and below the table:

```
8514 \renewenvironment{theglossary}%
8515 {\tablehead{\hline\bfseries \entryname &
8516 \bfseries \descriptionname\hline}%
8517 \tabletail{\hline}
8518 \begin{supertabular}{|l|>{\raggedright}p{\glsdescwidth}|}}%
8519 {\end{supertabular}}%
8520 }
```


superragged3col The superragged3col style is like the superragged style, but with 3 columns:

```
8521 \newglossarystyle{superragged3col}{%
```

Put the glossary in a supertabular environment with three columns and no head or tail:

```
8522 \renewenvironment{theglossary}{%
8523   {\tablehead{}}\tabletail{}}%
8524   \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}%
8525     >{\raggedright}p{\glspagelistwidth}}}%
8526   {\end{supertabular}}}%
```

Do nothing at the start of the table:

```
8527 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8528 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
8529 \renewcommand{\glossentry}[2]{%
8530   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8531   \glossentrydesc{##1} &
8532   ##2\tabularnewline
8533 }%
```

Sub entries on a row (no name, description in second column, page list in last column):

```
8534 \renewcommand{\subglossentry}[3]{%
8535   &
8536   \glsesubentryitem{##2}%
8537   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8538   ##3\tabularnewline
8539 }%
```

Blank row between groups:

```
8540 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else & \tabularnewline\fi}%
8541 }
```

superragged3colborder The superragged3colborder style is like the superragged3col style, but with a border:

```
8542 \newglossarystyle{superragged3colborder}{%
```

Base it on the glostylesuperragged3col style:

```
8543 \setglossarystyle{superragged3col}%
```

Put the glossary in a supertabular environment with three columns and a horizontal line in the head and tail:

```
8544 \renewenvironment{theglossary}{%
8545   {\tablehead{\hline}\tabletail{\hline}}%
8546   \begin{supertabular}{l|l|>{\raggedright}p{\glsdescwidth}|%
8547     >{\raggedright}p{\glspagelistwidth}|}}%
```

```
8548   {\end{supertabular}}}%
8549 }
```

`superragged3colheader` The `superragged3colheader` style is like the `superragged3col` style but with a header row:

```
8550 \newglossarystyle{superragged3colheader}{%
```

Base it on the `glostylesuperragged3col` style:

```
8551   \setglossarystyle{superragged3col}%
```

Put the glossary in a `supertabular` environment with three columns, a header and no tail:

```
8552   \renewenvironment{theglossary}%
8553     {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8554       \bfseries\pagelistname\tabularnewline}\tabletail{}}%
8555     \begin{supertabular}{1>{\raggedright}p{\glsdescwidth}%
8556       >{\raggedright}p{\glspagelistwidth}}}%
8557     {\end{supertabular}}}%
8558 }
```

`superragged3colheaderborder` The `superragged3colheaderborder` style is like the `superragged3col` style but with a header and border:

```
8559 \newglossarystyle{superragged3colheaderborder}{%
```

Base it on the `glostylesuperragged3colborder` style:

```
8560   \setglossarystyle{superragged3colborder}%
```

Put the glossary in a `supertabular` environment with three columns, a header with horizontal lines and a horizontal line in the tail:

```
8561   \renewenvironment{theglossary}%
8562     {\tablehead{\hline
8563       \bfseries\entryname&\bfseries\descriptionname&
8564       \bfseries\pagelistname\tabularnewline\hline}%
8565     \tabletail{\hline}%
8566     \begin{supertabular}{1|>{\raggedright}p{\glsdescwidth}|%
8567       >{\raggedright}p{\glspagelistwidth}|}%
8568     {\end{supertabular}}}%
8569 }
```

`altsuperragged4col` The `altsuperragged4col` glossary style is like `altsuper4col` style in the package but uses ragged right formatting in the description and page list columns.

```
8570 \newglossarystyle{altsuperragged4col}{%
```

Put the glossary in a `supertabular` environment with four columns and no head or tail:

```
8571   \renewenvironment{theglossary}%
8572     {\tablehead{}\tabletail{}}%
8573     \begin{supertabular}{1>{\raggedright}p{\glsdescwidth}1%
8574       >{\raggedright}p{\glspagelistwidth}}}%
8575     {\end{supertabular}}%
```

Do nothing at the start of the table:

```
8576 \renewcommand*\glossaryheader{}%
```

No group headings:

```
8577 \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:

```
8578 \renewcommand{\glossentry}[2]{%
8579   \glstarget{##1}\glstarget{##1}{\glossentryname{##1}} &
8580   \glossentrydesc{##1} &
8581   \glossentrysymbol{##1} & ##2\tabularnewline
8582 }%
```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```
8583 \renewcommand{\subglossentry}[3]{%
8584   &
8585   \glssubentryitem{##2}%
8586   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8587   \glossentrysymbol{##2} & ##3\tabularnewline
8588 }%
```

Blank row between groups:

```
8589 \renewcommand*\glsgroupskip{\ifglsnogroupskip\else & & \tabularnewline\fi}%
8590 }
```

`altsuperragged4colheader` The `altsuperragged4colheader` style is like the `altsuperragged4col` style but with a header row.

```
8591 \newglossarystyle{altsuperragged4colheader}{%
```

Base it on the `glostylealtsuperragged4col` style:

```
8592 \setglossarystyle{altsuperragged4col}%
```

Put the glossary in a supertabular environment with four columns, a header and no tail:

```
8593 \renewenvironment{theglossary}%
8594   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8595     \bfseries\symbolname &
8596     \bfseries\pagelistname\tabularnewline}\tabletail{}}%
8597   \begin{supertabular}{1>{\raggedright}p{\glsdescwidth}1%
8598     >{\raggedright}p{\glspagelistwidth}}}%
8599   {\end{supertabular}}%
8600 }
```

`altsuperragged4colborder` The `altsuperragged4colborder` style is like the `altsuperragged4col` style but with a border.

```
8601 \newglossarystyle{altsuperragged4colborder}{%
```

Base it on the `glostylealtsuperragged4col` style:

```
8602 \setglossarystyle{altsuperragged4col}%
```

Put the glossary in a supertabular environment with four columns and a horizontal line in the head and tail:

```
8603 \renewenvironment{theglossary}%
8604   {\tablehead{\hline}\tabletail{\hline}%
8605    \begin{supertabular}%
8606     {\l|>{\raggedright}p{\glsdescwidth}|l|%
8607      >{\raggedright}p{\glspagelistwidth}|}}%
8608   {\end{supertabular}}%
8609 }
```

`ged4colheaderborder` The `altsuperragged4colheaderborder` style is like the `altsuperragged4col` style but with a header and border.

```
8610 \newglossarystyle{altsuperragged4colheaderborder}{%
```

Base it on the `glostylealtsuperragged4col` style:

```
8611 \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:

```
8612 \renewenvironment{theglossary}%
8613   {\tablehead{\hline
8614     \bfseries\entryname &
8615     \bfseries\descriptionname &
8616     \bfseries\symbolname &
8617     \bfseries\pagelistname\stabularnewline\hline}%
8618   \tabletail{\hline}%
8619   \begin{supertabular}%
8620     {\l|>{\raggedright}p{\glsdescwidth}|l|%
8621      >{\raggedright}p{\glspagelistwidth}|}}%
8622   {\end{supertabular}}%
8623 }
```

3.9 Tree Styles (`glossary-tree.sty`)

The style file defines glossary styles that have a tree-like structure. These are designed for hierarchical glossaries.

```
8624 \ProvidesPackage{glossary-tree}[2015/09/09 v4.18 (NLCT)]
```

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```
8625 \providecommand{\indexspace}{%
8626   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
8627 }
```

`\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.

```
8628 \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.

```
8629 \newglossarystyle{index}{%
```

Set the paragraph indentation and skip and define `\item` to be the same as that used by `theindex`:

```
8630 \renewenvironment{theglossary}{%
8631   {\setlength{\parindent}{0pt}}%
8632   \setlength{\parskip}{0pt plus 0.3pt}}%
8633   \let\item\@idxitem}%
```

```
8634   {\par}%
```

Do nothing at the start of the environment:

```
8635 \renewcommand*{\glossaryheader}{}%
```

No group headers:

```
8636 \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.

```
8637 \renewcommand*{\glossentry}[2]{%
8638   \item\glstentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
8639   \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
8640   \space \glossentrydesc{##1}\glspostdescription\space ##2%
8641 }%
```

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.

```
8642 \renewcommand{\subglossentry}[3]{%
8643   \ifcase##1\relax
8644     % level 0
8645     \item
8646   \or
8647     % level 1
8648     \subitem
8649     \glssubentryitem{##2}%
8650   \else
8651     % all other levels
8652     \subsubitem
8653   \fi
8654   \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
8655   \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
8656   \space\glossentrydesc{##2}\glspostdescription\space ##3%
8657 }%
```

Vertical gap between groups is the same as that used by indices:

```
8658 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}}
```

`indexgroup` The `indexgroup` style is like the `index` style but has headings.

```
8659 \newglossarystyle{indexgroup}{%
```

Base it on the `glostyleindex` style:

```
8660 \setglossarystyle{index}%
```

Add a heading for each group. This puts the group's title in bold followed by a vertical gap.

```
8661 \renewcommand*{\glsgroupheading}[1]{%
```

```
8662 \item\glstreenamefmt{\glsgetgrouptitle{##1}}\indexspace}%
```

```
8663 }
```

`indexhypergroup` The `indexhypergroup` style is like the `indexgroup` style but has hyper navigation.

```
8664 \newglossarystyle{indexhypergroup}{%
```

Base it on the `glostyleindex` style:

```
8665 \setglossarystyle{index}%
```

Put navigation links to the groups at the start of the glossary:

```
8666 \renewcommand*{\glossaryheader}{%
```

```
8667 \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.

```
8668 \renewcommand*{\glsgroupheading}[1]{%
```

```
8669 \item\glstreenamefmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
```

```
8670 \indexspace}%
```

```
8671 }
```

`tree` The `tree` glossary style is similar in style to the `index` style, but can have arbitrary levels.

```
8672 \newglossarystyle{tree}{%
```

Set the paragraph indentation and skip:

```
8673 \renewenvironment{theglossary}%
```

```
8674 {\setlength{\parindent}{0pt}%
```

```
8675 \setlength{\parskip}{0pt plus 0.3pt}}%
```

```
8676 {}%
```

Do nothing at the start of the `theglossary` environment:

```
8677 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8678 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries: name in bold, followed by symbol in brackets (if it exists), the description and the page list:

```
8679 \renewcommand{\glossentry}[2]{%
```

```
8680 \hangindent0pt\relax
```

```

8681 \parindent0pt\relax
8682 \glstentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
8683 \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
8684 \space\glossentrydesc{##1}\glspostdescription\space##2\par
8685 }%

```

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.

```

8686 \renewcommand{\subglossentry}[3]{%
8687 \hangindent##1\glstreeindent\relax
8688 \parindent##1\glstreeindent\relax
8689 \ifnum##1=1\relax
8690 \glssubentryitem{##2}%
8691 \fi
8692 \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
8693 \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
8694 \space\glossentrydesc{##2}\glspostdescription\space ##3\par
8695 }%

```

Vertical gap between groups is the same as that used by indices:

```

8696 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}

```

treegroup Like the tree style but the glossary groups have headings.

```

8697 \newglossarystyle{treegroup}{%

```

Base it on the `glostyletree` style:

```

8698 \setglossarystyle{tree}%

```

Each group has a heading (in bold) followed by a vertical gap):

```

8699 \renewcommand{\glsgroupeheading}[1]{\par
8700 \noindent\glstreenamefmt{\glsggetgrouptitle{##1}}\par\indexspace}%
8701 }

```

treehypergroup The `treehypergroup` style is like the `treegroup` style, but has a set of links to the groups at the start of the glossary.

```

8702 \newglossarystyle{treehypergroup}{%

```

Base it on the `glostyletree` style:

```

8703 \setglossarystyle{tree}%

```

Put navigation links to the groups at the start of the `theglossary` environment:

```

8704 \renewcommand*{\glossaryheader}{%
8705 \par\noindent\glstreenamefmt{\glsnavigation}\par\indexspace}%

```

Each group has a heading (in bold with a target) followed by a vertical gap):

```

8706 \renewcommand*{\glsgroupeheading}[1]{%
8707 \par\noindent
8708 \glstreenamefmt{\glsnavhypertarget{##1}{\glsggetgrouptitle{##1}}}\par
8709 \indexspace}%
8710 }

```

`\glstreeindent` Length governing left indent for each level of the tree style.

```
8711 \newlength\glstreeindent
8712 \setlength{\glstreeindent}{10pt}
```

`treenoname` The `treenoname` glossary style is like the tree style, but doesn't print the name or symbol for sub-levels.

```
8713 \newglossarystyle{treenoname}{%
```

Set the paragraph indentation and skip:

```
8714 \renewenvironment{theglossary}%
8715   {\setlength{\parindent}{0pt}%
8716    \setlength{\parskip}{0pt plus 0.3pt}}%
8717   {}%
```

No header:

```
8718 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8719 \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.

```
8720 \renewcommand{\glossentry}[2]{}%
8721   \hangindent0pt\relax
8722   \parindent0pt\relax
8723   \glstryitem{##1}\glstreenamfmt{\glstarget{##1}{\glossentryname{##1}}}%
8724   \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
8725   \space\glossentrydesc{##1}\glspostdescription\space##2\par
8726   }%
```

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.

```
8727 \renewcommand{\subglossentry}[3]{}%
8728   \hangindent##1\glstreeindent\relax
8729   \parindent##1\glstreeindent\relax
8730   \ifnum##1=1\relax
8731     \glssubentryitem{##2}%
8732     \fi
8733     \glstarget{##2}{\strut}%
8734     \glossentrydesc{##2}\glspostdescription\space##3\par
8735   }%
```

Vertical gap between groups is the same as that used by indices:

```
8736 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
8737 }
```

`treenonamegroup` Like the `treenoname` style but the glossary groups have headings.

```
8738 \newglossarystyle{treenonamegroup}{%
```

Base it on the `glostyletreenoname` style:

```
8739 \setglossarystyle{treenoname}%
```


Give each group a heading:

```
8740 \renewcommand{\glsgroupheading}[1]{\par
8741 \noindent\glstreenamefmt{\glsgetgrouptitle{##1}}\par\indexspace}%
8742 }
```

`treenonamehypergroup` The `treenonamehypergroup` style is like the `treenonamegroup` style, but has a set of links to the groups at the start of the glossary.

```
8743 \newglossarystyle{treenonamehypergroup}{%
```

Base it on the `glostyletreenoname` style:

```
8744 \setglossarystyle{treenoname}{%
```

Put navigation links to the groups at the start of the `theglossary` environment:

```
8745 \renewcommand*\glossaryheader}{%
8746 \par\noindent\glstreenamefmt{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
8747 \renewcommand*\glsgroupheading}[1]{%
8748 \par\noindent
8749 \glstreenamefmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8750 \indexspace}%
8751 }
```

`\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.

```
8752 \newcommand*\glssetwidest}[2][0]{%
8753 \expandafter\def\csname @glswidestname\romannumeral#1\endcsname{%
8754 #2}%
8755 }
```

`\@glswidestname` Initialise `\@glswidestname`.

```
8756 \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`.

```
8757 \newglossarystyle{alttree}{%
```

Redefine the `theglossary` environment.

```
8758 \renewenvironment{theglossary}{%
8759 {\def\@gls@prevlevel{-1}%
8760 \mbox{}\par}%
8761 {\par}%
```

Set the header and group headers to nothing.

```
8762 \renewcommand*\glossaryheader}{}%
8763 \renewcommand*\glsgroupheading}[1]{}
```

Redefine the way that the level 0 entries are displayed.

```
8764 \renewcommand{\glossentry}[2]{%
8765 \ifnum\@gls@prevlevel=0\relax
8766 \else
```

Find out how big the indentation should be by measuring the widest entry.

```
8767     \settowidth{\glstreeindent}{\glstreenamefmt{\@glswidestname\space}}%
8768     \fi
```

Set the hangindent and paragraph indent.

```
8769     \hangindent\glstreeindent
8770     \parindent\glstreeindent
```

Put the name to the left of the paragraph block.

```
8771     \makebox[0pt][r]{\makebox[\glstreeindent][l]{%
8772         \glstentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}}}%
```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
8773     \ifglshassymbol{##1}{(\glossentrysymbol{##1})\space}{}%
```

Do the description followed by the description terminator and location list.

```
8774     \glossentrydesc{##1}\glspostdescription \space ##2\par
```

Set the previous level to 0.

```
8775     \def\@gls@prevlevel{0}%
8776     }%
```

Redefine the way sub-entries are displayed.

```
8777     \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.

```
8778     \ifnum##1=1\relax
8779         \glssubentryitem{##2}%
8780     \fi
```

If the level hasn't changed, keep the same settings, otherwise adjust `\glstreeindent` accordingly.

```
8781     \ifnum\@gls@prevlevel=##1\relax
8782     \else
```

Compute the widest entry for this level, or for level 0 if not defined for this level.

Store in `\gls@tmplen`

```
8783     \@ifundefined{@glswidestname\romannumeral##1}{%
8784         \settowidth{\gls@tmplen}{\glstreenamefmt{\@glswidestname\space}}}%
8785     \settowidth{\gls@tmplen}{\glstreenamefmt{%
8786         \csname @glswidestname\romannumeral##1\endcsname\space}}}%
```

Determine if going up or down a level

```
8787     \ifnum\@gls@prevlevel<##1\relax
```

Depth has increased, so add the width of the widest entry to `\glstreeindent`.

```
8788         \setlength\glstreeindent\gls@tmplen
8789         \addtolength\glstreeindent\parindent
8790         \parindent\glstreeindent
8791     \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`.

```
8792     \@ifundefined{@glswidestname\romannumeral\@gls@prevlevel}}{%
8793     \settowidth{\glstreeindent}{\glstreenamefmt{%
8794     \@glswidestname\space}}}%
8795     \settowidth{\glstreeindent}{\glstreenamefmt{%
8796     \csname @glswidestname\romannumeral\@gls@prevlevel
8797     \endcsname\space}}}%
```

Subtract this length from the previous level's paragraph indent and set to `\glstreeindent`.

```
8798     \addtolength\parindent{-\glstreeindent}%
8799     \setlength\glstreeindent\parindent
8800     \fi
8801     \fi
```

Set the hanging indentation.

```
8802     \hangindent\glstreeindent
```

Put the name to the left of the paragraph block

```
8803     \makebox[0pt][r]{\makebox[\gls@tmplen][l]{%
8804     \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}}}%
```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
8805     \ifglshassymbol{##2}{(\glossentrysymbol{##2})\space}{}%
```

Do the description followed by the description terminator and location list.

```
8806     \glossentrydesc{##2}\glspostdescription\space ##3\par
```

Set the previous level macro to the current level.

```
8807     \def\@gls@prevlevel{##1}%
8808     }%
```

Vertical gap between groups is the same as that used by indices:

```
8809     \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
8810 }
```

`almtreegroup` Like the `almtree` style but the glossary groups have headings.

```
8811 \newglossarystyle{almtreegroup}{%
```

Base it on the `glostylealmtree` style:

```
8812     \setglossarystyle{almtree}%
```

Give each group a heading.

```
8813     \renewcommand{\glsgroupheading}[1]{\par
8814     \def\@gls@prevlevel{-1}%
8815     \hangindent0pt\relax
8816     \parindent0pt\relax
8817     \glstreenamefmt{\glsgetgrouptitle{##1}}\par\indexspace}%
8818 }
```

`almtreehypergroup` The `almtreehypergroup` style is like the `almtreegroup` style, but has a set of links to the groups at the start of the glossary.

```
8819 \newglossarystyle{almtreehypergroup}{%
```

Base it on the `glostylealmtree` style:

```
8820 \setglossarystyle{almtree}%
```

Put the navigation links in the header

```
8821 \renewcommand*{\glossaryheader}{%
```

```
8822 \par
```

```
8823 \def\@gls@prevlevel{-1}%
```

```
8824 \hangindent0pt\relax
```

```
8825 \parindent0pt\relax
```

```
8826 \glstreenamefmt{\glsnavigation}\par\indexspace}%
```

Put a hypertarget at the start of each group

```
8827 \renewcommand*{\glsgroupheading}[1]{%
```

```
8828 \par
```

```
8829 \def\@gls@prevlevel{-1}%
```

```
8830 \hangindent0pt\relax
```

```
8831 \parindent0pt\relax
```

```
8832 \glstreenamefmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
```

```
8833 \indexspace}}
```

4 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.

```
8834 \NeedsTeXFormat{LaTeX2e}
```

```
8835 \ProvidesPackage{glossaries-compatible-207}[2015/09/09 v4.18 (NLCT)]
```

`\GlsAddXdyAttribute` Adds an attribute in old format.

```
8836 \ifglsxindy
```

```
8837 \renewcommand*\GlsAddXdyAttribute[1]{%
```

```
8838 \edef\@xdyattributes{\@xdyattributes ^^J \string"#1\string"}%
```

```
8839 \expandafter\toks@\expandafter{\@xdylocref}%
```

```
8840 \edef\@xdylocref{\the\toks@ ^^J%
```

```
8841 (markup-locref
```

```
8842 :open \string"\string~n\string\setentrycounter
```

```
8843 {\noexpand\glscounter}}%
```

```
8844 \expandafter\string\csname#1\endcsname
```

```
8845 \expandafter\@gobble\string\{\string" ^^J
```

```
8846 :close \string"\expandafter\@gobble\string\}\string" ^^J
```

```
8847 :attr \string"#1\string"))}}
```

Only has an effect before `\writeist`:

```
8848 \fi
```

`\GlsAddXdyCounters`

```
8849 \renewcommand*\GlsAddXdyCounters[1]{%
8850   \GlossariesWarning{\string\GlsAddXdyCounters\space not available
8851     in compatibility mode.}%
8852 }
```

Add predefined attributes

```
8853 \GlsAddXdyAttribute{glsnumberformat}
8854 \GlsAddXdyAttribute{textrm}
8855 \GlsAddXdyAttribute{textsf}
8856 \GlsAddXdyAttribute{texttt}
8857 \GlsAddXdyAttribute{textbf}
8858 \GlsAddXdyAttribute{textmd}
8859 \GlsAddXdyAttribute{textit}
8860 \GlsAddXdyAttribute{textup}
8861 \GlsAddXdyAttribute{textsl}
8862 \GlsAddXdyAttribute{textsc}
8863 \GlsAddXdyAttribute{emph}
8864 \GlsAddXdyAttribute{glsnumber}
8865 \GlsAddXdyAttribute{hyperrm}
8866 \GlsAddXdyAttribute{hypersf}
8867 \GlsAddXdyAttribute{hypertt}
8868 \GlsAddXdyAttribute{hyperbf}
8869 \GlsAddXdyAttribute{hypermd}
8870 \GlsAddXdyAttribute{hyperit}
8871 \GlsAddXdyAttribute{hyperup}
8872 \GlsAddXdyAttribute{hypersl}
8873 \GlsAddXdyAttribute{hypersc}
8874 \GlsAddXdyAttribute{hyperemph}
```

`\GlsAddXdyLocation` Restore v2.07 definition:

```
8875 \ifglsxindy
8876   \renewcommand*\GlsAddXdyLocation[2]{%
8877     \edef\@xdyuserlocationdefs{%
8878       \@xdyuserlocationdefs ^^J%
8879       (define-location-class \string"#1\string"^^J\space\space
8880         \space(#2))
8881     }%
8882     \edef\@xdyuserlocationnames{%
8883       \@xdyuserlocationnames^^J\space\space\space
8884       \string"#1\string"}%
8885   }
8886 \fi
```

`\@do@wrglossary`

```
8887 \renewcommand{\@do@wrglossary}[1]{%
    Determine whether to use xindy or makeindex syntax
8888 \ifglsxindy
```

Need to determine if the formatting information starts with a (or) indicating a range.

```

8889 \expandafter\@glo@check@mkidxrangear\@glsnumberformat\@nil
8890 \def\@glo@range{}%
8891 \expandafter\if\@glo@prefix(\relax
8892 \def\@glo@range{:open-range}%
8893 \else
8894 \expandafter\if\@glo@prefix)\relax
8895 \def\@glo@range{:close-range}%
8896 \fi
8897 \fi

```

Get the location and escape any special characters

```

8898 \protected@edef\@glslocref{\theglsentrycounter}%
8899 \@gls@checkmkidxchars\@glslocref

```

Write to the glossary file using xindy syntax.

```

8900 \glossary[\csname glo@#1@type\endcsname]{%
8901 (indexentry :tkey (\csname glo@#1@index\endcsname)
8902 :locref \string"\@glslocref\string" %
8903 :attr \string"\@glo@suffix\string" \@glo@range
8904 )
8905 }%
8906 \else

```

Convert the format information into the format required for makeindex

```

8907 \@set@glo@numformat\@glo@numfmt\@gls@counter\@glsnumberformat

```

Write to the glossary file using makeindex syntax.

```

8908 \glossary[\csname glo@#1@type\endcsname]{%
8909 \string\glossaryentry{\csname glo@#1@index\endcsname
8910 \@gls@encapchar\@glo@numfmt}\theglsentrycounter}%
8911 \fi
8912 }

```

\@set@glo@numformat Only had 3 arguments in v2.07

```

8913 \def\@set@glo@numformat#1#2#3{%
8914 \expandafter\@glo@check@mkidxrangear#3\@nil
8915 \protected@edef#1{%
8916 \@glo@prefix setentrycounter[] {#2}%
8917 \expandafter\string\csname\@glo@suffix\endcsname
8918 }%
8919 \@gls@checkmkidxchars#1%
8920 }

```

\writeist Redefine \writeist back to the way it was in v2.07, but change \istfile to \glswrite.

```

8921 \ifglsxindy
8922 \def\writeist{%
8923 \openout\glswrite=\istfilename

```

```

8924 \write\glswrite{;; xindy style file created by the glossaries
8925   package in compatible-2.07 mode}%
8926 \write\glswrite{;; for document '\jobname' on
8927   \the\year-\the\month-\the\day}%
8928 \write\glswrite{^^J; required styles^^J}
8929 \@for\@xdystyle:=\@xdyrequiredstyles\do{%
8930   \ifx\@xdystyle\@empty
8931     \else
8932       \protected@write\glswrite{{(require
8933         \string"\@xdystyle.xdy\string")}}%
8934     \fi
8935   }%
8936 \write\glswrite{^^J%
8937   ; list of allowed attributes (number formats)^^J}%
8938 \write\glswrite{(define-attributes ((\@xdyattributes)))}%
8939 \write\glswrite{^^J; user defined alphabets^^J}%
8940 \write\glswrite{\@xdyuseralphabets}%
8941 \write\glswrite{^^J; location class definitions^^J}%
8942 \protected@edef\@gls@roman{\@roman{0\string"
8943   \string"roman-numbers-lowercase\string" :sep \string"}}%
8944 \@onelevel@sanitize\@gls@roman
8945 \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
8946   :sep \string"}%
8947 \@onelevel@sanitize\@tmp
8948 \ifx\@tmp\@gls@roman
8949   \write\glswrite{(define-location-class
8950     \string"roman-page-numbers\string"^^J\space\space\space
8951     (\string"roman-numbers-lowercase\string")
8952     :min-range-length \@glsminrange)}%
8953   \else
8954     \write\glswrite{(define-location-class
8955       \string"roman-page-numbers\string"^^J\space\space\space
8956       (:sep "\@gls@roman")
8957       :min-range-length \@glsminrange)}%
8958   \fi
8959 \write\glswrite{(define-location-class
8960   \string"Roman-page-numbers\string"^^J\space\space\space
8961   (\string"roman-numbers-uppercase\string")
8962   :min-range-length \@glsminrange)}%
8963 \write\glswrite{(define-location-class
8964   \string"arabic-page-numbers\string"^^J\space\space\space
8965   (\string"arabic-numbers\string")
8966   :min-range-length \@glsminrange)}%
8967 \write\glswrite{(define-location-class
8968   \string"alpha-page-numbers\string"^^J\space\space\space
8969   (\string"alpha\string")
8970   :min-range-length \@glsminrange)}%
8971 \write\glswrite{(define-location-class
8972   \string"Alpha-page-numbers\string"^^J\space\space\space

```

```

8973     (\string"ALPHA\string")
8974         :min-range-length \@glsminrange}}%
8975 \write\glswrite{(define-location-class
8976     \string"Appendix-page-numbers\string"^^J\space\space\space
8977     (\string"ALPHA\string"
8978         :sep \string"\@glsAlphacompositor\string"
8979         \string"arabic-numbers\string")
8980         :min-range-length \@glsminrange}}%
8981 \write\glswrite{(define-location-class
8982     \string"arabic-section-numbers\string"^^J\space\space\space
8983     (\string"arabic-numbers\string"
8984         :sep \string"\glscompositor\string"
8985         \string"arabic-numbers\string")
8986         :min-range-length \@glsminrange}}%
8987 \write\glswrite{^^J; user defined location classes}%
8988 \write\glswrite{@xdyuserlocationdefs}%
8989 \write\glswrite{^^J; define cross-reference class^^J}%
8990 \write\glswrite{(define-crossref-class \string"see\string"
8991     :unverified )}%
8992 \write\glswrite{(markup-crossref-list
8993     :class \string"see\string"^^J\space\space\space
8994     :open \string"\string\glsseeformat\string"
8995     :close \string"{}\string")}%
8996 \write\glswrite{^^J; define the order of the location classes}%
8997 \write\glswrite{(define-location-class-order
8998     (\@xdylocationclassorder))}%
8999 \write\glswrite{^^J; define the glossary markup^^J}%
9000 \write\glswrite{(markup-index^^J\space\space\space
9001     :open \string"\string
9002     \glossarysection[\string\glossarytoctitle]{\string
9003     \glossarytitle}\string\glossarypreamble\string~n\string\begin
9004     {theglossary}\string\glossaryheader\string~n\string" ^^J\space
9005     \space\space:close \string"\expandafter\@gobble
9006     \string%\string~n\string
9007     \end{theglossary}\string\glossarypostamble
9008     \string~n\string" ^^J\space\space\space
9009     :tree}}%
9010 \write\glswrite{(markup-letter-group-list
9011     :sep \string"\string\glsgroupskip\string~n\string")}%
9012 \write\glswrite{(markup-indexentry
9013     :open \string"\string\relax \string\glsresetentrylist
9014     \string~n\string")}%
9015 \write\glswrite{(markup-locclass-list :open
9016     \string"\glsopenbrace\string\glossaryentrynumbers
9017     \glsopenbrace\string\relax\space \string"^^J\space\space\space
9018     :sep \string", \string"
9019     :close \string"\glsclosebrace\glsclosebrace\string")}%
9020 \write\glswrite{(markup-locref-list
9021     :sep \string"\string\delimN\space\string")}%

```



```

9022 \write\glswrite{(markup-range
9023 :sep \string"\string\delimR\space\string")}%
9024 \@onelevel@sanitize\gls@suffixF
9025 \@onelevel@sanitize\gls@suffixFF
9026 \ifx\gls@suffixF\@empty
9027 \else
9028 \write\glswrite{(markup-range
9029 :close "\gls@suffixF" :length 1 :ignore-end)}%
9030 \fi
9031 \ifx\gls@suffixFF\@empty
9032 \else
9033 \write\glswrite{(markup-range
9034 :close "\gls@suffixFF" :length 2 :ignore-end)}%
9035 \fi
9036 \write\glswrite{^^J; define format to use for locations^^J}%
9037 \write\glswrite{\@xdylocref}%
9038 \write\glswrite{^^J; define letter group list format^^J}%
9039 \write\glswrite{(markup-letter-group-list
9040 :sep \string"\string\glsgroupskip\string~n\string")}%
9041 \write\glswrite{^^J; letter group headings^^J}%
9042 \write\glswrite{(markup-letter-group
9043 :open-head \string"\string\glsgroupheading
9044 \glsopenbrace\string"^^J\space\space\space
9045 :close-head \string"\glsclosebrace\string")}%
9046 \write\glswrite{^^J; additional letter groups^^J}%
9047 \write\glswrite{\@xdylettergroups}%
9048 \write\glswrite{^^J; additional sort rules^^J}%
9049 \write\glswrite{\@xdysortrules}%
9050 \noist}
9051 \else
9052 \edef\@gls@actualchar{\string?}
9053 \edef\@gls@encapchar{\string|}
9054 \edef\@gls@levelchar{\string!}
9055 \edef\@gls@quotechar{\string"}
9056 \def\writeist{\relax
9057 \openout\glswrite=\istfilename
9058 \write\glswrite{\expandafter\@gobble\string\% makeindex style file
9059 created by the glossaries package}
9060 \write\glswrite{\expandafter\@gobble\string\% for document
9061 '\jobname' on \the\year-\the\month-\the\day}
9062 \write\glswrite{actual '\@gls@actualchar'}
9063 \write\glswrite{encap '\@gls@encapchar'}
9064 \write\glswrite{level '\@gls@levelchar'}
9065 \write\glswrite{quote '\@gls@quotechar'}
9066 \write\glswrite{keyword \string"\string\glossaryentry\string"}
9067 \write\glswrite{preamble \string"\string\glossarysection[\string
9068 \glossarytoctitle]{\string\glossarytitle}\string
9069 \glossarypreamble\string\n\string\begin{theglossary}\string
9070 \glossaryheader\string\n\string"}

