

Documented Code For glossaries v4.15

Nicola L.C. Talbot

Dickimaw Books

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

2015-03-16

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 v4.15: \LaTeX 2e Package to Assist Generating Glossaries”.

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

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

INSTALL Installation instructions.

CHANGES Change log.

README Package summary.

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

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

Contents

1 Main Package Code	4
1.1 Package Definition	4
1.2 Package Options	5
1.3 Predefined Text	30
1.4 Xindy	40
1.5 Loops and conditionals	49
1.6 Defining new glossaries	55
1.7 Defining new entries	59
1.8 Resetting and unsetting entry flags	80
1.9 Keeping Track of How Many Times an Entry Has Been Unset	83
1.10 Loading files containing glossary entries	88
1.11 Using glossary entries in the text	88
1.11.1 Links to glossary entries	99
1.11.2 Displaying entry details without adding information to the glossary	140
1.12 Adding an entry to the glossary without generating text	148
1.13 Creating associated files	150
1.14 Writing information to associated files	165
1.15 Glossary Entry Cross-References	171
1.16 Displaying the glossary	173
1.17 Acronyms	202
1.18 Predefined acronym styles	207
1.19 Predefined Glossary Styles	238
1.20 Debugging Commands	238
1.21 Compatibility with version 2.07 and below	244
2 Prefix Support (glossaries-prefix Code)	244
3 Mfirstuc Documented Code	250
4 Mfirstuc-english Documented Code	253
5 Glossary Styles	254
5.1 Glossary hyper-navigation definitions (glossary-hypernav package)	254
5.2 In-line Style (glossary-inline.sty)	256
5.3 List Style (glossary-list.sty)	259
5.4 Glossary Styles using longtable (the glossary-long package)	262
5.5 Glossary Styles using longtable (the glossary-longragged package)	268
5.6 Glossary Styles using multicol (glossary-mcols.sty)	273
5.7 Glossary Styles using supertabular environment (glossary-super package)	277

5.8	Glossary Styles using supertabular environment (glossary-superragged package)	284
5.9	Tree Styles (glossary-tree.sty)	290
6	glossaries-compatible-207	298
7	Accessibility Support (glossaries-accsupp Code)	318
7.1	Defining Replacement Text	319
7.2	Accessing Replacement Text	322
7.3	Displaying the Glossary	337
7.4	Acronyms	339
7.5	Debugging Commands	353
8	Multi-Lingual Support	354
8.1	Polyglossia Captions	355
	Glossary	356
	Change History	356
	Index	380

1 Main Package Code

1.1 Package Definition

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

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries}[2015/03/16 v4.15 (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
```

```
glsorg@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 redefined `\glossarysection`.

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

Determine whether or not to use numbered sections.

`\@glossarysecstar`

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

`\@glossaryseclabel`

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

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

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

`numberedsection`

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

```
60 \fi
61 }
```

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

`\@glossary@default@style`

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

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

`\glossaryentrynumbers`

```
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
```

@seeautonumberlist

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

no long

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

```
83 \@gls@declareoption{no long}{\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{}}
```

no super

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

```
87 \@gls@declareoption{no super}{\renewcommand*\@gls@loadsuper{}}
```

\@gls@loadlist

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

no list

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

```
89 \@gls@declareoption{no list}{\renewcommand*\@gls@loadlist{}}
```

\@gls@loadtree

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

no tree

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

```
91 \@gls@declareoption{no tree}{\renewcommand*\@gls@loadtree{}}
```

no styles

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{no styles}{%
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`

`\@gls@default@sorttype` Initialise default sort for `\printnoidxglossary`
123 `\newcommand*{\@gls@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*{\@gls@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@gls@storeentry{\@gls@storeentry}%`
No count register required for standard sort.
133 `\def\@gls@defsortcount##1{}`
Sort according to sort key (`\@gls@sort`) if provided otherwise sort according to the entry's name (`\@gls@name`). (First argument glossary type, second argument entry label.)
134 `\def\@gls@defsort##1##2{%`
135 `\ifx\@gls@sort\@gls@defaultsort`
136 `\let\@gls@sort\@gls@name`
137 `\fi`
138 `\let\glsdosanitizesort\@gls@sanitizesort`
139 `\glsprestandardsort{\@gls@sort}{##1}{##2}%`
140 `\expandafter\protected@xdef\csname glo@##2@sort\endcsname{\@gls@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\def\csname glo@##2@sort\endcsname{%  
163 \expandafter\glssortnumberfmt  
164 {\csname glossary@##1@sortcount\endcsname}}%  
165 }%
```

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

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

`\@gls@setupsort@use` Set up the macros for order of use sorting.

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

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

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

Defined count register associated with the glossary.

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

Initialise the sort key to empty.

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

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

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

Get the parent, if one exists

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

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

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

Set index information for this entry

```
183   \edef\@glo@type{\csname glo@##1@type\endcsname}%
184   \edef\@gls@tmp{\csname glo@##1@sort\endcsname}%
185   \ifx\@gls@tmp\@empty
186     \expandafter\global\expandafter
187     \advance\csname glossary@\@glo@type @sortcount\endcsname by 1\relax
188     \expandafter\protected\xdef\csname glo@##1@sort\endcsname{%
189       \expandafter\glssortnumberfmt
190       {\csname glossary@\@glo@type @sortcount\endcsname}}%
191     \@glo@storeentry{##1}%
192   \fi
193 }%
194 }
```

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

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

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

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

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

`\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}%
217     }%
218   \else
219     \let\@gls@do@acronymsdef\relax
220   \fi
221 }
```

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

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

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

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

`@gls@do@acronymsdef` Set default value

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

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

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

```

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

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

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

```
247 \newcommand*\@glsacronymlists{}
```

`\@addtoacronymlists`

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

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

```

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

```

`\glsIfListOfAcronyms`

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

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

```

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

```

Internal command requires label and list to be expanded:

```

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

```

Not found

```

268   #4%
269   \else

```

Found

```

270   #3%
271   \fi
272 }

```

`\if@glsisacronymlist`

Convenient boolean.

```
273 \newif\if@glsisacronymlist
```

`@checkisacronymlist`

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

```

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

```

`\SetAcronymLists`

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

```

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

```

`acronymlists`

```

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

```

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

`\glscounter`

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

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

```
285 \define@key{glossaries.sty}{counter}{%
```

```
286   \renewcommand*{\glscounter}{#1}%
```

```
287 }
```

`\@gls@nohyperlist`

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

`sDeclareNoHyperList`

```
289 \newcommand*{\GlsDeclareNoHyperList}[1]{%
```

```
290   \ifdefempty\@gls@nohyperlist
```

```
291   {%
```

```
292     \renewcommand*{\@gls@nohyperlist}{#1}%
```

```
293   }%
```

```
294   {%
```

```
295     \appto\@gls@nohyperlist{,#1}%
```

```
296   }%
```

```
297 }
```

`nohypertypes`

```
298 \define@key{glossaries.sty}{nohypertypes}{%
```

```
299   \GlsDeclareNoHyperList{#1}%
```

```
300 }
```

`\GlossariesWarning` Prints a warning message.

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

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

```
303 }
```

`sariesWarningNoLine` Prints a warning message without the line number.

```
304 \newcommand*{\GlossariesWarningNoLine}[1]{%
```

```
305   \PackageWarningNoLine{glossaries}{#1}%
```

```
306 }
```

`nowarn` Define package option to suppress warnings

```
307 \@gls@declareoption{nowarn}{%
```

```
308   \renewcommand*{\GlossariesWarning}[1]{}%
```

```
309   \renewcommand*{\GlossariesWarningNoLine}[1]{}%
```

```
310 }
```


`@warnonglossdefined` Issue a warning if overriding `\printglossary`

```

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

`nontheglossdefined` 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 }

```

lgnoidxstripaccents

```

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   \glssanitizesortfalse
455   \glssetexpandfield{sortvalue}%
456 }

```

```

457 \define@choicekey[gls]{sanitize}{sort}{true,false}[true]{%
458   \setbool{glssanitizesort}{#1}%
459   \ifglssanitizesort
460     \glssetnoexpandfield{sortvalue}%
461   \else
462     \glssetexpandfield{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       \glssetexpandfield{name}%
472       \glssetexpandfield{symbol}%
473       \glssetexpandfield{symbolplural}%
474       \glssetexpandfield{desc}%
475       \glssetexpandfield{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

```

`\@gls@notranslatorhook` `\@gls@notranslatorhook` has been removed.

`\@gls@usetranslator`

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

`fusedtranslatordict` Checks if given translator dictionary has been loaded.

```
500 \newcommand{\glsifusedtranslatordict}[3]{%
501   \glsifusetranslator
502   {\ifcsdef{ver@glossaries-dictionary-#1.dict}{#2}{#3}}%
503   {#3}%
504 }
```

`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   }
```

```

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}}
```

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

```
595 \newif\ifglsxindy
```

The default is `makeindex`:

```
596 \glsxindyfalse
```

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

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

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 \glsxindytrue
```

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

```
610 }
```

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

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

`xindynoglsnumbers` Provide a synonym for `xindy=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] [] {%
684     \newglossaryentry{##2}%
685     {type={index},name={##2},description={\nopostdesc},##1}}
686 }%
```

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

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

Now process options passed to the package:

```
698 \ProcessOptionsX
```

Load backward compatibility stuff:

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

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

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

Now define `\setupglossaries`:

```

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

```

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

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

```

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

```

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

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

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

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

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

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

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

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

1.3 Predefined Text

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

Main glossary title:

`\glossaryname`

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

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

`\acronymname`

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

`\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 `LaTeX`, `\@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 \ifglxindy
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@suffiX` 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@suffiX}{}
```

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

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

Only has an effect when used before `\makeglossaries`

```
890 \@onlypremakeg\glsSetSuffiX
```

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

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

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

```
892 \newcommand*\glsSetSuffixFF[1]{%
893   \renewcommand*\gls@suffixFF{#1}%
894 }
```

`\glsnumberformat` The command `\glsnumberformat` indicates the default format for the page numbers in the glossary. (Note that this is not the same as `\glossaryentrynumbers`, but applies to individual numbers or groups of numbers within an entry's associated number list.) If hyperlinks are defined, it will use `\glsnumber`, otherwise it will simply display its argument “as is”.

```
895 \ifcsundef{hyperlink}%
896 {%
897   \newcommand*\glsnumberformat[1]{#1}%
898 }%
899 {%
900   \newcommand*\glsnumberformat[1]{\glsnumber{#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 `\glossary` environment. It is designed to allow the user to add information pertaining to the glossary (e.g. “page numbers in italic indicate the primary definition”) therefore `\glossarypreamble` shouldn't be affected by the glossary style. (So if you define your own glossary style, don't have it change `\glossarypreamble`.) The preamble is empty by default. If you have multiple glossaries, and you want a different preamble for each glossary, you will need to use `\printglossary` for each glossary type, instead of `\printglossaries`, and redefine `\glossarypreamble` before each `\printglossary`.

`\glossarypreamble`

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

`\setglossarypreamble`

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

Code provided by Michael Pock.

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

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

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

`\glossarypostamble`

```
916 \newcommand*\glossarypostamble}{}
```

`\glossarysection`

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

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

926   \gls@glossarymark{\glossarytoctitle}%
927 }
```

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

```
928 \ifcsundef{glossarymark}%
929 {%
930   \newcommand{\glsglossarymark}[1]{\glossarymark{#1}}
931 }%
932 {%
933   \@ifclassloaded{memoir}
934   {%
935     \newcommand{\glsglossarymark}[1]{%
936       \ifglsucmark
937         \markboth{\memUHead{#1}}{\memUHead{#1}}%
938       \else
939         \markboth{#1}{#1}%
940       \fi
941     }
942   }%
943   {%
944     \newcommand{\glsglossarymark}[1]{%
945       \ifglsucmark
946         \@mkboth{\mfirstucMakeUppercase{#1}}{\mfirstucMakeUppercase{#1}}%
947       \else
948         \@mkboth{#1}{#1}%
949       \fi
950     }
951   }
952 }
```

`\glossarymark` Provided for backward compatibility:

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

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

`\setglossarysection`

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

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

`\@glossarysection`

```
962 \newcommand*\@glossarysection}[2]{%
963   \ifdefempty\@glossarysecstar
964   {%
965     \csname\@glossarysec\endcsname[#1]{#2}%
966   }%
967   {%
968     \csname\@glossarysec\endcsname*{#2}%
969     \@gls@toc{#1}{\@glossarysec}%
970   }%
```

Do automatic labelling if required

```
971   \@glossaryseclabel
972 }
```

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

`\@p@glossarysection`

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

Do automatic labelling if required

```
984   \@glossaryseclabel
985 }
```

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

```

\@xdyattributelist Comma-separated list of attributes.

```

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

```

\@xdylocref Define list of markup location references.

```

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

```

\@gls@ifinlist

```

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

```

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

```

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

```

Check if already in list before adding.

```

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

```

Only has an effect before \writeist:

```

1056   \@onlypremakeg\GlsAddXdyCounters

```

```

1057 \else
1058   \newcommand*\GlsAddXdyCounters[1]{%
1059     \glsnoxywarning\GlsAddXdyAttribute
1060   }
1061 \fi

```

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

```

Add to xindy markup location.

```

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

```

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

```

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

```

High-level command:

```

1084 \newcommand*\GlsAddXdyAttribute[1]{%

```

Add to comma-separated attribute list

```

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

```

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

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

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

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

Only has an effect before `\writeist`:

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

`redefinedattributes` Add known attributes for all defined counters

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

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

`\@xdyuseralphabets` List of additional alphabets

```

1132 \def\@xdyuseralphabets{}

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

1133 \ifglxsindy
1134 \newcommand*{\GlsAddXdyAlphabet}[2]{%
1135 \edef\@xdyuseralphabets{%
1136 \@xdyuseralphabets ^^J
1137 (define-alphabet "#1" (#2))}}
1138 \else
1139 \newcommand*{\GlsAddXdyAlphabet}[2]{%
1140 \glsnnoxindywarning\GlsAddXdyAlphabet}
1141 \fi

```

This code is only required for xindy:

```

1142 \ifglxsindy

```

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

```

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

```

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

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

```

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

```

```

1165     \edef\csname @gls@xdy@Lclass@roman-page-numbers\endcsname{
1166         :sep \string"\@gls@roman\string"%
1167     }%
1168     \fi

@Roman-page-numbers Upper case Roman numerals (I, II, ...).
1169     \expandafter\def\csname @gls@xdy@Lclass@Roman-page-numbers\endcsname{%
1170         \string"roman-numbers-uppercase\string"%
1171     }%

arabic-page-numbers Arabic numbers (1, 2, ...).
1172     \expandafter\def\csname @gls@xdy@Lclass@arabic-page-numbers\endcsname{%
1173         \string"arabic-numbers\string"%
1174     }%

@alpha-page-numbers Lower case alphabetical (a, b, ...).
1175     \expandafter\def\csname @gls@xdy@Lclass@alpha-page-numbers\endcsname{%
1176         \string"alpha\string"%
1177     }%

@Alpha-page-numbers Upper case alphabetical (A, B, ...).
1178     \expandafter\def\csname @gls@xdy@Lclass@Alpha-page-numbers\endcsname{%
1179         \string"ALPHA\string"%
1180     }%

pendix-page-numbers Appendix style locations (e.g. A-1, A-2, ..., B-1, B-2, ...). The separator is given
by \@glsAlphacompositor.
1181     \expandafter\def\csname @gls@xdy@Lclass@Appendix-page-numbers\endcsname{%
1182         \string"ALPHA\string"
1183         :sep \string"\@glsAlphacompositor\string"
1184         \string"arabic-numbers\string"%
1185     }

abic-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 location called *<name>*. The definition must use xindy syntax. (Note that this doesn't check to see if the location is already defined. That is left to xindy to complain about.)

```
1194 \ifglxindy
1195   \newcommand*{\GlsAddXdyLocation}[3] [] {%
1196     \def\@gls@tmp{#1}%
1197     \ifx\@gls@tmp\@empty
1198       \edef\@xdyuserlocationdefs{%
1199         \@xdyuserlocationdefs ^^J%
1200         (define-location-class \string"#2\string"^^J\space\space
1201         \space(:sep \string"{}\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     \glsnoxindywarning\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     \glsnoxindywarning\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     \glsnoxindywarning\GlsAddXdyStyle}
1265 \fi

```

`\GlsSetXdyStyles` Reset the list of required styles

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

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

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

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

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

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

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

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

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

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

```
1288 \ifglxindy
1289   \newcommand*\GlsSetXdyCodePage[1]{%
```



```

1290   \renewcommand*{\gls@codepage}{#1}%
1291   }
      Suggested by egreg:
1292   \AtBeginDocument{%
1293     \ifx\gls@codepage\@empty
1294       \@ifpackageloaded{fontspec}{\def\gls@codepage{utf8}}{}%
1295     \fi
1296   }
1297 \else
1298   \newcommand*{\GlsSetXdyCodePage}[1]{%
1299     \glsnoxywarning\GlsSetXdyCodePage}
1300 \fi

```

`\@xdylettergroups` Store letter group definitions.

```

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

```

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

```

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

```

1.5 Loops and conditionals

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

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

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

```

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

```

`\forallacronyms`

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

`\glsdetoklabel`

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

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

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

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

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

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

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

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

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

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

`\glsdoifexists`

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

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

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

`\glsdoifnoexists`

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

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

```

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

```

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

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

```

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

```

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

```

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

```

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

```

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

```

```
1389 }%
1390 }
```

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

```
1391 \newcommand*{\ifglshasdesc}[3]{%
1392 \ifcsemtyp{glo@\glsdetoklabel{#1}@desc}%
1393 {#3}%
1394 {#2}%
1395 }
```

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

```
1396 \newcommand*{\ifglsdescsuppressed}[3]{%
1397 \ifcsequal{glo@\glsdetoklabel{#1}@desc}{@no@post@desc}%
1398 {#2}%
1399 {#3}%
1400 }
```

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

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

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

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

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

```
1421 \newcommand*{\ifglshasshort}[3]{%
1422 \letcs{@glo@short}{glo@\glsdetoklabel{#1}@short}%
1423 \ifdefempty@glo@short
1424 {#3}%
1425 {%
```

```

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

```

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

```

1431 \newcommand*\ifglshasfield[4]{%
1432   \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       \ifdefequal\@glo@thisvalue\@gls@default@value{#4}{#3}%
1443     }%
1444   }%
1445 }%

```

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

```

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

```

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

```

1447   \ifdef\@gls@thisfield
1448   {%

```

Is defined, so now check if empty.

```

1449     \letcs{\@glo@thisvalue}{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         \ifdefequal\@glo@thisvalue\@gls@default@value{#4}{#3}%
1456         }%
1457     }%
1458     {%
```

Not defined.

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

1.6 Defining new glossaries

A comma-separated list of glossary names is stored in \@glo@types. When a new glossary type is created, its identifying name is added to this list. This is used by commands that iterate through all glossaries (such as \makeglossaries and \printglossaries).

\@glo@types

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

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

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

```

Only need to do this once.

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

\defglsentryfmt Allow different glossaries to have different display styles.

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

\gls@doentryfmt

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

\@gls@forbidtexext As a security precaution, don't allow the user to specify a 'tex' extension for any of the glossary files. (Just in case a seriously confused novice user doesn't know what they're doing.) The argument must be a control sequence whose replacement text is the requested extension.

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

```

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

```

A new glossary type is defined using `\newglossary`. Syntax:

```

\newglossary[log-ext]{name}[in-ext][out-ext]
{title}[counter]

```

where *log-ext* is the extension of the makeindex transcript file, *in-ext* is the extension of the glossary input file (read in by `\printglossary` and created by `makeindex`), *out-ext* is the extension of the glossary output file which is read in by `makeindex` (lines are written to this file by the `\glossary` command), *title* is the title of the glossary that is used in `\glossarysection` and *counter* is the default counter to be used by entries belonging to this glossary. The `makeglossaries` Perl script reads in the relevant extensions from the auxiliary file, and passes the appropriate file names and switches to `makeindex`.

`\newglossary`

```

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

```

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

```

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

```

`\ns@newglossary` Define the unstarred version.

```

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

```

Check if default has been set

```

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

```


Add this to the list of glossary types:

```
1503 \toks@{#2}\edef\@glo@types{\@glo@types\the\toks@,}%
```

Define a comma-separated list of labels for this glossary type, so that all the entries for this glossary can be reset with a single command. When a new entry is created, its label is added to this list.

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

Store the file extensions:

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

Store the title:

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

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

```
1513 \protected@write\@auxout{}\string\@newglossary{#2}{#1}{#3}{#4}}%
```

How to display this entry in the document text (uses `\glsentry` by default).

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

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

Define sort counter if required:

```
1519 \@gls@defsortcount{#2}%
```

Find out if the final optional argument has been specified, and use it to set the counter associated with this glossary. (Uses `\glscounter` if no optional argument is present.)

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

`\altnewglossary`

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

Only define new glossaries in the preamble:

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

Only define new glossaries before `\makeglossaries`

```
1527 \@onlypremakeg\newglossary
```

`\@newglossary` is used to specify the file extensions for the `makeindex` input, output and transcript files. It is written to the auxiliary file by `\newglossary`. Since it is not used by \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 @gls@#1@counter\endcsname{#2}%
```

Add counter to xindy list, if not already added:

```
1531 \ifglxindy
```

```
1532 \GlsAddXdyCounters{#2}%
```

```
1533 \fi
```

```
1534 }
```

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

`\@gls@getcounter`

```
1535 \newcommand*\@gls@getcounter}[1]{%
```

```
1536 \csname @gls@#1@counter\endcsname
```

```
1537 }
```

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

```
1538 \glsdefmain
```

Define the “acronym” glossaries if required.

```
1539 \@gls@do@acronymsdef
```

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

```
1540 \@gls@do@symbolsdef
```

```
1541 \@gls@do@numbersdef
```

```
1542 \@gls@do@indexdef
```

`\newignoredglossary` Creates a new glossary that doesn't have associated files. This glossary is ignored by and commands that iterate over glossaries, such as `\printglossaries`, and won't work with commands like `\printglossary`. It's intended for entries that are so commonly-known they don't require a glossary.

```
1543 \newcommand*\newignoredglossary}[1]{%
```

```
1544 \ifdefempty\@ignored@glossaries
```

```
1545 {%
```

```
1546 \edef\@ignored@glossaries{#1}%
```

```
1547 }%
```

```
1548 {%
```

```
1549 \eappto\@ignored@glossaries{,#1}%
```

```

1550 }%
1551 \csgdef{glolist@#1}{,}%
1552 \ifcsundef{gls@#1@entryfmt}%
1553 {%
1554   \defglentryfmt[#1]{\glentryfmt}%
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 `\glentryfmt` or using `\defglentryfmt`. The

description key is required when defining a new glossary entry. If a long description is required, use `\longnewglossaryentry` instead of `\newglossaryentry`.

```
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 }
```

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

```
1698 \newcommand*{\@gls@defaultsort}{\@glo@name}
```

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

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

`@gls@noexpand@field`

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

`gls@noexpand@fields`

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



```

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

```

\@@gls@expand@field

```

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

```

@gls@expand@fields

```

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

```

startswitexpandonce

```

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

```

```
\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 defined.
```

```
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 }
```

\glsaddkey

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

Allow user to add their own custom keys.

```
1875 \newcommand*{\glsaddkey}{\@ifstar\@sglsaddkey\@glsaddkey}
```

Starred version switches on expansion for this key.

```
1876 \newcommand*{\@sglsaddkey}[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 \@glsaddkey{#1}%  
1886 }
```

Unstarred version doesn't override default expansion.

```
1887 \newcommand*{\@glsaddkey}[7]{%
```

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

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

```

1899 \ifcsdef{@gls@user@#1@}%
1900 {%
1901   \PackageError{glossaries}%
1902   {Can't define '\string#5' as helper command
1903   '\expandafter\string\csname @gls@user@#1@\endcsname' already exists}%
1904   }%
1905 }%
1906 {%
1907   \expandafter\newcommand\expandafter*\expandafter
1908   {\csname @gls@user@#1@\endcsname}[2][ ]{%
1909     \new@ifnextchar[%
1910     {\csuse{@gls@user@#1@}{##1}{##2}}%
1911     {\csuse{@gls@user@#1@}{##1}{##2}[ ]}}%
1912   \csdef{@gls@user@#1@}##1##2[##3]{%
1913     \@gls@field@link{##1}{##2}{#3{##2}##3}%
1914   }%
1915   \newrobustcmd*{#5}{%
1916     \expandafter\@gls@hyp@opt\csname @gls@user@#1@\endcsname}%
1917   }%

```

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

```

1918 \ifcsdef{@Gls@user@#1@}%
1919 {%
1920   \PackageError{glossaries}%
1921   {Can't define '\string#6' as helper command
1922   '\expandafter\string\csname @Gls@user@#1@\endcsname' already exists}%
1923   }%
1924 }%
1925 {%
1926   \expandafter\newcommand\expandafter*\expandafter
1927   {\csname @Gls@user@#1@\endcsname}[2][ ]{%
1928     \new@ifnextchar[%
1929     {\csuse{@Gls@user@#1@}{##1}{##2}}%
1930     {\csuse{@Gls@user@#1@}{##1}{##2}[ ]}}%
1931   \csdef{@Gls@user@#1@}##1##2[##3]{%
1932     \@gls@field@link{##1}{##2}{#4{##2}##3}%
1933   }%
1934   \newrobustcmd*{#6}{%
1935     \expandafter\@gls@hyp@opt\csname @Gls@user@#1@\endcsname}%
1936   }%

```

Finally the all caps version:

```

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

```

```

1942     {}%
1943     }%
1944     {%

1945     \expandafter\newcommand\expandafter*\expandafter
1946         {\csname @GLS@user@#1\endcsname}[2] []{%
1947         \new@ifnextchar [%
1948             {\csuse{@GLS@user@#1@}{##1}{##2}}%
1949             {\csuse{@GLS@user@#1@}{##1}{##2} []}}%
1950     \csdef{@GLS@user@#1@}##1##2[##3]{%
1951         \@gls@field@link{##1}{##2}{\mfirstucMakeUppercase{#3{##2}##3}}%
1952     }%
1953     \newrobustcmd*{#7}{%
1954         \expandafter\@gls@hyp@opt\csname @GLS@user@#1\endcsname}%
1955     }%
1956     }%
1957     {%
1958     \PackageError{glossaries}{Key ‘#1’ already exists}{}%
1959     }%
1960 }

```

\glswritedefhook

```

1961 \newcommand*\glswritedefhook{}

```

\gls@assign@desc

```

1962 \newcommand*\gls@assign@desc[1]{%
1963     \gls@assign@field{#1}{desc}{\@glo@desc}%
1964     \gls@assign@field{\@glo@desc}{#1}{descplural}{\@glo@descplural}%
1965 }

```

ongnewglossaryentry

```

1966 \newcommand*\longnewglossaryentry[3]{%
1967     \glsdoifnoexists{#1}%
1968     {%
1969         \bgroup
1970         \let\@org@newglossaryentryprehook\@newglossaryentryprehook
1971         \long\def\@newglossaryentryprehook{%
1972             \long\def\@glo@desc{#3\leavevmode\unskip\nopostdesc}%
1973             \@org@newglossaryentryprehook
1974         }%
1975         \renewcommand*\gls@assign@desc[1]{%
1976             \global\cslet{glo@glsdetoklabel{#1}@desc}{\@glo@desc}%
1977             \global\cslet{glo@glsdetoklabel{#1}@descplural}{\@glo@desc}%
1978         }
1979         \gls@defglossaryentry{#1}{#2}%
1980     \egroup
1981     }
1982 }

```

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

```
1983 \onlypreamble{\longnewglossaryentry}
```

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

```
1984 \newcommand{\longprovideglossaryentry}[3]{%
1985   \ifglentryexists{#1}{}%
1986   {\longnewglossaryentry{#1}{#2}{#3}}%
1987 }
1988 \onlypreamble{\longprovideglossaryentry}
```

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

Defines a new entry without checking if it already exists.

```
1989 \newcommand{\gls@defglossaryentry}[2]{%
```

Store label

```
1990   \edef\@glo@label{\glsdetoklabel{#1}}%
```

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

```
1991   \let\glslabel\@glo@label
```

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

```
1992   \let\@glo@name\@gls@name
```

```
1993   \let\@glo@desc\@gls@desc
```

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

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

```
1996   \let\@glo@symbol\@gls@default@value
```

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

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

```
1999   \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.)

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

```
2001   \let\@glo@firstplural\@gls@default@value
```

Set the default sort:

```
2002   \let\@glo@sort\@gls@default@value
```

Set the default counter:

```
2003   \let\@glo@counter\@gls@default@value
```

```
2004   \def\@glo@see{}
```



```

2005 \def\@glo@parent{}%
2006 \def\@glo@prefix{}%

2007 \def\@glo@useri{}%
2008 \def\@glo@userii{}%
2009 \def\@glo@useriii{}%
2010 \def\@glo@useriv{}%
2011 \def\@glo@userv{}%
2012 \def\@glo@uservi{}%

2013 \def\@glo@short{}%
2014 \def\@glo@shortpl{}%
2015 \def\@glo@long{}%
2016 \def\@glo@longpl{}%

```

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

```
2017 \@newglossaryentryprehook
```

Extract key-val information from third parameter:

```
2018 \setkeys{glossentry}{#2}%
```

Check there is a default glossary.

```

2019 \ifundef\glsdefaultttype
2020 {%
2021 \PackageError{glossaries}%
2022 {No default glossary type (have you used ‘nomain’?)}%
2023 {If you use package option ‘nomain’ you must define
2024 a new glossary before you can define entries}%
2025 }%
2026 {}%

```

Assign type. This must be fully expandable

```

2027 \gls@assign@field{\glsdefaultttype}{\@glo@label}{type}{\@glo@type}%
2028 \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.

```

2029 \ifcsundef{glo@list@\@glo@type}%
2030 {%
2031 \PackageError{glossaries}%
2032 {Glossary type ‘\@glo@type’ has not been defined}%
2033 {You need to define a new glossary type, before making entries
2034 in it}%
2035 }%
2036 {}%

```

Check if it's an ignored glossary

```

2037 \ifignoredglossary\@glo@type
2038 {%

```

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

```
2039     \ifx\@glo@desc\@glsnodesc
2040         \let\@glo@desc\@empty
2041     \fi
2042 }%
2043 {%
2044 }%
2045 \protected@edef\@glo@list@{\csname glo@list@\@glo@type\endcsname}%
2046 \expandafter\xdef\csname glo@list@\@glo@type\endcsname{%
2047     \@glo@list@{\@glo@label},}%
2048 }%
```

Initialise level to 0.

```
2049 \gls@level=0\relax
```

Has this entry been assigned a parent?

```
2050 \ifx\@glo@parent\@empty
```

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

```
2051 \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%
2052 \else
```

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

```
2053 \ifdefequal\@glo@label\@glo@parent%
2054 {%
2055     \PackageError{glossaries}{Entry '@@glo@label' can't be its own parent}{}%
2056     \def\@glo@parent{}%
2057     \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%
2058 }%
2059 {%
```

Check the parent exists:

```
2060 \ifglsentryexists{\@glo@parent}%
2061 {%
```

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

```
2062 \expandafter\xdef\csname glo@\@glo@label @parent\endcsname{%
2063     \@glo@parent}%
```

Determine level.

```
2064 \gls@level=\csname glo@\@glo@parent @level\endcsname\relax
2065 \advance\gls@level by 1\relax
```

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

```
2066 \ifx\@glo@name\@glsnoname
2067     \expandafter\let\expandafter\@glo@name
2068     \csname glo@\@glo@parent @name\endcsname
```

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

```
2069 \ifx\@glo@plural\@gls@default@value
2070     \expandafter\let\expandafter\@glo@plural
2071     \csname glo@\@glo@parent @plural\endcsname
2072 \fi
```

```

2073     \fi
2074     }%
2075     {%

```

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

```

2076     \PackageError{glossaries}%
2077     {%
2078         Invalid parent '\@glo@parent'
2079         for entry '\@glo@label' - parent doesn't exist%
2080     }%
2081     {%
2082         Parent entries must be defined before their children%
2083     }%
2084     \def\@glo@parent{}%
2085     \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%
2086     }%
2087     }%
2088     \fi

```

Set the level for this entry

```

2089     \expandafter\xdef\csname glo@\@glo@label @level\endcsname{\number\gls@level}%

```

Define commands associated with this entry:

```

2090     \gls@assign@field{\@glo@name}{\@glo@label}{sortvalue}{\@glo@sort}%
2091     \letcs\@glo@sort{glo@\@glo@label @sortvalue}%
2092     \gls@assign@field{\@glo@name}{\@glo@label}{text}{\@glo@text}%
2093     \expandafter\gls@assign@field\expandafter
2094         {\csname glo@\@glo@label @text\endcsname\glspluralsuffix}%
2095         {\@glo@label}{plural}{\@glo@plural}%
2096     \expandafter\gls@assign@field\expandafter
2097         {\csname glo@\@glo@label @text\endcsname}%
2098         {\@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.

```

2099     \ifx\@glo@first\@gls@default@value
2100         \expandafter\gls@assign@field\expandafter
2101             {\csname glo@\@glo@label @plural\endcsname}%
2102             {\@glo@label}{firstpl}{\@glo@firstplural}%
2103     \else
2104         \expandafter\gls@assign@field\expandafter
2105             {\csname glo@\@glo@label @first\endcsname\glspluralsuffix}%
2106             {\@glo@label}{firstpl}{\@glo@firstplural}%
2107     \fi

2108     \ifcsundef{@glo@type@\@glo@type @counter}%
2109     {%
2110         \def\@glo@defaultcounter{\glscounter}%
2111     }%
2112     {%

```

```

2113     \letcs\@glo@defaultcounter{@glo@type@\@glo@type @counter}%
2114 }%
2115 \gls@assign@field{\@glo@defaultcounter}{\@glo@label}{counter}{\@glo@counter}%
2116 \gls@assign@field{\@glo@label}{useri}{\@glo@useri}%
2117 \gls@assign@field{\@glo@label}{userii}{\@glo@userii}%
2118 \gls@assign@field{\@glo@label}{useriii}{\@glo@useriii}%
2119 \gls@assign@field{\@glo@label}{useriv}{\@glo@useriv}%
2120 \gls@assign@field{\@glo@label}{userv}{\@glo@userv}%
2121 \gls@assign@field{\@glo@label}{uservi}{\@glo@uservi}%
2122 \gls@assign@field{\@glo@label}{short}{\@glo@short}%
2123 \gls@assign@field{\@glo@label}{shortpl}{\@glo@shortpl}%
2124 \gls@assign@field{\@glo@label}{long}{\@glo@long}%
2125 \gls@assign@field{\@glo@label}{longpl}{\@glo@longpl}%
2126 \ifx\@glo@name\@glsnoname
2127     \@glsnoname
2128     \let\@glo@name\@gls@default@value
2129 \fi
2130 \gls@assign@field{\@glo@label}{name}{\@glo@name}%

```

Set default numberlist if not defined:

```

2131 \ifcsundef{glo@\@glo@label @numberlist}%
2132 {%
2133     \csxdef{glo@\@glo@label @numberlist}{%
2134         \noexpand\@gls@missingnumberlist{\@glo@label}}%
2135 }%
2136 {}%

```

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.

```

2137 \def\@glo@@desc{\@glo@first}%
2138 \ifx\@glo@desc\@glo@@desc
2139     \let\@glo@desc\@glo@first
2140 \fi
2141 \ifx\@glo@desc\@glsnodesc
2142     \@glsnodesc
2143     \let\@glo@desc\@gls@default@value
2144 \fi
2145 \gls@assign@desc{\@glo@label}%

```

Set the sort key for this entry:

```

2146 \@gls@defsort{\@glo@type}{\@glo@label}%

2147 \def\@glo@@symbol{\@glo@text}%
2148 \ifx\@glo@symbol\@glo@@symbol
2149     \let\@glo@symbol\@glo@text
2150 \fi
2151 \gls@assign@field{\relax}{\@glo@label}{symbol}{\@glo@symbol}%
2152 \expandafter
2153 \gls@assign@field\expandafter

```

```

2154     {\csname glo@\glo@label @symbol\endcsname}
2155     {\@glo@label}{symbolplural}{\@glo@symbolplural}%

```

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

```

2156     \expandafter\xdef\csname glo@\glo@label @flagfalse\endcsname{%
2157         \noexpand\global
2158         \noexpand\let\expandafter\noexpand
2159         \csname ifglo@\glo@label @flag\endcsname\noexpand\iffalse
2160     }%
2161     \expandafter\xdef\csname glo@\glo@label @flagtrue\endcsname{%
2162         \noexpand\global
2163         \noexpand\let\expandafter\noexpand
2164         \csname ifglo@\glo@label @flag\endcsname\noexpand\iftrue
2165     }%
2166     \csname glo@\glo@label @flagfalse\endcsname

```

Sort out any cross-referencing if required.

```

2167     \ifdefined@glo@see
2168     {}%
2169     {%
2170         \protected@edef\do@glssee{%
2171             \noexpand\gls@fixbraces\noexpand@glo@list@glo@see
2172             \noexpand@nil
2173             \noexpand\expandafter\noexpand@glssee\noexpand@glo@list{\@glo@label}}%
2174         \@do@glssee
2175     }%

```

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

```

2176     \ifignoredglossary@glo@type
2177     {%
2178         \csdef{glo@\glo@label @index}{}%
2179     }
2180     {%
2181         \do@glo@storeentry{\@glo@label}%
2182     }%

```

Define entry counters if enabled:

```

2183     \@newglossaryentry@defcounters

```

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

```

2184     \@newglossaryentryposthook
2185 }

```

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

```

2186 \newcommand*{\@newglossaryentryprehook}{}

```

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

```

2187 \newcommand*{\@newglossaryentryposthook}{}

```

ryentry@defcounters

```
2188 \newcommand*{\@newglossaryentry@defcounters}{}
```

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

```
2189 \newcommand*{\glsmoveentry}[2]{%
2190   \edef\@glo@thislabel{\glsdetoklabel{#1}}%
2191   \edef\glo@type{\csname glo@\@glo@thislabel @type\endcsname}%
2192   \def\glo@list{,%}
2193   \forglentries[\glo@type]{\glo@label}%
2194   {%
2195     \ifdefequal\@glo@thislabel\glo@label
2196       {\eappto\glo@list{\glo@label,%}}%
2197     }%
2198     \cslet{glolist@\glo@type}{\glo@list}%
2199     \csdef{glo@\@glo@thislabel @type}{#2}%
2200 }
```

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

```
2201 \ifglxindy
2202   \newcommand*{\@glossaryentryfield}{\string\glossentry}
2203 \else
2204   \newcommand*{\@glossaryentryfield}{\string\glossentry}
2205 \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.)

```
2206 \ifglxindy
2207   \newcommand*{\@glossarysubentryfield}{%
2208     \string\subglossentry}
2209 \else
2210   \newcommand*{\@glossarysubentryfield}{%
2211     \string\subglossentry}
2212 \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.)

```
2213 \newcommand{\@glo@storeentry}[1]{%
```

Escape makeindex/xindy special characters in the label:

```
2214 \edef\@glo@esclabel{#1}%
2215 \@gls@checkmkidxchars\@glo@esclabel
```

Get the sort string and escape any special characters

```
2216 \protected@edef\@glo@sort{\csname glo@#1@sort\endcsname}%
2217 \@gls@checkmkidxchars\@glo@sort
```

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

```
2218 \@gls@checkmkidxchars\@glo@prefix
```

Get the parent, if one exists

```
2219 \edef\@glo@parent{\csname glo@#1@parent\endcsname}%
```

Write the information to the glossary file.

```
2220 \ifglxindy
```

Store using xindy syntax.

```
2221 \ifx\@glo@parent\@empty
```

Entry doesn't have a parent

```
2222 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2223 (\string"\@glo@sort\string" %
2224 \string"\@glo@prefix\@glossaryentryfield{\@glo@esclabel}\string") %
2225 }%
2226 \else
```

Entry has a parent

```
2227 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2228 \csname glo@\@glo@parent @index\endcsname
2229 (\string"\@glo@sort\string" %
2230 \string"\@glo@prefix\@glossarysubentryfield
2231 {\csname glo@#1@level\endcsname}{\@glo@esclabel}\string") %
2232 }%
2233 \fi
2234 \else
```

Store using makeindex syntax.

```
2235 \ifx\@glo@parent\@empty
```

Sanitize \@glo@prefix

```
2236 \@onelevel@sanitize\@glo@prefix
```

Entry doesn't have a parent

```
2237 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2238 \@glo@sort\@gls@actualchar\@glo@prefix
2239 \@glossaryentryfield{\@glo@esclabel}%
2240 }%
2241 \else
```

Entry has a parent

```
2242 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2243 \csname glo@\@glo@parent @index\endcsname\@gls@levelchar
```

```

2244     \@glo@sort\@gls@actualchar\@glo@prefix
2245     \@glossarysubentryfield
2246     {\csname glo@#1@level\endcsname}{\@glo@esclabel}%
2247   }%
2248   \fi
2249   \fi
2250 }

```

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`

```

2251 \AtBeginDocument{%
2252   \@ifpackageloaded{amsmath}%
2253   {\let\gls@ifnotmeasuring\@gls@ifnotmeasuring}%
2254   }%
2255 }
2256 \newcommand*\@gls@ifnotmeasuring[1]{%
2257   \ifmeasuring@
2258   \else
2259     #1%
2260   \fi
2261 }
2262 \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.

```

2263 \newcommand*\glsreset[1]{%
2264   \gls@ifnotmeasuring
2265   {%
2266     \glsdoifexists{#1}%
2267     {%
2268       \@glsreset{#1}%
2269     }%
2270   }%
2271 }

```

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

```

2272 \newcommand*\glslocalreset[1]{%
2273   \gls@ifnotmeasuring
2274   {%
2275     \glsdoifexists{#1}%
2276     {%

```



```

2277     \@glslocalreset{#1}%
2278   }%
2279 }%
2280 }

```

`\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.

```

2281 \newcommand*\glsunset}[1]{%
2282   \gls@ifnotmeasuring
2283   {%
2284     \glsdoifexists{#1}%
2285     {%
2286       \@glsunset{#1}%
2287     }%
2288   }%
2289 }

```

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

```

2290 \newcommand*\glslocalunset}[1]{%
2291   \gls@ifnotmeasuring
2292   {%
2293     \glsdoifexists{#1}%
2294     {%
2295       \@glslocalunset{#1}%
2296     }%
2297   }%
2298 }

```

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

```

2299 \newcommand*\@glslocalunset{\@@glslocalunset}

```

`\@@glslocalunset` Local unset without checks.

```

2300 \newcommand*\@@glslocalunset}[1]{%
2301   \expandafter\let\csname ifglo@glsdetoklabel{#1}@flag\endcsname\iftrue
2302 }

```

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

```

2303 \newcommand*\@glsunset{\@@glsunset}

```

`\@@glsunset` Global unset without checks.

```

2304 \newcommand*\@@glsunset}[1]{%
2305   \expandafter\global\csname glo@glsdetoklabel{#1}@flagtrue\endcsname
2306 }

```

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

```

2307 \newcommand*\@glslocalreset{\@@glslocalreset}

```

`\@glslocalreset` Local reset without checks.

```
2308 \newcommand*\@glslocalreset}[1]{%
2309   \expandafter\let\csname ifglo@glsdetoklabel{#1}@flag\endcsname\iffalse
2310 }
```

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

```
2311 \newcommand*\glsreset{\@glsreset}
```

`\@glsreset` Global reset without checks.

```
2312 \newcommand*\@glsreset}[1]{%
2313   \expandafter\global\csname glo@glsdetoklabel{#1}@flagfalse\endcsname
2314 }
```

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

`\glsresetall`

```
2315 \newcommand*\glsresetall}[1] [\@glo@types]{%
2316   \forallglsentries[#1]{\@glsentry}%
2317   {%
2318     \glsreset{\@glsentry}%
2319   }%
2320 }
```

As above, but with only a local effect:

`\glslocalresetall`

```
2321 \newcommand*\glslocalresetall}[1] [\@glo@types]{%
2322   \forallglsentries[#1]{\@glsentry}%
2323   {%
2324     \glslocalreset{\@glsentry}%
2325   }%
2326 }
```

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

`\glsunsetall`

```
2327 \newcommand*\glsunsetall}[1] [\@glo@types]{%
2328   \forallglsentries[#1]{\@glsentry}%
2329   {%
2330     \glsunset{\@glsentry}%
2331   }%
2332 }
```

As above, but with only a local effect:

`\glslocalunsetall`

```
2333 \newcommand*\glslocalunsetall}[1][\@glo@types]{%
2334   \forallglsentries[#1]{\@glsentry}%
2335   {%
2336     \glslocalunset{\@glsentry}%
2337   }%
2338 }
```

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

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

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

```
2339 \newcommand*\@newglossaryentry@defcounters){%
2340   \csdef{glo@\@glo@label @currcount}{0}%
2341   \csdef{glo@\@glo@label @prevcount}{0}%
2342 }
```

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

```
2343 \newcommand*\glsenableentrycount){%
```

Enable new entry fields.

```
2344 \let\@newglossaryentry@defcounters\@newglossaryentry@defcounters
```

Disable `\newglossaryentry` in the document environment.

```
2345 \renewcommand*\gls@defdocnewglossaryentry){%
2346   \renewcommand*\newglossaryentry[2]{%
2347     \PackageError{glossaries}{\string\newglossaryentry\space
2348     may only be used in the preamble when entry counting has
2349     been activated}{If you use \string\glsenableentrycount\space
2350     you must place all entry definitions in the preamble not in
2351     the document environment}%
2352   }%
2353 }
```

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

```
2354 \newcommand*\glsentrycurrcount}[1]{%
2355   \ifcsundef{glo@\glsdetoklabel{##1}@currcount}%
2356   {0}{\@gls@entry@field{##1}{currcount}}%
2357 }
```

```

2358 \newcommand*\glsentryprevcount}[1]{%
2359   \ifcsundef{glo@glsdetoklabel{##1}@prevcount}%
2360   {0}{\@gls@entry@field{##1}{prevcount}}%
2361 }%

```

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

```

2362 \renewcommand*\@glsunset}[1]{%
2363   \@glsunset{##1}%
2364   \@gls@increment@currcount{##1}%
2365 }%
2366 \renewcommand*\@glslocalunset}[1]{%
2367   \@glslocalunset{##1}%
2368   \@gls@local@increment@currcount{##1}%
2369 }%
2370 \renewcommand*\@glsreset}[1]{%
2371   \@glsreset{##1}%
2372   \csgdef{glo@glsdetoklabel{##1}@currcount}{0}%
2373 }%
2374 \renewcommand*\@glslocalreset}[1]{%
2375   \@glslocalreset{##1}%
2376   \csdef{glo@glsdetoklabel{##1}@currcount}{0}%
2377 }%

```

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.)

```

2378 \def\@cgl's@##1##2[##3]{%
2379   \ifnum\glsentryprevcount{##2}=1\relax
2380     \cgl'sformat{##2}{##3}%
2381     \glsunset{##2}%
2382   \else
2383     \@gls@{##1}{##2}[##3]%
2384   \fi
2385 }%

```

Similarly for the analogous commands. No case change plural:

```

2386 \def\@cgl'spl@##1##2[##3]{%
2387   \ifnum\glsentryprevcount{##2}=1\relax
2388     \cgl'splformat{##2}{##3}%
2389     \glsunset{##2}%
2390   \else
2391     \@glspl@{##1}{##2}[##3]%
2392   \fi
2393 }%

```

First letter uppercase singular:

```

2394 \def\@cGls@##1##2[##3]{%
2395   \ifnum\glsentryprevcount{##2}=1\relax
2396     \cGlsformat{##2}{##3}%
2397     \glsunset{##2}%
2398   \else

```

```

2399 \Gls@{##1}{##2}[##3]%
2400 \fi
2401 }%

```

First letter uppercase plural:

```

2402 \def\cGlspl@##1##2[##3]{%
2403 \ifnum\glsentryprevcount{##2}=1\relax
2404 \cGlsplformat{##2}{##3}%
2405 \glsunset{##2}%
2406 \else
2407 \Glspl@{##1}{##2}[##3]%
2408 \fi
2409 }%

```

Write information to aux file at the end of the document

```

2410 \AtEndDocument{\@gls@write@entrycounts}%

```

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

```

2411 \renewcommand*\@gls@entry@count}[2]{%
2412 \csgdef{glo@glsdetoklabel{##1}@prevcount}{##2}%
2413 }%

```

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

```

2414 \let\glsenableentrycount\relax
2415 }
2416 \@onlypreamble\glsenableentrycount

```

increment@currcount

```

2417 \newcommand*\@gls@increment@currcount}[1]{%
2418 \csxdef{glo@glsdetoklabel{##1}@currcount}{%
2419 \number\numexpr\glsentrycurrcount{##1}+1}%
2420 }

```

increment@currcount

```

2421 \newcommand*\@gls@local@increment@currcount}[1]{%
2422 \csedef{glo@glsdetoklabel{##1}@currcount}{%
2423 \number\numexpr\glsentrycurrcount{##1}+1}%
2424 }

```

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.)

```

2425 \newcommand*\@gls@write@entrycounts){%
2426 \immediate\write\@auxout
2427 {\string\providecommand*\string\@gls@entry@count}[2]{}}%
2428 \forallglsentries{\@glsentry}{%
2429 \ifglsused{\@glsentry}%

```

```

2430     {\immediate\write\@auxout
2431       {\string\@gls@entry@count{\@glsentry}{\glsentrycurrcount{\@glsentry}}}}%
2432     {}%
2433   }%
2434 }

```

`\@gls@entry@count` Default behaviour is to ignore arguments. Activated by `\glsenableentrycount`.

```

2435 \newcommand*{\@gls@entry@count}[2]{}

```

`\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.)

```

2436 \newrobustcmd*{\cgl}{\@gls@hyp@opt\@cgl}

```

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

```

2437 \newcommand*{\@cgl}[2] []{%
2438   \new@ifnextchar[{\@cgl@{#1}{#2}}{\@cgl@{#1}{#2} []}%
2439 }

```

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

```

2440 \def\@cgl@#1#2[#3]{%
2441   \GlossariesWarning{\string\cgl\space is defaulting to
2442     \string\gls\space since you haven't enabled entry counting}%
2443   \@gls@{#1}{#2}[#3]%
2444 }

```

`\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.

```

2445 \newcommand*{\cglformat}[2]{%
2446   \ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}#2%
2447 }

```

`\cG`ls 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.)

```

2448 \newrobustcmd*{\cG}{\@gls@hyp@opt\@cG}

```

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

```

2449 \newcommand*{\@cG}[2] []{%
2450   \new@ifnextchar[{\@cG@{#1}{#2}}{\@cG@{#1}{#2} []}%
2451 }

```

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

```

2452 \def\cGls@#1#2[#3]{%
2453 \GlossariesWarning{\string\cGls\space is defaulting to
2454 \string\Gls\space since you haven't enabled entry counting}%
2455 \cGls@{#1}{#2}[#3]%
2456 }

```

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

```

2457 \newcommand*\cGlsformat}[2]{%
2458 \ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}#2%
2459 }

```

`\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.)

```

2460 \newrobustcmd*\cglsp1{\@gls@hyp@opt\@cglsp1}

```

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

```

2461 \newcommand*\@cglsp1}[2][ ]{%
2462 \new@ifnextchar[{\@cglsp1@{#1}{#2}}{\@cglsp1@{#1}{#2} [ ]}%
2463 }

```

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

```

2464 \def\@cglsp1@#1#2[#3]{%
2465 \GlossariesWarning{\string\cglsp1\space is defaulting to
2466 \string\glsp1\space since you haven't enabled entry counting}%
2467 \@cglsp1@{#1}{#2}[#3]%
2468 }

```

`\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.

```

2469 \newcommand*\cglsp1format}[2]{%
2470 \ifglshaslong{#1}{\Glsentrylongpl{#1}}{\Glsentryfirstplural{#1}}#2%
2471 }

```

`\cG1sp1` Define command that works like `\G1sp1` but behaves differently if the entry count function is enabled. (If not enabled, it behaves the same as `\G1sp1` but issues a warning.)

```

2472 \newrobustcmd*\cG1sp1{\@gls@hyp@opt\@cG1sp1}

```

`\@cG1sp1` Defined the un-starred form. Need to determine if there is a final optional argument

```

2473 \newcommand*\@cG1sp1}[2][ ]{%
2474 \new@ifnextchar[{\@cG1sp1@{#1}{#2}}{\@cG1sp1@{#1}{#2} [ ]}%
2475 }

```

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

```
2476 \def\cGlspl@#1#2[#3]{%
2477 \GlossariesWarning{\string\cGlspl\space is defaulting to
2478 \string\Glspl\space since you haven't enabled entry counting}%
2479 \cGlspl@{#1}{#2}[#3]%
2480 }
```

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

```
2481 \newcommand*\cGlsplformat}[2]{%
2482 \ifglshaslong{#1}{\Glsentrylongpl{#1}}{\Glsentryfirstplural{#1}}#2%
2483 }
```

1.10 Loading files containing glossary entries

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

`\loadglsentries` [*type*] {*filename*}

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

`\loadglsentries`

```
2484 \newcommand*\loadglsentries}[2][\@gls@default]{%
2485 \let\@gls@default\glsdefaulttype
2486 \def\glsdefaulttype{#1}\input{#2}%
2487 \let\glsdefaulttype\@gls@default
2488 }
```

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

```
2489 \@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

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

commands (such as `\textbf` is governed by `\glstextformat`. By default this just displays the link text “as is”.

`\glstextformat`

```
2490 \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

```
2491 \newcommand*{\glsentryfmt}{%
2492   \@@gls@default@entryfmt\glsdisplayfirst\glsdisplay
2493 }
```

Format that provides backwards compatibility:

```
2494 \newcommand*{\@@gls@default@entryfmt}[2]{%
2495   \ifdefempty\glscustomtext
2496     {%
2497       \glsifplural
2498       {%
```

Plural form

```
2499     \glscapscase
2500     {%
```

Don't adjust case

```
2501     \ifglsused\glslabel
2502     {%
```

Subsequent use

```
2503         #2{\glsentryplural{\glslabel}}%
2504         {\glsentrydescplural{\glslabel}}%
2505         {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2506     }%
2507     {%
```

First use

```
2508         #1{\glsentryfirstplural{\glslabel}}%
2509         {\glsentrydescplural{\glslabel}}%
2510         {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2511     }%
2512     }%
2513     {%
```

Make first letter upper case

```
2514     \ifglsused\glslabel
2515     {%
```

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.)

```

2516     \ifbool{glscompatible-3.07}%
2517     {%
2518         \protected@edef\@glo@etext{%
2519             #2{\glsentryplural{\glslabel}}%
2520             {\glsentrydescplural{\glslabel}}%
2521             {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2522         \xmakefirstuc\@glo@etext
2523     }%
2524     {%
2525         #2{\Glsentryplural{\glslabel}}%
2526         {\glsentrydescplural{\glslabel}}%
2527         {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2528     }%
2529     }%
2530     {%

```

First use

```

2531     \ifbool{glscompatible-3.07}%
2532     {%
2533         \protected@edef\@glo@etext{%
2534             #1{\glsentryfirstplural{\glslabel}}%
2535             {\glsentrydescplural{\glslabel}}%
2536             {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2537         \xmakefirstuc\@glo@etext
2538     }%
2539     {%
2540         #1{\Glsentryfirstplural{\glslabel}}%
2541         {\glsentrydescplural{\glslabel}}%
2542         {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2543     }%
2544     }%
2545     }%
2546     {%

```

Make all upper case

```

2547     \ifglsused\glslabel
2548     {%

```

Subsequent use

```

2549         \mfirstucMakeUppercase{#2{\glsentryplural{\glslabel}}%
2550         {\glsentrydescplural{\glslabel}}%
2551         {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2552     }%
2553     {%

```

First use

```

2554         \mfirstucMakeUppercase{#1{\glsentryfirstplural{\glslabel}}%
2555         {\glsentrydescplural{\glslabel}}%
2556         {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%

```

2557 }%
 2558 }%
 2559 }%
 2560 {%

Singular form

2561 \glscapscase
 2562 {%

Don't adjust case

2563 \ifglsused\glslabel
 2564 {%

Subsequent use

2565 #2{\glsentrytext{\glslabel}}%
 2566 {\glsentrydesc{\glslabel}}%
 2567 {\glsentrysymbol{\glslabel}}{\glsinsert}%
 2568 }%
 2569 {%

First use

2570 #1{\glsentryfirst{\glslabel}}%
 2571 {\glsentrydesc{\glslabel}}%
 2572 {\glsentrysymbol{\glslabel}}{\glsinsert}%
 2573 }%
 2574 }%
 2575 {%

Make first letter upper case

2576 \ifglsused\glslabel
 2577 {%

Subsequent use

2578 \ifbool{glscompatible-3.07}%
 2579 {%
 2580 \protected@edef\@glo@etext{%
 2581 #2{\glsentrytext{\glslabel}}%
 2582 {\glsentrydesc{\glslabel}}%
 2583 {\glsentrysymbol{\glslabel}}{\glsinsert}}%
 2584 \xmakefirstuc\@glo@etext
 2585 }%
 2586 {%
 2587 #2{\Glsentrytext{\glslabel}}%
 2588 {\glsentrydesc{\glslabel}}%
 2589 {\glsentrysymbol{\glslabel}}{\glsinsert}}%
 2590 }%
 2591 }%
 2592 {%

First use

2593 \ifbool{glscompatible-3.07}%
 2594 {%

```

2595         \protected@edef\@glo@etext{%
2596             #1{\glsentryfirst{\glslabel}}}%
2597             {\glsentrydesc{\glslabel}}}%
2598             {\glsentrysymbol{\glslabel}}{\glsinsert}}}%
2599         \xmakefirstuc\@glo@etext
2600     }%
2601     {%
2602         #1{\Glsentryfirst{\glslabel}}}%
2603         {\glsentrydesc{\glslabel}}}%
2604         {\glsentrysymbol{\glslabel}}{\glsinsert}}}%
2605     }%
2606 }%
2607 }%
2608 {%

```

Make all upper case

```

2609     \ifglsused\glslabel
2610     {%

```

Subsequent use

```

2611         \mfirstucMakeUppercase{#2{\glsentrytext{\glslabel}}}%
2612         {\glsentrydesc{\glslabel}}}%
2613         {\glsentrysymbol{\glslabel}}{\glsinsert}}}%
2614     }%
2615     {%

```

First use

```

2616         \mfirstucMakeUppercase{#1{\glsentryfirst{\glslabel}}}%
2617         {\glsentrydesc{\glslabel}}}%
2618         {\glsentrysymbol{\glslabel}}{\glsinsert}}}%
2619     }%
2620 }%
2621 }%
2622 }%
2623 {%

```

Custom text provided in \glsdisp

```

2624     \ifglsused{\glslabel}%
2625     {%

```

Subsequent use

```

2626         #2{\glscustomtext}%
2627         {\glsentrydesc{\glslabel}}}%
2628         {\glsentrysymbol{\glslabel}}{\glsinsert}}}%
2629     }%
2630     {%

```

First use

```

2631         #1{\glscustomtext}%
2632         {\glsentrydesc{\glslabel}}}%
2633         {\glsentrysymbol{\glslabel}}{\glsinsert}}}%
2634     }%

```

```
2635 }%
2636 }
```

`\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.