```

```

9071 \write\glswrite{postamble \string"\string%\string\n\string
9072   \end{theglossary}\string\glossarypostamble\string\n
9073   \string"}
9074 \write\glswrite{group_skip \string"\string\glsgroupskip\string\n
9075   \string"}
9076 \write\glswrite{item_0 \string"\string%\string\n\string"}
9077 \write\glswrite{item_1 \string"\string%\string\n\string"}
9078 \write\glswrite{item_2 \string"\string%\string\n\string"}
9079 \write\glswrite{item_01 \string"\string%\string\n\string"}
9080 \write\glswrite{item_x1
9081   \string"\string\relax \string\glsresetentrylist\string\n
9082   \string"}
9083 \write\glswrite{item_12 \string"\string%\string\n\string"}
9084 \write\glswrite{item_x2
9085   \string"\string\relax \string\glsresetentrylist\string\n
9086   \string"}
9087 \write\glswrite{delim_0 \string"\string\{\string
9088   \glossaryentrynumbers\string\{\string\relax \string"}
9089 \write\glswrite{delim_1 \string"\string\{\string
9090   \glossaryentrynumbers\string\{\string\relax \string"}
9091 \write\glswrite{delim_2 \string"\string\{\string
9092   \glossaryentrynumbers\string\{\string\relax \string"}
9093 \write\glswrite{delim_t \string"\string\}\string\}\string"}
9094 \write\glswrite{delim_n \string"\string\delimN \string"}
9095 \write\glswrite{delim_r \string"\string\delimR \string"}
9096 \write\glswrite{headings_flag 1}
9097 \write\glswrite{heading_prefix
9098   \string"\string\glsgroupheading\string\{\string"}
9099 \write\glswrite{heading_suffix
9100   \string"\string\}\string\relax
9101   \string\glsresetentrylist \string"}
9102 \write\glswrite{symhead_positive \string"glssymbols\string"}
9103 \write\glswrite{numhead_positive \string"glnumbers\string"}
9104 \write\glswrite{page_compositor \string"glscpositor\string"}
9105 \@gls@escbsdq\gls@suffixF
9106 \@gls@escbsdq\gls@suffixFF
9107 \ifx\gls@suffixF\@empty
9108 \else
9109   \write\glswrite{suffix_2p \string"\gls@suffixF\string"}
9110 \fi
9111 \ifx\gls@suffixFF\@empty
9112 \else
9113   \write\glswrite{suffix_3p \string"\gls@suffixFF\string"}
9114 \fi
9115 \noist
9116 }
9117 \fi

```

\noist

```
9118 \renewcommand*\noist}{\let\writeist\relax}
```

Compatibility macros.

```
9119 \NeedsTeXFormat{LaTeX2e}
```

```
9120 \ProvidesPackage{glossaries-compatible-307}[2015/09/09 v4.18 (NLCT)]
```

Compatibility macros for predefined glossary styles:

`compatglossarystyle` Defines a compatibility glossary style.

```
9121 \newcommand{\compatglossarystyle}[2]{%
9122   \ifcsundef{@glscompstyle@#1}%
9123   {%
9124     \csdef{@glscompstyle@#1}{#2}%
9125   }%
9126   {%
9127     \PackageError{glossaries}{Glossary compatibility style ‘#1’ is already defined}{}%
9128   }%
9129 }
```

Backward compatible inline style.

```
9130 \compatglossarystyle{inline}{%
9131   \renewcommand{\glossaryentryfield}[5]{%
9132     \glsinlinedopostchild
9133     \gls@inlinesep
9134     \def\glo@desc{##3}%
9135     \def\@no@post@desc{\nopostdesc}%
9136     \glsentryitem{##1}\glsinlinenameformat{##1}{##2}%
9137     \ifx\glo@desc\@no@post@desc
9138       \glsinlineemptydescformat{##4}{##5}%
9139     \else
9140       \ifstrempy{##3}%
9141         {\glsinlineemptydescformat{##4}{##5}}%
9142         {\glsinlinedescformat{##3}{##4}{##5}}%
9143       \fi
9144       \ifglshaschildren{##1}%
9145       {%
9146         \glsresetsubentrycounter
9147         \glsinlineparentchildseparator
9148         \def\gls@inlinesubsep{}%
9149         \def\gls@inlinepostchild{\glsinlinepostchild}%
9150       }%
9151       {}%
9152     \def\gls@inlinesep{\glsinlineseparator}%
9153   }%
```

Sub-entries display description:

```
9154 \renewcommand{\glossarysubentryfield}[6]{%
9155   \gls@inlinesubsep%
9156   \glsinlinesubnameformat{##2}{##3}%
9157   \glssubentryitem{##2}\glsinlinesubdescformat{##4}{##5}{##6}%
```

```

9158 \def\gls@inlinesubsep{\glsinlinesubseparator}%
9159 }%
9160 }

```

Backward compatible list style.

```

9161 \compatglossarystyle{list}{%
9162 \renewcommand*\glossaryentryfield}[5]{%
9163 \item[\glsentryitem{##1}\glstarget{##1}{##2}]
9164 ##3\glspostdescription\space ##5}%

```

Sub-entries continue on the same line:

```

9165 \renewcommand*\glossarysubentryfield}[6]{%
9166 \glssubentryitem{##2}%
9167 \glstarget{##2}{\strut}##4\glspostdescription\space ##6.}%
9168 }

```

Backward compatible listgroup style.

```

9169 \compatglossarystyle{listgroup}{%
9170 \csuse{@glscompstyle@list}%
9171 }%

```

Backward compatible listhypergroup style.

```

9172 \compatglossarystyle{listhypergroup}{%
9173 \csuse{@glscompstyle@list}%
9174 }%

```

Backward compatible altlist style.

```

9175 \compatglossarystyle{altlist}{%
9176 \renewcommand*\glossaryentryfield}[5]{%
9177 \item[\glsentryitem{##1}\glstarget{##1}{##2}]%
9178 \mbox{\par\nobreak\@afterheading
9179 ##3\glspostdescription\space ##5}%
9180 \renewcommand*\glossarysubentryfield}[6]{%
9181 \par
9182 \glssubentryitem{##2}%
9183 \glstarget{##2}{\strut}##4\glspostdescription\space ##6}%
9184 }%

```

Backward compatible altlistgroup style.

```

9185 \compatglossarystyle{altlistgroup}{%
9186 \csuse{@glscompstyle@altlist}%
9187 }%

```

Backward compatible altlisthypergroup style.

```

9188 \compatglossarystyle{altlisthypergroup}{%
9189 \csuse{@glscompstyle@altlist}%
9190 }%

```

Backward compatible listdotted style.

```

9191 \compatglossarystyle{listdotted}{%
9192 \renewcommand*\glossaryentryfield}[5]{%
9193 \item[]\makebox[\glslistdottedwidth][l]{%

```

```

9194     \glstentryitem{##1}\glstarget{##1}{##2}%
9195     \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##3}%
9196 \renewcommand*\glossarysubentryfield}[6]{%
9197   \item[]\makebox[\glslistdottedwidth][l]{%
9198     \glssubentryitem{##2}%
9199     \glstarget{##2}{##3}%
9200     \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##4}%
9201 }%

```

Backward compatible sublistdotted style.

```

9202 \compatglossarystyle{sublistdotted}{%
9203   \csuse{@glscmpstyle@listdotted}%
9204   \renewcommand*\glossaryentryfield}[5]{%
9205     \item[\glstentryitem{##1}\glstarget{##1}{##2}]}%
9206 }%

```

Backward compatible long style.

```

9207 \compatglossarystyle{long}{%
9208   \renewcommand*\glossaryentryfield}[5]{%
9209     \glstentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
9210   \renewcommand*\glossarysubentryfield}[6]{%
9211     &
9212     \glssubentryitem{##2}%
9213     \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
9214 }%

```

Backward compatible longborder style.

```

9215 \compatglossarystyle{longborder}{%
9216   \csuse{@glscmpstyle@long}%
9217 }%

```

Backward compatible longheader style.

```

9218 \compatglossarystyle{longheader}{%
9219   \csuse{@glscmpstyle@long}%
9220 }%

```

Backward compatible longheaderborder style.

```

9221 \compatglossarystyle{longheaderborder}{%
9222   \csuse{@glscmpstyle@long}%
9223 }%

```

Backward compatible long3col style.

```

9224 \compatglossarystyle{long3col}{%
9225   \renewcommand*\glossaryentryfield}[5]{%
9226     \glstentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\\}%
9227   \renewcommand*\glossarysubentryfield}[6]{%
9228     &
9229     \glssubentryitem{##2}%
9230     \glstarget{##2}{\strut}##4 & ##6\\}%
9231 }%

```

Backward compatible long3colborder style.

9232 \compatglossarystyle{long3colborder}{%
9233 \csuse{@glscompstyle@long3col}%
9234 }%

Backward compatible long3colheader style.

9235 \compatglossarystyle{long3colheader}{%
9236 \csuse{@glscompstyle@long3col}%
9237 }%

Backward compatible long3colheaderborder style.

9238 \compatglossarystyle{long3colheaderborder}{%
9239 \csuse{@glscompstyle@long3col}%
9240 }%

Backward compatible long4col style.

9241 \compatglossarystyle{long4col}{%
9242 \renewcommand*{\glossaryentryfield}[5]{%
9243 \glsentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%
9244 \renewcommand*{\glossarysubentryfield}[6]{%
9245 &
9246 \glssubentryitem{##2}%
9247 \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%
9248 }%

Backward compatible long4colheader style.

9249 \compatglossarystyle{long4colheader}{%
9250 \csuse{@glscompstyle@long4col}%
9251 }%

Backward compatible long4colborder style.

9252 \compatglossarystyle{long4colborder}{%
9253 \csuse{@glscompstyle@long4col}%
9254 }%

Backward compatible long4colheaderborder style.

9255 \compatglossarystyle{long4colheaderborder}{%
9256 \csuse{@glscompstyle@long4col}%
9257 }%

Backward compatible altlong4col style.

9258 \compatglossarystyle{altlong4col}{%
9259 \csuse{@glscompstyle@long4col}%
9260 }%

Backward compatible altlong4colheader style.

9261 \compatglossarystyle{altlong4colheader}{%
9262 \csuse{@glscompstyle@long4col}%
9263 }%

Backward compatible altlong4colborder style.

9264 \compatglossarystyle{altlong4colborder}{%
9265 \csuse{@glscompstyle@long4col}%
9266 }%

Backward compatible altlong4colheaderborder style.

```
9267 \compatglossarystyle{altlong4colheaderborder}{%
9268 \csuse{@glscompstyle@long4col}%
9269 }%
```

Backward compatible long style.

```
9270 \compatglossarystyle{longragged}{%
9271 \renewcommand*{\glossaryentryfield}[5]{%
9272 \glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
9273 \tabularnewline}%
9274 \renewcommand*{\glossarysubentryfield}[6]{%
9275 &
9276 \glssubentryitem{##2}%
9277 \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
9278 \tabularnewline}%
9279 }%
```

Backward compatible longraggedborder style.

```
9280 \compatglossarystyle{longraggedborder}{%
9281 \csuse{@glscompstyle@longragged}%
9282 }%
```

Backward compatible longraggedheader style.

```
9283 \compatglossarystyle{longraggedheader}{%
9284 \csuse{@glscompstyle@longragged}%
9285 }%
```

Backward compatible longraggedheaderborder style.

```
9286 \compatglossarystyle{longraggedheaderborder}{%
9287 \csuse{@glscompstyle@longragged}%
9288 }%
```

Backward compatible longragged3col style.

```
9289 \compatglossarystyle{longragged3col}{%
9290 \renewcommand*{\glossaryentryfield}[5]{%
9291 \glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%
9292 \renewcommand*{\glossarysubentryfield}[6]{%
9293 &
9294 \glssubentryitem{##2}%
9295 \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%
9296 }%
```

Backward compatible longragged3colborder style.

```
9297 \compatglossarystyle{longragged3colborder}{%
9298 \csuse{@glscompstyle@longragged3col}%
9299 }%
```

Backward compatible longragged3colheader style.

```
9300 \compatglossarystyle{longragged3colheader}{%
9301 \csuse{@glscompstyle@longragged3col}%
9302 }%
```

Backward compatible longragged3colheaderborder style.

```
9303 \compatglossarystyle{longragged3colheaderborder}{%
9304 \csuse{@glscompstyle@longragged3col}%
9305 }%
```

Backward compatible altlongragged4col style.

```
9306 \compatglossarystyle{altlongragged4col}{%
9307 \renewcommand*{\glossaryentryfield}[5]{%
9308 \glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%
9309 \renewcommand*{\glossarysubentryfield}[6]{%
9310 &
9311 \glssubentryitem{##2}%
9312 \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%
9313 }%
```

Backward compatible altlongragged4colheader style.

```
9314 \compatglossarystyle{altlongragged4colheader}{%
9315 \csuse{@glscompstyle@altlong4col}%
9316 }%
```

Backward compatible altlongragged4colborder style.

```
9317 \compatglossarystyle{altlongragged4colborder}{%
9318 \csuse{@glscompstyle@altlong4col}%
9319 }%
```

Backward compatible altlongragged4colheaderborder style.

```
9320 \compatglossarystyle{altlongragged4colheaderborder}{%
9321 \csuse{@glscompstyle@altlong4col}%
9322 }%
```

Backward compatible index style.

```
9323 \compatglossarystyle{index}{%
9324 \renewcommand*{\glossaryentryfield}[5]{%
9325 \item\glstarget{##1}\textbf{\glstarget{##1}{##2}}%
9326 \ifx\relax##4\relax
9327 \else
9328 \space{##4}%
9329 \fi
9330 \space ##3\glspostdescription \space ##5}%
9331 \renewcommand*{\glossarysubentryfield}[6]{%
9332 \ifcase##1\relax
9333 % level 0
9334 \item
9335 \or
9336 % level 1
9337 \subitem
9338 \glssubentryitem{##2}%
9339 \else
9340 % all other levels
9341 \subsubitem
9342 \fi
```



```

9343 \textbf{\glstarget{##2}{##3}}%
9344 \ifx\relax##5\relax
9345 \else
9346 \space(##5)%
9347 \fi
9348 \space##4\glspostdescription\space ##6}%
9349 }%

```

Backward compatible indexgroup style.

```

9350 \compatglossarystyle{indexgroup}{%
9351 \csuse{@glscmpstyle@index}%
9352 }%

```

Backward compatible indexhypergroup style.

```

9353 \compatglossarystyle{indexhypergroup}{%
9354 \csuse{@glscmpstyle@index}%
9355 }%

```

Backward compatible tree style.

```

9356 \compatglossarystyle{tree}{%
9357 \renewcommand{\glossaryentryfield}[5]{%
9358 \hangindent0pt\relax
9359 \parindent0pt\relax
9360 \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
9361 \ifx\relax##4\relax
9362 \else
9363 \space(##4)%
9364 \fi
9365 \space ##3\glspostdescription \space ##5\par}%
9366 \renewcommand{\glossarysubentryfield}[6]{%
9367 \hangindent##1\glstreeindent\relax
9368 \parindent##1\glstreeindent\relax
9369 \ifnum##1=1\relax
9370 \glssubentryitem{##2}%
9371 \fi
9372 \textbf{\glstarget{##2}{##3}}%
9373 \ifx\relax##5\relax
9374 \else
9375 \space(##5)%
9376 \fi
9377 \space##4\glspostdescription\space ##6\par}%
9378 }%

```

Backward compatible treegroup style.

```

9379 \compatglossarystyle{treegroup}{%
9380 \csuse{@glscmpstyle@tree}%
9381 }%

```

Backward compatible treehypergroup style.

```

9382 \compatglossarystyle{treehypergroup}{%
9383 \csuse{@glscmpstyle@tree}%
9384 }%

```

Backward compatible treenoname style.

```

9385 \compatglossarystyle{treenoname}{%
9386   \renewcommand{\glossaryentryfield}[5]{%
9387     \hangindent0pt\relax
9388     \parindent0pt\relax
9389     \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
9390     \ifx\relax##4\relax
9391     \else
9392       \space{##4}%
9393     \fi
9394     \space ##3\glspostdescription \space ##5\par}%
9395 \renewcommand{\glossarysubentryfield}[6]{%
9396   \hangindent##1\glstreeindent\relax
9397   \parindent##1\glstreeindent\relax
9398   \ifnum##1=1\relax
9399     \glssubentryitem{##2}%
9400   \fi
9401   \glstarget{##2}{\strut}%
9402   ##4\glspostdescription\space ##6\par}%
9403 }%

```

Backward compatible treenonamegroup style.

```

9404 \compatglossarystyle{treenonamegroup}{%
9405   \csuse{@glscompstyle@treenoname}%
9406 }%

```

Backward compatible treenonamehypergroup style.

```

9407 \compatglossarystyle{treenonamehypergroup}{%
9408   \csuse{@glscompstyle@treenoname}%
9409 }%

```

Backward compatible alttree style.

```

9410 \compatglossarystyle{alttree}{%
9411   \renewcommand{\glossaryentryfield}[5]{%
9412     \ifnum\@gls@prevlevel=0\relax
9413     \else
9414       \settowidth{\glstreeindent}{\textbf{\@glswidestname\space}}%
9415       \hangindent\glstreeindent
9416       \parindent\glstreeindent
9417     \fi
9418     \makebox[0pt][r]{\makebox[\glstreeindent][l]{%
9419       \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}}%
9420     \ifx\relax##4\relax
9421     \else
9422       (##4)\space
9423     \fi
9424     ##3\glspostdescription \space ##5\par
9425     \def\@gls@prevlevel{0}%
9426   }%
9427   \renewcommand{\glossarysubentryfield}[6]{%

```

```

9428 \ifnum##1=1\relax
9429 \glssubentryitem{##2}%
9430 \fi
9431 \ifnum\@gls@prevlevel=##1\relax
9432 \else
9433 \@ifundefined{@glswidestname\romannumeral##1}{%
9434 \settowidth{\gls@tmplen}{\textbf{\@glswidestname\space}}{%
9435 \settowidth{\gls@tmplen}{\textbf{%
9436 \csname @glswidestname\romannumeral##1\endcsname\space}}}%
9437 \ifnum\@gls@prevlevel<##1\relax
9438 \setlength\glstreeindent\gls@tmplen
9439 \addtolength\glstreeindent\parindent
9440 \parindent\glstreeindent
9441 \else
9442 \@ifundefined{@glswidestname\romannumeral\@gls@prevlevel}{%
9443 \settowidth{\glstreeindent}{\textbf{%
9444 \@glswidestname\space}}{%
9445 \settowidth{\glstreeindent}{\textbf{%
9446 \csname @glswidestname\romannumeral\@gls@prevlevel
9447 \endcsname\space}}}%
9448 \addtolength\parindent{-\glstreeindent}%
9449 \setlength\glstreeindent\parindent
9450 \fi
9451 \fi
9452 \hangindent\glstreeindent
9453 \makebox[0pt][r]{\makebox[\gls@tmplen][l]{%
9454 \textbf{\glstarget{##2}{##3}}}%
9455 \ifx##5\relax\relax
9456 \else
9457 (##5)\space
9458 \fi
9459 ##4\glspostdescription\space ##6\par
9460 \def\@gls@prevlevel{##1}%
9461 }%
9462 }%

```

Backward compatible alttreegroup style.

```

9463 \compatglossarystyle{alttreegroup}{%
9464 \csuse{@glscompstyle@alttree}%
9465 }%

```

Backward compatible alttreehypergroup style.

```

9466 \compatglossarystyle{alttreehypergroup}{%
9467 \csuse{@glscompstyle@alttree}%
9468 }%

```

Backward compatible mcolindex style.

```

9469 \compatglossarystyle{mcolindex}{%
9470 \csuse{@glscompstyle@index}%
9471 }%

```

Backward compatible mcolindexgroup style.
9472 \compatglossarystyle{mcolindexgroup}{%
9473 \csuse{@glscompstyle@index}%
9474 }%

Backward compatible mcolindexhypergroup style.
9475 \compatglossarystyle{mcolindexhypergroup}{%
9476 \csuse{@glscompstyle@index}%
9477 }%

Backward compatible mcoltree style.
9478 \compatglossarystyle{mcoltree}{%
9479 \csuse{@glscompstyle@tree}%
9480 }%

Backward compatible mcoltreegroup style.
9481 \compatglossarystyle{mcolindextreegroup}{%
9482 \csuse{@glscompstyle@tree}%
9483 }%

Backward compatible mcoltreehypergroup style.
9484 \compatglossarystyle{mcolindextreehypergroup}{%
9485 \csuse{@glscompstyle@tree}%
9486 }%

Backward compatible mcoltreename style.
9487 \compatglossarystyle{mcoltreename}{%
9488 \csuse{@glscompstyle@tree}%
9489 }%

Backward compatible mcoltreenamegroup style.
9490 \compatglossarystyle{mcoltreenamegroup}{%
9491 \csuse{@glscompstyle@tree}%
9492 }%

Backward compatible mcoltreenamehypergroup style.
9493 \compatglossarystyle{mcoltreenamehypergroup}{%
9494 \csuse{@glscompstyle@tree}%
9495 }%

Backward compatible mcolalttree style.
9496 \compatglossarystyle{mcolalttree}{%
9497 \csuse{@glscompstyle@alttree}%
9498 }%

Backward compatible mcolalttreegroup style.
9499 \compatglossarystyle{mcolalttreegroup}{%
9500 \csuse{@glscompstyle@alttree}%
9501 }%

Backward compatible mcolalttreehypergroup style.
9502 \compatglossarystyle{mcolalttreehypergroup}{%
9503 \csuse{@glscompstyle@alttree}%
9504 }%

Backward compatible superragged style.

```
9505 \compatglossarystyle{superragged}{%
9506   \renewcommand*{\glossaryentryfield}[5]{%
9507     \glstarget{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
9508     \tabularnewline}%
9509   \renewcommand*{\glossarysubentryfield}[6]{%
9510     &
9511     \glssubentryitem{##2}%
9512     \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
9513     \tabularnewline}%
9514 }%
```

Backward compatible superraggedborder style.

```
9515 \compatglossarystyle{superraggedborder}{%
9516   \csuse{@glscompstyle@superragged}%
9517 }%
```

Backward compatible superraggedheader style.

```
9518 \compatglossarystyle{superraggedheader}{%
9519   \csuse{@glscompstyle@superragged}%
9520 }%
```

Backward compatible superraggedheaderborder style.

```
9521 \compatglossarystyle{superraggedheaderborder}{%
9522   \csuse{@glscompstyle@superragged}%
9523 }%
```

Backward compatible superragged3col style.

```
9524 \compatglossarystyle{superragged3col}{%
9525   \renewcommand*{\glossaryentryfield}[5]{%
9526     \glstarget{##1}\glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%
9527   \renewcommand*{\glossarysubentryfield}[6]{%
9528     &
9529     \glssubentryitem{##2}%
9530     \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%
9531 }%
```

Backward compatible superragged3colborder style.

```
9532 \compatglossarystyle{superragged3colborder}{%
9533   \csuse{@glscompstyle@superragged3col}%
9534 }%
```

Backward compatible superragged3colheader style.

```
9535 \compatglossarystyle{superragged3colheader}{%
9536   \csuse{@glscompstyle@superragged3col}%
9537 }%
```

Backward compatible superragged3colheaderborder style.

```
9538 \compatglossarystyle{superragged3colheaderborder}{%
9539   \csuse{@glscompstyle@superragged3col}%
9540 }%
```

Backward compatible altsuperragged4col style.

```
9541 \compatglossarystyle{altsuperragged4col}{%
9542   \renewcommand*{\glossaryentryfield}[5]{%
9543     \glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%
9544   \renewcommand*{\glossarysubentryfield}[6]{%
9545     &
9546     \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%
9547   \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%
9548 }
```

Backward compatible altsuperragged4colheader style.

```
9549 \compatglossarystyle{altsuperragged4colheader}{%
9550   \csuse{@glscompstyle@altsuperragged4col}%
9551 }
```

Backward compatible altsuperragged4colborder style.

```
9552 \compatglossarystyle{altsuperragged4colborder}{%
9553   \csuse{@glscompstyle@altsuperragged4col}%
9554 }
```

Backward compatible altsuperragged4colheaderborder style.

```
9555 \compatglossarystyle{altsuperragged4colheaderborder}{%
9556   \csuse{@glscompstyle@altsuperragged4col}%
9557 }
```

Backward compatible super style.

```
9558 \compatglossarystyle{super}{%
9559   \renewcommand*{\glossaryentryfield}[5]{%
9560     \glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
9561   \renewcommand*{\glossarysubentryfield}[6]{%
9562     &
9563     \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
9564   \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
9565 }
```

Backward compatible superborder style.

```
9566 \compatglossarystyle{superborder}{%
9567   \csuse{@glscompstyle@super}%
9568 }
```

Backward compatible superheader style.

```
9569 \compatglossarystyle{superheader}{%
9570   \csuse{@glscompstyle@super}%
9571 }
```

Backward compatible superheaderborder style.

```
9572 \compatglossarystyle{superheaderborder}{%
9573   \csuse{@glscompstyle@super}%
9574 }
```

Backward compatible super3col style.

```
9575 \compatglossarystyle{super3col}{%
```

```

9576 \renewcommand*\glossaryentryfield}[5]{%
9577   \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\\}%
9578 \renewcommand*\glossarysubentryfield}[6]{%
9579   &
9580   \glssubentryitem{##2}%
9581   \glstarget{##2}{\strut}##4 & ##6\\}%
9582 }%

  Backward compatible super3colborder style.
9583 \compatglossarystyle{super3colborder}{%
9584 \csuse{@glscompstyle@super3col}%
9585 }%

  Backward compatible super3colheader style.
9586 \compatglossarystyle{super3colheader}{%
9587 \csuse{@glscompstyle@super3col}%
9588 }%

  Backward compatible super3colheaderborder style.
9589 \compatglossarystyle{super3colheaderborder}{%
9590 \csuse{@glscompstyle@super3col}%
9591 }%

  Backward compatible super4col style.
9592 \compatglossarystyle{super4col}{%
9593   \renewcommand*\glossaryentryfield}[5]{%
9594     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%
9595   \renewcommand*\glossarysubentryfield}[6]{%
9596     &
9597     \glssubentryitem{##2}%
9598     \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%
9599 }%

  Backward compatible super4colheader style.
9600 \compatglossarystyle{super4colheader}{%
9601 \csuse{@glscompstyle@super4col}%
9602 }%

  Backward compatible super4colborder style.
9603 \compatglossarystyle{super4colborder}{%
9604 \csuse{@glscompstyle@super4col}%
9605 }%

  Backward compatible super4colheaderborder style.
9606 \compatglossarystyle{super4colheaderborder}{%
9607 \csuse{@glscompstyle@super4col}%
9608 }%

  Backward compatible altsuper4col style.
9609 \compatglossarystyle{altsuper4col}{%
9610 \csuse{@glscompstyle@super4col}%
9611 }%

```

Backward compatible altsuper4colheader style.

```
9612 \compatglossarystyle{altsuper4colheader}{%
9613 \csuse{@glscompstyle@super4col}%
9614 }%
```

Backward compatible altsuper4colborder style.

```
9615 \compatglossarystyle{altsuper4colborder}{%
9616 \csuse{@glscompstyle@super4col}%
9617 }%
```

Backward compatible altsuper4colheaderborder style.

```
9618 \compatglossarystyle{altsuper4colheaderborder}{%
9619 \csuse{@glscompstyle@super4col}%
9620 }%
```

5 Accessibility Support (glossaries-accsupp Code)

The package is experimental. It is intended to provide a means of using the PDF accessibility support in glossary entries. See the documentation for further details about accessibility support.

```
9621 \NeedsTeXFormat{LaTeX2e}
```

Package version number now in line with main glossaries package number but will only be updated when glossaries-accsupp.sty is modified.

```
9622 \ProvidesPackage{glossaries-accsupp}[2015/09/09 v4.18 (NLCT)
9623 Experimental glossaries accessibility]
```

Pass all options to glossaries:

```
9624 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
9625 \ProcessOptions
```

`compatibleglossentry` Override style compatibility macros:

```
9626 \def\compatibleglossentry#1#2{%
9627 \toks@{#2}%
9628 \protected@edef\@do@glossentry{%
9629 \noexpand\accsuppglossaryentryfield{#1}%
9630 {\noexpand\glsnamefont
9631 {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@name\endcsname}}%
9632 {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@desc\endcsname}}%
9633 {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@symbol\endcsname}}%
9634 {\the\toks@}%
9635 }%
9636 \@do@glossentry
9637 }
```

`compatiblesubglossentry`

```
9638 \def\compatiblesubglossentry#1#2#3{%
```



```

9639 \toks@{#3}%
9640 \protected@edef\@do@subglossentry{%
9641 \noexpand\accsuppglossarysubentryfield{\number#1}%
9642 {#2}%
9643 {\noexpand\glsnamefont
9644 {\expandafter\expandonce\csname glo@\glsdetoklabel{#2}@name\endcsname}}%
9645 {\expandafter\expandonce\csname glo@\glsdetoklabel{#2}@desc\endcsname}%
9646 {\expandafter\expandonce\csname glo@\glsdetoklabel{#2}@symbol\endcsname}%
9647 {\the\toks@}%
9648 }%
9649 \@do@subglossentry
9650 }

```

Required packages:

```

9651 \RequirePackage{glossaries}
9652 \RequirePackage{accsupp}

```

5.1 Defining Replacement Text

The version 0.1 stored the replacement text in the symbol key. This has been changed to use the new keys defined here. Example of use:

```
\newglossaryentry{dr}{name=Dr ,description={},access={Doctor}}
```

access The replacement text corresponding to the name key:

```

9653 \define@key{glossentry}{access}{%
9654 \def\@glo@access{#1}%
9655 }

```

textaccess The replacement text corresponding to the text key:

```

9656 \define@key{glossentry}{textaccess}{%
9657 \def\@glo@textaccess{#1}%
9658 }

```

firstaccess The replacement text corresponding to the first key:

```

9659 \define@key{glossentry}{firstaccess}{%
9660 \def\@glo@firstaccess{#1}%
9661 }

```

pluralaccess The replacement text corresponding to the plural key:

```

9662 \define@key{glossentry}{pluralaccess}{%
9663 \def\@glo@pluralaccess{#1}%
9664 }

```

firstpluralaccess The replacement text corresponding to the firstplural key:

```

9665 \define@key{glossentry}{firstpluralaccess}{%
9666 \def\@glo@firstpluralaccess{#1}%
9667 }

```

symbolaccess The replacement text corresponding to the symbol key:

```
9668 \define@key{glossentry}{symbolaccess}{%
9669   \def\@glo@symbolaccess{#1}%
9670 }
```

symbolpluralaccess The replacement text corresponding to the symbolplural key:

```
9671 \define@key{glossentry}{symbolpluralaccess}{%
9672   \def\@glo@symbolpluralaccess{#1}%
9673 }
```

descriptionaccess The replacement text corresponding to the description key:

```
9674 \define@key{glossentry}{descriptionaccess}{%
9675   \def\@glo@descaccess{#1}%
9676 }
```

descriptionpluralaccess The replacement text corresponding to the descriptionplural key:

```
9677 \define@key{glossentry}{descriptionpluralaccess}{%
9678   \def\@glo@descpluralaccess{#1}%
9679 }
```

shortaccess The replacement text corresponding to the short key:

```
9680 \define@key{glossentry}{shortaccess}{%
9681   \def\@glo@shortaccess{#1}%
9682 }
```

shortpluralaccess The replacement text corresponding to the shortplural key:

```
9683 \define@key{glossentry}{shortpluralaccess}{%
9684   \def\@glo@shortpluralaccess{#1}%
9685 }
```

longaccess The replacement text corresponding to the long key:

```
9686 \define@key{glossentry}{longaccess}{%
9687   \def\@glo@longaccess{#1}%
9688 }
```

longpluralaccess The replacement text corresponding to the longplural key:

```
9689 \define@key{glossentry}{longpluralaccess}{%
9690   \def\@glo@longpluralaccess{#1}%
9691 }
```

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:

```
9692 \appto\@gls@keymap{,%
9693   {access}{access},%
9694   {textaccess}{textaccess},%
9695   {firstaccess}{firstaccess},%
```

```

9696 {pluralaccess}{pluralaccess},%
9697 {firstpluralaccess}{firstpluralaccess},%
9698 {symbolaccess}{symbolaccess},%
9699 {symbolpluralaccess}{symbolpluralaccess},%
9700 {descaccess}{descaccess},%
9701 {descpluralaccess}{descpluralaccess},%
9702 {shortaccess}{shortaccess},%
9703 {shortpluralaccess}{shortpluralaccess},%
9704 {longaccess}{longaccess},%
9705 {longpluralaccess}{longpluralaccess}%
9706 }

```

`\@gls@noaccess` Indicates that no replacement text has been provided.

```

9707 \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):

```

9708 \let\@gls@oldnewglossaryentryprehook\@newglossaryentryprehook
9709 \renewcommand*\@newglossaryentryprehook{%
9710   \@gls@oldnewglossaryentryprehook
9711   \def\@glo@access{\@glo@symbol}%

```

Initialise the other keys:

```

9712   \def\@glo@textaccess{\@glo@access}%
9713   \def\@glo@firstaccess{\@glo@access}%
9714   \def\@glo@pluralaccess{\@glo@textaccess}%
9715   \def\@glo@firstpluralaccess{\@glo@pluralaccess}%
9716   \def\@glo@symbolaccess{\relax}%
9717   \def\@glo@symbolpluralaccess{\@glo@symbolaccess}%
9718   \def\@glo@descaccess{\relax}%
9719   \def\@glo@descpluralaccess{\@glo@descaccess}%
9720   \def\@glo@shortaccess{\relax}%
9721   \def\@glo@shortpluralaccess{\@glo@shortaccess}%
9722   \def\@glo@longaccess{\relax}%
9723   \def\@glo@longpluralaccess{\@glo@longaccess}%
9724 }

```

Add to the end hook:

```

9725 \let\@gls@oldnewglossaryentryposthook\@newglossaryentryposthook
9726 \renewcommand*\@newglossaryentryposthook{%
9727   \@gls@oldnewglossaryentryposthook

```

Store the access information:

```

9728   \expandafter
9729     \protected@xdef\csname glo@\@glo@label @access\endcsname{%
9730     \@glo@access}%
9731   \expandafter
9732     \protected@xdef\csname glo@\@glo@label @textaccess\endcsname{%
9733     \@glo@textaccess}%
9734   \expandafter

```

```

9735 \protected@xdef\csname glo@\@glo@label @firstaccess\endcsname{%
9736 \@glo@firstaccess}%
9737 \expandafter
9738 \protected@xdef\csname glo@\@glo@label @pluralaccess\endcsname{%
9739 \@glo@pluralaccess}%
9740 \expandafter
9741 \protected@xdef\csname glo@\@glo@label @firstpluralaccess\endcsname{%
9742 \@glo@firstpluralaccess}%
9743 \expandafter
9744 \protected@xdef\csname glo@\@glo@label @symbolaccess\endcsname{%
9745 \@glo@symbolaccess}%
9746 \expandafter
9747 \protected@xdef\csname glo@\@glo@label @symbolpluralaccess\endcsname{%
9748 \@glo@symbolpluralaccess}%
9749 \expandafter
9750 \protected@xdef\csname glo@\@glo@label @descaccess\endcsname{%
9751 \@glo@descaccess}%
9752 \expandafter
9753 \protected@xdef\csname glo@\@glo@label @descpluralaccess\endcsname{%
9754 \@glo@descpluralaccess}%
9755 \expandafter
9756 \protected@xdef\csname glo@\@glo@label @shortaccess\endcsname{%
9757 \@glo@shortaccess}%
9758 \expandafter
9759 \protected@xdef\csname glo@\@glo@label @shortpluralaccess\endcsname{%
9760 \@glo@shortpluralaccess}%
9761 \expandafter
9762 \protected@xdef\csname glo@\@glo@label @longaccess\endcsname{%
9763 \@glo@longaccess}%
9764 \expandafter
9765 \protected@xdef\csname glo@\@glo@label @longpluralaccess\endcsname{%
9766 \@glo@longpluralaccess}%
9767 }

```

5.2 Accessing Replacement Text

`\glsentryaccess` Get the value of the access key for the entry with the given label:

```

9768 \newcommand*\glsentryaccess}[1]{%
9769 \@gls@entry@field{#1}{access}%
9770 }

```

`\glsentrytextaccess` Get the value of the textaccess key for the entry with the given label:

```

9771 \newcommand*\glsentrytextaccess}[1]{%
9772 \@gls@entry@field{#1}{textaccess}%
9773 }

```

`\glsentryfirstaccess` Get the value of the firstaccess key for the entry with the given label:

```

9774 \newcommand*\glsentryfirstaccess}[1]{%
9775 \@gls@entry@field{#1}{firstaccess}%

```

9776 }

`\glsentrypluralaccess` Get the value of the `pluralaccess` key for the entry with the given label:

```
9777 \newcommand*{\glsentrypluralaccess}[1]{%
9778   \@gls@entry@field{#1}{pluralaccess}%
9779 }
```

`\glsentryfirstpluralaccess` Get the value of the `firstpluralaccess` key for the entry with the given label:

```
9780 \newcommand*{\glsentryfirstpluralaccess}[1]{%
9781   \csname glo@#1@firstpluralaccess\endcsname
9782 }
```

`\glsentrysymbolaccess` Get the value of the `symbolaccess` key for the entry with the given label:

```
9783 \newcommand*{\glsentrysymbolaccess}[1]{%
9784   \@gls@entry@field{#1}{symbolaccess}%
9785 }
```

`\glsentrysymbolpluralaccess` Get the value of the `symbolpluralaccess` key for the entry with the given label:

```
9786 \newcommand*{\glsentrysymbolpluralaccess}[1]{%
9787   \@gls@entry@field{#1}{symbolpluralaccess}%
9788 }
```

`\glsentrydescaccess` Get the value of the `descriptionaccess` key for the entry with the given label:

```
9789 \newcommand*{\glsentrydescaccess}[1]{%
9790   \@gls@entry@field{#1}{descaccess}%
9791 }
```

`\glsentrydescpluralaccess` Get the value of the `descriptionpluralaccess` key for the entry with the given label:

```
9792 \newcommand*{\glsentrydescpluralaccess}[1]{%
9793   \@gls@entry@field{#1}{descaccess}%
9794 }
```

`\glsentryshortaccess` Get the value of the `shortaccess` key for the entry with the given label:

```
9795 \newcommand*{\glsentryshortaccess}[1]{%
9796   \@gls@entry@field{#1}{shortaccess}%
9797 }
```

`\glsentryshortpluralaccess` Get the value of the `shortpluralaccess` key for the entry with the given label:

```
9798 \newcommand*{\glsentryshortpluralaccess}[1]{%
9799   \@gls@entry@field{#1}{shortpluralaccess}%
9800 }
```

`\glsentrylongaccess` Get the value of the `longaccess` key for the entry with the given label:

```
9801 \newcommand*{\glsentrylongaccess}[1]{%
9802   \@gls@entry@field{#1}{longaccess}%
9803 }
```

trylongpluralaccess Get the value of the longpluralaccess key for the entry with the given label:

```
9804 \newcommand*\glsentrylongpluralaccess}[1]{%
9805   \@gls@entry@field{#1}{longpluralaccess}%
9806 }
```

`\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.)

```
9807 \newcommand*\glsaccsupp}[2]{%
9808   \BeginAccSupp{ActualText=#1}#2\EndAccSupp{}%
9809 }
```

`\xglsaccsupp` Fully expands replacement text before calling `\glsaccsupp`

```
9810 \newcommand*\xglsaccsupp}[2]{%
9811   \protected@edef\@gls@replacementtext{#1}%
9812   \expandafter\glsaccsupp\expandafter{\@gls@replacementtext}{#2}%
9813 }
```

`@gls@access@display`

```
9814 \newcommand*\@gls@access@display}[2]{%
9815   \protected@edef\@glo@access{#2}%
9816   \ifx\@glo@access\@gls@noaccess
9817     #1%
9818   \else
9819     \xglsaccsupp{\@glo@access}{#1}%
9820   \fi
9821 }
```

`l$nameaccessdisplay` Displays the first argument with the accessibility text for the entry with the label given by the second argument (if set).

```
9822 \DeclareRobustCommand*\glsnameaccessdisplay}[2]{%
9823   \@gls@access@display{#1}{\glsentryaccess{#2}}%
9824 }
```

`l$textaccessdisplay` As above but for the textaccess replacement text.

```
9825 \DeclareRobustCommand*\gls$textaccessdisplay}[2]{%
9826   \@gls@access@display{#1}{\glsentrytextaccess{#2}}%
9827 }
```

`pluralaccessdisplay` As above but for the pluralaccess replacement text.

```
9828 \DeclareRobustCommand*\glspluralaccessdisplay}[2]{%
9829   \@gls@access@display{#1}{\glsentrypluralaccess{#2}}%
9830 }
```

`sfirstaccessdisplay` As above but for the firstaccess replacement text.

```
9831 \DeclareRobustCommand*\glsfirstaccessdisplay}[2]{%
9832   \@gls@access@display{#1}{\glsentryfirstaccess{#2}}%
9833 }
```

pluralaccessdisplay As above but for the firstpluralaccess replacement text.

```

9834 \DeclareRobustCommand*\glstfirstpluralaccessdisplay}[2]{%
9835   \@gls@access@display{#1}{\glstentryfirstpluralaccess{#2}}%
9836 }

```

symbolaccessdisplay As above but for the symbolaccess replacement text.

```

9837 \DeclareRobustCommand*\glssymbolaccessdisplay}[2]{%
9838   \@gls@access@display{#1}{\glstentrysymbolaccess{#2}}%
9839 }

```

pluralaccessdisplay As above but for the symbolpluralaccess replacement text.

```

9840 \DeclareRobustCommand*\glssymbolpluralaccessdisplay}[2]{%
9841   \@gls@access@display{#1}{\glstentrysymbolpluralaccess{#2}}%
9842 }

```

descriptionaccessdisplay As above but for the descriptionaccess replacement text.

```

9843 \DeclareRobustCommand*\glsdescriptionaccessdisplay}[2]{%
9844   \@gls@access@display{#1}{\glstentrydescaccess{#2}}%
9845 }

```

pluralaccessdisplay As above but for the descriptionpluralaccess replacement text.

```

9846 \DeclareRobustCommand*\glsdescriptionpluralaccessdisplay}[2]{%
9847   \@gls@access@display{#1}{\glstentrydescpluralaccess{#2}}%
9848 }

```

shortaccessdisplay As above but for the shortaccess replacement text.

```

9849 \DeclareRobustCommand*\glsshortaccessdisplay}[2]{%
9850   \@gls@access@display{#1}{\glstentryshortaccess{#2}}%
9851 }

```

pluralaccessdisplay As above but for the shortpluralaccess replacement text.

```

9852 \DeclareRobustCommand*\glsshortpluralaccessdisplay}[2]{%
9853   \@gls@access@display{#1}{\glstentryshortpluralaccess{#2}}%
9854 }

```

longaccessdisplay As above but for the longaccess replacement text.

```

9855 \DeclareRobustCommand*\glslongaccessdisplay}[2]{%
9856   \@gls@access@display{#1}{\glstentrylongaccess{#2}}%
9857 }

```

pluralaccessdisplay As above but for the longpluralaccess replacement text.

```

9858 \DeclareRobustCommand*\glslongpluralaccessdisplay}[2]{%
9859   \@gls@access@display{#1}{\glstentrylongpluralaccess{#2}}%
9860 }

```

\glsaccessdisplay Gets the replacement text corresponding to the named key given by the first argument and calls the appropriate command defined above.

```

9861 \DeclareRobustCommand*\glsaccessdisplay}[3]{%
9862   \@ifundefined{gls#1accessdisplay}%
9863   {%
9864     \PackageError{glossaries-accsupp}{No accessibility support
9865       for key ‘#1’}{}%
9866   }%
9867   {%
9868     \csname gls#1accessdisplay\endcsname{#2}{#3}%
9869   }%
9870 }

```

l@s@default@entryfmt Redefine the default entry format to use accessibility information

```

9871 \renewcommand*\@@gls@default@entryfmt}[2]{%
9872   \ifdefempty\glscustomtext
9873   {%
9874     \glsifplural
9875     {%
          Plural form
9876       \glsapscase
9877       {%
          Don't adjust case
9878         \ifglsused\glslabel
9879         {%
          Subsequent use
9880           #2{\glspluralaccessdisplay
9881             {\glsentryplural{\glslabel}}{\glslabel}}%
9882           {\glsdescriptionpluralaccessdisplay
9883             {\glsentrydescplural{\glslabel}}{\glslabel}}%
9884           {\glsymbolpluralaccessdisplay
9885             {\glsentrysymbolplural{\glslabel}}{\glslabel}}
9886           {\glsinsert}}%
9887         }%
9888       {%
          First use
9889         #1{\glsfirstpluralaccessdisplay
9890           {\glsentryfirstplural{\glslabel}}{\glslabel}}%
9891         {\glsdescriptionpluralaccessdisplay
9892           {\glsentrydescplural{\glslabel}}{\glslabel}}%
9893         {\glsymbolpluralaccessdisplay
9894           {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
9895         {\glsinsert}}%
9896       }%
9897     }%
9898   }%
          Make first letter upper case
9899     \ifglsused\glslabel
9900     {%

```


Subsequent use.

```
9901      #2{\glspluralaccessdisplay
9902          {\Glsentryplural{\glslabel}}{\glslabel}}%
9903          {\glsdescriptionpluralaccessdisplay
9904             {\glsentrydescplural{\glslabel}}{\glslabel}}%
9905          {\glsymbolpluralaccessdisplay
9906             {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
9907          {\glsinsert}}%
9908      }%
9909      {%
```

First use

```
9910      #1{\glsfirstpluralaccessdisplay
9911          {\Glsentryfirstplural{\glslabel}}{\glslabel}}%
9912          {\glsdescriptionpluralaccessdisplay
9913             {\glsentrydescplural{\glslabel}}{\glslabel}}%
9914          {\glsymbolpluralaccessdisplay
9915             {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
9916          {\glsinsert}}%
9917      }%
9918      }%
9919      {%
```

Make all upper case

```
9920      \ifglsused\glslabel
9921      {%
```

Subsequent use

```
9922      \MakeUppercase{%
9923          #2{\glspluralaccessdisplay
9924             {\glsentryplural{\glslabel}}{\glslabel}}%
9925             {\glsdescriptionpluralaccessdisplay
9926                {\glsentrydescplural{\glslabel}}{\glslabel}}%
9927             {\glsymbolpluralaccessdisplay
9928                {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
9929             {\glsinsert}}%
9930      }%
9931      {%
```

First use

```
9932      \MakeUppercase{%
9933          #1{\glsfirstpluralaccessdisplay
9934             {\glsentryfirstplural{\glslabel}}{\glslabel}}%
9935             {\glsdescriptionpluralaccessdisplay
9936                {\glsentrydescplural{\glslabel}}{\glslabel}}%
9937             {\glsymbolpluralaccessdisplay
9938                {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
9939             {\glsinsert}}%
9940      }%
9941      }%
9942      }%
```

```

9943     {%
        Singular form
9944     \glscapscase
9945     {%
        Don't adjust case
9946     \ifglused\glslabel
9947     {%
        Subsequent use
9948     #2{\glstextaccessdisplay
9949         {\glsentrytext{\glslabel}}{\glslabel}}%
9950     {\glsdescriptionaccessdisplay
9951         {\glsentrydesc{\glslabel}}{\glslabel}}%
9952     {\glssymbolaccessdisplay
9953         {\glsentrysymbol{\glslabel}}{\glslabel}}%
9954     {\glsinsert}%
9955     }%
9956     {%
        First use
9957     #1{\glsfirstaccessdisplay
9958         {\glsentryfirst{\glslabel}}{\glslabel}}%
9959     {\glsdescriptionaccessdisplay
9960         {\glsentrydesc{\glslabel}}{\glslabel}}%
9961     {\glssymbolaccessdisplay
9962         {\glsentrysymbol{\glslabel}}{\glslabel}}%
9963     {\glsinsert}%
9964     }%
9965     }%
9966     {%
        Make first letter upper case
9967     \ifglused\glslabel
9968     {%
        Subsequent use
9969     #2{\glstextaccessdisplay
9970         {\Glsentrytext{\glslabel}}{\glslabel}}%
9971     {\glsdescriptionaccessdisplay
9972         {\glsentrydesc{\glslabel}}{\glslabel}}%
9973     {\glssymbolaccessdisplay
9974         {\glsentrysymbol{\glslabel}}{\glslabel}}%
9975     {\glsinsert}%
9976     }%
9977     {%
        First use
9978     #1{\glsfirstaccessdisplay
9979         {\Glsentryfirst{\glslabel}}{\glslabel}}%
9980     {\glsdescriptionaccessdisplay

```

```

9981         {\glsentrydesc{\glslabel}}{\glslabel}}%
9982         {\glssymbolaccessdisplay
9983         {\glsentrysymbol{\glslabel}}{\glslabel}}%
9984         {\glsinsert}}%
9985     }%
9986 }%
9987 {%

```

Make all upper case

```

9988     \ifglsused\glslabel
9989     {%

```

Subsequent use

```

9990         \MakeUppercase{%
9991         #2{\glstextaccessdisplay
9992         {\glsentrytext{\glslabel}}{\glslabel}}%
9993         {\glsdescriptionaccessdisplay
9994         {\glsentrydesc{\glslabel}}{\glslabel}}%
9995         {\glssymbolaccessdisplay
9996         {\glsentrysymbol{\glslabel}}{\glslabel}}%
9997         {\glsinsert}}%
9998     }%
9999     {%

```

First use

```

10000        \MakeUppercase{%
10001        #1{\glsfirstaccessdisplay
10002        {\glsentryfirst{\glslabel}}{\glslabel}}%
10003        {\glsdescriptionaccessdisplay
10004        {\glsentrydesc{\glslabel}}{\glslabel}}%
10005        {\glssymbolaccessdisplay
10006        {\glsentrysymbol{\glslabel}}{\glslabel}}%
10007        {\glsinsert}}%
10008    }%
10009 }%
10010 }%
10011 }%
10012 {%

```

Custom text provided in \glsdisp

```

10013     \ifglsused{\glslabel}%
10014     {%

```

Subsequent use

```

10015         #2{\glscustomtext}%
10016         {\glsdescriptionaccessdisplay
10017         {\glsentrydesc{\glslabel}}{\glslabel}}%
10018         {\glssymbolaccessdisplay
10019         {\glsentrysymbol{\glslabel}}{\glslabel}}%
10020         {\glsinsert}}%
10021     }%
10022     {%

```

First use

```
10023      #1{\glscustomtext}%
10024      {\glsdescriptionaccessdisplay
10025       {\glsentrydesc{\glslabel}}{\glslabel}}%
10026      {\glsymbolaccessdisplay
10027       {\glsentrysymbol{\glslabel}}{\glslabel}}%
10028      {\glsinsert}%
10029    }%
10030  }%
10031 }
```

`\glsgenentryfmt` Redefine to use accessibility information.

```
10032 \renewcommand*{\glsgenentryfmt}{%
10033   \ifdefempty\glscustomtext
10034   {%
10035     \glsifplural
10036     {%
```

Plural form

```
10037     \glscapscase
10038     {%
```

Don't adjust case

```
10039     \ifglsused\glslabel
10040     {%
```

Subsequent use

```
10041     \glspluralaccessdisplay
10042     {\glsentryplural{\glslabel}}{\glslabel}%
10043     \glsinsert
10044   }%
10045   {%
```

First use

```
10046     \glsfirstpluralaccessdisplay
10047     {\glsentryfirstplural{\glslabel}}{\glslabel}%
10048     \glsinsert
10049   }%
10050   }%
10051   {%
```

Make first letter upper case

```
10052     \ifglsused\glslabel
10053     {%
```

Subsequent use.

```
10054     \glspluralaccessdisplay
10055     {\Glsentryplural{\glslabel}}{\glslabel}%
10056     \glsinsert
10057   }%
10058   {%
```

First use

```
10059      \glsfirstpluralaccessdisplay
10060      {\Glsentryfirstplural{\glslabel}}{\glslabel}%
10061      \glsinsert
10062      }%
10063      }%
10064      {%
```

Make all upper case

```
10065      \ifglsused\glslabel
10066      {%
```

Subsequent use

```
10067      \glspluralaccessdisplay
10068      {\mfirstucMakeUppercase{\glsentryplural{\glslabel}}}%
10069      {\glslabel}%
10070      \mfirstucMakeUppercase{\glsinsert}%
10071      }%
10072      {%
```

First use

```
10073      \glsfirstpluralaccessdisplay
10074      {\mfirstucMakeUppercase{\glsentryfirstplural{\glslabel}}}%
10075      {\glslabel}%
10076      \mfirstucMakeUppercase{\glsinsert}%
10077      }%
10078      }%
10079      }%
10080      {%
```

Singular form

```
10081      \glscapscase
10082      {%
```

Don't adjust case

```
10083      \ifglsused\glslabel
10084      {%
```

Subsequent use

```
10085      \glstextaccessdisplay{\glsentrytext{\glslabel}}{\glslabel}%
10086      \glsinsert
10087      }%
10088      {%
```

First use

```
10089      \glsfirstaccessdisplay{\glsentryfirst{\glslabel}}{\glslabel}%
10090      \glsinsert
10091      }%
10092      }%
10093      {%
```

Make first letter upper case

```
10094 \ifglsused\glslabel
10095 {%
```

Subsequent use

```
10096 \glstextaccessdisplay{\Glsentrytext{\glslabel}}{\glslabel}%
10097 \glsinsert
10098 }%
10099 {%
```

First use

```
10100 \glsfirstaccessdisplay{\Glsentryfirst{\glslabel}}{\glslabel}%
10101 \glsinsert
10102 }%
10103 }%
10104 {%
```

Make all upper case

```
10105 \ifglsused\glslabel
10106 {%
```

Subsequent use

```
10107 \glstextaccessdisplay
10108 {\mfirstucMakeUppercase{\glstext{\glslabel}}{\glslabel}%
10109 \mfirstucMakeUppercase{\glsinsert}%
10110 }%
10111 {%
```

First use

```
10112 \glsfirstaccessdisplay
10113 {\mfirstucMakeUppercase{\glstext{\glslabel}}{\glslabel}%
10114 \mfirstucMakeUppercase{\glsinsert}%
10115 }%
10116 }%
10117 }%
10118 }%
10119 {%
```

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.

```
10120 \glscustomtext\glsinsert
10121 }%
10122 }
```

`\glsacfmnt` Redefine to include accessibility information.

```
10123 \renewcommand*{\glsacfmnt}{%
10124 \ifdefempty\glscustomtext
10125 {%
10126 \ifglsused\glslabel
10127 {%
```

Subsequent use:

10128 \glsifplural
10129 {%

Subsequent plural form:

10130 \glscapscase
10131 {%

Subsequent plural form, don't adjust case:

10132 \acronymfont
10133 {\glsshortpluralaccessdisplay
10134 {\glsentryshortpl{\glslabel}}{\glslabel}}%
10135 \glsinsert
10136 }%
10137 {%

Subsequent plural form, make first letter upper case:

10138 \acronymfont
10139 {\glsshortpluralaccessdisplay
10140 {\Glsentryshortpl{\glslabel}}{\glslabel}}%
10141 \glsinsert
10142 }%
10143 {%

Subsequent plural form, all caps:

10144 \mfirstucMakeUppercase
10145 {\acronymfont
10146 {\glsshortpluralaccessdisplay
10147 {\Glsentryshortpl{\glslabel}}{\glslabel}}%
10148 \glsinsert}%
10149 }%
10150 }%
10151 {%

Subsequent singular form

10152 \glscapscase
10153 {%

Subsequent singular form, don't adjust case:

10154 \acronymfont
10155 {\glsshortaccessdisplay{\glsentryshort{\glslabel}}{\glslabel}}%
10156 \glsinsert
10157 }%
10158 {%

Subsequent singular form, make first letter upper case:

10159 \acronymfont
10160 {\glsshortaccessdisplay{\Glsentryshort{\glslabel}}{\glslabel}}%
10161 \glsinsert
10162 }%
10163 {%

Subsequent singular form, all caps:

```
10164      \mfirstucMakeUppercase
10165      {\acronymfont{%
10166      \glsshortaccessdisplay{\glsentryshort{\glslabel}}{\glslabel}}%
10167      \glsinsert}%
10168      }%
10169      }%
10170      }%
10171      {%
```

First use:

```
10172      \glsifplural
10173      {%
```

First use plural form:

```
10174      \glscapscase
10175      {%
```

First use plural form, don't adjust case:

```
10176      \genplacrfullformat{\glslabel}{\glsinsert}%
10177      }%
10178      {%
```

First use plural form, make first letter upper case:

```
10179      \Genplacrfullformat{\glslabel}{\glsinsert}%
10180      }%
10181      {%
```

First use plural form, all caps:

```
10182      \mfirstucMakeUppercase
10183      {\genplacrfullformat{\glslabel}{\glsinsert}}%
10184      }%
10185      }%
10186      {%
```

First use singular form

```
10187      \glscapscase
10188      {%
```

First use singular form, don't adjust case:

```
10189      \genacrfullformat{\glslabel}{\glsinsert}%
10190      }%
10191      {%
```

First use singular form, make first letter upper case:

```
10192      \Genacrfullformat{\glslabel}{\glsinsert}%
10193      }%
10194      {%
```

First use singular form, all caps:

```
10195      \mfirstucMakeUppercase
10196      {\genacrfullformat{\glslabel}{\glsinsert}}%
10197      }%
```



```

10198     }%
10199     }%
10200     }%
10201     {%

```

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.

```

10202     \glscustomtext
10203     }%
10204 }

```

`\genacrfullformat` Redefine to include accessibility information.

```

10205 \renewcommand*{\genacrfullformat}[2]{%
10206     \glslongaccessdisplay{\glsentrylong{#1}}{#1}#2\space
10207     (\glsshortaccessdisplay{\protect\firstacronymfont{\glsentryshort{#1}}}{#1})%
10208 }

```

`\Genacrfullformat` Redefine to include accessibility information.

```

10209 \renewcommand*{\Genacrfullformat}[2]{%
10210     \glslongaccessdisplay{\Glsentrylong{#1}}{#1}#2\space
10211     (\glsshortaccessdisplay{\protect\firstacronymfont{\Glsentryshort{#1}}}{#1})%
10212 }

```

`\genplacrfullformat` Redefine to include accessibility information.

```

10213 \renewcommand*{\genplacrfullformat}[2]{%
10214     \glslongpluralaccessdisplay{\glsentrylongpl{#1}}{#1}#2\space
10215     (\glsshortpluralaccessdisplay
10216         {\protect\firstacronymfont{\glsentryshortpl{#1}}}{#1})%
10217 }

```

`\Genplacrfullformat` Redefine to include accessibility information.

```

10218 \renewcommand*{\Genplacrfullformat}[2]{%
10219     \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}#2\space
10220     (\glsshortpluralaccessdisplay
10221         {\protect\firstacronymfont{\Glsentryshortpl{#1}}}{#1})%
10222 }

```

`\@acrshort`

```

10223 \def\@acrshort#1#2[#3]{%
10224     \glsdoifexists{#2}%
10225     {%
10226         \let\do@gls@link@checkfirsthyper\relax
10227         \let\glsifplural\@secondoftwo
10228         \let\glscapscase\@firstofthree
10229         \let\glsinsert\@empty
10230         \def\glscustomtext{%
10231             \acronymfont{\glsshortaccessdisplay{\glsentryshort{#2}}{#2}}#3%
10232         }%

```

Call \@gls@link

```
10233 \gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%  
10234 }%  
  
10235 \glspostlinkhook  
10236 }
```

\@Acrshort

```
10237 \def\@Acrshort#1#2[#3]{%  
10238 \glsdoifexists{#2}%  
10239 {%  
10240 \let\do@gls@link@checkfirsthyper\relax  
  
10241 \let\glsifplural\@secondoftwo  
10242 \let\glscapscase\@secondofthree  
10243 \let\glsinsert\@empty  
10244 \def\glscustomtext{%  
10245 \acronymfont{\glsshortaccessdisplay{\Glsentryshort{#2}}{#2}}#3%  
10246 }%
```

Call \@gls@link

```
10247 \gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%  
10248 }%  
  
10249 \glspostlinkhook  
10250 }
```

\@ACRshort

```
10251 \def\@ACRshort#1#2[#3]{%  
10252 \glsdoifexists{#2}%  
10253 {%  
10254 \let\do@gls@link@checkfirsthyper\relax  
  
10255 \let\glsifplural\@secondoftwo  
10256 \let\glscapscase\@thirdofthree  
10257 \let\glsinsert\@empty  
10258 \def\glscustomtext{%  
10259 \acronymfont{\glsshortaccessdisplay  
10260 {\MakeUppercase{\glsentryshort{#2}}}{#2}}#3%  
10261 }%
```

Call \@gls@link

```
10262 \gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%  
10263 }%  
  
10264 \glspostlinkhook  
10265 }
```

\@acrlong

```
10266 \def\@acrlong#1#2[#3]{%
```

```

10267 \glsdoifexists{#2}%
10268 {%
10269 \let\do@gls@link@checkfirsthyper\relax

10270 \let\glsifplural\@secondoftwo
10271 \let\glsifcaps\@firstofthree
10272 \let\glsinsert\@empty
10273 \def\glscustomtext{%
10274 \acronymfont{\glslongaccessdisplay{\glsentrylong{#2}}{#2}}#3%
10275 }%

Call \@gls@link
10276 \@gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
10277 }%

10278 \glspostlinkhook
10279 }

```

\@Acrlong

```

10280 \def\@Acrlong#1#2[#3]{%
10281 \glsdoifexists{#2}%
10282 {%
10283 \let\do@gls@link@checkfirsthyper\relax

10284 \let\glsifplural\@secondoftwo
10285 \let\glsifcaps\@firstofthree
10286 \let\glsinsert\@empty
10287 \def\glscustomtext{%
10288 \acronymfont{\glslongaccessdisplay{\Glsentrylong{#2}}{#2}}#3%
10289 }%

Call \@gls@link
10290 \@gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
10291 }%

10292 \glspostlinkhook
10293 }

```

\@ACRlong

```

10294 \def\@ACRlong#1#2[#3]{%
10295 \glsdoifexists{#2}%
10296 {%
10297 \let\do@gls@link@checkfirsthyper\relax

10298 \let\glsifplural\@secondoftwo
10299 \let\glsifcaps\@firstofthree
10300 \let\glsinsert\@empty
10301 \def\glscustomtext{%
10302 \acronymfont{\glslongaccessdisplay{%
10303 \MakeUppercase{\glsentrylong{#2}}}{#2}}#3%
10304 }%

```

Call `\@gls@link`

```
10305 \@gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%  
10306 }%  
  
10307 \glspostlinkhook  
10308 }
```

5.3 Displaying the Glossary

We need to redefine the way the glossary entries are formatted to include the accessibility support. The predefined glossary styles use `\glossentryname`, `\glossentrydesc` and `\glossentrysymbol`, but we need to provide compatibility with earlier versions in case users have defined their own styles using `\accsuppglossaryentryfield` and `\accsuppglossarysubentryfield`.

Now redefine `\glossentryname`, `\glossentrydesc` and `\glossentrysymbol` etc so they use the accessibility stuff.

```
10309 \renewcommand*{\glossentryname}[1]{%  
10310 \glsdoifexists{#1}%  
10311 {%  
10312 \glsnamefont{\glsnameaccessdisplay{\glsentryname{#1}}{#1}}%  
10313 }%  
10314 }  
  
10315 \renewcommand*{\glossentryname}[1]{%  
10316 \glsdoifexists{#1}%  
10317 {%  
10318 \glsnamefont{\glsnameaccessdisplay{\Glsentryname{#1}}{#1}}%  
10319 }%  
10320 }  
  
10321 \renewcommand*{\glossentrydesc}[1]{%  
10322 \glsdoifexists{#1}%  
10323 {%  
10324 \glsdescriptionaccessdisplay{\glsentrydesc{#1}}{#1}%  
10325 }%  
10326 }  
  
10327 \renewcommand*{\Glossentrydesc}[1]{%  
10328 \glsdoifexists{#1}%  
10329 {%  
10330 \glsdescriptionaccessdisplay{\Glsentrydesc{#1}}{#1}%  
10331 }%  
10332 }  
  
10333 \renewcommand*{\glossentrysymbol}[1]{%  
10334 \glsdoifexists{#1}%  
10335 {%  
10336 \glsymbolaccessdisplay{\glsentrysymbol{#1}}{#1}%  
10337 }%  
10338 }
```

```

10339 \renewcommand*{\Glossentrysymbol}[1]{%
10340   \glsdoifexists{#1}%
10341   {%
10342     \glssymbolaccessdisplay{\Glsentrysymbol{#1}}{#1}%
10343   }%
10344 }

```

pglossaryentryfield

```

10345 \newcommand*{\accsuppglossaryentryfield}[5]{%
10346   \glossaryentryfield{#1}%
10347   {\glsnameaccessdisplay{#2}{#1}}%
10348   {\glsdescriptionaccessdisplay{#3}{#1}}%
10349   {\glssymbolaccessdisplay{#4}{#1}}{#5}%
10350 }

```

glossarysubentryfield

```

10351 \newcommand*{\accsuppglossarysubentryfield}[6]{%
10352   \glossarysubentryfield{#1}{#2}%
10353   {\glsnameaccessdisplay{#3}{#2}}%
10354   {\glsdescriptionaccessdisplay{#4}{#2}}%
10355   {\glssymbolaccessdisplay{#5}{#2}}{#6}%
10356 }

```

5.4 Acronyms

Redefine acronym styles provided by glossaries:

long-short *<long>* (*<short>*) acronym style.

```

10357 \renewacronymstyle{long-short}%
10358 {%

```

Check for long form in case this is a mixed glossary.

```

10359   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
10360 }%
10361 {%
10362   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
10363   \renewcommand*{\genacrfullformat}[2]{%
10364     \glslongaccessdisplay{\Glsentrylong{##1}}{##1}##2\space
10365     (\glsshortaccessdisplay
10366       {\protect\firstacronymfont{\Glsentryshort{##1}}}{##1})%
10367   }%
10368   \renewcommand*{\Genacrfullformat}[2]{%
10369     \glslongaccessdisplay{\Glsentrylong{##1}}{##1}##2\space
10370     (\glsshortaccessdisplay
10371       {\protect\firstacronymfont{\Glsentryshort{##1}}}{##1})%
10372   }%
10373   \renewcommand*{\genplacrfullformat}[2]{%
10374     \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}##2\space
10375     (\glsshortpluralaccessdisplay

```

```

10376     {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1})%
10377 }%
10378 \renewcommand*\Genplacrfullformat}[2]{%
10379   \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}##2\space
10380   (\glsshortpluralaccessdisplay
10381     {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1})%
10382 }%
10383 \renewcommand*\acronymentry}[1]{%
10384   \glsshortaccessdisplay{acronymfont{\glsentryshort{##1}}{##1}}
10385 \renewcommand*\acronymsort}[2]{##1}%
10386 \renewcommand*\acronymfont}[1]{##1}%
10387 \renewcommand*\firstacronymfont}[1]{acronymfont{##1}}%
10388 \renewcommand*\acrpluralsuffix{\glspluralsuffix}%
10389 }

```

short-long <short> (<long>) acronym style.