```
2637 \newcommand*{\glsgenentryfmt}{%
2638   \ifdefempty\glscustomtext
2639   {%
2640     \glsifplural
2641     {%
```

Plural form

```
2642     \glscapscase
2643     {%
```

Don't adjust case

```
2644     \ifglsused\glslabel
2645     {%
```

Subsequent use

```
2646     \glsentryplural{\glslabel}\glsinsert
2647     }%
2648     {%
```

First use

```
2649     \glsentryfirstplural{\glslabel}\glsinsert
2650     }%
2651     }%
2652     {%
```

Make first letter upper case

```
2653     \ifglsused\glslabel
2654     {%
```

Subsequent use.

```
2655     \Glsentryplural{\glslabel}\glsinsert
2656     }%
2657     {%
```

First use

```
2658     \Glsentryfirstplural{\glslabel}\glsinsert
2659     }%
2660     }%
2661     {%
```

Make all upper case

```
2662     \ifglsused\glslabel
2663     {%
```

Subsequent use

```
2664     \mfirstucMakeUppercase
2665     {\glsentryplural{\glslabel}\glsinsert}%
2666     }%
2667     {%
```

First use

```
2668      \mfirstucMakeUppercase
2669      {\glsentryfirstplural{\glslabel}\glsinsert}%
2670      }%
2671      }%
2672      }%
2673      {%
```

Singular form

```
2674      \glscapscase
2675      {%
```

Don't adjust case

```
2676      \ifglsused\glslabel
2677      {%
```

Subsequent use

```
2678      \glsentrytext{\glslabel}\glsinsert
2679      }%
2680      {%
```

First use

```
2681      \glsentryfirst{\glslabel}\glsinsert
2682      }%
2683      }%
2684      {%
```

Make first letter upper case

```
2685      \ifglsused\glslabel
2686      {%
```

Subsequent use

```
2687      \Glsentrytext{\glslabel}\glsinsert
2688      }%
2689      {%
```

First use

```
2690      \Glsentryfirst{\glslabel}\glsinsert
2691      }%
2692      }%
2693      {%
```

Make all upper case

```
2694      \ifglsused\glslabel
2695      {%
```

Subsequent use

```
2696      \mfirstucMakeUppercase{\glsentrytext{\glslabel}\glsinsert}%
2697      }%
2698      {%
```

First use

```
2699      \mfirstucMakeUppercase{\glsentryfirst{\glslabel}\glsinsert}%
```

```

2700     }%
2701     }%
2702     }%
2703     }%
2704     {%

```

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

```

2705     \glscustomtext\glsinsert
2706     }%
2707 }

```

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

```

2708 \newcommand*{\glsngenacfmt}{%
2709   \ifdefempty\glscustomtext
2710   {%
2711     \ifglsused\glslabel
2712     {%

```

Subsequent use:

```

2713     \glsifplural
2714     {%

```

Subsequent plural form:

```

2715     \gls caps case
2716     {%

```

Subsequent plural form, don't adjust case:

```

2717     \acronymfont{\glsentryshortpl{\glslabel}}\glsinsert
2718     }%
2719     {%

```

Subsequent plural form, make first letter upper case:

```

2720     \acronymfont{\Glsentryshortpl{\glslabel}}\glsinsert
2721     }%
2722     {%

```

Subsequent plural form, all caps:

```

2723     \mfirstucMakeUppercase
2724     {\acronymfont{\glsentryshortpl{\glslabel}}\glsinsert}%
2725     }%
2726     }%
2727     {%

```

Subsequent singular form

```

2728     \gls caps case
2729     {%

```

Subsequent singular form, don't adjust case:

```

2730     \acronymfont{\glsentryshort{\glslabel}}\glsinsert
2731     }%
2732     {%

```

Subsequent singular form, make first letter upper case:

```
2733      \acronymfont{\Glsentryshort{\glslabel}}\glsinsert
2734      }%
2735      {%
```

Subsequent singular form, all caps:

```
2736      \mfirstucMakeUppercase
2737      {\acronymfont{\Glsentryshort{\glslabel}}\glsinsert}%
2738      }%
2739      }%
2740      }%
2741      {%
```

First use:

```
2742      \glsifplural
2743      {%
```

First use plural form:

```
2744      \gls caps case
2745      {%
```

First use plural form, don't adjust case:

```
2746      \genplacrfullformat{\glslabel}{\glsinsert}%
2747      }%
2748      {%
```

First use plural form, make first letter upper case:

```
2749      \Genplacrfullformat{\glslabel}{\glsinsert}%
2750      }%
2751      {%
```

First use plural form, all caps:

```
2752      \mfirstucMakeUppercase
2753      {\genplacrfullformat{\glslabel}{\glsinsert}}%
2754      }%
2755      }%
2756      {%
```

First use singular form

```
2757      \gls caps case
2758      {%
```

First use singular form, don't adjust case:

```
2759      \genacrfullformat{\glslabel}{\glsinsert}%
2760      }%
2761      {%
```

First use singular form, make first letter upper case:

```
2762      \Genacrfullformat{\glslabel}{\glsinsert}%
2763      }%
2764      {%
```


First use singular form, all caps:

```
2765     \mfirstucMakeUppercase
2766     {\genacrfullformat{\glslabel}{\glsinsert}}%
2767     }%
2768     }%
2769     }%
2770     }%
2771     {%
```

User supplied text.

```
2772     \glscustomtext
2773     }%
2774 }
```

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

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

```
2775 \newcommand*{\genacrfullformat}[2]{%
2776     \glsentrylong{#1}#2\space
2777     (\protect\firstacronymfont{\glsentryshort{#1}})%
2778 }
```

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

As above but makes the first letter upper case.

```
2779 \newcommand*{\Genacrfullformat}[2]{%
2780     \protected@edef\gls@text{\genacrfullformat{#1}{#2}}%
2781     \xmakefirstuc\gls@text
2782 }
```

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

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

```
2783 \newcommand*{\genplacrfullformat}[2]{%
2784     \glsentrylongpl{#1}#2\space
2785     (\protect\firstacronymfont{\glsentryshortpl{#1}})%
2786 }
```

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

As above but makes the first letter upper case.

```
2787 \newcommand*{\Genplacrfullformat}[2]{%
2788     \protected@edef\gls@text{\genplacrfullformat{#1}{#2}}%
```

```
2789 \xmakefirstuc\gls@text
2790 }
```

`\glsdisplayfirst` Deprecated. Kept for backward compatibility.

```
2791 \newcommand*{\glsdisplayfirst}[4]{#1#4}
```

`\glsdisplay` Deprecated. Kept for backward compatibility.

```
2792 \newcommand*{\glsdisplay}[4]{#1#4}
```

`\defglsdisplay` Deprecated. Kept for backward compatibility.

```
2793 \newcommand*{\defglsdisplay}[2][\glsdefaulttype]{%
2794 \GlossariesWarning{\string\defglsdisplay\space is now obsolete.^^J
2795 Use \string\defglsentryfmt\space instead}%
2796 \expandafter\def\csname gls@#1@display\endcsname##1##2##3##4{#2}%
2797 \edef\@gls@doentrydef{%
2798 \noexpand\defglsentryfmt[#1]{%
2799 \noexpand\ifcsdef{gls@#1@displayfirst}%
2800 {%
2801 \noexpand\@gls@default@entryfmt
2802 {\noexpand\csuse{gls@#1@displayfirst}}}%
2803 {\noexpand\csuse{gls@#1@display}}}%
2804 }%
2805 {%
2806 \noexpand\@gls@default@entryfmt
2807 {\noexpand\glsdisplayfirst}%
2808 {\noexpand\csuse{gls@#1@display}}}%
2809 }%
2810 }%
2811 }%
2812 \@gls@doentrydef
2813 }
```

`\defglsdisplayfirst` Deprecated. Kept for backward compatibility.

```
2814 \newcommand*{\defglsdisplayfirst}[2][\glsdefaulttype]{%
2815 \GlossariesWarning{\string\defglsdisplayfirst\space is now obsolete.^^J
2816 Use \string\defglsentryfmt\space instead}%
2817 \expandafter\def\csname gls@#1@displayfirst\endcsname##1##2##3##4{#2}%
2818 \edef\@gls@doentrydef{%
2819 \noexpand\defglsentryfmt[#1]{%
2820 \noexpand\ifcsdef{gls@#1@display}%
2821 {%
2822 \noexpand\@gls@default@entryfmt
2823 {\noexpand\csuse{gls@#1@displayfirst}}}%
2824 {\noexpand\csuse{gls@#1@display}}}%
2825 }%
2826 {%
2827 \noexpand\@gls@default@entryfmt
2828 {\noexpand\csuse{gls@#1@displayfirst}}}%
2829 }%
2830 }
```

```

2829         {\noexpand\glsdisplay}%
2830     }%
2831 }%
2832 }%
2833 \@gls@doentrydef
2834 }

```

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 \TeX norm to have an optional argument after the mandatory arguments, but it makes more sense to write, say, `\gls{label}[’s]` rather than, say, `\gls[append=’s]{label}`. Since these control sequences are defined to include the final square bracket, spaces will be ignored after them. This is likely to lead to confusion as most users would not expect, say, `\gls{<label>}` to ignore following spaces, so `\new@ifnextchar` from the package is required.

The following keys can be used in the first optional argument. The counter key checks that the value is the name of a valid counter.

```

2835 \define@key{glslink}{counter}{%
2836   \ifcsundef{c@#1}%
2837   {%
2838     \PackageError{glossaries}%
2839     {There is no counter called ‘#1’}%
2840     {%
2841       The counter key should have the name of a valid counter
2842       as its value%
2843     }%
2844   }%
2845   {%
2846     \def\@gls@counter{#1}%
2847   }%
2848 }

```

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.

```

2849 \define@key{glslink}{format}{%
2850   \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.

```

2851 \define@boolkey{glslink}{hyper}[true]{}

```

Initialise hyper key.

```
2852 \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.

```
2853 \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.

```
2854 \newcommand*{\glslinkvar}[3]{#1}
```

`\glsifhyper` Now deprecated.

```
2855 \newcommand*{\glsifhyper}[2]{%
```

```
2856 \glslinkvar{#1}{#2}{#1}%
```

```
2857 \GlossariesWarning{string\glsifhyper\space is deprecated. Did
```

```
2858 you mean \string\glsifhyperon\space or \string\glslinkvar?}%
```

```
2859 }
```

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

```
2860 \newcommand*{\@gls@hyp@opt}[1]{%
```

```
2861 \let\glslinkvar\@firstofthree
```

```
2862 \let\@gls@hyp@opt@cs#1\relax
```

```
2863 \ifstar{\s@gls@hyp@opt}%
```

```
2864 {\@ifnextchar+{\@firstoftwo{\p@gls@hyp@opt}}{#1}}%
```

```
2865 }
```

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

```
2866 \newcommand*{\s@gls@hyp@opt}[1][[]]{%
```

```
2867 \let\glslinkvar\@secondofthree
```

```
2868 \@gls@hyp@opt@cs[hyper=false,#1]}
```

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

```
2869 \newcommand*{\p@gls@hyp@opt}[1][[]]{%
```

```
2870 \let\glslinkvar\@thirdofthree
```

```
2871 \@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`

```
2872 \newrobustcmd*{\glslink}{%
2873 \@gls@hyp@opt\@gls@link
2874 }
```

`\@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.

```
2875 \newcommand*{\@gls@link}[3] []{%
2876 \ifglsentryexists{#2}%
2877 {%
2878 \let\do@gls@link@checkfirsthyper\relax
2879 \@gls@link[#1]{#2}{#3}%
2880 }{%
2881 \PackageError{glossaries}{Glossary entry ‘#2’ has not been
2882 defined}{You need to define a glossary entry before you
2883 can use it.}%

```

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

```
2884 \glstextformat{#3}%
2885 }%
2886 }
```

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

```
2887 \newcommand*{\@gls@link@checkfirsthyper}{%
2888 \ifglsused{\glslabel}%
2889 {%
2890 }%
2891 {%
2892 \gls@checkisacronymlist\glstype

```

```

2893 \ifglshyperfirst
2894   \if@glsisacronymlist
2895     \ifglssacrfootnote
2896       \KV@glslink@hyperfalse
2897     \fi
2898   \fi
2899 \else
2900   \KV@glslink@hyperfalse
2901 \fi
2902 }%

```

Allow user to hook into this

```

2903 \glslinkcheckfirsthyperhook
2904 }

```

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

```

2905 \newcommand*{\glslinkcheckfirsthyperhook}{}

```

\@gls@link

```

2906 \def\@gls@link[#1]#2#3{%

```

Inserting \leavevmode suggested by Donald Arseneau (avoids problem with tabularx).

```

2907   \leavevmode
2908   \edef\glslabel{\glsdetoklabel{#2}}%

```

Save options in \@gls@link@opts and label in \@gls@link@label

```

2909   \def\@gls@link@opts{#1}%
2910   \let\@gls@link@label\glslabel

```

```

2911   \def\@glsnumberformat{glsnumberformat}%
2912   \edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%

```

If this is in one of the “nohypertypes” glossaries, suppress the hyperlink by default

```

2913   \edef\glstype{\csname glo@\glslabel @type\endcsname}%

```

Save original setting

```

2914   \let\org@ifKV@glslink@hyper\ifKV@glslink@hyper

```

Switch off hyper setting if the glossary type has been identified in nohyperlist.

```

2915   \expandafter\DTLifinlist\expandafter
2916   {\glstype}{\@gls@nohyperlist}%
2917   {%
2918     \KV@glslink@hyperfalse
2919   }%
2920   {%
2921   }%

```

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.

```

2922 \do@gls@link@checkfirsthyper
2923 \setkeys{glslink}{#1}%
    Define \glsifhyperon
2924 \ifKV@glslink@hyper
2925 \let\glsifhyperon\@firstoftwo
2926 \else
2927 \let\glsifhyperon\@secondoftwo
2928 \fi
    Store the entry's counter in \theglsentrycounter
2929 \@gls@saveentrycounter
    Define sort key if necessary:
2930 \@gls@setsort{\glslabel}%
    (De-tok'ing done by \@do@wrglossary)
2931 \@do@wrglossary{#2}%
2932 \ifKV@glslink@hyper
2933 \@glslink{\glolinkprefix\glslabel}{\glstextformat{#3}}%
2934 \else
2935 \glstextformat{#3}%
2936 \fi
    Restore original setting
2937 \let\ifKV@glslink@hyper\org@ifKV@glslink@hyper
2938 }

```

`\glolinkprefix`

```
2939 \newcommand*{\glolinkprefix}{glo:}
```

`\glsentrycounter` Set default value of entry counter

```
2940 \def\glsentrycounter{\glscounter}%
```

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

```
2941 \newcommand*{\@gls@saveentrycounter}{%
```

```
2942 \def\@gls@Hcounter}{%
```

Are we using equation counter?

```
2943 \ifthenelse{\equal{\@gls@counter}{equation}}{%
```

```
2944 {
```

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

```
2945 \ifcsundef{xatlevel@}{%
```

```
2946 {%
```

```
2947 \edef\theglsentrycounter{\expandafter\noexpand
```

```
2948 \csname the\@gls@counter\endcsname}%
```

```
2949 }%
```

```
2950 {%
```

```

2951     \ifx\xatlevel@\empty
2952     \edef\theglentrycounter{\expandafter\noexpand
2953     \csname the\@gls@counter\endcsname}%
2954     \else
2955     \savecounters@
2956     \advance\c@equation by 1\relax
2957     \edef\theglentrycounter{\csname the\@gls@counter\endcsname}%

```

Check if hyperref version of this counter

```

2958     \ifcsundef{theH\@gls@counter}%
2959     {%
2960     \def\@gls@Hcounter{\theglentrycounter}%
2961     }%
2962     {%
2963     \def\@gls@Hcounter{\csname theH\@gls@counter\endcsname}%
2964     }%
2965     \protected@edef\theHglentrycounter{\@gls@Hcounter}%
2966     \restorecounters@
2967     \fi
2968     }%
2969     }%
2970     {%

```

Not using equation counter so no special measures:

```

2971     \edef\theglentrycounter{\expandafter\noexpand
2972     \csname the\@gls@counter\endcsname}%
2973     }%

```

Check if hyperref version of this counter

```

2974     \ifx\@gls@Hcounter@\empty
2975     \ifcsundef{theH\@gls@counter}%
2976     {%
2977     \def\theHglentrycounter{\theglentrycounter}%
2978     }%
2979     {%
2980     \protected@edef\theHglentrycounter{\expandafter\noexpand
2981     \csname theH\@gls@counter\endcsname}%
2982     }%
2983     \fi
2984     }

```

`\@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.

```

2985 \def\@set@glo@numformat#1#2#3#4{%
2986 \expandafter\@glo@check@mkidxrangechar#3\@nil
2987 \protected@edef#1{%

```



```

2988 \@glo@prefix setentrycounter[#4]{#2}%
2989 \expandafter\string\csname\@glo@suffix\endcsname
2990 }%
2991 \@gls@checkmkidxchars#1%
2992 }

```

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.

```

2993 \def\@glo@check@mkidxrangechar#1#2\@nil{%
2994 \if#1(\relax
2995 \def\@glo@prefix{)%
2996 \if\relax#2\relax
2997 \def\@glo@suffix{glsnumberformat}%
2998 \else
2999 \def\@glo@suffix{#2}%
3000 \fi
3001 \else
3002 \if#1)\relax
3003 \def\@glo@prefix{)}%
3004 \if\relax#2\relax
3005 \def\@glo@suffix{glsnumberformat}%
3006 \else
3007 \def\@glo@suffix{#2}%
3008 \fi
3009 \else
3010 \def\@glo@prefix{}\def\@glo@suffix{#1#2}%
3011 \fi
3012 \fi}

```

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

```

3013 \newcommand*\@gls@escbsdq[1]{%
3014 \def\@gls@checkedmkidx{}%
3015 \let\gls@xdystring=#1\relax
3016 \@onelevel@sanitize\gls@xdystring
3017 \edef\do@gls@xdycheckbackslash{%
3018 \noexpand\@gls@xdycheckbackslash\gls@xdystring\noexpand\@nil
3019 \@backslashchar\@backslashchar\noexpand\@null}%
3020 \do@gls@xdycheckbackslash
3021 \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdystring}%
3022 \def\@gls@checkedmkidx{}%
3023 \expandafter\@gls@xdycheckquote\gls@xdystring\@nil""\null
3024 \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdystring}%

```

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

```

3025 \@for\@gls@tmp:=\gls@protected@pagefmts\do

```

```

3026  {%
3027    \edef\@gls@sanitized@tmp{\expandafter\@gobble\string\\expandonce\@gls@tmp}%
3028    \@onelevel@sanitize\@gls@sanitized@tmp
3029    \edef\gls@dostsubst{%
3030      \noexpand\DTLsubstituteall\noexpand\gls@xdystring
3031      {\@gls@sanitized@tmp}{\expandonce\@gls@tmp}%
3032    }%
3033    \gls@dostsubst
3034  }%

  Assign to required control sequence
3035  \let#1=\gls@xdystring
3036 }

```

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

`\gls@checkmkidxchars`

```

3037 \newcommand{\@gls@checkmkidxchars}[1]{%
3038   \ifglsexindy
3039     \@gls@escbsdq{#1}%
3040   \else
3041     \def\@gls@checkedmkidx{%
3042       \expandafter\@gls@checkquote#1\@nil""\null
3043       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3044     \def\@gls@checkedmkidx{%
3045       \expandafter\@gls@checkescquote#1\@nil""\null
3046       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3047     \def\@gls@checkedmkidx{%
3048       \expandafter\@gls@checkescactual#1\@nil"??\null
3049       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3050     \def\@gls@checkedmkidx{%
3051       \expandafter\@gls@checkactual#1\@nil??\null
3052       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3053     \def\@gls@checkedmkidx{%
3054       \expandafter\@gls@checkbar#1\@nil||\null
3055       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3056     \def\@gls@checkedmkidx{%
3057       \expandafter\@gls@checkescbar#1\@nil\\|\null
3058       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3059     \def\@gls@checkedmkidx{%
3060       \expandafter\@gls@checklevel#1\@nil!!\null
3061       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3062     \fi
3063 }

```

Update the control sequence and strip trailing `\@nil`:

`\@gls@updatechecked`

```

3064 \def\@gls@updatechecked#1\@nil#2{\def#2{#1}}

```

\@gls@tmpb Define temporary token

3065 \newtoks \@gls@tmpb

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

```
3066 \def \@gls@checkquote#1"#2"#3\null{%
3067   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3068   \toks@={#1}%
3069   \ifx\null#2\null
3070   \ifx\null#3\null
3071   \edef \@gls@checkedmkidx{\the \@gls@tmpb\the \toks@}%
3072   \def \@gls@checkquote{\relax}%
3073   \else
3074   \edef \@gls@checkedmkidx{\the \@gls@tmpb\the \toks@
3075     \@gls@quotechar \@gls@quotechar \@gls@quotechar \@gls@quotechar}%
3076   \def \@gls@checkquote{\@gls@checkquote#3\null}%
3077   \fi
3078   \else
3079   \edef \@gls@checkedmkidx{\the \@gls@tmpb\the \toks@
3080     \@gls@quotechar \@gls@quotechar}%
3081   \ifx\null#3\null
3082   \def \@gls@checkquote{\@gls@checkquote#2""\null}%
3083   \else
3084   \def \@gls@checkquote{\@gls@checkquote#2"#3\null}%
3085   \fi
3086   \fi
3087   \@gls@checkquote
3088 }
```

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

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

```

```

3109 \fi
3110 \fi
3111 \@gls@checkescquote
3112 }

```

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

```

3113 \def\@gls@checkescactual#1\?#2\?#3\null{%
3114 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3115 \toks@={#1}%
3116 \ifx\null#2\null
3117 \ifx\null#3\null
3118 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3119 \def\@gls@checkescactual{\relax}%
3120 \else
3121 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3122 \@gls@quotechar\string"\@gls@actualchar
3123 \@gls@quotechar\string"\@gls@actualchar}%
3124 \def\@gls@checkescactual{\@gls@checkescactual#3\null}%
3125 \fi
3126 \else
3127 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3128 \@gls@quotechar\string"\@gls@actualchar}%
3129 \ifx\null#3\null
3130 \def\@gls@checkescactual{\@gls@checkescactual#2\?\?\null}%
3131 \else
3132 \def\@gls@checkescactual{\@gls@checkescactual#2\?#3\null}%
3133 \fi
3134 \fi
3135 \@gls@checkescactual
3136 }

```

`\@gls@checkescbar` Similarly for `\|`:

```

3137 \def\@gls@checkescbar#1\|#2\|#3\null{%
3138 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3139 \toks@={#1}%
3140 \ifx\null#2\null
3141 \ifx\null#3\null
3142 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3143 \def\@gls@checkescbar{\relax}%
3144 \else
3145 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3146 \@gls@quotechar\string"\@gls@encapchar
3147 \@gls@quotechar\string"\@gls@encapchar}%
3148 \def\@gls@checkescbar{\@gls@checkescbar#3\null}%
3149 \fi
3150 \else
3151 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3152 \@gls@quotechar\string"\@gls@encapchar}%
3153 \ifx\null#3\null

```

```

3154 \def\@gls@checkesbar{\@gls@checkesbar#2\|\|\null}%
3155 \else
3156 \def\@gls@checkesbar{\@gls@checkesbar#2\|#3\null}%
3157 \fi
3158 \fi
3159 \@gls@checkesbar
3160 }

```

\@gls@checkesclevel Similarly for \!:

```

3161 \def\@gls@checkesclevel#1\!#2\!#3\null{%
3162 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3163 \toks@={#1}%
3164 \ifx\null#2\null
3165 \ifx\null#3\null
3166 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3167 \def\@gls@checkesclevel{\relax}%
3168 \else
3169 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3170 \@gls@quotechar\string\@\@gls@levelchar
3171 \@gls@quotechar\string\@\@gls@levelchar}%
3172 \def\@gls@checkesclevel{\@gls@checkesclevel#3\null}%
3173 \fi
3174 \else
3175 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3176 \@gls@quotechar\string\@\@gls@levelchar}%
3177 \ifx\null#3\null
3178 \def\@gls@checkesclevel{\@gls@checkesclevel#2\!\!\null}%
3179 \else
3180 \def\@gls@checkesclevel{\@gls@checkesclevel#2\!#3\null}%
3181 \fi
3182 \fi
3183 \@gls@checkesclevel
3184 }

```

\@gls@checkbar and for |:

```

3185 \def\@gls@checkbar#1|#2|#3\null{%
3186 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3187 \toks@={#1}%
3188 \ifx\null#2\null
3189 \ifx\null#3\null
3190 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3191 \def\@gls@checkbar{\relax}%
3192 \else
3193 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3194 \@gls@quotechar\@gls@encapchar\@gls@quotechar\@gls@encapchar}%
3195 \def\@gls@checkbar{\@gls@checkbar#3\null}%
3196 \fi
3197 \else
3198 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@

```

```

3199 \@gls@quotecar\@gls@encapchar}%
3200 \ifx\null#3\null
3201 \def\@gls@checkbar{\@gls@checkbar#2|\null}%
3202 \else
3203 \def\@gls@checkbar{\@gls@checkbar#2|#3\null}%
3204 \fi
3205 \fi
3206 \@gls@checkbar
3207 }

```

\@gls@checklevel and for !:

```

3208 \def\@gls@checklevel#1!#2!#3\null{%
3209 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3210 \toks@={#1}%
3211 \ifx\null#2\null
3212 \ifx\null#3\null
3213 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3214 \def\@gls@checklevel{\relax}%
3215 \else
3216 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3217 \@gls@quotecar\@gls@levelchar\@gls@quotecar\@gls@levelchar}%
3218 \def\@gls@checklevel{\@gls@checklevel#3\null}%
3219 \fi
3220 \else
3221 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3222 \@gls@quotecar\@gls@levelchar}%
3223 \ifx\null#3\null
3224 \def\@gls@checklevel{\@gls@checklevel#2!\null}%
3225 \else
3226 \def\@gls@checklevel{\@gls@checklevel#2!#3\null}%
3227 \fi
3228 \fi
3229 \@gls@checklevel
3230 }

```

\@gls@checkactual and for ?:

```

3231 \def\@gls@checkactual#1?#2?#3\null{%
3232 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3233 \toks@={#1}%
3234 \ifx\null#2\null
3235 \ifx\null#3\null
3236 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3237 \def\@gls@checkactual{\relax}%
3238 \else
3239 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3240 \@gls@quotecar\@gls@actualchar\@gls@quotecar\@gls@actualchar}%
3241 \def\@gls@checkactual{\@gls@checkactual#3\null}%
3242 \fi
3243 \else

```

```

3244 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3245 \@gls@quotechar\@gls@actualchar}%
3246 \ifx\null#3\null
3247 \def\@gls@checkactual{\@gls@checkactual#2??\null}%
3248 \else
3249 \def\@gls@checkactual{\@gls@checkactual#2?#3\null}%
3250 \fi
3251 \fi
3252 \@gls@checkactual
3253 }

```

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

```

3254 \def\@gls@xdycheckquote#1"#2"#3\null{%
3255 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3256 \toks@={#1}%
3257 \ifx\null#2\null
3258 \ifx\null#3\null
3259 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3260 \def\@gls@xdycheckquote{\relax}%
3261 \else
3262 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3263 \string\}\string\}%
3264 \def\@gls@xdycheckquote{\@gls@xdycheckquote#3\null}%
3265 \fi
3266 \else
3267 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3268 \string\}%
3269 \ifx\null#3\null
3270 \def\@gls@xdycheckquote{\@gls@xdycheckquote#2""\null}%
3271 \else
3272 \def\@gls@xdycheckquote{\@gls@xdycheckquote#2"#3\null}%
3273 \fi
3274 \fi
3275 \@gls@xdycheckquote
3276 }

```

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

```

\@gls@xdycheckbackslash
3277 \edef\def\@gls@xdycheckbackslash{%
3278 \noexpand\def\noexpand\@gls@xdycheckbackslash##1\@backslashchar
3279 ##2\@backslashchar##3\noexpand\null{%
3280 \noexpand\@gls@tmpb=\noexpand\expandafter
3281 {\noexpand\@gls@checkedmkidx}%
3282 \noexpand\toks@={##1}%
3283 \noexpand\ifx\noexpand\null##2\noexpand\null
3284 \noexpand\ifx\noexpand\null##3\noexpand\null
3285 \noexpand\edef\noexpand\@gls@checkedmkidx{%
3286 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
3287 \noexpand\def\noexpand\@gls@xdycheckbackslash{\relax}%

```

```

3288 \noexpand\else
3289 \noexpand\edef\noexpand\@gls@checkedmkidx{%
3290 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
3291 \@backslashchar\@backslashchar\@backslashchar\@backslashchar}%
3292 \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3293 \noexpand\@gls@xdycheckbackslash##3\noexpand\null}%
3294 \noexpand\fi
3295 \noexpand\else
3296 \noexpand\edef\noexpand\@gls@checkedmkidx{%
3297 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
3298 \@backslashchar\@backslashchar}%
3299 \noexpand\ifx\noexpand\null##3\noexpand\null
3300 \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3301 \noexpand\@gls@xdycheckbackslash##2\@backslashchar
3302 \@backslashchar\noexpand\null}%
3303 \noexpand\else
3304 \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3305 \noexpand\@gls@xdycheckbackslash##2\@backslashchar
3306 ##3\noexpand\null}%
3307 \noexpand\fi
3308 \noexpand\fi
3309 \noexpand\@gls@xdycheckbackslash
3310 }%
3311 }

```

Now go ahead and define \gls@xdycheckbackslash

```
3312 \def@gls@xdycheckbackslash
```

\glsdohypertarget

```

3313 \newlength@gls@tmplen
3314 \newcommand*\glsdohypertarget}[2]{%
3315 \settoheight@gls@tmplen}{#2}%
3316 \raisebox@gls@tmplen}{\hypertarget{#1}{}}#2%
3317 }

```

\glsdohyperlink

```
3318 \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.

```

3319 \ifcsundef{hyperlink}%
3320 {%
3321 \let@glslink\@secondoftwo
3322 }%
3323 {%
3324 \let@glslink@glsdohyperlink
3325 }

```


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

```
3326 \ifcsundef{hypertarget}%
3327 {%
3328   \let\@glstarget\@secondoftwo
3329 }%
3330 {%
3331   \let\@glstarget\glsdohypertarget
3332 }
```

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

`\glsdisablehyper`

```
3333 \newcommand{\glsdisablehyper}{%
3334   \KV@glslink@hyperfalse
3335   \let\@glslink\@secondoftwo
3336   \let\@glstarget\@secondoftwo
3337 }
```

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

`\glsenablehyper`

```
3338 \newcommand{\glsenablehyper}{%
3339   \KV@glslink@hypertrue
3340   \let\@glslink\glsdohyperlink
3341   \let\@glstarget\glsdohypertarget
3342 }
```

Provide some convenience commands if not already defined:

```
3343 \providecommand{\@firstofthree}[3]{#1}
3344 \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`

```
3345 \newrobustcmd*{\gls}{\@gls@hyp@opt\@gls}
```

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

`\@gls`

```
3346 \newcommand*{\@gls}[2] [] {%
3347   \new@ifnextchar[{\@gls@{#1}{#2}}{\@gls@{#1}{#2} []}]%
3348 }
```

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

```
3349 \def\@gls@#1#2[#3]{%
3350   \glsdoifexists{#2}%
3351   {%
3352     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3353     \let\glsifplural\@secondoftwo
3354     \let\gls caps case\@firstofthree
3355     \let\gls custom text\@empty
3356     \def\glsinsert{#3}%
```

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

```
3357   \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.

```
3358   \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3359   \ifKV@gls@link@local
3360     \glslocalunset{#2}%
3361   \else
3362     \glsunset{#2}%
3363   \fi
3364 }%
3365 }
```

`\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`

```
3366 \newrobustcmd*{\Gls}{\@gls@hyp@opt\@Gls}
```

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

```
3367 \newcommand*{\@Gls}[2] [] {%
3368   \new@ifnextchar[{\@Gls@{#1}{#2}}{\@Gls@{#1}{#2} []}%
3369 }
```

\@Gls@ Read in the final optional argument:

```
3370 \def\@Gls@#1#2[#3]{%
3371   \glsdoifexists{#2}%
3372   {%
3373     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
3374     \let\glsifplural\@secondoftwo
3375     \let\gls caps case\@secondofthree
3376     \let\gls custom text\@empty
3377     \def\glsinsert{#3}%
```

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

```
3378   \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.

```
3379   \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3380   \ifKV@gls@link@local
3381     \glslocalunset{#2}%
3382   \else
3383     \glsunset{#2}%
3384   \fi
3385 }%
3386 }
```

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

\GLS

```
3387 \newrobustcmd*{\GLS}{\@gls@hyp@opt\@GLS}
```

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

```
3388 \newcommand*{\@GLS}[2] [] {%
3389   \new@ifnextchar[{\@GLS@{#1}{#2}}{\@GLS@{#1}{#2} []}%
3390 }
```

\@GLS@ Read in the final optional argument:

```
3391 \def\@GLS@#1#2[#3]{%
3392   \glsdoifexists{#2}%
3393   {%
3394     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
```

```

3395 \let\glsifplural\@secondoftwo
3396 \let\glscapscase\@thirdofthree
3397 \let\glscustomtext\@empty
3398 \def\glsinsert{#3}%

```

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

```

3399 \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.

```

3400 \@gls@link[#1]{#2}{\@glo@text}%

```

Indicate that this entry has now been used

```

3401 \ifKV@glslink@local
3402 \glslocalunset{#2}%
3403 \else
3404 \glsunset{#2}%
3405 \fi
3406 }%
3407 }

```

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

\glspl

```

3408 \newrobustcmd*{\glspl}{\@gls@hyp@opt\@glspl}

```

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

```

3409 \newcommand*{\@glspl}[2][ ]{%
3410 \new@ifnextchar[{\@glspl@{#1}{#2}}{\@glspl@{#1}{#2}[ ]}%
3411 }

```

\@glspl@ Read in the final optional argument:

```

3412 \def\@glspl@#1#2[#3]{%
3413 \glsdoifexists{#2}%
3414 {%
3415 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3416 \let\glsifplural\@firstoftwo
3417 \let\glscapscase\@firstofthree
3418 \let\glscustomtext\@empty
3419 \def\glsinsert{#3}%

```

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

```

3420 \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.

```
3421 \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3422 \ifKV@gls@link@local
3423 \glslocalunset{#2}%
3424 \else
3425 \glsunset{#2}%
3426 \fi
3427 }%
3428 }
```

`\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`

```
3429 \newrobustcmd*{\Glspl}{\@gls@hyp@opt\@Glspl}
```

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

```
3430 \newcommand*{\@Glspl}[2][ ]{%
3431 \new@ifnextchar[{\@Glspl@{#1}{#2}}{\@Glspl@{#1}{#2}[ ]}%
3432 }
```

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

```
3433 \def\@Glspl@#1#2[#3]{%
3434 \glsdoifexists{#2}%
3435 {%
3436 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3437 \let\gls@ifplural\@firstoftwo
3438 \let\gls@capscase\@secondofthree
3439 \let\gls@customtext\@empty
3440 \def\gls@insert{#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`.

```
3441 \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.

```
3442 \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3443 \ifKV@glslink@local
3444 \glslocalunset{#2}%
3445 \else
3446 \glsunset{#2}%
3447 \fi
3448 }%
3449 }
```

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

`\GLSp1`

```
3450 \newrobustcmd*{\GLSp1}{\@gls@hyp@opt\@GLSp1}
```

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

```
3451 \newcommand*{\@GLSp1}[2][ ]{%
3452 \new@ifnextchar[{\@GLSp1@{#1}{#2}}{\@GLSp1@{#1}{#2}[ ]}%
3453 }
```

`\@GLSp1` Read in the final optional argument:

```
3454 \def\@GLSp1@#1#2[#3]{%
3455 \glsdoifexists{#2}%
3456 {%
3457 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3458 \let\glsifplural\@firstoftwo
3459 \let\gls@caps\@thirdofthree
3460 \let\gls@customtext\@empty
3461 \def\glsinsert{#3}%
```

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

```
3462 \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.

```
3463 \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3464 \ifKV@glslink@local
3465 \glslocalunset{#2}%
3466 \else
3467 \glsunset{#2}%
3468 \fi
3469 }%
3470 }
```

`\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:

```
3471 \newrobustcmd*{\glsdisp}{\@gls@hyp@opt\@glsdisp}
```

Defined the un-starred form.

`\@glsdisp`

```
3472 \newcommand*{\@glsdisp}[3][ ]{%
```

```
3473   \glsdoifexists{#2}{%
```

```
3474     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
```

```
3475     \let\glsifplural\@secondoftwo
```

```
3476     \let\gls caps case\@firstofthree
```

```
3477     \def\gls custom text{#3}%
```

```
3478     \def\gls insert{ }%
```

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

```
3479     \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.

```
3480     \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3481     \ifKV@gls@link@local
```

```
3482       \glslocalunset{#2}%
```

```
3483     \else
```

```
3484       \glsunset{#2}%
```

```
3485     \fi
```

```
3486   }%
```

```
3487 }
```

`\@gls@field@link`

```
3488 \newcommand{\@gls@field@link}[3]{%
```

```
3489   \glsdoifexists{#2}%
```

```
3490   {%
```

```
3491     \let\do@gls@link@checkfirsthyper\relax
```

```
3492     \@gls@link[#1]{#2}{#3}%
```

```
3493   }%
```

```
3494 }
```

`\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.

`\glstext`

```
3495 \newrobustcmd*{\glstext}{\@gls@hyp@opt\@glstext}
```

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

```
3496 \newcommand*{\@glstext}[2][\@glstext]
```

```
3497 \new@ifnextchar[{\@glstext@{#1}{#2}}{\@glstext@{#1}{#2}[]}]
```

Read in the final optional argument:

```
3498 \def\@glstext@#1#2[#3]{%
```

```
3499 \@gls@field@link{#1}{#2}{\glsentrytext{#2}#3}%
```

```
3500 }
```

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

`\GLStext`

```
3501 \newrobustcmd*{\GLStext}{\@gls@hyp@opt\@GLStext}
```

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

```
3502 \newcommand*{\@GLStext}[2][\@GLStext]
```

```
3503 \new@ifnextchar[{\@GLStext@{#1}{#2}}{\@GLStext@{#1}{#2}[]}]
```

Read in the final optional argument:

```
3504 \def\@GLStext@#1#2[#3]{%
```

```
3505 \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrytext{#2}#3}}%
```

```
3506 }
```

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

`\Glstext`

```
3507 \newrobustcmd*{\Glstext}{\@gls@hyp@opt\@Glstext}
```

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

```
3508 \newcommand*{\@Glstext}[2][\@Glstext]
```

```
3509 \new@ifnextchar[{\@Glstext@{#1}{#2}}{\@Glstext@{#1}{#2}[]}]
```

Read in the final optional argument:

```
3510 \def\@Glstext@#1#2[#3]{%
```

```
3511 \@gls@field@link{#1}{#2}{\Glsentrytext{#2}#3}%
```

```
3512 }
```

`\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`

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


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

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

Read in the final optional argument:

```
3516 \def\@glsfirst@#1#2[#3] {%
3517   \@gls@field@link{#1}{#2}{\glsentryfirst{#2}#3}%
3518 }
```

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

`\Glsfirst`

```
3519 \newrobustcmd*{\Glsfirst}{\@gls@hyp@opt\@Glsfirst}
```

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

```
3520 \newcommand*{\@Glsfirst}[2] [] {%
3521   \new@ifnextchar[{\@Glsfirst@{#1}{#2}}{\@Glsfirst@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3522 \def\@Glsfirst@#1#2[#3] {%
3523   \@gls@field@link{#1}{#2}{\Glsentryfirst{#2}#3}%
3524 }
```

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

`\GLSfirst`

```
3525 \newrobustcmd*{\GLSfirst}{\@gls@hyp@opt\@GLSfirst}
```

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

```
3526 \newcommand*{\@GLSfirst}[2] [] {%
3527   \new@ifnextchar[{\@GLSfirst@{#1}{#2}}{\@GLSfirst@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3528 \def\@GLSfirst@#1#2[#3] {%
3529   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryfirst{#2}#3}}%
3530 }
```

`\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`

```
3531 \newrobustcmd*{\glsplural}{\@gls@hyp@opt\@glsplural}
```

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

```
3532 \newcommand*{\@glsplural}[2] [] {%
3533   \new@ifnextchar[{\@glsplural@{#1}{#2}}{\@glsplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3534 \def\@glsplural@#1#2[#3]{%
3535   \@gls@field@link{#1}{#2}{\glsentryplural{#2}#3}%
3536 }
```

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

`\Glsplural`

```
3537 \newrobustcmd*{\Glsplural}{\@gls@hyp@opt\@Glsplural}
```

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

```
3538 \newcommand*{\@Glsplural}[2] []{%
3539   \new@ifnextchar[{\@Glsplural@{#1}{#2}}{\@Glsplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3540 \def\@Glsplural@#1#2[#3]{%
3541   \@gls@field@link{#1}{#2}{\Glsentryplural{#2}#3}%
3542 }
```

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

`\GLSplural`

```
3543 \newrobustcmd*{\GLSplural}{\@gls@hyp@opt\@GLSplural}
```

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

```
3544 \newcommand*{\@GLSplural}[2] []{%
3545   \new@ifnextchar[{\@GLSplural@{#1}{#2}}{\@GLSplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3546 \def\@GLSplural@#1#2[#3]{%
3547   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryplural{#2}#3}}%
3548 }
```

`\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`

```
3549 \newrobustcmd*{\glsfirstplural}{\@gls@hyp@opt\@glsfirstplural}
```

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

```
3550 \newcommand*{\@glsfirstplural}[2] []{%
3551   \new@ifnextchar[{\@glsfirstplural@{#1}{#2}}{\@glsfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3552 \def\@glsfirstplural@#1#2[#3]{%
3553   \@gls@field@link{#1}{#2}{\glsentryfirstplural{#2}#3}%
3554 }
```

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

`\Glsfirstplural`

```
3555 \newrobustcmd*{\Glsfirstplural}{\@gls@hyp@opt\@Glsfirstplural}
```

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

```
3556 \newcommand*{\@Glsfirstplural}[2] [] {%
```

```
3557   \new@ifnextchar[{\@Glsfirstplural@{#1}{#2}}{\@Glsfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3558 \def\@Glsfirstplural@#1#2[#3] {%
```

```
3559   \@gls@field@link{#1}{#2}{\Glsentryfirstplural{#2}#3}%
```

```
3560 }
```

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

`\GLSfirstplural`

```
3561 \newrobustcmd*{\GLSfirstplural}{\@gls@hyp@opt\@GLSfirstplural}
```

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

```
3562 \newcommand*{\@GLSfirstplural}[2] [] {%
```

```
3563   \new@ifnextchar[{\@GLSfirstplural@{#1}{#2}}{\@GLSfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3564 \def\@GLSfirstplural@#1#2[#3] {%
```

```
3565   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryfirstplural{#2}#3}}%
```

```
3566 }
```

`\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`

```
3567 \newrobustcmd*{\glsname}{\@gls@hyp@opt\@glsname}
```

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

```
3568 \newcommand*{\@glsname}[2] [] {%
```

```
3569   \new@ifnextchar[{\@glsname@{#1}{#2}}{\@glsname@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3570 \def\@glsname@#1#2[#3] {%
```

```
3571   \@gls@field@link{#1}{#2}{\glsentryname{#2}#3}%
```

```
3572 }
```

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

`\Glsname`

```
3573 \newrobustcmd*{\Glsname}{\@gls@hyp@opt\@Glsname}
```

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

```
3574 \newcommand*{\@Glsname}[2][\%  
3575 \new@ifnextchar[{\@Glsname@{#1}{#2}}{\@Glsname@{#1}{#2}}{[]}]}
```

Read in the final optional argument:

```
3576 \def\@Glsname@#1#2[#3]{%  
3577 \@gls@field@link{#1}{#2}{\Glsentryname{#2}#3}%  
3578 }
```

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

\GLSname

```
3579 \newrobustcmd*{\GLSname}{\@gls@hyp@opt\@GLSname}
```

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

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

Read in the final optional argument:

```
3582 \def\@GLSname@#1#2[#3]{%  
3583 \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryname{#2}#3}}%  
3584 }
```

\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

```
3585 \newrobustcmd*{\glsdesc}{\@gls@hyp@opt\@glsdesc}
```

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

```
3586 \newcommand*{\@glsdesc}[2][\%  
3587 \new@ifnextchar[{\@glsdesc@{#1}{#2}}{\@glsdesc@{#1}{#2}}{[]}]}
```

Read in the final optional argument:

```
3588 \def\@glsdesc@#1#2[#3]{%  
3589 \@gls@field@link{#1}{#2}{\glsentrydesc{#2}#3}%  
3590 }
```

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

\Glsdesc

```
3591 \newrobustcmd*{\Glsdesc}{\@gls@hyp@opt\@Glsdesc}
```

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

```
3592 \newcommand*{\@Glsdesc}[2][\%  
3593 \new@ifnextchar[{\@Glsdesc@{#1}{#2}}{\@Glsdesc@{#1}{#2}}{[]}]}
```

Read in the final optional argument:

```
3594 \def\@GLSdesc@#1#2[#3]{%
3595   \@gls@field@link{#1}{#2}{\Glsentrydesc{#2}#3}%
3596 }
```

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

`\GLSdesc`

```
3597 \newrobustcmd*{\GLSdesc}{\@gls@hyp@opt\@GLSdesc}
```

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

```
3598 \newcommand*{\@GLSdesc}[2][ ]{%
3599   \new@ifnextchar[{\@GLSdesc@{#1}{#2}}{\@GLSdesc@{#1}{#2}[ ]}]
```

Read in the final optional argument:

```
3600 \def\@GLSdesc@#1#2[#3]{%
3601   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrydesc{#2}#3}}%
3602 }
```

`\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`

```
3603 \newrobustcmd*{\glsdescplural}{\@gls@hyp@opt\@glsdescplural}
```

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

```
3604 \newcommand*{\@glsdescplural}[2][ ]{%
3605   \new@ifnextchar[{\@glsdescplural@{#1}{#2}}{\@glsdescplural@{#1}{#2}[ ]}]
```

Read in the final optional argument:

```
3606 \def\@glsdescplural@#1#2[#3]{%
3607   \@gls@field@link{#1}{#2}{\glsentrydescplural{#2}#3}%
3608 }
```

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

`\Glsdescplural`

```
3609 \newrobustcmd*{\Glsdescplural}{\@gls@hyp@opt\@Glsdescplural}
```

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

```
3610 \newcommand*{\@Glsdescplural}[2][ ]{%
3611   \new@ifnextchar[{\@Glsdescplural@{#1}{#2}}{\@Glsdescplural@{#1}{#2}[ ]}]
```

Read in the final optional argument:

```
3612 \def\@Glsdescplural@#1#2[#3]{%
3613   \@gls@field@link{#1}{#2}{\Glsentrydescplural{#2}#3}%
3614 }
```

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

`\GLSdescplural`

```
3615 \newrobustcmd*{\GLSdescplural}{\@gls@hyp@opt\@GLSdescplural}
```

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

```
3616 \newcommand*{\@GLSdescplural}[2] [] {%
```

```
3617   \new@ifnextchar[{\@GLSdescplural@{#1}{#2}}{\@GLSdescplural@{#1}{#2} []}]
```

Read in the final optional argument:

```
3618 \def\@GLSdescplural@#1#2[#3] {%
```

```
3619   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrydescplural{#2}#3}}%
```

```
3620 }
```

`\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`

```
3621 \newrobustcmd*{\glsymbol}{\@gls@hyp@opt\@glsymbol}
```

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

```
3622 \newcommand*{\@glsymbol}[2] [] {%
```

```
3623   \new@ifnextchar[{\@glsymbol@{#1}{#2}}{\@glsymbol@{#1}{#2} []}]
```

Read in the final optional argument:

```
3624 \def\@glsymbol@#1#2[#3] {%
```

```
3625   \@gls@field@link{#1}{#2}{\glsentrysymbol{#2}#3}}%
```

```
3626 }
```

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

`\Glsymbol`

```
3627 \newrobustcmd*{\Glsymbol}{\@gls@hyp@opt\@Glsymbol}
```

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

```
3628 \newcommand*{\@Glsymbol}[2] [] {%
```

```
3629   \new@ifnextchar[{\@Glsymbol@{#1}{#2}}{\@Glsymbol@{#1}{#2} []}]
```

Read in the final optional argument:

```
3630 \def\@Glsymbol@#1#2[#3] {%
```

```
3631   \@gls@field@link{#1}{#2}{\Glsentrysymbol{#2}#3}}%
```

```
3632 }
```

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

`\GLSsymbol`

```
3633 \newrobustcmd*{\GLSsymbol}{\@gls@hyp@opt\@GLSsymbol}
```

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

```
3634 \newcommand*{\@GLSsymbol}[2] [] {%
3635   \new@ifnextchar [{\@GLSsymbol@{#1}{#2}}{\@GLSsymbol@{#1}{#2} []}}
```

Read in the final optional argument:

```
3636 \def\@GLSsymbol@#1#2[#3] {%
3637   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrysymbol{#2}#3}}%
3638 }
```

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

`\glsymbolplural`

```
3639 \newrobustcmd*{\glsymbolplural}{\@gls@hyp@opt\@glsymbolplural}
```

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

```
3640 \newcommand*{\@glssymbolplural}[2] [] {%
3641   \new@ifnextchar [{\@glssymbolplural@{#1}{#2}}{\@glssymbolplural@{#1}{#2} []}}
```

Read in the final optional argument:

```
3642 \def\@glssymbolplural@#1#2[#3] {%
3643   \@gls@field@link{#1}{#2}{\glsentrysymbolplural{#2}#3}%
3644 }
```

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

`\Glsymbolplural`

```
3645 \newrobustcmd*{\Glsymbolplural}{\@gls@hyp@opt\@Glsymbolplural}
```

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

```
3646 \newcommand*{\@GLSsymbolplural}[2] [] {%
3647   \new@ifnextchar [{\@GLSsymbolplural@{#1}{#2}}{\@GLSsymbolplural@{#1}{#2} []}}
```

Read in the final optional argument:

```
3648 \def\@GLSsymbolplural@#1#2[#3] {%
3649   \@gls@field@link{#1}{#2}{\GLSentrysymbolplural{#2}#3}%
3650 }
```

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

`\GLSsymbolplural`

```
3651 \newrobustcmd*{\GLSsymbolplural}{\@gls@hyp@opt\@GLSsymbolplural}
```

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

```
3652 \newcommand*{\@GLSsymbolplural}[2] [] {%
3653   \new@ifnextchar [{\@GLSsymbolplural@{#1}{#2}}{\@GLSsymbolplural@{#1}{#2} []}}
```

Read in the final optional argument:

```
3654 \def\@GLSsymbolplural@#1#2[#3]{%
3655   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrysymbolplural{#2}#3}}%
3656 }
```

`\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`

```
3657 \newrobustcmd*{\glsuseri}{\@gls@hyp@opt\@glsuseri}
```

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

```
3658 \newcommand*{\@glsuseri}[2] [] {%
3659   \new@ifnextchar[{\@glsuseri@{#1}{#2}}{\@glsuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3660 \def\@glsuseri@#1#2[#3]{%
3661   \@gls@field@link{#1}{#2}{\glsentryuseri{#2}#3}%
3662 }
```

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

`\Glsuseri`

```
3663 \newrobustcmd*{\Glsuseri}{\@gls@hyp@opt\@Glsuseri}
```

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

```
3664 \newcommand*{\@Glsuseri}[2] [] {%
3665   \new@ifnextchar[{\@Glsuseri@{#1}{#2}}{\@Glsuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3666 \def\@Glsuseri@#1#2[#3]{%
3667   \@gls@field@link{#1}{#2}{\Glsentryuseri{#2}#3}%
3668 }
```

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

`\GLSuseri`

```
3669 \newrobustcmd*{\GLSuseri}{\@gls@hyp@opt\@GLSuseri}
```

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

```
3670 \newcommand*{\@GLSuseri}[2] [] {%
3671   \new@ifnextchar[{\@GLSuseri@{#1}{#2}}{\@GLSuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3672 \def\@GLSuseri@#1#2[#3]{%
3673   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseri{#2}#3}}%
3674 }
```


`\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`

```
3675 \newrobustcmd*{\glsuserii}{\@gls@hyp@opt\@glsuserii}
```

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

```
3676 \newcommand*{\@glsuserii}[2] [] {%
```

```
3677   \new@ifnextchar[{\@glsuserii@{#1}{#2}}{\@glsuserii@{#1}{#2} []}]
```

Read in the final optional argument:

```
3678 \def\@glsuserii@#1#2[#3] {%
```

```
3679   \@gls@field@link{#1}{#2}{\glsentryuserii{#2}#3}%
```

```
3680 }
```

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

`\Glsuserii`

```
3681 \newrobustcmd*{\Glsuserii}{\@gls@hyp@opt\@Glsuserii}
```

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

```
3682 \newcommand*{\@Glsuserii}[2] [] {%
```

```
3683   \new@ifnextchar[{\@Glsuserii@{#1}{#2}}{\@Glsuserii@{#1}{#2} []}]
```

Read in the final optional argument:

```
3684 \def\@Glsuserii@#1#2[#3] {%
```

```
3685   \@gls@field@link{#1}{#2}{\Glsentryuserii{#2}#3}%
```

```
3686 }
```

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

`\GLSuserii`

```
3687 \newrobustcmd*{\GLSuserii}{\@gls@hyp@opt\@GLSuserii}
```

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

```
3688 \newcommand*{\@GLSuserii}[2] [] {%
```

```
3689   \new@ifnextchar[{\@GLSuserii@{#1}{#2}}{\@GLSuserii@{#1}{#2} []}]
```

Read in the final optional argument:

```
3690 \def\@GLSuserii@#1#2[#3] {%
```

```
3691   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuserii{#2}#3}}%
```

```
3692 }
```

`\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`

```
3693 \newrobustcmd*{\glsuseriii}{\@gls@hyp@opt\@glsuseriii}
```

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

```
3694 \newcommand*{\@glsuseriii}[2] [] {%
3695   \new@ifnextchar[{\@glsuseriii@{#1}{#2}}{\@glsuseriii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3696 \def\@glsuseriii@#1#2[#3]{%
3697   \@gls@field@link{#1}{#2}{\glsentryuseriii{#2}#3}%
3698 }
```

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

\Glsuseriii

```
3699 \newrobustcmd*{\Glsuseriii}{\@gls@hyp@opt\@Glsuseriii}
```

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

```
3700 \newcommand*{\@Glsuseriii}[2] [] {%
3701   \new@ifnextchar[{\@Glsuseriii@{#1}{#2}}{\@Glsuseriii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3702 \def\@Glsuseriii@#1#2[#3]{%
3703   \@gls@field@link{#1}{#2}{\Glsentryuseriii{#2}#3}%
3704 }
```

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

\GLSuseriii

```
3705 \newrobustcmd*{\GLSuseriii}{\@gls@hyp@opt\@GLSuseriii}
```

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

```
3706 \newcommand*{\@GLSuseriii}[2] [] {%
3707   \new@ifnextchar[{\@GLSuseriii@{#1}{#2}}{\@GLSuseriii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3708 \def\@GLSuseriii@#1#2[#3]{%
3709   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseriii{#2}#3}}%
3710 }
```

\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

```
3711 \newrobustcmd*{\glsuseriv}{\@gls@hyp@opt\@glsuseriv}
```

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

```
3712 \newcommand*{\@glsuseriv}[2] [] {%
3713   \new@ifnextchar[{\@glsuseriv@{#1}{#2}}{\@glsuseriv@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3714 \def\@glsuseriv@#1#2[#3]{%
3715   \@gls@field@link{#1}{#2}{\glsentryuseriv{#2}#3}%
3716 }
```

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

`\Glsuseriv`

```
3717 \newrobustcmd*{\Glsuseriv}{\@gls@hyp@opt\@Glsuseriv}
```

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

```
3718 \newcommand*{\@Glsuseriv}[2] []{%
3719   \new@ifnextchar[{\@Glsuseriv@{#1}{#2}}{\@Glsuseriv@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3720 \def\@Glsuseriv@#1#2[#3]{%
3721   \@gls@field@link{#1}{#2}{\glsentryuseriv{#2}#3}%
3722 }
```

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

`\GLSuseriv`

```
3723 \newrobustcmd*{\GLSuseriv}{\@gls@hyp@opt\@GLSuseriv}
```

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

```
3724 \newcommand*{\@GLSuseriv}[2] []{%
3725   \new@ifnextchar[{\@GLSuseriv@{#1}{#2}}{\@GLSuseriv@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3726 \def\@GLSuseriv@#1#2[#3]{%
3727   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseriv{#2}#3}}%
3728 }
```

`\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`

```
3729 \newrobustcmd*{\glsuserv}{\@gls@hyp@opt\@glsuserv}
```

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

```
3730 \newcommand*{\@glsuserv}[2] []{%
3731   \new@ifnextchar[{\@glsuserv@{#1}{#2}}{\@glsuserv@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3732 \def\@glsuserv@#1#2[#3]{%
3733   \@gls@field@link{#1}{#2}{\glsentryuserv{#2}#3}%
3734 }
```

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

`\Glsuserv`

```
3735 \newrobustcmd*{\Glsuserv}{\@gls@hyp@opt\@Glsuserv}
```

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

```
3736 \newcommand*{\@Glsuserv}[2] [] {%
```

```
3737 \new@ifnextchar [ {\@Glsuserv@{#1}{#2}} {\@Glsuserv@{#1}{#2} [] } }
```

Read in the final optional argument:

```
3738 \def\@Glsuserv@#1#2[#3] {%
```

```
3739 \@gls@field@link{#1}{#2}{\Glsentryuserv{#2}#3}%
```

```
3740 }
```

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

`\GLSuserv`

```
3741 \newrobustcmd*{\GLSuserv}{\@gls@hyp@opt\@GLSuserv}
```

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

```
3742 \newcommand*{\@GLSuserv}[2] [] {%
```

```
3743 \new@ifnextchar [ {\@GLSuserv@{#1}{#2}} {\@GLSuserv@{#1}{#2} [] } }
```

Read in the final optional argument:

```
3744 \def\@GLSuserv@#1#2[#3] {%
```

```
3745 \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuserv{#2}#3}}%
```

```
3746 }
```

`\glsuservi` behaves like `\gls` except it always uses the value given by the `user6` key and it doesn't mark the entry as used.

`\glsuservi`

```
3747 \newrobustcmd*{\glsuservi}{\@gls@hyp@opt\@glsuservi}
```

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

```
3748 \newcommand*{\@glsuservi}[2] [] {%
```

```
3749 \new@ifnextchar [ {\@glsuservi@{#1}{#2}} {\@glsuservi@{#1}{#2} [] } }
```

Read in the final optional argument:

```
3750 \def\@glsuservi@#1#2[#3] {%
```

```
3751 \@gls@field@link{#1}{#2}{\glsentryuservi{#2}#3}%
```

```
3752 }
```

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

`\Glsuservi`

```
3753 \newrobustcmd*{\Glsuservi}{\@gls@hyp@opt\@Glsuservi}
```

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

```
3754 \newcommand*{\@Glsuservi}[2] [] {%
3755   \new@ifnextchar[{\@Glsuservi@{#1}{#2}}{\@Glsuservi@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3756 \def\@Glsuservi@#1#2[#3] {%
3757   \@gls@field@link{#1}{#2}{\Glsentryuservi{#2}#3}%
3758 }
```

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

\GLSuservi

```
3759 \newrobustcmd*{\GLSuservi}{\@gls@hyp@opt\@GLSuservi}
```

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

```
3760 \newcommand*{\@GLSuservi}[2] [] {%
3761   \new@ifnextchar[{\@GLSuservi@{#1}{#2}}{\@GLSuservi@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3762 \def\@GLSuservi@#1#2[#3] {%
3763   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuservi{#2}#3}}%
3764 }
```

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

\acrshort

```
3765 \newrobustcmd*{\acrshort}{\@gls@hyp@opt\@ns@acrshort}
```

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

```
3766 \newcommand*{\@ns@acrshort}[2] [] {%
3767   \new@ifnextchar[{\@acrshort{#1}{#2}}{\@acrshort{#1}{#2} []}]%
3768 }
```

Read in the final optional argument:

```
3769 \def\@acrshort#1#2[#3] {%
3770   \glsdoifexists{#2}%
3771   {%
3772     \let\do@gls@link@checkfirsthyper\relax
3773     \let\glsifplural\@secondoftwo
3774     \let\glscapscase\@firstofthree
3775     \let\glsinsert\@empty
3776     \def\glscustomtext{%
3777       \acronymfont{\glsentryshort{#2}#3}%
3778     }%
```

Call `\@gls@link` Note that `\@gls@link` sets `\glstype`.

```
3779 \@gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%  
3780 }%  
3781 }
```

`\Acrshort`

```
3782 \newrobustcmd*{\Acrshort}{\@gls@hyp@opt\ns@Acrshort}
```

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

```
3783 \newcommand*{\ns@Acrshort}[2] [] {%  
3784 \new@ifnextchar[{\@Acrshort{#1}{#2}}{\@Acrshort{#1}{#2} []}]%  
3785 }
```

Read in the final optional argument:

```
3786 \def\@Acrshort#1#2[#3] {%  
3787 \glsdoifexists{#2}%  
3788 {%  
3789 \let\do@gls@link@checkfirsthyper\relax  
  
3790 \def\glslabel{#2}%  
3791 \let\glsifplural\@secondoftwo  
3792 \let\glsapscase\@secondofthree  
3793 \let\glsinsert\@empty  
3794 \def\glscustomtext{%  
3795 \acronymfont{\Glsentryshort{#2}}#3%  
3796 }%
```

Call `\@gls@link` Note that `\@gls@link` sets `\glstype`.

```
3797 \@gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%  
3798 }%  
3799 }
```

`\ACRshort`

```
3800 \newrobustcmd*{\ACRshort}{\@gls@hyp@opt\ns@ACRshort}
```

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

```
3801 \newcommand*{\ns@ACRshort}[2] [] {%  
3802 \new@ifnextchar[{\@ACRshort{#1}{#2}}{\@ACRshort{#1}{#2} []}]%  
3803 }
```

Read in the final optional argument:

```
3804 \def\@ACRshort#1#2[#3] {%  
3805 \glsdoifexists{#2}%  
3806 {%  
3807 \let\do@gls@link@checkfirsthyper\relax
```

```

3808 \def\glslabel{#2}%
3809 \let\glsifplural\@secondoftwo
3810 \let\glsifcaps\@thirdofthree
3811 \let\glsinsert\@empty
3812 \def\glscustomtext{%
3813 \mfirstucMakeUppercase{\acronymfont{\glsentryshort{#2}}#3}%
3814 }%

```

Call `\@gls@link` Note that `\@gls@link` sets `\gls@type`.

```

3815 \@gls@link[#1]{#2}{\csname gls@\gls@type @entryfmt\endcsname}%
3816 }%
3817 }

```

Short plural:

`\acrshortpl`

```

3818 \newrobustcmd*{\acrshortpl}{\@gls@hyp@opt\ns@acrshortpl}

```

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

```

3819 \newcommand*{\ns@acrshortpl}[2] [] {%
3820 \new@ifnextchar[{\acrshortpl{#1}{#2}}{\acrshortpl{#1}{#2} []}%
3821 }

```

Read in the final optional argument:

```

3822 \def\@acrshortpl#1#2[#3] {%
3823 \glsdoifexists{#2}%
3824 {%
3825 \let\do@gls@link@checkfirsthyper\relax

3826 \def\glslabel{#2}%
3827 \let\glsifplural\@firstoftwo
3828 \let\glsifcaps\@firstofthree
3829 \let\glsinsert\@empty
3830 \def\glscustomtext{%
3831 \acronymfont{\glsentryshortpl{#2}}#3%
3832 }%

```

Call `\@gls@link` Note that `\@gls@link` sets `\gls@type`.

```

3833 \@gls@link[#1]{#2}{\csname gls@\gls@type @entryfmt\endcsname}%
3834 }%
3835 }

```

`\Acrshortpl`

```

3836 \newrobustcmd*{\Acrshortpl}{\@gls@hyp@opt\ns@Acrshortpl}

```

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

```

3837 \newcommand*{\ns@Acrshortpl}[2] [] {%
3838 \new@ifnextchar[{\Acrshortpl{#1}{#2}}{\Acrshortpl{#1}{#2} []}%
3839 }

```

Read in the final optional argument:

```
3840 \def\@Acrshortpl#1#2[#3]{%
3841   \glsdoifexists{#2}%
3842   {%
3843     \let\do@gls@link@checkfirsthyper\relax

3844     \def\glslabel{#2}%
3845     \let\glsifplural\@firstoftwo
3846     \let\gls caps case\@secondofthree
3847     \let\glsinsert\@empty
3848     \def\gls custom text{%
3849       \acronymfont{\Glsentryshortpl{#2}}#3%
3850     }%
```

Call \@gls@link Note that \@gls@link sets \gls type.

```
3851   \@gls@link[#1]{#2}{\csname gls@\gls type @entryfmt\endcsname}%
3852   }%
3853 }
```

\ACRshortpl

```
3854 \newrobustcmd*{\ACRshortpl}{\@gls@hyp@opt\@ns@ACRshortpl}
```

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

```
3855 \newcommand*{\@ns@ACRshortpl}[2][ ]{%
3856   \new@ifnextchar[{\@ACRshortpl{#1}{#2}}{\@ACRshortpl{#1}{#2} [ ]}%
3857 }
```

Read in the final optional argument:

```
3858 \def\@ACRshortpl#1#2[#3]{%
3859   \glsdoifexists{#2}%
3860   {%
3861     \let\do@gls@link@checkfirsthyper\relax

3862     \def\glslabel{#2}%
3863     \let\glsifplural\@firstoftwo
3864     \let\gls caps case\@thirdofthree
3865     \let\glsinsert\@empty
3866     \def\gls custom text{%
3867       \mfirstucMakeUppercase{\acronymfont{\glsentryshortpl{#2}}#3}%
3868     }%
```

Call \@gls@link Note that \@gls@link sets \gls type.

```
3869   \@gls@link[#1]{#2}{\csname gls@\gls type @entryfmt\endcsname}%
3870   }%
3871 }
```

\acrlong

```
3872 \newrobustcmd*{\acrlong}{\@gls@hyp@opt\@ns@acrlong}
```


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

```
3873 \newcommand*{\ns@acrlong}[2] [] {%
3874   \new@ifnextchar[{\@acrlong{#1}{#2}}{\@acrlong{#1}{#2} []}%
3875 }
```

Read in the final optional argument:

```
3876 \def\@acrlong#1#2[#3] {%
3877   \glsdoifexists{#2}%
3878   {%
3879     \let\do@gls@link@checkfirsthyper\relax

3880     \def\glslabel{#2}%
3881     \let\glsifplural\@secondoftwo
3882     \let\gls caps case\@firstofthree
3883     \let\glsinsert\@empty
```

Bug fix v4.02 removed `\acronymfont` from `\gls customtext` (`\acronymfont` only designed for short form).

```
3884   \def\gls customtext{%
3885     \glsentrylong{#2}#3%
3886   }%
```

Call `\@gls@link` Note that `\@gls@link` sets `\gls type`.

```
3887   \@gls@link[#1]{#2}{\csname gls@gls type @entryfmt\endcsname}%
3888   }%
3889 }
```

`\Acrlong`

```
3890 \newrobustcmd*{\Acrlong}{\@gls@hyp@opt\ns@Acrlong}
```

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

```
3891 \newcommand*{\ns@Acrlong}[2] [] {%
3892   \new@ifnextchar[{\@Acrlong{#1}{#2}}{\@Acrlong{#1}{#2} []}%
3893 }
```

Read in the final optional argument:

```
3894 \def\@Acrlong#1#2[#3] {%
3895   \glsdoifexists{#2}%
3896   {%
3897     \let\do@gls@link@checkfirsthyper\relax

3898     \def\glslabel{#2}%
3899     \let\glsifplural\@secondoftwo
3900     \let\gls caps case\@secondofthree
3901     \let\glsinsert\@empty
```

Bug fix v4.02 removed `\acronymfont` from `\gls customtext` (`\acronymfont` only designed for short form).

```

3902 \def\glscustomtext{%
3903 \Glsentrylong{#2}#3%
3904 }%

```

Call \@gls@link. Note that \@gls@link sets \glstype.

```

3905 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3906 }%
3907 }

```

\ACRlong

```

3908 \newrobustcmd*{\ACRlong}{\@gls@hyp@opt\@ns@ACRlong}

```

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

```

3909 \newcommand*{\ns@ACRlong}[2] []{%
3910 \new@ifnextchar[{\@ACRlong{#1}{#2}}{\@ACRlong{#1}{#2} []}%
3911 }

```

Read in the final optional argument:

```

3912 \def\@ACRlong#1#2[#3]{%
3913 \glsdoifexists{#2}%
3914 {%
3915 \let\do@gls@link@checkfirsthyper\relax

3916 \def\glslabel{#2}%
3917 \let\glsifplural\@secondoftwo
3918 \let\glscapscase\@thirdofthree
3919 \let\glsinsert\@empty

```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```

3920 \def\glscustomtext{%
3921 \mfirstucMakeUppercase{\glsentrylong{#2}#3}%
3922 }%

```

Call \@gls@link. Note that \@gls@link sets \glstype.

```

3923 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3924 }%
3925 }

```

Short plural:

\acrlongpl

```

3926 \newrobustcmd*{\acrlongpl}{\@gls@hyp@opt\@ns@acrlongpl}

```

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

```

3927 \newcommand*{\ns@acrlongpl}[2] []{%
3928 \new@ifnextchar[{\@acrlongpl{#1}{#2}}{\@acrlongpl{#1}{#2} []}%
3929 }

```

Read in the final optional argument:

```
3930 \def\@acrlongpl#1#2[#3]{%
3931   \glsdoifexists{#2}%
3932   {%
3933     \let\do@gls@link@checkfirsthyper\relax
3934   }
3935   \def\glslabel{#2}%
3936   \let\glsifplural\@firstoftwo
3937   \let\glsifcaps\@firstofthree
3938   \let\glsinsert\@empty
```

Bug fix v4.02 removed `\acronymfont` from `\glscustomtext` (`\acronymfont` only designed for short form).

```
3938   \def\glscustomtext{%
3939     \glsentrylongpl{#2}#3%
3940   }%
```

Call `\@gls@link`. Note that `\@gls@link` sets `\glstype`.

```
3941   \@gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
3942   }%
3943 }
```

`\Acrlongpl`

```
3944 \newrobustcmd*{\Acrlongpl}{\@gls@hyp@opt\ns@Acrlongpl}
```

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

```
3945 \newcommand*{\ns@Acrlongpl}[2] []{%
3946   \new@ifnextchar[{\@Acrlongpl{#1}{#2}}{\@Acrlongpl{#1}{#2} []}%
3947 }
```

Read in the final optional argument:

```
3948 \def\@Acrlongpl#1#2[#3]{%
3949   \glsdoifexists{#2}%
3950   {%
3951     \let\do@gls@link@checkfirsthyper\relax
3952   }
3953   \def\glslabel{#2}%
3954   \let\glsifplural\@firstoftwo
3955   \let\glsifcaps\@secondofthree
3956   \let\glsinsert\@empty
```

Bug fix v4.02 removed `\acronymfont` from `\glscustomtext` (`\acronymfont` only designed for short form).

```
3956   \def\glscustomtext{%
3957     \Glsentrylongpl{#2}#3%
3958   }%
```

Call `\@gls@link`. Note that `\@gls@link` sets `\glstype`.

```
3959   \@gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
3960   }%
3961 }
```

`\ACRlongpl`

```
3962 \newrobustcmd*{\ACRlongpl}{\@gls@hyp@opt\ns@ACRlongpl}
```

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

```
3963 \newcommand*{\ns@ACRlongpl}[2] []{%
3964   \new@ifnextchar[{\@ACRlongpl{#1}{#2}}{\@ACRlongpl{#1}{#2} []}%
3965 }
```

Read in the final optional argument:

```
3966 \def\@ACRlongpl#1#2[#3]{%
3967   \glsdoifexists{#2}%
3968   {%
3969     \let\do@gls@link@checkfirsthyper\relax

3970     \def\glslabel{#2}%
3971     \let\glsifplural\@firstoftwo
3972     \let\gls caps case\@thirdofthree
3973     \let\glsinsert\@empty
```

Bug fix v4.02 removed `\acronymfont` from `\gls customtext` (`\acronymfont` only designed for short form).

```
3974   \def\gls customtext{%
3975     \mfirstucMakeUppercase{\glsentrylongpl{#2}#3}%
3976   }%
```

Call `\@gls@link`. Note that `\@gls@link` sets `\glstype`.

```
3977   \@gls@link[#1]{#2}{\csname gls@gls type @entryfmt\endcsname}%
3978   }%
3979 }
```

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>}
```

```
3980 \newcommand*{\@gls@entry@field}[2]{%
3981   \csname glo@glsdetoklabel{#1}@#2\endcsname
3982 }
```

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

```

3983 \newcommand*\glsletentryfield}[3]{%
3984   \letcs{#1}{glo@\glsdetoklabel{#2}@#3}%
3985 }

```

`\@Gls@entry@field` Generic first letter uppercase version.

```
\@Gls@entry@field{<label>}{<field>}
```

```

3986 \newcommand*\@Gls@entry@field}[2]{%
3987   \letcs@glo@text{glo@\glsdetoklabel{#1}@#2}%
3988   \ifdef@glo@text
3989     {%
3990       \xmakefirstuc@glo@text}%
3991     }%
3992     {%
3993       \PackageError{glossaries}{Either glossary entry
3994         ‘\glsdetoklabel{#1}’ doesn’t exist or the field ‘#2’
3995         doesn’t exist}{Check you have correctly spelt the entry
3996         label and the field name}%
3997     }%
3998 }

```

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`

```
3999 \newcommand*\glsentryname}[1]{\@Gls@entry@field{#1}{name}}
```

`\Glsentryname`

```

4000 \newrobustcmd*\Glsentryname}[1]{%
4001   \@Gls@entryname{#1}%
4002 }

```

`\@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:

```

4003 \newcommand*\@Gls@entryname}[1]{%
4004   \@Gls@entry@field{#1}{name}%
4005 }

```

`\@Gls@acrentryname` Now the behaviour when `\setacronymstyle` is used:

```

4006 \newcommand*\@Gls@acrentryname}[1]{%
4007   \ifglshaslong{#1}%
4008     {%
4009       \letcs@glo@text{glo@\glsdetoklabel{#1}@name}%
4010       \expandafter@gls@getbody@glo@text{}@nil

```

```

4011 \expandafter\ifx\@gls@body\glsentrylong\relax
4012 \expandafter\Glsentrylong\@gls@rest
4013 \else
4014 \expandafter\ifx\@gls@body\glsentryshort\relax
4015 \expandafter\Glsentryshort\@gls@rest
4016 \else
4017 \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.

```

4018     {%
4019         \let\glsentryshort\Glsentryshort
4020         \@glo@text
4021     }%
4022 \else
4023     \xmakefirstuc{\@glo@text}%
4024 \fi
4025 \fi
4026 \fi
4027 }%
4028 {%

```

Not an acronym

```

4029 \@Gls@entry@field{#1}{name}%
4030 }%
4031 }

```

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`

```

4032 \newcommand*{\glsentrydesc}[1]{\@gls@entry@field{#1}{desc}}

```

`\Glsentrydesc`

```

4033 \newrobustcmd*{\Glsentrydesc}[1]{%
4034 \@Gls@entry@field{#1}{desc}%
4035 }

```

Plural form:

`\glsentrydescplural`

```

4036 \newcommand*{\glsentrydescplural}[1]{%
4037 \@gls@entry@field{#1}{descplural}%
4038 }

```

`\Glsentrydescplural`

```
4039 \newrobustcmd*{\Glsentrydescplural}[1]{%
4040   \@Gls@entry@field{#1}{descplural}%
4041 }
```

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`

```
4042 \newcommand*{\glsentrytext}[1]{\@Gls@entry@field{#1}{text}}
```

`\Glsentrytext`

```
4043 \newrobustcmd*{\Glsentrytext}[1]{%
4044   \@Gls@entry@field{#1}{text}%
4045 }
```

Get the plural form:

`\glsentryplural`

```
4046 \newcommand*{\glsentryplural}[1]{%
4047   \@Gls@entry@field{#1}{plural}%
4048 }
```

`\Glsentryplural`

```
4049 \newrobustcmd*{\Glsentryplural}[1]{%
4050   \@Gls@entry@field{#1}{plural}%
4051 }
```

Get the symbol associated with this entry. The argument is the label associated with the entry.

`\glsentrysymbol`

```
4052 \newcommand*{\glsentrysymbol}[1]{%
4053   \@Gls@entry@field{#1}{symbol}%
4054 }
```

`\Glsentrysymbol`

```
4055 \newrobustcmd*{\Glsentrysymbol}[1]{%
4056   \@Gls@entry@field{#1}{symbol}%
4057 }
```

Plural form:

`\glsentrysymbolplural`

```
4058 \newcommand*{\glsentrysymbolplural}[1]{%
4059   \@Gls@entry@field{#1}{symbolplural}%
4060 }
```

glsentrysymbolplural

```
4061 \newrobustcmd*{\Glsentrysymbolplural}[1]{%
4062   \@Gls@entry@field{#1}{symbolplural}%
4063 }
```

Get the entry text to be used when the entry is first used in the document (as specified by the first key when the entry was defined).

\glsentryfirst

```
4064 \newcommand*{\glsentryfirst}[1]{%
4065   \@Gls@entry@field{#1}{first}%
4066 }
```

\Glsentryfirst

```
4067 \newrobustcmd*{\Glsentryfirst}[1]{%
4068   \@Gls@entry@field{#1}{first}%
4069 }
```

Get the plural form (as specified by the firstplural key when the entry was defined).

glsentryfirstplural

```
4070 \newcommand*{\glsentryfirstplural}[1]{%
4071   \@Gls@entry@field{#1}{firstpl}%
4072 }
```

Glsentryfirstplural

```
4073 \newrobustcmd*{\Glsentryfirstplural}[1]{%
4074   \@Gls@entry@field{#1}{firstpl}%
4075 }
```

Display the glossary type with which this entry is associated (as specified by the type key used when the entry was defined)

\glsentrytype

```
4076 \newcommand*{\glsentrytype}[1]{\@Gls@entry@field{#1}{type}}
```

Display the sort text used for this entry. Note that the sort key is sanitized, so unexpected results may occur if the sort key contained commands.

\glsentrysort

```
4077 \newcommand*{\glsentrysort}[1]{%
4078   \@Gls@entry@field{#1}{sort}%
4079 }
```

\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.

```
4080 \newcommand*{\glsentryuseri}[1]{%
4081   \@Gls@entry@field{#1}{useri}%
4082 }
```


`\Glsentryuseri`

```
4083 \newrobustcmd*{\Glsentryuseri}[1]{%
4084   \@Gls@entry@field{#1}{useri}%
4085 }
```

`\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.

```
4086 \newcommand*{\glsentryuserii}[1]{%
4087   \@Gls@entry@field{#1}{userii}%
4088 }
```

`\Glsentryuserii`

```
4089 \newrobustcmd*{\Glsentryuserii}[1]{%
4090   \@Gls@entry@field{#1}{userii}%
4091 }
```

`\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.

```
4092 \newcommand*{\glsentryuseriii}[1]{%
4093   \@Gls@entry@field{#1}{useriii}%
4094 }
```

`\Glsentryuseriii`

```
4095 \newrobustcmd*{\Glsentryuseriii}[1]{%
4096   \@Gls@entry@field{#1}{useriii}%
4097 }
```

`\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.

```
4098 \newcommand*{\glsentryuseriv}[1]{%
4099   \@Gls@entry@field{#1}{useriv}%
4100 }
```

`\Glsentryuseriv`

```
4101 \newrobustcmd*{\Glsentryuseriv}[1]{%
4102   \@Gls@entry@field{#1}{useriv}%
4103 }
```

`\glsentryuserv` Get the fifth user key (as specified by the user5 when the entry was defined).
The argument is the label associated with the entry.

```
4104 \newcommand*{\glsentryuserv}[1]{%
4105   \@Gls@entry@field{#1}{userv}%
4106 }
```

`\Glsentryuserv`

```
4107 \newrobustcmd*{\Glsentryuserv}[1]{%
4108   \@Gls@entry@field{#1}{userv}%
4109 }
```

`\glsentryuservi` Get the sixth user key (as specified by the `user6` when the entry was defined). The argument is the label associated with the entry.

```

4110 \newrobustcmd*{\glsentryuservi}[1]{%
4111   \@gls@entry@field{#1}{uservi}%
4112 }

```

`\Glsentryuservi`

```

4113 \newrobustcmd*{\Glsentryuservi}[1]{%
4114   \@Gls@entry@field{#1}{uservi}%
4115 }

```

`\glsentryshort` Get the short key (as specified by the `short` when the entry was defined). The argument is the label associated with the entry.

```

4116 \newcommand*{\glsentryshort}[1]{\@gls@entry@field{#1}{short}}

```

`\Glsentryshort`

```

4117 \newrobustcmd*{\Glsentryshort}[1]{%
4118   \@Gls@entry@field{#1}{short}%
4119 }

```

`\glsentryshortpl` Get the short plural key (as specified by the `shortplural` when the entry was defined). The argument is the label associated with the entry.

```

4120 \newcommand*{\glsentryshortpl}[1]{\@gls@entry@field{#1}{shortpl}}

```

`\Glsentryshortpl`

```

4121 \newrobustcmd*{\Glsentryshortpl}[1]{%
4122   \@Gls@entry@field{#1}{shortpl}%
4123 }

```

`\glsentrylong` Get the long key (as specified by the `long` when the entry was defined). The argument is the label associated with the entry.

```

4124 \newcommand*{\glsentrylong}[1]{\@gls@entry@field{#1}{long}}

```

`\Glsentrylong`

```

4125 \newrobustcmd*{\Glsentrylong}[1]{%
4126   \@Gls@entry@field{#1}{long}%
4127 }

```

`\glsentrylongpl` Get the long plural key (as specified by the `longplural` when the entry was defined). The argument is the label associated with the entry.

```

4128 \newcommand*{\glsentrylongpl}[1]{\@gls@entry@field{#1}{longpl}}

```

`\Glsentrylongpl`

```

4129 \newrobustcmd*{\Glsentrylongpl}[1]{%
4130   \@Gls@entry@field{#1}{longpl}%
4131 }

```

Short cut macros to access full form:

`\glsentryfull`

```
4132 \newcommand*\glsentryfull}[1]{%
4133   \acrfullformat{\glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%
4134 }
```

`\Glsentryfull`

```
4135 \newrobustcmd*\Glsentryfull}[1]{%
4136   \acrfullformat{\Glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%
4137 }
```

`\glsentryfullpl`

```
4138 \newcommand*\glsentryfullpl}[1]{%
4139   \acrfullformat{\glsentrylongpl{#1}}{\acronymfont{\glsentryshortpl{#1}}}%
4140 }
```

`\Glsentryfullpl`

```
4141 \newrobustcmd*\Glsentryfullpl}[1]{%
4142   \acrfullformat{\Glsentrylongpl{#1}}{\acronymfont{\glsentryshortpl{#1}}}%
4143 }
```

`\glsentrynumberlist` Displays the number list as is.

```
4144 \newcommand*\glsentrynumberlist}[1]{%
4145   \glsdoifexists{#1}%
4146   {%
4147     \@gls@entry@field{#1}{numberlist}%
4148   }%
4149 }
```

`\glsdisplaynumberlist` Formats the number list for the given entry label. Doesn't work with hyperref.

```
4150 \@ifpackageloaded{hyperref} {%
4151   \newcommand*\glsdisplaynumberlist}[1]{%
4152     \GlossariesWarning
4153     {%
4154       \string\glsdisplaynumberlist\space
4155       doesn't work with hyperref.^^JUsing
4156       \string\glsentrynumberlist\space instead%
4157     }%
4158     \glsentrynumberlist{#1}%
4159   }%
4160 }%
4161 {%
4162   \newcommand*\glsdisplaynumberlist}[1]{%
4163     \glsdoifexists{#1}%
4164     {%
4165       \bgroup
```

```

4166     \edef\@glo@label{\glsdetoklabel{#1}}%
4167     \let\@org@glnumberformat\glsnumberformat
4168     \def\glsnumberformat##1{##1}%
4169     \protected@edef\the@numberlist{%
4170       \csname glo@\@glo@label @numberlist\endcsname}%
4171     \def\@gls@numlist@sep{}%
4172     \def\@gls@numlist@nextsep{}%
4173     \def\@gls@numlist@lastsep{}%
4174     \def\@gls@thislist{}%
4175     \def\@gls@donext@def{}%
4176     \renewcommand\do[1]{%
4177       \protected@edef\@gls@thislist{%
4178         \@gls@thislist
4179         \noexpand\@gls@numlist@sep
4180         ##1%
4181       }%
4182       \let\@gls@numlist@sep\@gls@numlist@nextsep
4183       \def\@gls@numlist@nextsep{\glsnumlistsep}%
4184       \@gls@donext@def
4185       \def\@gls@donext@def{%
4186         \def\@gls@numlist@lastsep{\glsnumlistlastsep}%
4187       }%
4188     }%
4189     \expandafter \glsnumlistparser \expandafter{\the@numberlist}%
4190     \let\@gls@numlist@sep\@gls@numlist@lastsep
4191     \@gls@thislist
4192   \egroup
4193 }%
4194 }
4195 }

```

`\glsnumlistsep`

```
4196 \newcommand*{\glsnumlistsep}{, }
```

`\glsnumlistlastsep`

```
4197 \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.

```

4198 \newcommand*{\glshyperlink}[2][\glsentrytext{\@glo@label}]{%
4199   \def\@glo@label{#2}}%
4200 \@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`:

```
4201 \define@key{glossadd}{counter}{\def\@gls@counter{#1}}
```

```
4202 \define@key{glossadd}{format}{\def\@glsnumberformat{#1}}
```

This key is only used by `\glsaddall`:

```
4203 \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`

```
4204 \newrobustcmd*{\glsadd}[2][ ]{%
```

Need to move to horizontal mode if not already in it, but only if not in preamble.

```
4205 \@gls@adjustmode
```

```
4206 \glsdoifexists{#2}%
```

```
4207 {%
```

```
4208 \def\@glsnumberformat{glsnumberformat}%
```

```
4209 \edef\@gls@counter{\csname glo\@glsdetoklabel{#2}@counter\endcsname}%
```

```
4210 \setkeys{glossadd}{#1}%
```

Store the entry's counter in `\theglentrycounter`

```
4211 \@gls@saveentrycounter
```

```
4212 \do@wrglossary{#2}%
```

```
4213 }%
```

```
4214 }
```

`\@gls@adjustmode`

```
4215 \newcommand*{\@gls@adjustmode}{}
```

```
4216 \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`

```
4217 \newrobustcmd*{\glsaddall}[1][ ]{%
```

```
4218 \edef\@glo@type{\@glo@types}%
```

```
4219 \setkeys{glossadd}{#1}%
```

```
4220 \forallglsentries[\@glo@type]{\@glo@entry}{%
```

```
4221 \glsadd[#1]{\@glo@entry}%
```

```
4222 }%
```

```
4223 }
```

`\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.

```
4224 \newrobustcmd*{\glsaddallunused}[1][\@glo@types]{%
4225 \forallglsentries[#1]{\@glo@entry}%
4226 {%
4227     \ifglsused{\@glo@entry}{\glsadd[format=glsignore]{\@glo@entry}}%
4228 }%
4229 }
```

`\glsignore`

```
4230 \newcommand*{\glsignore}[1]{}
```

1.13 Creating associated files

The `\writeist` command creates the associated customized `.ist` `makeindex` style file. While defining this command, some characters have their catcodes temporarily changed to ensure they get written to the `.ist` file correctly. The `makeindex` actual character (usually `@`) is redefined to be a `?`, to allow internal commands to be written to the glossary file output file.

The special characters are stored in `\@gls@actualchar`, `\@gls@encapchar`, `\@gls@levelchar` and `\@gls@quotechar` to make them easier to use later, but don't change these values, because the characters are encoded in the command definitions that are used to escape the special characters (which means that the user no longer needs to worry about `makeindex` special characters).

The symbols and numbers label for group headings are hardwired into the `.ist` file as `glsymbols` and `glsnumbers`, the group titles can be translated (so that `\glsymbolsgroupname` replaces `glsymbols` and `\glsnumbersgroupname` replaces `glsnumbers`) using the command `\glsgetgrouptitle` which is defined in `.`. This is done to prevent any problem characters in `\glsymbolsgroupname` and `\glsnumbersgroupname` from breaking hyperlinks.

`\glsopenbrace` Define `\glsopenbrace` to make it easier to write an opening brace to a file.

```
4231 \edef\glsopenbrace{\expandafter\@gobble\string\{}
```

`\glsclosebrace` Define `\glsclosebrace` to make it easier to write an opening brace to a file.

```
4232 \edef\glsclosebrace{\expandafter\@gobble\string\}}
```

`\glsbackslash` Define `\glsbackslash` to make it easier to write a backslash to a file.

```
4233 \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.

```
4234 \edef\glsquote#1{\string"#1\string"}
```

`\glspercentchar` Define `\glspercentchar` to make it easier to write a percent character to a file.

```
4235 \edef\glspercentchar{\expandafter@gobble\string\%}
```

`\glstildechar` Define `\glstildechar` to make it easier to write a tilde character to a file.

```
4236 \edef\glstildechar{\string~}
```

`\@glsfirstletter` Define the first letter to come after the digits 0,...,9. Only required for xindy.

```
4237 \ifglsxindy
4238   \newcommand*{\@glsfirstletter}{A}
4239 \fi
```

`\GlsSetXdyFirstLetterAfterDigits` Sets the first letter to come after the digits 0,...,9.

```
4240 \ifglsxindy
4241   \newcommand*{\GlsSetXdyFirstLetterAfterDigits}[1]{%
4242     \renewcommand*{\@glsfirstletter}{#1}}
4243 \else
4244   \newcommand*{\GlsSetXdyFirstLetterAfterDigits}[1]{%
4245     \glsnoxindywarning\GlsSetXdyFirstLetterAfterDigits}
4246 \fi
```

`\@glsminrange` Define the minimum number of successive location references to merge into a range.

```
4247 \newcommand*{\@glsminrange}{2}
```

`\GlsSetXdyMinRangeLength` 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.

```
4248 \ifglsxindy
4249   \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
4250     \renewcommand*{\@glsminrange}{#1}}
4251 \else
4252   \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
4253     \glsnoxindywarning\GlsSetXdyMinRangeLength}
4254 \fi
```

`\writeist`

```
4255 \ifglsxindy
    Code to use if xindy is required.
4256   \def\writeist{%
    Define write register if not already defined
4257     \ifundef{\glswrite}{\newwrite\glswrite}{}%
    Update attributes list
4258     \@gls@addpredefinedattributes
    Open the file.
4259     \openout\glswrite=\istfilename
```

Write header comment at the start of the file

```
4260 \write\glswrite{;; xindy style file created by the glossaries
4261 package}%
4262 \write\glswrite{;; for document '\jobname' on
4263 \the\year-\the\month-\the\day}%
```

Specify the required styles

```
4264 \write\glswrite{^^J; required styles^^J}
4265 \@for\@xdystyle:=\@xdyrequiredstyles\do{%
4266 \ifx\@xdystyle\@empty
4267 \else
4268 \protected@write\glswrite{}{(require
4269 \string"\@xdystyle.xdy\string")}%
4270 \fi
4271 }%
```

List the allowed attributes (possible values used by the format key)

```
4272 \write\glswrite{^^J%
4273 ; list of allowed attributes (number formats)^^J}%
4274 \write\glswrite{(define-attributes ((\@xdyattributes)))}%
```

Define any additional alphabets

```
4275 \write\glswrite{^^J; user defined alphabets^^J}%
4276 \write\glswrite{\@xdyuseralphabets}%
```

Define location classes.

```
4277 \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.

```
4278 \@for\@gls@classI:=\@gls@xdy@locationlist\do{%
```

Case were $\langle Hprefix \rangle$ is empty:

```
4279 \protected@write\glswrite{}{(define-location-class
4280 \string"\@gls@classI\string"^^J\space\space\space
4281 (
4282 :sep "{}{"
4283 \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
4284 :sep "}"
4285 )
4286 ^^J\space\space\space
4287 :min-range-length \@glsminrange^^J%
4288 )
4289 }%
```

Nested iteration over all classes:

```
4290 {%
4291 \@for\@gls@classII:=\@gls@xdy@locationlist\do{%
4292 \protected@write\glswrite{}{(define-location-class
4293 \string"\@gls@classII-\@gls@classI\string"
4294 ^^J\space\space\space
4295 (
```



```

4296             :sep "{"
4297             \csname @gls@xdy@Lclass@\@gls@classII\endcsname\space
4298             :sep "}"
4299             \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
4300             :sep "}"
4301         )
4302         ^^J\space\space\space
4303         :min-range-length \@glsminrange^^J%
4304     )
4305 }%
4306 }%
4307 }%
4308 }%

```

User defined location classes (needs checking for new location format).

```

4309 \write\glswrite{^^J; user defined location classes}%
4310 \write\glswrite{\@xdyuserlocationdefs}%

```

Cross-reference class. (The unverified option is used as the cross-references are supplied using the list of labels along with the optional argument for `\glsseeformat` which xindy won't recognise.)

```

4311 \write\glswrite{^^J; define cross-reference class^^J}%
4312 \write\glswrite{(define-crossref-class \string"see\string"
4313                 :unverified )}%

```

Define how cross-references should be displayed. This adds an empty set of braces after the cross-referencing information allowing for the final argument of `\glsseeformat` which gets ignored. (When using `makeindex` this final argument contains the location information which is not required.)

```

4314 \write\glswrite{(markup-crossref-list
4315                 :class \string"see\string"^^J\space\space\space
4316                 :open \string"\string\glsseeformat\string"
4317                 :close \string"{}\string")}%

```

List the order to sort the classes.

```

4318 \write\glswrite{^^J; define the order of the location classes}%
4319 \write\glswrite{(define-location-class-order
4320                 (\@xdylocationclassorder))}%

```

Specify what to write to the start and end of the glossary file.

```

4321 \write\glswrite{^^J; define the glossary markup^^J}%

4322 \write\glswrite{(markup-index^^J\space\space\space
4323                 :open \string"\string
4324                 \glossarysection[\string\glossarytoctitle]{\string
4325                 \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`)

```

4326 \@for\@this@ctr:=\@xdycounters\do{%
4327     {%

```

```

4328     \@for\@this@attr:=\@xdyattributelist\do{%
4329         \protected@write\glswrite{ }\string\providecommand*%
4330         \expandafter\string
4331         \csname glsX\@this@ctr X\@this@attr\endcsname[2]%
4332         {%
4333             \string\setentrycounter
4334             [\expandafter\@gobble\string\#1]{\@this@ctr}%
4335             \expandafter\string
4336             \csname\@this@attr\endcsname
4337             {\expandafter\@gobble\string\#2}%
4338         }%
4339     }%
4340 }%
4341 }%
4342 }%

```

Add the end part of the open tag and the rest of the markup-index information:

```

4343     \write\glswrite{%
4344         \string\begin
4345         {theglossary}\string\glossaryheader\glstildechar n\string" ^^J\space
4346         \space\space:close \string"\glpercentchar\glstildechar n\string
4347         \end{theglossary}\string\glossarypostamble
4348         \glstildechar n\string" ^^J\space\space\space
4349         :tree)}}%

```

Specify what to put between letter groups

```

4350     \write\glswrite{(markup-letter-group-list
4351         :sep \string"\string\glsgroupskip\glstildechar n\string")}%

```

Specify what to put between entries

```

4352     \write\glswrite{(markup-indexentry
4353         :open \string"\string\relax \string\glsresetentrylist
4354         \glstildechar n\string")}%

```

Specify how to format entries

```

4355     \write\glswrite{(markup-locclass-list :open
4356         \string"\glsopenbrace\string\glossaryentrynumbers
4357         \glsopenbrace\string\relax\space \string"^^J\space\space\space
4358         :sep \string", \string"
4359         :close \string"\glsclosebrace\glsclosebrace\string")}%

```

Specify how to separate location numbers

```

4360     \write\glswrite{(markup-locref-list
4361         :sep \string"\string\delimN\space\string")}%

```

Specify how to indicate location ranges

```

4362     \write\glswrite{(markup-range
4363         :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.

```

4364     \@onelevel@sanitize\gls@suffixF
4365     \@onelevel@sanitize\gls@suffixFF

```

```

4366 \ifx\gls@suffixF\@empty
4367 \else
4368 \write\glswrite{(markup-range
4369 :close "\gls@suffixF" :length 1 :ignore-end)}%
4370 \fi
4371 \ifx\gls@suffixFF\@empty
4372 \else
4373 \write\glswrite{(markup-range
4374 :close "\gls@suffixFF" :length 2 :ignore-end)}%
4375 \fi

```

Specify how to format locations.

```

4376 \write\glswrite{^^J; define format to use for locations^^J}%
4377 \write\glswrite{\@xdylocref}%

```

Specify how to separate letter groups.

```

4378 \write\glswrite{^^J; define letter group list format^^J}%
4379 \write\glswrite{(markup-letter-group-list
4380 :sep \string"\string\glsgroupskip\glstildechar n\string")}%

```

Define letter group headings.

```

4381 \write\glswrite{^^J; letter group headings^^J}%
4382 \write\glswrite{(markup-letter-group
4383 :open-head \string"\string\glsgroupheading
4384 \glsopenbrace\string"^^J\space\space\space
4385 :close-head \string"\glsclosebrace\string")}%

```

Define additional letter groups.

```

4386 \write\glswrite{^^J; additional letter groups^^J}%
4387 \write\glswrite{\@xdylettergroups}%

```

Define additional sort rules

```

4388 \write\glswrite{^^J; additional sort rules^^J}
4389 \write\glswrite{\@dysortrules}%

```

Close the style file

```

4390 \closeout\glswrite

```

Suppress any further calls.

```

4391 \let\writeist\relax
4392 }
4393 \else

```

Code to use if makeindex is required.

```

4394 \edef\@gls@actualchar{\string?}
4395 \edef\@gls@encapchar{\string|}
4396 \edef\@gls@levelchar{\string!}
4397 \edef\@gls@quotechar{\string"}
4398 \def\writeist{\relax
4399 \ifundef{\glswrite}{\newwrite\glswrite}{}\relax
4400 \openout\glswrite=\istfilename
4401 \write\glswrite{\glspercentchar\space makeindex style file

```

```

4402     created by the glossaries package}
4403 \write\glswrite{\glspersentchar\space for document
4404   '\jobname' on \the\year-\the\month-\the\day}
4405 \write\glswrite{actual '@gls@actualchar'}
4406 \write\glswrite{encap '@gls@encapchar'}
4407 \write\glswrite{level '@gls@levelchar'}
4408 \write\glswrite{quote '@gls@quotechar'}
4409 \write\glswrite{keyword \string"\string\glossaryentry\string"}
4410 \write\glswrite{preamble \string"\string\glossarysection[\string
4411   \glossarytoctitle]{\string\glossarytitle}\string
4412   \glossarypreamble\string\n\string\begin{theglossary}\string
4413   \glossaryheader\string\n\string"}
4414 \write\glswrite{postamble \string"\string%\string\n\string
4415   \end{theglossary}\string\glossarypostamble\string\n
4416   \string"}
4417 \write\glswrite{group_skip \string"\string\glsgroupskip\string\n
4418   \string"}
4419 \write\glswrite{item_0 \string"\string%\string\n\string"}
4420 \write\glswrite{item_1 \string"\string%\string\n\string"}
4421 \write\glswrite{item_2 \string"\string%\string\n\string"}
4422 \write\glswrite{item_01 \string"\string%\string\n\string"}
4423 \write\glswrite{item_x1
4424   \string"\string\relax \string\glsresetentrylist\string\n
4425   \string"}
4426 \write\glswrite{item_12 \string"\string%\string\n\string"}
4427 \write\glswrite{item_x2
4428   \string"\string\relax \string\glsresetentrylist\string\n
4429   \string"}

4430 \write\glswrite{delim_0 \string"\string\{\string
4431   \glossaryentrynumbers\string\{\string\relax \string"}
4432 \write\glswrite{delim_1 \string"\string\{\string
4433   \glossaryentrynumbers\string\{\string\relax \string"}
4434 \write\glswrite{delim_2 \string"\string\{\string
4435   \glossaryentrynumbers\string\{\string\relax \string"}
4436 \write\glswrite{delim_t \string"\string\}\string\}\string"}
4437 \write\glswrite{delim_n \string"\string\delimN \string"}
4438 \write\glswrite{delim_r \string"\string\delimR \string"}
4439 \write\glswrite{headings_flag 1}
4440 \write\glswrite{heading_prefix
4441   \string"\string\glsgroupheading\string\{\string"}
4442 \write\glswrite{heading_suffix
4443   \string"\string\}\string\relax
4444   \string\glsresetentrylist \string"}
4445 \write\glswrite{symhead_positive \string"glssymbols\string"}
4446 \write\glswrite{numhead_positive \string"glssymbols\string"}
4447 \write\glswrite{page_compositor \string"glscpositor\string"}
4448 \@gls@escbsdq\gls@suffixF
4449 \@gls@escbsdq\gls@suffixFF
4450 \ifx\gls@suffixF\@empty

```

```

4451 \else
4452   \write\glswrite{suffix_2p \string"\gls@suffixF\string"}
4453 \fi
4454 \ifx\gls@suffixFF\@empty
4455 \else
4456   \write\glswrite{suffix_3p \string"\gls@suffixFF\string"}
4457 \fi
4458 \closeout\glswrite
4459 \let\writeist\relax
4460 }
4461 \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`

```

4462 \newcommand{\noist}{%
  Update attributes list
4463   \@gls@addpredefinedattributes
4464   \let\writeist\relax
4465 }

```

`\@makeglossary` is an internal command that takes an argument indicating the glossary type. This command will create the glossary file required by `makeindex` for the given glossary type, using the extension supplied by the `<out-ext>` parameter used in `\newglossary` (and it will also activate the `\glossary` command, and create the customized `.ist` `makeindex` style file).

Note that you can't use `\@makeglossary` for only some of the defined glossaries. You either need to have a `\makeglossary` for all glossaries or none (otherwise you will end up with a situation where `TEX` is trying to write to a non-existent file). The relevant glossary must be defined prior to using `\@makeglossary`.

`\@makeglossary`

```

4466 \newcommand*{\@makeglossary}[1]{%
4467   \ifglossaryexists{#1}%
4468   {%

```

Only create a new write if `savewrites=false` otherwise create a token to collect the information.

```

4469   \ifglssavewrites
4470     \expandafter\newtoks\csname glo@#1@filetok\endcsname
4471   \else
4472     \expandafter\newwrite\csname glo@#1@file\endcsname
4473     \expandafter\@glsopenfile\csname glo@#1@file\endcsname{#1}%
4474   \fi
4475   \@gls@renewglossary
4476   \writeist

```

```

4477 }%
4478 {%
4479   \PackageError{glossaries}%
4480   {Glossary type ‘#1’ not defined}%
4481   {New glossaries must be defined before using \string\makeglossary}%
4482 }%
4483 }

```

`\@glsopenfile` Open write file associated with the given glossary.

```

4484 \newcommand*{\@glsopenfile}[2]{%
4485   \immediate\openout#1=\jobname.\csname @glotype@#2@out\endcsname
4486   \PackageInfo{glossaries}{Writing glossary file
4487     \jobname.\csname @glotype@#2@out\endcsname}%
4488 }

```

`\@closegls`

```

4489 \newcommand*{\@closegls}[1]{%
4490   \closeout\csname glo@#1@file\endcsname
4491 }
4492 %   \end{macrocode}
4493 %\end{macro}
4494 %
4495 %\begin{macro}{\@gls@automake}
4496 %\changes{4.08}{2014-07-30}{new}
4497 %   \begin{macrocode}
4498 \ifglsxindy
4499   \newcommand*{\@gls@automake}[1]{%
4500     \ifglossaryexists{#1}
4501     {%
4502       \@closegls{#1}%
4503       \ifdefstring{\glsorder}{letter}%
4504         {\def\@gls@order{-M ord/letorder }}%
4505         {\let\@gls@order\@empty}%
4506       \ifcsundef{@xdy@#1@language}%
4507         {\let\@gls@langmod\@xdy@main@language}%
4508         {\letcs\@gls@langmod{@xdy@#1@language}}%
4509       \edef\@gls@dothiswrite{\noexpand\write18{xindy
4510         -I xindy
4511         \@gls@order
4512         -L \@gls@langmod\space
4513         -M \gls@istfilebase\space
4514         -C \gls@codepage\space
4515         -t \jobname.\csuse{@glotype@#1@log}
4516         -o \jobname.\csuse{@glotype@#1@in}
4517         \jobname.\csuse{@glotype@#1@out}}%
4518       }%
4519       \@gls@dothiswrite
4520     }%
4521     {%

```

```

4522     \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
4523   }%
4524 }
4525 \else
4526 \newcommand*{\@gls@automake}[1]{%
4527   \ifglossaryexists{#1}
4528   {%
4529     \@closegls{#1}%
4530     \ifdefstring{glsorder}{letter}%
4531     {\def\@gls@order{-1 }}%
4532     {\let\@gls@order\@empty}%
4533     \edef\@gls@dothiswrite{\noexpand\write18{makeindex \@gls@order
4534       -s \istfilename\space
4535       -t \jobname.\csuse{@glotype@#1@log}
4536       -o \jobname.\csuse{@glotype@#1@in}
4537       \jobname.\csuse{@glotype@#1@out}}}%
4538   }%
4539   \@gls@dothiswrite
4540 }%
4541 {%
4542   \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
4543 }%
4544 }
4545 \fi

```

`\warn@nomakeglossaries` Issue warning that `\makeglossaries` hasn't been used.

```
4546 \newcommand*{\@warn@nomakeglossaries}{}

```

Only use this if warning if `\printglossary` has been used without `\makeglossaries`

```
4547 \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`

```
4548 \newcommand*{\makeglossaries}{%

```

Define the write used for style file also used for all other output files if `savewrites=true`.

```
4549 \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.

```
4550 \protected@write\@auxout{}{\string\providecommand\string\@glsorder[1]{}

```

```
4551 \protected@write\@auxout{}{\string\providecommand\string\@istfilename[1]{}

```

Write the name of the style file to the aux file (needed by makeglossaries)

```
4552 \protected@write\auxout{}\string\@istfilename{\istfilename}}%  
4553 \protected@write\auxout{}\string\@glsorder{\glsorder}}
```

Iterate through each glossary type and activate it.

```
4554 \@for\@glo@type:=\@glo@types\do{%  
4555 \ifthenelse{\equal{\@glo@type}{}}{}}{%  
4556 \@makeglossary{\@glo@type}}%  
4557 }%
```

New glossaries must be created before \makeglossaries so disable \newglossary.

```
4558 \renewcommand*\newglossary[4] []{%  
4559 \PackageError{glossaries}{New glossaries  
4560 must be created before \string\makeglossaries}{You need  
4561 to move \string\makeglossaries\space after all your  
4562 \string\newglossary\space commands}}%
```

Any subsequent instances of this command should have no effect

```
4563 \let\@makeglossary\relax  
4564 \let\makeglossary\relax  
4565 \let\makeglossaries\relax
```

Disable all commands that have no effect after \makeglossaries

```
4566 \@disable@onlypremakeg
```

Allow see key:

```
4567 \let\gls@checkseeallowed\relax
```

Suppress warning about no \makeglossaries

```
4568 \let\warn@nomakeglossaries\relax
```

Activate warning about missing \printglossary

```
4569 \def\warn@noprintglossary{%  
4570 \GlossariesWarningNoLine{No \string\printglossary\space  
4571 or \string\printglossaries\space  
4572 found.^^J(Remove \string\makeglossaries\space if you don't want  
4573 any glossaries.)^^JThis document will not have a glossary}%  
4574 }%
```

Declare list parser for \glsdisplaynumberlist

```
4575 \ifglssavenumberlist  
4576 \edef\@gls@dodeflistparser{\noexpand\DeclareListParser  
4577 {\noexpand\glsnumlistparser}{\delimN}}%  
4578 \@gls@dodeflistparser  
4579 \fi
```

Prevent user from also using \makenoidxglossaries

```
4580 \let\makenoidxglossaries\@no@makeglossaries
```

Prohibit sort key in printgloss family:

```
4581 \renewcommand*\@printgloss@setsort}{%  
4582 \let\@glo@assign@sortkey\@glo@no@assign@sortkey  
4583 }%
```


Check the automake setting:

```
4584 \ifglsautomake
4585   \renewcommand*{\@gls@doautomake}{%
4586     \@for\@gls@type:=\@glo@types\do{%
4587       \ifdefempty{\@gls@type}{}%
4588       {\@gls@automake{\@gls@type}}%
4589     }%
4590   }%
4591 \fi
4592 }
```

Must occur in the preamble:

```
4593 \@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`

```
4594 \let\makeglossary\makeglossaries
```

If `\makeglossaries` hasn't been used, issue a warning. Also issue a warning if neither `\printglossaries` nor `\printglossary` have been used.

```
4595 \AtEndDocument{%
4596   \warn@nomakeglossaries
4597   \warn@noprintglossary
4598 }
```

`\makenoidxglossaries` Analogous to `\makeglossaries` this activates the commands needed for `\printnoidxglossary`

```
4599 \newcommand*\makenoidxglossaries{%
```

Redefine empty glossary warning:

```
4600   \renewcommand{\@gls@noref@warn}[1]{%
4601     \GlossariesWarning{Empty glossary for
4602       \string\printnoidxglossary[type={##1}].
4603     Rerun may be required (or you may have forgotten to use
4604     commands like \string\gls).}%
4605   }%
```

Don't escape makeindex/xindy characters

```
4606   \let\@gls@checkmkidxchars\@gobble
```

Write glossary information to aux instead of glossary files

```
4607   \let\@do@wrglossary\gls@noidxglossary
```

Switch on group headings that use the character code:

```
4608   \let\@gls@getgrouptitle\@gls@noidx@getgrouptitle
```

Allow see key:

```
4609 \let\gls@checkseeallowed\relax
```

Redefine cross-referencing macro:

```
4610 \renewcommand{\@do@seeglossary}[2]{%
4611   \edef\@gls@label{\glsdetoklabel{##1}}%
4612   \protected@write\@auxout{}{%
4613     \string\@gls@reference
4614     {\csname glo@\@gls@label @type\endcsname}%
4615     {\@gls@label}%
4616     {%
4617       \string\glsseeformat##2}%
4618     }%
4619   }%
4620 }%
```

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.