```

10390 \renewacronymstyle{short-long}%
10391 {%

```

Check for long form in case this is a mixed glossary.

```

10392   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
10393 }%
10394 {%
10395   \renewcommand*\GenericAcronymFields{description={\the\glslongtok}}%
10396   \renewcommand*\genacrfullformat}[2]{%
10397     \glsshortaccessdisplay
10398     {\protect\firstacronymfont{\glsentryshort{##1}}{##1}##2\space
10399     (\glslongaccessdisplay{\glsentrylong{##1}}{##1})%
10400   }%
10401   \renewcommand*\Genacrfullformat}[2]{%
10402     \glsshortaccessdisplay
10403     {\protect\firstacronymfont{\Glsentryshort{##1}}{##1}##2\space
10404     (\glslongaccessdisplay{\glsentrylong{##1}}{##1})%
10405   }%
10406   \renewcommand*\genplacrfullformat}[2]{%
10407     \glsshortpluralaccessdisplay
10408     {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}##2\space
10409     (\glslongpluralaccessdisplay
10410       {\glsentrylongpl{##1}}{##1})%
10411   }%
10412   \renewcommand*\Genplacrfullformat}[2]{%
10413     \glsshortpluralaccessdisplay
10414     {\protect\firstacronymfont{\Glsentryshortpl{##1}}{##1}##2\space
10415     (\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1})%
10416   }%
10417   \renewcommand*\acronymentry}[1]{%
10418     \glsshortaccessdisplay{acronymfont{\glsentryshort{##1}}{##1}}%
10419   \renewcommand*\acronymsort}[2]{##1}%
10420   \renewcommand*\acronymfont}[1]{##1}%
10421   \renewcommand*\firstacronymfont}[1]{acronymfont{##1}}%

```

```

10422 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
10423 }

```

long-short-desc *⟨long⟩* (*{⟨short⟩}*) acronym style that has an accompanying description (which the user needs to supply).

```

10424 \renewacronymstyle{long-short-desc}%
10425 {%
10426 \GlsUseAcrEntryDispStyle{long-short}%
10427 }%
10428 {%
10429 \GlsUseAcrStyleDefs{long-short}%
10430 \renewcommand*{\GenericAcronymFields}{}%
10431 \renewcommand*{\acronymsort}[2]{##2}%
10432 \renewcommand*{\acronymentry}[1]{%
10433 \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10434 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1)}}%
10435 }

```

long-sc-short-desc *⟨long⟩* (*\textsc{⟨short⟩}*) acronym style that has an accompanying description (which the user needs to supply).

```

10436 \renewacronymstyle{long-sc-short-desc}%
10437 {%
10438 \GlsUseAcrEntryDispStyle{long-sc-short}%
10439 }%
10440 {%
10441 \GlsUseAcrStyleDefs{long-sc-short}%
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 }

```

long-sm-short-desc *⟨long⟩* (*\textsmaller{⟨short⟩}*) acronym style that has an accompanying description (which the user needs to supply).

```

10448 \renewacronymstyle{long-sm-short-desc}%
10449 {%
10450 \GlsUseAcrEntryDispStyle{long-sm-short}%
10451 }%
10452 {%
10453 \GlsUseAcrStyleDefs{long-sm-short}%
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 }

```

short-long-desc *<short>* (*{<long>}*) acronym style that has an accompanying description (which the user needs to supply).

```
10460 \renewacronymstyle{short-long-desc}%
10461 {%
10462   \GlsUseAcrEntryDispStyle{short-long}%
10463 }%
10464 {%
10465   \GlsUseAcrStyleDefs{short-long}%
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 }
```

sc-short-long-desc *<long>* (`\textsc{<short>}`) acronym style that has an accompanying description (which the user needs to supply).

```
10472 \renewacronymstyle{sc-short-long-desc}%
10473 {%
10474   \GlsUseAcrEntryDispStyle{sc-short-long}%
10475 }%
10476 {%
10477   \GlsUseAcrStyleDefs{sc-short-long}%
10478   \renewcommand*\GenericAcronymFields{}%
10479   \renewcommand*\acronymsort}[2]{##2}%
10480   \renewcommand*\acronymentry}[1]{%
10481     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10482     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10483 }
```

sm-short-long-desc *<long>* (`\textsmaller{<short>}`) acronym style that has an accompanying description (which the user needs to supply).

```
10484 \renewacronymstyle{sm-short-long-desc}%
10485 {%
10486   \GlsUseAcrEntryDispStyle{sm-short-long}%
10487 }%
10488 {%
10489   \GlsUseAcrStyleDefs{sm-short-long}%
10490   \renewcommand*\GenericAcronymFields{}%
10491   \renewcommand*\acronymsort}[2]{##2}%
10492   \renewcommand*\acronymentry}[1]{%
10493     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10494     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10495 }
```

dua *<long>* only acronym style.

```
10496 \renewacronymstyle{dua}%
10497 }
```


Check for long form in case this is a mixed glossary.

```
10498 \ifdefempty\glscustomtext
10499 {%
10500 \ifglshaslong{\glslabel}%
10501 {%
10502 \glsifplural
10503 {%
```

Plural form:

```
10504 \glscapscase
10505 {%
```

Plural form, don't adjust case:

```
10506 \glslongpluralaccessdisplay{\glsentrylongpl{\glslabel}}{\glslabel}%
10507 \glsinsert
10508 }%
10509 {%
```

Plural form, make first letter upper case:

```
10510 \glslongpluralaccessdisplay{\Glsentrylongpl{\glslabel}}{\glslabel}%
10511 \glsinsert
10512 }%
10513 {%
```

Plural form, all caps:

```
10514 \glslongpluralaccessdisplay
10515 {\mfirstucMakeUppercase{\glsentrylongpl{\glslabel}}}{\glslabel}%
10516 \mfirstucMakeUppercase{\glsinsert}%
10517 }%
10518 }%
10519 {%
```

Singular form

```
10520 \glscapscase
10521 {%
```

Singular form, don't adjust case:

```
10522 \glslongaccessdisplay{\glsentrylong{\glslabel}}{\glslabel}\glsinsert
10523 }%
10524 {%
```

Subsequent singular form, make first letter upper case:

```
10525 \glslongaccessdisplay{\Glsentrylong{\glslabel}}{\glslabel}\glsinsert
10526 }%
10527 {%
```

Subsequent singular form, all caps:

```
10528 \glslongaccessdisplay
10529 {\mfirstucMakeUppercase
10530 {\glsentrylong{\glslabel}\glsinsert}}{\glslabel}%
10531 \mfirstucMakeUppercase{\glsinsert}%
10532 }%
```

10533 }%
 10534 }%
 10535 {%

Not an acronym:

10536 \glsgenentryfmt
 10537 }%
 10538 }%
 10539 {\glscustomtext\glsinsert}%
 10540 }%
 10541 {%
 10542 \renewcommand*\GenericAcronymFields{description={\the\glslongtok}}%
 10543 \renewcommand*\acrfullfmt}[3]{%
 10544 \glslink[##1]{##2}{%
 10545 \glslongaccessdisplay{\glsentrylong{##2}}{##2}##3\space
 10546 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2)}}}%
 10547 \renewcommand*\Acrfullfmt}[3]{%
 10548 \glslink[##1]{##2}{%
 10549 \glslongaccessdisplay{\Glsentrylong{##2}}{##2}##3\space
 10550 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2)}}}%
 10551 \renewcommand*\ACRfullfmt}[3]{%
 10552 \glslink[##1]{##2}{%
 10553 \glslongaccessdisplay
 10554 {\mfirstucMakeUppercase{\glsentrylong{##2}}{##2}##3\space
 10555 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2)}}}}}%
 10556 \renewcommand*\acrfullplfmt}[3]{%
 10557 \glslink[##1]{##2}{%
 10558 \glslongpluralaccessdisplay
 10559 {\glsentrylongpl{##2}}{##2}##3\space
 10560 (\glsshortpluralaccessdisplay
 10561 {\acronymfont{\glsentryshortpl{##2}}}{##2)}}}%
 10562 \renewcommand*\Acrfullplfmt}[3]{%
 10563 \glslink[##1]{##2}{%
 10564 \glslongpluralaccessdisplay
 10565 {\Glsentrylongpl{##2}}{##2}##3\space
 10566 (\glsshortpluralaccessdisplay
 10567 {\acronymfont{\glsentryshortpl{##2}}}{##2)}}}}}%
 10568 \renewcommand*\ACRfullplfmt}[3]{%
 10569 \glslink[##1]{##2}{%
 10570 \glslongpluralaccessdisplay
 10571 {\mfirstucMakeUppercase{\glsentrylongpl{##2}}{##2}##3\space
 10572 (\glsshortpluralaccessdisplay
 10573 {\acronymfont{\glsentryshortpl{##2}}}{##2)}}}}}%
 10574 \renewcommand*\glsentryfull}[1]{%
 10575 \glslongaccessdisplay{\glsentrylong{##1}}\space
 10576 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}}%
 10577 }%
 10578 \renewcommand*\Glsentryfull}[1]{%
 10579 \glslongaccessdisplay{\Glsentrylong{##1}}{##1}\space
 10580 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}}%

```

10581 }%
10582 \renewcommand*\glstentryfullpl}[1]{%
10583   \glslongpluralaccessdisplay{\glstentrylongpl{##1}}{##1}\space
10584   (\glsshortpluralaccessdisplay{\acronymfont{\glstentryshortpl{##1}}}{##1})%
10585 }%
10586 \renewcommand*\Glstentryfullpl}[1]{%
10587   \glslongpluralaccessdisplay{\Glstentrylongpl{##1}}{##1}\space
10588   (\glsshortpluralaccessdisplay{\acronymfont{\glstentryshortpl{##1}}}{##1})%
10589 }%
10590 \renewcommand*\acronymentry}[1]{%
10591   \glsshortaccessdisplay{\acronymfont{\glstentryshort{##1}}}{##1}%
10592 \renewcommand*\acronymsort}[2]{##1}%
10593 \renewcommand*\acronymfont}[1]{##1}%
10594 \renewcommand*\acrpluralsuffix{\glspluralsuffix}%
10595 }

```

dua-desc *<long>* only acronym style with user-supplied description.

```

10596 \renewacronymstyle{dua-desc}%
10597 {%
10598   \GlsUseAcrEntryDispStyle{dua}%
10599 }%
10600 {%
10601   \GlsUseAcrStyleDefs{dua}%
10602   \renewcommand*\GenericAcronymFields{}%
10603   \renewcommand*\acronymentry}[1]{%
10604     \glslongaccessdisplay{\acronymfont{\glstentrylong{##1}}}{##1}}%
10605   \renewcommand*\acronymsort}[2]{##2}%
10606 }%

```

footnote *<short>*\footnote{*<long>*} acronym style.

```

10607 \renewacronymstyle{footnote}%
10608 {%

```

Check for long form in case this is a mixed glossary.

```

10609   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
10610 }%
10611 {%
10612   \renewcommand*\GenericAcronymFields{description={\the\glslongtok}}%

```

Need to ensure hyperlinks are switched off on first use:

```

10613   \glshyperfirstfalse
10614   \renewcommand*\genacrfullformat}[2]{%
10615     \glsshortaccessdisplay
10616     {\protect\firstacronymfont{\glstentryshort{##1}}}{##1}##2%
10617     \protect\footnote{\glslongaccessdisplay{\glstentrylong{##1}}{##1}}%
10618   }%
10619   \renewcommand*\Genacrfullformat}[2]{%
10620     \glsshortaccessdisplay
10621     {\firstacronymfont{\Glstentryshort{##1}}}{##1}##2%
10622     \protect\footnote{\glslongaccessdisplay{\glstentrylong{##1}}{##1}}%

```

```

10623 }%
10624 \renewcommand*\genplacrfullformat}[2]{%
10625   \glsshortpluralaccessdisplay
10626     {\protect\firstacronymfont{\glentryshortpl{##1}}{##1}##2%
10627   \protect\footnote{\glslongpluralaccessdisplay{\glentrylongpl{##1}}{##1}}%
10628 }%
10629 \renewcommand*\Genplacrfullformat}[2]{%
10630   \glsshortpluralaccessdisplay
10631     {\protect\firstacronymfont{\Glsentryshortpl{##1}}{##1}##2%
10632   \protect\footnote{\glslongpluralaccessdisplay{\glentrylongpl{##1}}{##1}}%
10633 }%
10634 \renewcommand*\acronymentry}[1]{%
10635   \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}{##1}}%
10636 \renewcommand*\acronymsort}[2]{##1}%
10637 \renewcommand*\acronymfont}[1]{##1}%
10638 \renewcommand*\acrpluralsuffix{\glspluralsuffix}%

```

Don't use footnotes for \acrfull:

```

10639 \renewcommand*\acrfullfnt}[3]{%
10640   \glslink[##1]{##2}{%
10641     \glsshortaccessdisplay{\acronymfont{\glentryshort{##2}}{##2}##3\space
10642     (\glslongaccessdisplay{\glentrylong{##2}}{##2})}}%
10643 \renewcommand*\Acrfullfnt}[3]{%
10644   \glslink[##1]{##2}{%
10645     \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##2}}{##2}##3\space
10646     (\glslongaccessdisplay{\glentrylong{##2}}{##2})}}%
10647 \renewcommand*\ACRfullfnt}[3]{%
10648   \glslink[##1]{##2}{%
10649     \glsshortaccessdisplay
10650     {\mfirstucMakeUppercase
10651     {\acronymfont{\glentryshort{##2}}{##2}##3\space
10652     (\glslongaccessdisplay{\glentrylong{##2}}{##2})}}}}%
10653 \renewcommand*\acrfullplfmt}[3]{%
10654   \glslink[##1]{##2}{%
10655     \glsshortpluralaccessdisplay
10656     {\acronymfont{\glentryshortpl{##2}}{##2}##3\space
10657     (\glslongpluralaccessdisplay{\glentrylongpl{##2}}{##2})}}%
10658 \renewcommand*\Acrfullplfmt}[3]{%
10659   \glslink[##1]{##2}{%
10660     \glsshortpluralaccessdisplay
10661     {\acronymfont{\Glsentryshortpl{##2}}{##2}##3\space
10662     (\glslongpluralaccessdisplay{\glentrylongpl{##2}}{##2})}}%
10663 \renewcommand*\ACRfullplfmt}[3]{%
10664   \glslink[##1]{##2}{%
10665     \glsshortpluralaccessdisplay
10666     {\mfirstucMakeUppercase
10667     {\acronymfont{\glentryshortpl{##2}}{##2}##3\space
10668     (\glslongpluralaccessdisplay{\glentrylongpl{##2}}{##2})}}}}%

```

Similarly for \glentryfull etc:

```

10669 \renewcommand*\glentryfull}[1]{%
10670   \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}{##1}\space
10671   (\glslongaccessdisplay{\glentrylong{##1}}{##1})}%
10672 \renewcommand*\Glsentryfull}[1]{%
10673   \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##1}}{##1}\space
10674   (\glslongaccessdisplay{\glentrylong{##1}}{##1})}%
10675 \renewcommand*\glentryfullpl}[1]{%
10676   \glsshortpluralaccessdisplay
10677   {\acronymfont{\glentryshortpl{##1}}{##1}\space
10678   (\glslongpluralaccessdisplay{\glentrylongpl{##1}}{##1})}%
10679 \renewcommand*\Glsentryfullpl}[1]{%
10680   \glsshortpluralaccessdisplay
10681   {\acronymfont{\Glsentryshortpl{##1}}{##1}\space
10682   (\glslongpluralaccessdisplay{\glentrylongpl{##1}}{##1})}%
10683 }

```

footnote-sc \textsc{<short>} \footnote{<long>} acronym style.

```

10684 \renewacronymstyle{footnote-sc}%
10685 {%
10686   \GlsUseAcrEntryDispStyle{footnote}%
10687 }%
10688 {%
10689   \GlsUseAcrStyleDefs{footnote}%
10690   \renewcommand{\acronymentry}[1]{%
10691     \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}{##1}}
10692   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
10693   \renewcommand*\acrpluralsuffix{\glstextup{\glspluralsuffix}}%
10694 }%

```

footnote-sm \textsmaller{<short>} \footnote{<long>} acronym style.

```

10695 \renewacronymstyle{footnote-sm}%
10696 {%
10697   \GlsUseAcrEntryDispStyle{footnote}%
10698 }%
10699 {%
10700   \GlsUseAcrStyleDefs{footnote}%
10701   \renewcommand{\acronymentry}[1]{%
10702     \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}{##1}}
10703   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
10704   \renewcommand*\acrpluralsuffix{\glspluralsuffix}%
10705 }%

```

footnote-desc <short> \footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

10706 \renewacronymstyle{footnote-desc}%
10707 {%
10708   \GlsUseAcrEntryDispStyle{footnote}%
10709 }%
10710 {%

```

```

10711 \GlsUseAcrStyleDefs{footnote}%
10712 \renewcommand*\GenericAcronymFields{}%
10713 \renewcommand*\acronymsort}[2]{##2}%
10714 \renewcommand*\acronymentry}[1]{%
10715     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10716     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10717 }

```

footnote-sc-desc \textsc{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

10718 \renewacronymstyle{footnote-sc-desc}%
10719 {%
10720     \GlsUseAcrEntryDispStyle{footnote-sc}%
10721 }%
10722 {%
10723     \GlsUseAcrStyleDefs{footnote-sc}%
10724     \renewcommand*\GenericAcronymFields{}%
10725     \renewcommand*\acronymsort}[2]{##2}%
10726     \renewcommand*\acronymentry}[1]{%
10727         \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10728         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10729 }

```

footnote-sm-desc \textsmaller{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

10730 \renewacronymstyle{footnote-sm-desc}%
10731 {%
10732     \GlsUseAcrEntryDispStyle{footnote-sm}%
10733 }%
10734 {%
10735     \GlsUseAcrStyleDefs{footnote-sm}%
10736     \renewcommand*\GenericAcronymFields{}%
10737     \renewcommand*\acronymsort}[2]{##2}%
10738     \renewcommand*\acronymentry}[1]{%
10739         \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10740         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10741 }

```

Use \newacronymhook to modify the key list to set the access text to the long version by default.

```

10742 \renewcommand*\newacronymhook{%
10743     \edef\@gls@keylist{shortaccess=\the\glslongtok,%
10744         \the\glskeylisttok}%
10745     \expandafter\glskeylisttok\expandafter{\@gls@keylist}%
10746 }

```

defaultNewAcronymDef Modify default style to use access text:

```

10747 \renewcommand*\DefaultNewAcronymDef{%

```

```

10748 \edef\@do@newglossaryentry{%
10749   \noexpand\newglossaryentry{\the\glslabeltok}%
10750   {%
10751     type=\acronymtype,%
10752     name={\the\glsshorttok},%
10753     description={\the\glslongtok},%
10754     descriptionaccess=\relax,
10755     text={\the\glsshorttok},%
10756     access={\noexpand\@glo@textaccess},%
10757     sort={\the\glsshorttok},%
10758     short={\the\glsshorttok},%
10759     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10760     shortaccess={\the\glslongtok},%
10761     long={\the\glslongtok},%
10762     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10763     descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10764     first={\noexpand\glslongaccessdisplay
10765       {\the\glslongtok}{\the\glslabeltok}\space
10766       (\noexpand\glsshortaccessdisplay
10767         {\the\glsshorttok}{\the\glslabeltok})},%
10768     plural={\the\glsshorttok\acrpluralsuffix},%
10769     firstplural={\noexpand\glslongpluralaccessdisplay
10770       {\noexpand\@glo@longpl}{\the\glslabeltok}\space
10771       (\noexpand\glsshortpluralaccessdisplay
10772         {\noexpand\@glo@shortpl}{\the\glslabeltok})},%
10773     firstaccess=\relax,
10774     firstpluralaccess=\relax,
10775     textaccess={\noexpand\@glo@shortaccess},%
10776     \the\glskeylisttok
10777   }%
10778 }%
10779 \let\@org@gls@assign@firstpl\gls@assign@firstpl
10780 \let\@org@gls@assign@plural\gls@assign@plural
10781 \let\@org@gls@assign@descplural\gls@assign@descplural
10782 \def\gls@assign@firstpl##1##2{%
10783   \@@gls@expand@field{##1}{firstpl}{##2}%
10784 }%
10785 \def\gls@assign@plural##1##2{%
10786   \@@gls@expand@field{##1}{plural}{##2}%
10787 }%
10788 \def\gls@assign@descplural##1##2{%
10789   \@@gls@expand@field{##1}{descplural}{##2}%
10790 }%
10791 \@do@newglossaryentry
10792 \let\gls@assign@firstpl\@org@gls@assign@firstpl
10793 \let\gls@assign@plural\@org@gls@assign@plural
10794 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
10795 }

```

otnoteNewAcronymDef

```
10796 \renewcommand*{\DescriptionFootnoteNewAcronymDef}{%
10797   \edef\@do@newglossaryentry{%
10798     \noexpand\newglossaryentry{\the\glslabeltok}%
10799     {%
10800       type=\acronymtype,%
10801       name={\noexpand\acronymfont{\the\glsshorttok}},%
10802       sort={\the\glsshorttok},%
10803       text={\the\glsshorttok},%
10804       short={\the\glsshorttok},%
10805       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10806       shortaccess={\the\glslongtok},%
10807       long={\the\glslongtok},%
10808       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10809       access={\noexpand\@glo@textaccess},%
10810       plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10811       symbol={\the\glslongtok},%
10812       symbolplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10813       firstpluralaccess=\relax,
10814       textaccess={\noexpand\@glo@shortaccess},%
10815       \the\glskeylisttok
10816     }%
10817   }%
10818   \let\@org@gls@assign@firstpl\gls@assign@firstpl
10819   \let\@org@gls@assign@plural\gls@assign@plural
10820   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
10821   \def\gls@assign@firstpl##1##2{%
10822     \@@gls@expand@field{##1}{firstpl}{##2}%
10823   }%
10824   \def\gls@assign@plural##1##2{%
10825     \@@gls@expand@field{##1}{plural}{##2}%
10826   }%
10827   \def\gls@assign@symbolplural##1##2{%
10828     \@@gls@expand@field{##1}{symbolplural}{##2}%
10829   }%
10830   \@do@newglossaryentry
10831   \let\gls@assign@plural\@org@gls@assign@plural
10832   \let\gls@assign@firstpl\@org@gls@assign@firstpl
10833   \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
10834 }
```

iptionNewAcronymDef

```
10835 \renewcommand*{\DescriptionNewAcronymDef}{%
10836   \edef\@do@newglossaryentry{%
10837     \noexpand\newglossaryentry{\the\glslabeltok}%
10838     {%
10839       type=\acronymtype,%
10840       name={\noexpand
10841         \acrnameformat{\the\glsshorttok}{\the\glslongtok}},%
```



```

10842     access={\noexpand\@glo@textaccess},%
10843     sort={\the\glsshorttok},%
10844     short={\the\glsshorttok},%
10845     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10846     shortaccess={\the\glslongtok},%
10847     long={\the\glslongtok},%
10848     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10849     first={\the\glslongtok},%
10850     firstaccess=\relax,
10851     firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10852     text={\the\glsshorttok},%
10853     textaccess={\the\glslongtok},%
10854     plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10855     symbol={\noexpand\@glo@text},%
10856     symbolaccess={\noexpand\@glo@textaccess},%
10857     symbolplural={\noexpand\@glo@plural},%
10858     firstpluralaccess=\relax,
10859     textaccess={\noexpand\@glo@shortaccess},%
10860     \the\glskeylisttok}%
10861 }%
10862 \let\@org@gls@assign@firstpl\gls@assign@firstpl
10863 \let\@org@gls@assign@plural\gls@assign@plural
10864 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
10865 \def\gls@assign@firstpl##1##2{%
10866   \@@gls@expand@field{##1}{firstpl}{##2}%
10867 }%
10868 \def\gls@assign@plural##1##2{%
10869   \@@gls@expand@field{##1}{plural}{##2}%
10870 }%
10871 \def\gls@assign@symbolplural##1##2{%
10872   \@@gls@expand@field{##1}{symbolplural}{##2}%
10873 }%
10874 \do@newglossaryentry
10875 \let\gls@assign@firstpl\@org@gls@assign@firstpl
10876 \let\gls@assign@plural\@org@gls@assign@plural
10877 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
10878 }

```

otnoteNewAcronymDef

```

10879 \renewcommand*{\FootnoteNewAcronymDef}{%
10880   \edef\@do@newglossaryentry{%
10881     \noexpand\newglossaryentry{\the\glslabeltok}%
10882     {%
10883       type=\acronymtype,%
10884       name={\noexpand\acronymfont{\the\glsshorttok}},%
10885       sort={\the\glsshorttok},%
10886       text={\the\glsshorttok},%
10887       textaccess={\the\glslongtok},%
10888       access={\noexpand\@glo@textaccess},%

```

```

10889 plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10890 short={\the\glsshorttok},%
10891 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10892 long={\the\glslongtok},%
10893 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10894 description={\the\glslongtok},%
10895 descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10896 \the\glskeylisttok
10897 }%
10898 }%
10899 \let\org@gls@assign@plural\gls@assign@plural
10900 \let\org@gls@assign@firstpl\gls@assign@firstpl
10901 \let\org@gls@assign@descplural\gls@assign@descplural
10902 \def\gls@assign@firstpl##1##2{%
10903 \@@gls@expand@field{##1}{firstpl}{##2}%
10904 }%
10905 \def\gls@assign@plural##1##2{%
10906 \@@gls@expand@field{##1}{plural}{##2}%
10907 }%
10908 \def\gls@assign@descplural##1##2{%
10909 \@@gls@expand@field{##1}{descplural}{##2}%
10910 }%
10911 \do@newglossaryentry
10912 \let\gls@assign@plural\org@gls@assign@plural
10913 \let\gls@assign@firstpl\org@gls@assign@firstpl
10914 \let\gls@assign@descplural\org@gls@assign@descplural
10915 }

```

\SmallNewAcronymDef

```

10916 \renewcommand*{\SmallNewAcronymDef}{%
10917 \edef\do@newglossaryentry{%
10918 \noexpand\newglossaryentry{\the\glslabeltok}%
10919 {%
10920 type=\acronymtype,%
10921 name={\noexpand\acronymfont{\the\glsshorttok}},%
10922 access={\noexpand\@glo@symbolaccess},%
10923 sort={\the\glsshorttok},%
10924 short={\the\glsshorttok},%
10925 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10926 shortaccess={\the\glslongtok},%
10927 long={\the\glslongtok},%
10928 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10929 text={\noexpand\@glo@short},%
10930 textaccess={\noexpand\@glo@shortaccess},%
10931 plural={\noexpand\@glo@shortpl},%
10932 first={\the\glslongtok},%
10933 firstaccess=\relax,
10934 firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10935 description={\noexpand\@glo@first},%

```

```

10936     descriptionplural={\noexpand\@glo@firstplural},%
10937     symbol={\the\glsshorttok},%
10938     symbolaccess={\the\glslongtok},%
10939     symbolplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10940     \the\glskeylisttok
10941   }%
10942 }%
10943 \let\@org@gls@assign@firstpl\gls@assign@firstpl
10944 \let\@org@gls@assign@plural\gls@assign@plural
10945 \let\@org@gls@assign@descplural\gls@assign@descplural
10946 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
10947 \def\gls@assign@firstpl##1##2{%
10948   \@@gls@expand@field{##1}{firstpl}{##2}%
10949 }%
10950 \def\gls@assign@plural##1##2{%
10951   \@@gls@expand@field{##1}{plural}{##2}%
10952 }%
10953 \def\gls@assign@descplural##1##2{%
10954   \@@gls@expand@field{##1}{descplural}{##2}%
10955 }%
10956 \def\gls@assign@symbolplural##1##2{%
10957   \@@gls@expand@field{##1}{symbolplural}{##2}%
10958 }%
10959 \do@newglossaryentry
10960 \let\gls@assign@firstpl\@org@gls@assign@firstpl
10961 \let\gls@assign@plural\@org@gls@assign@plural
10962 \let\gls@assign@descplural\@org@gls@assign@descplural
10963 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
10964 }

```

The following are kept for compatibility with versions before 3.0:

```

\glsshortaccesskey
10965 \newcommand*{\glsshortaccesskey}{\glsshortkey access}%

\glsshortpluralaccesskey
10966 \newcommand*{\glsshortpluralaccesskey}{\glsshortpluralkey access}%

\glslongaccesskey
10967 \newcommand*{\glslongaccesskey}{\glslongkey access}%

\glslongpluralaccesskey
10968 \newcommand*{\glslongpluralaccesskey}{\glslongpluralkey access}%

```

5.5 Debugging Commands

```

\showglongnameaccess
10969 \newcommand*{\showglongnameaccess}[1]{%
10970   \expandafter\show\curname glo@\glsdetoklabel{##1}@textaccess\endcsname
10971 }

```

```

\showglotextaccess
10972 \newcommand*{\showglotextaccess}[1]{%
10973   \expandafter\show\csname glo@\glsdetoklabel{#1}@textaccess\endcsname
10974 }

showglopluralaccess
10975 \newcommand*{\showglopluralaccess}[1]{%
10976   \expandafter\show\csname glo@\glsdetoklabel{#1}@pluralaccess\endcsname
10977 }

\showglofirstaccess
10978 \newcommand*{\showglofirstaccess}[1]{%
10979   \expandafter\show\csname glo@\glsdetoklabel{#1}@firstaccess\endcsname
10980 }

lofirstpluralaccess
10981 \newcommand*{\showglofirstpluralaccess}[1]{%
10982   \expandafter\show\csname glo@\glsdetoklabel{#1}@firstpluralaccess\endcsname
10983 }

showglosymbolaccess
10984 \newcommand*{\showglosymbolaccess}[1]{%
10985   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolaccess\endcsname
10986 }

osymbolpluralaccess
10987 \newcommand*{\showglosymbolpluralaccess}[1]{%
10988   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolpluralaccess\endcsname
10989 }

\showglodescaccess
10990 \newcommand*{\showglodescaccess}[1]{%
10991   \expandafter\show\csname glo@\glsdetoklabel{#1}@descaccess\endcsname
10992 }

glodescpluralaccess
10993 \newcommand*{\showglodescpluralaccess}[1]{%
10994   \expandafter\show\csname glo@\glsdetoklabel{#1}@descpluralaccess\endcsname
10995 }

\showgloshortaccess
10996 \newcommand*{\showgloshortaccess}[1]{%
10997   \expandafter\show\csname glo@\glsdetoklabel{#1}@shortaccess\endcsname
10998 }

loshortpluralaccess
10999 \newcommand*{\showgloshortpluralaccess}[1]{%
11000   \expandafter\show\csname glo@\glsdetoklabel{#1}@shortpluralaccess\endcsname
11001 }

```

`\showglolongaccess`

```
11002 \newcommand*{\showglolongaccess}[1]{%
11003   \expandafter\show\csname glo@\glsdetoklabel{#1}@longaccess\endcsname
11004 }
```

`glolongpluralaccess`

```
11005 \newcommand*{\showglolongpluralaccess}[1]{%
11006   \expandafter\show\csname glo@\glsdetoklabel{#1}@longpluralaccess\endcsname
11007 }
```

6 Multi-Lingual Support

Many thanks to everyone who contributed to the translations both via email and on `comp.text.tex`. Language support has now been split off into independent language modules.

```
11008 \NeedsTeXFormat{LaTeX2e}
11009 \ProvidesPackage{glossaries-babel}[2015/09/09 v4.18 (NLCT)]
```

Load `tracklang` to obtain language settings.

```
11010 \RequirePackage{tracklang}
11011 \let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
11012 \AnyTrackedLanguages
11013 {%
11014   \ForEachTrackedDialect{\this@dialect}{%
11015     \IfTrackedLanguageFileExists{\this@dialect}%
11016     {glossaries-}% prefix
11017     {.ldf}%
11018     {%
11019       \RequireGlossariesLang{\CurrentTrackedTag}%
11020     }%
11021   }%
11022   \PackageWarningNoLine{glossaries}%
11023   {No language module detected for ‘\this@dialect’.\MessageBreak
11024     Language modules need to be installed separately.\MessageBreak
11025     Please check on CTAN for a bundle called\MessageBreak
11026     ‘glossaries-\CurrentTrackedLanguage’ or similar}%
11027 }%
11028 }%
11029 }%
11030 }
```

6.1 Polyglossia Captions

Language support has now been split off into independent language modules.

```
11031 \NeedsTeXFormat{LaTeX2e}
11032 \ProvidesPackage{glossaries-polyglossia}[2015/09/09 v4.18 (NLCT)]
```

Load tracklang to obtain language settings.

```
11033 \RequirePackage{tracklang}
11034 \let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
11035 \AnyTrackedLanguages
11036 {%
11037   \ForEachTrackedDialect{\this@dialect}{%
11038     \IfTrackedLanguageFileExists{\this@dialect}%
11039     {glossaries-}% prefix
11040     {.ldf}%
11041     {%
11042       \RequireGlossariesLang{\CurrentTrackedTag}%
11043     }%
11044     {%
11045       \PackageWarningNoLine{glossaries}%
11046       {No language module detected for ‘\this@dialect’.\MessageBreak
11047       Language modules need to be installed separately.\MessageBreak
11048       Please check on CTAN for a bundle called\MessageBreak
11049       ‘glossaries-\CurrentTrackedLanguage’ or similar}%
11050     }%
11051   }%
11052 }%
11053 }
```

Glossary

`makeindex` An indexing application. [10](#), [25](#), [26](#), [170](#)

`xindy` An flexible indexing application with multilingual support written in Perl. [10](#), [25](#), [26](#), [170](#)