```
4621 \AtBeginDocument
4622 {%
4623   \write\@auxout{\string\providecommand\string\@gls@reference[3]{}%
4624 }%
```

Change warning about no glossares

```
4625 \def\warn@noprintglossary{%
4626   \GlossariesWarningNoLine{No \string\printnoidxglossary\space
4627   or \string\printnoidxglossaries ^^J
4628   found. (Remove \string\makenoidxglossaries\space if you
4629   don't want any glossaries.)^^JThis document will not have a glossary}%
4630 }%
```

Suppress warning about no \makeglossaries

```
4631 \let\warn@nomakeglossaries\relax
```

Prevent user from also using \makeglossaries

```
4632 \let\makeglossaries\@no@makeglossaries
```

Allow sort key in printgloss family:

```
4633 \renewcommand*{\@printgloss@setsort}{%
4634   \let\@glo@assign@sortkey\@glo@assign@sortkey
```

Initialise default sort order:

```
4635   \def\@glo@sorttype{\@glo@default@sorttype}%
4636 }%
```

All entries must be defined in the preamble:

```
4637 \renewcommand*\new@glossaryentry[2]{%
4638   \PackageError{glossaries}{Glossary entries must be
4639   defined in the preamble^^Jwhen you use
4640   \string\makenoidxglossaries}%
4641   {Either move your definitions to the preamble or use
```

```

4642     \string\makeglossaries}%
4643 }%

  Redefine \glsentrynumberlist
4644 \renewcommand*\glsentrynumberlist}[1]{%
4645   \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
4646   \ifdef\@gls@loclist
4647   {%
4648     \glsnoidxloclist{\@gls@loclist}%
4649   }%
4650   {%
4651     \ifglsentryexists{##1}%
4652     {%
4653       \GlossariesWarning{Missing location list for ‘##1’. Either
4654         a rerun is required or you haven’t referenced the entry.}%
4655     }%
4656     {%
4657       \PackageError{glossaries}{Glossary entry ‘##1’ has not been
4658         defined.}{}%
4659     }%
4660   }%
4661 }%

  Redefine \glsdisplaynumberlist
4662 \renewcommand*\glsdisplaynumberlist}[1]{%
4663   \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
4664   \ifdef\@gls@loclist
4665   {%
4666     \def\@gls@noidxloclist@sep{%
4667       \def\@gls@noidxloclist@sep{%
4668         \def\@gls@noidxloclist@sep{%
4669           \glsnumlistsep
4670         }%
4671       \def\@gls@noidxloclist@finalsep{\glsnumlistlastsep}%
4672     }%
4673   }%
4674   \def\@gls@noidxloclist@finalsep{}%
4675   \def\@gls@noidxloclist@prev{}%
4676   \forlistloop{\glsnoidxdisplayloclisthandler}{\@gls@loclist}%
4677   \@gls@noidxloclist@finalsep
4678   \@gls@noidxloclist@prev
4679 }%
4680 {%
4681   ??\ifglsentryexists{##1}%
4682   {%
4683     \GlossariesWarning{Missing location list for ‘##1’. Either
4684       a rerun is required or you haven’t referenced the entry.}%
4685   }%
4686   {%
4687     \PackageError{glossaries}{Glossary entry ‘##1’ has not been

```

```

4688         defined.}{}%
4689     }%
4690 }%
4691 }%

```

Provide a generic way of iterating through the number list:

```

4692 \renewcommand*\glsnumberlistloop}[3]{%
4693     \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
4694     \let\@gls@org@glsnoidxdisplayloc\glsnoidxdisplayloc
4695     \let\@gls@org@glsseeformat\glsseeformat
4696     \let\glsnoidxdisplayloc##2\relax
4697     \let\glsseeformat##3\relax
4698     \ifdef\@gls@loclist
4699     {%
4700         \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
4701     }%
4702     {%
4703         \ifglsentryexists{##1}%
4704         {%
4705             \GlossariesWarning{Missing location list for ‘##1’. Either
4706                 a rerun is required or you haven’t referenced the entry.}%
4707         }%
4708         {%
4709             \PackageError{glossaries}{Glossary entry ‘##1’ has not been
4710                 defined.}{}%
4711         }%
4712     }%
4713     \let\glsnoidxdisplayloc\@gls@org@glsnoidxdisplayloc
4714     \let\glsseeformat\@gls@org@glsseeformat
4715 }%

```

Modify sanitize sort function

```

4716 \let\@gls@sanitizesort\@gls@noidx@sanitizesort
4717 \let\@gls@nosanitizesort\@gls@noidx@nosanitizesort
4718 \@gls@noidx@setsanitizesort
4719 }

```

Preamble-only command:

```

4720 \@onlypreamble{\makenoidxglossaries}

```

`\glsnumberlistloop` `\glsnumberlistloop{<label>}{<handler>}`

```

4721 \newcommand*\glsnumberlistloop}[2]{%
4722     \PackageError{glossaries}{\string\glsnumberlistloop\space
4723         only works with \string\makenoidxglossaries}{}%
4724 }

```

`numberlistloophandler` Handler macro for `\glsnumberlistloop`. (The argument should be in the form `\glsnoidxdisplayloc{<prefix>}{<counter>}{<format>}{<n>}`)

```

4725 \newcommand*\glsnoidxnumberlistloophandler}[1]{%
4726   #1%
4727 }

```

`\@no@makeglossaries` Can't use both `\makeglossaries` and `\makenoidxglossaries`

```

4728 \newcommand*\@no@makeglossaries}{%
4729   \PackageError{glossaries}{You can't use both
4730   \string\makeglossaries\space and \string\makenoidxglossaries}%
4731   {Either use one or other (or none) of those commands but not both
4732   together.}%
4733 }

```

`\@gls@noref@warn` Warning when no instances of `\@gls@reference` found.

```

4734 \newcommand{\@gls@noref@warn}[1]{%
4735   \GlossariesWarning{\string\makenoidxglossaries\space
4736   is required to make \string\printnoidxglossary[type={#1}] work}%
4737 }

```

`\gls@noidxglossary` Write the glossary information to the aux file:

```

4738 \newcommand*\gls@noidxglossary}{%
4739   \protected@write\@auxout{}{%
4740     \string\@gls@reference
4741     {\csname glo@\@gls@label @type\endcsname}%
4742     {\@gls@label}%
4743     {\string\glsnoidxdisplayloc
4744     {\@glo@counterprefix}%
4745     {\@gls@counter}%
4746     {\@glsnumberformat}%
4747     {\@glslocref}%
4748     }%
4749   }%
4750 }

```

1.14 Writing information to associated files

`\istfile` Deprecated.

```

4751 \def\istfile{\glswrite}

```

At the end of the document, the files should be created if `savewrites=true`.

```

4752 \AtEndDocument{%
4753   \glswritefiles
4754 }

```

`\@glswritefiles` Only write the files if `savewrites=true`

```

4755 \newcommand*\@glswritefiles}{%
  Iterate through all the glossaries
4756   \forallglossaries{\@glo@type}{%

```

Check for empty glossaries (patch provided by Patrick Häcker)

```
4757 \ifcsundef{glo@\@glo@type @filetok}%
4758 {%
4759 \def\gls@tmp{%
4760 }%
4761 {%
4762 \edef\gls@tmp{\expandafter\the
4763 \csname glo@\@glo@type @filetok\endcsname}%
4764 }%
4765 \ifx\gls@tmp\@empty
4766 \ifx\@glo@type\glsdefaulttype
4767 \GlossariesWarningNoLine{Glossary ‘\@glo@type’ has no
4768 entries.^^JRemember to use package option ‘nomain’ if
4769 you
4770 don’t want to^^Juse the main glossary}%
4771 \else
4772 \GlossariesWarningNoLine{Glossary ‘\@glo@type’ has no
4773 entries}%
4774 \fi
4775 \else
4776 \@glsopenfile{\glswrite}{\@glo@type}%
4777 \immediate\write\glswrite{%
4778 \expandafter\the
4779 \csname glo@\@glo@type @filetok\endcsname}%
4780 \immediate\closeout\glswrite
4781 \fi
4782 }%
4783 }
```

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`

```
4784 \if@gls@docloaded
4785 \else
4786 \renewcommand*{\glossary}[1][main]{\gls@glossary{#1}}
4787 \fi
```

The associated number should be stored in `\theglsentrycounter` before using `\gls@glossary`.

`\gls@glossary`

```
4788 \newcommand*{\gls@glossary}[1]{%
4789   \@gls@glossary{#1}%
4790 }
```

`\@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.

```
4791 \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`

```
4792 \newcommand{\@gls@renewglossary}{%
4793   \gdef\@gls@glossary##1{\@bsphack\begingroup\gls@wrglossary{##1}}%
4794   \let\@gls@renewglossary\@empty
4795 }
```

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`

```
4796 \newcommand*{\gls@wrglossary}[2]{%
4797   \ifglssavewrites
4798     \protected@edef\@gls@tmp{\the\csname glo@#1@filetok\endcsname#2}%
4799     \expandafter\global\expandafter\csname glo@#1@filetok\endcsname
4800       \expandafter{\@gls@tmp^^J}%
4801   \else
4802     \ifcsdef{glo@#1@file}%
4803     {%
4804       \expandafter\protected@write\csname glo@#1@file\endcsname{%
4805         \gls@disablepagerefexpansion}{#2}%
4806     }%
4807     {%
4808       \ifignoredglossary{#1}{}%
4809       {%
4810         \GlossariesWarning{No file defined for glossary '#1'}%
4811       }%
4812     }%
4813   \fi
4814   \endgroup\@esphack
4815 }
```

`\do@wrglossary`

```
4816 \newcommand*\do@wrglossary}[1]{%
4817   \ifglsindexonlyfirst
4818     \ifglsused{#1}{\do@wrglossary{#1}}%
4819   \else
4820     \do@wrglossary{#1}%
4821   \fi
4822 }
```

`@protected@pagefmts` List of page formats to be protected against expansion.

```
4823 \newcommand\gls@protected@pagefmts{%
4824   \gls@numberpage,\gls@alphpage,\gls@Alphpage,\gls@romanpage,\gls@Romanpage%
4825 }
```

`blepagerefexpansion`

```
4826 \newcommand*\gls@disablepagerefexpansion{%
4827   \@for\gls@this:=\gls@protected@pagefmts\do
4828   {%
4829     \expandafter\let\gls@this\relax
4830   }%
4831 }
```

`\gls@alphpage`

```
4832 \newcommand*\gls@alphpage{\@alph\c@page}
```

`\gls@Alphpage`

```
4833 \newcommand*\gls@Alphpage{\@Alph\c@page}
```

`\gls@numberpage`

```
4834 \newcommand*\gls@numberpage{\number\c@page}
```

`\gls@romanpage`

```
4835 \newcommand*\gls@romanpage{\romannumeral\c@page}
```

`\gls@Romanpage`

```
4836 \newcommand*\gls@Romanpage{\@Roman\c@page}
```

`saddprotectedpagefmt`

```
\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).

```
4837 \newcommand*\glsaddprotectedpagefmt}[1]{%
4838   \eappto\gls@protected@pagefmts{\expandonce{\csname gls#1page\endcsname}}%
4839   \csedef{gls#1page}{\expandonce{\csname#1\endcsname}\noexpand\c@page}%
4840   \eappto\@wrglossarynumberhook{%
```



```

4841 \noexpand\let\expandonce{\csname org@gls#1\endcsname}%
4842 \expandonce{\csname#1\endcsname}%
4843 \noexpand\def\expandonce{\csname#1\endcsname}{%
4844 \noexpand\@wrglossary@pageformat
4845 \expandonce{\csname gls#1page\endcsname}%
4846 \expandonce{\csname org@gls#1\endcsname}%
4847 }%
4848 }%
4849 }

```

rglossarynumberhook Hook used by \@do@wrglossary

```
4850 \newcommand*\@wrglossarynumberhook{}
```

glossary@pageformat

```

4851 \newcommand{\@wrglossary@pageformat}[3]{%
4852 \ifx#3\c@page #1\else #2#3\fi
4853 }

```

\@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.

```

4854 \newcommand*{\@do@wrglossary}[1]{%
4855 \begingroup

```

First a bit of hackery to prevent premature expansion of \c@page. Store original definitions:

```

4856 \let\orgthe\the
4857 \let\orgnumber\number
4858 \let\orgromannumeral\romannumeral
4859 \let\orgalph\@alph
4860 \let\orgAlph\@Alph
4861 \let\orgRoman\@Roman

```

Redefine:

```

4862 \def\the##1{%
4863 \ifx##1\c@page \gls@numberpage\else\orgthe##1\fi}%
4864 \def\number##1{%
4865 \ifx##1\c@page \gls@numberpage\else\orgnumber##1\fi}%
4866 \def\romannumeral##1{%
4867 \ifx##1\c@page \gls@romanpage\else\orgromannumeral##1\fi}%
4868 \def\@Roman##1{%
4869 \ifx##1\c@page \gls@Romanpage\else\orgRoman##1\fi}%
4870 \def\@alph##1{%
4871 \ifx##1\c@page \gls@alphpage\else\orgalph##1\fi}%
4872 \def\@Alph##1{%
4873 \ifx##1\c@page \gls@Alphpage\else\orgAlph##1\fi}%

```

Add hook to allow for other number formats:

```
4874 \@wrglossarynumberhook
```

Prevent expansion:

```
4875 \gls@disablepagerefexpansion
```

Now store location in \@glslocref:

```
4876 \protected@xdef\@glslocref{\theglsentrycounter}%
4877 \endgroup
```

Escape any special characters

```
4878 \@gls@checkmkidxchars\@glslocref
```

Check if the hyper-location is the same as the location and set the hyper prefix.

```
4879 \expandafter\ifx\theHglentrycounter\theglsentrycounter\relax
4880 \def\@glo@counterprefix{%
4881 \else
4882 \protected@edef\@glsHlocref{\theHglentrycounter}%
4883 \@gls@checkmkidxchars\@glsHlocref
4884 \edef\@do@gls@getcounterprefix{\noexpand\@gls@getcounterprefix
4885 {\@glslocref}{\@glsHlocref}%
4886 }%
4887 \@do@gls@getcounterprefix
4888 \fi
```

De-tok label if required

```
4889 \edef\@gls@label{\glsdetoklabel{#1}}%
```

Write the information to file:

```
4890 \@do@wrglossary
4891 }
```

```
\@do@wrglossary
```

```
4892 \newcommand*\@do@wrglossary{%
```

Determine whether to use xindy or makeindex syntax

```
4893 \ifglxindy
```

Need to determine if the formatting information starts with a (or) indicating a range.

```
4894 \expandafter\@glo@check@mkidxrangechar\@glsnumberformat\@nil
4895 \def\@glo@range{%
4896 \expandafter\if\@glo@prefix(\relax
4897 \def\@glo@range{:open-range}%
4898 \else
4899 \expandafter\if\@glo@prefix)\relax
4900 \def\@glo@range{:close-range}%
4901 \fi
4902 \fi
```

Write to the glossary file using xindy syntax.

```
4903 \gls@glossary{\csname glo@\@gls@label @type\endcsname}{%
4904 (indexentry :tkey (\csname glo@\@gls@label @index\endcsname)
4905 :locref \string"\@glo@counterprefix}{\@glslocref}\string" %
4906 :attr \string"\@gls@counter\@glo@suffix\string"
4907 \@glo@range
```

```

4908 )
4909 }%
4910 \else

Convert the format information into the format required for makeindex
4911 \set@glo@numformat{\@glo@numfmt}{\@gls@counter}{\@glsnumberformat}%
4912 {\@glo@counterprefix}%

Write to the glossary file using makeindex syntax.
4913 \gls@glossary{\csname glo@\@gls@label @type\endcsname}{%
4914 \string@glossaryentry{\csname glo@\@gls@label @index\endcsname
4915 \@gls@encapchar\@glo@numfmt}{\@gls@locref}}%
4916 \fi
4917 }

```

`\@do@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.)

```

4918 \newcommand*\@gls@getcounterprefix[2]{%
4919 \edef\@gls@thisloc{#1}\edef\@gls@thisHloc{#2}%
4920 \ifx\@gls@thisloc\@gls@thisHloc
4921 \def\@glo@counterprefix{}%
4922 \else
4923 \def\@gls@get@counterprefix##1.#1##2\end@getprefix{%
4924 \def\@glo@tmp{##2}%
4925 \ifx\@glo@tmp\@empty
4926 \def\@glo@counterprefix{}%
4927 \else
4928 \def\@glo@counterprefix{##1}%
4929 \fi
4930 }%
4931 \@gls@get@counterprefix#2.#1\end@getprefix

```

Warn if no prefix can be formed.

```

4932 \ifx\@glo@counterprefix\@empty
4933 \GlossariesWarning{Hyper target '#2' can't be formed by
4934 prefixing~Jlocation '#1'. You need to modify the
4935 definition of \string\theH\@gls@counter~Jotherwise you
4936 will get the warning: "'name{\@gls@counter.#1}' has been~J
4937 referenced but does not exist"%
4938 \fi
4939 \fi
4940 }

```

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 `[\langle tag \rangle]{\langle list \rangle}`, where `\langle tag \rangle` is a tag such

as “see” and *<list>* is a list of labels.

```
4941 \newcommand{\do@seeglossary}[2]{%
4942 \def\@gls@xref{#2}%
4943 \@onelevel@sanitize\@gls@xref
4944 \@gls@checkmkidxchars\@gls@xref
4945 \ifglsxindy
4946   \gls@glossary{\csname glo@#1@type\endcsname}{%
4947     (indexentry
4948       :tkey (\csname glo@#1@index\endcsname)
4949       :xref (\string"\@gls@xref\string")
4950       :attr \string"see\string"
4951     )
4952   }%
4953 \else
4954   \gls@glossary{\csname glo@#1@type\endcsname}{%
4955     \string\glossaryentry{\csname glo@#1@index\endcsname
4956     \@gls@encapchar glsseeformat\@gls@xref}{Z}}%
4957 \fi
4958 }
```

`\@gls@fixbraces` If no optional argument is specified, list needs to be enclosed in a set of braces.

```
4959 \def\@gls@fixbraces#1#2#3\@nil{%
4960   \ifx#2[\relax
4961     \@gls@fixbraces#1#2#3\@end@fixbraces
4962   \else
4963     \def#1{{#2#3}}%
4964   \fi
4965 }
```

`\@@gls@fixbraces`

```
4966 \def\@@gls@fixbraces#1[#2]#3\@end@fixbraces{%
4967   \def#1{{#2}{#3}}%
4968 }
```

`\glssee` `\glssee{<label>}{<cross-ref list>}`

```
4969 \DeclareRobustCommand*\glssee}[3][\seename]{%
4970   \do@seeglossary{#2}{[#1]{#3}}
4971 \newcommand*\@glssee}[3][\seename]{%
4972   \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.

```
4973 \DeclareRobustCommand*\glsseeformat}[3][\seename]{%
4974   \emph{#1} \glsseelist{#2}}
```

`\glsseelist` `\glsseelist{<list>}` formats list of entry labels.

```
4975 \DeclareRobustCommand*\glsseelist}[1]{%
```

If there is only one item in the list, set the last separator to do nothing.

```
4976 \let\@gls@dolast\relax
```

Don't display separator on the first iteration of the loop

```
4977 \let\@gls@donext\relax
```

Iterate through the labels

```
4978 \@for\@gls@thislabel:=#1\do{%
```

Check if on last iteration of loop

```
4979 \ifx\@xfor@nextelement\@nnil
```

```
4980 \@gls@dolast
```

```
4981 \else
```

```
4982 \@gls@donext
```

```
4983 \fi
```

Display the entry for this label. (Expanding label as it's a temporary control sequence that's used elsewhere.)

```
4984 \expandafter\glsseeitem\expandafter{\@gls@thislabel}%
```

Update separators

```
4985 \let\@gls@dolast\glsseeelastsep
```

```
4986 \let\@gls@donext\glsseesep
```

```
4987 }%
```

```
4988 }
```

`\glsseeelastsep` Separator to use between penultimate and ultimate entries in a cross-referencing list.

```
4989 \newcommand*\glsseeelastsep{\space\andname\space}
```

`\glsseesep` Separator to use between entries in a cross-referencing list.

```
4990 \newcommand*\glsseesep{, }
```

`\glsseeitem` `\glsseeitem{<label>}` formats individual entry in a cross-referencing list.

```
4991 \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.)

```
4992 \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.

```
4993 \newcommand*{\gls@save@numberlist}[1]{%
4994   \ifglssavenumberlist
4995     \toks@{#1}%
4996     \edef\@do@writeaux@info{%
4997       \noexpand\csgdef{glo@glscurrententrylabel @numberlist}{\the\toks@}%
4998     }%
4999     \@onelevel@sanitize\@do@writeaux@info
5000     \protected@write\@auxout-{}{\@do@writeaux@info}%
5001   \fi
5002 }
```

`\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.)

```
5003 \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.

```
5004 \ifcsundef{printglossary}{}%
5005 {%
```

If `\printglossary` is already defined, issue a warning and undefine it.

```
5006   \@gls@warnonglossdefined
5007   \undef\printglossary
5008 }
```

`\printglossary` has an optional argument. The default value is to set the glossary type to the main glossary.

```
5009 \newcommand*{\printglossary}[1][type=\glsdefaulttype]{%
5010   \@printglossary{#1}{\@print@glossary}%
5011 }
```

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`

```
5012 \newcommand*{\printglossaries}{%
5013   \forallglossaries{\@glo@type}{\printglossary[type=\@glo@type]}%
5014 }
```

`\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.

```

5015 \newcommand*{\printnoidxglossary}[1] [type=\glsdefaulttype]{%
5016   \@printglossary{#1}{\@printnoidxglossary}%
5017 }

```

`\printnoidxglossaries` Analogous to `\printglossaries`

```

5018 \newcommand*{\printnoidxglossaries}{%
5019   \forallglossaries{\@glo@type}{\printnoidxglossary[type=\@glo@type]}%
5020 }

```

`\@printgloss@setsort` Initialise to do nothing.

```

5021 \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.

```

5022 \newcommand{\@printglossary}[2]{%

```

Set up defaults.

```

5023   \def\@glo@type{\glsdefaulttype}%
5024   \def\glossarytitle{\csname @glo@type\@glo@type @title\endcsname}%

```

```

5025   \def\glossarytoctitle{\glossarytitle}%
5026   \let\org@glossarytitle\glossarytitle
5027   \def\@glossarystyle{}%
5028   \def\gls@dotoc@title{\glssettoc@title{\@glo@type}}%

```

Store current value of `\glossaryentrynumbers`. (This may be changed via the optional argument)

```

5029   \let\org@glossaryentrynumbers\glossaryentrynumbers

```

Localise the effects of the optional argument

```

5030   \bgroup

```

Activate or deactivate sort key:

```

5031     \@printgloss@setsort

```

Determine settings specified in the optional argument.

```

5032     \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)

```

5033     \ifx\glossarytitle\org@glossarytitle
5034     \else
5035       \expandafter\let\csname @glo@type\@glo@type @title\endcsname
5036         \glossarytitle
5037     \fi

```

Allow a high-level user command to indicate the current glossary

```

5038     \let\currentglossary\@glo@type

```

Enable individual number lists to be suppressed.

```
5039 \let\org@glossaryentrynumbers\glossaryentrynumbers
5040 \let\glsnonextpages\@glsnonextpages
```

Enable individual number list to be activated:

```
5041 \let\glsnextpages\@glsnextpages
```

Enable suppression of description terminators.

```
5042 \let\nopostdesc\@nopostdesc
```

Set up the entry for the TOC

```
5043 \gls@dotoc@title
```

Set the glossary style

```
5044 \@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):

```
5045 \let\gls@org@glossaryentryfield\glossentry
5046 \let\gls@org@glossarysubentryfield\subglossentry
5047 \renewcommand{\glossentry}[1]{%
5048   \xdef\glscurrententrylabel{\glsdetoklabel{##1}}%
5049   \gls@org@glossaryentryfield{##1}%
5050 }%
5051 \renewcommand{\subglossentry}[2]{%
5052   \xdef\glscurrententrylabel{\glsdetoklabel{##2}}%
5053   \gls@org@glossarysubentryfield{##1}{##2}%
5054 }%
```

Now do the handler macro that deals with the actual glossary:

```
5055 #2%
```

End the current scope

```
5056 \egroup
```

Reset `\glossaryentrynumbers`

```
5057 \global\let\glossaryentrynumbers\@org@glossaryentrynumbers
```

Suppress warning about no `\printglossary`

```
5058 \global\let\warn@noprintglossary\relax
5059 }
```

`\@print@glossary` Internal workings of `\printglossary` dealing with reading the external file.

```
5060 \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.)

```
5061 \makeatletter
```

Input the glossary file, if it exists.

```
5062 \@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.

```
5063 \IfFileExists{\jobname.\csname @glo@type @in\endcsname}%
5064 {}%
5065 {\null}%
```

If `xindy` is being used, need to write the language dependent information to the `.aux` file for `makeglossaries`.

```
5066 \ifglxindy
5067 \ifcsundef{@xdy@\@glo@type @language}%
5068 {%
5069 \edef\@do@auxoutstuff{%
5070 \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.

```
5071 \noexpand\immediate\noexpand\write\@auxout{%
5072 \string\providecommand\string\@xdylanguage[2]{}%
5073 \noexpand\immediate\noexpand\write\@auxout{%
5074 \string\@xdylanguage{\@glo@type}{\@xdy@main@language}}%
5075 }%
5076 }%
5077 }%
5078 {%
5079 \edef\@do@auxoutstuff{%
5080 \noexpand\AtEndDocument{%
5081 \noexpand\immediate\noexpand\write\@auxout{%
5082 \string\providecommand\string\@xdylanguage[2]{}%
5083 \noexpand\immediate\noexpand\write\@auxout{%
5084 \string\@xdylanguage{\@glo@type}{\csname @xdy@\@glo@type
5085 @language\endcsname}}%
5086 }%
5087 }%
5088 }%
5089 \@do@auxoutstuff
5090 \edef\@do@auxoutstuff{%
5091 \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.

```
5092 \noexpand\immediate\noexpand\write\@auxout{%
5093 \string\providecommand\string\@gls@codepage[2]{}%
5094 \noexpand\immediate\noexpand\write\@auxout{%
5095 \string\@gls@codepage{\@glo@type}{\@gls@codepage}}%
5096 }%
5097 }%
```

```

5098   \@do@auxoutstuff
5099   \fi

  Activate warning if \makeglossaries hasn't been used.
5100   \renewcommand*{\@warn@nomakeglossaries}{%
5101     \GlossariesWarningNoLine{\string\makeglossaries\space
5102     hasn't been used,^^Jthe glossaries will not be updated}%
5103   }%
5104 }

```

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

```

5105 \newcommand*{\@glo@sortentries}[2]{%
5106   \def\@glo@sortinglist{}%
5107   \def\@glo@sortinghandler{#1}%
5108   \edef\@glo@type{#2}%
5109   \forlistcsloop{\@glo@do@sortentries}{@glsref@#2}%
5110   \csdef{@glsref@#2}{}%
5111   \@for\@this@label:=\@glo@sortinglist\do{%

```

Has this entry already been added?

```

5112   \xifinlistcs{\@this@label}{@glsref@#2}%
5113   {}%
5114   {%
5115     \listcsxadd{@glsref@#2}{\@this@label}%
5116   }%
5117   \ifcsdef{@glo@sortingchildren@\@this@label}%
5118   {%
5119     \@glo@addchildren{#2}{\@this@label}%
5120   }%
5121   {}%
5122 }%
5123 }

```

\@glo@addchildren \@glo@addchildren{<type>}{<parent>}

```

5124 \newcommand*{\@glo@addchildren}[2]{%
  Scope to allow nesting.
5125   \bgroup
5126   \letcs{\@glo@childlist}{@glo@sortingchildren@#2}%

```

```

5127 \@for\@this@childlabel:=\@glo@childlist\do
5128 {%

```

Check this label hasn't already been added.

```

5129 \xifinlistcs{\@this@childlabel}{@glsref@#1}%
5130 }%
5131 {%
5132 \listcsxadd{@glsref@#1}{\@this@childlabel}%
5133 }%

```

Does this child have children?

```

5134 \ifcsdef{@glo@sortingchildren@\@this@childlabel}%
5135 {%
5136 \@glo@addchildren{#1}{\@this@childlabel}%
5137 }%
5138 {%
5139 }%
5140 }%
5141 \egroup
5142 }

```

@glo@do@sortentries

```

5143 \newcommand*{\@glo@do@sortentries}[1]{%
5144 \ifglshasparent{#1}%
5145 {%

```

This entry has a parent, so add it to the child list

```

5146 \edef\@glo@parent{\csuse{glo@\glsdetoklabel{#1}@parent}}%
5147 \ifcsundef{@glo@sortingchildren@\@glo@parent}%
5148 {%
5149 \csdef{@glo@sortingchildren@\@glo@parent}{}%
5150 }%
5151 }%
5152 \expandafter\@glo@sortedinsert
5153 \csname @glo@sortingchildren@\@glo@parent\endcsname{#1}%

```

Has the parent been added?

```

5154 \xifinlistcs{\@glo@parent}{@glsref@\@glo@type}%
5155 {%

```

Yes, it has so do nothing.

```

5156 }%
5157 {%

```

No, it hasn't so add it now.

```

5158 \expandafter\@glo@do@sortentries\expandafter{\@glo@parent}%
5159 }%
5160 }%
5161 {%
5162 \@glo@sortedinsert{\@glo@sortinglist}{#1}%
5163 }%
5164 }

```

```
\@glo@sortedinsert \@glo@sortedinsert{<list>}{<entry label>}
```

Insert into list.

```
5165 \newcommand*{\@glo@sortedinsert}[2]{%
5166   \dtl@insertinto{#2}{#1}{\@glo@sortinghandler}%
5167 }%
```

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`

```
5168 \newcommand*{\@glo@sorthandler@word}[2]{%
5169   \letcs\@gls@sort@A{glo@\glsdetoklabel{#1}@sort}%
5170   \letcs\@gls@sort@B{glo@\glsdetoklabel{#2}@sort}%
5171   \edef\glo@do@compare{%
5172     \noexpand\dtlwordindexcompare{\noexpand\dtl@sortresult}%
5173     {\expandonce\@gls@sort@B}%
5174     {\expandonce\@gls@sort@A}%
5175   }%
5176   \glo@do@compare
5177 }
```

`\@glo@sorthandler@letter`

```
5178 \newcommand*{\@glo@sorthandler@letter}[2]{%
5179   \letcs\@gls@sort@A{glo@\glsdetoklabel{#1}@sort}%
5180   \letcs\@gls@sort@B{glo@\glsdetoklabel{#2}@sort}%
5181   \edef\glo@do@compare{%
5182     \noexpand\dtlletterindexcompare{\noexpand\dtl@sortresult}%
5183     {\expandonce\@gls@sort@B}%
5184     {\expandonce\@gls@sort@A}%
5185   }%
5186   \glo@do@compare
5187 }
```

`\@glo@sorthandler@case` Case-sensitive sort.

```
5188 \newcommand*{\@glo@sorthandler@case}[2]{%
5189   \letcs\@gls@sort@A{glo@\glsdetoklabel{#1}@sort}%
5190   \letcs\@gls@sort@B{glo@\glsdetoklabel{#2}@sort}%
5191   \edef\glo@do@compare{%
5192     \noexpand\dtlcompare{\noexpand\dtl@sortresult}%
5193     {\expandonce\@gls@sort@B}%
5194     {\expandonce\@gls@sort@A}%
5195   }%
5196   \glo@do@compare
5197 }
```

`\@glo@sorthandler@nocase` Case-insensitive sort.

```

5198 \newcommand*{\@glo@sorthandler@nocase}[2]{%
5199   \letcs\@gls@sort@A{glo\glsdetoklabel{#1}@sort}%
5200   \letcs\@gls@sort@B{glo\glsdetoklabel{#2}@sort}%
5201   \edef\glo@do@compare{%
5202     \noexpand\dtlicompare{\noexpand\dtl@sortresult}%
5203     {\expandonce\@gls@sort@B}%
5204     {\expandonce\@gls@sort@A}%
5205   }%
5206   \glo@do@compare
5207 }

```

@glo@sortmacro@word Sort macro for ‘word’

```

5208 \newcommand*{\@glo@sortmacro@word}[1]{%
5209   \ifdefstring{\@glo@default@sorttype}{standard}%
5210   {%
5211     \@glo@sortentries{\@glo@sorthandler@word}{#1}%
5212   }%
5213   {%
5214     \PackageError{glossaries}{Conflicting sort options:^^J
5215       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5216       \string\printnoidxglossary[sort=word]}{}%
5217   }%
5218 }

```

@glo@sortmacro@letter Sort macro for ‘letter’

```

5219 \newcommand*{\@glo@sortmacro@letter}[1]{%
5220   \ifdefstring{\@glo@default@sorttype}{standard}%
5221   {%
5222     \@glo@sortentries{\@glo@sorthandler@letter}{#1}%
5223   }%
5224   {%
5225     \PackageError{glossaries}{Conflicting sort options:^^J
5226       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5227       \string\printnoidxglossary[sort=letter]}{}%
5228   }%
5229 }

```

@glo@sortmacro@standard Sort macro for ‘standard’. (Use either ‘word’ or ‘letter’ order.)

```

5230 \newcommand*{\@glo@sortmacro@standard}[1]{%
5231   \ifdefstring{\@glo@default@sorttype}{standard}%
5232   {%
5233     \ifcsdef{\@glo@sorthandler@\glsorder}%
5234     {%
5235       \@glo@sortentries{\csuse{\@glo@sorthandler@\glsorder}}{#1}%
5236     }%
5237     {%
5238       \PackageError{glossaries}{Unknown sort handler ‘\glsorder’}{}%
5239     }%
5240   }%

```

```

5241  {%
5242    \PackageError{glossaries}{Conflicting sort options:^^J
5243      \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5244      \string\printnoidxglossary[sort=standard]}{}%
5245  }%
5246 }

```

`\@glo@sortmacro@case` Sort macro for ‘case’

```

5247 \newcommand*{\@glo@sortmacro@case}[1]{%
5248   \ifdefstring{\@glo@default@sorttype}{standard}%
5249   {%
5250     \@glo@sortentries{\@glo@sorthandler@case}{#1}%
5251   }%
5252   {%
5253     \PackageError{glossaries}{Conflicting sort options:^^J
5254       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5255       \string\printnoidxglossary[sort=case]}{}%
5256   }%
5257 }

```

`\@glo@sortmacro@nocase` Sort macro for ‘nocase’

```

5258 \newcommand*{\@glo@sortmacro@nocase}[1]{%
5259   \ifdefstring{\@glo@default@sorttype}{standard}%
5260   {%
5261     \@glo@sortentries{\@glo@sorthandler@nocase}{#1}%
5262   }%
5263   {%
5264     \PackageError{glossaries}{Conflicting sort options:^^J
5265       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5266       \string\printnoidxglossary[sort=nocase]}{}%
5267   }%
5268 }

```

`\@glo@sortmacro@def` Sort macro for ‘def’. The order of definition is given in `\glo@list@<type>`.

```

5269 \newcommand*{\@glo@sortmacro@def}[1]{%
5270   \def\@glo@sortinglist{}%
5271   \forl@sentries[#1]{\@gls@thislabel}%
5272   {%
5273     \xifinlistcs{\@gls@thislabel}{\@glsref@#1}%
5274     {%
5275       \listead{\@glo@sortinglist}{\@gls@thislabel}%
5276     }%
5277   }%

```

Hasn't been referenced.

```

5278   }%
5279 }%
5280 \cslet{\@glsref@#1}{\@glo@sortinglist}%
5281 }

```

`\@glo@sortmacro@def@do` This won't include parent entries that haven't been referenced.

```
5282 \newcommand*{\@glo@sortmacro@def@do}[1]{%
5283   \ifinlistcs{#1}{\@glsref@\@glo@type}%
5284   }%
5285   {%
5286     \listcsadd{\@glsref@\@glo@type}{#1}%
5287   }%
5288   \ifcsdef{\@glo@sortingchildren@#1}%
5289   {%
5290     \@glo@addchildren{\@glo@type}{#1}%
5291   }%
5292   }%
5293 }
```

`\@glo@sortmacro@use` Sort macro for 'use'. (No sorting is required, as the entries are already in order of use, so do nothing.)

```
5294 \newcommand*{\@glo@sortmacro@use}[1] {}
```

`\print@noidx@glossary` Glossary handler for `\printnoidxglossary` which doesn't use an indexing application. Since `\printnoidxglossary` may occur at the start of the document, we can't just check if an entry has been used. Instead, the first pass needs to write information to the aux file every time an entry is referenced. This needs to be read in on the second run and stored in a list corresponding to the appropriate glossary.

```
5295 \newcommand*{\@print@noidx@glossary}{%
5296   \ifcsdef{\@glsref@\@glo@type}%
5297   {%
```

Sort the entries:

```
5298   \ifcsdef{\@glo@sortmacro@\@glo@sorttype}%
5299   {%
5300     \csuse{\@glo@sortmacro@\@glo@sorttype}{\@glo@type}%
5301   }%
5302   {%
5303     \PackageError{glossaries}{Unknown sort handler '\@glo@sorttype'}{}%
5304   }%
```

Do the glossary heading and preamble

```
5305   \glossarysection[\glossarytoctitle]{\glossarytitle}%
5306   \glossarypreamble
5307   \begin{theglossary}%
5308   \glossaryheader
5309   \glsresetentrylist
5310   \def\@gls@currentlettergroup{}
```

Iterate through the entries.

```
5311   \forlistcsloop{\@gls@noidx@do}{\@glsref@\@glo@type}%
```

Finally end the glossary and do the postamble:

```
5312   \end{theglossary}%
```

```

5313   \glossarypostamble
5314 }%
5315 {%
5316   \@gls@noref@warn{\@glo@type}%
5317 }%
5318 }

```

`\glo@grabfirst`

```

5319 \def\glo@grabfirst#1#2\@nil{%
5320   \def\@gls@firsttok{#1}%
5321   \ifdefempty\@gls@firsttok
5322   {%
5323     \def\@glo@thislettergrp{0}%
5324   }%
5325   {%

```

Sanitize it:

```

5326     \@onelevel@sanitize\@gls@firsttok

```

Fetch the first letter:

```

5327     \expandafter\@glo@grabfirst\@gls@firsttok{}{}\@nil
5328   }%
5329 }

```

`\@glo@grabfirst`

```

5330 \def\@glo@grabfirst#1#2\@nil{%
5331   \ifdefempty\@glo@thislettergrp
5332   {%
5333     \def\@glo@thislettergrp{glssymbols}%
5334   }%
5335   {%
5336     \count@=\ucode'#1\relax
5337     \ifnum\count@=0\relax
5338     \def\@glo@thislettergrp{glssymbols}%
5339   \else
5340     \ifdefstring\@glo@sorttype{case}%
5341     {%
5342       \count@='#1\relax
5343     }%
5344     {%
5345     }%
5346     \edef\@glo@thislettergrp{\the\count@}%
5347   \fi
5348 }%
5349 }

```

`\@gls@noidx@do` Handler for list iteration used by `\@print@noidx@glossary`. The argument is the entry label. This only allows one sublevel.

```

5350 \newcommand{\@gls@noidx@do}[1]{%

```


Get this entry's location list

```
5351 \global\letcs{\@gls@loclist}{glo@glsdetoklabel{#1}@loclist}%
```

Does this entry have a parent?

```
5352 \ifglshasparent{#1}%
```

```
5353 {%
```

Has a parent.

```
5354 \gls@level=\csuse{glo@glsdetoklabel{#1}@level}\relax
```

```
5355 \ifdefvoid{\@gls@loclist}
```

```
5356 {%
```

```
5357 \subglossentry{\gls@level}{#1}{}%
```

```
5358 }%
```

```
5359 {%
```

```
5360 \subglossentry{\gls@level}{#1}%
```

```
5361 {%
```

```
5362 \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
```

```
5363 }%
```

```
5364 }%
```

```
5365 }%
```

```
5366 {%
```

Doesn't have a parent Get this entry's sort key

```
5367 \letcs{\@gls@sort}{glo@glsdetoklabel{#1}@sort}%
```

Fetch the first letter:

```
5368 \expandafter\glo@grabfirst\@gls@sort{}{}\@nil
```

```
5369 \ifdefequal{\@glo@thislettergrp}{\@gls@currentlettergroup}%
```

```
5370 {}%
```

```
5371 {%
```

Do the group header:

```
5372 \ifdefempty{\@gls@currentlettergroup}{\@gls@groupskip}%
```

```
5373 \gls@groupheading{\@glo@thislettergrp}%
```

```
5374 }%
```

```
5375 \let\@gls@currentlettergroup\@glo@thislettergrp
```

Do this entry:

```
5376 \ifdefvoid{\@gls@loclist}
```

```
5377 {%
```

```
5378 \glossentry{#1}{}%
```

```
5379 }%
```

```
5380 {%
```

```
5381 \glossentry{#1}%
```

```
5382 {%
```

```
5383 \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
```

```
5384 }%
```

```
5385 }%
```

```
5386 }%
```

```
5387 }
```

`\glsnoidxloclist` `\glsnoidxloclist{<list cs>}`

Display location list.

```
5388 \newcommand*\glsnoidxloclist[1]{%
5389   \def\@gls@noidxloclist@sep{}%
5390   \def\@gls@noidxloclist@prev{}%
5391   \forlistloop{\glsnoidxloclisthandler}{#1}%
5392 }
```

`noidxloclisthandler` Handler for location list iterator.

```
5393 \newcommand*\glsnoidxloclisthandler[1]{%
5394   \ifdefstring{\@gls@noidxloclist@prev}{#1}%
5395   {%
```

Same as previous location so skip.

```
5396   }%
5397   {%
5398     \@gls@noidxloclist@sep
5399     #1%
5400     \def\@gls@noidxloclist@sep{\delimN}%
5401     \def\@gls@noidxloclist@prev{#1}%
5402   }%
5403 }
```

`splayloclisthandler` Handler for location list iterator when used with `\glsdisplaynumberlist`.

```
5404 \newcommand*\glsnoidxdisplayloclisthandler[1]{%
5405   \ifdefstring{\@gls@noidxloclist@prev}{#1}%
5406   {%
```

Same as previous location so skip.

```
5407   }%
5408   {%
5409     \@gls@noidxloclist@sep
5410     \@gls@noidxloclist@prev
5411     \def\@gls@noidxloclist@prev{#1}%
5412   }%
5413 }
```

`\glsnoidxdisplayloc` `\glsnoidxdisplayloc{<prefix>}{<counter>}{<format>}{<location>}`

Display a location in the location list.

```
5414 \newcommand*\glsnoidxdisplayloc[4]{%
5415   \setentrycounter[#1]{#2}%
5416   \csuse{#3}{#4}%
5417 }
```

`\@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.

```
5418 \newcommand*{\@gls@reference}[3]{%
  Add to label list
5419   \glsdoifexistsorwarn{#2}%
5420   {%
5421     \ifcsundef{@glsref@#1}{\csgdef{@glsref@#1}{}}{}%
5422     \ifinlistcs{#2}{@glsref@#1}%
5423     {}%
5424     {\listcsgadd{@glsref@#1}{#2}}%
  Add to location list
5425   \ifcsundef{glo@glsdetoklabel{#2}@loclist}%
5426   {\csgdef{glo@glsdetoklabel{#2}@loclist}{}}%
5427   {}%
5428   \listcsgadd{glo@glsdetoklabel{#2}@loclist}{#3}%
5429   }%
5430 }
```

The keys that can be used in the optional argument to `\printglossary` or `\printnoidxglossary` are as follows: The `type` key sets the glossary type.

```
5431 \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`.

```
5432 \define@key{printgloss}{title}{%
5433   \def\glossarytitle{#1}%
5434   \let\gls@dotoc@title\relax
5435 }
```

The `toctitle` sets the text used for the relevant entry in the table of contents.

```
5436 \define@key{printgloss}{toctitle}{%
5437   \def\glossarytoctitle{#1}%
5438   \let\gls@dotoc@title\relax
5439 }
```

The `style` key sets the glossary style (but only for the given glossary).

```
5440 \define@key{printgloss}{style}{%
5441   \ifcsundef{@glsstyle@#1}%
5442   {%
5443     \PackageError{glossaries}%
5444     {Glossary style ‘#1’ undefined}{}%
5445   }%
5446   {%
5447     \def\@glossarystyle{\setglossentrycompatibility
5448       \csname @glsstyle@#1\endcsname}%
5449   }%
5450 }
```

The `numberedsection` key determines if this glossary should be in a numbered section.

```

5451 \define@choicekey{printgloss}{numberedsection}[\val\nr]{%
5452 false,nolabel,autolabel,nameref}[nolabel]{%
5453   \ifcase\nr\relax
5454     \renewcommand*{\@@glossarysecstar}{*}%
5455     \renewcommand*{\@@glossaryseclabel}{}%
5456   \or
5457     \renewcommand*{\@@glossarysecstar}{}%
5458     \renewcommand*{\@@glossaryseclabel}{}%
5459   \or
5460     \renewcommand*{\@@glossarysecstar}{}%
5461     \renewcommand*{\@@glossaryseclabel}{\label{\glsautoprefix\@glo@type}}%
5462   \or
5463     \renewcommand*{\@@glossarysecstar}{*}%
5464     \renewcommand*{\@@glossaryseclabel}{%
5465       \protected@edef\@currentlabelname{\glossarytoctitle}%
5466       \label{\glsautoprefix\@glo@type}}%
5467   \fi
5468 }

```

The `nogroupskip` key determines whether or not there should be a vertical gap between glossary groups.

```

5469 \define@choicekey{printgloss}{nogroupskip}{true,false}[true]{%
5470   \csuse{glsnogroupskip#1}%
5471 }

```

The `nopostdot` key has the same effect as the package option of the same name.

```

5472 \define@choicekey{printgloss}{nopostdot}{true,false}[true]{%
5473   \csuse{glsnopostdot#1}%
5474 }

```

The `entrycounter` key is the same as the package option but localised to the current glossary.

```

5475 \define@choicekey{printgloss}{entrycounter}{true,false}[true]{%
5476   \csuse{glsentrycounter#1}%
5477   \ifglsentrycounter
5478     \ifx\@gls@counterwithin\@empty
5479       \newcounter{glossaryentry}%
5480     \else
5481       \newcounter{glossaryentry}[\@gls@counterwithin]%
5482     \fi
5483     \def\theHglossaryentry{\currentglossary.\theglossaryentry}%
5484     \renewcommand*{\glsresetentrycounter}{%
5485       \setcounter{glossaryentry}{0}%
5486     }%
5487     \renewcommand*{\glsstepentry}[1]{%
5488       \refstepcounter{glossaryentry}%
5489       \label{glsentry-\glsdetoklabel{##1}}%

```

```

5490 }%
5491 \renewcommand*{\glsentrycounterlabel}{\theglossaryentry.\space}%
5492 \renewcommand*{\glsentryitem}[1]{%
5493   \glsstepentry{##1}\glsentrycounterlabel
5494 }%
5495 \else
5496   \renewcommand*{\glsresetentrycounter}{}%
5497   \renewcommand*{\glsstepentry}[1]{}%
5498   \renewcommand*{\glsentrycounterlabel}{}%
5499   \renewcommand*{\glsentryitem}[1]{\glsresetsubentrycounter}
5500 \fi
5501 }

```

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.

```

5502 \define@choicekey{printgloss}{subentrycounter}{true,false}[true]{}%
5503   \csuse{glssubentrycounter#1}%
5504   \ifglssubentrycounter
5505     \ifundef\c@glossarysubentry
5506     {%
5507       \ifglsentrycounter
5508         \newcounter{glossarysubentry}[glossaryentry]%
5509       \else
5510         \newcounter{glossarysubentry}
5511       \fi
5512     }{}%
5513   \renewcommand*{\glsstepsubentry}[1]{}%
5514   \edef\currentglssubentry{\glsdetoklabel{##1}}%
5515   \refstepcounter{glossarysubentry}%
5516   \label{glsentry-\currentglssubentry}%
5517 }%
5518 \renewcommand*{\glsresetsubentrycounter}{}%
5519 \setcounter{glossarysubentry}{0}%
5520 }%
5521 \renewcommand*{\glssubentryitem}[1]{}%
5522   \glsstepsubentry{##1}\glssubentrycounterlabel
5523 }%
5524 \renewcommand*{\glssubentrycounterlabel}{\theglossarysubentry}\space}%
5525 \def\theHglossarysubentry{\currentglssubentry.\theglossarysubentry}
5526 \else
5527   \renewcommand*{\glssubentryitem}[1]{}%
5528   \renewcommand*{\glsstepsubentry}[1]{}%
5529   \renewcommand*{\glsresetsubentrycounter}{}%
5530   \renewcommand*{\glssubentrycounterlabel}{}%
5531 \fi
5532 }

```

The `nonumberlist` key determines if this glossary should have a number list.

```

5533 \define@boolkey{printgloss}[gls]{nonumberlist}[true]{%
5534 \ifglsnonumberlist
5535   \def\glossaryentrynumbers##1{}%
5536 \else
5537   \def\glossaryentrynumbers##1{##1}%
5538 \fi}

```

The sort key sets the glossary sort handler (`\printnoidxglossary` only).

```

5539 \define@key{printgloss}{sort}{\@glo@assign@sortkey{#1}}

```

`\@glo@no@assign@sortkey` Issue error if used with `\printglossary`

```

5540 \newcommand*\@glo@no@assign@sortkey[1]{%
5541   \PackageError{glossaries}'sort' key not permitted with
5542   \string\printglossary}%
5543   {The 'sort' key may only be used with \string\printnoidxglossary}%
5544 }

```

`\@glo@assign@sortkey` For use with `\printnoidxglossary`

```

5545 \newcommand*\@glo@assign@sortkey[1]{%
5546   \def\@glo@sorttype{#1}%
5547 }

```

`\@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.

```

5548 \newcommand*\@glsnonextpages{%
5549   \gdef\glossaryentrynumbers##1{%
5550     \glsresetentrylist
5551   }%
5552 }

```

`\@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.

```

5553 \newcommand*\@glsnextpages{%
5554   \gdef\glossaryentrynumbers##1{%
5555     ##1\glsresetentrylist}}

```

`\glsresetentrylist` Resets `\glossaryentrynumbers`

```

5556 \newcommand*\glsresetentrylist{%
5557   \global\let\glossaryentrynumbers\org@glossaryentrynumbers}

```

`\glsnonextpages` Outside of `\printglossary` this does nothing.

```

5558 \newcommand*\glsnonextpages{}

```

`\glsnextpages` Outside of `\printglossary` this does nothing.

```
5559 \newcommand*{\glsnextpages}{}
```

`glossaryentry` If the `entrycounter` package option has been used, define a counter to number each level 0 entry.

```
5560 \ifglentrycounter
5561   \ifx\@gls@counterwithin\@empty
5562     \newcounter{glossaryentry}
5563   \else
5564     \newcounter{glossaryentry}[\@gls@counterwithin]
5565   \fi
5566   \def\theHglossaryentry{\currentglossary.\theglossaryentry}
5567 \fi
```

`glossarysubentry` If the `subentrycounter` package option has been used, define a counter to number each level 1 entry.

```
5568 \ifglssubentrycounter
5569   \ifglentrycounter
5570     \newcounter{glossarysubentry}[glossaryentry]
5571   \else
5572     \newcounter{glossarysubentry}
5573   \fi
5574   \def\theHglossarysubentry{\currentglssubentry.\theglossarysubentry}
5575 \fi
```

`resetsubentrycounter` Resets the `glossarysubentry` counter.

```
5576 \ifglssubentrycounter
5577   \newcommand*{\glsresetsubentrycounter}{%
5578     \setcounter{glossarysubentry}{0}%
5579   }
5580 \else
5581   \newcommand*{\glsresetsubentrycounter}{}
5582 \fi
```

`resetentrycounter` Resets the `glossaryentry` counter.

```
5583 \ifglentrycounter
5584   \newcommand*{\glsresetentrycounter}{%
5585     \setcounter{glossaryentry}{0}%
5586   }
5587 \else
5588   \newcommand*{\glsresetentrycounter}{}
5589 \fi
```

`\glsstepentry` Advance the `glossaryentry` counter if in use. The argument is the label associated with the entry.

```
5590 \ifglentrycounter
5591   \newcommand*{\glsstepentry}[1]{%
5592     \refstepcounter{glossaryentry}%
```

```

5593   \label{glsentry-\glsdetoklabel{#1}}%
5594   }
5595 \else
5596   \newcommand*{\glsstepentry}[1]{%
5597 \fi

```

`\glsstepsubentry` Advance the glossarysubentry counter if in use. The argument is the label associated with the subentry.

```

5598 \ifglssubentrycounter
5599   \newcommand*{\glsstepsubentry}[1]{%
5600     \edef\currentglssubentry{\glsdetoklabel{#1}}%
5601     \refstepcounter{glossarysubentry}%
5602     \label{glsentry-\currentglssubentry}%
5603   }
5604 \else
5605   \newcommand*{\glsstepsubentry}[1]{%
5606 \fi

```

`\glsrefentry` Reference the entry or sub-entry counter if in use, otherwise just do `\gls`.

```

5607 \ifglentrycounter
5608   \newcommand*{\glsrefentry}[1]{\ref{glsentry-\glsdetoklabel{#1}}}
5609 \else
5610   \ifglssubentrycounter
5611     \newcommand*{\glsrefentry}[1]{\ref{glsentry-\glsdetoklabel{#1}}}
5612   \else
5613     \newcommand*{\glsrefentry}[1]{\gls{#1}}
5614   \fi
5615 \fi

```

`lentrycounterlabel` Defines how to display the glossaryentry counter.

```

5616 \ifglentrycounter
5617   \newcommand*{\glsentrycounterlabel}{\theglossaryentry.\space}
5618 \else
5619   \newcommand*{\glsentrycounterlabel}{}
5620 \fi

```

`subentrycounterlabel` Defines how to display the glossarysubentry counter.

```

5621 \ifglssubentrycounter
5622   \newcommand*{\glssubentrycounterlabel}{\theglossarysubentry)\space}
5623 \else
5624   \newcommand*{\glssubentrycounterlabel}{}
5625 \fi

```

`\glsentryitem` Step and display glossaryentry counter, if appropriate.

```

5626 \ifglentrycounter
5627   \newcommand*{\glsentryitem}[1]{%
5628     \glsstepentry{#1}\glsentrycounterlabel
5629   }

```



```

5630 \else
5631   \newcommand*{\glstentryitem}[1]{\glsresetsubentrycounter}
5632 \fi

```

`\glssubentryitem` Step and display glossarysubentry counter, if appropriate.

```

5633 \ifglssubentrycounter
5634   \newcommand*{\glssubentryitem}[1]{%
5635     \glsstepsubentry{#1}\glssubentrycounterlabel
5636   }
5637 \else
5638   \newcommand*{\glssubentryitem}[1]{%
5639 \fi

```

`theglossary` If the `theglossary` environment has already been defined, a warning will be issued. This environment should be redefined by glossary styles.

```

5640 \ifcsundef{theglossary}%
5641 {%
5642   \newenvironment{theglossary}{}{}%
5643 }%
5644 {%
5645   \@gls@warnontheglossdefined
5646   \renewenvironment{theglossary}{}{}%
5647 }

```

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`

```

5648 \newcommand*{\glossaryheader}{}

```

`\glstarget` `\glstarget{<label>}{<name>}`

Provide user interface to `\@glstarget` to make it easier to modify the glossary style in the document.

```

5649 \newcommand*{\glstarget}[2]{\@glstarget{\glslinkprefix#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>}
```

```

5650 \providecommand*\compatibleglossentry}[2]{%
5651   \toks@{#2}%
5652   \protected@edef\@do@glossentry{\noexpand\glossaryentryfield{#1}%
5653     {\noexpand\glsnamefont
5654       {\expandafter\expandonce\csname glo@#1@name\endcsname}}}%
5655     {\expandafter\expandonce\csname glo@#1@desc\endcsname}}%
5656     {\expandafter\expandonce\csname glo@#1@symbol\endcsname}}%
5657     {\the\toks@}%
5658   }%
5659   \@do@glossentry
5660 }

```

`\glossentryname`

```

5661 \newcommand*\glossentryname}[1]{%
5662   \glsdoifexistsorwarn{#1}%
5663   {%
5664     \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
5665     \expandafter\glsnamefont\expandafter{\glo@name}%
5666   }%
5667 }

```

`\Glossentryname`

```

5668 \newcommand*\Glossentryname}[1]{%
5669   \glsdoifexistsorwarn{#1}%
5670   {%
5671     \glsnamefont{\Glsentryname{#1}}%
5672   }%
5673 }

```

`\glossentrydesc`

```

5674 \newcommand*\glossentrydesc}[1]{%
5675   \glsdoifexistsorwarn{#1}%
5676   {%
5677     \glsentrydesc{#1}%
5678   }%
5679 }

```

`\Glossentrydesc`

```

5680 \newcommand*\Glossentrydesc}[1]{%
5681   \glsdoifexistsorwarn{#1}%
5682   {%
5683     \Glsentrydesc{#1}%
5684   }%
5685 }

```

`\glossentrysymbol`

```

5686 \newcommand*\glossentrysymbol}[1]{%
5687   \glsdoifexistsorwarn{#1}%
5688   {%

```

```

5689   \glsentrysymbol{#1}%
5690   }%
5691 }

```

\Glossentrysymbol

```

5692 \newcommand*{\Glossentrysymbol}[1]{%
5693   \glsdoifexistsorwarn{#1}%
5694   {%
5695     \Glsentrysymbol{#1}%
5696   }%
5697 }

```

compatiblesubglossentry

```
\subglossentry{<level>}{<label>}{<page-list>}
```

```

5698 \providecommand*{\compatiblesubglossentry}[3]{%
5699   \toks@{#3}%
5700   \protected@edef\@do@subglossentry{\noexpand\glossarysubentryfield{\number#1}%
5701     {#2}}%
5702     {\noexpand\glsnamefont
5703       {\expandafter\expandonce\csname glo@#2@name\endcsname}}%
5704     {\expandafter\expandonce\csname glo@#2@desc\endcsname}%
5705     {\expandafter\expandonce\csname glo@#2@symbol\endcsname}%
5706     {\the\toks@}%
5707   }%
5708   \@do@subglossentry
5709 }

```

sentrycompatibility

```

5710 \newcommand*{\setglossentrycompatibility}{%
5711   \let\glossentry\compatibleglossentry
5712   \let\subglossentry\compatiblesubglossentry
5713 }
5714 \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.

```

5715 \newcommand{\glossaryentryfield}[5]{%
5716   \GlossariesWarning
5717   {Deprecated use of \string\glossaryentryfield.^J
5718     I recommend you change to \string\glossentry.^J
5719     If you've just upgraded, try removing your gls auxiliary
5720     files^J and recompile}%
5721   \noindent\textbf{\glstarget{#1}{#2}} #4 #3. #5\par}

```

`\glossarysubentryfield`

```
\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.)

```
5722 \newcommand*\glossarysubentryfield[6]{%
5723   \GlossariesWarning
5724   {Deprecated use of \string\glossarysubentryfield.^^J
5725    I recommend you change to \string\subglossentry.^^J
5726    If you've just upgraded, try removing your gls auxiliary
5727    files^^J and recompile}%
5728   \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`

```
5729 \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`, `glnumbers`, A, ..., Z. Glossary styles must redefine this command. (In between groups, `\glsgroupheading` comes immediately after `\glsgroupskip`.)

`\glsgroupheading`

```
5730 \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
```

```
5731 \newcommand*{\glsgetgrouptitle}[1]{%
5732   \@gls@getgrouptitle{#1}{\@gls@grptitle}%
5733   \@gls@grptitle
5734 }
```

```
\@gls@getgrouptitle
```

Gets the group title specified by the label (first argument) and stores in the second argument, which must be a control sequence.