Change History

??		1.05	
	super: fixed typo in <code>\subglossentry</code>		<code>\glossarysection:</code> added
	(<code>\glossentrydesc</code>) 280		<code>\@mkboth</code> to <code>\glossarysection</code>
1.01		 37
	General: Added range facility in		<code>\gls@defglossaryentry:</code>
	format key 108		Changed the default value of
	<code>\writeist:</code> Added spaces after		the sort key to just the value of
	<code>\delimN</code> and <code>\delimR</code> in <code>ist</code>		the name key 76
	file 156	1.07	
1.04			<code>\@gls@link:</code> fixed bug caused by
	General: Added <code>\glstextformat</code> 92		<code>\theglstrycounter</code> set-

ting the page number too soon	106		
\glsadd: fixed bug caused by \theglsentrycounter set- ting the page number too soon	153		
1.08			
General: Added babel support ...	31		
listgroup: changed listgroup style to use \glsgetgrouptitle	262		
atlistgroup: changed al- tlistgroup style to use \glsgetgrouptitle	263		
1.1			
\@glossarysection: numbered sections and auto label added	38		
\@gls@tmpb: changed \toksdef to \newtoks	110		
\@gls@toc: numberline added ..	40		
\@p@glossarysection: num- bered sections and auto label added	39		
General: amsgen now loaded (\new@ifnextchar needed) ..	4		
translate: translate option added	22		
\setglossarysection: new ...	38		
numberedsection: numbered- section package option added .	6		
numberline: numberline option added	5		
1.12			
\@GLSpl: now uses \glsentrydescplural and \glsentrysymbolplural instead of \glsentrydesc and \glsentrysymbol	122		
\@Glspl@: now uses \glsentrydescplural and \glsentrysymbolplural instead of \glsentrydesc and \glsentrysymbol	121		
\@glspl@: now uses \glsentrydescplural and \glsentrysymbolplural instead of \glsentrydesc and \glsentrysymbol	120		
General: added check for \hypertarget separate to \hyperlink (memoir de- fines \hyperlink but not			
\hypertarget)	116		
descriptionplural: new	59		
\gls@defglossaryentry: Changed default first plural to be first key with s appended (was text key with s appended)	76		
descriptionplural support added	76		
symbolplural support added ..	76		
\Glsentrydescplural: New ..	147		
\glsentrydescplural: New ..	147		
\Glsentrysymbolplural: New	148		
\glsentrysymbolplural: New	148		
\SetDescriptionFootnoteAcronymStyle: Added \protect before \footnote and \glslink .	230		
\SetFootnoteAcronymStyle: Added \protect before \footnote and \glslink .	236		
symbolplural: new	60		
1.13			
General: fixed bug that ignored 3rd parameter	124–131		
\ACRfullpl: new	211		
\Acrfullpl: new	210		
\acrfullpl: new	210		
\acrpluralsuffix: New	208		
\gls@defglossaryentry: Changed default first value ..	76		
Changed default firstplural value	76		
Removed restriction on only using \newglossaryentry in the preamble	81		
\newacronym: Removed re- striction on only using \newacronym in the preamble	208		
\@gls@hypergroup: new	257		
General: added nonnumberlist key to \printglossary	194		
added numberedsection key to \printglossary	192		
\firstacronymfont: new	211		
\glsautoprefix: new	6		
\glsnavhyperlink: changed 'edef to 'protected@edef ...	257		
\glsnavhypertarget: added write to aux file	257		

\glsnavigation: changed to only use labels for groups that are present	258	\gls@defglossaryentry: Changed def to let	76
1.15		1.17	
\@gls@link: added \glslabel	106	\@do@wrglossary: new	173
\gls@defglossaryentry: check for \@glo@first in description	80	\@do@seeglossary: new	176
check for \@glo@text in symbol	80	\@glo@storeentry: new	82
\gls@hypergroup: new	257	\@gls@glossary: changed definition to use \index instead of \@index	171
\glsnavhypertarget: added check if rerun required	257	\@glsdefaultplural: new	63
\glssettoctitle: new	30	\@glsdefaultsort: new	63
\printglossary: changed the way the TOC title is set	178	\@gls@hypernumber: new	205
1.16		\@glsnoname: new	63
\@GLS@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	120	\@glsnonextpages: new	194
\@GLSpl: Test glossary type is \acronymtype in addition to checking if footnote option has been used	122	General: added xindy support ...	25
\@GLs@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	119	parent: new	61
\@GLspl@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	121	see: new	61
\@gls@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	118	\gls@defglossaryentry: added nonumberlist key	76
\@glsdisp: Test glossary type is \acronymtype in addition to checking if footnote option has been used	123	added parent key	76
\@glspl@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	120	added see key	76
\@glstarget: raised the hypertarget so the target text doesn't scroll off the top of the page	116	Stored main part of entry format when entry is defined	81
		\gls@suffixF: new	35
		\gls@suffixFF: new	35
		\gls@wrglossary: modified to allow for xindy support	171
		\gls@hyperlink: new	153
		\gls@hypernumber: modified to allow material to be attached to location	204
		\glsnavhyperlink: replaced 'hyperlink' to '@glslink	257
		\glsnavhypertarget: replaced 'hypertarget' to '@glstarget ..	257
		\glssee: new	177
		\glsseeformat: new	177
		\glsSetSuffixF: new	35
		\glsSetSuffixFF: new	35
		\ifglsxindy: new	25
		\istfilename: added xindy support	34
		\newglossarystyle: made \newglossarystyle long ..	203
		\nopostdesc: new	33
		nonumberlist: new	61
		\printglossary: added check to determine if \printglossary is already defined	178

added print language to aux file	178	\forlsglentries:	replaced
order: order package option		\ifthenelse with \ifx 50
added	24	\glsdefmain: new 12
\writeist: added xindy support	156	\glsdescwidth:	changed
1.18		\linewidth to \hsize	. 264, 280
\@gls@loadlist: new	8	\glslistdottedwidth:	changed
\@gls@loadlong: new	8	\linewidth to \hsize 264
\@gls@loadsuper: new	8	\glspagelistwidth:	changed
\@gls@loadtree: new	8	\linewidth to \hsize	. 265, 280
\gls@defglossaryentry:		nomain: added nomain package	
Changed default value of sort		option 13
to \@gls@defaultsort	76	\writeist: removed item_02 - no	
moved sort sanitization to		such makeindex key 160
\newglossaryentry	80	2.02	
\glstarget: new	198	\@printglossary:	suppressed
\oldacronym: new	207	warning globally rather than	
nolist: new	8	locally 181
nolong: new	8	\glossarysection:	changed
sort: moved sanitization to		\@mkboth to \glossarymark	37
\newglossaryentry	59	\gls glossarymark: New 37
nostyles: new	8	2.03	
nosuper: new	8	\@GLS@:	Added check for hyper-
notree: new	8	first 120
1.19		\@GLSp1:	Added check for hyper-
\gls clearpage: new	40	first 122
\glsdisp: new	122	\@Gls@:	Added check for hyper-
\SetDescriptionAcronymStyle:		first 119
changed \acronymfont to		\@Glspl@:	Added check for hyper-
use \textsmaller instead of		first 121
\smaller	234	\@gls@:	Added check for hyper-
\SetDescriptionFootnoteAcronymStyle:		first 118
changed \acronymfont to		\@gls@@link:	new
use \textsmaller instead of		\@gls@link:	added \leavevmode
\smaller	230	106
\SetFootnoteAcronymStyle:		Moved entry existence check to	
changed \acronymfont to		avoid duplicate code 106
use \textsmaller instead of		\@glsdisp:	Added check for hyper-
\smaller	236	perfirst 123
\SetSmallAcronymStyle:		\@glspl@:	Added check for hyper-
changed \acronymfont to		first 120
use \textsmaller instead of		\gls glossarymark:	Added check
\smaller	239	to see if it's already defined	.. 37
2.01		hyperfirst: new 23
\@gls@link: moved \@do@wrglossary	2.04	\@GLS@:	Changed test to check if
before term is displayed to pre-		glossary type has been identi-	
vent unwanted whatsit	107	fied as a list of acronyms	... 120
\forall glossaries:	replaced	\@GLSp1:	Changed test to check if
\ifthenelse with \ifx	49		

glossary type has been identified as a list of acronyms ...	122	\SetAcronymLists:new	15
\@Gls@: Changed test to check if glossary type has been identified as a list of acronyms ...	119	\SetDefaultAcronymDisplayStyle:new	226
\@Glspl@: Changed test to check if glossary type has been identified as a list of acronyms ..	121	\SetDefaultAcronymStyle:new	227
\@glossaryentryfield:new ..	82	\SetDescriptionAcronymDisplayStyle:new	232
\@glossarysubentryfield:new	82	\SetDescriptionDUAAcronymDisplayStyle:new	230
\@Gls@: Changed test to check if glossary type has been identified as a list of acronyms ...	118	\SetDescriptionFootnoteAcronymDisplayStyle:new	228
\@Glsacronymlists:new	14	\SetDUADisplayStyle:new ..	240
\@Glsdisp: Changed test to check if glossary type has been identified as a list of acronyms ..	123	\SetFootnoteAcronymDisplayStyle:new	234
\@Glspl@: Changed test to check if glossary type has been identified as a list of acronyms ..	120	\SetSmallAcronymDisplayStyle:new	237
\@newglossaryentryposthook:new	81	2.05	
\@newglossaryentryprehook:new	81	\@Glsdisp: Added closing brace. Patch provided by Sergiu Dotenco	123
acronymlists:new	15	Removed spurious brace. Patch provided by Sergiu Dotenco	123
\DeclareAcronymList:new ...	14	\writeist: Added \string before opening and closing braces. Patch provided by Segiu Dotenco	160
\DefineAcronymSynonyms:new	224	2.06	
\Gls@defglossaryentry: added user1-6 keys	76	\altnewglossary:new	57
\Glsadd: fixed bug that ignored counter	153	\CustomAcronymFields:new .	242
\Glsentryuseri:new	149	\CustomNewAcronymDef:new .	242
\Glsentryuserii:new	149	\SetCustomDisplayStyle:new	242
\Glsentryuseriii:new	149	\SetCustomStyle:new	243
\Glsentryuseriiii:new	149	2.07	
\Glsentryuseriv:new	150	General: glssadd format key stored in \@Glsnumberformat (was mistakenly stored in \@glo@format)	153
\Glsentryuseriv:new	149	3.0	
\Glsentryuseriv:new	150	\@do@wrglossary: added check for hyper location prefix ...	174
\Glsentryuseriv:new	150	modified to use new format ..	173
\Glsentryuseriv:new	150	\@glossarysec: replaced \@ifundefined with \@ifcsundef	5
\Glsentryuseriv:new	150	\@do@seeglossary: Sanitize and escape cross-referencing information	176
\ns@newglossary: added check to determine if \@Gls@<type>@display and \@Gls@<type>@displayfirst have been defined.	57	\@Gls@counterwithin:new	9
		\@Gls@ifinlist:new	41

<code>\@gls@link</code> : added <code>\@gls@saveentrycounter</code>	<code>\gls@crossarystyle</code> : replaced
..... 107	<code>\@ifundefined</code> with
added <code>\@gls@setsort</code> 107	<code>\ifcsundef</code> 203
<code>\@gls@saveentrycounter</code> : new 107	<code>\gls@codepage</code> : replaced
<code>\@gls@setupsort@def</code> : new ... 11	<code>\@ifundefined</code> with
<code>\@gls@setupsort@standard</code> :	<code>\ifcsundef</code> 25
new 10	<code>\gls@defglossaryentry</code> : added
<code>\@gls@setupsort@use</code> : new ... 11	<code>\@gls@defsort</code> 80
<code>\@gls@xdy@locationlist</code> : new 44	added short and long keys 76
<code>\@glslink</code> : replaced <code>\@ifundefined</code>	replaced <code>\@ifundefined</code> with
with <code>\ifcsundef</code> 116	<code>\ifcsundef</code> 77
<code>\@glsnextpages</code> : new 195	<code>\gls@docclearpage</code> : replaced
<code>\@makeglossary</code> : Added check	<code>\@ifundefined</code> with
for savewrites 162	<code>\ifcsundef</code> 39
<code>\@print@glossary</code> : replaced	<code>\gls@wrglossary</code> : modified to
<code>\@ifundefined</code> with	take into account savewrites 171
<code>\ifcsundef</code> 181	<code>\glsadd</code> : added <code>\@gls@saveentrycounter</code>
<code>\@printglossary</code> : added 153
<code>\currentglossary</code> 180	<code>\GlsAddXdyCounters</code> : new 41
added <code>\glsnextpages</code> 180	<code>\glsentrycounterlabel</code> : new 197
make toctitle default to title .. 180	<code>\glsentryitem</code> : new 197
<code>\@set@glo@numformat</code> : added	<code>\Glsentrylong</code> : new 151
4th argument 108	<code>\glsentrylong</code> : new 150
<code>\@xdy@attributelist</code> : new 40	<code>\Glsentrylongpl</code> : new 151
General: added prefix to hyperlink	<code>\glsentrylongpl</code> : new 151
..... 206	<code>\Glsentryshort</code> : new 150
etoolbox now loaded 4	<code>\glsentryshort</code> : new 150
replaced <code>\@ifundefined</code> with	<code>\Glsentryshortpl</code> : new 150
<code>\ifcsundef</code> ... 29, 32, 103, 192	<code>\glsentryshortpl</code> : new 150
<code>\acrfootnote</code> : new 228	<code>\glsgetgrouptitle</code> : re-
<code>\ACRfull</code> : added starred version 210	placed <code>\@ifundefined</code> with
<code>\Acrfull</code> : added starred version 209	<code>\ifcsundef</code> 201
<code>\acrfull</code> : added starred version 208	<code>\gls@glossarymark</code> : replaced
<code>\ACRfullpl</code> : added starred ver-	<code>\@ifundefined</code> with
sion 211	<code>\ifcsundef</code> 37
<code>\Acrfullpl</code> : added starred ver-	<code>\gls@hyperlink</code> : changed de-
sion 210	fault from <code>\glsentryname</code> to
<code>\acrfullpl</code> : added starred ver-	<code>\glsentrytext</code> 153
sion 210	<code>\gls@hypernumber</code> : replaced
<code>\acrlinkfootnote</code> : new 228	<code>\@ifundefined</code> with
<code>\acrno linkfootnote</code> : new ... 228	<code>\ifcsundef</code> 204
savewrites: new 26	<code>\gls@numberformat</code> : replaced
see: added <code>\@glo@seeautonumberlist</code>	<code>\@ifundefined</code> with
..... 61	<code>\ifcsundef</code> 36
seeautonumberlist: new 8	<code>\gls@refentry</code> : new 196
<code>\glossarysection</code> : replaced	<code>\gls@resetsubentrycounter</code> :
<code>\@ifundefined</code> with	new 196
<code>\ifcsundef</code> 37	<code>\gls@seeitem</code> : hyperlink uses
	<code>\gls@seeitemformat</code> instead

of\glstentryname	178	\setentrycounter: added op-	
\glstseeitemformat:new	178	tional argument	202
\glstsortnumberfmt:new	11	\showacronymlists:new	248
\glststepentry:new	196	\showglocounter:new	245
\glststepsubentry:new	196	\showglodesc:new	246
\glstsubentrycounterlabel:		\showglodescplural:new ...	247
new	197	\showglofirst:new	245
\glstsubentryitem:new	197	\showglofirstpl:new	245
theglossary:replaced\@ifundefined		\showgloflag:new	248
with\ifcsundef	197	\showgloindex:new	248
short:new	62	\showglolevel:new	244
shortplural:new	62	\showglongame:new	246
\ifglossaryexists: replaced		\showgloparent:new	244
\@ifundefined with		\showgloplural:new	244
\ifcsundef	50	\showglosort:new	247
\ifglstentryexists: replaced		\showglossaries:new	248
\@ifundefined with		\showglossarycounter:new .	249
\ifcsundef	51	\showglossaryentries:new .	249
\istfile: deprecated	170	\showglossaryin:new	249
glossaryentry:new	195	\showglossaryout:new	249
glossarysubentry:new	195	\showglossarytitle:new ...	249
\newglossaryentry: replaced		\showglosymbol:new	247
\DeclareRobustCommand		\showglosymbolplural:new .	247
with\newrobustcmd	66	\showglotext:new	244
\newglossarystyle: replaced		\showglotype:new	245
\@ifundefined with		\showglouserii:new	245
\ifcsundef	203	\showglouseriii:new	246
\ns@newglossary: added		\showglouseriv:new	246
\@glst@defsortcount	57	\showglouservi:new	246
replaced \@ifundefined with		subentrycounter:new	9
\ifcsundef	57	\writeist: added xindy-only	
entrycounter:new	9	macro definitions to glossary	
entrycounterwithin:new	9	open tag	158
\oldacronym:replaced\@ifundefined		modified to support new for-	
with\ifcsundef	207	mat	156
compatible-2.07: compatible-			
2.07 option added	26		
long:new	63		
longplural:new	63	\@glstwritefiles: added check	
nonumberlist: now boolean ...	61	for empty glossaries	170
sort:new	10	General: made robust	119
counter:replaced\@ifundefined		\ACRfull: made robust	210
with\ifcsundef	61	\Acrfull: made robust	209
\printglossary: replaced		\acrfull: made robust	208
\@ifundefined with		\acrfullformat: removed	
\ifcsundef	178	\acronymfont as it should al-	
\SetDescriptionFootnoteAcronymDisplayStyle:		be set in the second ar-	
expanded options link op-		gument	209
tions	228	\ACRfullpl: made robust	211

<code>\Acrfullpl: made robust</code>	210	<code>\GLSuserii: made robust</code>	133
<code>\acrfullpl: made robust</code>	210	<code>\Glsuserii: made robust</code>	133
<code>\ACRlong: made robust</code>	142	<code>\glsuserii: made robust</code>	132
<code>\Acrlong: made robust</code>	141	<code>\GLSuseriii: made robust</code>	134
<code>\acrlong: made robust</code>	141	<code>\Glsuseriii: made robust</code>	134
<code>\ACRlongpl: made robust</code>	144	<code>\glsuseriii: made robust</code>	133
<code>\Acrlongpl: made robust</code>	143	<code>\GLSuseriv: made robust</code>	135
<code>\acrlongpl: made robust</code>	142	<code>\Glsuseriv: made robust</code>	135
<code>\ACRshort: made robust</code>	138	<code>\glsuseriv: made robust</code>	134
<code>\Acrshort: made robust</code>	138	<code>\GLSuseriv: made robust</code>	136
<code>\acrshort: made robust</code>	137	<code>\Glsuseriv: made robust</code>	135
<code>\ACRshortpl: made robust</code>	140	<code>\glsuseriv: made robust</code>	135
<code>\Acrshortpl: made robust</code>	139	<code>\GLSuservi: made robust</code>	137
<code>\acrshortpl: made robust</code>	139	<code>\Glsuservi: made robust</code>	136
<code>\Gls: made robust</code>	118	<code>\glsuservi: made robust</code>	136
<code>\glsadd: made robust</code>	153			
<code>\glsaddall: made robust</code>	154	3.02		
<code>\GLSdesc: made robust</code>	129	<code>\@@do@wrglossary:</code>	changed	
<code>\glsdesc: made robust</code>	128	<code>\@glslocref to \theglentrycounter</code>		175
<code>\glsdesc: made robust</code>	128	<code>\@do@wrglossary:</code>	changed	
<code>\GLSdescplural: made robust</code>	. 129		<code>\@do@wr@glossary</code>	to test for	
<code>\Glsdescplural: made robust</code>	. 129		<code>indexonlyfirst</code>	option; put old	
<code>\glsdescplural: made robust</code>	. 129		<code>\@do@wr@glossary</code>	code into	
<code>\glsfirst: made robust</code>	124	<code>\@@do@wrglossary</code>	172
<code>\GLSfirstplural: made robust</code>	127		<code>\@gls@missingnumberlist:</code>		
<code>\Glsfirstplural: made robust</code>	126		<code>new</code>	63
<code>\glsfirstplural: made robust</code>	126		<code>\@gls@writefiles:</code>	added check	
<code>\glslink: made robust</code>	104	<code>for existence of token in case</code>		
<code>\GLSname: made robust</code>	128	<code>\makeglossaries</code>	has been	
<code>\Glsname: made robust</code>	127	<code>omitted</code>	170
<code>\glsname: made robust</code>	127	<code>\@printglossary:</code>	add a way to	
<code>\GLSpl: made robust</code>	122	<code>fetch current entry label</code>	... 180	
<code>\Glspl: made robust</code>	121	<code>savenumberlist: new</code>	7
<code>\glspl: made robust</code>	120	<code>ucmark: new</code>	9
<code>\GLSplural: made robust</code>	126	<code>\gls@defglossaryentry:</code>	added	
<code>\GLSsymbol: made robust</code>	130	<code>numberlist element</code>	79
<code>\Glsymbol: made robust</code>	130	<code>\gls@save@numberlist: new</code>	. 178	
<code>\glsymbol: made robust</code>	130	<code>\gls@wrglossary:</code>	added check	
<code>\GLSsymbolplural: made robust</code>			<code>for glossary file defined</code>	172
<code>.....</code>			<code>\glsdisplaynumberlist: new</code>	151	
<code>\Glsymbolplural: made robust</code>			<code>\glentrycounter:</code>	set default	
<code>.....</code>			<code>value</code>	107
<code>\glsymbolplural: made robust</code>			<code>\Glsentryfull:</code>	fixed bug (re-	
<code>.....</code>			<code>placed \glentryshortpl</code>		
<code>\Glstext: made robust</code>	124	<code>with \glentryshort)</code>	151
<code>\glstext: made robust</code>	123	<code>\glsentryfullpl: fixed bug (re-</code>		
<code>\GLSuseri: made robust</code>	132	<code>placed \glentryshort</code>	with	
<code>\Glsuseri: made robust</code>	132	<code>\glentryshortpl)</code>	151
<code>\glsuseri: made robust</code>	132	<code>\glentrynumberlist: new</code>	.. 151	

<code>\glsmoveentry:new</code>	81	<code>longragged:</code> added check for	
<code>\glsnumlistlastsep:new</code> ...	152	<code>glsnogroupskip</code>	271
<code>\glsnumlistsep:new</code>	152	<code>longragged3col:</code> added check	
<code>\glsresetsubentrycounter:</code>		for <code>glsnogroupskip</code>	273
<code>new</code>	196	<code>nopostdot:new</code>	9
<code>\ifglshaschildren:new</code>	52	<code>tree:</code> added check for	
<code>\ifglshasparent:new</code>	52	<code>glsnogroupskip</code>	295
<code>\makeglossaries:</code> added list		<code>treenoname:</code> added check for	
<code>parser</code>	165	<code>glsnogroupskip</code>	296
<code>indexonlyfirst:new</code>	23	<code>super:</code> added check for	
<code>\renewglossarystyle:new</code> ..	204	<code>glsnogroupskip</code>	281
<code>\showglossaryentries:</code> fixed		<code>super3col:</code> added check for	
<code>misspelt command</code>	249	<code>glsnogroupskip</code>	282
<code>\SmallNewAcronymDef:</code> fixed		<code>super4col:</code> added check for	
<code>broken short and long plural</code>	238	<code>glsnogroupskip</code>	284
3.03		<code>superragged:</code> added check for	
<code>\@gls@sanitizesort:new</code>	18	<code>glsnogroupskip</code>	288
<code>\@gls@setupsort@standard:</code>		<code>superragged3col:</code> added check	
<code>used \@gls@sanitizesort</code> .	10	for <code>glsnogroupskip</code>	289
<code>\@printglossary:</code> allow title to		3.04	
<code>override default toctitle</code>	179	<code>\@do@wrglossary:</code> changed	
General: allow title to set toctitle	192	<code>\theglsentrycounter</code> back	
<code>\glsinlinedescformat:new</code> .	260	to <code>\@glslocref</code>	175
<code>\glsinlineemptydescformat:</code>		<code>\@do@wrglossary:</code> modified to	
<code>new</code>	261	<code>compensate for possible incor-</code>	
<code>\glsinlinenameformat:new</code> .	260	<code>rect page number</code>	173
<code>\glsinlinepostchild:new</code> ..	260	<code>\@gls@escbsdq:</code> unsani-	
<code>\glsinlinesubdescformat:</code>		<code>tize</code> <code>\gls@numberpage,</code>	
<code>new</code>	261	<code>\gls@alphpage,\gls@Alphpage</code>	
<code>\glsinlinesubnameformat:</code>		and <code>\gls@romanpage</code>	109
<code>new</code>	261	<code>\@print@glossary:</code> Moved aux	
<code>\glspostinline:</code> replaced “.”		<code>write to end of document to</code>	
<code>with \glspostdescription</code>	260	<code>prevent unwanted whatsit oc-</code>	
<code>atlongragged4col:</code> added		<code>curring here.</code>	181
<code>check for glsnogroupskip</code> ..	274	General: Added check for doc	
<code>altsuperragged4col:</code> added		<code>package</code>	4
<code>check for glsnogroupskip</code> ..	291	<code>added datatool-base as a re-</code>	
<code>alttree:</code> added check for		<code>quired package</code>	4
<code>glsnogroupskip</code>	299	<code>added local key</code>	103
<code>index:</code> added check for		<code>\gls@Alphpage:new</code>	172
<code>glsnogroupskip</code>	294	<code>\gls@alphpage:new</code>	172
<code>nogroupskip:new</code>	9	<code>\gls@disablepagerefexpansion:</code>	
<code>long:</code> added check for		<code>new</code>	172
<code>glsnogroupskip</code>	265	<code>\gls@numberpage:new</code>	173
<code>long3col:</code> added check for		<code>\gls@protected@pagefmts:</code>	
<code>glsnogroupskip</code>	267	<code>new</code>	172
<code>long4col:</code> added check for		<code>\gls@romanpage:new</code>	173
<code>glsnogroupskip</code>	268	<code>\glsdefmain:</code> added check for	
		<code>doc package</code>	12

<code>\glsorg@endtheglossary: new</code>	5	3.08a	
<code>\glsorg@theglossary: new</code>	5		<code>\@glo@storeentry:</code> no longer need to check for special characters in any of the fields other than sort 82
<code>altlist:</code> replaced <code>\newline</code> with paragraph break	263		updated for <code>\glossentry</code> 83
<code>\PrintChanges:</code> new	5		<code>\@glossaryentryfield:</code> switched to <code>\glossentry</code> 82
3.05			<code>\@glossarysubentryfield:</code> switched to <code>\subglossentry</code> 82
<code>\@do@wrglossary:</code> add Roman case. Fixed bugs in the else statements	174		General: added <code>nogroupskip</code> key to <code>\printglossary</code> 192
<code>\@gls@link:</code> added check for “no-hypertypes”	106		removed definition of <code>\@glossaryentryfield</code> .. 340
<code>\@gls@nohyperlist:</code> new	16		removed definition of <code>\@glossarysubentryfield</code> 340
<code>mcolalmtree:</code> replaced ‘2’ with <code>\glsmcols</code>	279		<code>\compatibleglossentry:</code> new 198
<code>mcolindex:</code> replaced ‘2’ with <code>\glsmcols</code>	276		<code>\compatiblesubglossentry:</code> new 199
<code>mcoltree:</code> replaced ‘2’ with <code>\glsmcols</code>	277		<code>\glossaryentryfield:</code> deprecated 200
<code>mcoltreename:</code> replaced ‘2’ with <code>\glsmcols</code>	278		<code>\Glossentrydesc:</code> new 199
<code>\gls@protected@pagefmts:</code> added Roman to list	172		<code>\glossentrydesc:</code> new 199
<code>\gls@Romanpage:</code> new	173		<code>\Glossentryname:</code> new 198
<code>\GlsDeclareNoHyperList:</code> new	16		<code>\glossentryname:</code> new 198
<code>\glsgetgrouplabel:</code> fixed bug (typo in <code>\equal</code>)	202		<code>\Glossentrysymbol:</code> new 199
<code>\nopostdesc:</code> made robust	33		<code>\glossentrysymbol:</code> new 199
<code>nohypertypes:</code> new	16		<code>\gls@assign@desc@field:</code> new 17
3.06			<code>\gls@assign@descplural@field:</code> new 17
<code>\@xdy@main@language:</code> Changed back to using <code>\languagename</code>	25		<code>\gls@assign@field:</code> new 65
<code>\findrootlanguage:</code> Obsoleted	47		<code>\gls@ifnotmeasuring:</code> new . . . 83
3.07			<code>\glsaddallunused:</code> new 154
<code>\@gls@link:</code> fixed bug that failed to find entry in list	106		<code>\glsexpandfields:</code> new 65
<code>\glossary preamble:</code> modified to work with <code>\setglossary preamble</code>	36		<code>\glsnoexpandfields:</code> new 65
<code>\gls@doclearpage:</code> added check for <code>openright</code>	39		<code>\glssee:</code> made robust 177
<code>\glspostdescription:</code> Added <code>spacefactor</code> code	9		<code>\glsseeformat:</code> made robust .. 177
<code>\GlsSetXdyCodePage:</code> Added check for <code>fontspec</code>	48		<code>\glsseeitem:</code> made robust 178
<code>\SetDescriptionAcronymDisplayStyle:</code> now using <code>\glsdoparenifnotempty</code>	232		<code>\glsseelist:</code> made robust 177
<code>\setglossary preamble:</code> new ..	36		<code>\ifglsdescsuppressed:</code> new .. 53
			<code>\ifglshasdesc:</code> new 52
			<code>\ifglshassymbol:</code> new 53
			list: updated list style to use <code>\glossentry</code> and <code>\subglossentry</code> 261
			listdotted: updated listdotted style to use <code>\glossentry</code> and <code>\subglossentry</code> 264

altlist: updated altlist style to use \glossentry and \subglossentry	262	\Glsentryshortpl: made robust	150
altlongragged4col: updated to use \glossentry and \subglossentry	274	\Glsentrysymbol: made robust	148
alttree: updated to use \glossentry and \subglossentry	297	\Glsentrysymbolplural: made robust	148
index: added paragraph break at end of environment	293	\Glsentrytext: made robust ..	147
updated to use \glossentry and \subglossentry	293	\Glsentryuseri: made robust .	149
inline: updated inline style to use \glossentry and \subglossentry	259	\Glsentryuserii: made robust	149
long: updated to use \glossentry and \subglossentry	265	\Glsentryuseriii: made robust	149
longragged: updated to use \glossentry and \subglossentry	271	\Glsentryuseriv: made robust	150
longragged3col: updated to use \glossentry and \subglossentry	272	\Glsentryuserv: made robust .	150
tree: updated to use \glossentry and \subglossentry	294	\Glsentryuservi: made robust	150
\setglossarystyle: new	203	\glstextup: new	208
\setglossentrycompatibility: new	200	\ifglshassymbol: changed test to check for \@gls@default@symbol	53
superragged: updated to use \glossentry and \subglossentry	287	3.10a	
3.09a		\@gls@keymap: new	67
\@gls@assign@symbolplural@field: new	18	\@gls@provide@newglossary: new	55
\@gls@default@value: new ...	60	\@gls@writedef: new	67
\Glsentrydesc: made robust ..	147	\@gls@defaultplural: Obsolete .	63
\Glsentrydescplural: made robust	147	\@gls@nodesc: new	63
\Glsentryfirst: made robust .	148	\@print@glossary: Added providecommand code to aux file	181, 182
\Glsentryfirstplural: made robust	148	\gls@assign@type@field: new	17
\Glsentryfull: made robust ..	151	\gls@defglossaryentry: Changed to using \@gls@default@value	76
\Glsentryfullpl: made robust	151	new	75
\Glsentrylong: made robust ..	151	\gls@writedefhook: new	75
\Glsentrylongpl: made robust	151	\makeglossaries: Added providecommand code to aux file	164
\Glsentryname: made robust ..	145	\new@glossaryentry: new	66
\Glsentryplural: made robust	147	\ns@newglossary: added \@gls@provide@newglossary	57
\Glsentryshort: made robust .	150	3.11a	
		\@ACRlong: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	339
		\@ACRshort: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	338