```
5735 \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.

```
5736 \dtl@ifsingle{#1}%
5737 {%
5738   \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
5739 }%
5740 {%
5741   \ifboolexpr{test{\ifstrequal{#1}{glsymbols}}
5742               or test{\ifstrequal{#1}{glsnumbers}}}%
5743   {%
5744     \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
5745   }%
5746   {%
5747     \def#2{#1}%
5748   }%
5749 }%
5750 }
```

```
@getothergrouptitle
```

Version for the no-indexing app option:

```
5751 \newcommand*{\@gls@noidx@getgrouptitle}[2]{%
5752   \DTLifint{#1}%
5753   {\edef#2{\char#1\relax}}%
5754   {%
5755     \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
5756   }%
5757 }
```

```
\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 `\glssetgrouptitle`, you will also need to redefine `\glsgetgrouplabel`.

```
\glsgetgrouplabel
```

```
5758 \newcommand*{\glsgetgrouplabel}[1]{%
5759 \ifthenelse{\equal{#1}{\glsymbolsgroupname}}{\glsymbols}{%
5760 \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
```

```
5761 \newcommand*{\setentrycounter}[2][ ]{%
5762 \def\@glo@counterprefix{#1}%
5763 \ifx\@glo@counterprefix\@empty
5764 \def\@glo@counterprefix{.}%
5765 \else
5766 \def\@glo@counterprefix{.#1.}%
5767 \fi
5768 \def\glsentrycounter{#2}%
5769 }
```

The current glossary style can be set using `\setglossarystyle{<style>}`.

```
\setglossarystyle
```

```
5770 \newcommand*{\setglossarystyle}[1]{%
5771 \ifcsundef{@glsstyle@#1}%
5772 {%
5773 \PackageError{glossaries}{Glossary style ‘#1’ undefined}{%
5774 }%
5775 {%
5776 \csname @glsstyle@#1\endcsname
5777 }%
5778 }
```

```
\glossarystyle
```

```
5779 \newcommand*{\glossarystyle}[1]{%
5780 \ifcsundef{@glsstyle@#1}%
5781 {%
5782 \PackageError{glossaries}{Glossary style ‘#1’ undefined}{%
5783 }%
5784 {%
5785 \GlossariesWarning
5786 {Deprecated command \string\glossarystyle.^~J
5787 I recommend you switch to \string\setglossarystyle\space unless
```

```

5788     you want to maintain backward compatibility}%
5789     \setglossentrycompatibility
5790     \csname @glsstyle@#1\endcsname

5791     \ifcsdef{@glscompstyle@#1}%
5792     {\setglossentrycompatibility\csuse{@glscompstyle@#1}}%
5793     {}%
5794 }%
5795 }

```

`\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 `\glossarypreamble` and `\glossarypostamble`, as the user should be able to switch between styles without affecting the pre- and postambles.

```

5796 \newcommand{\newglossarystyle}[2]{%
5797   \ifcsundef{@glsstyle@#1}%
5798   {%
5799     \expandafter\def\csname @glsstyle@#1\endcsname{#2}%
5800   }%
5801   {%
5802     \PackageError{glossaries}{Glossary style ‘#1’ is already defined}{}%
5803   }%
5804 }

```

`\renewglossarystyle` Code for this macro supplied by Marco Daniel.

```

5805 \newcommand{\renewglossarystyle}[2]{%
5806   \ifcsundef{@glsstyle@#1}%
5807   {%
5808     \PackageError{glossaries}{Glossary style ‘#1’ isn’t already defined}{}%
5809   }%
5810   {%
5811     \csdef{@glsstyle@#1}{#2}%
5812   }%
5813 }

```

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`

```
5814 \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`

```
5815 \ifcsundef{hyperlink}%
5816 {%
5817   \def\glsnumber#1{#1}%
5818 }%
5819 {%
5820   \def\glsnumber#1{\@glsnumber#1\nohyperpage{}}\@nil}
5821 }
```

`\@glsnumber` This code was provided by Heiko Oberdiek to allow material to be attached to the location.

```
5822 \def\@glsnumber#1\nohyperpage#2#3\@nil{%
5823   \ifx\#1\%
5824   \else
5825     \@delimR#1\delimR\delimR\%
5826   \fi
5827   \ifx\#2\%
5828   \else
5829     #2%
5830   \fi
5831   \ifx\#3\%
5832   \else
5833     \@glsnumber#3\@nil
5834   \fi
5835 }
```

`\@delimR` displays a range of numbers for the counter whose name is given by `\@gls@counter` (which must be set prior to using `\glsnumber`).

`\@delimR`

```
5836 \def\@delimR#1\delimR #2\delimR #3\%
5837 \ifx\#2\%
```



```

5838 \@delimN{#1}%
5839 \else
5840 \@gls@numberlink{#1}\delimR\@gls@numberlink{#2}%
5841 \fi}

```

\@delimN displays a list of individual numbers, instead of a range:

\@delimN

```

5842 \def\@delimN#1{\@@delimN#1\delimN \delimN\}
5843 \def\@@delimN#1\delimN #2\delimN#3\{\{
5844 \ifx\#3\}%
5845 \@gls@numberlink{#1}%
5846 \else
5847 \@gls@numberlink{#1}\delimN\@gls@numberlink{#2}%
5848 \fi
5849 }

```

The following code is modified from hyperref's \HyInd@pagelink where the name of the counter being used is given by \@gls@counter.

```

5850 \def\@gls@numberlink#1{%
5851 \begingroup
5852 \toks@={}%
5853 \@gls@removespaces#1 \@nil
5854 \endgroup}

5855 \def\@gls@removespaces#1 #2\@nil{%
5856 \toks@=\expandafter{\the\toks@#1}%
5857 \ifx\#2\}%
5858 \edef\x{\the\toks@}%
5859 \ifx\x\empty
5860 \else

5861 \hyperlink{\glsentrycounter\@glo@counterprefix\the\toks@}%
5862 {\the\toks@}%
5863 \fi
5864 \else
5865 \@gls@ReturnAfterFi{%
5866 \@gls@removespaces#2\@nil
5867 }%
5868 \fi
5869 }
5870 \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

```

5871 \newcommand*\hyperrm}[1]{\textrm{\glsnumber{#1}}}

```

`\hypersf`
5872 `\newcommand*{\hypersf}[1]{\textsf{\glshypernumber{#1}}}`

`\hypertt`
5873 `\newcommand*{\hypertt}[1]{\texttt{\glshypernumber{#1}}}`

`\hyperbf`
5874 `\newcommand*{\hyperbf}[1]{\textbf{\glshypernumber{#1}}}`

`\hypermd`
5875 `\newcommand*{\hypermd}[1]{\textmd{\glshypernumber{#1}}}`

`\hyperit`
5876 `\newcommand*{\hyperit}[1]{\textit{\glshypernumber{#1}}}`

`\hypersl`
5877 `\newcommand*{\hypersl}[1]{\textsl{\glshypernumber{#1}}}`

`\hyperup`
5878 `\newcommand*{\hyperup}[1]{\textup{\glshypernumber{#1}}}`

`\hypersc`
5879 `\newcommand*{\hypersc}[1]{\textsc{\glshypernumber{#1}}}`

`\hyperemph`
5880 `\newcommand*{\hyperemph}[1]{\emph{\glshypernumber{#1}}}`

1.17 Acronyms

`\oldacronym` `\oldacronym[⟨label⟩]{⟨abbrv⟩}{⟨long⟩}{⟨key-val list⟩}`

This emulates the way the old package defined acronyms. It is equivalent to `\newacronym[⟨key-val list⟩]{⟨label⟩}{⟨abbrv⟩}{⟨long⟩}` and it additionally defines the command `\⟨label⟩` which is equivalent to `\gls{⟨label⟩}` (thus `⟨label⟩` must only contain alphabetical characters). If `⟨label⟩` is omitted, `⟨abbrv⟩` is used. This only emulates the syntax of the old package. The way the acronyms appear in the list of acronyms is determined by the definition of `\newacronym` and the glossary style.

Note that `\⟨label⟩` can't have an optional argument if the package is loaded. If hasn't been loaded then you can do `\⟨label⟩[⟨insert⟩]` but you can't do `\⟨label⟩[⟨key-val list⟩]`. For example if you define the acronym `svm`, then you can do `\svm['s]` but you can't do `\svm[format=textbf]`. If the package is loaded, `\svm['s]` will appear as `svm ['s]` which is unlikely to be the desired

result. In this case, you will need to use `\gls` explicitly, e.g. `\gls{svm}[’s]`. Note that it is up to the user to load if desired.

```

5881 \newcommand{\oldacronym}[4] [\gls@label] {%
5882   \def\gls@label{#2}%
5883   \newacronym[#4]{#1}{#2}{#3}%
5884   \ifcsundef{xspace}%
5885     {%
5886       \expandafter\edef\csname#1\endcsname{%
5887         \noexpand\@ifstar{\noexpand\Gls{#1}}{\noexpand\gls{#1}}%
5888       }%
5889     }%
5890     {%
5891       \expandafter\edef\csname#1\endcsname{%
5892         \noexpand\@ifstar{\noexpand\Gls{#1}\noexpand\xspace}{%
5893         \noexpand\gls{#1}\noexpand\xspace}%
5894       }%
5895     }%
5896 }

```

<code>\newacronym[⟨key-val list⟩]{⟨label⟩}{⟨abbrev⟩}{⟨long⟩}</code>

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`

```

5897 \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.

```

5898 \newcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}

```

If `garamondx` has been loaded, need to use `\textulc` instead of `\textup`.

```
\glstextup
5899 \newrobustcmd*{\glstextup}[1]{\ifdef\textulc{\textulc{#1}}{\textup{#1}}}
```

The following are defined for compatibility with version 2.07 and earlier.

```
\glsshortkey
5900 \newcommand*{\glsshortkey}{short}
```

```
\glsshortpluralkey
5901 \newcommand*{\glsshortpluralkey}{shortplural}
```

```
\glslongkey
5902 \newcommand*{\glslongkey}{long}
```

```
\glslongpluralkey
5903 \newcommand*{\glslongpluralkey}{longplural}
```

```
\acrfull Full form of the acronym.
5904 \newrobustcmd*{\acrfull}{\@gls@hyp@opt\ns@acrfull}

5905 \newcommand*\ns@acrfull[2][\%
5906 \new@ifnextchar[{\@acrfull{#1}{#2}}%
5907 \@acrfull{#1}{#2}[]}%
5908 }
```

```
\@acrfull Low-level macro:
5909 \def\@acrfull#1#2[#3]{%
    Make it easier for acronym styles to change this:
5910 \acrfullfmt{#1}{#2}{#3}%
5911 }
```

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.
5912 \newcommand*{\acrfullfmt}[3]{%
5913 \acrlinkfullformat{\@acrlong}{\@acrshort}{#1}{#2}{#3}%
5914 }
```

```
\acrlinkfullformat Format for full links like \acrfull. Syntax: \acrlinkfullformat{<long
cs>}{<short cs>}{<options>}{<label>}{<insert>}
5915 \newcommand{\acrlinkfullformat}[5]{%
5916 \acrfullformat{#1{#3}{#4}[#5]}{#2{#3}{#4}[]}%
5917 }
```

```
\acrfullformat Default full form is <long> (<short>).
5918 \newcommand{\acrfullformat}[2]{#1\glsspace(#2)}
```

`\glsspace` Robust space to ensure it's written to the `.glsdefs` file.

```
5919 \newrobustcmd{\glsspace}{\space}
```

Default format for full acronym

`\Acrfull`

```
5920 \newrobustcmd*{\Acrfull}{\@gls@hyp@opt\ns@Acrfull}
```

```
5921 \newcommand*\ns@Acrfull[2][\@Acrfull{#1}{#2}]%
```

```
5922 \new@ifnextchar[{\@Acrfull{#1}{#2}}%
```

```
5923 \@Acrfull{#1}{#2}[]%
```

```
5924 }
```

Low-level macro:

```
5925 \def\@Acrfull#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
5926 \Acrfullfmt{#1}{#2}{#3}%
```

```
5927 }
```

`\Acrfullfmt` First letter upper case full format.

```
5928 \newcommand*\Acrfullfmt[3]{%
```

```
5929 \acrlinkfullformat{\@Acrlong}{\@acrshort}{#1}{#2}{#3}%
```

```
5930 }
```

`\ACRfull`

```
5931 \newrobustcmd*{\ACRfull}{\@gls@hyp@opt\ns@ACRfull}
```

```
5932 \newcommand*\ns@ACRfull[2][\@ACRfull{#1}{#2}]%
```

```
5933 \new@ifnextchar[{\@ACRfull{#1}{#2}}%
```

```
5934 \@ACRfull{#1}{#2}[]%
```

```
5935 }
```

Low-level macro:

```
5936 \def\@ACRfull#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
5937 \ACRfullfmt{#1}{#2}{#3}%
```

```
5938 }
```

`\ACRfullfmt` All upper case full format.

```
5939 \newcommand*\ACRfullfmt[3]{%
```

```
5940 \acrlinkfullformat{\@ACRlong}{\@ACRshort}{#1}{#2}{#3}%
```

```
5941 }
```

Plural:

`\acrfullpl`

```
5942 \newrobustcmd*{\acrfullpl}{\@gls@hyp@opt\ns@acrfullpl}
```

```

5943 \newcommand*\ns@acrfullpl[2] [] {%
5944   \new@ifnextchar [{\@acrfullpl{#1}{#2}}%
5945     {\@acrfullpl{#1}{#2} []}%
5946 }

```

Low-level macro:

```
5947 \def\@acrfullpl#1#2[#3] {%
```

Make it easier for acronym styles to change this:

```

5948   \acrfullplfmt{#1}{#2}{#3}%
5949 }

```

`\acrfullplfmt` No case change plural full format.

```

5950 \newcommand*\acrfullplfmt[3] {%
5951   \acrlinkfullformat{\@acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%
5952 }

```

`\Acrfullpl`

```
5953 \newrobustcmd*\Acrfullpl}{\@gls@hyp@opt\ns@Acrfullpl}
```

```

5954 \newcommand*\ns@Acrfullpl[2] [] {%
5955   \new@ifnextchar [{\@Acrfullpl{#1}{#2}}%
5956     {\@Acrfullpl{#1}{#2} []}%
5957 }

```

Low-level macro:

```
5958 \def\@Acrfullpl#1#2[#3] {%
```

Make it easier for acronym styles to change this:

```

5959   \Acrfullplfmt{#1}{#2}{#3}%
5960 }

```

`\Acrfullplfmt` First letter upper case plural full format.

```

5961 \newcommand*\Acrfullplfmt[3] {%
5962   \acrlinkfullformat{\@Acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%
5963 }

```

`\ACRfullpl`

```
5964 \newrobustcmd*\ACRfullpl}{\@gls@hyp@opt\ns@ACRfullpl}
```

```

5965 \newcommand*\ns@ACRfullpl[2] [] {%
5966   \new@ifnextchar [{\@ACRfullpl{#1}{#2}}%
5967     {\@ACRfullpl{#1}{#2} []}%
5968 }

```

Low-level macro:

```
5969 \def\@ACRfullpl#1#2[#3] {%
```

Make it easier for acronym styles to change this:

```

5970   \ACRfullplfmt{#1}{#2}{#3}%
5971 }

```

`\ACRfullplfmt` All upper case plural full format.
5972 `\newcommand*\ACRfullplfmt}[3]{%`
5973 `\acrlinkfullformat{\@ACRlongpl}{\@ACRshortpl}{#1}{#2}{#3}%`
5974 `}`

1.18 Predefined acronym styles

`\acronymfont` This is only used with the additional acronym styles:
5975 `\newcommand{\acronymfont}[1]{#1}`

`\firstacronymfont` This is only used with the additional acronym styles:
5976 `\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.
5977 `\newcommand*\acrnameformat}[2]{\acronymfont{#1}}`

Define some tokens used by `\newacronym`:

`\glskeylisttok`
5978 `\newtoks\glskeylisttok`

`\glslabeltok`
5979 `\newtoks\glslabeltok`

`\glsshorttok`
5980 `\newtoks\glsshorttok`

`\glslongtok`
5981 `\newtoks\glslongtok`

`\newacronymhook` Provide a hook for `\newacronym`:
5982 `\newcommand*\newacronymhook{}`

`\SetGenericNewAcronym` New improved version of setting the acronym style.
5983 `\newcommand*\SetGenericNewAcronym}{%`

Change the behaviour of `\Glsentryname` to workaround expansion issues that cause a problem for `\makefirstuc`

5984 `\let\@Gls@entryname\@Gls@acentryname`

Change the way acronyms are defined:

5985 `\renewcommand{\newacronym}[4][]{%`
5986 `\ifdefempty{\@glsacronymlists}%`
5987 `{%`
5988 `\def\@glo@type{\acronymtype}%`
5989 `\setkeys{glossentry}{##1}%`
5990 `\DeclareAcronymList{\@glo@type}%`

```

5991 }%
5992 {}%
5993 \glskeylisttok{##1}%
5994 \glslabeltok{##2}%
5995 \glsshorttok{##3}%
5996 \glslongtok{##4}%
5997 \newacronymhook
5998 \protected@edef\@do@newglossaryentry{%
5999   \noexpand\newglossaryentry{\the\glslabeltok}%
6000   {%
6001     type=\acronymtype,%
6002     name={\expandonce{\acronymentry{##2}}},%
6003     sort={\acronymstok{\the\glsshorttok}{\the\glslongtok}},%
6004     text={\the\glsshorttok},%
6005     short={\the\glsshorttok},%
6006     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6007     long={\the\glslongtok},%
6008     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6009     \GenericAcronymFields,%
6010     \the\glskeylisttok
6011   }%
6012 }%
6013 \@do@newglossaryentry
6014 }%

```

Make sure that `\acrfull` etc reflects the new style:

```

6015 \renewcommand*\acrfullfmt}[3]{%
6016   \glslink[##1]{##2}{\genacrfullformat{##2}{##3}}%
6017 \renewcommand*\Acrfullfmt}[3]{%
6018   \glslink[##1]{##2}{\Genacrfullformat{##2}{##3}}%
6019 \renewcommand*\ACRfullfmt}[3]{%
6020   \glslink[##1]{##2}{%
6021     \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}%
6022 \renewcommand*\acrfullplfmt}[3]{%
6023   \glslink[##1]{##2}{\genplacrfullformat{##2}{##3}}%
6024 \renewcommand*\Acrfullplfmt}[3]{%
6025   \glslink[##1]{##2}{\Genplacrfullformat{##2}{##3}}%
6026 \renewcommand*\ACRfullplfmt}[3]{%
6027   \glslink[##1]{##2}{%
6028     \mfirstucMakeUppercase{\genplacrfullformat{##2}{##3}}}%

```

Make sure that `\glsentryfull` etc reflects the new style:

```

6029 \renewcommand*\glsentryfull}[1]{\genacrfullformat{##1}{}}%
6030 \renewcommand*\Glsentryfull}[1]{\Genacrfullformat{##1}{}}%
6031 \renewcommand*\glsentryfullpl}[1]{\genplacrfullformat{##1}{}}%
6032 \renewcommand*\Glsentryfullpl}[1]{\Genplacrfullformat{##1}{}}%
6033 }

```

`\GenericAcronymFields` Fields used by `\SetGenericNewAcronym` that can be changed by the acronym style.


```
6034 \newcommand*{\GenericAcronymFields}{description={\the\glslongtok}}
```

```
\acronymentry \acronymentry{<label>}
```

Display style for the name field in the list of acronyms.

```
6035 \newcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{#1}}}
```

```
\acronymsort \acronymsort{<short>}{<long>}
```

Default sort format for acronyms.

```
6036 \newcommand*{\acronymsort}[2]{#1}
```

```
\setacronymstyle \setacronymstyle{<style name>}
```

```
6037 \newcommand*{\setacronymstyle}[1]{%
6038   \ifcsundef{@glsacr@dispstyle@#1}%
6039   {%
6040     \PackageError{glossaries}{Undefined acronym style ‘#1’}{}%
6041   }%
6042   {%
6043     \ifdefempty{\@glsacronymlists}%
6044     {%
6045       \DeclareAcronymList{\acronymtype}%
6046     }%
6047   }%
6048   \SetGenericNewAcronym
6049   \GlsUseAcrStyleDefs{#1}%
6050   \@for\@gls@type:=\@glsacronymlists\do{%
6051     \defglsentryfmt[\@gls@type]{\GlsUseAcrEntryDispStyle{#1}}%
6052   }%
6053 }%
6054 }
```

```
\newacronymstyle \newacronymstyle{<style name>}{<entry format definition>}{<display
definitions>}
```

Defines a new acronym style called *<style name>*.

```
6055 \newcommand*{\newacronymstyle}[3]{%
6056   \ifcsdef{@glsacr@dispstyle@#1}%
6057   {%
6058     \PackageError{glossaries}{Acronym style ‘#1’ already exists}{}%
6059   }%
6060   {%
```

```

6061 \csdef{@glsacr@dispstyle@#1}{#2}%
6062 \csdef{@glsacr@styledefs@#1}{#3}%
6063 }%
6064 }

```

`\renewacronymstyle` Redefines the given acronym style.

```

6065 \newcommand*{\renewacronymstyle}[3]{%
6066 \ifcsdef{@glsacr@dispstyle@#1}%
6067 {%
6068 \csdef{@glsacr@dispstyle@#1}{#2}%
6069 \csdef{@glsacr@styledefs@#1}{#3}%
6070 }%
6071 {%
6072 \PackageError{glossaries}{Acronym style ‘#1’ doesn’t exist}{}%
6073 }%
6074 }

```

`useAcrEntryDispStyle`

```

6075 \newcommand*{\GlsUseAcrEntryDispStyle}[1]{\csuse{@glsacr@dispstyle@#1}}

```

`\GlsUseAcrStyleDefs`

```

6076 \newcommand*{\GlsUseAcrStyleDefs}[1]{\csuse{@glsacr@styledefs@#1}}

```

Predefined acronym styles:

`long-short` *<long>* (*<short>*) acronym style.

```

6077 \newacronymstyle{long-short}%
6078 {%

```

Check for long form in case this is a mixed glossary.

```

6079 \ifglshaslong{\glslabel}{\glsngenacfmt}{\glsngenentryfmt}%
6080 }%
6081 {%
6082 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6083 \renewcommand*{\genacrfullformat}[2]{%
6084 \glsentrylong{##1}##2\space
6085 (\protect\firstacronymfont{\glsentryshort{##1}})}%
6086 }%
6087 \renewcommand*{\Genacrfullformat}[2]{%
6088 \Glsentrylong{##1}##2\space
6089 (\protect\firstacronymfont{\glsentryshort{##1}})}%
6090 }%
6091 \renewcommand*{\genplacrfullformat}[2]{%
6092 \glsentrylongpl{##1}##2\space
6093 (\protect\firstacronymfont{\glsentryshortpl{##1}})}%
6094 }%
6095 \renewcommand*{\Genplacrfullformat}[2]{%
6096 \Glsentrylongpl{##1}##2\space
6097 (\protect\firstacronymfont{\glsentryshortpl{##1}})}%

```

```

6098 }%
6099 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}
6100 \renewcommand*{\acronymsort}[2]{##1}%
6101 \renewcommand*{\acronymfont}[1]{##1}%
6102 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
6103 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6104 }

```

short-long <short> (<long>) acronym style.

```

6105 \newacronymstyle{short-long}%
6106 {%
    Check for long form in case this is a mixed glossary.
6107 \ifglshaslong{\glslabel}{\glsngenacfmt}{\glsngenentryfmt}%
6108 }%
6109 {%
6110 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6111 \renewcommand*{\genacrfullformat}[2]{%
6112 \protect\firstacronymfont{\glsentryshort{##1}}##2\space
6113 (\glsentrylong{##1})%
6114 }%
6115 \renewcommand*{\Genacrfullformat}[2]{%
6116 \protect\firstacronymfont{\Glsentryshort{##1}}##2\space
6117 (\glsentrylong{##1})%
6118 }%
6119 \renewcommand*{\genplacrfullformat}[2]{%
6120 \protect\firstacronymfont{\glsentryshortpl{##1}}##2\space
6121 (\glsentrylongpl{##1})%
6122 }%
6123 \renewcommand*{\Genplacrfullformat}[2]{%
6124 \protect\firstacronymfont{\Glsentryshortpl{##1}}##2\space
6125 (\glsentrylongpl{##1})%
6126 }%

6127 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}
6128 \renewcommand*{\acronymsort}[2]{##1}%
6129 \renewcommand*{\acronymfont}[1]{##1}%
6130 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
6131 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6132 }

```

long-sc-short <long> (\textsc{<short>}) acronym style.

```

6133 \newacronymstyle{long-sc-short}%
6134 {%
6135 \GlsUseAcrEntryDispStyle{long-short}%
6136 }%
6137 {%
6138 \GlsUseAcrStyleDefs{long-short}%
6139 \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6140 \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%

```

6141 }

long-sm-short *<long>* (`\textsmaller{<short>}`) acronym style.

```
6142 \newacronymstyle{long-sm-short}%
6143 {%
6144   \GlsUseAcrEntryDispStyle{long-short}%
6145 }%
6146 {%
6147   \GlsUseAcrStyleDefs{long-short}%
6148   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6149   \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6150 }
```

sc-short-long *<short>* (`\textsc{<long>}`) acronym style.

```
6151 \newacronymstyle{sc-short-long}%
6152 {%
6153   \GlsUseAcrEntryDispStyle{short-long}%
6154 }%
6155 {%
6156   \GlsUseAcrStyleDefs{short-long}%
6157   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6158   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6159 }
```

sm-short-long *<short>* (`\textsmaller{<long>}`) acronym style.

```
6160 \newacronymstyle{sm-short-long}%
6161 {%
6162   \GlsUseAcrEntryDispStyle{short-long}%
6163 }%
6164 {%
6165   \GlsUseAcrStyleDefs{short-long}%
6166   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6167   \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6168 }
```

long-short-desc *<long>* (`{<short>}`) acronym style that has an accompanying description (which the user needs to supply).

```
6169 \newacronymstyle{long-short-desc}%
6170 {%
6171   \GlsUseAcrEntryDispStyle{long-short}%
6172 }%
6173 {%
6174   \GlsUseAcrStyleDefs{long-short}%
6175   \renewcommand*{\GenericAcronymFields}{}%
6176   \renewcommand*{\acronymsort}[2]{##2}%
6177   \renewcommand*{\acronymentry}[1]{%
6178     \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})%
6179 }
```

long-sc-short-desc *<long>* (`\textsc{<short>}`) acronym style that has an accompanying description (which the user needs to supply).

```
6180 \newacronymstyle{long-sc-short-desc}%
6181 {%
6182   \GlsUseAcrEntryDispStyle{long-sc-short}%
6183 }%
6184 {%
6185   \GlsUseAcrStyleDefs{long-sc-short}%
6186   \renewcommand*{\GenericAcronymFields}{}%
6187   \renewcommand*{\acronymsort}[2]{##2}%
6188   \renewcommand*{\acronymentry}[1]{%
6189     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6190 }
```

long-sm-short-desc *<long>* (`\textsmaller{<short>}`) acronym style that has an accompanying description (which the user needs to supply).

```
6191 \newacronymstyle{long-sm-short-desc}%
6192 {%
6193   \GlsUseAcrEntryDispStyle{long-sm-short}%
6194 }%
6195 {%
6196   \GlsUseAcrStyleDefs{long-sm-short}%
6197   \renewcommand*{\GenericAcronymFields}{}%
6198   \renewcommand*{\acronymsort}[2]{##2}%
6199   \renewcommand*{\acronymentry}[1]{%
6200     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6201 }
```

short-long-desc *<short>* (`{<long>}`) acronym style that has an accompanying description (which the user needs to supply).

```
6202 \newacronymstyle{short-long-desc}%
6203 {%
6204   \GlsUseAcrEntryDispStyle{short-long}%
6205 }%
6206 {%
6207   \GlsUseAcrStyleDefs{short-long}%
6208   \renewcommand*{\GenericAcronymFields}{}%
6209   \renewcommand*{\acronymsort}[2]{##2}%
6210   \renewcommand*{\acronymentry}[1]{%
6211     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6212 }
```

sc-short-long-desc *<long>* (`\textsc{<short>}`) acronym style that has an accompanying description (which the user needs to supply).

```
6213 \newacronymstyle{sc-short-long-desc}%
6214 {%
6215   \GlsUseAcrEntryDispStyle{sc-short-long}%
6216 }%
```

```

6217 {%
6218   \GlsUseAcrStyleDefs{sc-short-long}%
6219   \renewcommand*\GenericAcronymFields{}%
6220   \renewcommand*\acronymsort}[2]{##2}%
6221   \renewcommand*\acronymentry}[1]{%
6222     \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6223 }

```

sm-short-long-desc *<long>* (*\textsmaller{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

6224 \newacronymstyle{sm-short-long-desc}%
6225 {%
6226   \GlsUseAcrEntryDispStyle{sm-short-long}%
6227 }%
6228 {%
6229   \GlsUseAcrStyleDefs{sm-short-long}%
6230   \renewcommand*\GenericAcronymFields{}%
6231   \renewcommand*\acronymsort}[2]{##2}%
6232   \renewcommand*\acronymentry}[1]{%
6233     \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6234 }

```

dua *<long>* only acronym style.

```

6235 \newacronymstyle{dua}%
6236 {%

```

Check for long form in case this is a mixed glossary.

```

6237   \ifdefempty\glscustomtext
6238   {%
6239     \ifglshaslong{\glslabel}%
6240     {%
6241       \glsifplural
6242       {%

```

Plural form:

```

6243       \glscapscase
6244       {%

```

Plural form, don't adjust case:

```

6245       \glsentrylongpl{\glslabel}\glsinsert
6246       }%
6247       {%

```

Plural form, make first letter upper case:

```

6248       \Glsentrylongpl{\glslabel}\glsinsert
6249       }%
6250       {%

```

Plural form, all caps:

```

6251       \mfirstucMakeUppercase
6252       {\glsentrylongpl{\glslabel}\glsinsert}%

```

6253 }%
6254 }%
6255 {%

Singular form

6256 \gls caps case
6257 {%

Singular form, don't adjust case:

6258 \glsentrylong{\glslabel}\glsinsert
6259 }%
6260 {%

Subsequent singular form, make first letter upper case:

6261 \Glsentrylong{\glslabel}\glsinsert
6262 }%
6263 {%

Subsequent singular form, all caps:

6264 \mfirstucMakeUppercase
6265 {\glsentrylong{\glslabel}\glsinsert}%
6266 }%
6267 }%
6268 }%
6269 {%

Not an acronym:

6270 \gls gen entry fmt
6271 }%
6272 }%
6273 {\gls custom text \glsinsert}%
6274 }%
6275 {%

6276 \renewcommand*{\GenericAcronymFields}{description={\the\gls long tok}}%

6277 \renewcommand*{\acr full fmt}[3]{%
6278 \gls link[##1]{##2}{\glsentrylong{##2}##3\space
6279 (\acronymfont{\glsentryshort{##2}})}}%
6280 \renewcommand*{\Acr full fmt}[3]{%
6281 \gls link[##1]{##2}{\Glsentrylong{##2}##3\space
6282 (\acronymfont{\glsentryshort{##2}})}}%
6283 \renewcommand*{\ACR full fmt}[3]{%
6284 \gls link[##1]{##2}{%
6285 \mfirstucMakeUppercase{\glsentrylong{##2}##3\space
6286 (\acronymfont{\glsentryshort{##2}})}}}%

6287 \renewcommand*{\acr full pl fmt}[3]{%
6288 \gls link[##1]{##2}{\glsentrylongpl{##2}##3\space
6289 (\acronymfont{\glsentryshortpl{##2}})}}%

6290 \renewcommand*{\Acr full pl fmt}[3]{%
6291 \gls link[##1]{##2}{\Glsentrylongpl{##2}##3\space

```

6292     (\acronymfont{\glsentryshortpl{##2}})}%
6293 \renewcommand*\ACRfullplfmt}[3]{%
6294   \glslink[##1]{##2}{%
6295     \mfirstucMakeUppercase{\glsentrylongpl{##2}##3\space
6296     (\acronymfont{\glsentryshortpl{##2}})}%
6297 \renewcommand*\glsentryfull}[1]{%
6298   \glsentrylong{##1}\space(\acronymfont{\glsentryshort{##1}})%
6299 }%
6300 \renewcommand*\Glsentryfull}[1]{%
6301   \Glsentrylong{##1}\space(\acronymfont{\glsentryshort{##1}})%
6302 }%
6303 \renewcommand*\glsentryfullpl}[1]{%
6304   \glsentrylongpl{##1}\space(\acronymfont{\glsentryshortpl{##1}})%
6305 }%
6306 \renewcommand*\Glsentryfullpl}[1]{%
6307   \Glsentrylongpl{##1}\space(\acronymfont{\glsentryshortpl{##1}})%
6308 }%
6309 \renewcommand*\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6310 \renewcommand*\acronymsort}[2]{##1}%
6311 \renewcommand*\acronymfont}[1]{##1}%
6312 \renewcommand*\acrpluralsuffix{\glsacrpluralsuffix}%
6313 }

```

dua-desc *<long>* only acronym style with user-supplied description.

```

6314 \newacronymstyle{dua-desc}%
6315 {%
6316   \GlsUseAcrEntryDispStyle{dua}%
6317 }%
6318 {%
6319   \GlsUseAcrStyleDefs{dua}%
6320 \renewcommand*\GenericAcronymFields{}%
6321 \renewcommand*\acronymentry}[1]{\acronymfont{\glsentrylong{##1}}}%
6322 \renewcommand*\acronymsort}[2]{##2}%
6323 }%

```

footnote *<short>*\footnote{*<long>*} acronym style.

```

6324 \newacronymstyle{footnote}%
6325 {%
6326   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6327 }%
6328 {%
6329   \renewcommand*\GenericAcronymFields{description={\the\glslongtok}}%
6330   \glshyperfirstfalse
6331   \renewcommand*\genacrfullformat}[2]{%
6332     \protect\firstacronymfont{\glsentryshort{##1}}##2%

```



```

6333 \protect\footnote{\glsentrylong{##1}}%
6334 }%
6335 \renewcommand*\Genacrfullformat}[2]{%
6336 \firstacronymfont{\Glsentryshort{##1}}##2%
6337 \protect\footnote{\glsentrylong{##1}}%
6338 }%
6339 \renewcommand*\genplacrfullformat}[2]{%
6340 \protect\firstacronymfont{\glsentryshortpl{##1}}##2%
6341 \protect\footnote{\glsentrylongpl{##1}}%
6342 }%
6343 \renewcommand*\Genplacrfullformat}[2]{%
6344 \protect\firstacronymfont{\Glsentryshortpl{##1}}##2%
6345 \protect\footnote{\glsentrylongpl{##1}}%
6346 }%
6347 \renewcommand*\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6348 \renewcommand*\acronymsort}[2]{##1}%
6349 \renewcommand*\acronymfont}[1]{##1}%
6350 \renewcommand*\acrpluralsuffix{\glsacrpluralsuffix}%

```

Don't use footnotes for \acrfull:

```

6351 \renewcommand*\acrfullfmt}[3]{%
6352 \glslink[##1]{##2}{\acronymfont{\glsentryshort{##2}}##3\space
6353 (\glsentrylong{##2})}%
6354 \renewcommand*\Acrfullfmt}[3]{%
6355 \glslink[##1]{##2}{\acronymfont{\Glsentryshort{##2}}##3\space
6356 (\glsentrylong{##2})}%
6357 \renewcommand*\ACRfullfmt}[3]{%
6358 \glslink[##1]{##2}{%
6359 \mfirstucMakeUppercase{\acronymfont{\glsentryshort{##2}}##3\space
6360 (\glsentrylong{##2})}}%
6361 \renewcommand*\acrfullplfmt}[3]{%
6362 \glslink[##1]{##2}{\acronymfont{\glsentryshortpl{##2}}##3\space
6363 (\glsentrylongpl{##2})}%
6364 \renewcommand*\Acrfullplfmt}[3]{%
6365 \glslink[##1]{##2}{\acronymfont{\Glsentryshortpl{##2}}##3\space
6366 (\glsentrylongpl{##2})}%
6367 \renewcommand*\ACRfullplfmt}[3]{%
6368 \glslink[##1]{##2}{%
6369 \mfirstucMakeUppercase{\acronymfont{\glsentryshortpl{##2}}##3\space
6370 (\glsentrylongpl{##2})}}%

```

Similarly for \glsentryfull etc:

```

6371 \renewcommand*\glsentryfull}[1]{%
6372 \acronymfont{\glsentryshort{##1}}\space(\glsentrylong{##1})}%
6373 \renewcommand*\Glsentryfull}[1]{%
6374 \acronymfont{\Glsentryshort{##1}}\space(\glsentrylong{##1})}%
6375 \renewcommand*\glsentryfullpl}[1]{%
6376 \acronymfont{\glsentryshortpl{##1}}\space(\glsentrylongpl{##1})}%
6377 \renewcommand*\Glsentryfullpl}[1]{%
6378 \acronymfont{\Glsentryshortpl{##1}}\space(\glsentrylongpl{##1})}%

```

6379 }

footnote-sc \textsc{<short>}\footnote{<long>} acronym style.

```
6380 \newacronymstyle{footnote-sc}%
6381 {%
6382 \GlsUseAcrEntryDispStyle{footnote}%
6383 }%
6384 {%
6385 \GlsUseAcrStyleDefs{footnote}%
6386 \renewcommand{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}
6387 \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6388 \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6389 }%
```

footnote-sm \textsmaller{<short>}\footnote{<long>} acronym style.

```
6390 \newacronymstyle{footnote-sm}%
6391 {%
6392 \GlsUseAcrEntryDispStyle{footnote}%
6393 }%
6394 {%
6395 \GlsUseAcrStyleDefs{footnote}%
6396 \renewcommand{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}
6397 \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6398 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6399 }%
```

footnote-desc <short>\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```
6400 \newacronymstyle{footnote-desc}%
6401 {%
6402 \GlsUseAcrEntryDispStyle{footnote}%
6403 }%
6404 {%
6405 \GlsUseAcrStyleDefs{footnote}%
6406 \renewcommand*{\GenericAcronymFields}{}%
6407 \renewcommand*{\acronymsort}[2]{##2}%
6408 \renewcommand*{\acronymentry}[1]{%
6409 \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6410 }
```

footnote-sc-desc \textsc{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```
6411 \newacronymstyle{footnote-sc-desc}%
6412 {%
6413 \GlsUseAcrEntryDispStyle{footnote-sc}%
6414 }%
6415 {%
6416 \GlsUseAcrStyleDefs{footnote-sc}%
```

```

6417 \renewcommand*{\GenericAcronymFields}{}%
6418 \renewcommand*{\acronymsort}[2]{##2}%
6419 \renewcommand*{\acronymentry}[1]{%
6420   \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6421 }

```

footnote-sm-desc \textsmaller{\short}\footnote{\long} acronym style that has an accompanying description (which the user needs to supply).

```

6422 \newacronymstyle{footnote-sm-desc}%
6423 {%
6424   \GlsUseAcrEntryDispStyle{footnote-sm}%
6425 }%
6426 {%
6427   \GlsUseAcrStyleDefs{footnote-sm}%
6428   \renewcommand*{\GenericAcronymFields}{}%
6429   \renewcommand*{\acronymsort}[2]{##2}%
6430   \renewcommand*{\acronymentry}[1]{%
6431     \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6432 }

```

fineAcronymSynonyms

```

6433 \newcommand*{\DefineAcronymSynonyms}{%

```

Short form

\acs

```

6434 \let\acs\acrshort

```

First letter uppercase short form

\Acs

```

6435 \let\Acs\Acrshort

```

Plural short form

\acsp

```

6436 \let\acsp\acrshortpl

```

First letter uppercase plural short form

\Acsp

```

6437 \let\Acsp\Acrshortpl

```

Long form

\acl

```

6438 \let\acl\aclong

```

Plural long form

`\aclp`
 6439 `\let\aclp\acrlongpl`
 First letter upper case long form

`\Acl`
 6440 `\let\Acl\Acrlong`
 First letter upper case plural long form

`\Aclp`
 6441 `\let\Aclp\Acrlongpl`
 Full form

`\acf`
 6442 `\let\acf\acrfull`
 Plural full form

`\acfp`
 6443 `\let\acfp\acrfullpl`
 First letter upper case full form

`\Acf`
 6444 `\let\Acf\Acrfull`
 First letter upper case plural full form

`\Acfp`
 6445 `\let\Acfp\Acrfullpl`
 Standard form

`\ac`
 6446 `\let\ac\gls`
 First upper case standard form

`\Ac`
 6447 `\let\Ac\Gls`
 Standard plural form

`\acp`
 6448 `\let\acp\glspl`
 Standard first letter upper case plural form

`\Acp`
 6449 `\let\Acp\Glspl`

6450 }

Define synonyms if required

```
6451 \ifglacrshortcuts
6452   \DefineAcronymSynonyms
6453 \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.

```
6454 \newcommand*\SetDefaultAcronymDisplayStyle[1]{%
6455   \defglentryfmt[#1]{\glsgenentryfmt}%
6456 }
```

defaultNewAcronymDef Sets up the acronym definition for the default style. The information is provided by the tokens `\glslabeltok`, `\glsshorttok`, `\glslongtok` and `\glskeylisttok`.

```
6457 \newcommand*\DefaultNewAcronymDef{%
6458   \edef\@do@newglossaryentry{%
6459     \noexpand\newglossaryentry{\the\glslabeltok}%
6460     {%
6461       type=\acronymtype,%
6462       name={\the\glsshorttok},%
6463       sort={\the\glsshorttok},%
6464       text={\the\glsshorttok},%
6465       first={\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
6466       plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6467       firstplural={\acrfullformat{\noexpand\expandonce\noexpand\@glo@longpl}%
6468                   {\noexpand\expandonce\noexpand\@glo@shortpl}},%
6469       short={\the\glsshorttok},%
6470       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6471       long={\the\glslongtok},%
6472       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6473       description={\the\glslongtok},%
6474       descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
```

Remaining options specified by the user:

```
6475     \the\glskeylisttok
6476   }%
6477 }%
6478 \let\@org@gls@assign@firstpl\gls@assign@firstpl
6479 \let\@org@gls@assign@plural\gls@assign@plural
6480 \let\@org@gls@assign@descplural\gls@assign@descplural
6481 \def\gls@assign@firstpl##1##2{%
6482   \@@gls@expand@field{##1}{firstpl}{##2}%
6483 }%
6484 \def\gls@assign@plural##1##2{%
6485   \@@gls@expand@field{##1}{plural}{##2}%
6486 }%
```

```

6487 \def\gls@assign@descplural##1##2{%
6488   \@@gls@expand@field{##1}{descplural}{##2}%
6489 }%
6490 \do@newglossaryentry
6491 \let\gls@assign@firstpl\@org@gls@assign@firstpl
6492 \let\gls@assign@plural\@org@gls@assign@plural
6493 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
6494 }

```

`DefaultAcronymStyle` Set up the default acronym style:

```
6495 \newcommand*\SetDefaultAcronymStyle{%
```

Set the display style:

```

6496 \@for\@gls@type:=\@gls@acronymlists\do{%
6497   \SetDefaultAcronymDisplayStyle{\@gls@type}%
6498 }%

```

Set up the definition of `\newacronym`:

```
6499 \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).

```

6500   \ifx\@gls@acronymlists\@empty
6501     \def\@glo@type{\acronymtype}%
6502     \setkeys{glossentry}{##1}%
6503     \DeclareAcronymList{\@glo@type}%
6504     \SetDefaultAcronymDisplayStyle{\@glo@type}%
6505   \fi
6506   \glskeylisttok{##1}%
6507   \glslabeltok{##2}%
6508   \glsshorttok{##3}%
6509   \glslongtok{##4}%
6510   \newacronymhook
6511   \DefaultNewAcronymDef
6512 }%
6513 \renewcommand*\acrpluralsuffix{\glsacrpluralsuffix}%
6514 }

```

`\acrfootnote` Used by the footnote acronym styles.

```
6515 \newcommand*\acrfootnote}[3]{\acrlinkfootnote{##1}{##2}{##3}}
```

`\acrlinkfootnote`

```

6516 \newcommand*\acrlinkfootnote}[3]{%
6517   \footnote{\glslink[##1]{##2}{##3}}%
6518 }

```

`\acrnofootnote`

```

6519 \newcommand*\acrnofootnote}[3]{%
6520   \footnote{##3}%
6521 }

```

AcronymDisplayStyle Sets the acronym display style for given glossary for the description and footnote combination.

```

6522 \newcommand*{\SetDescriptionFootnoteAcronymDisplayStyle}[1]{%
6523   \def\glsentryfmt[#1]{%

6524     \ifdefempty\glscustomtext
6525     {%
6526       \ifglsused{\glslabel}%
6527       {%
6528         \acronymfont{\glsgenentryfmt}%
6529       }%
6530     {%
6531       \firstacronymfont{\glsgenentryfmt}%
6532       \ifgls hassymbol{\glslabel}%
6533       {%
6534         \expandafter\protect\expandafter\acrfootnote\expandafter
6535         {\@gls@link@opts}{\@gls@link@label}%
6536       {%
6537         \glsifplural
6538         {\glsentrysymbolplural{\glslabel}}%
6539         {\glsentrysymbol{\glslabel}}%
6540       }%
6541     }%
6542   }%
6543 }%
6544 {\glscustomtext\glsinsert}%
6545 }%
6546 }

```

otnoteNewAcronymDef

```

6547 \newcommand*{\DescriptionFootnoteNewAcronymDef}{%
6548   \edef\@do@newglossaryentry{%
6549     \noexpand\newglossaryentry{\the\glslabeltok}%
6550   {%
6551     type=\acronymtype,%
6552     name={\noexpand\acronymfont{\the\glsshorttok}},%
6553     sort={\the\glsshorttok},%
6554     first={\the\glsshorttok},%
6555     firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6556     text={\the\glsshorttok},%
6557     plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6558     short={\the\glsshorttok},%
6559     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6560     long={\the\glslongtok},%
6561     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6562     symbol={\the\glslongtok},%
6563     symbolplural={\noexpand\expandonce\noexpand\@glo@longpl},%
6564     \the\glskeylisttok
6565   }%

```

```

6566 }%
6567 \let\@org@gls@assign@firstpl\gls@assign@firstpl
6568 \let\@org@gls@assign@plural\gls@assign@plural
6569 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
6570 \def\gls@assign@firstpl##1##2{%
6571   \@@gls@expand@field{##1}{firstpl}{##2}%
6572 }%
6573 \def\gls@assign@plural##1##2{%
6574   \@@gls@expand@field{##1}{plural}{##2}%
6575 }%
6576 \def\gls@assign@symbolplural##1##2{%
6577   \@@gls@expand@field{##1}{symbolplural}{##2}%
6578 }%
6579 \do@newglossaryentry
6580 \let\gls@assign@plural\@org@gls@assign@plural
6581 \let\gls@assign@firstpl\@org@gls@assign@firstpl
6582 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
6583 }

```

`FootnoteAcronymStyle` If a description and footnote are both required, store the long form in the symbol key. Store the short form in text key. Note that since the long form is stored in the symbol key, if you want the long form to appear in the list of acronyms, you need to use a glossary style that displays the symbol key.

```

6584 \newcommand*{\SetDescriptionFootnoteAcronymStyle}{%
6585   \renewcommand{\newacronym}[4][]{%
6586     \ifx\@glsacronymlists\@empty
6587       \def\@glo@type{\acronymtype}%
6588       \setkeys{glossentry}{##1}%
6589       \DeclareAcronymList{\@glo@type}%
6590       \SetDescriptionFootnoteAcronymDisplayStyle{\@glo@type}%
6591     \fi
6592     \glskeylisttok{##1}%
6593     \glslabeltok{##2}%
6594     \glsshorttok{##3}%
6595     \glslongtok{##4}%
6596     \newacronymhook
6597     \DescriptionFootnoteNewAcronymDef
6598   }%

```

If footnote package option is specified, set the first use to append the long form (stored in symbol) as a footnote.

```

6599 \@for\@gls@type:=\@glsacronymlists\do{%
6600   \SetDescriptionFootnoteAcronymDisplayStyle{\@gls@type}%
6601 }%

```

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.

```

6602 \ifglsacrsmallcaps
6603   \renewcommand*{\acronymfont}[1]{\textsc{##1}}%

```



```

6604 \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6605 \else
6606 \ifglsacrsmaller
6607 \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
6608 \fi
6609 \fi

```

Check for package option clash

```

6610 \ifglsacrdua
6611 \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’
6612 can’t both be set}{}%
6613 \fi
6614 }%

```

AcronymDisplayStyle Sets the acronym display style for given glossary with description and dua combination.

```

6615 \newcommand*{\SetDescriptionDUAAcronymDisplayStyle}[1]{%
6616 \defglsentryfmt[##1]{\glsgenentryfmt}%
6617 }

```

ionDUANewAcronymDef

```

6618 \newcommand*{\DescriptionDUANewAcronymDef}{%
6619 \edef\@do@newglossaryentry{%
6620 \noexpand\newglossaryentry{\the\glslabeltok}%
6621 {%
6622 type=\acronymtype,%
6623 name={\the\glslongtok},%
6624 sort={\the\glslongtok},
6625 text={\the\glslongtok},%
6626 first={\the\glslongtok},%
6627 plural={\noexpand\expandonce\noexpand\@glo@longpl},%
6628 firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
6629 short={\the\glsshorttok},%
6630 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6631 long={\the\glslongtok},%
6632 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6633 symbol={\the\glsshorttok},%
6634 symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6635 \the\glskeylisttok
6636 }%
6637 }%
6638 \let\@org@gls@assign@firstpl\gls@assign@firstpl
6639 \let\@org@gls@assign@plural\gls@assign@plural
6640 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
6641 \def\gls@assign@firstpl##1##2{%
6642 \@gls@expand@field{##1}{firstpl}{##2}%
6643 }%
6644 \def\gls@assign@plural##1##2{%
6645 \@gls@expand@field{##1}{plural}{##2}%

```

```

6646 }%
6647 \def\gls@assign@symbolplural##1##2{%
6648   \@@gls@expand@field{##1}{symbolplural}{##2}%
6649 }%
6650 \@do@newglossaryentry
6651 \let\gls@assign@firstpl\@org@gls@assign@firstpl
6652 \let\gls@assign@plural\@org@gls@assign@plural
6653 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
6654 }

```

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.

```

6655 \newcommand*\SetDescriptionDUAAcronymStyle{%
6656   \ifglsacrsmallcaps
6657     \PackageError{glossaries}{Option clash: 'smallcaps' and 'dua'
6658       can't both be set}{}%
6659   \else
6660     \ifglsacrsmaller
6661       \PackageError{glossaries}{Option clash: 'smaller' and 'dua'
6662         can't both be set}{}%
6663     \fi
6664   \fi
6665   \renewcommand{\newacronym}[4][[]]{%
6666     \ifx\@glsacronymlists\@empty
6667       \def\@glo@type{\acronymtype}%
6668       \setkeys{glossentry}{##1}%
6669       \DeclareAcronymList{\@glo@type}%
6670       \SetDescriptionDUAAcronymDisplayStyle{\@glo@type}%
6671     \fi
6672     \glskeylisttok{##1}%
6673     \glslabeltok{##2}%
6674     \glsshorttok{##3}%
6675     \glslongtok{##4}%
6676     \newacronymhook
6677     \DescriptionDUANewAcronymDef
6678   }%

```

Set display.

```

6679   \@for\@gls@type:=\@glsacronymlists\do{%
6680     \SetDescriptionDUAAcronymDisplayStyle{\@gls@type}%
6681   }%
6682 }%

```

AcronymDisplayStyle Sets the acronym display style for given glossary using the description setting (but not footnote or dua).

```

6683 \newcommand*\SetDescriptionAcronymDisplayStyle}[1]{%
6684   \defglsentryfmt[#1]{%

```

```

6685 \ifdefempty\glscustomtext
6686 {%
6687 \ifglssused{\glslabel}%
6688 {%

Move the inserted text outside of \acronymfont
6689 \let\gls@org@insert\glsinsert
6690 \let\glsinsert\@empty
6691 \acronymfont{\gls@genentryfmt}\gls@org@insert
6692 }%
6693 {%
6694 \gls@genentryfmt
6695 \ifglsshassymbol{\glslabel}%
6696 {%
6697 \glsifplural
6698 {%
6699 \def\@glo@symbol{\glsentrysymbolplural{\glslabel}}%
6700 }%
6701 {%
6702 \def\@glo@symbol{\glsentrysymbol{\glslabel}}%
6703 }%
6704 \space(\protect\firstacronymfont
6705 {\gls@scapscase
6706 {\@glo@symbol}
6707 {\@glo@symbol}
6708 {\mfirstucMakeUppercase{\@glo@symbol}}})%
6709 }%
6710 }%
6711 }%
6712 }%
6713 {\glscustomtext\glsinsert}%
6714 }%
6715 }

```

ptionNewAcronymDef

```

6716 \newcommand*\DescriptionNewAcronymDef{%
6717 \edef\@do@newglossaryentry{%
6718 \noexpand\newglossaryentry{\the\glslabeltok}%
6719 {%
6720 type=\acronymtype,%
6721 name={\noexpand
6722 \acrnameformat{\the\glsshorttok}{\the\glslongtok}},%
6723 sort={\the\glsshorttok},%
6724 first={\the\glslongtok},%
6725 firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
6726 text={\the\glsshorttok},%
6727 plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6728 short={\the\glsshorttok},%
6729 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6730 long={\the\glslongtok},%

```

```

6731     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6732     symbol={\noexpand\@glo@text},%
6733     symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6734     \the\glskeylisttok}%
6735 }%
6736 \let\@org@gls@assign@firstpl\gls@assign@firstpl
6737 \let\@org@gls@assign@plural\gls@assign@plural
6738 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
6739 \def\gls@assign@firstpl##1##2{%
6740     \@@gls@expand@field{##1}{firstpl}{##2}%
6741 }%
6742 \def\gls@assign@plural##1##2{%
6743     \@@gls@expand@field{##1}{plural}{##2}%
6744 }%
6745 \def\gls@assign@symbolplural##1##2{%
6746     \@@gls@expand@field{##1}{symbolplural}{##2}%
6747 }%
6748 \do@newglossaryentry
6749 \let\gls@assign@firstpl\@org@gls@assign@firstpl
6750 \let\gls@assign@plural\@org@gls@assign@plural
6751 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
6752 }

```

`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.

```

6753 \newcommand*{\SetDescriptionAcronymStyle}{%
6754     \renewcommand{\newacronym}[4] []{%
6755         \ifx\@glsacronymlists\@empty
6756             \def\@glo@type{\acronymtype}%
6757             \setkeys{glossentry}{##1}%
6758             \DeclareAcronymList{\@glo@type}%
6759             \SetDescriptionAcronymDisplayStyle{\@glo@type}%
6760             \fi
6761             \glskeylisttok{##1}%
6762             \glslabeltok{##2}%
6763             \glsshorttok{##3}%
6764             \glslongtok{##4}%
6765             \newacronymhook
6766             \DescriptionNewAcronymDef
6767         }%

```

Set display.

```

6768     \@for\@gls@type:=\@glsacronymlists\do{%
6769         \SetDescriptionAcronymDisplayStyle{\@gls@type}%
6770     }%

```

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.

```
6771 \ifglsacrsmallcaps
6772 \renewcommand{\acronymfont}[1]{\textsc{##1}}
6773 \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6774 \else
6775 \ifglsacrsmaller
6776 \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
6777 \fi
6778 \fi
6779}%
```

AcronymDisplayStyle Sets the acronym display style for given glossary with footnote setting (but not description or dua).

```
6780 \newcommand*{\SetFootnoteAcronymDisplayStyle}[1]{%
6781 \defglsentryfmt[#1]{%
```

```
6782 \ifdefempty\glscustomtext
6783 {%
```

Move the inserted text outside of `\acronymfont`

```
6784 \let\gls@org@insert\glsinsert
6785 \let\glsinsert\@empty
6786 \ifglsused{\glslabel}%
6787 {%
6788 \acronymfont{\glsgenentryfmt}\gls@org@insert
6789 }%
6790 {%
6791 \firstacronymfont{\glsgenentryfmt}\gls@org@insert
6792 \ifglsahaslong{\glslabel}%
6793 {%
6794 \expandafter\protect\expandafter\acrfootnote\expandafter
6795 {\@gls@link@opts}{\@gls@link@label}%
6796 {%
6797 \glsifplural
6798 {\glsentrylongpl{\glslabel}}%
6799 {\glsentrylong{\glslabel}}%
6800 }%
6801 }%
6802 }%
6803 }%
6804 }%
6805 {\glscustomtext\glsinsert}%
6806 }%
6807 }
```

FootnoteNewAcronymDef

```
6808 \newcommand*{\FootnoteNewAcronymDef}{%
6809 \edef\@do@newglossaryentry{%
```

```

6810 \noexpand\newglossaryentry{\the\glslabeltok}%
6811 {%
6812   type=\acronymtype,%
6813   name={\noexpand\acronymfont{\the\glsshorttok}},%
6814   sort={\the\glsshorttok},%
6815   text={\the\glsshorttok},%
6816   plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6817   first={\the\glsshorttok},%
6818   firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6819   short={\the\glsshorttok},%
6820   shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6821   long={\the\glslongtok},%
6822   longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6823   description={\the\glslongtok},%
6824   descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
6825   \the\glskeylisttok
6826 }%
6827 }%
6828 \let\@org@gls@assign@plural\gls@assign@plural
6829 \let\@org@gls@assign@firstpl\gls@assign@firstpl
6830 \let\@org@gls@assign@descplural\gls@assign@descplural
6831 \def\gls@assign@firstpl##1##2{%
6832   \@@gls@expand@field{##1}{firstpl}{##2}%
6833 }%
6834 \def\gls@assign@plural##1##2{%
6835   \@@gls@expand@field{##1}{plural}{##2}%
6836 }%
6837 \def\gls@assign@descplural##1##2{%
6838   \@@gls@expand@field{##1}{descplural}{##2}%
6839 }%
6840 \@do@newglossaryentry
6841 \let\gls@assign@plural\@org@gls@assign@plural
6842 \let\gls@assign@firstpl\@org@gls@assign@firstpl
6843 \let\gls@assign@descplural\@org@gls@assign@descplural
6844 }