`\@Acrlong:` added `\glslabel`,
`\glsifplural`, `\glscapscase`,
`\glsinsert` and `\glscustomtext`
..... 339
`\@Acrshort:` added `\glslabel`,
`\glsifplural`, `\glscapscase`,
`\glsinsert` and `\glscustomtext`
..... 338
`\@GLS@:` add `\glslabel`,
`\glsifplural`, `\glscapscase`,
`\glscustomtext` and
`\glsinsert` 119
change to using `\glsentryfmt`
style commands 119
removed `\MakeUppercase`
(now moved to `\glsentryfmt`)
..... 120
`\@GLSpl@:` add `\glslabel`,
`\glsifplural`, `\glscapscase`,
`\glscustomtext` and
`\glsinsert` 122
change to using `\glsentryfmt`
style commands 122
removed `\MakeUppercase`
as now dealt with in
`\glsentryfmt` 122
`\@GLs@:` add `\glsifplural`,
`\glscapscase`, `\glscustomtext`
and `\glsinsert` 119
change to using `\glsentryfmt`
style commands 119
removed `\makefirstuc` (now
dealt with in `\glsentryfmt`) 119
`\@GLspl@:` add `\glsifplural`,
`\glscapscase`, `\glscustomtext`
and `\glsinsert` 121
change to using `\glsentryfmt`
style commands 121
removed `\makefirstuc` (now
dealt with in `\glsentryfmt`) 121
`\@acrlong:` added `\glslabel`,
`\glsifplural`, `\glscapscase`,
`\glsinsert` and `\glscustomtext`
..... 339
`\@acrshort:` added `\glslabel`,
`\glsifplural`, `\glscapscase`,
`\glsinsert` and `\glscustomtext`
..... 337
`\@gls@:` add `\glslabel`,
`\glsifplural`, `\glscapscase`,
`\glscustomtext` and
`\glsinsert` 118
change to using `\glsentryfmt`
style commands 118
`\@gls@noexpand@fields:` Fixed
bug `expand` replaced with
`noexpand` 64
`\@glsdisp:` add `\glslabel`,
`\glsifplural`, `\glscapscase`,
`\glscustomtext` and
`\glsinsert` 123
change to using `\glsentryfmt`
style commands 123
`\@glspl@:` add `\glslabel`,
`\glsifplural`, `\glscapscase`,
`\glscustomtext` and
`\glsinsert` 120
change to using `\glsentryfmt`
style commands 120
General: added `\glslabel`,
`\glsifplural`, `\glscapscase`,
`\glsinsert` and `\glscustomtext`
..... 137–144
changed to just use `\Glsentrydescplural`
..... 129
changed to just use `\glsentrydescplural`
..... 129, 130
changed to just use `\Glsentrydesc`
..... 128
changed to just use `\glsentrydesc`
..... 128, 129
changed to just use `\Glsentryfirstplural`
..... 127
changed to just use `\glsentryfirstplural`
..... 126, 127
changed to just use `\Glsentryfirst`
..... 125
changed to just use `\glsentryfirst`
..... 125
changed to just use `\Glsentryname`
..... 128
changed to just use `\glsentryname`
..... 127, 128
changed to just use `\Glsentryplural`
..... 126
changed to just use `\glsentryplural`
..... 125, 126

changed to just use `\Glsentrysymbolplural` default counter if none supplied 131 79
 changed to just use `\Glsentrysymbolplural` `\gls@doentryfmt:new` 131 55
 changed to just use `\Glsentrysymbol` `\glsdisplay:obsoleted` 130 101
 changed to just use `\Glsentrysymbol` `\glsdisplayfirst:obsoleted` . 101
 130, 131 `\glsgenentryfmt:new` 96
 Changed to just use `\Glsentrytext` 124 `\glsgetgrouptitle:` Added
 check in case non-Latin alpha-
 bet in use 201
 changed to just use `\Glsentrytext` 124 `\gls glossarymark:` replaced
`\MakeUppercase` with
`\mfirstucMakeUppercase` . 37
 changed to just use `\Glsentryuseriii` `\glsnavigation:` switched to us-
 ing `\@gls@getgrouptitle` 258
 changed to just use `\Glsentryuseriii` `\ifglshasdesc:` replaced
`\ifdefempty` with `\ifcsempy`
 134 52
 changed to just use `\Glsentryuserii` `\ifglshaslong:` new 53
 133 `\ifglshasshort:` new 53
 changed to just use `\Glsentryuserii` `\ifglshassymbol:` replaced
`\ifdefempty` with `\ifcsempy`
 133 53
 changed to just use `\Glsentryuseriv` `\ifglused:` replaced `\ifthenelse`
 135 with `\ifbool` 51
 changed to just use `\Glsentryuseriv` 134, 135 `\longnewglossaryentry:` new . 75
 changed to just use `\Glsentryuseri` 132 `\ns@newglossary:` replaced
`\glsdisplay` and `\glsdisplayfirst`
 changed to just use `\Glsentryuseri` 132 with `\glsentryfmt` 57
 changed to just use `\Glsentryuservi` 137 compatible-3.07: cnew 26
 changed to just use `\Glsentryuservi` 136, 137 `\SetCustomDisplayStyle:` up-
 dated to use `\defglentryfmt`
 136 242
 changed to just use `\Glsentryuserv` 136 `\SetDefaultAcronymDisplayStyle:`
 changed to use `\defglentryfmt`
 135, 136 226
 Now requires `textcase` 4 `\SetDescriptionAcronymDisplayStyle:`
 updated to use `\defglentryfmt`
 232
`acronymlists:` replaced `\SetDescriptionDUAAcronymDisplayStyle:`
`\@addtoacronymlists` with updated to use `\defglentryfmt`
`\DeclareAcronymList` 15 230
`\defglsdisplay:` obsoleted ... 101 `\SetDescriptionFootnoteAcronymDisplayStyle:`
 updated to use `\defglentryfmt`
 102 228
`\defglsdisplayfirst:` obso-
 leted 102 `\SetDUADisplayStyle:` updated
 to use `\defglentryfmt` .. 240
`\defglentryfmt:new` 55 `\SetFootnoteAcronymDisplayStyle:`
 updated to use `\defglentryfmt`
 50 234
`\forglentries:` replaced `\ifx`
 with `\ifdefempty` 50
`\gls@assign@desc:` new 75
`\gls@defglossaryentry:` Fixed

<code>\SetSmallAcronymDisplayStyle:</code> updated to use <code>\defglsentryfmt</code> 237	<code>\Glsentryfull:</code> changed to use <code>\acrfullformat</code> 151
<code>\setupglossaries:</code> new 28	<code>\glsentryfull:</code> changed to use <code>\acrfullformat</code> 151
<code>\showglolong:</code> new 247	<code>\Glsentryfullpl:</code> changed to use <code>\acrfullformat</code> 151
<code>\showgloshort:</code> new 247	<code>\glsentryfullpl:</code> changed to use <code>\acrfullformat</code> 151
numbers: new 27	<code>\glsglossarymark:</code> renamed <code>\glossarymark</code> to <code>\glsglossarymark</code> to avoid conflict with memoir 37
symbols: new 27	<code>\glsprestandardsort:</code> new ... 10
3.12a	<code>\glssetexpandfield:</code> new ... 17
<code>\gls@defglossaryentry:</code> added	<code>\glssetnoexpandfield:</code> new .. 17
<code>\glslabel</code> 75	<code>altsuper4colheader:</code> switched to <code>\tabularnewline</code> 285
<code>\glsaddkey:</code> new 69	<code>altsuper4colheaderborder:</code> switched to <code>\tabularnewline</code> 286
3.13a	long: switched to <code>\tabularnewline</code> 265
<code>\@gls@assign@symbol@field:</code> changed to use <code>\glssetnoexpandfield</code> 18	long3col: switched to <code>\tabularnewline</code> 266
<code>\@gls@assign@symbolplural@field:</code> changed to use <code>\glssetnoexpandfield</code> 18	long3colheader: switched to <code>\tabularnewline</code> 267
<code>\@gls@link:</code> removed <code>\relax</code> . 107	long3colheaderborder: switched to <code>\tabularnewline</code> 267
<code>\@gls@notranslatorhook:</code> new 21	long4col: switched to <code>\tabularnewline</code> 268
<code>\@gls@setupsort@standard:</code> moved <code>\@gls@santizesort</code> to <code>\glsprestandardsort</code> .. 10	long4colheader: switched to <code>\tabularnewline</code> 268
ucmark: added check for memoir . 9	longheader: switched to <code>\tabularnewline</code> 266
see: added <code>\gls@checkseeallowed</code> 61	longheaderborder: switched to <code>\tabularnewline</code> 266
<code>\glossarysection:</code> changed	<code>\SetFootnoteAcronymDisplayStyle:</code> fixed missing argument bug 235
<code>\glossarymark</code> to <code>\glsglossarymark</code> 37	super: switched to <code>\tabularnewline</code> 280
<code>\glossarystyle:</code> fixed bug caused by using <code>\ifdef</code> in- stead of <code>\ifcsdef</code> 203	super3col: switched to <code>\tabularnewline</code> 282
<code>\gls@assign@desc@field:</code> changed to use <code>\glssetnoexpandfield</code> 17	super3colheader: switched to <code>\tabularnewline</code> 283
<code>\gls@assign@descplural@field:</code> changed to use <code>\glssetnoexpandfield</code> 17	super4col: switched to <code>\tabularnewline</code> 284
<code>\gls@assign@name@field:</code> changed to use <code>\glssetnoexpandfield</code> 18	super4colheader: switched to <code>\tabularnewline</code> 284
<code>\gls@assign@type@field:</code> changed to use <code>\glssetexpandfield</code> 17	super4colheaderborder: switched to <code>\tabularnewline</code>
<code>\gls@checkseeallowed:</code> new .. 61	
<code>\glsaddallunused:</code> set default to <code>\@glo@types</code> 154	

.....	285	sanitize: fixed bug that caused	
superheader: switched to		an error here	21
\tabularnewline	281	sc-short-long: new	217
superheaderborder: switched to		sc-short-long-desc: new ...	219
\tabularnewline	281	\Genacrfullformat: new	101
3.14a		\genacrfullformat: new	100
\@glswritefiles: renamed		\GenericAcronymFields: new	213
\glswritefiles to \@glswritefiles		\Genplacrfullformat: new ..	101
and used “savewrites” option		\genplacrfullformat: new ..	101
to set \glswritefiles	170	\Glsentryfull: bug fix: added	
General: new	250	missing \acronymfont	151
acronyms: new	14	\glentryfull: bug fix: added	
\gls@defglossaryentry: added		missing \acronymfont	151
check for existence of default		\Glsentryfullpl: bug fix: added	
glossary	77	missing \acronymfont	151
set the default for firstplural to		\glentryfullpl: bug fix: added	
be the value of plural	79	missing \acronymfont	151
xindygloss: new	25	\glsgenacfmt: new	98
\longprovideglossaryentry:		\GlsUseAcrEntryDisplayStyle:	
new	75	new	214
compatible-2.07: added check		\GlsUseAcrStyleDefs: new ..	214
for 2.07 before setting 3.07		short-long: new	216
compatibility	27	short-long-desc: new	219
notranslate: new	22	xindynoglsnumbers: new	26
\provideglossaryentry: new .	66	sm-short-long: new	217
4.0		sm-short-long-desc: new ...	219
\gls@defglossaryentry: added		\makeglossaries: made pream-	
check for first key	79	ble only	165
4.01		index: new	27
General: fixed non-value options		\newacronymstyle: new	214
so that they can be passed to		long-sc-short: new	217
document class	7	long-sc-short-desc: new ...	218
\CustomAcronymFields: in-		long-short: new	215
serted missing comma	242	long-short-desc: new	218
4.02		long-sm-short: new	217
\@acrfull: now using \acrfullfmt		long-sm-short-desc: new ...	218
.....	209	long-sp-short-desc: new ...	218
\@gls@indexdef: new	28	footnote: new	222
\@gls@numbersdef: new	27	footnote-desc: new	224
\@gls@symbolsdef: new	27	footnote-sc: new	223
General: Removed \acronymfont		footnote-sc-desc: new	224
.....	141–144	footnote-sm: new	223
\ACRfullfmt: new	210	footnote-sm-desc: new	224
\Acrfullfmt: new	209	\setacronymstyle: new	213
\acrfullfmt: new	209	\SetDescriptionAcronymDisplayStyle:	
\ACRfullplfmt: new	211	Moved check for empty cus-	
\Acrfullplfmt: new	211	tom text to prevent unwanted	
\acrfullplfmt: new	210	parenthetical material	232
\acronymentry: new	213		

<code>\SetDescriptionFootnoteAcronymDisplayStyle:</code>	<code>\@gls@link@opts</code>
Moved check for empty custom text to prevent unwanted parenthetical material	and <code>\@gls@link@label</code> to <code>\@gls@link</code> 106
228	<code>\@gls@writedef:</code> added
<code>\SetFootnoteAcronymDisplayStyle:</code>	<code>\glsdetoklabel</code> 67
Moved check for empty custom text to prevent unwanted parenthetical material	<code>\@glsdisp:</code> removed <code>\glslabel</code> (defined in <code>\@gls@link</code>) . . . 123
234	<code>\@glspl@:</code> removed <code>\glslabel</code> (defined in <code>\@gls@link</code>) . . . 120
<code>\SetGenericNewAcronym:</code> new	<code>\@printglossary:</code> added
212	<code>\glsdetoklabel</code> 180
<code>\SetSmallAcronymDisplayStyle:</code>	General: changed default to <code>\@empty</code> instead of <code>\relax</code> . . 26
Moved check for empty custom text to prevent unwanted parenthetical material	removed <code>\glslabel</code> (defined in <code>\@gls@link</code>) 137
237	<code>sc-short-long-desc:</code> redefined to use accessibility information 344
<code>dua:</code> new 220	<code>\compatibleglossentry:</code> added
<code>dua-desc:</code> new 222	<code>\glsdetoklabel</code> 320
<code>numberedsection:</code> added	<code>\compatiblesubglossentry:</code> added <code>\glsdetoklabel</code> . . . 320
<code>nameref</code> option 6	<code>\Genacrfullformat:</code> redefined to use accessibility information 337
4.03	<code>\genacrfullformat:</code> redefined to use accessibility information 337
<code>\@do@wrglossary:</code> added	<code>\Genplacrfullformat:</code> redefined to use accessibility information 337
<code>\glsdetoklabel</code> 174	<code>\genplacrfullformat:</code> redefined to use accessibility information 337
<code>\@ACRlong:</code> removed <code>\glslabel</code> (defined in <code>\@gls@link</code>) . . . 339	<code>\glossentryname:</code> added
<code>\@ACRshort:</code> removed <code>\glslabel</code> (defined in <code>\@gls@link</code>) . . . 338	<code>\glsdetoklabel</code> 198
<code>\@Acrlong:</code> removed <code>\glslabel</code> (defined in <code>\@gls@link</code>) . . . 339	<code>\gls@defglossaryentry:</code> added
<code>\@Acrshort:</code> removed <code>\glslabel</code> (defined in <code>\@gls@link</code>) . . . 338	<code>\glsdetoklabel</code> 75
<code>\@GLS@:</code> removed <code>\glslabel</code> (defined in <code>\@gls@link</code>) 119	replaced <code>#1</code> with <code>\@glo@label</code> 77
<code>\@GLSpl:</code> removed <code>\glslabel</code> (defined in <code>\@gls@link</code>) . . . 122	replaced <code>\ifthenelse</code> with <code>\ifdefequal</code> 78
<code>\@Gls@:</code> removed <code>\glslabel</code> (defined in <code>\@gls@link</code>) 119	<code>\glsadd:</code> added <code>\glsdetoklabel</code> 153
<code>\@Gls@entry@field:</code> new . . . 145	<code>\glsaddkey:</code> switched to using <code>\@gls@field@link</code> 70
<code>\@Glspl@:</code> removed <code>\glslabel</code> (defined in <code>\@gls@link</code>) . . . 121	<code>\glsdetoklabel:</code> new 50
<code>\@acrlong:</code> removed <code>\glslabel</code> (defined in <code>\@gls@link</code>) . . . 339	<code>\glsdisplaynumberlist:</code> added
<code>\@acrshort:</code> removed <code>\glslabel</code> (defined in <code>\@gls@link</code>) . . . 337	<code>\glsdetoklabel</code> 152
<code>\@gls@:</code> removed <code>\glslabel</code> (defined in <code>\@gls@link</code>) 118	<code>\glsdoifexistsorwarn:</code> new . . 52
<code>\@gls@access@display:</code> new . 326	
<code>\@gls@entry@field:</code> new . . . 145	
<code>\@gls@fetchfield:</code> new 68	
<code>\@gls@field@link:</code> new 123	
<code>\@gls@link:</code> added <code>\glsdetoklabel</code> 106	

<code>\glsentryaccess</code> : switched to using <code>\@gls@entry@field</code> . 324	<code>\glsreset</code> :added <code>\glsdetoklabel</code> 84
<code>\glsentrydescaccess</code> : switched to using <code>\@gls@entry@field</code> 325	<code>\glsseelist</code> :added <code>\expandafter</code> commands 177
<code>\glsentrydescpluralaccess</code> : switched to using <code>\@gls@entry@field</code> 325	<code>\glsstepentry</code> :added <code>\glsdetoklabel</code> 196
<code>\glsentryfirstaccess</code> :switched to using <code>\@gls@entry@field</code> 324	<code>\glsstepsubentry</code> : added <code>\glsdetoklabel</code> 196
<code>\glsentryfirstplural</code> : added <code>\glsdetoklabel</code> 148	<code>\glsunset</code> :added <code>\glsdetoklabel</code> 84
<code>\glsentrylongaccess</code> : switched to using <code>\@gls@entry@field</code> 325	<code>short-long</code> : commented spurious EOL 216
<code>\glsentrylongpluralaccess</code> : switched to using <code>\@gls@entry@field</code> 326	redefined to use accessibility information 342
<code>\glsentrypluralaccess</code> : switched to using <code>\@gls@entry@field</code> 325	<code>short-long-desc</code> : redefined to use accessibility information 344
<code>\glsentryshortaccess</code> :switched to using <code>\@gls@entry@field</code> 325	<code>\ifglsdescsuppressed</code> : added <code>\glsdetoklabel</code> 53
<code>\glsentryshortpluralaccess</code> : switched to using <code>\@gls@entry@field</code> 325	fixed typo 53
<code>\glsentrysymbolaccess</code> : switched to using <code>\@gls@entry@field</code> 325	<code>\ifglsentryexists</code> : added <code>\glsdetoklabel</code> 51
<code>\glsentrysymbolpluralaccess</code> : switched to using <code>\@gls@entry@field</code> 325	<code>\ifglschaschildren</code> : added <code>\glsdetoklabel</code> 52
<code>\glsentrytextaccess</code> : switched to using <code>\@gls@entry@field</code> 324	<code>\ifglschasdesc</code> :added <code>\glsdetoklabel</code> 52
<code>\glsngenacfmt</code> : redefined to use accessibility information ... 334	<code>\ifglschasfield</code> :new 53
<code>\glsngenentryfmt</code> : redefined to use accessibility information 332	<code>\ifglschaslong</code> :added <code>\glsdetoklabel</code> 53
<code>\gls hyperlink</code> :added <code>\glsdetoklabel</code> 153	<code>\ifglschasparent</code> : added <code>\glsdetoklabel</code> 52
<code>\glslocalreset</code> : added <code>\glsdetoklabel</code> 84	<code>\ifglschasshort</code> : added <code>\glsdetoklabel</code> 53
<code>\glslocalunset</code> : added <code>\glsdetoklabel</code> 84	<code>\ifglschassymbol</code> : added <code>\glsdetoklabel</code> 53
<code>\glsmoveentry</code> :added <code>\glsdetoklabel</code> 81	replaced <code>\ifcsemtyp</code> with <code>\ifdefempty</code> and replaced <code>\ifx</code> with <code>\ifdefequal</code> 53
replaced <code>\ifthenelse</code> with <code>\ifdefequal</code> 81	<code>\ifglsused</code> :added <code>\glsdetoklabel</code> 51
<code>\glsrefentry</code> :added <code>\glsdetoklabel</code> 196	<code>sm-short-long-desc</code> : redefined to use accessibility information 344
	<code>long-sc-short-desc</code> : redefined to use accessibility information 343
	<code>long-short</code> : redefined to use accessibility information 341
	<code>long-short-desc</code> : redefined to use accessibility information 343

long-sm-short-desc: redefined to use accessibility information	343	<code>\showglonameaccess:</code>	added
footnote: redefined to use accessibility information	347	<code>\glsdetoklabel</code> 355
footnote-desc: redefined to use accessibility information ...	349	<code>\showgloparent:</code>	added
footnote-sc: redefined to use accessibility information	349	<code>\glsdetoklabel</code> 244
footnote-sc-desc: redefined to use accessibility information	350	<code>\showgloplural:</code>	added
footnote-sm: redefined to use accessibility information	349	<code>\glsdetoklabel</code> 244
footnote-sm-desc: redefined to use accessibility information	350	<code>\showglopluralaccess:</code>	added
<code>\renewacronymstyle:</code> new ...	214	<code>\glsdetoklabel</code> 356
<code>\showglocounter:</code>	added	<code>\showgloshort:</code>	added <code>\glsdetoklabel</code>
<code>\glsdetoklabel</code> 245	247
<code>\showgloDESC:</code> added <code>\glsdetoklabel</code> 246	<code>\showgloshortaccess:</code>	added
<code>\showgloDESCaccess:</code>	added	<code>\glsdetoklabel</code> 356
<code>\glsdetoklabel</code> 356	<code>\showgloshortpluralaccess:</code>	added <code>\glsdetoklabel</code> ...
<code>\showgloDESCplural:</code>	added	356
<code>\glsdetoklabel</code> 247	<code>\showglosort:</code>	added <code>\glsdetoklabel</code>
<code>\showgloDESCpluralaccess:</code>	added <code>\glsdetoklabel</code>	247
..... 356		<code>\showglosymbol:</code>	added
<code>\showgloFIRST:</code> added <code>\glsdetoklabel</code> 245	<code>\glsdetoklabel</code> 247
<code>\showgloFIRSTaccess:</code>	added	<code>\showglosymbolaccess:</code>	added
<code>\glsdetoklabel</code> 356	<code>\glsdetoklabel</code> 356
<code>\showgloFIRSTpl:</code>	added	<code>\showglosymbolplural:</code>	added
<code>\glsdetoklabel</code> 245	<code>\glsdetoklabel</code> 247
<code>\showgloFIRSTpluralaccess:</code>	added <code>\glsdetoklabel</code> ...	<code>\showglosymbolpluralaccess:</code>	added <code>\glsdetoklabel</code> ...
..... 356		356
<code>\showgloFLAG:</code> added <code>\glsdetoklabel</code> 248	<code>\showgloTEXT:</code> added <code>\glsdetoklabel</code> 244
<code>\showgloINDEX:</code> added <code>\glsdetoklabel</code> 248	<code>\showgloTEXTaccess:</code>	added
<code>\showgloLEVEL:</code> added <code>\glsdetoklabel</code> 244	<code>\glsdetoklabel</code> 356
<code>\showgloLONG:</code> added <code>\glsdetoklabel</code> 247	<code>\showgloTYPE:</code> added <code>\glsdetoklabel</code> 245
<code>\showgloLONGaccess:</code>	added	<code>\showgloUSERi:</code> added <code>\glsdetoklabel</code> 245
<code>\glsdetoklabel</code> 357	245
<code>\showgloLONGpluralaccess:</code>	added <code>\glsdetoklabel</code> ...	<code>\showgloUSERii:</code>	added
..... 357		<code>\glsdetoklabel</code> 245
<code>\showgloname:</code> added <code>\glsdetoklabel</code> 246	<code>\showgloUSERiii:</code>	added
		<code>\glsdetoklabel</code> 246
		<code>\showgloUSERiv:</code>	added
		<code>\glsdetoklabel</code> 246
		<code>\showgloUSERv:</code> added <code>\glsdetoklabel</code> 246
		246
		<code>\showgloUSERvi:</code>	added
		<code>\glsdetoklabel</code> 246
		dua: fixed bug in <code>\acrfullfmt</code>	. 221
		fixed bug in <code>\Acrfullplfmt</code>	. 221
		fixed bug in <code>\acrfullplfmt</code>	. 221
		redefined to use accessibility in-	
		formation	344
		dua-desc: commented spurious	
		EOL	222

redefined to use accessibility in- formation	347		
4.04			
\@gls@noidx@nosanitizesort: new	18	\compatibleglossentry: changed \newcommand to \def as is may or may not be defined	320
\@gls@noidx@sanitizesort: new	18	\compatiblesubglossentry: changed \newcommand to \def as is may or may not be defined	320
\@gls@nosanitizesort:new ..	18	\defglsdisplayfirst: fixed un- wanted space	102
\@gls@sanitizesort:new ...	18	\glo@grabfirst:new	188
\glo@addchildren:new	183	\gls@defglossaryentry: re- placed \ifx with \ifdefvoid	80
\glo@do@sortentries:new ..	183	\glsnoidxdisplayloc:new ..	191
\glo@grabfirst:new	189	\glsnoidxdisplayloclisthandler: new	191
\glo@sortedinsert:new ...	184	\glsnoidxloclist:new	190
\glo@sortentries:new	182	\glsnoidxloclisthandler: new	190
\glo@sorthandler@case:new	185	\glsnoidxstripaccents:new ..	19
\glo@sorthandler@letter: new	185	almtree: moved hangindent and parindent assignments out- side level test	298
\glo@sorthandler@nocase: new	185	\makeglossaries: Moved def- inition of \glswrite to \makeglossaries	164
\glo@sorthandler@word:new	184	\makenoidxglossaries:new ..	166
\glo@sortmacro@case:new ..	186	\printglossary: changed to use new \@printglossary ...	179
\glo@sortmacro@def:new ..	187	\printnoidxglossaries:new	179
\glo@sortmacro@def@do:new	187	\printnoidxglossary:new ..	179
\glo@sortmacro@letter:new	186	\showgloclist:new	248
\glo@sortmacro@nocase:new	186	\warn@noprintglossary: Acti- vate warning in \makeglossaries	178
\glo@sortmacro@standard: new	186	\writeist: checked for definition of \glswrite	156, 160
\glo@sortmacro@use:new ..	187	4.06	
\glo@sortmacro@word:new ..	185	\@GLS@: added \glsifhyper ..	119
\@gls@getcounterprefix: added warning if no prefix can be formed	176	\@GLSpl: added \glsifhyper ..	122
\@gls@getothergrouptitle: new	202	\@Gls@: added \glsifhyper ..	119
\@gls@noidx@do:new	189	\@Glspl@: added \glsifhyper	121
\@gls@noref@warn:new	169	\@gls@: added \glsifhyper ..	118
\@gls@reference:new	191	\@gls@numbersdef: added hook to set toc title	27
\@gls@warnonglossdefined: new	16	\@gls@symbolsdef: added hook to set toc title	27
\@gls@warnontheglossdefined: new	16	\@glsdisp: added \glsifhyper	123
\@no@makeglossaries:new ..	169	\@glspl@: added \glsifhyper	120
\@print@glossary:new	181		
\@print@noidx@glossary:new	187		
\@print@gloss@setsort:new ..	179		
\@print@glossary:new	179		
General: added sort key to print- gloss group	194		

General: added \glsifhyper ..	\@gls@field@link: added as-
..... 137-144	signment of \do@gls@link@checkfirsthyper
acronym: added hook to set toc ti- 123
tle	\@gls@forbidtexext:new 55
13	\@gls@hyp@opt:new
acronyms: added hook to set toc	104
title	\@gls@link: removed redun-
14	dancy
\glsdefmain: added hook to set	106
toc title	renamed \gls@type to
12	\glstype
4.07	106
\@glossarysection: added op-	\@glsdisp:moved \glsifhyper 123
tional argument when using	moved check for first use to
unstarred version	\@gls@link
38	123
\@gls@noidx@do: added \global	\@glspl@: moved \glsifhyper 120
in case it's used in a tabular-	moved check for first use to
like style	\@gls@link
189	120
\Acrfullplfmt: fixed no case	\@ignored@glossaries:new .. 59
change bug	General: added entrycounter op-
211	tion to printgloss family . 193
\glsletentryfield: new 145	added nopostdot option to
4.08	printgloss family
\@ACRlong: added \do@gls@link@checkfirsthyper	193
..... 339	added subentrycounter option
\@ACRshort: added \do@gls@link@checkfirsthyper	to printgloss family
..... 338	193
\@Acrlong: added \do@gls@link@checkfirsthyper	explicitly initialise hyper key . 103
..... 339	moved \glsifhyper ... 137-144
\@Acrshort: added \do@gls@link@checkfirsthyper	removed \@sACRlongpl 144
..... 338	removed \@sAcrlongpl 143
\@GLS@: moved \glsifhyper .. 119	removed \@sacrlongpl 143
moved check for first use to	removed \@sACRlong
\@gls@link	142
119	removed \@sAcrlong
\@GLSpl: moved \glsifhyper . 122	141
moved check for first use to	removed \@sacrlong
\@gls@link	141
122	removed \@sACRshortpl ... 140
\@Gls@: moved \glsifhyper .. 119	removed \@sAcrshortpl ... 139
moved check for first use to	removed \@sacrshortpl ... 139
\@gls@link	removed \@sACRshort
119	138
\@Glspl@: moved \glsifhyper 121	removed \@sAcrshort
moved check for first use to	137
\@gls@link	removed \@sgls@link
121	104
\@acrlong: added \do@gls@link@checkfirsthyper	removed \@sGLSdescplural 130
..... 338	removed \@sGlsdescplural 129
\@acrshort: added \do@gls@link@checkfirsthyper	removed \@sglsdescplural 129
..... 337	removed \@sGLSdesc
\@closegls: new	129
162	removed \@sGlsdesc
\@gls@: moved \glsifhyper .. 118	128
moved check for first use to	removed \@sglsdesc
\@gls@link	128
118	removed \@sglsdisp
\@gls@doautomake: new	123
26	removed \@sGLSfirstplural 127
	removed \@sGlsfirstplural 127
	removed \@sglsfirstplural 126
	removed \@sGLSfirst
	125
	removed \@sGlsfirst
	125

removed \@sglsfirst	124	removed sPglSpl	254
removed \@sGLSname	128	removed spglSpl	253
removed \@sGlsname	127	\ACRfull: removed \s@ACRfull	210
removed \@sglsname	127	switched to using \@gls@hyp@opt	
removed \@sGLSplural	126	210
removed \@sGlsplural	126	\Acrfull: removed \@sAcrfull	209
removed \@sglsplural	125	switched to using \@gls@hyp@opt	
removed \@sGLSpl	122	209
removed \@sGlspl	121	\acrfull: removed \@sacrfull	208
removed \@sglspl	120	switched to using \@gls@hyp@opt	
removed \@sGLSsymbolplural		208
.....	131	\ACRfullpl: removed \s@ACRfullpl	
removed \@sGLssymbolplural		211
.....	131	switched to using \@gls@hyp@opt	
removed \@sglssymbolplural		211
.....	131	\Acrfullpl: removed \s@Acrfullpl	
removed \@sGLSsymbol	130	210
removed \@sGLssymbol	130	switched to using \@gls@hyp@opt	
removed \@sglssymbol	130	210
removed \@sGLStext	124	\acrfullpl: removed \s@acrfullpl	
removed \@sGLstext	124	210
removed \@sglstext	124	switched to using \@gls@hyp@opt	
removed \@sGLSuseriii	134	210
removed \@sGlsuseriii	134	\ACRlong: switched to using	
removed \@sglsuseriii	133	\@gls@hyp@opt	142
removed \@sGLSuserii	133	\Acrlong: switched to using	
removed \@sGlsuserii	133	\@gls@hyp@opt	141
removed \@sglsuserii	133	\acrlong: switched to using	
removed \@sGLSuseriv	135	\@gls@hyp@opt	141
removed \@sGlsuseriv	135	\ACRlongpl: switched to using	
removed \@sglsuseriv	134	\@gls@hyp@opt	144
removed \@sGLSuseri	132	\Acrlongpl: switched to using	
removed \@sGlsuseri	132	\@gls@hyp@opt	143
removed \@sglsuseri	132	\acrlongpl: switched to using	
removed \@sGLSuservi	137	\@gls@hyp@opt	142
removed \@sGlsuservi	136	\ACRshort: switched to using	
removed \@sglsuservi	136	\@gls@hyp@opt	138
removed \@sGLSuserv	136	\Acrshort: switched to using	
removed \@sGlsuserv	136	\@gls@hyp@opt	138
removed \@sglsuserv	135	\acrshort: switched to using	
removed \@sGLS	119	\@gls@hyp@opt	137
removed \@sGls	118	\ACRshortpl: switched to using	
removed \@sgls	117	\@gls@hyp@opt	140
removed \@thirdofthree (de-		\Acrshortpl: switched to using	
defined in kernel)	117	\@gls@hyp@opt	139
removed sPGLS	255	\acrshortpl: switched to using	
removed sPglS	254	\@gls@hyp@opt	139
removed spglS	252	\forallacronyms: new	49
removed sPGLSpl	256		