```

`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.

```

6845 \newcommand*\SetFootnoteAcronymStyle{%
6846   \renewcommand{\newacronym}[4][ ]{%
6847     \ifx\@glsacronymlists\@empty
6848       \def\@glo@type{\acronymtype}%
6849       \setkeys{glossentry}{##1}%
6850       \DeclareAcronymList{\@glo@type}%
6851       \SetFootnoteAcronymDisplayStyle{\@glo@type}%
6852     \fi
6853     \glskeylisttok{##1}%
6854     \glslabeltok{##2}%

```

```

6855 \glsshorttok{##3}%
6856 \glslongtok{##4}%
6857 \newacronymhook
6858 \FootnoteNewAcronymDef
6859 }%

```

Set display

```

6860 \@for\@gls@type:=\@gls@acronymlists\do{%
6861 \SetFootnoteAcronymDisplayStyle{\@gls@type}%
6862 }%

```

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.

```

6863 \ifglsacrsmallcaps
6864 \renewcommand*\acronymfont}[1]{\textsc{##1}}%
6865 \renewcommand*\acrpluralsuffix{\glsupacrpluralsuffix}%
6866 \else
6867 \ifglsacrsmaller
6868 \renewcommand*\acronymfont}[1]{\textsmaller{##1}}%
6869 \fi
6870 \fi

```

Check for option clash

```

6871 \ifglsacrdua
6872 \PackageError{glossaries}{Option clash: 'footnote' and 'dua'
6873 can't both be set}{}%
6874 \fi
6875 }%

```

`\doparenifnotempty` 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.

```

6876 \DeclareRobustCommand*\glsdoparenifnotempty}[2]{%
6877 \protected@edef\gls@tmp{##1}%
6878 \ifdefempty\gls@tmp
6879 {}%
6880 {%
6881 \ifx\gls@tmp\@gls@default@value
6882 \else
6883 \space (#2{##1})%
6884 \fi
6885 }%
6886 }

```

`AcronymDisplayStyle` Sets the acronym display style for given glossary where neither footnote nor description is required, but smallcaps or smaller specified.

```

6887 \newcommand*\SetSmallAcronymDisplayStyle}[1]{%
6888 \defglsentryfmt[##1]{%

```

```

6889 \ifdefempty\glscustomtext
6890 {%
    Move the inserted text outside of \acronymfont
6891     \let\gls@org@insert\glsinsert
6892     \let\glsinsert\@empty
6893     \ifglsused{\glslabel}%
6894     {%
6895         \acronymfont{\glsgenentryfmt}\gls@org@insert
6896     }%
6897     {%
6898         \glsgenentryfmt
6899         \ifgls hassymbol{\glslabel}%
6900         {%
6901             \glsifplural
6902             {%
6903                 \def\@glo@symbol{\glsentrysymbolplural{\glslabel}}%
6904             }%
6905             {%
6906                 \def\@glo@symbol{\glsentrysymbol{\glslabel}}%
6907             }%
6908             \space
6909             (\gls caps case
6910             {\firstacronymfont{\@glo@symbol}}%
6911             {\firstacronymfont{\@glo@symbol}}%
6912             {\firstacronymfont{\mfirstucMakeUppercase{\@glo@symbol}}})%
6913         }%
6914     }%
6915 }%
6916 }%
6917 {\glscustomtext\glsinsert}%
6918 }%
6919 }

```

\SmallNewAcronymDef

```

6920 \newcommand*\SmallNewAcronymDef{%
6921     \edef\@do@newglossaryentry{%
6922         \noexpand\newglossaryentry{\the\glslabeltok}%
6923         {%
6924             type=\acronymtype,%
6925             name={\noexpand\acronymfont{\the\glsshorttok}},%
6926             sort={\the\glsshorttok},%
6927             text={\the\glsshorttok},%
        }
        Default to the short plural.
6928         plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6929         first={\the\gls longtok},%
        Default to the long plural.
6930         firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%

```



```

6931 short={\the\glsshorttok},%
6932 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6933 long={\the\glslongtok},%
6934 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6935 description={\noexpand\@glo@first},%
6936 descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
6937 symbol={\the\glsshorttok},%

```

Default to the short plural.

```

6938 symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6939 \the\glskeylisttok
6940 }%
6941 }%
6942 \let\@org@gls@assign@firstpl\gls@assign@firstpl
6943 \let\@org@gls@assign@plural\gls@assign@plural
6944 \let\@org@gls@assign@descplural\gls@assign@descplural
6945 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
6946 \def\gls@assign@firstpl##1##2{%
6947 \@@gls@expand@field{##1}{firstpl}{##2}%
6948 }%
6949 \def\gls@assign@plural##1##2{%
6950 \@@gls@expand@field{##1}{plural}{##2}%
6951 }%
6952 \def\gls@assign@descplural##1##2{%
6953 \@@gls@expand@field{##1}{descplural}{##2}%
6954 }%
6955 \def\gls@assign@symbolplural##1##2{%
6956 \@@gls@expand@field{##1}{symbolplural}{##2}%
6957 }%
6958 \do@newglossaryentry
6959 \let\gls@assign@firstpl\@org@gls@assign@firstpl
6960 \let\gls@assign@plural\@org@gls@assign@plural
6961 \let\gls@assign@descplural\@org@gls@assign@descplural
6962 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
6963 }

```

`\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.

```

6964 \newcommand*\SetSmallAcronymStyle{%
6965 \renewcommand{\newacronym}[4][[]]{%
6966 \ifx\@glsacronymlists\@empty
6967 \def\@glo@type{\acronymtype}%
6968 \setkeys{glossentry}{##1}%
6969 \DeclareAcronymList{\@glo@type}%
6970 \SetSmallAcronymDisplayStyle{\@glo@type}%
6971 \fi
6972 \glskeylisttok{##1}%
6973 \glslabeltok{##2}%
6974 \glsshorttok{##3}%
6975 \glslongtok{##4}%

```

```

6976 \newacronymhook
6977 \SmallNewAcronymDef
6978 }%

```

Change the display since first only contains long form.

```

6979 \@for\@gls@type:=\@glsacronymlists\do{%
6980 \SetSmallAcronymDisplayStyle{\@gls@type}%
6981 }%

```

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.

```

6982 \ifglsacrsmallcaps
6983 \renewcommand*\acronymfont}[1]{\textsc{##1}}
6984 \renewcommand*\acrpluralsuffix{\glsupacrpluralsuffix}%
6985 \else
6986 \renewcommand*\acronymfont}[1]{\textsmaller{##1}}
6987 \fi

```

check for option clash

```

6988 \ifglsacrdua
6989 \ifglsacrsmallcaps
6990 \PackageError{glossaries}{Option clash: 'smallcaps' and 'dua'
6991 can't both be set}{}%
6992 \else
6993 \PackageError{glossaries}{Option clash: 'smaller' and 'dua'
6994 can't both be set}{}%
6995 \fi
6996 \fi
6997 }%

```

`\SetDUADisplayStyle` Sets the acronym display style for given glossary with dua setting.

```

6998 \newcommand*\SetDUADisplayStyle}[1]{%
6999 \defglsentryfmt[#1]{\glsentryfmt}%
7000 }

```

`\DUANewAcronymDef`

```

7001 \newcommand*\DUANewAcronymDef}{%
7002 \edef\@do@newglossaryentry{%
7003 \noexpand\newglossaryentry{\the\glslabeltok}%
7004 {%
7005 type=\acronymtype,%
7006 name={\the\glsshorttok},%
7007 text={\the\glslongtok},%
7008 first={\the\glslongtok},%
7009 plural={\noexpand\expandonce\noexpand\@glo@longpl},%
7010 firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7011 short={\the\glsshorttok},%
7012 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7013 long={\the\glslongtok},%

```

```

7014     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7015     description={\the\glslongtok},%
7016     descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7017     symbol={\the\glsshorttok},%
7018     symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7019     \the\glskeylisttok
7020 }%
7021 }%
7022 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7023 \let\@org@gls@assign@plural\gls@assign@plural
7024 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7025 \let\@org@gls@assign@descplural\gls@assign@descplural
7026 \def\gls@assign@firstpl##1##2{%
7027   \@@gls@expand@field{##1}{firstpl}{##2}%
7028 }%
7029 \def\gls@assign@plural##1##2{%
7030   \@@gls@expand@field{##1}{plural}{##2}%
7031 }%
7032 \def\gls@assign@symbolplural##1##2{%
7033   \@@gls@expand@field{##1}{symbolplural}{##2}%
7034 }%
7035 \def\gls@assign@descplural##1##2{%
7036   \@@gls@expand@field{##1}{descplural}{##2}%
7037 }%
7038 \do@newglossaryentry
7039 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7040 \let\gls@assign@plural\@org@gls@assign@plural
7041 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7042 \let\gls@assign@descplural\@org@gls@assign@descplural
7043 }

```

`\SetDUASyle` Always expand acronyms.

```

7044 \newcommand*\SetDUASyle{%
7045   \renewcommand{\newacronym}[4][[]]{%
7046     \ifx\@glsacronymlists\@empty
7047       \def\@glo@type{\acronymtype}%
7048       \setkeys{glossentry}{##1}%
7049       \DeclareAcronymList{\@glo@type}%
7050       \SetDUADisplayStyle{\@glo@type}%
7051     \fi
7052     \glskeylisttok{##1}%
7053     \glslabeltok{##2}%
7054     \glsshorttok{##3}%
7055     \glslongtok{##4}%
7056     \newacronymhook
7057     \DUANewAcronymDef
7058   }%

```

Set the display

```

7059   \@for\@gls@type:=\@glsacronymlists\do{%

```

```

7060 \SetDUADisplayStyle{\@gls@type}%
7061 }%
7062 }

```

\SetAcronymStyle

```

7063 \newcommand*\SetAcronymStyle{%
7064 \SetDefaultAcronymStyle
7065 \ifglsacrdescription
7066 \ifglsacrfootnote
7067 \SetDescriptionFootnoteAcronymStyle
7068 \else
7069 \ifglsacrdua
7070 \SetDescriptionDUAACronymStyle
7071 \else
7072 \SetDescriptionAcronymStyle
7073 \fi
7074 \fi
7075 \else
7076 \ifglsacrfootnote
7077 \SetFootnoteAcronymStyle
7078 \else
7079 \ifthenelse{\boolean{glsacrsmalldcaps}\OR
7080 \boolean{glsacrsmaller}}{%
7081 }%
7082 \SetSmallAcronymStyle
7083 }%
7084 }%
7085 \ifglsacrdua
7086 \SetDUASStyle
7087 \fi
7088 }%
7089 \fi
7090 \fi
7091 }

```

Set the acronym style according to the package options

```
7092 \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.

```

7093 \newcommand*\SetCustomDisplayStyle}[1]{%
7094 \defglsentryfmt[#1]{\glsentryfmt}%
7095 }

```

CustomAcronymFields

```

7096 \newcommand*{\CustomAcronymFields}{%
7097   name={\the\glsshorttok},%
7098   description={\the\glslongtok},%
7099   first={\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
7100   firstplural={\acrfullformat
7101     {\noexpand\glsentrylongpl{\the\glslabeltok}}%
7102     {\noexpand\glsentryshortpl{\the\glslabeltok}}},%

7103   text={\the\glsshorttok},%
7104   plural={\the\glsshorttok\noexpand\acrpluralsuffix}%
7105 }

```

CustomNewAcronymDef

```

7106 \newcommand*{\CustomNewAcronymDef}{%
7107   \protected@edef\@do@newglossaryentry{%
7108     \noexpand\newglossaryentry{\the\glslabeltok}%
7109     {%
7110       type=\acronymtype,%
7111       short={\the\glsshorttok},%
7112       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7113       long={\the\glslongtok},%
7114       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7115       user1={\the\glsshorttok},%
7116       user2={\the\glsshorttok\noexpand\acrpluralsuffix},%
7117       user3={\the\glslongtok},%
7118       user4={\the\glslongtok\noexpand\acrpluralsuffix},%
7119       \CustomAcronymFields,%
7120       \the\glskeylisttok
7121     }%
7122   }%
7123   \@do@newglossaryentry
7124 }

```

\SetCustomStyle

```

7125 \newcommand*{\SetCustomStyle}{%
7126   \renewcommand{\newacronym}[4][[]]{%
7127     \ifx\@glsacronymlists\@empty
7128       \def\@glo@type{\acronymtype}%
7129       \setkeys{glossentry}{##1}%
7130       \DeclareAcronymList{\@glo@type}%
7131       \SetCustomDisplayStyle{\@glo@type}%
7132     \fi
7133     \glskeylisttok{##1}%
7134     \glslabeltok{##2}%
7135     \glsshorttok{##3}%
7136     \glslongtok{##4}%
7137     \newacronymhook
7138     \CustomNewAcronymDef
7139   }%

```

Set the display

```
7140 \@for\@gls@type:=\@glsacronymlists\do{%  
7141   \SetCustomDisplayStyle{\@gls@type}%  
7142 }%  
7143 }
```

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.

```
7144 \RequirePackage{glossary-hypernav}
```

The styles that use list-like environments. These are not loaded if the `nolist` option is used:

```
7145 \@gls@loadlist
```

The styles that use the `longtable` environment. These are not loaded if the `no-long` package option is used.

```
7146 \@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.

```
7147 \@gls@loadsuper
```

The tree-like styles. These are not loaded if the `notree` package option is used.

```
7148 \@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.

```
7149 \ifx\@glossary@default@style\relax  
7150 \else  
7151   \setglossarystyle{\@glossary@default@style}  
7152 \fi
```

1.20 Debugging Commands

`\showgloparent` `\showgloparent{\langle label \rangle}`

```
7153 \newcommand*{\showgloparent}[1]{%  
7154   \expandafter\show\csname glo@glsdetoklabel{#1}@parent\endcsname  
7155 }
```

`\showglolevel` `\showglolevel{\langle label \rangle}`

```
7156 \newcommand*\showglolevel}[1]{%
7157   \expandafter\show\csname glo@\glsdetoklabel{#1}@level\endcsname
7158 }
```

`\showglolevel` `\showglolevel{<label>}`

```
7159 \newcommand*\showglolevel}[1]{%
7160   \expandafter\show\csname glo@\glsdetoklabel{#1}@level\endcsname
7161 }
```

`\showglolevel` `\showglolevel{<label>}`

```
7162 \newcommand*\showglolevel}[1]{%
7163   \expandafter\show\csname glo@\glsdetoklabel{#1}@level\endcsname
7164 }
```

`\showglolevel` `\showglolevel{<label>}`

```
7165 \newcommand*\showglolevel}[1]{%
7166   \expandafter\show\csname glo@\glsdetoklabel{#1}@level\endcsname
7167 }
```

`\showglolevel` `\showglolevel{<label>}`

```
7168 \newcommand*\showglolevel}[1]{%
7169   \expandafter\show\csname glo@\glsdetoklabel{#1}@level\endcsname
7170 }
```

`\showglolevel` `\showglolevel{<label>}`

```
7171 \newcommand*\showglolevel}[1]{%
7172   \expandafter\show\csname glo@\glsdetoklabel{#1}@level\endcsname
7173 }
```

`\showglolevel` `\showglolevel{<label>}`

```
7174 \newcommand*{\showglocounter}[1]{%
7175   \expandafter\show\csname glo@glstetoklabel{#1}@counter\endcsname
7176 }
```

```
\showglouserii \showglouserii{<label>}
```

```
7177 \newcommand*{\showglouserii}[1]{%
7178   \expandafter\show\csname glo@glstetoklabel{#1}@userii\endcsname
7179 }
```

```
\showglouseriii \showglouseriii{<label>}
```

```
7180 \newcommand*{\showglouseriii}[1]{%
7181   \expandafter\show\csname glo@glstetoklabel{#1}@useriii\endcsname
7182 }
```

```
\showglouseriv \showglouseriv{<label>}
```

```
7183 \newcommand*{\showglouseriv}[1]{%
7184   \expandafter\show\csname glo@glstetoklabel{#1}@useriv\endcsname
7185 }
```

```
\showglouserv \showglouserv{<label>}
```

```
7186 \newcommand*{\showglouserv}[1]{%
7187   \expandafter\show\csname glo@glstetoklabel{#1}@userv\endcsname
7188 }
```

```
\showglouservi \showglouservi{<label>}
```

```
7189 \newcommand*{\showglouservi}[1]{%
7190   \expandafter\show\csname glo@glstetoklabel{#1}@uservi\endcsname
7191 }
```



```
7192 \newcommand*{\showglouservi}[1]{%
7193   \expandafter\show\csname glo@\glsdetoklabel{#1}@uservi\endcsname
7194 }
```

`\showgloname` `\showgloname{<label>}`

```
7195 \newcommand*{\showgloname}[1]{%
7196   \expandafter\show\csname glo@\glsdetoklabel{#1}@name\endcsname
7197 }
```

`\showglodesc` `\showglodesc{<label>}`

```
7198 \newcommand*{\showglodesc}[1]{%
7199   \expandafter\show\csname glo@\glsdetoklabel{#1}@desc\endcsname
7200 }
```

`\showglodescplural` `\showglodescplural{<label>}`

```
7201 \newcommand*{\showglodescplural}[1]{%
7202   \expandafter\show\csname glo@\glsdetoklabel{#1}@descplural\endcsname
7203 }
```

`\showglosort` `\showglosort{<label>}`

```
7204 \newcommand*{\showglosort}[1]{%
7205   \expandafter\show\csname glo@\glsdetoklabel{#1}@sort\endcsname
7206 }
```

`\showglosymbol` `\showglosymbol{<label>}`

```
7207 \newcommand*{\showglosymbol}[1]{%
7208   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbol\endcsname
7209 }
```

`\showglosymbolplural` `\showglosymbolplural{<label>}`

```
7210 \newcommand*{\showglosymbolplural}[1]{%
7211   \expandafter\show\csname glo@glstetoklabel{#1}@symbolplural\endcsname
7212 }
```

`\showgloshort` `\showgloshort{<label>}`

```
7213 \newcommand*{\showgloshort}[1]{%
7214   \expandafter\show\csname glo@glstetoklabel{#1}@short\endcsname
7215 }
```

`\showglolong` `\showglolong{<label>}`

```
7216 \newcommand*{\showglolong}[1]{%
7217   \expandafter\show\csname glo@glstetoklabel{#1}@long\endcsname
7218 }
```

`\showgloindex` `\showgloindex{<label>}`

```
7219 \newcommand*{\showgloindex}[1]{%
7220   \expandafter\show\csname glo@glstetoklabel{#1}@index\endcsname
7221 }
```

`\showgloflag` `\showgloflag{<label>}`

```
7222 \newcommand*{\showgloflag}[1]{%
7223   \expandafter\show\csname ifglo@glstetoklabel{#1}@flag\endcsname
7224 }
```

`\showgloclist` `\showgloclist{<label>}`

```
7225 \newcommand*{\showgloclist}[1]{%
7226   \expandafter\show\csname glo@glstetoklabel{#1}@loclist\endcsname
7227 }
```

`\showacronymlists` `\showacronymlists`

Show list of glossaries that have been flagged as a list of acronyms.

```
7228 \newcommand*{\showacronymlists}{%  
7229   \show\@glsacronymlists  
7230 }
```

`\showglossaries` `\showglossaries`

Show list of defined glossaries.

```
7231 \newcommand*{\showglossaries}{%  
7232   \show\@glo@types  
7233 }
```

`\showglossaryin` `\showglossaryin{<glossary-label>}`

Show the 'in' extension for the given glossary.

```
7234 \newcommand*{\showglossaryin}[1]{%  
7235   \expandafter\show\csname @glotype@#1@in\endcsname  
7236 }
```

`\showglossaryout` `\showglossaryout{<glossary-label>}`

Show the 'out' extension for the given glossary.

```
7237 \newcommand*{\showglossaryout}[1]{%  
7238   \expandafter\show\csname @glotype@#1@out\endcsname  
7239 }
```

`\showglossarytitle` `\showglossarytitle{<glossary-label>}`

Show the title for the given glossary.

```
7240 \newcommand*{\showglossarytitle}[1]{%  
7241   \expandafter\show\csname @glotype@#1@title\endcsname  
7242 }
```

`\showglossarycounter` `\showglossarycounter{<glossary-label>}`

Show the counter for the given glossary.

```
7243 \newcommand*{\showglossarycounter}[1]{%  
7244   \expandafter\show\csname @glotype@#1@counter\endcsname  
7245 }
```

```
\showglossaryentries \showglossaryentries{<glossary-label>}
```

Show the list of entry labels for the given glossary.

```
7246 \newcommand*{\showglossaryentries}[1]{%
7247   \expandafter\show\csname glolist@#1\endcsname
7248 }
```

1.21 Compatibility with version 2.07 and below

In order to fix some bugs in v3.0, it was necessary to change the way information is written to the `glo` file, which also meant a change in the format of the Xindy style file. The compatibility option is meant for documents that use a customised Xindy style file with `\noist`. With the compatibility option, hopefully xindy will still be able to process the old document, but the bugs will remain. The issues in versions 2.07 and below:

- With xindy, the counter used by the entry was hard-coded into the Xindy style file. This meant that you couldn't use the counter to swap counters.
- With both xindy and makeindex, if used with hyperref and `\theH<counter>` was different to `\thecounter`, the link in the location number would be undefined.

```
7249 \csname ifglscpatible-2.07\endcsname
7250   \RequirePackage{glossaries-compatible-207}
7251 \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.

```
7252 \NeedsTeXFormat{LaTeX2e}
7253 \ProvidesPackage{glossaries-prefix}[2014/07/30 v4.08 (NLCT)]
```

Pass all options to glossaries:

```
7254 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
7255 \ProcessOptions
```

Load glossaries:

```
7256 \RequirePackage{glossaries}
```

Add the new keys:

```
7257 \define@key{glossentry}{prefixfirst}{\def\@glo@entryprefixfirst{#1}}%
7258 \define@key{glossentry}{prefixfirstplural}{\def\@glo@entryprefixfirstplural{#1}}%
7259 \define@key{glossentry}{prefix}{\def\@glo@entryprefix{#1}}%
7260 \define@key{glossentry}{prefixplural}{\def\@glo@entryprefixplural{#1}}%
```

Add them to \@gls@keymap:

```
7261 \appto\@gls@keymap{,%  
7262   {prefixfirst}{prefixfirst},%  
7263   {prefixfirstplural}{prefixfirstplural},%  
7264   {prefix}{prefix},%  
7265   {prefixplural}{prefixplural}}%  
7266 }
```

Set the default values:

```
7267 \appto\@newglossaryentryprehook{%  
7268   \def\@glo@entryprefix{}}%  
7269   \def\@glo@entryprefixplural{}}%  
7270   \let\@glo@entryprefixfirst\@gls@default@value  
7271   \let\@glo@entryprefixfirstplural\@gls@default@value  
7272 }
```

Set the assignment code:

```
7273 \appto\@newglossaryentryposthook{%  
7274   \gls@assign@field{\@glo@label}{prefix}{\@glo@entryprefix}}%  
7275   \gls@assign@field{\@glo@label}{prefixplural}{\@glo@entryprefixplural}}%
```

If prefixfirst has not been supplied, make it the same as prefix.

```
7276   \expandafter\gls@assign@field\expandafter  
7277     {\csname glo@\@glo@label @prefix\endcsname}{\@glo@label}{prefixfirst}}%  
7278     {\@glo@entryprefixfirst}}%
```

If prefixfirstplural has not been supplied, make it the same as prefixplural.

```
7279   \expandafter\gls@assign@field\expandafter  
7280     {\csname glo@\@glo@label @prefixplural\endcsname}{\@glo@label}}%  
7281     {prefixfirstplural}{\@glo@entryprefixfirstplural}}%  
7282 }
```

Define commands to access these fields:

glsentryprefixfirst

```
7283 \newcommand*\glsentryprefixfirst[1]{\csuse{glo@#1@prefixfirst}}
```

glsentryprefixfirstplural

```
7284 \newcommand*\glsentryprefixfirstplural[1]{\csuse{glo@#1@prefixfirstplural}}
```

\glsentryprefix

```
7285 \newcommand*\glsentryprefix[1]{\csuse{glo@#1@prefix}}
```

\glsentryprefixplural

```
7286 \newcommand*\glsentryprefixplural[1]{\csuse{glo@#1@prefixplural}}
```

Now for the initial upper case variants:

Glsentryprefixfirst

```
7287 \newrobustcmd*\Glsentryprefixfirst[1]{%  
7288   \protected@edef\@glo@text{\csname glo@#1@prefixfirst\endcsname}}%  
7289   \xmakefirstuc\@glo@text  
7290 }
```

ryprefixfirstplural

```
7291 \newrobustcmd*{\Glsentryprefixfirstplural}[1]{%
7292   \protected@edef\@glo@text{\csname glo@#1@prefixfirstplural\endcsname}%
7293   \xmakefirstuc\@glo@text
7294 }
```

\Glsentryprefix

```
7295 \newrobustcmd*{\Glsentryprefix}[1]{%
7296   \protected@edef\@glo@text{\csname glo@#1@prefix\endcsname}%
7297   \xmakefirstuc\@glo@text
7298 }
```

lsentryprefixplural

```
7299 \newrobustcmd*{\Glsentryprefixplural}[1]{%
7300   \protected@edef\@glo@text{\csname glo@#1@prefixplural\endcsname}%
7301   \xmakefirstuc\@glo@text
7302 }
```

Define commands to determine if the prefix keys have been set:

\ifglshasprefix

```
7303 \newcommand*{\ifglshasprefix}[3]{%
7304   \ifcseempty{glo@#1@prefix}%
7305   {#3}%
7306   {#2}%
7307 }
```

fglshasprefixplural

```
7308 \newcommand*{\ifglshasprefixplural}[3]{%
7309   \ifcseempty{glo@#1@prefixplural}%
7310   {#3}%
7311   {#2}%
7312 }
```

ifglshasprefixfirst

```
7313 \newcommand*{\ifglshasprefixfirst}[3]{%
7314   \ifcseempty{glo@#1@prefixfirst}%
7315   {#3}%
7316   {#2}%
7317 }
```

asprefixfirstplural

```
7318 \newcommand*{\ifglshasprefixfirstplural}[3]{%
7319   \ifcseempty{glo@#1@prefixfirstplural}%
7320   {#3}%
7321   {#2}%
7322 }
```

Define commands that insert the prefix before commands like \gls:

\pgls

```
7323 \newrobustcmd{\pgls}{\@gls@hyp@opt\@pgls}
```

\@pgls Unstarred version.

```
7324 \newcommand*{\@pgls}[2][{}]{%
7325   \new@ifnextchar[%
7326     {\@pgls@{#1}{#2}}%
7327     {\@pgls@{#1}{#2}[]}%
7328 }
```

\@pgls@ Read in the final optional argument:

```
7329 \def\@pgls@#1#2[#3]{%
7330   \glsdoifexists{#2}%
7331   {%
7332     \ifglsused{#2}%
7333     {%
7334       \glsentryprefix{#2}%
7335     }%
7336     {%
7337       \glsentryprefixfirst{#2}%
7338     }%
7339     \@gls@{#1}{#2}[#3]%
7340   }%
7341 }
```

Similarly for the plural version:

\pglsp1

```
7342 \newrobustcmd{\pglsp1}{\@gls@hyp@opt\@pglsp1}
```

\@pglsp1 Unstarred version.

```
7343 \newcommand*{\@pglsp1}[2][{}]{%
7344   \new@ifnextchar[%
7345     {\@pglsp1@{#1}{#2}}%
7346     {\@pglsp1@{#1}{#2}[]}%
7347 }
```

\@pglsp1@ Read in the final optional argument:

```
7348 \def\@pglsp1@#1#2[#3]{%
7349   \glsdoifexists{#2}%
7350   {%
7351     \ifglsused{#2}%
7352     {%
7353       \glsentryprefixplural{#2}%
7354     }%
7355     {%
7356       \glsentryprefixfirstplural{#2}%

```

```

7357   }%
7358   \@glspl@{#1}{#2}[#3]%
7359   }%
7360 }

```

Now for the first letter upper case versions:

`\Pgl`

```

7361 \newrobustcmd{\Pgl}{\@gls@hyp@opt\Pgl}

```

`\@Pgl` Unstarred version.

```

7362 \newcommand*{\@Pgl}[2] [] {%
7363   \new@ifnextchar [%
7364     {\@Pgl@{#1}{#2}}%
7365     {\@Pgl@{#1}{#2} []}%
7366 }

```

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

```

7367 \def\@Pgl@#1#2[#3] {%
7368   \glsdoifexists{#2}%
7369   {%
7370     \ifglsused{#2}%
7371     {%
7372       \ifglshasprefix{#2}%
7373       {%
7374         \Glsentryprefix{#2}%
7375         \@gls@{#1}{#2}[#3]%
7376       }%
7377       {\@Gls@{#1}{#2}[#3]}%
7378     }%
7379     {%
7380       \ifglshasprefixfirst{#2}%
7381       {%
7382         \Glsentryprefixfirst{#2}%
7383         \@gls@{#1}{#2}[#3]%
7384       }%
7385       {\@Gls@{#1}{#2}[#3]}%
7386     }%
7387   }%
7388 }

```

Similarly for the plural version:

`\Pglpl`

```

7389 \newrobustcmd{\Pglpl}{\@gls@hyp@opt\Pglpl}

```

`\@Pglpl` Unstarred version.

```

7390 \newcommand*{\@Pglpl}[2] [] {%

```



```

7391 \new@ifnextchar[%
7392 {\@Pglsp1@{#1}{#2}}%
7393 {\@Pglsp1@{#1}{#2} []}%
7394 }

```

\@Pglsp1@ Read in the final optional argument:

```

7395 \def\@Pglsp1@#1#2[#3]{%
7396 \glsdoifexists{#2}%
7397 {%
7398 \ifglsused{#2}%
7399 {%
7400 \ifglshasprefixplural{#2}%
7401 {%
7402 \Glsentryprefixplural{#2}%
7403 \@glspl@{#1}{#2}[#3]%
7404 }%
7405 {\@Glspl@{#1}{#2}[#3]}%
7406 }%
7407 {%
7408 \ifglshasprefixfirstplural{#2}%
7409 {%
7410 \Glsentryprefixfirstplural{#2}%
7411 \@glspl@{#1}{#2}[#3]%
7412 }%
7413 {\@Glspl@{#1}{#2}[#3]}%
7414 }%
7415 }%
7416 }

```

Finally the all upper case versions:

\PGLS

```

7417 \newrobustcmd{\PGLS}{\@gls@hyp@opt\@PGLS}

```

\@PGLS Unstarred version.

```

7418 \newcommand*{\@PGLS}[2] []{%
7419 \new@ifnextchar[%
7420 {\@PGLS@{#1}{#2}}%
7421 {\@PGLS@{#1}{#2} []}%
7422 }

```

\@PGLS@ Read in the final optional argument:

```

7423 \def\@PGLS@#1#2[#3]{%
7424 \glsdoifexists{#2}%
7425 {%
7426 \ifglsused{#2}%
7427 {%
7428 \mfirstucMakeUppercase{\glsentryprefix{#2}}%

```

```

7429 }%
7430 {%
7431     \mfirstucMakeUppercase{\glsentryprefixfirst{#2}}%
7432 }%
7433     \@GLS@{#1}{#2}[#3]%
7434 }%
7435 }

```

Plural version:

`\PGLSp1`

```
7436 \newrobustcmd{\PGLSp1}{\@gls@hyp@opt\PGLSp1}
```

`\@PGLSp1` Unstarred version.

```

7437 \newcommand*{\@PGLSp1}[2][{}]{%
7438     \new@ifnextchar[%
7439     {\@PGLSp1@{#1}{#2}}%
7440     {\@PGLSp1@{#1}{#2}[]}%
7441 }

```

`\@PGLSp1@` Read in the final optional argument:

```

7442 \def\@PGLSp1@#1#2[#3]{%
7443     \glsdoifexists{#2}%
7444     {%
7445         \ifglsused{#2}%
7446         {%
7447             \mfirstucMakeUppercase{\glsentryprefixplural{#2}}%
7448         }%
7449         {%
7450             \mfirstucMakeUppercase{\glsentryprefixfirstplural{#2}}%
7451         }%
7452         \@GLSp1@{#1}{#2}[#3]%
7453     }%
7454 }

```

3 Mfirstuc Documented Code

```

7455 \NeedsTeXFormat{LaTeX2e}
7456 \ProvidesPackage{mfirstuc}[2015/02/03 v1.10 (NLCT)]

```

Requires etoolbox:

```
7457 \RequirePackage{etoolbox}
```

`\makefirstuc` Syntax:

```
\makefirstuc{text}
```

Makes the first letter uppercase, but will skip initial control sequences if they are followed by a group and make the first thing in the group uppercase,

unless the group is empty. Thus `\makefirstuc{abc}` will produce: `Abc`, `\makefirstuc{\ae bc}` will produce: `Æbc`, but `\makefirstuc{\emph{abc}}` will produce `Abc`. This is required by `\Gls` and `\Glspl`.

```

7458 \newif\if@glscs
7459 \newtoks\@glsmfirst
7460 \newtoks\@glsmrest
7461 \newrobustcmd*{\makefirstuc}[1]{%
7462   \def\gls@argi{#1}%
7463   \ifx\gls@argi\@empty

   If the argument is empty, do nothing.
7464   \else

7465     \def\@gls@tmp{\ #1}%
7466     \@onelevel@sanitize\@gls@tmp
7467     \expandafter\@gls@checkcs\@gls@tmp\relax\relax
7468     \if@glscs
7469       \@gls@getbody #1}\@nil
7470       \ifx\@gls@rest\@empty
7471         \glsmakefirstuc{#1}%
7472       \else
7473         \expandafter\@gls@split\@gls@rest\@nil
7474         \ifx\@gls@first\@empty
7475           \glsmakefirstuc{#1}%
7476         \else
7477           \expandafter\@glsmfirst\expandafter{\@gls@first}%
7478           \expandafter\@glsmrest\expandafter{\@gls@rest}%
7479           \edef\@gls@domfirstuc{\noexpand\@gls@body
7480             {\noexpand\glsmakefirstuc\the\@glsmfirst}%
7481             \the\@glsmrest}%
7482           \@gls@domfirstuc
7483         \fi
7484       \fi
7485     \else
7486       \glsmakefirstuc{#1}%
7487     \fi
7488   \fi
7489 }
```

Put first argument in `\@gls@first` and second argument in `\@gls@rest`:

```

7490 \def\@gls@split#1#2\@nil{%
7491   \def\@gls@first{#1}\def\@gls@rest{#2}%
7492 }

7493 \def\@gls@checkcs#1 #2#3\relax{%
7494   \def\@gls@argi{#1}\def\@gls@argii{#2}%
7495   \ifx\@gls@argi\@gls@argii
7496     \@glscstrue
7497   \else
7498     \@glscsfalse
```

```
7499 \fi
7500 }
```

`\@gls@makefirstuc` Make first thing upper case:

```
7501 \def\@gls@makefirstuc#1{\mfirstucMakeUppercase #1}
```

`mfirstucMakeUppercase` Allow user to replace `\MakeUppercase` with another case changing command.

```
7502 \newcommand*\mfirstucMakeUppercase{\MakeUppercase}
```

`\glsmakefirstuc` Provide a user command to make it easier to customise.

```
7503 \newcommand*\glsmakefirstuc}[1]{\@gls@makefirstuc{#1}}
```

Get the first grouped argument and store in `\@gls@body`.

```
7504 \def\@gls@getbody#1#\def\@gls@body{#1}\@gls@gobbletonil}
```

Scoup up everything to `\@nil` and store in `\@gls@rest`:

```
7505 \def\@gls@gobbletonil#1\@nil{\def\@gls@rest{#1}}
```

`\xmakefirstuc` Expand argument once before applying `\makefirstuc` (added v1.01).

```
7506 \newcommand*\xmakefirstuc}[1]{%
```

```
7507 \expandafter\makefirstuc\expandafter{#1}}
```

`\emakefirstuc` Fully expand argument before applying `\makefirstuc`

```
7508 \DeclareRobustCommand*\emakefirstuc}[1]{%
```

```
7509 \protected@edef\@MFU@caparg{#1}%
```

```
7510 \expandafter\makefirstuc\expandafter{\@MFU@caparg}%
```

```
7511 }
```

`\capitalisewords` Capitalise each word in the argument. Words are considered to be separated by plain spaces (i.e. non-breakable spaces won't be considered a word break).

```
7512 \newrobustcmd*\capitalisewords}[1]{%
```

```
7513 \def\gls@add@space{ }%
```

```
7514 \let\@mfu@domakefirstuc\makefirstuc
```

```
7515 \let\@mfu@checkword\@gobble
```

```
7516 \mfu@capitalisewords#1 \@nil\mfu@endcap
```

```
7517 }
```

```
7518 \def\mfu@capitalisewords#1 #2\mfu@endcap{%
```

```
7519 \def\mfu@cap@first{#1}%
```

```
7520 \def\mfu@cap@second{#2}%
```

```
7521 \gls@add@space
```

```
7522 \@mfu@checkword{#1}%
```

```
7523 \@mfu@domakefirstuc{#1}%
```

```
7524 \def\gls@add@space{ }%
```

```
7525 \ifx\mfu@cap@second\@nnil
```

```
7526 \let\next@mfu@cap\mfu@noop
```

```
7527 \else
```

```
7528 \let\next@mfu@cap\mfu@capitalisewords
```

```
7529 \let\@mfu@checkword\mfu@checkword
```

```

7530 \fi
7531 \next@mfu@cap#2\mfu@endcap
7532 }
7533 \def\mfu@noop#1\mfu@endcap{}

```

`\mfu@checkword` Check if word should be capitalised.

```

7534 \newcommand*\mfu@checkword[1]{%
7535 \ifinlist{#1}{\@mfu@nocaplist}%
7536 {%
7537 \let\@mfu@domakefirstuc\@firstofone
7538 }%
7539 {%
7540 \let\@mfu@domakefirstuc\makefirstuc
7541 }%
7542 }

```

`\@mfu@nocaplist` List of words that shouldn't be capitalised.

```

7543 \newcommand*\@mfu@nocaplist{}

```

`\MFUnocap` Provide the user with a means to add a word to the list.

```

7544 \newcommand*\MFUnocap[1]{\listadd{\@mfu@nocaplist}{#1}}

```

`\gMFUnocap` Global version.

```

7545 \newcommand*\gMFUnocap[1]{\listgadd{\@mfu@nocaplist}{#1}}

```

`\MFUclear` Clear the list

```

7546 \newcommand*\MFUclear{\renewcommand*\@mfu@nocaplist{}}

```

`\xcapitalisewords` Short-cut command:

```

7547 \newcommand*\xcapitalisewords[1]{%
7548 \expandafter\capitalisewords\expandafter{#1}%
7549 }

```

`\ecapitalisewords` Fully expand argument before applying `\capitalisewords`

```

7550 \DeclareRobustCommand*\ecapitalisewords[1]{%
7551 \protected@edef\@MFU@caparg{#1}%
7552 \expandafter\capitalisewords\expandafter{\@MFU@caparg}%
7553 }

```

4 Mfirstuc-english Documented Code

```

7554 \NeedsTeXFormat{LaTeX2e}
7555 \ProvidesPackage{mfirstuc-english}[2014/07/30 v1.0 (NLCT)]

```

Load mfirstuc if not already loaded:

```

7556 \RequirePackage{mfirstuc}

```

Add no-cap words. (List isn't a complete list.)

```

7557 \MFUnocap{a}
7558 \MFUnocap{an}
7559 \MFUnocap{and}
7560 \MFUnocap{but}
7561 \MFUnocap{for}
7562 \MFUnocap{in}
7563 \MFUnocap{of}
7564 \MFUnocap{or}
7565 \MFUnocap{no}
7566 \MFUnocap{nor}
7567 \MFUnocap{so}
7568 \MFUnocap{some}
7569 \MFUnocap{the}
7570 \MFUnocap{with}
7571 \MFUnocap{yet}

```

5 Glossary Styles

5.1 Glossary hyper-navigation definitions (glossary-hypernav package)

Package Definition:

```
7572 \ProvidesPackage{glossary-hypernav}[2013/11/14 v4.0 (NLCT)]
```

The commands defined in this package are provided to help navigate around the groups within a glossary (see [subsection 1.16.](#)) `\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`

```

7573 \newcommand*{\glsnavhyperlink}[3][\@glo@type]{%
7574   \edef\gls@grplabel{#2}\protected@edef\gls@grptitle{#3}%
7575   \@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`

```

7576 \newcommand*{\glsnavhypertarget}[3][\@glo@type]{%
  Add this group to the aux file for re-run check.
7577   \protected@write\@auxout{}{\string\@gls@hypergroup{#1}{#2}}%

```

Add the target.

```
7578 \@gls@target{glsn:#1@#2}#{3}%
```

Check list of know groups to determine if a re-run is required.

```
7579 \expandafter\let
```

```
7580 \expandafter\@gls@list\csname @gls@hypergroup@#1\endcsname
```

Iterate through list and terminate loop if this group is found.

```
7581 \@for\@gls@elem:=\@gls@list\do{%
```

```
7582 \ifthenelse{\equal{\@gls@elem}#{2}}{\@endfortrue}{}%}
```

Check if list terminated prematurely.

```
7583 \if@endfor
```

```
7584 \else
```

This group was not included in the list, so issue a warning.

```
7585 \GlossariesWarningNoLine{Navigation panel
```

```
7586 for glossary type '#1'^^Jmissing group '#2'}%
```

```
7587 \gdef\gls@hypergroup@rerun{%
```

```
7588 \GlossariesWarningNoLine{Navigation panel
```

```
7589 has changed. Rerun LaTeX}}%
```

```
7590 \fi
```

```
7591 }
```

`\gls@hypergroup@rerun` Give a warning at the end if re-run required

```
7592 \let\gls@hypergroup@relax
```

```
7593 \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.

```
7594 \newcommand*{\@gls@hypergroup}[2]{%
```

```
7595 \@ifundefined{@gls@hypergroup@list@#1}{%
```

```
7596 \expandafter\xdef\csname @gls@hypergroup@list@#1\endcsname{#2}%
```

```
7597 }{%
```

```
7598 \expandafter\let\expandafter\@gls@tmp
```

```
7599 \csname @gls@hypergroup@list@#1\endcsname
```

```
7600 \expandafter\xdef\csname @gls@hypergroup@list@#1\endcsname{%
```

```
7601 \@gls@tmp,#2}%
```

```
7602 }%
```

```
7603 }
```

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`

```
7604 \newcommand*{\glsnavigation}{%
7605 \def\@gls@between{}%
7606 \@ifundefined{gls@hypergroup@list@{glo@type}}{%
7607 \def\@gls@list{}%
7608 }%
7609 \expandafter\let\expandafter\@gls@list
7610 \csname @gls@hypergroup@list@{glo@type}\endcsname
7611 }%
7612 \@for\@gls@tmp:=\@gls@list\do{%
7613 \@gls@between

7614 \@gls@getgrouptitle{\@gls@tmp}{\@gls@grptitle}%
7615 \glsnavhyperlink{\@gls@tmp}{\@gls@grptitle}%
7616 \let\@gls@between\gls@hypernavsep%
7617 }%
7618 }
```

`\gls@hypernavsep` Separator for the hyper navigation bar.

```
7619 \newcommand*{\gls@hypernavsep}{\space\textbar\space}
```

The `\gls@symbolnav` 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.

`\gls@symbolnav`

```
7620 \newcommand*{\gls@symbolnav}{%
7621 \glsnavhyperlink{gls@symbols}{\gls@getgrouptitle{gls@symbols}}%
7622 \gls@hypernavsep
7623 \glsnavhyperlink{gls@numbers}{\gls@getgrouptitle{gls@numbers}}%
7624 \gls@hypernavsep
7625 }
```

5.2 In-line Style (glossary-inline.sty)

This defines an in-line style where the entries are comma-separated with just the name and description displayed.

```
7626 \ProvidesPackage{glossary-inline}[2013/11/14 v4.0 (NLCT)]
```

`inline` Define the inline style.

```
7627 \newglossarystyle{inline}{%
```

Start of glossary sets up first empty separator between entries. (This is then changed by `\gls@entry`)

```
7628 \renewenvironment{theglossary}%
7629 {%
7630 \def\@gls@inlinesep{}%
7631 \def\@gls@inlinesubsep{}%
7632 \def\@gls@inlinepostchild{}%
```



```
7633 }%
7634 {\glspostinline}%
```

No header:

```
7635 \renewcommand*\glossaryheader{}
```

No group headings (if heading is required, add `\glsinlinedopostchild` to start definition in case heading follows a child entry):

```
7636 \renewcommand*\glsgroupheading[1]{}
```

Just display separator followed by name and description:

```
7637 \renewcommand{\glossentry}[2]{%
7638   \glsinlinedopostchild
7639   \gls@inlinesep
7640   \glsentryitem{##1}%
7641   \glsinlinenameformat{##1}{%
7642     \glossentryname{##1}%
7643   }%
7644   \ifglsdescsuppressed{##1}%
7645   {%
7646     \glsinlineemptydescformat
7647     {%
7648       \glossentrysymbol{##1}%
7649     }%
7650     {%
7651       ##2%
7652     }%
7653   }%
7654   {%
7655     \ifglsdesc{##1}%
7656     {\glsinlinedescformat{\glossentrydesc{##1}}{\glossentrysymbol{##1}}{##2}}%
7657     {\glsinlineemptydescformat{\glossentrysymbol{##1}}{##2}}%
7658   }%
7659   \ifglsdesc{##1}%
7660   {%
7661     \glsresetsubentrycounter
7662     \glsinlineparentchildseparator
7663     \def\gls@inlinesubsep{}%
7664     \def\gls@inlinepostchild{\glsinlinepostchild}%
7665   }%
7666   {}%
7667   \def\gls@inlinesep{\glsinlineseparator}%
7668 }%
```

Sub-entries display description:

```
7669 \renewcommand{\subglossentry}[3]{%
7670   \gls@inlinesubsep%
7671   \glsinlinesubnameformat{##2}{%
7672     \glossentryname{##2}}%
7673   \glsentryitem{##2}%
7674   \glsinlinesubdescformat{\glossentrydesc{##2}}{\glossentrysymbol{##2}}{##3}%
```

```
7675 \def\gls@inlinesubsep{\glsinlinesubseparator}%
7676 }%
```

Nothing special between groups:

```
7677 \renewcommand*\glsgroupskip{}%
7678 }
```

`\glsinlinedopostchild`

```
7679 \newcommand*\glsinlinedopostchild{%
7680 \gls@inlinepostchild
7681 \def\gls@inlinepostchild{}%
7682 }
```

`\glsinlineseparator` Separator to use between entries.

```
7683 \newcommand*\glsinlineseparator}{;\space}
```

`\glsinlinesubseparator` Separator to use between sub-entries.

```
7684 \newcommand*\glsinlinesubseparator}{,\space}
```

`\glsinlinenparentchildseparator` Separator to use between parent and children.

```
7685 \newcommand*\glsinlinenparentchildseparator}{:\space}
```

`\glsinlinepostchild` Hook to use between child and next entry

```
7686 \newcommand*\glsinlinepostchild{}
```

`\glspostinline` Terminator for inline glossary.

```
7687 \newcommand*\glspostinline}{\glspostdescription\space}
```

`\glsinlinenameformat` Formats the name of the entry (first argument label, second argument name):

```
7688 \newcommand*\glsinlinenameformat}[2]{\glstarget{#1}{#2}}
```

`\glsinlinedescformat` Formats the entry's description, symbol and location list:

```
7689 \newcommand*\glsinlinedescformat}[3]{\space#1}
```

`\glsinlineemptydescformat` Formats the entry's symbol and location list when the description is empty:

```
7690 \newcommand*\glsinlineemptydescformat}[2]{}
```

`\glsinlinesubnameformat` Formats the name of the subentry (first argument label, second argument name):

```
7691 \newcommand*\glsinlinesubnameformat}[2]{\glstarget{#1}{}}
```

`\glsinlinesubdescformat` Formats the subentry's description, symbol and location list:

```
7692 \newcommand*\glsinlinesubdescformat}[3]{#1}
```

5.3 List Style (glossary-list.sty)

The style file defines glossary styles that use the description environment. Note that since the entry name is placed in the optional argument to the `\item` command, it will appear in a bold font by default.

```
7693 \ProvidesPackage{glossary-list}[2015/02/03 v4.13 (NLCT)]
```

`\indexspace` The are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```
7694 \providecommand{\indexspace}{%
7695   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
7696 }
```

`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.

```
7697 \newglossarystyle{list}{%
```

Use description environment:

```
7698   \renewenvironment{theglossary}{%
7699     {\begin{description}}{\end{description}}%
```

No header at the start of the environment:

```
7700   \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
7701   \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries start a new item in the list:

```
7702   \renewcommand*{\glossentry}[2]{%
7703     \item[\glsentryitem{##1}%
7704       \glstarget{##1}{\glossentryname{##1}}]
7705     \glossentrydesc{##1}\glspostdescription\space ##2}%
```

Sub-entries continue on the same line:

```
7706   \renewcommand*{\subglossentry}[3]{%
7707     \glssubentryitem{##2}%
7708     \glstarget{##2}{\strut}%
7709     \glossentrydesc{##2}\glspostdescription\space ##3.}%
7710 %   \end{macrocode}
```

7711 % Add vertical space between groups:

```
7712 %\changes{3.03}{2012/09/21}{added check for glsnogroupskip}
```

```
7713 %   \begin{macrocode}
```

```
7714   \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
```

```
7715 }
```

`listgroup` The listgroup style is like the list style, but the glossary groups have headings.

```
7716 \newglossarystyle{listgroup}{%
```

Base it on the list style:

```
7717 \setglossarystyle{list}%
```

Each group has a heading:

```
7718 \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.

```
7719 \newglossarystyle{listhypergroup}{%
```

Base it on the list style:

```
7720 \setglossarystyle{list}%
```

Add navigation links at the start of the environment:

```
7721 \renewcommand*{\glossaryheader}{%
```

```
7722 \item[\glsnavigation]}%
```

Each group has a heading with a hypertarget:

```
7723 \renewcommand*{\glsgroupheading}[1]{%
```

```
7724 \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.

```
7725 \newglossarystyle{altlist}{%
```

Base it on the list style:

```
7726 \setglossarystyle{list}%
```

Main (level 0) entries start a new item in the list with a line break after the entry name:

```
7727 \renewcommand*{\glossentry}[2]{%
```

```
7728 \item[\glsentryitem{##1}%
```

```
7729 \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.

```
7730 \mbox{}\par\nobreak\@afterheading
```

```
7731 \glossentrydesc{##1}\glspostdescription\space ##2}%
```

Sub-entries start a new paragraph:

```
7732 \renewcommand{\subglossentry}[3]{%
```

```
7733 \par
```

```
7734 \glssubentryitem{##2}%
```

```
7735 \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space ##3}%
```

```
7736 }
```

altlistgroup The altlistgroup glossary style is like the altlist style, but the glossary groups have headings.

```
7737 \newglossarystyle{altlistgroup}{%
```

Base it on the altlist style:

```
7738 \setglossarystyle{altlist}%
```

Each group has a heading:

```
7739 \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.

```
7740 \newglossarystyle{altlisthypergroup}{%
```

Base it on the altlist style:

```
7741 \setglossarystyle{altlist}%
```

Add navigation links at the start of the environment:

```
7742 \renewcommand*{\glossaryheader}{%
```

```
7743 \item[\glsnavigation]}%
```

Each group has a heading with a hypertarget:

```
7744 \renewcommand*{\glsgroupheading}[1]{%
```

```
7745 \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.

```
7746 \newglossarystyle{listdotted}{%
```

Base it on the list style:

```
7747 \setglossarystyle{list}%
```

Each main (level 0) entry starts a new item:

```
7748 \renewcommand*{\glossentry}[2]{%
```

```
7749 \item[]\makebox[\glslistdottedwidth][1]{%
```

```
7750 \glsentryitem{##1}%
```

```
7751 \glstarget{##1}{\glossentryname{##1}}%
```

```
7752 \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##1}}%
```

Sub entries have the same format as main entries:

```
7753 \renewcommand*{\subglossentry}[3]{%
```

```
7754 \item[]\makebox[\glslistdottedwidth][1]{%
```

```
7755 \glssubentryitem{##2}%
```

```
7756 \glstarget{##2}{\glossentryname{##2}}%
```

```
7757 \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##2}}%
```

```
7758 }
```

`\glslistdottedwidth`

```
7759 \newlength\glslistdottedwidth
```

```
7760 \setlength{\glslistdottedwidth}{.5\hsize}
```

`sublistdotted` This style is similar to the `glostylelistdotted` style, except that the main entries just have the name displayed.

```

7761 \newglossarystyle{sublistdotted}{%
    Base it on the listdotted style:
7762 \setglossarystyle{listdotted}%
    Main (level 0) entries just display the name:
7763 \renewcommand*{\glossentry}[2]{%
7764 \item[\glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}}]}%
7765 }
```

5.4 Glossary Styles using `longtable` (the `glossary-long` package)

The glossary styles defined in the package used the `longtable` environment in the glossary.

```

7766 \ProvidesPackage{glossary-long}[2013/11/14 v4.0 (NLCT)]
    Requires the package:
7767 \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`.)

```

7768 \@ifundefined{glsdescwidth}{%
7769 \newlength\glsdescwidth
7770 \setlength{\glsdescwidth}{0.6\hsize}
7771 }{}
```

`\glspagelistwidth` This is a length that governs the width of the page list column.

```

7772 \@ifundefined{glspagelistwidth}{%
7773 \newlength\glspagelistwidth
7774 \setlength{\glspagelistwidth}{0.1\hsize}
7775 }{}
```

`long` The long glossary style command which uses the `longtable` environment:

```

7776 \newglossarystyle{long}{%
    Use longtable with two columns:
7777 \renewenvironment{theglossary}%
7778 {\begin{longtable}[lp{\glsdescwidth}]}%
7779 {\end{longtable}}%
    Do nothing at the start of the environment:
7780 \renewcommand*{\glossaryheader}{}%
    No heading between groups:
7781 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries displayed in a row:

```
7782 \renewcommand{\glossentry}[2]{%
7783   \glstarget{##1}\glstarget{##1}{\glossentryname{##1}} &
7784   \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline
7785 }%
```

Sub entries displayed on the following row without the name:

```
7786 \renewcommand{\subglossentry}[3]{%
7787   &
7788   \glssubentryitem{##2}%
7789   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
7790   ##3\tabularnewline
7791 }%
```

Blank row between groups:

```
7792 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else &
7793 \tabularnewline\fi}%
7794 }
```

longborder The longborder style is like the above, but with horizontal and vertical lines:

```
7795 \newglossarystyle{longborder}{%
```

Base it on the glostylelong style:

```
7796 \setglossarystyle{long}%
```

Use longtable with two columns with vertical lines between each column:

```
7797 \renewenvironment{theglossary}{%
7798   \begin{longtable}[|l|p{\glsdescwidth}|]{\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
7799 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
7800 }
```

longheader The longheader style is like the long style but with a header:

```
7801 \newglossarystyle{longheader}{%
```

Base it on the glostylelong style:

```
7802 \setglossarystyle{long}%
```

Set the table's header:

```
7803 \renewcommand*{\glossaryheader}{%
7804   \bfseries \entryname & \bfseries \descriptionname\tabularnewline\endhead}%
7805 }
```

longheaderborder The longheaderborder style is like the long style but with a header and border:

```
7806 \newglossarystyle{longheaderborder}{%
```

Base it on the glostylelongborder style:

```
7807 \setglossarystyle{longborder}%
```

Set the table's header and add horizontal line to table's foot:

```
7808 \renewcommand*{\glossaryheader}{%
7809 \hline\bfseries \entryname & \bfseries
7810 \descriptionname\tabularnewline\hline
7811 \endhead
7812 \hline\endfoot}%
7813 }
```

`long3col` The `long3col` style is like `long` but with 3 columns

```
7814 \newglossarystyle{long3col}{%
```

Use a `longtable` with 3 columns:

```
7815 \renewenvironment{theglossary}%
7816 {\begin{longtable}{lp{\glsdescwidth}p{\glspagelistwidth}}}%
7817 {\end{longtable}}%
```

No table header:

```
7818 \renewcommand*{\glossaryheader}{}%
```

No headings between groups:

```
7819 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
7820 \renewcommand{\glosentry}[2]{%
7821 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
7822 \glossentrydesc{##1} & ##2\tabularnewline
7823 }%
```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```
7824 \renewcommand{\subglosentry}[3]{%
7825 &
7826 \glssubentryitem{##2}%
7827 \glstarget{##2}{\strut}\glossentrydesc{##2} &
7828 ##3\tabularnewline
7829 }%
```

Blank row between groups:

```
7830 \renewcommand*{\glsgroupskip}{%
7831 \ifglsnogroupskip\else & &\tabularnewline\fi}%
7832 }
```

`long3colborder` The `long3colborder` style is like the `long3col` style but with a border:

```
7833 \newglossarystyle{long3colborder}{%
```

Base it on the `glostylelong3col` style:

```
7834 \setglossarystyle{long3col}%
```

Use a `longtable` with 3 columns with vertical lines around them:

```
7835 \renewenvironment{theglossary}%
7836 {\begin{longtable}{|l|p{\glsdescwidth}|p{\glspagelistwidth}|}%
7837 {\end{longtable}}%
```


Place horizontal lines at the head and foot of the table:

```
7838 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
7839 }
```

`long3colheader` The `long3colheader` style is like `long3col` but with a header row:

```
7840 \newglossarystyle{long3colheader}{%
Base it on the glostylelong3col style:
7841 \setglossarystyle{long3col}%
Set the table's header:
7842 \renewcommand*{\glossaryheader}{%
7843 \bfseries\entryname&\bfseries\descriptionname&
7844 \bfseries\pagelistname\tabularnewline\endhead}%
7845 }
```

`long3colheaderborder` The `long3colheaderborder` style is like the above but with a border

```
7846 \newglossarystyle{long3colheaderborder}{%
Base it on the glostylelong3colborder style:
7847 \setglossarystyle{long3colborder}%
Set the table's header and add horizontal line at table's foot:
7848 \renewcommand*{\glossaryheader}{%
7849 \hline
7850 \bfseries\entryname&\bfseries\descriptionname&
7851 \bfseries\pagelistname\tabularnewline\hline\endhead
7852 \hline\endfoot}%
7853 }
```

`long4col` The `long4col` style has four columns where the third column contains the value of the associated symbol key.

```
7854 \newglossarystyle{long4col}{%
Use a longtable with 4 columns:
7855 \renewenvironment{theglossary}%
7856 {\begin{longtable}{l111}}%
7857 {\end{longtable}}%
No table header:
7858 \renewcommand*{\glossaryheader}{}%
No group headings:
7859 \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):
7860 \renewcommand{\glossentry}[2]{%
7861 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
7862 \glossentrydesc{##1} &
7863 \glossentrysymbol{##1} &
7864 ##2\tabularnewline
7865 }%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
7866 \renewcommand{\subglossentry}[3]{%
7867     &
7868     \glssubentryitem{##2}%
7869     \glstarget{##2}{\strut}\glossentrydesc{##2} &
7870     \glossentrysymbol{##2} & ##3\tabularnewline
7871 }%
```

Blank row between groups:

```
7872 \renewcommand*{\glsgroupskip}{%
7873     \ifglsnogroupskip\else & & \tabularnewline\fi}%
7874 }
```

`long4colheader` The `long4colheader` style is like `long4col` but with a header row.

```
7875 \newglossarystyle{long4colheader}{%
    Base it on the glostylelong4col style:
7876     \setglossarystyle{long4col}%
    Table has a header:
7877     \renewcommand*{\glossaryheader}{%
7878         \bfseries\entryname&\bfseries\descriptionname&
7879         \bfseries \symbolname&
7880         \bfseries\pagelistname\tabularnewline\endhead}%
7881 }
```

`long4colborder` The `long4colborder` style is like `long4col` but with a border.

```
7882 \newglossarystyle{long4colborder}{%
    Base it on the glostylelong4col style:
7883     \setglossarystyle{long4col}%
    Use a longtable with 4 columns surrounded by vertical lines:
7884     \renewenvironment{theglossary}%
7885         {\begin{longtable}{|l|l|l|l|}}%
7886         {\end{longtable}}%
    Add horizontal lines to the head and foot of the table:
7887     \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
7888 }
```

`long4colheaderborder` The `long4colheaderborder` style is like the above but with a border.

```
7889 \newglossarystyle{long4colheaderborder}{%
    Base it on the glostylelong4col style:
7890     \setglossarystyle{long4col}%
    Use a longtable with 4 columns surrounded by vertical lines:
7891     \renewenvironment{theglossary}%
7892         {\begin{longtable}{|l|l|l|l|}}%
7893         {\end{longtable}}%
```

Add table header and horizontal line at the table's foot:

```
7894 \renewcommand*{\glossaryheader}{%
7895 \hline\bfseries\entryname&\bfseries\descriptionname&
7896 \bfseries \symbolname&
7897 \bfseries\pagelistname\tabularnewline\hline\endhead
7898 \hline\endfoot}%
7899 }
```

`altlong4col` The `altlong4col` style is like the `long4col` style but can have multiline descriptions and page lists.

```
7900 \newglossarystyle{altlong4col}{%
```

Base it on the `glostylelong4col` style:

```
7901 \setglossarystyle{long4col}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
7902 \renewenvironment{theglossary}%
7903 {\begin{longtable}{lp{\glsdescwidth}lp{\glspagelistwidth}}%
7904 {\end{longtable}}%
7905 }
```

`altlong4colheader` The `altlong4colheader` style is like `altlong4col` but with a header row.

```
7906 \newglossarystyle{altlong4colheader}{%
```

Base it on the `glostylelong4colheader` style:

```
7907 \setglossarystyle{long4colheader}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
7908 \renewenvironment{theglossary}%
7909 {\begin{longtable}{lp{\glsdescwidth}lp{\glspagelistwidth}}%
7910 {\end{longtable}}%
7911 }
```

`altlong4colborder` The `altlong4colborder` style is like `altlong4col` but with a border.

```
7912 \newglossarystyle{altlong4colborder}{%
```

Base it on the `glostylelong4colborder` style:

```
7913 \setglossarystyle{long4colborder}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
7914 \renewenvironment{theglossary}%
7915 {\begin{longtable}{|lp{\glsdescwidth}|lp{\glspagelistwidth}|}%
7916 {\end{longtable}}%
7917 }
```

`altlong4colheaderborder` The `altlong4colheaderborder` style is like the above but with a header as well as a border.

```
7918 \newglossarystyle{altlong4colheaderborder}{%
```

Base it on the `glostylelong4colheaderborder` style:

```
7919 \setglossarystyle{long4colheaderborder}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
7920 \renewenvironment{theglossary}%  
7921   {\begin{longtable}{|l|p{\glsdescwidth}|l|p{\glspagelistwidth}|}%  
7922   {\end{longtable}}%  
7923 }
```

5.5 Glossary Styles using `longtable` (the `glossary-longragged` package)

The glossary styles defined in the package used the `longtable` environment in the glossary and use ragged right formatting for the multiline columns.

```
7924 \ProvidesPackage{glossary-longragged}[2014/07/30 v4.08 (NLCT)]
```

Requires the package:

```
7925 \RequirePackage{array}
```

Requires the package:

```
7926 \RequirePackage{longtable}
```

`\glsdescwidth` This is a length that governs the width of the description column. This may have already been defined.

```
7927 \@ifundefined{glsdescwidth}{%  
7928   \newlength{glsdescwidth}  
7929   \setlength{glsdescwidth}{0.6\hsize}  
7930 }{}
```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```
7931 \@ifundefined{glspagelistwidth}{%  
7932   \newlength{glspagelistwidth}  
7933   \setlength{glspagelistwidth}{0.1\hsize}  
7934 }{}
```

`longragged` The `longragged` glossary style is like the `long` but uses ragged right formatting for the description column.

```
7935 \newglossarystyle{longragged}{%
```

Use `longtable` with two columns:

```
7936 \renewenvironment{theglossary}%  
7937   {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}}%  
7938   {\end{longtable}}%
```

Do nothing at the start of the environment:

```
7939 \renewcommand*{\glossaryheader}{}%
```

No heading between groups:

```
7940 \renewcommand*\glsgroupheading}[1]{%
```

Main (level 0) entries displayed in a row:

```
7941 \renewcommand{\glossentry}[2]{%
7942   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
7943   \glossentrydesc{##1}\glspostdescription\space ##2%
7944   \tabularnewline
7945 }%
```

Sub entries displayed on the following row without the name:

```
7946 \renewcommand{\subglossentry}[3]{%
7947   &
7948   \glssubentryitem{##2}%
7949   \glstarget{##2}{\strut}\glossentrydesc{##2}%
7950   \glspostdescription\space ##3%
7951   \tabularnewline
7952 }%
```

Blank row between groups:

```
7953 \renewcommand*\glsgroupskip{\ifglsnogroupskip\else & \tabularnewline\fi}%
7954 }
```

`longraggedborder` The `longraggedborder` style is like the above, but with horizontal and vertical lines:

```
7955 \newglossarystyle{longraggedborder}{%
```

Base it on the `glostylelongragged` style:

```
7956 \setglossarystyle{longragged}{%
```

Use `longtable` with two columns with vertical lines between each column:

```
7957 \renewenvironment{theglossary}{%
7958   \begin{longtable}[1|>{\raggedright}p{\glsdescwidth}|]}%
7959   {\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
7960 \renewcommand*\glossaryheader{\hline\endhead\hline\endfoot}%
7961 }
```

`longraggedheader` The `longraggedheader` style is like the `longragged` style but with a header:

```
7962 \newglossarystyle{longraggedheader}{%
```

Base it on the `glostylelongragged` style:

```
7963 \setglossarystyle{longragged}{%
```

Set the table's header:

```
7964 \renewcommand*\glossaryheader{%
7965   \bfseries \entryname & \bfseries \descriptionname
7966   \tabularnewline\endhead}%
7967 }
```

`longraggedheaderborder` The `longraggedheaderborder` style is like the `longragged` style but with a header and border:

```
7968 \newglossarystyle{longraggedheaderborder}{%
```

Base it on the `glostylelongraggedborder` style:

```
7969 \setglossarystyle{longraggedborder}{%
```

Set the table's header and add horizontal line to table's foot:

```
7970 \renewcommand*{\glossaryheader}{%
7971 \hline\bfseries \entryname & \bfseries \descriptionname
7972 \tabularnewline\hline
7973 \endhead
7974 \hline\endfoot}%
7975 }
```

`longragged3col` The `longragged3col` style is like `longragged` but with 3 columns

```
7976 \newglossarystyle{longragged3col}{%
```

Use a `longtable` with 3 columns:

```
7977 \renewenvironment{theglossary}{%
7978 \begin{longtable}[1>{\raggedright}p{\glsdescwidth}%
7979 >{\raggedright}p{\glspagelistwidth}}{%
7980 \end{longtable}}%
```

No table header:

```
7981 \renewcommand*{\glossaryheader}{}%
```

No headings between groups:

```
7982 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
7983 \renewcommand{\glossentry}[2]{%
7984 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
7985 \glossentrydesc{##1} & ##2\tabularnewline
7986 }%
```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```
7987 \renewcommand{\subglossentry}[3]{%
7988 &
7989 \glssubentryitem{##2}%
7990 \glstarget{##2}{\strut}\glossentrydesc{##2} &
7991 ##3\tabularnewline
7992 }%
```

Blank row between groups:

```
7993 \renewcommand*{\glsgroupskip}{%
7994 \ifglsnogroupskip\else & &\tabularnewline\fi}%
7995 }
```

`longragged3colborder` The `longragged3colborder` style is like the `longragged3col` style but with a border:

```
7996 \newglossarystyle{longragged3colborder}{%
```

Base it on the `glostylelongragged3col` style:

```
7997 \setglossarystyle{longragged3col}{%
```

Use a `longtable` with 3 columns with vertical lines around them:

```
7998 \renewenvironment{theglossary}{%
```

```
7999   {\begin{longtable}{|l|>{\raggedright}p{\glsdescwidth}}|%
```

```
8000     >{\raggedright}p{\glspagelistwidth}}|}%
```

```
8001   {\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
8002 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
```

```
8003 }
```

`longragged3colheader` The `longragged3colheader` style is like `longragged3col` but with a header row:

```
8004 \newglossarystyle{longragged3colheader}{%
```

Base it on the `glostylelongragged3col` style:

```
8005 \setglossarystyle{longragged3col}{%
```

Set the table's header:

```
8006 \renewcommand*{\glossaryheader}{%
```

```
8007   \bfseries\entryname&\bfseries\descriptionname&
```

```
8008   \bfseries\pagelistname\tabularnewline\endhead}%
```

```
8009 }
```

`longragged3colheaderborder` The `longragged3colheaderborder` style is like the above but with a border

```
8010 \newglossarystyle{longragged3colheaderborder}{%
```

Base it on the `glostylelongragged3colborder` style:

```
8011 \setglossarystyle{longragged3colborder}{%
```

Set the table's header and add horizontal line at table's foot:

```
8012 \renewcommand*{\glossaryheader}{%
```

```
8013   \hline
```

```
8014   \bfseries\entryname&\bfseries\descriptionname&
```

```
8015   \bfseries\pagelistname\tabularnewline\hline\endhead
```

```
8016   \hline\endfoot}%
```

```
8017 }
```

`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.

```
8018 \newglossarystyle{altlongragged4col}{%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8019 \renewenvironment{theglossary}{%
```

```

8020   {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}l%
8021     >{\raggedright}p{\glspagelistwidth}}}%
8022   {\end{longtable}}%

```

No table header:

```
8023   \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8024   \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):

```

8025   \renewcommand{\glossentry}[2]{%
8026     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8027     \glossentrydesc{##1} & \glossentrysymbol{##1} &
8028     ##2\tabularnewline
8029   }%

```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```

8030   \renewcommand{\subglossentry}[3]{%
8031     &
8032     \glssubentryitem{##2}%
8033     \glstarget{##2}{\strut}\glossentrydesc{##2} &
8034     \glossentrysymbol{##2} & ##3\tabularnewline
8035   }%

```

Blank row between groups:

```

8036   \renewcommand*{\glsgroupskip}{}%
8037   \ifglsnogroupskip\else & & \tabularnewline\fi}%
8038 }

```

`ongragged4colheader` The `altlongragged4colheader` style is like `altlongragged4col` but with a header row.

```
8039 \newglossarystyle{altlongragged4colheader}{}%
```

Base it on the `glostylealtlongragged4col` style:

```
8040   \setglossarystyle{altlongragged4col}{}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```

8041   \renewenvironment{theglossary}%
8042     {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}l%
8043       >{\raggedright}p{\glspagelistwidth}}}%
8044     {\end{longtable}}%

```

Table has a header:

```

8045   \renewcommand*{\glossaryheader}{}%
8046   \bfseries\entryname&\bfseries\descriptionname&
8047   \bfseries \symbolname&
8048   \bfseries\pagelistname\tabularnewline\endhead}%
8049 }

```


`longragged4colborder` The `altlongragged4colborder` style is like `altlongragged4col` but with a border.

```

8050 \newglossarystyle{altlongragged4colborder}{%
      Base it on the glostylealtlongragged4col style:
8051   \setglossarystyle{altlongragged4col}%
      Use a longtable with 4 columns where the second and last columns may have
      multiple lines in each row:
8052   \renewenvironment{theglossary}%
8053     {\begin{longtable}{|l|>{\raggedright}p{\glsdescwidth}|l|%
8054       >{\raggedright}p{\glspagelistwidth}|}%
8055     {\end{longtable}}%
      Add horizontal lines to the head and foot of the table:
8056   \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8057 }
```

`longragged4colheaderborder` The `altlongragged4colheaderborder` style is like the above but with a header as well as a border.

```

8058 \newglossarystyle{altlongragged4colheaderborder}{%
      Base it on the glostylealtlongragged4col style:
8059   \setglossarystyle{altlongragged4col}%
      Use a longtable with 4 columns where the second and last columns may have
      multiple lines in each row:
8060   \renewenvironment{theglossary}%
8061     {\begin{longtable}{|l|>{\raggedright}p{\glsdescwidth}|l|%
8062       >{\raggedright}p{\glspagelistwidth}|}%
8063     {\end{longtable}}%
      Add table header and horizontal line at the table's foot:
8064   \renewcommand*{\glossaryheader}{%
8065     \hline\bfseries\entryname&\bfseries\descriptionname&
8066     \bfseries \symbolname&
8067     \bfseries\pagelistname\tabularnewline\hline\endhead
8068     \hline\endfoot}%
8069 }
```

5.6 Glossary Styles using `multicol` (`glossary-mcols.sty`)

The style file defines glossary styles that use the `multicol` package. These use the tree-like glossary styles in a `multicol` environment.

```
8070 \ProvidesPackage{glossary-mcols}[2015/02/03 v4.13 (NLCT)]
```

Required packages:

```

8071 \RequirePackage{multicol}
8072 \RequirePackage{glossary-tree}
```

`\indexspace` The are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```
8073 \providecommand{\indexspace}{%
8074   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
8075 }
```

`\glsmcols` Define macro in which to store the number of columns. (Defaults to 2.)

```
8076 \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.)

```
8077 \newglossarystyle{mcolindex}{%
8078   \setglossarystyle{index}%
8079   \renewenvironment{theglossary}%
8080     {%
8081       \begin{multicols}{\glsmcols}
8082       \setlength{\parindent}{0pt}%
8083       \setlength{\parskip}{0pt plus 0.3pt}%
8084       \let\item\@idxitem}%
8085     {\end{multicols}}%
8086 }
```

`mcolindexgroup` As `mcolindex` but has headings:

```
8087 \newglossarystyle{mcolindexgroup}{%
8088   \setglossarystyle{mcolindex}%
8089   \renewcommand*\glsgroupheading}[1]{%
8090     \item\textbf{\glsgetgrouptitle{##1}}\indexspace}%
8091 }
```

`mcolindexhypergroup` The `mcolindexhypergroup` style is like the `mcolindexgroup` style but has hyper navigation.

```
8092 \newglossarystyle{mcolindexhypergroup}{%
```

Base it on the `glostylemcolindex` style:

```
8093   \setglossarystyle{mcolindex}%
```

Put navigation links to the groups at the start of the glossary:

```
8094   \renewcommand*\glossaryheader}{%
8095     \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.

```
8096   \renewcommand*\glsgroupheading}[1]{%
8097     \item\textbf{\glsnavhypertarget{##1}}{\glsgetgrouptitle{##1}}}%
8098     \indexspace}%
8099 }
```

`mcoltree` Multi-column index style. Same as the tree, but puts the glossary in multiple columns.

```
8100 \newglossarystyle{mcoltree}{%
8101   \setglossarystyle{tree}%
8102   \renewenvironment{theglossary}%
8103   {%
8104     \begin{multicols}{\glsmcols}
8105     \setlength{\parindent}{0pt}%
8106     \setlength{\parskip}{0pt plus 0.3pt}%
8107   }%
8108   {\end{multicols}}%
8109 }
```

`mcoltreegroup` Like the `mcoltree` style but the glossary groups have headings.

```
8110 \newglossarystyle{mcoltreegroup}{%
      Base it on the glostylemcoltree style:
8111   \setglossarystyle{mcoltree}%
      Each group has a heading (in bold) followed by a vertical gap):
8112   \renewcommand{\glsgroupheading}[1]{\par
8113     \noindent\textbf{\glsgetgrouptitle{##1}}\par\indexspace}%
8114 }
```

`mcoltreehypergroup` The `mcoltreehypergroup` style is like the `treegroup` style, but has a set of links to the groups at the start of the glossary.

```
8115 \newglossarystyle{mcoltreehypergroup}{%
      Base it on the glostylemcoltree style:
8116   \setglossarystyle{mcoltree}%
      Put navigation links to the groups at the start of the theglossary environment:
8117   \renewcommand*\glossaryheader}{%
8118     \par\noindent\textbf{\glsnavigation}\par\indexspace}%
      Each group has a heading (in bold with a target) followed by a vertical gap):
8119   \renewcommand*\glsgroupheading}[1]{%
8120     \par\noindent
8121     \textbf{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8122     \indexspace}%
8123 }
```

`mcoltreename` Multi-column index style. Same as the `treenoname`, but puts the glossary in multiple columns.

```
8124 \newglossarystyle{mcoltreename}{%
8125   \setglossarystyle{treenoname}%
8126   \renewenvironment{theglossary}%
8127   {%
```

```

8128     \begin{multicols}{\glsmcols}
8129     \setlength{\parindent}{0pt}%
8130     \setlength{\parskip}{0pt plus 0.3pt}%
8131   }%
8132   {\end{multicols}}}%
8133 }

```

`mcoltreenamegroup` Like the `mcoltreename` style but the glossary groups have headings.

```

8134 \newglossarystyle{mcoltreenamegroup}{%
      Base it on the glostylemcoltreename style:
8135   \setglossarystyle{mcoltreename}%
      Give each group a heading:
8136   \renewcommand{\glsgroupheading}[1]{\par
8137     \noindent\textbf{\glsgetgrouptitle{##1}}\par\indexspace}%
8138 }

```

`treenamehypergroup` The `mcoltreenamehypergroup` style is like the `mcoltreenamegroup` style, but has a set of links to the groups at the start of the glossary.

```

8139 \newglossarystyle{mcoltreenamehypergroup}{%
      Base it on the glostylemcoltreename style:
8140   \setglossarystyle{mcoltreename}%
      Put navigation links to the groups at the start of the theglossary environment:
8141   \renewcommand*{\glossaryheader}{%
8142     \par\noindent\textbf{\glsnavigation}\par\indexspace}%
      Each group has a heading (in bold with a target) followed by a vertical gap):
8143   \renewcommand*{\glsgroupheading}[1]{%
8144     \par\noindent
8145     \textbf{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8146     \indexspace}%
8147 }

```

`mcolalmtree` Multi-column index style. Same as the `almtree`, but puts the glossary in multiple columns.

```

8148 \newglossarystyle{mcolalmtree}{%
8149   \setglossarystyle{almtree}%
8150   \renewenvironment{theglossary}%
8151   {%
8152     \begin{multicols}{\glsmcols}
8153     \def\@gls@prevlevel{-1}%
8154     \mbox{}\par
8155   }%
8156   {\par\end{multicols}}}%
8157 }

```

`mcolalmtreegroup` Like the `mcolalmtree` style but the glossary groups have headings.

```
8158 \newglossarystyle{mcolalmtreegroup}{%
      Base it on the glostylemcolalmtree style:
8159 \setglossarystyle{mcolalmtree}%
      Give each group a heading.
8160 \renewcommand{\glsgroupheading}[1]{\par
8161 \def\@gls@prevlevel{-1}%
8162 \hangindent0pt\relax
8163 \parindent0pt\relax
8164 \textbf{\glsgetgrouptitle{##1}}\par\indexspace}%
8165 }
```

`colalmtreehypergroup` The `mcolalmtreehypergroup` style is like the `mcolalmtreegroup` style, but has a set of links to the groups at the start of the glossary.

```
8166 \newglossarystyle{mcolalmtreehypergroup}{%
      Base it on the glostylemcolalmtree style:
8167 \setglossarystyle{mcolalmtree}%
      Put the navigation links in the header
8168 \renewcommand*\glossaryheader}{%
8169 \par
8170 \def\@gls@prevlevel{-1}%
8171 \hangindent0pt\relax
8172 \parindent0pt\relax
8173 \textbf{\glsnavigation}\par\indexspace}%
      Put a hypertarget at the start of each group
8174 \renewcommand*\glsgroupheading[1]{%
8175 \par
8176 \def\@gls@prevlevel{-1}%
8177 \hangindent0pt\relax
8178 \parindent0pt\relax
8179 \textbf{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8180 \indexspace}}
```

5.7 Glossary Styles using supertabular environment (`glossary-super` package)

The glossary styles defined in the package use the `supertabular` environment.

```
8181 \ProvidesPackage{glossary-super}[2013/11/14 v4.0 (NLCT)]
```

Requires the package:

```
8182 \RequirePackage{supertabular}
```

`\glsdescwidth` This is a length that governs the width of the description column. This may already have been defined if `glossary-super` has been loaded.

```
8183 \@ifundefined{glsdescwidth}{%
```

```

8184 \newlength\glsdescwidth
8185 \setlength{\glsdescwidth}{0.6\hsize}
8186 }{}

```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined if has been loaded.

```

8187 \@ifundefined{glspagelistwidth}{%
8188 \newlength\glspagelistwidth
8189 \setlength{\glspagelistwidth}{0.1\hsize}
8190 }{}

```

`super` The `super` glossary style uses the `supertabular` environment (it uses lengths defined in the package.)

```
8191 \newglossarystyle{super}{%
```

Put the glossary in a `supertabular` environment with two columns and no head or tail:

```

8192 \renewenvironment{theglossary}%
8193 {\tablehead{}\tabletail{}}%
8194 \begin{supertabular}{lp{\glsdescwidth}}%
8195 {\end{supertabular}}%

```

Do nothing at the start of the table:

```
8196 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8197 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```

8198 \renewcommand{\glossentry}[2]{%
8199 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8200 \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline
8201 }%

```

Sub entries put in a row (no name, description and page list in second column):

```

8202 \renewcommand{\subglossentry}[3]{%
8203 &
8204 \glsesubentryitem{##2}%
8205 \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
8206 ##3\tabularnewline
8207 }%

```

Blank row between groups:

```

8208 \renewcommand*{\glsgroupskip}{%
8209 \ifglsnogroupskip\else & \tabularnewline\fi}%
8210 }

```

`superborder` The `superborder` style is like the above, but with horizontal and vertical lines:

```
8211 \newglossarystyle{superborder}{%
```

Base it on the `glostylesuper` style:

```
8212 \setglossarystyle{super}%
```

Put the glossary in a `supertabular` environment with two columns and a horizontal line in the head and tail:

```
8213 \renewenvironment{theglossary}%  
8214   {\tablehead{\hline}\tabletail{\hline}%  
8215   \begin{supertabular}{|l|p{\glsdescwidth}|}%  
8216   {\end{supertabular}}%  
8217 }
```

`superheader` The `superheader` style is like the `super` style, but with a header:

```
8218 \newglossarystyle{superheader}{%
```

Base it on the `glostylesuper` style:

```
8219 \setglossarystyle{super}%
```

Put the glossary in a `supertabular` environment with two columns, a header and no tail:

```
8220 \renewenvironment{theglossary}%  
8221   {\tablehead{\bfseries \entryname &  
8222   \bfseries \descriptionname \tabularnewline}%  
8223   \tabletail{}}%  
8224   \begin{supertabular}{|lp{\glsdescwidth}|}%  
8225   {\end{supertabular}}%  
8226 }
```

`superheaderborder` The `superheaderborder` style is like the `super` style but with a header and border:

```
8227 \newglossarystyle{superheaderborder}{%
```

Base it on the `glostylesuper` style:

```
8228 \setglossarystyle{super}%
```

Put the glossary in a `supertabular` environment with two columns, a header and horizontal lines above and below the table:

```
8229 \renewenvironment{theglossary}%  
8230   {\tablehead{\hline\bfseries \entryname &  
8231   \bfseries \descriptionname \tabularnewline\hline}%  
8232   \tabletail{\hline}  
8233   \begin{supertabular}{|l|p{\glsdescwidth}|}%  
8234   {\end{supertabular}}%  
8235 }
```

`super3col` The `super3col` style is like the `super` style, but with 3 columns:

```
8236 \newglossarystyle{super3col}{%
```

Put the glossary in a `supertabular` environment with three columns and no head or tail:

```
8237 \renewenvironment{theglossary}%  
8238   {\tablehead{} \tabletail{}}%  
8239   \begin{supertabular}{|lp{\glsdescwidth}p{\glspagelistwidth}|}%  
8240   {\end{supertabular}}%
```

Do nothing at the start of the table:

```
8241 \renewcommand*\glossaryheader}{}%
```

No group headings:

```
8242 \renewcommand*\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
8243 \renewcommand{\glossentry}[2]{%
8244   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8245   \glossentrydesc{##1} & ##2\tabularnewline
8246 }%
```

Sub entries on a row (no name, description in second column, page list in last column):

```
8247 \renewcommand{\subglossentry}[3]{%
8248   &
8249   \glssubentryitem{##2}%
8250   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8251   ##3\tabularnewline
8252 }%
```

Blank row between groups:

```
8253 \renewcommand*\glsgroupskip}{%
8254   \ifglsnogroupskip\else & &\tabularnewline\fi}%
8255 }
```

super3colborder The `super3colborder` style is like the `super3col` style, but with a border:

```
8256 \newglossarystyle{super3colborder}{%
```

Base it on the `glostylesuper3col` style:

```
8257 \setglossarystyle{super3col}{%
```

Put the glossary in a `supertabular` environment with three columns and a horizontal line in the head and tail:

```
8258 \renewenvironment{theglossary}%
8259   {\tablehead{\hline}\tabletail{\hline}%
8260   \begin{supertabular}{|l|p{\glsdescwidth}|p{\glspagelistwidth}|}%
8261   {\end{supertabular}}%
8262 }
```

super3colheader The `super3colheader` style is like the `super3col` style but with a header row:

```
8263 \newglossarystyle{super3colheader}{%
```

Base it on the `glostylesuper3col` style:

```
8264 \setglossarystyle{super3col}{%
```

Put the glossary in a `supertabular` environment with three columns, a header and no tail:

```
8265 \renewenvironment{theglossary}%
8266   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
```



```

8267     \bfseries\pagelistname\tabularnewline}\tabletail{}}%
8268     \begin{supertabular}{lp{\glstdescwidth}p{\glspagelistwidth}}}%
8269     {\end{supertabular}}}%
8270 }

```

`super3colheaderborder` The `super3colheaderborder` style is like the `super3col` style but with a header and border:

```

8271 \newglossarystyle{super3colheaderborder}{%
      Base it on the glostylesuper3colborder style:
8272   \setglossarystyle{super3colborder}%
      Put the glossary in a supertabular environment with three columns, a header
      with horizontal lines and a horizontal line in the tail:
8273   \renewenvironment{theglossary}%
8274     {\tablehead{\hline
8275       \bfseries\entryname&\bfseries\descriptionname&
8276       \bfseries\pagelistname\tabularnewline\hline}%
8277     \tabletail{\hline}%
8278     \begin{supertabular}{|l|p{\glstdescwidth}|p{\glspagelistwidth}|}}}%
8279     {\end{supertabular}}}%
8280 }

```

`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.

```

8281 \newglossarystyle{super4col}{%
      Put the glossary in a supertabular environment with four columns and no head
      or tail:
8282   \renewenvironment{theglossary}%
8283     {\tablehead{}\tabletail{}}%
8284     \begin{supertabular}{|l|l|l|l|}}}%
8285     \end{supertabular}}}%

```

Do nothing at the start of the table:

```

8286 \renewcommand*{\glossaryheader}{}%

```

No group headings:

```

8287 \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:

```

8288 \renewcommand{\glossentry}[2]{%
8289   \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8290   \glossentrydesc{##1} &
8291   \glossentrysymbol{##1} & ##3\tabularnewline
8292 }%

```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```

8293 \renewcommand{\subglossentry}[3]{%

```

```

8294     &
8295     \glssubentryitem{##2}%
8296     \glstarget{##2}{\strut}\glossentrydesc{##2} &
8297     \glossentrysymbol{##2} & ##3\tabularnewline
8298 }%

```

Blank row between groups:

```

8299 \renewcommand*{\glsgroupskip}{%
8300     \ifglsnogroupskip\else & & \tabularnewline\fi}%
8301 }

```

super4colheader The super4colheader style is like the super4col but with a header row.

```

8302 \newglossarystyle{super4colheader}{%
      Base it on the glostylesuper4col style:
8303   \setglossarystyle{super4col}%
      Put the glossary in a supertabular environment with four columns, a header and
      no tail:
8304   \renewenvironment{theglossary}%
8305     {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8306               \bfseries\symbolname &
8307               \bfseries\pagelistname\tabularnewline}%
8308     \tabletail{}}%
8309     \begin{supertabular}{|l|l|l|l|}%
8310     {\end{supertabular}}%
8311 }

```

super4colborder The super4colborder style is like the super4col but with a border.

```

8312 \newglossarystyle{super4colborder}{%
      Base it on the glostylesuper4col style:
8313   \setglossarystyle{super4col}%
      Put the glossary in a supertabular environment with four columns and a hori-
      zontal line in the head and tail:
8314   \renewenvironment{theglossary}%
8315     {\tablehead{\hline}\tabletail{\hline}%
8316     \begin{supertabular}{|l|l|l|l|}%
8317     {\end{supertabular}}%
8318 }

```

super4colheaderborder The super4colheaderborder style is like the super4col but with a header and border.

```

8319 \newglossarystyle{super4colheaderborder}{%
      Base it on the glostylesuper4col style:
8320   \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:

```

8321 \renewenvironment{theglossary}%
8322   {\tablehead{\hline\bfseries\entryname&\bfseries\descriptionname&
8323     \bfseries\symbolname &
8324     \bfseries\pagelistname\tabularnewline\hline}%
8325   \tabletail{\hline}%
8326   \begin{supertabular}{|1|1|1|1|}%
8327   {\end{supertabular}}%
8328 }

```

`altsuper4col` The `altsuper4col` glossary style is like `super4col` but has provision for multiline descriptions.

```
8329 \newglossarystyle{altsuper4col}{%
```

Base it on the `glostylesuper4col` style:

```
8330 \setglossarystyle{super4col}%
```

Put the glossary in a supertabular environment with four columns and no head or tail:

```

8331 \renewenvironment{theglossary}%
8332   {\tablehead{}\tabletail{}}%
8333   \begin{supertabular}{lp{\glsdescwidth}lp{\glspagelistwidth}}%
8334   {\end{supertabular}}%
8335 }

```

`altsuper4colheader` The `altsuper4colheader` style is like the `altsuper4col` but with a header row.

```
8336 \newglossarystyle{altsuper4colheader}{%
```

Base it on the `glostylesuper4colheader` style:

```
8337 \setglossarystyle{super4colheader}%
```

Put the glossary in a supertabular environment with four columns, a header and no tail:

```

8338 \renewenvironment{theglossary}%
8339   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8340     \bfseries\symbolname &
8341     \bfseries\pagelistname\tabularnewline}\tabletail{}}%
8342   \begin{supertabular}{lp{\glsdescwidth}lp{\glspagelistwidth}}%
8343   {\end{supertabular}}%
8344 }

```

`altsuper4colborder` The `altsuper4colborder` style is like the `altsuper4col` but with a border.

```
8345 \newglossarystyle{altsuper4colborder}{%
```

Base it on the `glostylesuper4colborder` style:

```
8346 \setglossarystyle{super4colborder}%
```

Put the glossary in a supertabular environment with four columns and a horizontal line in the head and tail:

```

8347 \renewenvironment{theglossary}%
8348   {\tablehead{\hline}\tabletail{\hline}}%
8349   \begin{supertabular}%
8350     {\l|p{\glsdescwidth}|l|p{\glspagelistwidth}|}%
8351   {\end{supertabular}}%
8352 }

```

`per4colheaderborder` The `altsuper4colheaderborder` style is like the `altsuper4col` but with a header and border.

```
8353 \newglossarystyle{altsuper4colheaderborder}{%
```

Base it on the `glostylesuper4colheaderborder` style:

```
8354 \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:

```

8355 \renewenvironment{theglossary}%
8356   {\tablehead{\hline
8357     \bfseries\entryname &
8358     \bfseries\descriptionname &
8359     \bfseries\symbolname &
8360     \bfseries\pagelistname\tabularnewline\hline}}%
8361   \tabletail{\hline}}%
8362   \begin{supertabular}%
8363     {\l|p{\glsdescwidth}|l|p{\glspagelistwidth}|}%
8364   {\end{supertabular}}%
8365 }

```

5.8 Glossary Styles using `supertabular` environment (`glossary-superragged` package)

The glossary styles defined in the package use the `supertabular` environment. These styles are like those provided by the package, except that the multiline columns have ragged right justification.

```
8366 \ProvidesPackage{glossary-superragged}[2013/11/14 v4.0 (NLCT)]
```

Requires the package:

```
8367 \RequirePackage{array}
```

Requires the package:

```
8368 \RequirePackage{supertabular}
```

`\glsdescwidth` This is a length that governs the width of the description column. This may already have been defined.

```

8369 \@ifundefined{glsdescwidth}{%
8370   \newlength\glsdescwidth
8371   \setlength{\glsdescwidth}{0.6\hsize}
8372 }{}

```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```
8373 \@ifundefined{glspagelistwidth}{%
8374   \newlength{glspagelistwidth}
8375   \setlength{glspagelistwidth}{0.1\hsize}
8376 }{}
```

`superragged` The `superragged` glossary style uses the `supertabular` environment.

```
8377 \newglossarystyle{superragged}{%
```

Put the glossary in a `supertabular` environment with two columns and no head or tail:

```
8378   \renewenvironment{theglossary}%
8379     {\tablehead{ }\tabletail{ }}%
8380     \begin{supertabular}{1>{\raggedright}p{glstdescwidth}}%
8381     {\end{supertabular}}%
```

Do nothing at the start of the table:

```
8382   \renewcommand*{glossaryheader}{ }%
```

No group headings:

```
8383   \renewcommand*{glsgroupheading}[1]{}%
```

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```
8384   \renewcommand{glossentry}[2]{%
8385     \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8386     \glossentrydesc{##1}\glspostdescription\space ##2%
8387     \tabularnewline
8388   }%
```

Sub entries put in a row (no name, description and page list in second column):

```
8389   \renewcommand{\subglossentry}[3]{%
8390     &
8391     \glssubentryitem{##2}%
8392     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
8393     ##3%
8394     \tabularnewline
8395   }%
```

Blank row between groups:

```
8396   \renewcommand*{glsgroupskip}{\ifglsnogroupskip\else & \tabularnewline\fi}%
8397 }
```

`superraggedborder` The `superraggedborder` style is like the above, but with horizontal and vertical lines:

```
8398 \newglossarystyle{superraggedborder}{%
```

Base it on the `glostylessuperragged` style:

```
8399   \setglossarystyle{superragged}%
```

Put the glossary in a supertabular environment with two columns and a horizontal line in the head and tail:

```
8400 \renewenvironment{theglossary}%
8401   {\tablehead{\hline}\tabletail{\hline}%
8402    \begin{supertabular}{|l|>{\raggedright}p{\glsdescwidth}|}}%
8403   {\end{supertabular}}%
8404 }
```

`superraggedheader` The `superraggedheader` style is like the `super` style, but with a header:

```
8405 \newglossarystyle{superraggedheader}{%
```

Base it on the `glostylesuperragged` style:

```
8406 \setglossarystyle{superragged}%
```

Put the glossary in a supertabular environment with two columns, a header and no tail:

```
8407 \renewenvironment{theglossary}%
8408   {\tablehead{\bfseries \entryname & \bfseries \descriptionname
8409    \tabularnewline}%
8410    \tabletail{}}%
8411   \begin{supertabular}{|l|>{\raggedright}p{\glsdescwidth}}}%
8412   {\end{supertabular}}%
8413 }
```

`superraggedheaderborder` The `superraggedheaderborder` style is like the `superragged` style but with a header and border:

```
8414 \newglossarystyle{superraggedheaderborder}{%
```

Base it on the `glostylesuper` style:

```
8415 \setglossarystyle{superragged}%
```

Put the glossary in a supertabular environment with two columns, a header and horizontal lines above and below the table:

```
8416 \renewenvironment{theglossary}%
8417   {\tablehead{\hline\bfseries \entryname &
8418    \bfseries \descriptionname\tabularnewline\hline}%
8419    \tabletail{\hline}
8420    \begin{supertabular}{|l|>{\raggedright}p{\glsdescwidth}|}}%
8421   {\end{supertabular}}%
8422 }
```

`superragged3col` The `superragged3col` style is like the `superragged` style, but with 3 columns:

```
8423 \newglossarystyle{superragged3col}{%
```

Put the glossary in a supertabular environment with three columns and no head or tail:

```
8424 \renewenvironment{theglossary}%
8425   {\tablehead{}\tabletail{}}%
8426   \begin{supertabular}{|l|>{\raggedright}p{\glsdescwidth}%
8427    >{\raggedright}p{\glsdescwidth}}}%
8428   {\end{supertabular}}%
```

Do nothing at the start of the table:

```
8429 \renewcommand*\glossaryheader}{}%
```

No group headings:

```
8430 \renewcommand*\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
8431 \renewcommand{\glossentry}[2]{%
8432   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8433   \glossentrydesc{##1} &
8434   ##2\tabularnewline
8435 }%
```

Sub entries on a row (no name, description in second column, page list in last column):

```
8436 \renewcommand{\subglossentry}[3]{%
8437   &
8438   \glssubentryitem{##2}%
8439   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8440   ##3\tabularnewline
8441 }%
```

Blank row between groups:

```
8442 \renewcommand*\glsgroupskip{\ifglsnogroupskip\else & \tabularnewline\fi}%
8443 }
```

`superragged3colborder` The `superragged3colborder` style is like the `superragged3col` style, but with a border:

```
8444 \newglossarystyle{superragged3colborder}{%
```

Base it on the `glostylesuperragged3col` style:

```
8445 \setglossarystyle{superragged3col}%
```

Put the glossary in a `supertabular` environment with three columns and a horizontal line in the head and tail:

```
8446 \renewenvironment{theglossary}%
8447   {\tablehead{\hline}\tabletail{\hline}%
8448   \begin{supertabular}{|l|>{\raggedright}p{\glsdescwidth}|%
8449     >{\raggedright}p{\glspagerlistwidth}|}%
8450   {\end{supertabular}}%
8451 }
```

`superragged3colheader` The `superragged3colheader` style is like the `superragged3col` style but with a header row:

```
8452 \newglossarystyle{superragged3colheader}{%
```

Base it on the `glostylesuperragged3col` style:

```
8453 \setglossarystyle{superragged3col}%
```

Put the glossary in a supertabular environment with three columns, a header and no tail:

```
8454 \renewenvironment{theglossary}%
8455   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8456     \bfseries\pagelistname\tabularnewline}\tabletail{}}%
8457   \begin{supertabular}{l>{\raggedright}p{\glstdescwidth}%
8458     >{\raggedright}p{\glspagelistwidth}}}%
8459   {\end{supertabular}}%
8460 }
```

`ght3colheaderborder` The `superragged3colheaderborder` style is like the `superragged3col` style but with a header and border:

```
8461 \newglossarystyle{superragged3colheaderborder}{%
```

Base it on the `glostylesuperragged3colborder` style:

```
8462 \setglossarystyle{superragged3colborder}%
```

Put the glossary in a supertabular environment with three columns, a header with horizontal lines and a horizontal line in the tail:

```
8463 \renewenvironment{theglossary}%
8464   {\tablehead{\hline
8465     \bfseries\entryname&\bfseries\descriptionname&
8466     \bfseries\pagelistname\tabularnewline\hline}%
8467   \tabletail{\hline}%
8468   \begin{supertabular}{|l|>{\raggedright}p{\glstdescwidth}|%
8469     >{\raggedright}p{\glspagelistwidth}|}%
8470   {\end{supertabular}}%
8471 }
```

`altsuperragged4col` The `altsuperragged4col` glossary style is like `altsuper4col` style in the package but uses ragged right formatting in the description and page list columns.

```
8472 \newglossarystyle{altsuperragged4col}{%
```

Put the glossary in a supertabular environment with four columns and no head or tail:

```
8473 \renewenvironment{theglossary}%
8474   {\tablehead{}\tabletail{}}%
8475   \begin{supertabular}{l>{\raggedright}p{\glstdescwidth}l%
8476     >{\raggedright}p{\glspagelistwidth}}}%
8477   {\end{supertabular}}%
```

Do nothing at the start of the table:

```
8478 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8479 \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:

```
8480 \renewcommand{\glossentry}[2]{}%
```



```

8481 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8482 \glossentrydesc{##1} &
8483 \glossentrysymbol{##1} & ##2\tabularnewline
8484 }%

```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```

8485 \renewcommand{\subglossentry}[3]{%
8486   &
8487   \glssubentryitem{##2}%
8488   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8489   \glossentrysymbol{##2} & ##3\tabularnewline
8490 }%

```

Blank row between groups:

```

8491 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else & & \tabularnewline\fi}%
8492 }

```

`altperragged4colheader` The `altperragged4colheader` style is like the `altperragged4col` style but with a header row.

```

8493 \newglossarystyle{altperragged4colheader}{%

```

Base it on the `glostylealtperragged4col` style:

```

8494 \setglossarystyle{altperragged4col}%

```

Put the glossary in a `supertabular` environment with four columns, a header and no tail:

```

8495 \renewenvironment{theglossary}%
8496   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8497     \bfseries\symbolname &
8498     \bfseries\pagelistname\tabularnewline}\tabletail{}}%
8499   \begin{supertabular}{1>{\raggedright}p{\glsdescwidth}1%
8500     >{\raggedright}p{\glspagelistwidth}}}%
8501   {\end{supertabular}}%
8502 }

```

`altperragged4colborder` The `altperragged4colborder` style is like the `altperragged4col` style but with a border.

```

8503 \newglossarystyle{altperragged4colborder}{%

```

Base it on the `glostylealtperragged4col` style:

```

8504 \setglossarystyle{altperragged4col}%

```

Put the glossary in a `supertabular` environment with four columns and a horizontal line in the head and tail:

```

8505 \renewenvironment{theglossary}%
8506   {\tablehead{\hline}\tabletail{\hline}}%
8507   \begin{supertabular}%
8508     {1|>{\raggedright}p{\glsdescwidth}|1|%
8509     >{\raggedright}p{\glspagelistwidth}|}}%
8510   {\end{supertabular}}%
8511 }

```

`ged4colheaderborder` The `altsuperragged4colheaderborder` style is like the `altsuperragged4col` style but with a header and border.

```
8512 \newglossarystyle{altsuperragged4colheaderborder}{%
```

Base it on the `glostylealtsuperragged4col` style:

```
8513 \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:

```
8514 \renewenvironment{theglossary}{%
8515   {\tablehead{\hline
8516     \bfseries\entryname &
8517     \bfseries\descriptionname &
8518     \bfseries\symbolname &
8519     \bfseries\pagelistname\stabularnewline\hline}%
8520   \tabletail{\hline}%
8521   \begin{supertabular}%
8522     {|l|>{\raggedright}p{\glstdescwidth}|l|%
8523       >{\raggedright}p{\glspagelistwidth}|}%
8524   {\end{supertabular}}%
8525 }
```

5.9 Tree Styles (`glossary-tree.sty`)

The style file defines glossary styles that have a tree-like structure. These are designed for hierarchical glossaries.

```
8526 \ProvidesPackage{glossary-tree}[2015/02/03 v4.13 (NLCT)]
```

`\indexspace` The are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```
8527 \providecommand{\indexspace}{%
8528   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
8529 }
```

`\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.

```
8530 \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.

```
8531 \newglossarystyle{index}{%
```

Set the paragraph indentation and skip and define `\item` to be the same as that used by the `index`:

```
8532 \renewenvironment{theglossary}{%
8533   {\setlength{\parindent}{0pt}}%
```

```

8534 \setlength{\parskip}{0pt plus 0.3pt}%
8535 \let\item\@idxitem}%

8536 {\par}%

```

Do nothing at the start of the environment:

```
8537 \renewcommand*{\glossaryheader}{}%
```

No group headers:

```
8538 \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.

```

8539 \renewcommand*{\glossentry}[2]{%
8540 \item\glsentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
8541 \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
8542 \space \glossentrydesc{##1}\glspostdescription\space ##2%
8543 }%

```

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.

```

8544 \renewcommand{\subglossentry}[3]{%
8545 \ifcase##1\relax
8546 % level 0
8547 \item
8548 \or
8549 % level 1
8550 \subitem
8551 \glssubentryitem{##2}%
8552 \else
8553 % all other levels
8554 \subsubitem
8555 \fi
8556 \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
8557 \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
8558 \space\glossentrydesc{##2}\glspostdescription\space ##3%
8559 }%

```

Vertical gap between groups is the same as that used by indices:

```
8560 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}}
```

`indexgroup` The `indexgroup` style is like the `index` style but has headings.

```
8561 \newglossarystyle{indexgroup}{%
```

Base it on the `glostyleindex` style:

```
8562 \setglossarystyle{index}%
```

Add a heading for each group. This puts the group's title in bold followed by a vertical gap.

```

8563 \renewcommand*\glsgroupheading}[1]{%
8564 \item\glstreenamefmt{\glsgetgrouptitle{##1}}\indexspace}%
8565 }

```

`indexhypergroup` The `indexhypergroup` style is like the `indexgroup` style but has hyper navigation.

```
8566 \newglossarystyle{indexhypergroup}{%
```

Base it on the `glostyleindex` style:

```
8567 \setglossarystyle{index}{%
```

Put navigation links to the groups at the start of the glossary:

```
8568 \renewcommand*\glossaryheader}{%
```

```
8569 \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.

```
8570 \renewcommand*\glsgroupheading}[1]{%
```

```
8571 \item\glstreenamefmt{\glsnavhypertarget{##1}}{\glsgetgrouptitle{##1}}}%

```

```
8572 \indexspace}%

```

```
8573 }
```

`tree` The `tree` glossary style is similar in style to the `index` style, but can have arbitrary levels.

```
8574 \newglossarystyle{tree}{%
```

Set the paragraph indentation and skip:

```
8575 \renewenvironment{theglossary}{%
```

```
8576 {\setlength{\parindent}{0pt}%

```

```
8577 \setlength{\parskip}{0pt plus 0.3pt}}%

```

```
8578 {}%
```

Do nothing at the start of the `theglossary` environment:

```
8579 \renewcommand*\glossaryheader}{}%

```

No group headings:

```
8580 \renewcommand*\glsgroupheading}[1]{}%

```

Main (level 0) entries: name in bold, followed by symbol in brackets (if it exists), the description and the page list:

```
8581 \renewcommand{\glossentry}[2]{%
```

```
8582 \hangindent0pt\relax

```

```
8583 \parindent0pt\relax

```

```
8584 \glsentryitem{##1}\glstreenamefmt{\glstarget{##1}}{\glossentryname{##1}}}%

```

```
8585 \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%

```

```
8586 \space\glossentrydesc{##1}\glspostdescription\space##2\par

```

```
8587 }%
```

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.

```
8588 \renewcommand{\subglossentry}[3]{%
```

```

8589 \hangindent##1\glstreeindent\relax
8590 \parindent##1\glstreeindent\relax
8591 \ifnum##1=1\relax
8592 \glssubentryitem{##2}%
8593 \fi
8594 \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
8595 \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
8596 \space\glossentrydesc{##2}\glspostdescription\space ##3\par
8597 }%

```

Vertical gap between groups is the same as that used by indices:

```

8598 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}

```

treegroup Like the tree style but the glossary groups have headings.

```

8599 \newglossarystyle{treegroup}{%

```

Base it on the glostyletree style:

```

8600 \setglossarystyle{tree}%

```

Each group has a heading (in bold) followed by a vertical gap):

```

8601 \renewcommand{\glsgroupheading}[1]{\par
8602 \noindent\glstreenamefmt{\glsgrouptitle{##1}}\par\indexspace}%
8603 }

```

treehypergroup The treehypergroup style is like the treegroup style, but has a set of links to the groups at the start of the glossary.

```

8604 \newglossarystyle{treehypergroup}{%

```

Base it on the glostyletree style:

```

8605 \setglossarystyle{tree}%

```

Put navigation links to the groups at the start of the theglossary environment:

```

8606 \renewcommand*{\glossaryheader}{%
8607 \par\noindent\glstreenamefmt{\glsgroupnavigation}\par\indexspace}%

```

Each group has a heading (in bold with a target) followed by a vertical gap):

```

8608 \renewcommand*{\glsgroupheading}[1]{%
8609 \par\noindent
8610 \glstreenamefmt{\glsgrouphypertarget{##1}{\glsgrouptitle{##1}}}\par
8611 \indexspace}%
8612 }

```

\glstreeindent Length governing left indent for each level of the tree style.

```

8613 \newlength\glstreeindent
8614 \setlength{\glstreeindent}{10pt}

```

treenoname The treenoname glossary style is like the tree style, but doesn't print the name or symbol for sub-levels.

```

8615 \newglossarystyle{treenoname}{%

```

Set the paragraph indentation and skip:

```
8616 \renewenvironment{theglossary}%  
8617   {\setlength{\parindent}{0pt}}%  
8618   \setlength{\parskip}{0pt plus 0.3pt}}%  
8619   {}%
```

No header:

```
8620 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8621 \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.

```
8622 \renewcommand{\glossentry}[2]{}%  
8623   \hangindent0pt\relax  
8624   \parindent0pt\relax  
8625   \glsentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%  
8626   \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%  
8627   \space\glossentrydesc{##1}\glspostdescription\space##2\par  
8628   }%
```

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.

```
8629 \renewcommand{\subglossentry}[3]{}%  
8630   \hangindent##1\glstreeindent\relax  
8631   \parindent##1\glstreeindent\relax  
8632   \ifnum##1=1\relax  
8633     \glssubentryitem{##2}%  
8634     \fi  
8635     \glstarget{##2}{\strut}%  
8636     \glossentrydesc{##2}\glspostdescription\space##3\par  
8637   }%
```

Vertical gap between groups is the same as that used by indices:

```
8638 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%  
8639 }
```

`treenonamegroup` Like the `treenoname` style but the glossary groups have headings.

```
8640 \newglossarystyle{treenonamegroup}{}%
```

Base it on the `glostyletreenoname` style:

```
8641 \setglossarystyle{treenoname}{}%
```

Give each group a heading:

```
8642 \renewcommand{\glsgroupheading}[1]{\par  
8643   \noindent\glstreenamefmt{\glsgetgrouptitle{##1}}\par\indexspace}%  
8644 }
```

`treenonamehypergroup` The `treenonamehypergroup` style is like the `treenonamegroup` style, but has a set of links to the groups at the start of the glossary.

```
8645 \newglossarystyle{treenonamehypergroup}{}%
```

Base it on the `glostytreename` style:

```
8646 \setglossarystyle{treename}%
```

Put navigation links to the groups at the start of the `theglossary` environment:

```
8647 \renewcommand*\glossaryheader}{%
8648 \par\noindent\glstreenamefmt{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
8649 \renewcommand*\glsgroupheading}[1]{%
8650 \par\noindent
8651 \glstreenamefmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8652 \indexspace}%
8653 }
```

`\glssetwidest` `\glssetwidest[level]{text}` sets the widest text for the given level. It is used by the `almtree` glossary styles to determine the indentation of each level.

```
8654 \newcommand*\glssetwidest}[2][0]{%
8655 \expandafter\def\csname @glswidestname\romannumeral#1\endcsname{%
8656 #2}%
8657 }
```

`\@glswidestname` Initialise `\@glswidestname`.

```
8658 \newcommand*\@glswidestname{}
```

`almtree` The `almtree` 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`.

```
8659 \newglossarystyle{almtree}{%
```

Redefine the `theglossary` environment.

```
8660 \renewenvironment{theglossary}%
8661 {\def\@gls@prevlevel{-1}%
8662 \mbox{}\par}%
8663 {\par}%
```

Set the header and group headers to nothing.

```
8664 \renewcommand*\glossaryheader}{}%
8665 \renewcommand*\glsgroupheading}[1]{}%
```

Redefine the way that the level 0 entries are displayed.

```
8666 \renewcommand{\glossentry}[2]{%
8667 \ifnum\@gls@prevlevel=0\relax
8668 \else
```

Find out how big the indentation should be by measuring the widest entry.

```
8669 \settowidth{\glstreeindent}{\glstreenamefmt{\@glswidestname\space}}%
8670 \fi
```

Set the `hangindent` and `paragraph indent`.

```
8671 \hangindent\glstreeindent
8672 \parindent\glstreeindent
```

Put the name to the left of the paragraph block.

```
8673 \makebox[0pt][r]{\makebox[\glstreeindent][l]{%  
8674 \glstryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}}}%
```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
8675 \ifglshassymbol{##1}{(\glossentrysymbol{##1})\space}{}%
```

Do the description followed by the description terminator and location list.

```
8676 \glossentrydesc{##1}\glspostdescription \space ##2\par
```

Set the previous level to 0.

```
8677 \def\@gls@prevlevel{0}%  
8678 }%
```

Redefine the way sub-entries are displayed.

```
8679 \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.

```
8680 \ifnum##1=1\relax  
8681 \glssubentryitem{##2}%  
8682 \fi
```

If the level hasn't changed, keep the same settings, otherwise adjust `\glstreeindent` accordingly.

```
8683 \ifnum\@gls@prevlevel=##1\relax  
8684 \else
```

Compute the widest entry for this level, or for level 0 if not defined for this level.

Store in `\gls@tmplen`

```
8685 \@ifundefined{@glswidestname\romannumeral##1}{%  
8686 \settowidth{\gls@tmplen}{\glstreenamefmt{\@glswidestname\space}}{%  
8687 \settowidth{\gls@tmplen}{\glstreenamefmt{%  
8688 \csname @glswidestname\romannumeral##1\endcsname\space}}}%
```

Determine if going up or down a level

```
8689 \ifnum\@gls@prevlevel<##1\relax
```

Depth has increased, so add the width of the widest entry to `\glstreeindent`.

```
8690 \setlength\glstreeindent\gls@tmplen  
8691 \addtolength\glstreeindent\parindent  
8692 \parindent\glstreeindent  
8693 \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`.

```
8694 \@ifundefined{@glswidestname\romannumeral\@gls@prevlevel}{%  
8695 \settowidth{\glstreeindent}{\glstreenamefmt{%  
8696 \@glswidestname\space}}{%  
8697 \settowidth{\glstreeindent}{\glstreenamefmt{%  
8698 \csname @glswidestname\romannumeral\@gls@prevlevel  
8699 \endcsname\space}}}%
```


Subtract this length from the previous level's paragraph indent and set to `\glstreeindent`.

```
8700     \addtolength\parindent{-\glstreeindent}%
8701     \setlength\glstreeindent\parindent
8702     \fi
8703     \fi
```

Set the hanging indentation.

```
8704     \hangindent\glstreeindent
```

Put the name to the left of the paragraph block

```
8705     \makebox[0pt][r]{\makebox[\gls@tmplen][l]{%
8706     \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}}}%
```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
8707     \ifglshassymbol{##2}{(\glossentrysymbol{##2})\space}{}%
```

Do the description followed by the description terminator and location list.

```
8708     \glossentrydesc{##2}\glspostdescription\space ##3\par
```

Set the previous level macro to the current level.

```
8709     \def\@gls@prevlevel{##1}%
8710     }%
```

Vertical gap between groups is the same as that used by indices:

```
8711     \renewcommand*\glsgroupskip{\ifglsnogroupskip\else\indexspace\fi}%
8712 }
```

`almtreegroup` Like the `almtree` style but the glossary groups have headings.