<code>\GLS:</code> switched to using <code>\@gls@hyp@opt</code>	119	<code>\glsifhyper:</code> deprecated	104
<code>\Gls:</code> switched to using <code>\@gls@hyp@opt</code>	118	<code>\glslink:</code> switched to using <code>\@gls@hyp@opt</code>	104
<code>\gls:</code> switched to using <code>\@gls@hyp@opt</code>	117	<code>\glslinkcheckfirsthyperhook:</code> new	106
<code>\gls@defglossaryentry:</code> added check for ignored glossary ...	77	<code>\glslinkvar:</code> new	104
<code>\gls@istfilebase:</code> new	34	<code>\GLSname:</code> switched to using <code>\@gls@hyp@opt</code>	128
<code>\glsaddkey:</code> removed <code>\@sGLS@user@⟨key⟩</code>	71	<code>\Glsname:</code> switched to using <code>\@gls@hyp@opt</code>	127
removed <code>\@sGLS@user@⟨key⟩</code> .	71	<code>\glsname:</code> switched to using <code>\@gls@hyp@opt</code>	127
removed <code>\@sgls@user@⟨key⟩</code> .	70	<code>\GLSpl:</code> switched to using <code>\@gls@hyp@opt</code>	122
switched to using <code>\@gls@hyp@opt</code>	70, 71	<code>\Glspl:</code> switched to using <code>\@gls@hyp@opt</code>	121
<code>\GLSdesc:</code> switched to using <code>\@gls@hyp@opt</code>	129	<code>\glspl:</code> switched to using <code>\@gls@hyp@opt</code>	120
<code>\Glsdesc:</code> switched to using <code>\@gls@hyp@opt</code>	128	<code>\GLSplural:</code> switched to using <code>\@gls@hyp@opt</code>	126
<code>\glsdesc:</code> switched to using <code>\@gls@hyp@opt</code>	128	<code>\Glsplural:</code> switched to using <code>\@gls@hyp@opt</code>	126
<code>\GLSdescplural:</code> switched to using <code>\@gls@hyp@opt</code>	129	<code>\glsplural:</code> switched to using <code>\@gls@hyp@opt</code>	125
<code>\Glsdescplural:</code> switched to using <code>\@gls@hyp@opt</code>	129	<code>\glsspace:</code> new	209
<code>\glsdescplural:</code> switched to using <code>\@gls@hyp@opt</code>	129	<code>\GLSsymbol:</code> switched to using <code>\@gls@hyp@opt</code>	130
<code>\glsdisablehyper:</code> added <code>\KV@glslink@hyperfalse</code> to definition	117	<code>\Glsymbol:</code> switched to using <code>\@gls@hyp@opt</code>	130
<code>\glsdisp:</code> switched to using <code>\@gls@hyp@opt</code>	122	<code>\glssymbol:</code> switched to using <code>\@gls@hyp@opt</code>	130
<code>\glsdohyperlink:</code> new	116	<code>\GLSsymbolplural:</code> switched to using <code>\@gls@hyp@opt</code>	131
<code>\glsdohypertarget:</code> new	116	<code>\Glsymbolplural:</code> switched to using <code>\@gls@hyp@opt</code>	131
<code>\glsenablehyper:</code> added <code>\KV@glslink@hypertrue</code> to definition	117	<code>\glssymbolplural:</code> switched to using <code>\@gls@hyp@opt</code>	131
<code>\GLSfirst:</code> switched to using <code>\@gls@hyp@opt</code>	125	<code>\GLStext:</code> switched to using <code>\@gls@hyp@opt</code>	124
<code>\Glsfirst:</code> switched to using <code>\@gls@hyp@opt</code>	125	<code>\Glstext:</code> switched to using <code>\@gls@hyp@opt</code>	124
<code>\glsfirst:</code> switched to using <code>\@gls@hyp@opt</code>	124	<code>\glstext:</code> switched to using <code>\@gls@hyp@opt</code>	123
<code>\GLSfirstplural:</code> switched to using <code>\@gls@hyp@opt</code>	127	<code>\glstreenamefmt:</code> new	292
<code>\Glsfirstplural:</code> switched to using <code>\@gls@hyp@opt</code>	126	<code>\GLSuseri:</code> switched to using <code>\@gls@hyp@opt</code>	132
<code>\glsfirstplural:</code> switched to using <code>\@gls@hyp@opt</code>	126	<code>\Glsuseri:</code> switched to using <code>\@gls@hyp@opt</code>	132

\glsuseri: switched to using \@gls@hyp@opt	132	\PGLSpl: changed to use \@gls@hyp@opt	256
\GLSuserii: switched to using \@gls@hyp@opt	133	\Pglspl: changed to use \@gls@hyp@opt	254
\Glsuserii: switched to using \@gls@hyp@opt	133	\pglspl: changed to use \@gls@hyp@opt	253
\glsuserii: switched to using \@gls@hyp@opt	132	\s@gls@hyp@opt:new	104
\GLSuseriii: switched to using \@gls@hyp@opt	134	\s@newglossary:new	56
\Glsuseriii: switched to using \@gls@hyp@opt	134	automake:new	26
\glsuseriii: switched to using \@gls@hyp@opt	133	4.09	
\GLSuseriv: switched to using \@gls@hyp@opt	135	\glsaddkey: fixed bug in user commands	70
\Glsuseriv: switched to using \@gls@hyp@opt	135	4.10	
\glsuseriv: switched to using \@gls@hyp@opt	134	\@Gls@acentryname:new ...	146
\GLSuserv: switched to using \@gls@hyp@opt	136	\@Gls@entryname:new	146
\Glsuserv: switched to using \@gls@hyp@opt	135	\@gls@glossary: Renamed \@glossary to \@gls@glossary	171
\glsuserv: switched to using \@gls@hyp@opt	134	\glspercentchar:new	155
\GLSuserw: switched to using \@gls@hyp@opt	136	\glstildechar:new	155
\Glsuserw: switched to using \@gls@hyp@opt	135	alltree: moved space after sym- bol	298, 299
\glsuserw: switched to using \@gls@hyp@opt	135	4.11	
\GLSuserx: switched to using \@gls@hyp@opt	137	\@@do@wrglossary: added hook	174
\Glsuserx: switched to using \@gls@hyp@opt	136	sanitize: none option	21
\glsuserx: switched to using \@gls@hyp@opt	136	\gls@wrglossary: renamed from \@wrglossary to \gls@wrglossary	171
\ifignoredglossary:new	59	\glsaddprotectedpagefmt: new	173
altragged4col: fixed bug that displayed description in- stead of symbol	274	\glsbackslash:new	155
\newglossary: added starred ver- sion	56	4.12	
\newignoredglossary:new ...	58	\@gls@addpredefinedattributes: Added glsignore attribute ...	43
\ns@newglossary: added \@glotype@<name>@log ...	56	\@gls@adjustmode:new	153
new	56	\@gls@notranslatorhook: re- moved	21
\p@gls@hyp@opt:new	104	\@gls@toc: added \protect to \numberline	40
\PGLS: changed to use \@gls@hyp@opt	255	\@gls@usetranslator:new ...	21
\PglS: changed to use \@gls@hyp@opt	254	\glsacrpluralsuffix:new ...	31
\pglS: changed to use \@gls@hyp@opt	252	\glsadd: added check for vertical mode	153
		\glsaddallunused: replaced @gobble with glsignore	154
		\glsifusedtranslatordict: new	22
		\glsignore:new	154
		\glsupacrpluralsuffix:new .	31

\ProvidesGlossariesLang: new	31	\glsunset: switched to \@glsunset	84
\RequireGlossariesLang: new	31	4.15	
4.13		General: bug fix replaced	
\indexspace: new ...	261, 276, 292	\@glo@type with \glstype	144
4.14		4.16	
\@@glslocalreset: new	85	\@ACRlong: added \glspostlinkhook	340
\@@glslocalunset: new	85	\@ACRshort: added \glspostlinkhook	338
\@@glsreset: new	85	\@Acrlong: added \glspostlinkhook	339
\@@glsunset: new	85	\@Acrshort: added \glspostlinkhook	338
\@newglossaryentry@defcounters: new	87	\@GLS@: added \glspostlinkhook	120
\@cGls: new	90	\@GLSpl: added \glspostlinkhook	122
\@cGls@: new	90	\@Gls@: added \glspostlinkhook	119
\@cGlspl@: new	91	\@Glspl@: added \glspostlinkhook	122
\@cgls: new	89	\@acrlong: added \glspostlinkhook	339
\@cgls@: new	90	\@acrshort: added \glspostlinkhook	338
\@cglspl: new	90, 91	\@gls@: added \glspostlinkhook	118
\@cglspl@: new	91	\@gls@@link: added \glspostlinkhook	105
\@gls@entry@count: new	89	\@gls@field@link: added \glspostlinkhook	123
\@gls@increment@currcount: new	89	\@gls@link: moved definition of \glsifhyperon outside of this macro	107
\@gls@local@increment@currcount: new	89	\@glsdisp: added \glspostlinkhook	123
\@gls@write@entrycounts: new	89	\@glspl@: added \glspostlinkhook	121
\@glslocalreset: new	85	General: added \glspostlinkhook	137-144
\@glslocalunset: new	85	\glsacspace: new	216
\@glsreset: new	85	\glsadd: changed \@do@wrglossary to \@do@wrglossary	153
\@glsunset: new	85	\glsaddstoragekey: new	68
\@newglossaryentry@defcounters: new	81	\glsfielddef: new	73
\cGls: new	90	\glsfieldedef: new	72
\cgls: new	89	\glsfieldfetch: new	73
\cGlsformat: new	90	\glsfieldgdef: new	72
\cglsformat: new	90		
\cGlspl: new	91		
\cglspl: new	90		
\cGlsplformat: new	91		
\cglsplformat: new	91		
\gls@defdocnewglossaryentry: new	66		
\glsenableentrycount: new ..	87		
\glslocalreset: switched to \@glslocalreset	84		
\glslocalunset: switched to \@glslocalunset	84		
\glsreset: switched to \@glsreset	84		

<code>\glsfieldxdef:new</code>	72	<code>\ifglsfielddefeq:new</code>	74
<code>\glsifhyperon</code> : moved defini-		<code>\ifglsfieldeq:new</code>	73
tion of <code>\glsifhyperon</code>	106	<code>long-sp-short:new</code>	215
<code>\glslinkpostsetkeys:new</code> ..	106	<code>\showglofield:new</code>	248
<code>\glspostlinkhook:new</code>	105	4.18	
<code>\glswriteentry:new</code>	172	General: split <code>mfirstuc</code> into sepa-	
<code>\ifglsfieldcseq:new</code>	74	rate bundle	4

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	
<code>\@do@wrglossary</code>	174
<code>\@do@wrglossary</code>	173
<code>\@glo@assign@sortkey</code>	194
<code>\@glossarysec</code>	5
<code>\@glossaryseclabel</code>	6
<code>\@glossarysecstar</code>	6
<code>\@gls@default@entryfmt</code>	328
<code>\@gls@expand@field</code>	64
<code>\@gls@fixbraces</code>	177
<code>\@gls@noidx@nosanitizesort</code> .	18
<code>\@gls@noidx@sanitizesort</code> ...	18
<code>\@gls@nosanitizesort</code>	18
<code>\@gls@sanitizesort</code>	18
<code>\@glslocalreset</code>	85
<code>\@glslocalunset</code>	85
<code>\@glsreset</code>	85
<code>\@glsunset</code>	85
<code>\@newglossaryentry@defcounters</code>	87
<code>\@ACRlong</code>	339
<code>\@ACRshort</code>	338
<code>\@Acrlong</code>	339
<code>\@Acrshort</code>	338
<code>\@GLS@</code>	119
<code>\@GLSpl</code>	122
<code>\@Gls@</code>	118
<code>\@Gls@acrentryname</code>	146
<code>\@Gls@entry@field</code>	145
<code>\@Gls@entryname</code>	146
<code>\@Glspl@</code>	121
<code>\@PGLS</code>	255
<code>\@PGLS@</code>	255
<code>\@PGLSpl</code>	256
<code>\@PGLSpl@</code>	256
<code>\@Pgls</code>	254
<code>\@Pgls@</code>	254
<code>\@Pglspl</code>	254
<code>\@Pglspl@</code>	255
<code>\@acrfull</code>	209
<code>\@acrlong</code>	338
<code>\@acrshort</code>	337
<code>\@addtoacronymlists</code>	14
<code>\@cGls</code>	90
<code>\@cGls@</code>	90
<code>\@cGlspl@</code>	91
<code>\@cgls</code>	89
<code>\@cgls@</code>	90
<code>\@cglspl</code>	90, 91
<code>\@cglspl@</code>	91
<code>\@closegls</code>	162
<code>\@delimN</code>	205
<code>\@delimR</code>	205
<code>\@disable@onlypremakeg</code>	30
<code>\@disable@premakecs</code>	30
<code>\@disabled@glsaddxdycounters</code>	41
<code>\@do@seeglossary</code>	176
<code>\@do@wrglossary</code>	172, 301
<code>\@glo@addchildren</code>	183
<code>\@glo@default@sorttype</code>	10
<code>\@glo@do@sortentries</code>	183
<code>\@glo@grabfirst</code>	189
<code>\@glo@no@assign@sortkey</code>	194
<code>\@glo@seeautonumberlist</code>	7
<code>\@glo@sortedinsert</code>	184
<code>\@glo@sortentries</code>	182
<code>\@glo@sorthandler@case</code>	185
<code>\@glo@sorthandler@letter</code> ...	185
<code>\@glo@sorthandler@nocase</code> ...	185
<code>\@glo@sorthandler@word</code>	184
<code>\@glo@sortmacro@case</code>	186
<code>\@glo@sortmacro@def</code>	187
<code>\@glo@sortmacro@def@do</code>	187
<code>\@glo@sortmacro@letter</code>	186
<code>\@glo@sortmacro@nocase</code>	186
<code>\@glo@sortmacro@standard</code> ...	186
<code>\@glo@sortmacro@use</code>	187
<code>\@glo@sortmacro@word</code>	185
<code>\@glo@storeentry</code>	82
<code>\@glo@types</code>	55
<code>\@glossary@default@style</code>	7
<code>\@glossaryentryfield</code>	82
<code>\@glossarysection</code>	38
<code>\@glossarysubentryfield</code>	82
<code>\@gls</code>	117
<code>\@gls@</code>	118

<code>\@gls@link</code>	105	<code>\@gls@missingnumberlist</code>	63
<code>\@gls@access@display</code>	326	<code>\@gls@noaccess</code>	323
<code>\@gls@addpredefinedattributes</code>	43	<code>\@gls@noexpand@field</code>	64
<code>\@gls@adjustmode</code>	153	<code>\@gls@noexpand@fields</code>	64
<code>\@gls@assign@symbol@field</code> ...	18	<code>\@gls@nohyperlist</code>	16
<code>\@gls@assign@symbolplural@field</code>	18	<code>\@gls@noidx@do</code>	189
<code>\@gls@checkactual</code>	114	<code>\@gls@noidx@setsanitizesort</code> .	21
<code>\@gls@checkbar</code>	113	<code>\@gls@noref@warn</code>	169
<code>\@gls@checkesactual</code>	111	<code>\@gls@notranslatorhook</code>	21
<code>\@gls@checkeschar</code>	112	<code>\@gls@numbersdef</code>	27
<code>\@gls@checkescllevel</code>	113	<code>\@gls@onlypremakeg</code>	29
<code>\@gls@checkesquote</code>	111	<code>\@gls@provide@newglossary</code> ...	55
<code>\@gls@checklevel</code>	114	<code>\@gls@reference</code>	191
<code>\@gls@checkmkidxchars</code>	110	<code>\@gls@renewglossary</code>	171
<code>\@gls@checkquote</code>	110	<code>\@gls@sanitizedesc</code>	17
<code>\@gls@codepage</code>	48	<code>\@gls@sanitizename</code>	17
<code>\@gls@counterwithin</code>	9	<code>\@gls@sanitizesort</code>	18
<code>\@gls@declareoption</code>	7	<code>\@gls@sanitizesymbol</code>	18
<code>\@gls@default@value</code>	60	<code>\@gls@saveentrycounter</code>	107
<code>\@gls@do@acronymsdef</code>	14	<code>\@gls@setacrstyle</code>	23
<code>\@gls@doautomake</code>	26	<code>\@gls@setcounter</code>	57
<code>\@gls@entry@count</code>	89	<code>\@gls@setupsort@def</code>	11
<code>\@gls@entry@field</code>	145	<code>\@gls@setupsort@standard</code>	10
<code>\@gls@escbsdq</code>	109	<code>\@gls@setupsort@use</code>	11
<code>\@gls@expand@fields</code>	64	<code>\@gls@startswithexpandonce</code> ..	65
<code>\@gls@fetchfield</code>	68	<code>\@gls@symbolsdef</code>	27
<code>\@gls@field@link</code>	123	<code>\@gls@tmpb</code>	110
<code>\@gls@fixbraces</code>	176	<code>\@gls@toc</code>	40
<code>\@gls@forbidtext</code>	55	<code>\@gls@updatechecked</code>	110
<code>\@gls@getcounter</code>	58	<code>\@gls@usetranslator</code>	21
<code>\@gls@getcounterprefix</code>	175	<code>\@gls@warnonglossdefined</code>	16
<code>\@gls@getgrouptitle</code>	201	<code>\@gls@warnontheGLOSSdefined</code> .	16
<code>\@gls@getothergrouptitle</code> ...	202	<code>\@gls@write@entrycounts</code>	89
<code>\@gls@glossary</code>	171	<code>\@gls@writedef</code>	67
<code>\@gls@hyp@opt</code>	104	<code>\@gls@xdy@Lclass@Alpha-page-numbers</code>	45
<code>\@gls@hypergroup</code>	257	<code>\@gls@xdy@Lclass@Appendix-page-numbers</code>	45
<code>\@gls@ifinlist</code>	41	<code>\@gls@xdy@Lclass@Roman-page-numbers</code>	44
<code>\@gls@increment@currcount</code> ...	89	<code>\@gls@xdy@Lclass@alpha-page-numbers</code>	45
<code>\@gls@indexdef</code>	28	<code>\@gls@xdy@Lclass@arabic-page-numbers</code>	45
<code>\@gls@keymap</code>	67, 250	<code>\@gls@xdy@Lclass@arabic-section-numbers</code>	45
<code>\@gls@link</code>	106	<code>\@gls@xdy@Lclass@roman-page-numbers</code>	44
<code>\@gls@loadlist</code>	8	<code>\@gls@xdy@locationlist</code>	44
<code>\@gls@loadlong</code>	8		
<code>\@gls@loadsUPER</code>	8		
<code>\@gls@loadtree</code>	8		
<code>\@gls@local@increment@currcount</code>	89		

<code>\@gls@xdycheckbackslash</code>	115
<code>\@gls@xdycheckquote</code>	115
<code>\@glsAlphacompositor</code>	35, 45
<code>\@glsacronymlists</code>	14
<code>\@glsdefaultplural</code>	63
<code>\@glsdefaultsort</code>	63
<code>\@glsdisp</code>	123
<code>\@glsfirstletter</code>	155
<code>\@glshypernumber</code>	205
<code>\@glslink</code>	116
<code>\@glslocalreset</code>	85
<code>\@glslocalunset</code>	85
<code>\@glsminrange</code>	155
<code>\@glsnextpages</code>	195
<code>\@glsnodesc</code>	63
<code>\@glsnoname</code>	63
<code>\@glsnonextpages</code>	194
<code>\@glsopenfile</code>	162
<code>\@glsorder</code>	24
<code>\@glspl@</code>	120
<code>\@glsreset</code>	85
<code>\@glstarget</code>	116
<code>\@glsunset</code>	85
<code>\@glswidestname</code>	297
<code>\@glswritefiles</code>	170
<code>\@ignored@glossaries</code>	59
<code>\@istfilename</code>	34
<code>\@makeglossary</code>	162
<code>\@newglossary</code>	57
<code>\@newglossaryentry@defcounters</code>	81
<code>\@newglossaryentryposthook</code>	81
<code>\@newglossaryentryprehook</code>	81
<code>\@no@makeglossaries</code>	169
<code>\@no@post@desc</code>	33
<code>\@nopostdesc</code>	33
<code>\@onlypremakeg</code>	29
<code>\@p@glossarysection</code>	39
<code>\@pgls</code>	253
<code>\@pgls@</code>	253
<code>\@pglspl</code>	253
<code>\@pglspl@</code>	253
<code>\@print@glossary</code>	181
<code>\@print@noidx@glossary</code>	187
<code>\@printgloss@setsort</code>	179
<code>\@printglossary</code>	179
<code>\@set@glo@numformat</code>	108, 302
<code>\@wrglossary@pageformat</code>	173
<code>\@wrglossarynumberhook</code>	173

<code>\@xdy@main@language</code>	25
<code>\@xdyattributelist</code>	40
<code>\@xdyattributes</code>	40
<code>\@xdy@language</code>	48
<code>\@xdy@lettergroups</code>	49
<code>\@xdy@locationclassorder</code>	46
<code>\@xdy@locref</code>	41
<code>\@xdy@requiredstyles</code>	47
<code>\@xdy@sortrules</code>	47
<code>\@xdy@useralphabets</code>	43
<code>\@xdy@userlocationdefs</code>	45
<code>\@xdy@userlocationnames</code>	45

A

<code>\Ac</code>	226
<code>\ac</code>	226
<code>access (key)</code>	321
<code>accsupp package</code>	320
<code>\accsuppglossaryentryfield</code>	341
<code>\accsuppglossarysubentryfield</code>	341
<code>\Acf</code>	225
<code>\acf</code>	225
<code>\Acfp</code>	226
<code>\acfp</code>	225
<code>\Acl</code>	225
<code>\acl</code>	225
<code>\Aclp</code>	225
<code>\aclp</code>	225
<code>\Acp</code>	226
<code>\acp</code>	226
<code>\acrfootnote</code>	228
<code>\ACRfull</code>	210
<code>\Acrfull</code>	209
<code>\acrfull</code>	208, 209, 213, 222, 348
<code>\ACRfullfmt</code>	210
<code>\Acrfullfmt</code>	209
<code>\acrfullfmt</code>	209
<code>\acrfullformat</code>	98, 209
<code>\ACRfullpl</code>	211
<code>\Acrfullpl</code>	210
<code>\acrfullpl</code>	210
<code>\ACRfullplfmt</code>	211
<code>\Acrfullplfmt</code>	211
<code>\acrfullplfmt</code>	210
<code>\acrlinkfootnote</code>	228
<code>\acrlinkfullformat</code>	209
<code>\ACRlong</code>	142
<code>\Acrlong</code>	141

<code>\acrlong</code>	141	<code>\addglossarytocaptions</code>	31
<code>\ACRlongpl</code>	144	<code>\addto</code>	32
<code>\Acrlongpl</code>	143	<code>align</code> (environment)	83, 107
<code>\acrlongpl</code>	142	<code>altlist</code> (style)	262
<code>\acrnameformat</code>	211, 234	<code>altlistgroup</code> (style)	263
<code>\acrnolinkfootnote</code>	228	<code>altlisthypergroup</code> (style)	263
<code>acronym</code> (option)	13	<code>altlong4col</code> (style)	269
acronym styles:		<code>altlong4colborder</code> (style)	270
<code>dua</code>	220, 344	<code>altlong4colheader</code> (style)	269
<code>dua-desc</code>	222, 347	<code>altlong4colheaderborder</code> (style)	270
<code>footnote</code>	222, 347	<code>altlongragged4col</code> (style)	274
<code>footnote-desc</code>	224, 349	<code>altlongragged4colborder</code> (style)	275
<code>footnote-sc</code>	223, 349	<code>altlongragged4colheader</code> (style)	274
<code>footnote-sc-desc</code>	224, 350	<code>altlongragged4colheaderborder</code> (style)	275
<code>footnote-sm</code>	223, 349	<code>\altnewglossary</code>	57
<code>footnote-sm-desc</code>	224, 350	<code>altsuper4col</code> (style)	285
<code>long-sc-short</code>	217	<code>altsuper4colborder</code> (style)	286
<code>long-sc-short-desc</code> ..	218, 343	<code>altsuper4colheader</code> (style)	285
<code>long-short</code>	215, 341	<code>altsuper4colheaderborder</code> (style)	286
<code>long-short-desc</code>	218, 343	<code>altsuperragged4col</code> (style)	290
<code>long-sm-short</code>	217	<code>altsuperragged4colborder</code> (style)	291
<code>long-sm-short-desc</code> ..	218, 343	<code>altsuperragged4colheader</code> (style)	291
<code>long-sp-short</code>	215	<code>altsuperragged4colheaderborder</code> (style)	292
<code>long-sp-short-desc</code>	218	<code>almtree</code> (style)	297
<code>sc-short-long</code>	217	<code>almtreegroup</code> (style)	299
<code>sc-short-long-desc</code> ..	219, 344	<code>almtreehypergroup</code> (style)	300
<code>short-long</code>	216, 342	<code>amsgen</code> package	4, 103
<code>short-long-desc</code>	219, 344	<code>amsmath</code> package	83
<code>sm-short-long</code>	217	<code>\andname</code>	31
<code>sm-short-long-desc</code> ..	219, 344	<code>array</code> package	270, 286
<code>\acronymentry</code>	213	<code>article</code> class	175
<code>\acronymfont</code>		<code>automake</code> (option)	26
.....	98, 211, 230, 234, 236, 239		
<code>acronymlists</code> (option)	15		
<code>\acronymname</code>	30		
<code>acronyms</code> (option)	14		
<code>\acronymsort</code>	213		
<code>\acronymtype</code>	13, 208		
<code>\acrpluralsuffix</code>	208		
<code>\ACRshort</code>	138		
<code>\Acrshort</code>	138		
<code>\acrshort</code>	137		
<code>\ACRshortpl</code>	140		
<code>\Acrshortpl</code>	139		
<code>\acrshortpl</code>	139		
<code>\Acs</code>	225		
<code>\acs</code>	225		
<code>\Acsp</code>	225		
<code>\acsp</code>	225		
		B	
		<code>babel</code> package	21, 30, 32, 48
		C	
		<code>\cGls</code>	90
		<code>\cgls</code>	89
		<code>\cGlsformat</code>	90
		<code>\cglsformat</code>	90
		<code>\cGlspl</code>	91
		<code>\cglspl</code>	90
		<code>\cGlsplformat</code>	91
		<code>\cglsplformat</code>	91

`\compatglossarystyle` 307
`compatible-2.07 (option)` 26
`compatible-3.07 (option)` 26
`\compatibleglossentry` .. 198, 320
`\compatiblesubglossentry` 199, 320
`counter (key)` 61
`counter (option)` 16
`\CustomAcronymFields` 242
`\CustomNewAcronymDef` 242

D

`datatool package` 184
`\DeclareAcronymList` 14
`\DefaultNewAcronymDef` .. 226, 350
`\defentryfmt` 102
`\defgldisplay` 101
`\defgldisplayfirst` 102
`\defglentry` 57
`\defglentryfmt` 55, 59, 60, 93
`\DefineAcronymSynonyms` 224
`\delimN` 36, 204
`\delimR` 36, 204
`description (environment)` 261
`description (key)` 59
`description (option)` 24
`descriptionaccess (key)` 322
`\DescriptionDUANewAcronymDef` 230
`\DescriptionFootnoteNewAcronymDef`
..... 229, 352
`\descriptionname` 30
`\DescriptionNewAcronymDef` ..
..... 233, 352
`descriptionplural (key)` 59
`descriptionpluralaccess (key)` 322
`\detokenize` 50
`doc package` 4, 5, 12
`document (environment)` 66, 87
`dua (acrstyle)` 220, 344
`dua (option)` 24
`dua-desc (acrstyle)` 222, 347
`\DUANewAcronymDef` 240

E

`entrycounter (option)` 9
`entrycounterwithin (option)` 9
`\entryname` 30
environments:
 `align` 83, 107
 `description` 261
 `document` 66, 87

`longtable` 8, 243, 264–275
`multicols` 276
`supertabular` ... 8, 243, 280–292
`theglossary` 5, 16, 36, 37, 197,
198, 203, 277, 278, 294, 295, 297
`theindex` 293
`equation (counter)` 107, 108
`etoolbox package` 4

F

file types
 `.aux` 181
 `.glo` 82
 `.ist` 154, 161
 `.toc` 40
 `.xdy` 34
 `glo` 249
`\findrootlanguage` 47
`first (key)` 60
`firstaccess (key)` 321
`\firstacronymfont` 98, 211
`firstplural (key)` 60
`firstpluralaccess (key)` 321
`footnote (acrstyle)` 222, 347
`footnote (option)` 23
`footnote-desc (acrstyle)` .. 224, 349
`footnote-sc (acrstyle)` 223, 349
`footnote-sc-desc (acrstyle)` 224, 350
`footnote-sm (acrstyle)` 223, 349
`footnote-sm-desc (acrstyle)` 224, 350
`\FootnoteNewAcronymDef` . 235, 353
`\forallacronyms` 49
`\forallglossaries` 49
`\forallglentries` 50
`\forallglentries` 49

G

`garamondx package` 208
`\Genacrfullformat` 101, 337
`\genacrfullformat` 100, 337
`\GenericAcronymFields` 213
`\Genplacrfullformat` 101, 337
`\genplacrfullformat` 101, 337
`\glo@grabfirst` 188
`\glolinkprefix` 107
`glossareentry (counter)` 196
`glossaries package` 28,
48, 155, 243, 250, 261, 300, 320
`glossaries-accsupp package` ... 82, 320
`\GlossariesWarning` 16

<code>\GlossariesWarningNoLine</code>	16	<code>altlistgroup</code>	263
<code>\glossary</code>	56, 161, 171, 202	<code>altlisthypergroup</code>	263, 308
glossary counters:		<code>altlisthypergroup</code>	263
<code>glossaryentry</code>	195	<code>altlong4col</code>	269, 270, 274, 310
<code>glossarysubentry</code>	195	<code>altlong4col</code>	269
glossary keys:		<code>altlong4colborder</code>	270, 310
<code>access</code>	321	<code>altlong4colborder</code>	270
<code>counter</code>	61	<code>altlong4colheader</code>	269, 310
<code>description</code>	59	<code>altlong4colheader</code>	269
<code>descriptionaccess</code>	322	<code>altlong4colheaderborder</code>	270, 311
<code>descriptionplural</code>	59	<code>altlong4colheaderborder</code>	270
<code>descriptionpluralaccess</code>	322	<code>altlongragged4col</code>	274, 275, 312
<code>first</code>	60	<code>altlongragged4col</code>	274
<code>firstaccess</code>	321	<code>altlongragged4colborder</code>	275, 312
<code>firstplural</code>	60	<code>altlongragged4colborder</code>	275
<code>firstpluralaccess</code>	321	<code>altlongragged4colheader</code>	274, 312
<code>long</code>	63	<code>altlongragged4colheader</code>	274
<code>longaccess</code>	322	<code>altlongragged4colheaderborder</code>	275, 312
<code>longplural</code>	63	<code>altlongragged4colheaderborder</code>	275
<code>longpluralaccess</code>	322	<code>altsuper4col</code>	285, 286, 290, 319
<code>name</code>	59	<code>altsuper4col</code>	285
<code>nonumberlist</code>	61	<code>altsuper4colborder</code>	286, 320
<code>parent</code>	61	<code>altsuper4colborder</code>	286
<code>plural</code>	60	<code>altsuper4colheader</code>	285, 320
<code>pluralaccess</code>	321	<code>altsuper4colheader</code>	285
<code>see</code>	61	<code>altsuper4colheaderborder</code>	286, 320
<code>short</code>	62	<code>altsuper4colheaderborder</code>	286
<code>shortaccess</code>	322	<code>altsuperragged4col</code>	290–292, 318
<code>shortplural</code>	62	<code>altsuperragged4col</code>	290
<code>shortpluralaccess</code>	322	<code>altsuperragged4colborder</code>	291, 318
<code>sort</code>	59	<code>altsuperragged4colborder</code>	291
<code>symbol</code>	60	<code>altsuperragged4colheader</code>	291, 318
<code>symbolaccess</code>	322	<code>altsuperragged4colheader</code>	291
<code>symbolplural</code>	60	<code>altsuperragged4colheaderborder</code>	292, 318
<code>symbolpluralaccess</code>	322	<code>altsuperragged4colheaderborder</code>	292
<code>text</code>	60	<code>alttree</code>	279, 297, 299, 314
<code>textaccess</code>	321	<code>alttree</code>	297
<code>type</code>	61	<code>alttreegroup</code>	300, 315
<code>user1</code>	62	<code>alttreegroup</code>	299
<code>user2</code>	62		
<code>user3</code>	62		
<code>user4</code>	62		
<code>user5</code>	62		
<code>user6</code>	62		
glossary package	1, 207		
glossary styles:			
<code>altlist</code>	262, 263, 308		
<code>altlist</code>	262		
<code>altlistgroup</code>	263, 308		

alttreehypergroup ...	300, 315	longragged3colheader	273
alttreehypergroup	300	longragged3colheaderborder	
index	276, 293, 294, 312	273, 312
index	293	longragged3colheaderborder	
indexgroup	294, 313	273
indexgroup	294	longraggedborder	271, 311
indexhypergroup	294, 313	longraggedborder	271
indexhypergroup	294	longraggedheader	272, 311
inline	307	longraggedheader	272
inline	259	longraggedheaderborder	272, 311
list	6, 261–263, 308	longraggedheaderborder ..	272
list	261	mcolalttree	279, 316
listdotted	263, 264, 308	mcolalttree	279
listdotted	263	mcolalttreegroup	279, 316
listgroup	262, 308	mcolalttreegroup	279
listgroup	262	mcolalttreehypergroup	279, 316
listhypergroup	262, 308	mcolalttreehypergroup ...	279
listhypergroup	262	mcolindex	276, 315
long	265, 266, 271, 309, 311	mcolindex	276
long	265	mcolindexgroup	276, 316
long3col	266, 267, 309	mcolindexgroup	276
long3col	266	mcolindexhypergroup .	276, 316
long3colborder	267, 309	mcolindexhypergroup	276
long3colborder	267	mcoltree	277, 316
long3colheader	267, 310	mcoltree	277
long3colheader	267	mcoltreegroup	316
long3colheaderborder	267, 310	mcoltreegroup	277
long3colheaderborder	267	mcoltreehypergroup ..	277, 316
long4col	267–269, 310	mcoltreehypergroup	277
long4col	267	mcoltreenoname	278, 316
long4colborder	268, 310	mcoltreenoname	278
long4colborder	268	mcoltreenonamegroup .	278, 316
long4colheader	268, 310	mcoltreenonamegroup	278
long4colheader	268	mcoltreenonamehypergroup	
long4colheaderborder	269, 310	278, 316
long4colheaderborder	269	mcoltreenonamehypergroup	278
longborder	265, 309	sublistdotted	309
longborder	265	sublistdotted	264
longheader	266, 309	super	280–282, 288, 318
longheader	266	super	280
longheaderborder	266, 309	super3col	282, 283, 318
longheaderborder	266	super3col	282
longragged	271, 272	super3colborder	282, 319
longragged	271	super3colborder	282
longragged3col ...	272, 273, 311	super3colheader	283, 319
longragged3col	272	super3colheader	283
longragged3colborder	273, 311	super3colheaderborder	283, 319
longragged3colborder	273	super3colheaderborder ...	283
longragged3colheader	273, 311	super4col	283–285, 319

super4col	283	glossary-super package	8, 264, 280, 286, 290
super4colborder	284, 319	glossary-superragged package	286
super4colborder	284	glossary-tree package	8, 292
super4colheader	284, 319	glossaryentry (counter)	9, 196, 197
super4colheader	284	glossaryentry (counter)	195
super4colheaderborder	285, 319	\glossaryentryfield	200, 203, 204
super4colheaderborder	285	\glossaryentrynumber	195
superborder	281, 318	\glossaryentrynumbers	7, 36, 180, 181
superborder	281	\glossaryheader	198, 203
superheader	281, 318	\glossarymark	38
superheader	281	\glossaryname	30, 32
superheaderborder	281, 318	\glossarypostamble	37, 203
superheaderborder	281	\glossarypreamble	36, 203
superragged	287–289, 317	\glossarysection	6, 37, 56
superragged	287	\glossarystyle	203, 244
superragged3col	289, 290, 317	glossarysubentry (counter)	9, 196, 197
superragged3col	289	glossarysubentry (counter)	195
superragged3colborder	289, 317	\glossarysubentryfield	200
superragged3colborder	289	\glossentry	60, 198
superragged3colheader	290, 317	\Glossentrydesc	199
superragged3colheader	290	\glossentrydesc	199, 340
superragged3colheaderborder	290, 317	\Glossentryname	198
superraggedborder	288, 317	\glossentryname	198, 340
superraggedborder	288	\Glossentrysymbol	199
superraggedheader	288, 317	\glossentrysymbol	199, 340
superraggedheader	288	\GLS	119
superraggedheaderborder	288, 317	\Gls	118, 121
superraggedheaderborder	288	\gls	4, 60, 92, 103, 117, 119, 120, 123–136, 196, 252
superraggedright3colheaderborder	290	\gls@Alphpage	172
tree	277, 294–297, 313	\gls@alphpage	172
tree	294	\gls@assign@desc	75
treegroup	277, 295, 313	\gls@assign@desc@field	17
treegroup	295	\gls@assign@desc@plural@field	17
treehypergroup	295, 313	\gls@assign@field	65
treehypergroup	295	\gls@assign@name@field	18
treenoname	278, 296, 314	\gls@assign@type@field	17
treenoname	296	\gls@checkisacronymlist	15
treenonamegroup	297, 314	\gls@checkseeallowed	61
treenonamegroup	296	\gls@codepage	25
treenonamehypergroup	297, 314	\gls@defdocnewglossaryentry	66
treenonamehypergroup	297	\gls@defglossaryentry	75
glossary-hypernav package	155	\gls@disablepagerefexpansion	172
glossary-list package	6, 8, 261	\gls@docclearpage	39
glossary-long package	8, 264, 274, 280	\gls@doentryfmt	55
glossary-longragged package	270	\gls@glossary	171
glossary-mcols package	276		

<code>\gls@hypergroup</code>	257	<code>\glsdescplural</code>	129
<code>\gls@ifnotmeasuring</code>	83	<code>\glsdescriptionaccessdisplay</code>	327
<code>\gls@istfilebase</code>	34	<code>\glsdescriptionpluralaccessdisplay</code>	327
<code>\gls@level</code>	64	327
<code>\gls@noidxglossary</code>	169	<code>\glsdescwidth</code>	264, 270, 280, 287
<code>\gls@numberpage</code>	173	<code>\glsdetoklabel</code>	50
<code>\gls@protected@pagefmts</code>	172	<code>\glsdisablehyper</code>	117
<code>\gls@Romanpage</code>	173	<code>\glsdisp</code>	122
<code>\gls@romanpage</code>	173	<code>\glsdisplay</code>	92, 101, 117
<code>\gls@save@numberlist</code>	178	<code>\glsdisplayfirst</code>	92, 101, 117
<code>\gls@suffixF</code>	35	<code>\glsdisplaynumberlist</code>	151, 167, 191
<code>\gls@suffixFF</code>	35	<code>\glsdohyperlink</code>	116
<code>\gls@wrglossary</code>	171	<code>\glsdohypertarget</code>	116
<code>\glsaccessdisplay</code>	327	<code>\glsdoifexists</code>	51
<code>\glsaccsupp</code>	326	<code>\glsdoifexistsorwarn</code>	51
<code>\glsacrpluralsuffix</code>	31	<code>\glsdoifnoexists</code>	51
<code>\glsacspace</code>	216	<code>\glsdoparenifnotempty</code>	237
<code>\glsadd</code>	92, 153, 202	<code>\glsenableentrycount</code>	87
<code>\glsadd options</code>		<code>\glsenablehyper</code>	117
<code>counter</code>	153	<code>\glsentryaccess</code>	324
<code>format</code>	153, 204	<code>\glsentrycounter</code>	107
<code>\glsaddall</code>	50, 92, 154	<code>\glsentrycounterlabel</code>	197
<code>\glsaddall options</code>		<code>\glsentrycurrcount</code>	87
<code>types</code>	153, 154	<code>\Glsentrydesc</code>	147
<code>\glsaddallunused</code>	154	<code>\glsentrydesc</code>	146
<code>\glsaddkey</code>	69	<code>\glsentrydescaccess</code>	325
<code>\GlsAddLetterGroup</code>	49	<code>\Glsentrydescplural</code>	147
<code>\glsaddprotectedpagefmt</code>	173	<code>\glsentrydescplural</code>	147
<code>\GlsAddSortRule</code>	47	<code>\glsentrydescpluralaccess</code>	325
<code>\glsaddstoragekey</code>	68	<code>\Glsentryfirst</code>	148
<code>\GlsAddXdyAlphabet</code>	43	<code>\glsentryfirst</code>	148
<code>\GlsAddXdyAttribute</code>	42, 300	<code>\glsentryfirstaccess</code>	324
<code>\GlsAddXdyCounters</code>	41, 301	<code>\Glsentryfirstplural</code>	148
<code>\GlsAddXdyLocation</code>	45, 301	<code>\glsentryfirstplural</code>	148
<code>\GlsAddXdyStyle</code>	47	<code>\glsentryfirstpluralaccess</code>	325
<code>\glsautoprefix</code>	6	<code>\glsentryfmt</code>	59, 60, 92
<code>\glsbackslash</code>	155	<code>\Glsentryfull</code>	151
<code>\glsclearpage</code>	40	<code>\glsentryfull</code>	151, 213, 223, 348
<code>\glsclosebrace</code>	155	<code>\Glsentryfullpl</code>	151
<code>\glscompositor</code>	34, 45	<code>\glsentryfullpl</code>	151
<code>\glscounter</code>	15, 57	<code>\glsentryitem</code>	197
<code>\GlsDeclareNoHyperList</code>	16	<code>\Glsentrylong</code>	151
<code>\glsdefaultttype</code>	13	<code>\glsentrylong</code>	150
<code>\glsdefmain</code>	12	<code>\glsentrylongaccess</code>	325
<code>\GLSdesc</code>	129	<code>\Glsentrylongpl</code>	151
<code>\Glsdesc</code>	128	<code>\glsentrylongpl</code>	151
<code>\glsdesc</code>	128, 129	<code>\glsentrylongpluralaccess</code>	326
<code>\GLSdescplural</code>	129	<code>\Glsentryname</code>	145
<code>\Glsdescplural</code>	129	<code>\glsentryname</code>	145, 178

<code>\glsentrynumberlist</code>	151, 167	<code>\Glsfirst</code>	125
<code>\Glsentryplural</code>	147	<code>\glsfirst</code>	124, 125
<code>\glsentryplural</code>	147	<code>\glsfirstaccessdisplay</code>	326
<code>\glsentrypluralaccess</code>	325	<code>\GLSfirstplural</code>	127
<code>\Glsentryprefix</code>	252	<code>\Glsfirstplural</code>	126
<code>\glsentryprefix</code>	251	<code>\glsfirstplural</code>	126, 127
<code>\Glsentryprefixfirst</code>	251	<code>\glsfirstpluralaccessdisplay</code>	327
<code>\glsentryprefixfirst</code>	251	<code>\glsgenacfmt</code>	98, 334
<code>\Glsentryprefixfirstplural</code>	251	<code>\glsgenentryfmt</code>	96, 332
<code>\glsentryprefixfirstplural</code>	251	<code>\glsgetgrouplabel</code>	202
<code>\Glsentryprefixplural</code>	252	<code>\glsgetgrouptitle</code>	155, 201
<code>\glsentryprefixplural</code>	251	<code>\gls glossarymark</code>	9, 37
<code>\glsentryprevcount</code>	87	<code>\gls groupheading</code>	201, 203
<code>\Glsentryshort</code>	150	<code>\gls groupskip</code>	201, 203, 261
<code>\glsentryshort</code>	150	<code>\gls hyperlink</code>	153
<code>\glsentryshortaccess</code>	325	<code>\gls hypernavsep</code>	258
<code>\Glsentryshortpl</code>	150	<code>\gls hypernumber</code>	36, 204
<code>\glsentryshortpl</code>	150	<code>\gls ifhyper</code>	104
<code>\glsentryshortpluralaccess</code>	325	<code>\gls ifhyperon</code>	103, 106
<code>\glsentrysort</code>	149	<code>\gls IfListOfAcronyms</code>	14
<code>\Glsentrysymbol</code>	148	<code>\gls ifusedtranslatordict</code>	22
<code>\glsentrysymbol</code>	147	<code>\gls ignore</code>	154
<code>\glsentrysymbolaccess</code>	325	<code>\gls inlinedescformat</code>	260
<code>\Glsentrysymbolplural</code>	148	<code>\gls inlinedopostchild</code>	259, 260
<code>\glsentrysymbolplural</code>	148	<code>\gls inlineemptydescformat</code>	261
<code>\glsentrysymbolpluralaccess</code>	325	<code>\gls inlinenameformat</code>	260
<code>\Glsentrytext</code>	147	<code>\gls inlineparentchildseparator</code>	260
<code>\glsentrytext</code>	50, 147, 178	<code>\gls inlinepostchild</code>	260
<code>\glsentrytextaccess</code>	324	<code>\gls inlineseparator</code>	260
<code>\glsentrytype</code>	148	<code>\gls inlinesubdescformat</code>	261
<code>\Glsentryuseri</code>	149	<code>\gls inlinesubnameformat</code>	261
<code>\glsentryuseri</code>	149	<code>\gls inlinesubseparator</code>	260
<code>\Glsentryuserii</code>	149	<code>\gls keylisttok</code>	211
<code>\glsentryuserii</code>	149	<code>\gls labeltok</code>	212
<code>\Glsentryuseriii</code>	149	<code>\gls letentryfield</code>	145
<code>\glsentryuseriii</code>	149	<code>\gls link</code>	92, 104, 117, 153, 202, 204
<code>\Glsentryuseriv</code>	150	<code>\gls link options</code>	
<code>\glsentryuseriv</code>	149	counter	103, 117, 250
<code>\Glsentryuserv</code>	150	format	103, 117, 204
<code>\glsentryuserv</code>	150	hyper	103, 106, 117
<code>\Glsentryuservi</code>	150	local	103
<code>\glsentryuservi</code>	150	<code>\gls linkcheckfirsthyperhook</code>	106
<code>\glsexpandfields</code>	65	<code>\gls linkpostsetkeys</code>	106
<code>\glsfielddef</code>	73	<code>\gls linkvar</code>	104
<code>\glsfieldedef</code>	72	<code>\gls listdottedwidth</code>	264
<code>\glsfieldfetch</code>	73	<code>\gls localreset</code>	84
<code>\glsfieldgdef</code>	72	<code>\gls localresetall</code>	86
<code>\glsfieldxdef</code>	72	<code>\gls localunset</code>	84
<code>\GLSfirst</code>	125		

<code>\glslocalunsetall</code>	86	<code>\glspluralaccessdisplay</code>	326
<code>\glslongaccessdisplay</code>	327	<code>\glspluralsuffix</code>	31, 60
<code>\glslongaccesskey</code>	355	<code>\glspostdescription</code>	9
<code>\glslongkey</code>	208	<code>\glspostinline</code>	260
<code>\glslongpluralaccessdisplay</code>	327	<code>\glspostlinkhook</code>	105
<code>\glslongpluralaccesskey</code>	355	<code>\glsprestandardsort</code>	10
<code>\glslongpluralkey</code>	208	<code>\glsquote</code>	155
<code>\glslongtok</code>	212	<code>\glsrefentry</code>	196
<code>\glsmcols</code>	276	<code>\glsreset</code>	84
<code>\glsmoveentry</code>	81	<code>\glsresetall</code>	86
<code>\GLSname</code>	128	<code>\glsresetentrylist</code>	195
<code>\Glsname</code>	127	<code>\glsresetsubentrycounter</code>	196
<code>\glsname</code>	127, 128	<code>\glssee</code>	177
<code>\glsnameaccessdisplay</code>	326	<code>\glsseeformat</code>	157, 177
<code>\glsnamefont</code>	204, 292	<code>\glsseeitem</code>	178
<code>\glsnavhyperlink</code>	257	<code>\glsseeitemformat</code>	178
<code>\glsnavhypertarget</code>	257	<code>\glsseeelastsep</code>	177
<code>\glsnavigation</code>	258	<code>\glsseeelist</code>	177
<code>\glsnextpages</code>	195	<code>\glsseesep</code>	178
<code>\glsnoexpandfields</code>	65	<code>\glsSetAlphaCompositor</code>	35
<code>\glsnoidxdisplayloc</code>	191	<code>\glsSetCompositor</code>	34, 35
<code>\glsnoidxdisplayloclisthandler</code>	191	<code>\glssetexpandfield</code>	17
<code>\glsnoidxloclist</code>	190	<code>\glssetnoexpandfield</code>	17
<code>\glsnoidxloclisthandler</code>	190	<code>\glsSetSuffixF</code>	35
<code>\glsnoidxnumberlistloophandler</code>	169	<code>\glsSetSuffixFF</code>	35
<code>\glsnoidxstripaccents</code>	19	<code>\glssettoctitle</code>	30
<code>\glsnonextpages</code>	195	<code>\glssetwidest</code>	297
<code>\glsnoxindywarning</code>	40	<code>\GlsSetXdyCodePage</code>	48
<code>\glsnumberformat</code>	36	<code>\GlsSetXdyFirstLetterAfterDigits</code>	155
<code>\glsnumberlistloop</code>	169	<code>\GlsSetXdyLanguage</code>	48
<code>\glsnumbersgroupname</code>	31, 154, 155, 201	<code>\GlsSetXdyLocationClassOrder</code>	46
<code>\glsnumlistlastsep</code>	152	<code>\GlsSetXdyMinRangeLength</code>	155
<code>\glsnumlistsep</code>	152	<code>\GlsSetXdyStyles</code>	47
<code>\glsopenbrace</code>	155	<code>\glsshortaccessdisplay</code>	327
<code>\glsorder</code>	24	<code>\glsshortaccesskey</code>	355
<code>\glsorg@endtheglossary</code>	5	<code>\glsshortkey</code>	208
<code>\glsorg@theglossary</code>	5	<code>\glsshortpluralaccessdisplay</code>	327
<code>\glspagelistwidth</code>	265, 271, 280, 287	<code>\glsshortpluralaccesskey</code>	355
<code>\glspar</code>	34	<code>\glsshortpluralkey</code>	208
<code>\glspercentchar</code>	155	<code>\glsshorttok</code>	212
<code>\GLSpl</code>	122	<code>\glssortnumberfmt</code>	11
<code>\Glspl</code>	121	<code>\glsspace</code>	209
<code>\glspl</code>	92, 120–122	<code>\glsstepentry</code>	196
<code>\GLSplural</code>	126	<code>\glsstepsubentry</code>	196
<code>\Glsplural</code>	126	<code>\glssubentrycounterlabel</code>	197
<code>\glsplural</code>	125, 126	<code>\glssubentryitem</code>	197
		<code>\GLSsymbol</code>	130
		<code>\Glsymbol</code>	130

<code>\glssymbol</code>	130
<code>\glssymbolaccessdisplay</code>	327
<code>\glssymbolnav</code>	258
<code>\GLSsymbolplural</code>	131
<code>\Glsymbolplural</code>	131
<code>\glssymbolplural</code>	131
<code>\glssymbolpluralaccessdisplay</code>	327
<code>\glssymbolsgroupname</code>	
.....	31, 154, 155, 201
<code>\glstarget</code>	198
<code>\GLStext</code>	124
<code>\Glstext</code>	124
<code>\glstext</code>	123
<code>\glstextaccessdisplay</code>	326
<code>\glstextformat</code>	92
<code>\glstextup</code>	208
<code>\glstildechar</code>	155
<code>\glstreeindent</code>	295, 296
<code>\glstreenamfmt</code>	292
<code>\glsunset</code>	84
<code>\glsunsetall</code>	86
<code>\glsupacrpluralsuffix</code>	31
<code>\GlsUseAcrEntryDispStyle</code> ...	214
<code>\GlsUseAcrStyleDefs</code>	214
<code>\GLSuseri</code>	132
<code>\Glsuseri</code>	132
<code>\glsuseri</code>	132
<code>\GLSuserii</code>	133
<code>\Glsuserii</code>	133
<code>\glsuserii</code>	132, 133
<code>\GLSuseriii</code>	134
<code>\Glsuseriii</code>	134
<code>\glsuseriii</code>	133, 134
<code>\GLSuseriv</code>	135
<code>\Glsuseriv</code>	135
<code>\glsuseriv</code>	134, 135
<code>\GLSuseriv</code>	136
<code>\Glsuseriv</code>	135
<code>\glsuseriv</code>	135, 136
<code>\GLSuserivi</code>	137
<code>\Glsuserivi</code>	136
<code>\glsuserivi</code>	136, 137
<code>\glswrite</code>	165
<code>\glswritedefhook</code>	75
<code>\glswriteentry</code>	172

H

<code>\hyperbf</code>	206
-----------------------------	-----

<code>\hyperemph</code>	207
<code>hyperfirst (option)</code>	23
<code>\hyperit</code>	206
<code>\hyperlink</code>	116
<code>\hypermd</code>	206
<code>\hyperpage</code>	204
<code>hyperref package</code> ...	175, 178, 204, 250
<code>\hyperrm</code>	206
<code>\hypersc</code>	207
<code>\hypersf</code>	206
<code>\hypersl</code>	206
<code>\hypertarget</code>	116
<code>\hypertt</code>	206
<code>\hyperup</code>	207

I

<code>\if@gls@docloaded</code>	4
<code>\if@gls@isacronymlist</code>	15
<code>\ifglossaryexists</code>	50
<code>\ifglsdescsuppressed</code>	53
<code>\ifglsentryexists</code>	51
<code>\ifglsfieldcseq</code>	74
<code>\ifglsfielddefeq</code>	74
<code>\ifglsfieldeq</code>	73
<code>\ifglschildren</code>	52
<code>\ifglsdesc</code>	52
<code>\ifglsfield</code>	53
<code>\ifglsfieldlong</code>	53
<code>\ifglsfieldparent</code>	52
<code>\ifglsfieldprefix</code>	252
<code>\ifglsfieldprefixfirst</code>	252
<code>\ifglsfieldprefixfirstplural</code> .	252
<code>\ifglsfieldprefixplural</code>	252
<code>\ifglsfieldshort</code>	53
<code>\ifglsfieldsymbol</code>	53
<code>\ifglsfieldtranslate</code>	21
<code>\ifglsused</code>	51, 83
<code>\ifglsxindy</code>	25
<code>\ifignoredglossary</code>	59
<code>index (option)</code>	27
<code>index (style)</code>	293
<code>indexgroup (style)</code>	294
<code>indexhypergroup (style)</code>	294
<code>indexonlyfirst (option)</code>	23
<code>\indexspace</code>	261, 276, 292
<code>inline (style)</code>	259
<code>\inputencodingname</code>	25
<code>\istfile</code>	170
<code>\istfilename</code>	34

\item 204, 261, 293

L

link text 92
list (style) 261
listdotted (style) 263
listgroup (style) 262
listhypergroup (style) 262
\loadglsentries 13, 92
long (key) 63
long (style) 265
long-sc-short (acrstyle) 217
long-sc-short-desc (acrstyle) ..
..... 218, 343
long-short (acrstyle) 215, 341
long-short-desc (acrstyle) . 218, 343
long-sm-short (acrstyle) 217
long-sm-short-desc (acrstyle) ..
..... 218, 343
long-sp-short (acrstyle) 215
long-sp-short-desc (acrstyle) .. 218
long3col (style) 266
long3colborder (style) 267
long3colheader (style) 267
long3colheaderborder (style) .. 267
long4col (style) 267
long4colborder (style) 268
long4colheader (style) 268
long4colheaderborder (style) .. 269
longaccess (key) 322
longborder (style) 265
longheader (style) 266
longheaderborder (style) 266
\longnewglossaryentry 59, 75
longplural (key) 63
longpluralaccess (key) 322
\longprovideglossaryentry ... 75
longragged (style) 271
longragged3col (style) 272
longragged3colborder (style) .. 273
longragged3colheader (style) .. 273
longragged3colheaderborder
(style) 273
longraggedborder (style) 271
longraggedheader (style) 272
longraggedheaderborder (style) 272
longtable (environment)
..... 8, 243, 264–275
longtable package 264, 270

M

makeglossaries 24, 34, 48, 56, 164, 181
\makeglossaries
... 26, 29, 34, 35, 55, 57, 164, 165
\makeglossary 165
makeindex 358
makeindex
. 10, 25, 26, 31, 34–36, 56, 57,
59, 83, 108, 111, 154, 157, 160,
161, 170, 175, 200, 201, 301, 302
delim_n 36
delim_r 36
page_compositor 34
special characters 110, 154
makeindex (option) 25
\makenoidxglossaries 21, 166
mcolalttree (style) 279
mcolalttreegroup (style) 279
mcolalttreehypergroup (style) . 279
mcolindex (style) 276
mcolindexgroup (style) 276
mcolindexhypergroup (style) ... 276
mcoltree (style) 277
mcoltreegroup (style) 277
mcoltreehypergroup (style) 277
mcoltreenoname (style) 278
mcoltreenonamegroup (style) ... 278
mcoltreenonamehypergroup
(style) 278
memoir class 171
mfirstuc package 1
multicol package 276
multicols (environment) 276

N

name (key) 59
\new@glossaryentry 66
\newacronym 24, 62, 63, 91, 92, 207, 208
\newacronymhook 212
\newacronymstyle 214
\newglossary 15, 56, 57, 161, 164, 192
\newglossaryentry 59, 66, 91, 92, 208
\newglossaryentry options
access 323, 324
counter 61
description 24, 59,
63, 66, 76, 128, 146, 208, 236, 322
descriptionaccess 325, 327
descriptionplural 129, 322

descriptionpluralaccess ..	325, 327
first ..	60,
	79, 117, 124, 148, 234, 239, 321
firstaccess	324, 326
firstplural	60, 126, 148, 321
firstpluralaccess	325, 327
format	156
long	98, 150, 322
longaccess	325, 327
longplural	151, 322
longpluralaccess	326, 327
name	59,
	60, 63, 66, 76, 127, 145, 178, 321
nonumberlist	61
parent	61, 66
plural	60, 79, 125, 321
pluralaccess	325, 326
prefix	251
prefixfirst	251
prefixfirstplural	251
prefixplural	251
see	8, 61, 164, 166
short	98, 150, 322
shortaccess	325, 327
shortplural	150, 322
shortpluralaccess	325, 327
sort	59, 149, 200, 201
symbol	59, 60, 130, 229–
	231, 234, 239, 267, 283, 321–323
symbolaccess	325, 327
symbolplural	131, 322
symbolpluralaccess	325, 327
text	60, 117, 123, 147, 229, 234, 321
textaccess	324, 326
type	12, 61, 92, 148
user1	132, 149, 322
user2	132, 149
user3	133, 149
user4	134, 149
user5	135, 150
user6	136, 150, 322
\newglossarystyle	203
\newignoredglossary	58
nogroupskip (option)	9
nohypertypes (option)	16
\noist	161, 250, 306
nolist (option)	8
nolong (option)	8
nomain (option)	13
nonumberlist (key)	61
nonumberlist (option)	7
\nopostdesc	33
nopostdot (option)	9
noredefwarn (option)	17
nostyles (option)	8
nosuper (option)	8
notranslate (option)	22
notree (option)	8
nowarn (option)	16
\ns@newglossary	56
numberedsection (option)	6
numberline (option)	5
numbers (option)	27
O	
\oldacronym	207
order (option)	24
P	
\p@gl@s@hyp@opt	104
package options:	
acronym	13, 30, 179, 208
true	14
acronym	13
acronymlists	15
acronyms	14
automake	26
compatible-2.07	26
compatible-3.07	26
counter	16
counter	16
description	233, 234
description	24
dua	232–234
dua	24
entrycounter	193, 195
true	9
entrycounter	9
entrycounterwithin	9
footnote	118–123, 230, 232, 233, 236
footnote	23
hyperfirst	
false	118–123
hyperfirst	23
index	27
indexonlyfirst	365
indexonlyfirst	23
makeindex	158, 250
makeindex	25

nogroupskip	9	toc	5
nohypertypes	16	true	5
nolist	243	toc	5
nolist	8	translate	22
nolong	243, 264	false	22
nolong	8	translate	22
nomain	12, 13	translator	21
nomain	13	ucmark	9
nonumberlist	7	xindy	25, 158, 250
nonumberlist	7	xindy	25
nopostdot	9	xindygloss	25
noredefwarn	17	xindynoglsnumbers	26
nostyles	8	\pagelistname	30
nosuper	243	parent (key)	61
nosuper	8	\PGLS	255
notranslate	22	\PglS	254
notree	244	\pgls	252
notree	8	\PGLSpl	256
nowarn	16	\PglSpl	254
numberedsection	6	\pglSpl	253
numberline	5	\phantomsection	37–39
numberline	5	plural (key)	60
numbers	27	pluralaccess (key)	321
order	24	polyglossia package	21, 32
sanitize	19, 59, 145, 146	\printacronyms	13
sanitize	21	\PrintChanges	5
sanitizesort	17	\printglossaries	
sanitizesort	20	. 12, 36, 55, 58, 165, 178, 179, 256	
savenumberlist	7	\printglossary	36, 37,
savewrites	26, 363	56, 58, 165, 178, 181, 191, 256, 257	
false	162	\printglossary options	
true	164, 170	entrycounter	193
savewrites	26	nogroupskip	192
section	6, 38	nonumberlist	194
section	6	nopostdot	193
seeautonumberlist	8	numberedsection	192
shotcuts	24	style	192
smallcaps	24	subentrycounter	193
smaller	24	title	192
sort		toctitle	192
def	10, 11	type	12, 178, 191
standard	10	\printnoidxglossaries	179
use	10, 11	\printnoidxglossary	179, 191
sort	10	\printnoidxglossary options	
style	7, 243, 244	sort	194
style	7	\provideglossaryentry	66
subentrycounter	193, 195	\ProvidesGlossariesLang	31
subentrycounter	9		
symbols	27		
		R	
		\renewacronymstyle	214

<code>\renewglossarystyle</code>	204	<code>\setglossentrycompatibility</code>	200
<code>\RequireGlossariesLang</code>	31	<code>\SetSmallAcronymDisplayStyle</code>	237
<code>\roman</code>	44	<code>\SetSmallAcronymStyle</code>	239
S			
<code>\s@glshyp@opt</code>	104	<code>\setstylefile</code>	34
<code>\s@newglossary</code>	56	<code>\setupglossaries</code>	28
<code>sanitize</code> (option)	21	<code>short</code> (key)	62
<code>sanitizesort</code> (option)	20	<code>short-long</code> (acrstyle)	216, 342
<code>savenumberlist</code> (option)	7	<code>short-long-desc</code> (acrstyle) .	219, 344
<code>savewrites</code> (option)	26	<code>shortaccess</code> (key)	322
<code>sc-short-long</code> (acrstyle)	217	<code>shortplural</code> (key)	62
<code>sc-short-long-desc</code> (acrstyle) ..		<code>shortpluralaccess</code> (key)	322
.....	219, 344	<code>shotcuts</code> (option)	24
<code>\scantokens</code>	50	<code>\showacronymlists</code>	248
<code>\section</code>	50	<code>\showglocounter</code>	245
<code>section</code> (option)	6	<code>\showglodesc</code>	246
<code>see</code> (key)	61	<code>\showglodescaccess</code>	356
<code>seeautonumberlist</code> (option)	8	<code>\showglodescplural</code>	247
<code>\seename</code>	31	<code>\showglodescpluralaccess</code> ..	356
<code>\SetAcronymLists</code>	15	<code>\showglofield</code>	248
<code>\SetAcronymStyle</code>	14, 241	<code>\showglofirst</code>	245
<code>\setacronymstyle</code>	213	<code>\showglofirstaccess</code>	356
<code>\SetCustomDisplayStyle</code>	242	<code>\showglofirstpl</code>	245
<code>\SetCustomStyle</code>	243	<code>\showglofirstpluralaccess</code> ..	356
<code>\SetDefaultAcronymDisplayStyle</code>		<code>\showgloflag</code>	248
.....	226	<code>\showgloindex</code>	248
<code>\SetDefaultAcronymStyle</code>	227	<code>\showglolevel</code>	244
<code>\SetDescriptionAcronymDisplayStyle</code>		<code>\showgloloclist</code>	248
.....	232	<code>\showglolong</code>	247
<code>\SetDescriptionAcronymStyle</code>	233	<code>\showglolongaccess</code>	357
<code>\SetDescriptionDUAAcronymDisplayStyle</code>		<code>\showglolongpluralaccess</code> ..	357
.....	230	<code>\showgloname</code>	246
<code>\SetDescriptionDUAAcronymStyle</code>		<code>\showglonameaccess</code>	355
.....	231	<code>\showgloparent</code>	244
<code>\SetDescriptionFootnoteAcronymDisplayStyle</code>		<code>\showgloplural</code>	244
.....	228	<code>\showglopluralaccess</code>	356
<code>\SetDescriptionFootnoteAcronymStyle</code>		<code>\showgloshort</code>	247
.....	229	<code>\showgloshortaccess</code>	356
<code>\SetDUADisplayStyle</code>	240	<code>\showgloshortpluralaccess</code> ..	356
<code>\SetDUAStyle</code>	241	<code>\showglosort</code>	247
<code>\setentrycounter</code>	202	<code>\showglossaries</code>	248
<code>\SetFootnoteAcronymDisplayStyle</code>		<code>\showglossarycounter</code>	249
.....	234	<code>\showglossaryentries</code>	249
<code>\SetFootnoteAcronymStyle</code> ...	236	<code>\showglossaryin</code>	249
<code>\SetGenericNewAcronym</code>	212	<code>\showglossaryout</code>	249
<code>\setglossarypreamble</code>	36	<code>\showglossarytitle</code>	249
<code>\setglossarysection</code>	38	<code>\showglosymbol</code>	247
<code>\setglossarystyle</code>	203	<code>\showglosymbolaccess</code>	356
		<code>\showglosymbolplural</code>	247
		<code>\showglosymbolpluralaccess</code> .	356

`\showglotext` 244
`\showglotextaccess` 356
`\showglotype` 245
`\showglouseri` 245
`\showglouserii` 245
`\showglouseriii` 246
`\showglouseriv` 246
`\showglouserv` 246
`\showglouservi` 246
`sm-short-long (acrstyle)` 217
`sm-short-long-desc (acrstyle)` ..
..... 219, 344
`smallcaps (option)` 24
`smaller (option)` 24
`\SmallNewAcronymDef` 238, 354
`sort (key)` 59
`sort (option)` 10
`style (option)` 7
`subentrycounter (option)` 9
`\subglossentry` 198
`\subitem` 293
`sublistdotted (style)` 264
`\subsubitem` 293
`super (style)` 280
`super3col (style)` 282
`super3colborder (style)` 282
`super3colheader (style)` 283
`super3colheaderborder (style)` . 283
`super4col (style)` 283
`super4colborder (style)` 284
`super4colheader (style)` 284
`super4colheaderborder (style)` . 285
`superborder (style)` 281
`superheader (style)` 281
`superheaderborder (style)` 281
`superragged (style)` 287
`superragged3col (style)` 289
`superragged3colborder (style)` . 289
`superragged3colheader (style)` . 290
`superraggedborder (style)` 288
`superraggedheader (style)` 288
`superraggedheaderborder (style)` 288
`superraggedright3colheaderborder
(style)` 290
`supertabular (environment)` ...
..... 8, 243, 280–292
`supertabular package` .. 8, 243, 280, 287
`symbol (key)` 60
`symbolaccess (key)` 322

`\symbolname` 30
`symbolplural (key)` 60
`symbolpluralaccess (key)` 322
`symbols (option)` 27

T

`text (key)` 60
`textaccess (key)` 321
`textcase package` 4
`\theequation` 175
`theglossary (environment)`
..... 5, 16, 36, 37, 197,
198, 203, 277, 278, 294, 295, 297
`\theHequation` 175
`theindex (environment)` 293
`toc (option)` 5
`tracklang package` 31, 357, 358
`\translate` 32
`translate (option)` 22
`translator package`
..... 12–14, 22, 27, 31–33, 178
`tree (style)` 294
`treegroup (style)` 295
`treehypergroup (style)` 295
`treenoname (style)` 296
`treenonamegroup (style)` 296
`treenonamehypergroup (style)` .. 297
`type (key)` 61

U

`ucmark (option)` 9
`user1 (key)` 62
`user2 (key)` 62
`user3 (key)` 62
`user4 (key)` 62
`user5 (key)` 62
`user6 (key)` 62

W

`\warn@nomakeglossaries` 163
`\warn@noprintglossary` 178
`\writeist` 34, 41, 43, 46, 156, 300, 302

X

`\xglsaccsupp` 326
`xindy` 358
`xindy` . 10, 25, 26, 34, 35, 40, 43, 45,
47–49, 82, 115, 155–157, 170,
175, 181, 200, 201, 250, 301, 302
`xindy (option)` 25

xindygloss (option) 25 xspace package 4, 207
xindynoglsnumbers (option) 26