```
8713 \newglossarystyle{almtreegroup}{%
```

Base it on the `glostylealmtree` style:

```
8714     \setglossarystyle{almtree}%
```

Give each group a heading.

```
8715     \renewcommand{\glsgroupheading}[1]{\par
8716     \def\@gls@prevlevel{-1}%
8717     \hangindent0pt\relax
8718     \parindent0pt\relax
8719     \glstreenamefmt{\glsgetgrouptitle{##1}}\par\indexspace}%
8720 }
```

`almtreehypergroup` The `almtreehypergroup` style is like the `almtreegroup` style, but has a set of links to the groups at the start of the glossary.

```
8721 \newglossarystyle{almtreehypergroup}{%
```

Base it on the `glostylealmtree` style:

```
8722     \setglossarystyle{almtree}%
```

Put the navigation links in the header

```
8723     \renewcommand*\glossaryheader}{%
8724     \par
```

```

8725 \def\@gls@prevlevel{-1}%
8726 \hangindent0pt\relax
8727 \parindent0pt\relax
8728 \glsstrenamfmt{\glsnavigation}\par\indexspace}%

```

Put a hypertarget at the start of each group

```

8729 \renewcommand*\glsgroupheading}[1]{%
8730 \par
8731 \def\@gls@prevlevel{-1}%
8732 \hangindent0pt\relax
8733 \parindent0pt\relax
8734 \glsstrenamfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8735 \indexspace}}

```

6 glossaries-compatible-207

Provides compatibility with version 2.07 and below. This uses original glossaries xindy and makeindex formatting, so can be used with old documents that had customized style files, but hyperlinks may not work properly.

```

8736 \NeedsTeXFormat{LaTeX2e}
8737 \ProvidesPackage{glossaries-compatible-207}[2011/04/02 v1.0 (NLCT)]

```

`\GlsAddXdyAttribute` Adds an attribute in old format.

```

8738 \ifglsxindy
8739 \renewcommand*\GlsAddXdyAttribute[1]{%
8740 \edef\@xdyattributes{\@xdyattributes ^^J \string"#1\string"}%
8741 \expandafter\toks@ \expandafter{\@xdylocref}%
8742 \edef\@xdylocref{\the\toks@ ^^J%
8743 (markup-locref
8744 :open \string"\string~n\string\setentrycounter
8745 {\noexpand\glscounter}%
8746 \expandafter\string\csname#1\endcsname
8747 \expandafter\@gobble\string\{\string" ^^J
8748 :close \string"\expandafter\@gobble\string\}\string" ^^J
8749 :attr \string"#1\string"))}}

```

Only has an effect before `\writeist`:

```
8750 \fi
```

`\GlsAddXdyCounters`

```

8751 \renewcommand*\GlsAddXdyCounters[1]{%
8752 \GlossariesWarning{\string\GlsAddXdyCounters\space not available
8753 in compatibility mode.}%
8754 }

```

Add predefined attributes

```

8755 \GlsAddXdyAttribute{glsnumberformat}
8756 \GlsAddXdyAttribute{textrm}

```

```

8757 \GlsAddXdyAttribute{textsf}
8758 \GlsAddXdyAttribute{texttt}
8759 \GlsAddXdyAttribute{textbf}
8760 \GlsAddXdyAttribute{textmd}
8761 \GlsAddXdyAttribute{textit}
8762 \GlsAddXdyAttribute{textup}
8763 \GlsAddXdyAttribute{textsl}
8764 \GlsAddXdyAttribute{textsc}
8765 \GlsAddXdyAttribute{emph}
8766 \GlsAddXdyAttribute{glshypernumber}
8767 \GlsAddXdyAttribute{hyperrrm}
8768 \GlsAddXdyAttribute{hypersf}
8769 \GlsAddXdyAttribute{hypertt}
8770 \GlsAddXdyAttribute{hyperbf}
8771 \GlsAddXdyAttribute{hypermd}
8772 \GlsAddXdyAttribute{hyperit}
8773 \GlsAddXdyAttribute{hyperup}
8774 \GlsAddXdyAttribute{hypersl}
8775 \GlsAddXdyAttribute{hypersc}
8776 \GlsAddXdyAttribute{hyperemph}

```

\GlsAddXdyLocation Restore v2.07 definition:

```

8777 \ifglxindy
8778 \renewcommand*{\GlsAddXdyLocation}[2]{%
8779 \edef\xdyuserlocationdefs{%
8780 \xdyuserlocationdefs ^^J%
8781 (define-location-class \string"#1\string"^^J\space\space
8782 \space(#2))
8783 }%
8784 \edef\xdyuserlocationnames{%
8785 \xdyuserlocationnames^^J\space\space\space
8786 \string"#1\string"}%
8787 }
8788 \fi

```

\do@wrglossary

```

8789 \renewcommand{\do@wrglossary}[1]{%
  Determine whether to use xindy or makeindex syntax
8790 \ifglxindy
  Need to determine if the formatting information starts with a ( or ) indicating a
  range.
8791 \expandafter\@glo@check@mkidxrangear\@glsnumberformat\@nil
8792 \def\@glo@range{}%
8793 \expandafter\if\@glo@prefix(\relax
8794 \def\@glo@range{:open-range}%
8795 \else
8796 \expandafter\if\@glo@prefix)\relax
8797 \def\@glo@range{:close-range}%

```

8798 \fi

8799 \fi

Get the location and escape any special characters

8800 \protected@edef\@glslocref{\theglsentrycounter}%

8801 \@gls@checkmkidxchars\@glslocref

Write to the glossary file using xindy syntax.

8802 \glossary[\csname glo@#1@type\endcsname]{%

8803 (indexentry :tkey {\csname glo@#1@index\endcsname)

8804 :locref \string"\@glslocref\string" %

8805 :attr \string"\@glo@suffix\string" \@glo@range

8806)

8807 }%

8808 \else

Convert the format information into the format required for makeindex

8809 \@set@glo@numformat\@glo@numfmt\@gls@counter\@glsnumberformat

Write to the glossary file using makeindex syntax.

8810 \glossary[\csname glo@#1@type\endcsname]{%

8811 \string\glossaryentry{\csname glo@#1@index\endcsname

8812 \@gls@encapchar\@glo@numfmt}{\theglsentrycounter}}%

8813 \fi

8814 }

\@set@glo@numformat Only had 3 arguments in v2.07

8815 \def\@set@glo@numformat#1#2#3{%

8816 \expandafter\@glo@check@mkidxrangechar#3\@nil

8817 \protected@edef#1{%

8818 \@glo@prefix setentrycounter []{#2}%

8819 \expandafter\string\csname\@glo@suffix\endcsname

8820 }%

8821 \@gls@checkmkidxchars#1%

8822 }

\writeist Redefine \writeist back to the way it was in v2.07, but change \istfile to
\glswrite.

8823 \ifglxindy

8824 \def\writeist{%

8825 \openout\glswrite=\istfilename

8826 \write\glswrite{;; xindy style file created by the glossaries

8827 package in compatible-2.07 mode}%

8828 \write\glswrite{;; for document '\jobname' on

8829 \the\year-\the\month-\the\day}%

8830 \write\glswrite{^^J; required styles^^J}

8831 \@for\@xdystyle:=\@xdyrequiredstyles\do{%

8832 \ifx\@xdystyle\@empty

8833 \else

8834 \protected@write\glswrite{{(require

8835 \string"\@xdystyle.xdy\string"}}%

```

8836     \fi
8837 }%
8838 \write\glswrite{^^J%
8839     ; list of allowed attributes (number formats)^^J}%
8840 \write\glswrite{(define-attributes ((\@xdyattributes)))}%
8841 \write\glswrite{^^J; user defined alphabets^^J}%
8842 \write\glswrite{\@xdyuseralphabets}%
8843 \write\glswrite{^^J; location class definitions^^J}%
8844 \protected@edef\@gls@roman{\@roman{0\string"
8845     \string"roman-numbers-lowercase\string" :sep \string"}}%
8846 \@onelevel@sanitize\@gls@roman
8847 \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
8848     :sep \string"}%
8849 \@onelevel@sanitize\@tmp
8850 \ifx\@tmp\@gls@roman
8851     \write\glswrite{(define-location-class
8852         \string"roman-page-numbers\string"^^J\space\space\space
8853         (\string"roman-numbers-lowercase\string")
8854         :min-range-length \@glsminrange)}}%
8855 \else
8856     \write\glswrite{(define-location-class
8857         \string"roman-page-numbers\string"^^J\space\space\space
8858         (:sep "\@gls@roman")
8859         :min-range-length \@glsminrange)}}%
8860 \fi
8861 \write\glswrite{(define-location-class
8862     \string"Roman-page-numbers\string"^^J\space\space\space
8863     (\string"roman-numbers-uppercase\string")
8864     :min-range-length \@glsminrange)}}%
8865 \write\glswrite{(define-location-class
8866     \string"arabic-page-numbers\string"^^J\space\space\space
8867     (\string"arabic-numbers\string")
8868     :min-range-length \@glsminrange)}}%
8869 \write\glswrite{(define-location-class
8870     \string"alpha-page-numbers\string"^^J\space\space\space
8871     (\string"alpha\string")
8872     :min-range-length \@glsminrange)}}%
8873 \write\glswrite{(define-location-class
8874     \string"Alpha-page-numbers\string"^^J\space\space\space
8875     (\string"ALPHA\string")
8876     :min-range-length \@glsminrange)}}%
8877 \write\glswrite{(define-location-class
8878     \string"Appendix-page-numbers\string"^^J\space\space\space
8879     (\string"ALPHA\string"
8880     :sep \string"\@glsAlphacompositor\string"
8881     \string"arabic-numbers\string")
8882     :min-range-length \@glsminrange)}}%
8883 \write\glswrite{(define-location-class
8884     \string"arabic-section-numbers\string"^^J\space\space\space

```

```

8885     (\string"arabic-numbers\string"
8886       :sep \string"\glscompositor\string"
8887       \string"arabic-numbers\string")
8888       :min-range-length \@glsminrange))}%
8889 \write\glswrite{^^J; user defined location classes}%
8890 \write\glswrite{\@xdyuserlocationdefs}%
8891 \write\glswrite{^^J; define cross-reference class^^J}%
8892 \write\glswrite{(define-crossref-class \string"see\string"
8893   :unverified )}%
8894 \write\glswrite{(markup-crossref-list
8895   :class \string"see\string"^^J\space\space\space
8896   :open \string"\string\glsseeformat\string"
8897   :close \string"{}\string")}%
8898 \write\glswrite{^^J; define the order of the location classes}%
8899 \write\glswrite{(define-location-class-order
8900   (\@xdylocationclassorder))}%
8901 \write\glswrite{^^J; define the glossary markup^^J}%
8902 \write\glswrite{(markup-index^^J\space\space\space
8903   :open \string"\string
8904     \glossarysection[\string\glossarytoctitle]{\string
8905     \glossarytitle}\string\glossarypreamble\string~n\string\begin
8906     {theglossary}\string\glossaryheader\string~n\string" ^^J\space
8907     \space\space:close \string"\expandafter\@gobble
8908     \string%\string~n\string
8909     \end{theglossary}\string\glossarypostamble
8910     \string~n\string" ^^J\space\space\space
8911     :tree)}}%
8912 \write\glswrite{(markup-letter-group-list
8913   :sep \string"\string\glsgroupskip\string~n\string")}%
8914 \write\glswrite{(markup-indexentry
8915   :open \string"\string\relax \string\glsresetentrylist
8916     \string~n\string")}%
8917 \write\glswrite{(markup-locclass-list :open
8918   \string"\glsopenbrace\string\glossaryentrynumbers
8919   \glsopenbrace\string\relax\space \string"^^J\space\space\space
8920   :sep \string", \string"
8921   :close \string"\glsclosebrace\glsclosebrace\string")}%
8922 \write\glswrite{(markup-locref-list
8923   :sep \string"\string\delimN\space\string")}%
8924 \write\glswrite{(markup-range
8925   :sep \string"\string\delimR\space\string")}%
8926 \@onelevel@sanitize\gls@suffixF
8927 \@onelevel@sanitize\gls@suffixFF
8928 \ifx\gls@suffixF\@empty
8929 \else
8930   \write\glswrite{(markup-range
8931     :close "\gls@suffixF" :length 1 :ignore-end)}%
8932 \fi
8933 \ifx\gls@suffixFF\@empty

```

```

8934 \else
8935   \write\glswrite{(markup-range
8936     :close "\gls@suffixFF" :length 2 :ignore-end)}}%
8937 \fi
8938 \write\glswrite{^^J; define format to use for locations^^J}%
8939 \write\glswrite{\@xdylocref}%
8940 \write\glswrite{^^J; define letter group list format^^J}%
8941 \write\glswrite{(markup-letter-group-list
8942   :sep \string"\string\glsgroupskip\string~n\string")}}%
8943 \write\glswrite{^^J; letter group headings^^J}%
8944 \write\glswrite{(markup-letter-group
8945   :open-head \string"\string\glsgroupheading
8946     \glsopenbrace\string"^^J\space\space\space
8947     :close-head \string"\glsclosebrace\string")}}%
8948 \write\glswrite{^^J; additional letter groups^^J}%
8949 \write\glswrite{\@xdylettergroups}%
8950 \write\glswrite{^^J; additional sort rules^^J}
8951 \write\glswrite{\@xdysortrules}%
8952 \noist}
8953 \else
8954 \edef\@gls@actualchar{\string?}
8955 \edef\@gls@encapchar{\string|}
8956 \edef\@gls@levelchar{\string!}
8957 \edef\@gls@quotechar{\string"}
8958 \def\writeist{\relax
8959   \openout\glswrite=\istfilename
8960   \write\glswrite{\expandafter\@gobble\string\% makeindex style file
8961     created by the glossaries package}
8962   \write\glswrite{\expandafter\@gobble\string\% for document
8963     '\jobname' on \the\year-\the\month-\the\day}
8964   \write\glswrite{actual '\@gls@actualchar'}
8965   \write\glswrite{encap '\@gls@encapchar'}
8966   \write\glswrite{level '\@gls@levelchar'}
8967   \write\glswrite{quote '\@gls@quotechar'}
8968   \write\glswrite{keyword \string"\string\glossaryentry\string"}
8969   \write\glswrite{preamble \string"\string\glossarysection[\string
8970     \glossarytoctitle]{\string\glossarytitle}\string
8971     \glossarypreamble\string\n\string\begin{theglossary}\string
8972     \glossaryheader\string\n\string"}
8973   \write\glswrite{postamble \string"\string\%\string\n\string
8974     \end{theglossary}\string\glossarypostamble\string\n
8975     \string"}
8976   \write\glswrite{group_skip \string"\string\glsgroupskip\string\n
8977     \string"}
8978   \write\glswrite{item_0 \string"\string\%\string\n\string"}
8979   \write\glswrite{item_1 \string"\string\%\string\n\string"}
8980   \write\glswrite{item_2 \string"\string\%\string\n\string"}
8981   \write\glswrite{item_01 \string"\string\%\string\n\string"}
8982   \write\glswrite{item_x1

```

```

8983     \string"\string\relax \string\glsresetentrylist\string\n
8984     \string"}
8985     \write\glswrite{item_12 \string"\string%\string\n\string"}
8986     \write\glswrite{item_x2
8987     \string"\string\relax \string\glsresetentrylist\string\n
8988     \string"}
8989     \write\glswrite{delim_0 \string"\string\{\string
8990     \glossaryentrynumbers\string\{\string\relax \string"}
8991     \write\glswrite{delim_1 \string"\string\{\string
8992     \glossaryentrynumbers\string\{\string\relax \string"}
8993     \write\glswrite{delim_2 \string"\string\{\string
8994     \glossaryentrynumbers\string\{\string\relax \string"}
8995     \write\glswrite{delim_t \string"\string\}\string\}\string"}
8996     \write\glswrite{delim_n \string"\string\delimN \string"}
8997     \write\glswrite{delim_r \string"\string\delimR \string"}
8998     \write\glswrite{headings_flag 1}
8999     \write\glswrite{heading_prefix
9000     \string"\string\glsgroupheading\string\{\string"}
9001     \write\glswrite{heading_suffix
9002     \string"\string\}\string\relax
9003     \string\glsresetentrylist \string"}
9004     \write\glswrite{symhead_positive \string"glssymbols\string"}
9005     \write\glswrite{numhead_positive \string"glnumbers\string"}
9006     \write\glswrite{page_compositor \string"glscpositor\string"}
9007     \@gls@escbsdq\gls@suffixF
9008     \@gls@escbsdq\gls@suffixFF
9009     \ifx\gls@suffixF\@empty
9010     \else
9011     \write\glswrite{suffix_2p \string"\gls@suffixF\string"}
9012     \fi
9013     \ifx\gls@suffixFF\@empty
9014     \else
9015     \write\glswrite{suffix_3p \string"\gls@suffixFF\string"}
9016     \fi
9017     \noist
9018   }
9019 \fi

```

\noist

```
9020 \renewcommand*{\noist}{\let\writeist\relax}
```

Compatibility macros.

```
9021 \NeedsTeXFormat{LaTeX2e}
```

```
9022 \ProvidesPackage{glossaries-compatible-307}[2013/11/14 v4.0 (NLCT)]
```

Compatibility macros for predefined glossary styles:

`compatglossarystyle` Defines a compatibility glossary style.

```
9023 \newcommand{\compatglossarystyle}[2]{%
```

```
9024   \ifcsundef{@glscompstyle@#1}%
```



```

9025  {%
9026    \csdef{@glscompstyle@#1}{#2}%
9027  }%
9028  {%
9029    \PackageError{glossaries}{Glossary compatibility style ‘#1’ is already defined}{}%
9030  }%
9031 }

```

Backward compatible inline style.

```

9032 \compatglossarystyle{inline}{%
9033   \renewcommand{\glossaryentryfield}[5]{%
9034     \glsinlinedopostchild
9035     \gls@inlinesep
9036     \def\glo@desc{##3}%
9037     \def\@no@post@desc{\nopostdesc}%
9038     \glsentryitem{##1}\glsinlinenameformat{##1}{##2}%
9039     \ifx\glo@desc\@no@post@desc
9040       \glsinlineemptydescformat{##4}{##5}%
9041     \else
9042       \ifstrempy{##3}%
9043         {\glsinlineemptydescformat{##4}{##5}}%
9044         {\glsinlinedescformat{##3}{##4}{##5}}%
9045     \fi
9046     \ifglshaschildren{##1}%
9047     {%
9048       \glsresetsubentrycounter
9049       \glsinlineparentchildseparator
9050       \def\gls@inlinesubsep{}%
9051       \def\gls@inlinepostchild{\glsinlinepostchild}%
9052     }%
9053   }%
9054   \def\gls@inlinesep{\glsinlineseparator}%
9055 }%

```

Sub-entries display description:

```

9056 \renewcommand{\glossarysubentryfield}[6]{%
9057   \gls@inlinesubsep%
9058   \glsinlinesubnameformat{##2}{##3}%
9059   \glssubentryitem{##2}\glsinlinesubdescformat{##4}{##5}{##6}%
9060   \def\gls@inlinesubsep{\glsinlinesubseparator}%
9061 }%
9062 }

```

Backward compatible list style.

```

9063 \compatglossarystyle{list}{%
9064   \renewcommand*{\glossaryentryfield}[5]{%
9065     \item[\glsentryitem{##1}\glstarget{##1}{##2}]
9066     ##3\glspostdescription\space ##5}%

```

Sub-entries continue on the same line:

```

9067 \renewcommand*{\glossarysubentryfield}[6]{%

```

```

9068 \glssubentryitem{##2}%
9069 \glstarget{##2}{\strut}##4\glspostdescription\space ##6.}%
9070 }

```

Backward compatible listgroup style.

```

9071 \compatglossarystyle{listgroup}{%
9072 \csuse{@glscmpstyle@list}%
9073 }%

```

Backward compatible listhypergroup style.

```

9074 \compatglossarystyle{listhypergroup}{%
9075 \csuse{@glscmpstyle@list}%
9076 }%

```

Backward compatible altlist style.

```

9077 \compatglossarystyle{altlist}{%
9078 \renewcommand*{\glossaryentryfield}[5]{%
9079 \item[\glssubentryitem{##1}\glstarget{##1}{##2}]%
9080 \mbox{}\par\nobreak\@afterheading
9081 ##3\glspostdescription\space ##5}%
9082 \renewcommand{\glossarysubentryfield}[6]{%
9083 \par
9084 \glssubentryitem{##2}%
9085 \glstarget{##2}{\strut}##4\glspostdescription\space ##6}%
9086 }%

```

Backward compatible altlistgroup style.

```

9087 \compatglossarystyle{altlistgroup}{%
9088 \csuse{@glscmpstyle@altlist}%
9089 }%

```

Backward compatible altlisthypergroup style.

```

9090 \compatglossarystyle{altlisthypergroup}{%
9091 \csuse{@glscmpstyle@altlist}%
9092 }%

```

Backward compatible listdotted style.

```

9093 \compatglossarystyle{listdotted}{%
9094 \renewcommand*{\glossaryentryfield}[5]{%
9095 \item[]\makebox[\glslistdottedwidth][l]{%
9096 \glssubentryitem{##1}\glstarget{##1}{##2}%
9097 \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##3}%
9098 \renewcommand*{\glossarysubentryfield}[6]{%
9099 \item[]\makebox[\glslistdottedwidth][l]{%
9100 \glssubentryitem{##2}%
9101 \glstarget{##2}{##3}%
9102 \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##4}%
9103 }%

```

Backward compatible sublistdotted style.

```

9104 \compatglossarystyle{sublistdotted}{%
9105 \csuse{@glscmpstyle@listdotted}%

```

```

9106 \renewcommand*\glossaryentryfield}[5]{%
9107   \item[\glsentryitem{##1}\glstarget{##1}{##2}]}%
9108}%

  Backward compatible long style.
9109 \compatglossarystyle{long}{%
9110   \renewcommand*\glossaryentryfield}[5]{%
9111     \glsentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
9112   \renewcommand*\glossarysubentryfield}[6]{%
9113     &
9114     \glssubentryitem{##2}%
9115     \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
9116}%

  Backward compatible longborder style.
9117 \compatglossarystyle{longborder}{%
9118   \csuse{@glscompstyle@long}%
9119}%

  Backward compatible longheader style.
9120 \compatglossarystyle{longheader}{%
9121   \csuse{@glscompstyle@long}%
9122}%

  Backward compatible longheaderborder style.
9123 \compatglossarystyle{longheaderborder}{%
9124   \csuse{@glscompstyle@long}%
9125}%

  Backward compatible long3col style.
9126 \compatglossarystyle{long3col}{%
9127   \renewcommand*\glossaryentryfield}[5]{%
9128     \glsentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\\}%
9129   \renewcommand*\glossarysubentryfield}[6]{%
9130     &
9131     \glssubentryitem{##2}%
9132     \glstarget{##2}{\strut}##4 & ##6\\}%
9133}%

  Backward compatible long3colborder style.
9134 \compatglossarystyle{long3colborder}{%
9135   \csuse{@glscompstyle@long3col}%
9136}%

  Backward compatible long3colheader style.
9137 \compatglossarystyle{long3colheader}{%
9138   \csuse{@glscompstyle@long3col}%
9139}%

  Backward compatible long3colheaderborder style.
9140 \compatglossarystyle{long3colheaderborder}{%
9141   \csuse{@glscompstyle@long3col}%
9142}%

```

Backward compatible long4col style.

```
9143 \compatglossarystyle{long4col}{%
9144   \renewcommand*{\glossaryentryfield}[5]{%
9145     \glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%
9146   \renewcommand*{\glossarysubentryfield}[6]{%
9147     &
9148     \glssubentryitem{##2}%
9149     \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%
9150 }
```

Backward compatible long4colheader style.

```
9151 \compatglossarystyle{long4colheader}{%
9152   \csuse{@glscompstyle@long4col}%
9153 }
```

Backward compatible long4colborder style.

```
9154 \compatglossarystyle{long4colborder}{%
9155   \csuse{@glscompstyle@long4col}%
9156 }
```

Backward compatible long4colheaderborder style.

```
9157 \compatglossarystyle{long4colheaderborder}{%
9158   \csuse{@glscompstyle@long4col}%
9159 }
```

Backward compatible altlong4col style.

```
9160 \compatglossarystyle{altlong4col}{%
9161   \csuse{@glscompstyle@long4col}%
9162 }
```

Backward compatible altlong4colheader style.

```
9163 \compatglossarystyle{altlong4colheader}{%
9164   \csuse{@glscompstyle@long4col}%
9165 }
```

Backward compatible altlong4colborder style.

```
9166 \compatglossarystyle{altlong4colborder}{%
9167   \csuse{@glscompstyle@long4col}%
9168 }
```

Backward compatible altlong4colheaderborder style.

```
9169 \compatglossarystyle{altlong4colheaderborder}{%
9170   \csuse{@glscompstyle@long4col}%
9171 }
```

Backward compatible long style.

```
9172 \compatglossarystyle{longragged}{%
9173   \renewcommand*{\glossaryentryfield}[5]{%
9174     \glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
9175     \tabularnewline}%
9176   \renewcommand*{\glossarysubentryfield}[6]{%
9177     &
```

```

9178     \glssubentryitem{##2}%
9179     \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
9180     \tabularnewline}%
9181 }%

```

Backward compatible longraggedborder style.

```

9182 \compatglossarystyle{longraggedborder}{%
9183 \csuse{@glscmpstyle@longragged}%
9184 }%

```

Backward compatible longraggedheader style.

```

9185 \compatglossarystyle{longraggedheader}{%
9186 \csuse{@glscmpstyle@longragged}%
9187 }%

```

Backward compatible longraggedheaderborder style.

```

9188 \compatglossarystyle{longraggedheaderborder}{%
9189 \csuse{@glscmpstyle@longragged}%
9190 }%

```

Backward compatible longragged3col style.

```

9191 \compatglossarystyle{longragged3col}{%
9192 \renewcommand*{\glossaryentryfield}[5]{%
9193     \glstarget{##1}{\strut}##4 & ##3 & ##5\tabularnewline}%
9194 \renewcommand*{\glossarysubentryfield}[6]{%
9195     &
9196     \glssubentryitem{##2}%
9197     \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%
9198 }%

```

Backward compatible longragged3colborder style.

```

9199 \compatglossarystyle{longragged3colborder}{%
9200 \csuse{@glscmpstyle@longragged3col}%
9201 }%

```

Backward compatible longragged3colheader style.

```

9202 \compatglossarystyle{longragged3colheader}{%
9203 \csuse{@glscmpstyle@longragged3col}%
9204 }%

```

Backward compatible longragged3colheaderborder style.

```

9205 \compatglossarystyle{longragged3colheaderborder}{%
9206 \csuse{@glscmpstyle@longragged3col}%
9207 }%

```

Backward compatible altlongragged4col style.

```

9208 \compatglossarystyle{altlongragged4col}{%
9209 \renewcommand*{\glossaryentryfield}[5]{%
9210     \glstarget{##1}{\strut}##4 & ##3 & ##4 & ##5\tabularnewline}%
9211 \renewcommand*{\glossarysubentryfield}[6]{%
9212     &
9213     \glssubentryitem{##2}%

```

```

9214 \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%
9215 }%

```

Backward compatible altlongragged4colheader style.

```

9216 \compatglossarystyle{altlongragged4colheader}{%
9217 \csuse{@glscompstyle@altlong4col}%
9218 }%

```

Backward compatible altlongragged4colborder style.

```

9219 \compatglossarystyle{altlongragged4colborder}{%
9220 \csuse{@glscompstyle@altlong4col}%
9221 }%

```

Backward compatible altlongragged4colheaderborder style.

```

9222 \compatglossarystyle{altlongragged4colheaderborder}{%
9223 \csuse{@glscompstyle@altlong4col}%
9224 }%

```

Backward compatible index style.

```

9225 \compatglossarystyle{index}{%
9226 \renewcommand*{\glossaryentryfield}[5]{%
9227 \item\glstarget{##1}{##2}}%
9228 \ifx\relax##4\relax
9229 \else
9230 \space(##4)%
9231 \fi
9232 \space ##3\glspostdescription \space ##5}%
9233 \renewcommand*{\glossarysubentryfield}[6]{%
9234 \ifcase##1\relax
9235 % level 0
9236 \item
9237 \or
9238 % level 1
9239 \subitem
9240 \glssubentryitem{##2}%
9241 \else
9242 % all other levels
9243 \subsubitem
9244 \fi
9245 \textbf{\glstarget{##2}{##3}}%
9246 \ifx\relax##5\relax
9247 \else
9248 \space(##5)%
9249 \fi
9250 \space##4\glspostdescription\space ##6}%
9251 }%

```

Backward compatible indexgroup style.

```

9252 \compatglossarystyle{indexgroup}{%
9253 \csuse{@glscompstyle@index}%
9254 }%

```

Backward compatible indexhypergroup style.

```
9255 \compatglossarystyle{indexhypergroup}{%
9256 \csuse{@glscompstyle@index}%
9257 }%
```

Backward compatible tree style.

```
9258 \compatglossarystyle{tree}{%
9259 \renewcommand{\glossaryentryfield}[5]{%
9260 \hangindent0pt\relax
9261 \parindent0pt\relax
9262 \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
9263 \ifx\relax##4\relax
9264 \else
9265 \space{##4}%
9266 \fi
9267 \space ##3\glspostdescription \space ##5\par}%
9268 \renewcommand{\glossarysubentryfield}[6]{%
9269 \hangindent##1\glstreeindent\relax
9270 \parindent##1\glstreeindent\relax
9271 \ifnum##1=1\relax
9272 \glssubentryitem{##2}%
9273 \fi
9274 \textbf{\glstarget{##2}{##3}}%
9275 \ifx\relax##5\relax
9276 \else
9277 \space{##5}%
9278 \fi
9279 \space##4\glspostdescription\space ##6\par}%
9280 }%
```

Backward compatible treegroup style.

```
9281 \compatglossarystyle{treegroup}{%
9282 \csuse{@glscompstyle@tree}%
9283 }%
```

Backward compatible treehypergroup style.

```
9284 \compatglossarystyle{treehypergroup}{%
9285 \csuse{@glscompstyle@tree}%
9286 }%
```

Backward compatible treenoname style.

```
9287 \compatglossarystyle{treenoname}{%
9288 \renewcommand{\glossaryentryfield}[5]{%
9289 \hangindent0pt\relax
9290 \parindent0pt\relax
9291 \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
9292 \ifx\relax##4\relax
9293 \else
9294 \space{##4}%
9295 \fi
9296 \space ##3\glspostdescription \space ##5\par}%

```

```

9297 \renewcommand{\glossarysubentryfield}[6]{%
9298   \hangindent##1\glstreeindent\relax
9299   \parindent##1\glstreeindent\relax
9300   \ifnum##1=1\relax
9301     \glssubentryitem{##2}%
9302   \fi
9303   \glstarget{##2}{\strut}%
9304   ##4\glspostdescription\space ##6\par}%
9305 }%

```

Backward compatible treenonamegroup style.

```

9306 \compatglossarystyle{treenonamegroup}{%
9307   \csuse{@glscompstyle@treenoname}%
9308 }%

```

Backward compatible treenonamehypergroup style.

```

9309 \compatglossarystyle{treenonamehypergroup}{%
9310   \csuse{@glscompstyle@treenoname}%
9311 }%

```

Backward compatible altree style.

```

9312 \compatglossarystyle{alttree}{%
9313   \renewcommand{\glossaryentryfield}[5]{%
9314     \ifnum \@gls@prevlevel=0\relax
9315     \else
9316       \settowidth{\glstreeindent}{\textbf{\@glswidestname\space}}%
9317       \hangindent\glstreeindent
9318       \parindent\glstreeindent
9319     \fi
9320     \makebox[0pt][r]{\makebox[\glstreeindent][l]{%
9321       \glssubentryitem{##1}\textbf{\glstarget{##1}{##2}}}}%
9322     \ifx\relax##4\relax
9323     \else
9324       (##4)\space
9325     \fi
9326     ##3\glspostdescription \space ##5\par
9327     \def\@gls@prevlevel{0}%
9328   }%
9329   \renewcommand{\glossarysubentryfield}[6]{%
9330     \ifnum##1=1\relax
9331       \glssubentryitem{##2}%
9332     \fi
9333     \ifnum \@gls@prevlevel=##1\relax
9334     \else
9335       \@ifundefined{@glswidestname\romannumeral##1}{%
9336         \settowidth{\gls@tmplen}{\textbf{\@glswidestname\space}}{%
9337         \settowidth{\gls@tmplen}{\textbf{%
9338           \csname @glswidestname\romannumeral##1\endcsname\space}}}%
9339       \ifnum \@gls@prevlevel<##1\relax
9340         \setlength\glstreeindent\gls@tmplen
9341         \addtolength\glstreeindent\parindent

```



```

9342     \parindent\glstreeindent
9343     \else
9344         \@ifundefined{@glswidestname\romannumeral\@gls@prevlevel}{%
9345             \settowidth{\glstreeindent}{\textbf{%
9346                 \@glswidestname\space}}{%
9347                 \settowidth{\glstreeindent}{\textbf{%
9348                     \csname @glswidestname\romannumeral\@gls@prevlevel
9349                         \endcsname\space}}}%
9350             \addtolength\parindent{-\glstreeindent}%
9351             \setlength\glstreeindent\parindent
9352     \fi
9353     \fi
9354     \hangindent\glstreeindent
9355     \makebox[0pt][r]{\makebox[\@gls@tmplen][l]{%
9356         \textbf{\glstarget{##2}{##3}}}}%
9357     \ifx##5\relax\relax
9358     \else
9359         (##5)\space
9360     \fi
9361     ##4\glspostdescription\space ##6\par
9362     \def\@gls@prevlevel{##1}%
9363 }%
9364 }%

```

Backward compatible alttreegroup style.

```

9365 \compatglossarystyle{alttreegroup}{%
9366 \csuse{@glscompstyle@almtree}%
9367 }%

```

Backward compatible alttreehypergroup style.

```

9368 \compatglossarystyle{alttreehypergroup}{%
9369 \csuse{@glscompstyle@almtree}%
9370 }%

```

Backward compatible mcolindex style.

```

9371 \compatglossarystyle{mcolindex}{%
9372 \csuse{@glscompstyle@index}%
9373 }%

```

Backward compatible mcolindexgroup style.

```

9374 \compatglossarystyle{mcolindexgroup}{%
9375 \csuse{@glscompstyle@index}%
9376 }%

```

Backward compatible mcolindexhypergroup style.

```

9377 \compatglossarystyle{mcolindexhypergroup}{%
9378 \csuse{@glscompstyle@index}%
9379 }%

```

Backward compatible mcoltree style.

```

9380 \compatglossarystyle{mcoltree}{%
9381 \csuse{@glscompstyle@tree}%

```

9382 }%

Backward compatible mcoltreegroup style.

9383 \compatglossarystyle{mcolindextreegroup}{%

9384 \csuse{@glscompstyle@tree}%

9385 }%

Backward compatible mcoltreehypergroup style.

9386 \compatglossarystyle{mcolindextreehypergroup}{%

9387 \csuse{@glscompstyle@tree}%

9388 }%

Backward compatible mcoltreenoname style.

9389 \compatglossarystyle{mcoltreenoname}{%

9390 \csuse{@glscompstyle@tree}%

9391 }%

Backward compatible mcoltreenonamegroup style.

9392 \compatglossarystyle{mcoltreenonamegroup}{%

9393 \csuse{@glscompstyle@tree}%

9394 }%

Backward compatible mcoltreenonamehypergroup style.

9395 \compatglossarystyle{mcoltreenonamehypergroup}{%

9396 \csuse{@glscompstyle@tree}%

9397 }%

Backward compatible mcolalmtree style.

9398 \compatglossarystyle{mcolalmtree}{%

9399 \csuse{@glscompstyle@almtree}%

9400 }%

Backward compatible mcolalmtreegroup style.

9401 \compatglossarystyle{mcolalmtreegroup}{%

9402 \csuse{@glscompstyle@almtree}%

9403 }%

Backward compatible mcolalmtreehypergroup style.

9404 \compatglossarystyle{mcolalmtreehypergroup}{%

9405 \csuse{@glscompstyle@almtree}%

9406 }%

Backward compatible superragged style.

9407 \compatglossarystyle{superragged}{%

9408 \renewcommand*{\glossaryentryfield}[5]{%

9409 \glsentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5%

9410 \tabularnewline}%

9411 \renewcommand*{\glossarysubentryfield}[6]{%

9412 &

9413 \glssubentryitem{##2}%

9414 \glstarget{##2}{\strut}##4\glspostdescription\space ##6%

9415 \tabularnewline}%

9416 }%

Backward compatible superraggedborder style.

```
9417 \compatglossarystyle{superraggedborder}{%
9418 \csuse{@glscompstyle@superragged}%
9419 }%
```

Backward compatible superraggedheader style.

```
9420 \compatglossarystyle{superraggedheader}{%
9421 \csuse{@glscompstyle@superragged}%
9422 }%
```

Backward compatible superraggedheaderborder style.

```
9423 \compatglossarystyle{superraggedheaderborder}{%
9424 \csuse{@glscompstyle@superragged}%
9425 }%
```

Backward compatible superragged3col style.

```
9426 \compatglossarystyle{superragged3col}{%
9427 \renewcommand*{\glossaryentryfield}[5]{%
9428 \glsentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%
9429 \renewcommand*{\glossarysubentryfield}[6]{%
9430 &
9431 \glssubentryitem{##2}%
9432 \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%
9433 }%
```

Backward compatible superragged3colborder style.

```
9434 \compatglossarystyle{superragged3colborder}{%
9435 \csuse{@glscompstyle@superragged3col}%
9436 }%
```

Backward compatible superragged3colheader style.

```
9437 \compatglossarystyle{superragged3colheader}{%
9438 \csuse{@glscompstyle@superragged3col}%
9439 }%
```

Backward compatible superragged3colheaderborder style.

```
9440 \compatglossarystyle{superragged3colheaderborder}{%
9441 \csuse{@glscompstyle@superragged3col}%
9442 }%
```

Backward compatible altsuperragged4col style.

```
9443 \compatglossarystyle{altsuperragged4col}{%
9444 \renewcommand*{\glossaryentryfield}[5]{%
9445 \glsentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%
9446 \renewcommand*{\glossarysubentryfield}[6]{%
9447 &
9448 \glssubentryitem{##2}%
9449 \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%
9450 }%
```

Backward compatible altsuperragged4colheader style.

```
9451 \compatglossarystyle{altsuperragged4colheader}{%
```

9452 \csuse{@glscompstyle@altsuperragged4col}%
9453 }%

Backward compatible altsuperragged4colborder style.

9454 \compatglossarystyle{altsuperragged4colborder}{%
9455 \csuse{@glscompstyle@altsuperragged4col}%
9456 }%

Backward compatible altsuperragged4colheaderborder style.

9457 \compatglossarystyle{altsuperragged4colheaderborder}{%
9458 \csuse{@glscompstyle@altsuperragged4col}%
9459 }%

Backward compatible super style.

9460 \compatglossarystyle{super}{%
9461 \renewcommand*{\glossaryentryfield}[5]{%
9462 \glsentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
9463 \renewcommand*{\glossarysubentryfield}[6]{%
9464 &
9465 \glssubentryitem{##2}%
9466 \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
9467 }%

Backward compatible superborder style.

9468 \compatglossarystyle{superborder}{%
9469 \csuse{@glscompstyle@super}%
9470 }%

Backward compatible superheader style.

9471 \compatglossarystyle{superheader}{%
9472 \csuse{@glscompstyle@super}%
9473 }%

Backward compatible superheaderborder style.

9474 \compatglossarystyle{superheaderborder}{%
9475 \csuse{@glscompstyle@super}%
9476 }%

Backward compatible super3col style.

9477 \compatglossarystyle{super3col}{%
9478 \renewcommand*{\glossaryentryfield}[5]{%
9479 \glsentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\\}%
9480 \renewcommand*{\glossarysubentryfield}[6]{%
9481 &
9482 \glssubentryitem{##2}%
9483 \glstarget{##2}{\strut}##4 & ##6\\}%
9484 }%

Backward compatible super3colborder style.

9485 \compatglossarystyle{super3colborder}{%
9486 \csuse{@glscompstyle@super3col}%
9487 }%

Backward compatible super3colheader style.

```
9488 \compatglossarystyle{super3colheader}{%
9489 \csuse{@glscompstyle@super3col}%
9490 }%
```

Backward compatible super3colheaderborder style.

```
9491 \compatglossarystyle{super3colheaderborder}{%
9492 \csuse{@glscompstyle@super3col}%
9493 }%
```

Backward compatible super4col style.

```
9494 \compatglossarystyle{super4col}{%
9495 \renewcommand*{\glossaryentryfield}[5]{%
9496 \glsentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%
9497 \renewcommand*{\glossarysubentryfield}[6]{%
9498 &
9499 \glssubentryitem{##2}%
9500 \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%
9501 }%
```

Backward compatible super4colheader style.

```
9502 \compatglossarystyle{super4colheader}{%
9503 \csuse{@glscompstyle@super4col}%
9504 }%
```

Backward compatible super4colborder style.

```
9505 \compatglossarystyle{super4colborder}{%
9506 \csuse{@glscompstyle@super4col}%
9507 }%
```

Backward compatible super4colheaderborder style.

```
9508 \compatglossarystyle{super4colheaderborder}{%
9509 \csuse{@glscompstyle@super4col}%
9510 }%
```

Backward compatible altsuper4col style.

```
9511 \compatglossarystyle{altsuper4col}{%
9512 \csuse{@glscompstyle@super4col}%
9513 }%
```

Backward compatible altsuper4colheader style.

```
9514 \compatglossarystyle{altsuper4colheader}{%
9515 \csuse{@glscompstyle@super4col}%
9516 }%
```

Backward compatible altsuper4colborder style.

```
9517 \compatglossarystyle{altsuper4colborder}{%
9518 \csuse{@glscompstyle@super4col}%
9519 }%
```

Backward compatible altsuper4colheaderborder style.

```
9520 \compatglossarystyle{altsuper4colheaderborder}{%
9521 \csuse{@glscompstyle@super4col}%
9522 }%
```

7 Accessibility Support (glossaries-accsupp Code)

The package is experimental. It is intended to provide a means of using the PDF accessibility support in glossary entries. See the documentation for further details about accessibility support.

```
9523 \NeedsTeXFormat{LaTeX2e}
```

Package version number now in line with main glossaries package number but will only be updated when glossaries-accsupp.sty is modified.

```
9524 \ProvidesPackage{glossaries-accsupp}[2014/07/30 v4.08 (NLCT)]
```

```
9525 Experimental glossaries accessibility]
```

Pass all options to glossaries:

```
9526 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
9527 \ProcessOptions
```

compatibleglossentry Override style compatibility macros:

```
9528 \def\compatibleglossentry#1#2{%
```

```
9529 \toks@{#2}%
```

```
9530 \protected@edef\@do@glossentry{%
```

```
9531 \noexpand\accsuppglossaryentryfield{#1}%
```

```
9532 {\noexpand\glsnamefont
```

```
9533 {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@name\endcsname}}%
```

```
9534 {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@desc\endcsname}}%
```

```
9535 {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@symbol\endcsname}}%
```

```
9536 {\the\toks@}%
```

```
9537 }%
```

```
9538 \@do@glossentry
```

```
9539 }
```

compatiblesubglossentry

```
9540 \def\compatiblesubglossentry#1#2#3{%
```

```
9541 \toks@{#3}%
```

```
9542 \protected@edef\@do@subglossentry{%
```

```
9543 \noexpand\accsuppglossarysubentryfield{\number#1}%
```

```
9544 {#2}%
```

```
9545 {\noexpand\glsnamefont
```

```
9546 {\expandafter\expandonce\csname glo@\glsdetoklabel{#2}@name\endcsname}}%
```

```
9547 {\expandafter\expandonce\csname glo@\glsdetoklabel{#2}@desc\endcsname}}%
```

```
9548 {\expandafter\expandonce\csname glo@\glsdetoklabel{#2}@symbol\endcsname}}%
```

```
9549 {\the\toks@}%
```

```
9550 }%
```

```
9551 \@do@subglossentry
```

```
9552 }
```

Required packages:

```
9553 \RequirePackage{glossaries}
```

```
9554 \RequirePackage{accsupp}
```

7.1 Defining Replacement Text

The version 0.1 stored the replacement text in the symbol key. This has been changed to use the new keys defined here. Example of use:

```
\newglossaryentry{dr}{name=Dr,description={},access={Doctor}}
```

access The replacement text corresponding to the name key:

```
9555 \define@key{glossentry}{access}{%
9556   \def\@glo@access{#1}%
9557 }
```

textaccess The replacement text corresponding to the text key:

```
9558 \define@key{glossentry}{textaccess}{%
9559   \def\@glo@textaccess{#1}%
9560 }
```

firstaccess The replacement text corresponding to the first key:

```
9561 \define@key{glossentry}{firstaccess}{%
9562   \def\@glo@firstaccess{#1}%
9563 }
```

pluralaccess The replacement text corresponding to the plural key:

```
9564 \define@key{glossentry}{pluralaccess}{%
9565   \def\@glo@pluralaccess{#1}%
9566 }
```

firstpluralaccess The replacement text corresponding to the firstplural key:

```
9567 \define@key{glossentry}{firstpluralaccess}{%
9568   \def\@glo@firstpluralaccess{#1}%
9569 }
```

symbolaccess The replacement text corresponding to the symbol key:

```
9570 \define@key{glossentry}{symbolaccess}{%
9571   \def\@glo@symbolaccess{#1}%
9572 }
```

symbolpluralaccess The replacement text corresponding to the symbolplural key:

```
9573 \define@key{glossentry}{symbolpluralaccess}{%
9574   \def\@glo@symbolpluralaccess{#1}%
9575 }
```

descriptionaccess The replacement text corresponding to the description key:

```
9576 \define@key{glossentry}{descriptionaccess}{%
9577   \def\@glo@descaccess{#1}%
9578 }
```

descriptionpluralaccess The replacement text corresponding to the descriptionplural key:

```
9579 \define@key{glossentry}{descriptionpluralaccess}{%
9580   \def\@glo@descpluralaccess{#1}%
9581 }
```

shortaccess The replacement text corresponding to the short key:

```
9582 \define@key{glossentry}{shortaccess}{%
9583   \def\@glo@shortaccess{#1}%
9584 }
```

shortpluralaccess The replacement text corresponding to the shortplural key:

```
9585 \define@key{glossentry}{shortpluralaccess}{%
9586   \def\@glo@shortpluralaccess{#1}%
9587 }
```

longaccess The replacement text corresponding to the long key:

```
9588 \define@key{glossentry}{longaccess}{%
9589   \def\@glo@longaccess{#1}%
9590 }
```

longpluralaccess The replacement text corresponding to the longplural key:

```
9591 \define@key{glossentry}{longpluralaccess}{%
9592   \def\@glo@longpluralaccess{#1}%
9593 }
```

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:

```
9594 \appto\@gls@keymap{,%
9595   {access}{access},%
9596   {textaccess}{textaccess},%
9597   {firstaccess}{firstaccess},%
9598   {pluralaccess}{pluralaccess},%
9599   {firstpluralaccess}{firstpluralaccess},%
9600   {symbolaccess}{symbolaccess},%
9601   {symbolpluralaccess}{symbolpluralaccess},%
9602   {descaccess}{descaccess},%
9603   {descpluralaccess}{descpluralaccess},%
9604   {shortaccess}{shortaccess},%
9605   {shortpluralaccess}{shortpluralaccess},%
9606   {longaccess}{longaccess},%
9607   {longpluralaccess}{longpluralaccess}%
9608 }
```

\@gls@noaccess Indicates that no replacement text has been provided.

```
9609 \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):

```
9610 \let\@gls@oldnewglossaryentryprehook\@newglossaryentryprehook
9611 \renewcommand*\@newglossaryentryprehook{%
9612   \@gls@oldnewglossaryentryprehook
9613   \def\@glo@access{\@glo@symbol}%
```

Initialise the other keys:

```
9614   \def\@glo@textaccess{\@glo@access}%
9615   \def\@glo@firstaccess{\@glo@access}%
9616   \def\@glo@pluralaccess{\@glo@textaccess}%
9617   \def\@glo@firstpluralaccess{\@glo@pluralaccess}%
9618   \def\@glo@symbolaccess{\relax}%
9619   \def\@glo@symbolpluralaccess{\@glo@symbolaccess}%
9620   \def\@glo@descaccess{\relax}%
9621   \def\@glo@descpluralaccess{\@glo@descaccess}%
9622   \def\@glo@shortaccess{\relax}%
9623   \def\@glo@shortpluralaccess{\@glo@shortaccess}%
9624   \def\@glo@longaccess{\relax}%
9625   \def\@glo@longpluralaccess{\@glo@longaccess}%
9626 }
```

Add to the end hook:

```
9627 \let\@gls@oldnewglossaryentryposthook\@newglossaryentryposthook
9628 \renewcommand*\@newglossaryentryposthook{%
9629   \@gls@oldnewglossaryentryposthook
```

Store the access information:

```
9630   \expandafter
9631     \protected@xdef\csname glo@\@glo@label @access\endcsname{%
9632     \@glo@access}%
9633   \expandafter
9634     \protected@xdef\csname glo@\@glo@label @textaccess\endcsname{%
9635     \@glo@textaccess}%
9636   \expandafter
9637     \protected@xdef\csname glo@\@glo@label @firstaccess\endcsname{%
9638     \@glo@firstaccess}%
9639   \expandafter
9640     \protected@xdef\csname glo@\@glo@label @pluralaccess\endcsname{%
9641     \@glo@pluralaccess}%
9642   \expandafter
9643     \protected@xdef\csname glo@\@glo@label @firstpluralaccess\endcsname{%
9644     \@glo@firstpluralaccess}%
9645   \expandafter
9646     \protected@xdef\csname glo@\@glo@label @symbolaccess\endcsname{%
9647     \@glo@symbolaccess}%
9648   \expandafter
9649     \protected@xdef\csname glo@\@glo@label @symbolpluralaccess\endcsname{%
9650     \@glo@symbolpluralaccess}%
9651   \expandafter
```

```

9652 \protected@xdef\csname glo@\@glo@label @descaccess\endcsname{%
9653 \@glo@descaccess}%
9654 \expandafter
9655 \protected@xdef\csname glo@\@glo@label @descpluralaccess\endcsname{%
9656 \@glo@descpluralaccess}%
9657 \expandafter
9658 \protected@xdef\csname glo@\@glo@label @shortaccess\endcsname{%
9659 \@glo@shortaccess}%
9660 \expandafter
9661 \protected@xdef\csname glo@\@glo@label @shortpluralaccess\endcsname{%
9662 \@glo@shortpluralaccess}%
9663 \expandafter
9664 \protected@xdef\csname glo@\@glo@label @longaccess\endcsname{%
9665 \@glo@longaccess}%
9666 \expandafter
9667 \protected@xdef\csname glo@\@glo@label @longpluralaccess\endcsname{%
9668 \@glo@longpluralaccess}%
9669 }

```

7.2 Accessing Replacement Text

`\glsentryaccess` Get the value of the access key for the entry with the given label:

```

9670 \newcommand*\glsentryaccess}[1]{%
9671 \@gls@entry@field{#1}{access}%
9672 }

```

`\glsentrytextaccess` Get the value of the textaccess key for the entry with the given label:

```

9673 \newcommand*\glsentrytextaccess}[1]{%
9674 \@gls@entry@field{#1}{textaccess}%
9675 }

```

`\glsentryfirstaccess` Get the value of the firstaccess key for the entry with the given label:

```

9676 \newcommand*\glsentryfirstaccess}[1]{%
9677 \@gls@entry@field{#1}{firstaccess}%
9678 }

```

`\glsentrypluralaccess` Get the value of the pluralaccess key for the entry with the given label:

```

9679 \newcommand*\glsentrypluralaccess}[1]{%
9680 \@gls@entry@field{#1}{pluralaccess}%
9681 }

```

`\glsentryfirstpluralaccess` Get the value of the firstpluralaccess key for the entry with the given label:

```

9682 \newcommand*\glsentryfirstpluralaccess}[1]{%
9683 \csname glo@#1@firstpluralaccess\endcsname
9684 }

```

`\glsentrysymbolaccess` Get the value of the symbolaccess key for the entry with the given label:

```

9685 \newcommand*\glsentrysymbolaccess}[1]{%

```

```
9686 \@gls@entry@field{#1}{symbolaccess}%
9687 }
```

symbolpluralaccess Get the value of the symbolpluralaccess key for the entry with the given label:

```
9688 \newcommand*{\glsentrysymbolpluralaccess}[1]{%
9689 \@gls@entry@field{#1}{symbolpluralaccess}%
9690 }
```

\glsentrydescaccess Get the value of the descriptionaccess key for the entry with the given label:

```
9691 \newcommand*{\glsentrydescaccess}[1]{%
9692 \@gls@entry@field{#1}{descaccess}%
9693 }
```

trydescpluralaccess Get the value of the descriptionpluralaccess key for the entry with the given label:

```
9694 \newcommand*{\glsentrydescpluralaccess}[1]{%
9695 \@gls@entry@field{#1}{descaccess}%
9696 }
```

glsentryshortaccess Get the value of the shortaccess key for the entry with the given label:

```
9697 \newcommand*{\glsentryshortaccess}[1]{%
9698 \@gls@entry@field{#1}{shortaccess}%
9699 }
```

ryshortpluralaccess Get the value of the shortpluralaccess key for the entry with the given label:

```
9700 \newcommand*{\glsentryshortpluralaccess}[1]{%
9701 \@gls@entry@field{#1}{shortpluralaccess}%
9702 }
```

\glsentrylongaccess Get the value of the longaccess key for the entry with the given label:

```
9703 \newcommand*{\glsentrylongaccess}[1]{%
9704 \@gls@entry@field{#1}{longaccess}%
9705 }
```

trylongpluralaccess Get the value of the longpluralaccess key for the entry with the given label:

```
9706 \newcommand*{\glsentrylongpluralaccess}[1]{%
9707 \@gls@entry@field{#1}{longpluralaccess}%
9708 }
```

```
\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.)

```
9709 \newcommand*{\glsaccsupp}[2]{%
9710 \BeginAccSupp{ActualText=#1}#2\EndAccSupp{}}%
9711 }
```

`\xglsaccsupp` Fully expands replacement text before calling `\glsaccsupp`

```
9712 \newcommand*\xglsaccsupp}[2]{%
9713   \protected@edef\@gls@replacementtext{#1}%
9714   \expandafter\glsaccsupp\expandafter{\@gls@replacementtext}{#2}%
9715 }
```

`@gls@access@display`

```
9716 \newcommand*\@gls@access@display}[2]{%
9717   \protected@edef\@glo@access{#2}%
9718   \ifx\@glo@access\@gls@noaccess
9719     #1%
9720   \else
9721     \xglsaccsupp{\@glo@access}{#1}%
9722   \fi
9723 }
```

`l$nameaccessdisplay` Displays the first argument with the accessibility text for the entry with the label given by the second argument (if set).

```
9724 \DeclareRobustCommand*\glsnameaccessdisplay}[2]{%
9725   \@gls@access@display{#1}{\glsentryaccess{#2}}%
9726 }
```

`l$textaccessdisplay` As above but for the `textaccess` replacement text.

```
9727 \DeclareRobustCommand*\gls$textaccessdisplay}[2]{%
9728   \@gls@access@display{#1}{\glsentrytextaccess{#2}}%
9729 }
```

`pl$pluralaccessdisplay` As above but for the `pluralaccess` replacement text.

```
9730 \DeclareRobustCommand*\glspluralaccessdisplay}[2]{%
9731   \@gls@access@display{#1}{\glsentrypluralaccess{#2}}%
9732 }
```

`sf$firstaccessdisplay` As above but for the `firstaccess` replacement text.

```
9733 \DeclareRobustCommand*\glsfirstaccessdisplay}[2]{%
9734   \@gls@access@display{#1}{\glsentryfirstaccess{#2}}%
9735 }
```

`pl$firstpluralaccessdisplay` As above but for the `firstpluralaccess` replacement text.

```
9736 \DeclareRobustCommand*\glsfirstpluralaccessdisplay}[2]{%
9737   \@gls@access@display{#1}{\glsentryfirstpluralaccess{#2}}%
9738 }
```

`symbolaccessdisplay` As above but for the `symbolaccess` replacement text.

```
9739 \DeclareRobustCommand*\gls$symbolaccessdisplay}[2]{%
9740   \@gls@access@display{#1}{\glsentrysymbolaccess{#2}}%
9741 }
```

pluralaccessdisplay As above but for the symbolpluralaccess replacement text.

```

9742 \DeclareRobustCommand*\glssymbolpluralaccessdisplay}[2]{%
9743   \@gls@access@display{#1}{\glsentrysymbolpluralaccess{#2}}}%
9744 }

```

descriptionaccessdisplay As above but for the descriptionaccess replacement text.

```

9745 \DeclareRobustCommand*\glsdescriptionaccessdisplay}[2]{%
9746   \@gls@access@display{#1}{\glsentrydescaccess{#2}}}%
9747 }

```

descriptionpluralaccessdisplay As above but for the descriptionpluralaccess replacement text.

```

9748 \DeclareRobustCommand*\glsdescriptionpluralaccessdisplay}[2]{%
9749   \@gls@access@display{#1}{\glsentrydescpluralaccess{#2}}}%
9750 }

```

shortaccessdisplay As above but for the shortaccess replacement text.

```

9751 \DeclareRobustCommand*\glsshortaccessdisplay}[2]{%
9752   \@gls@access@display{#1}{\glsentryshortaccess{#2}}}%
9753 }

```

shortpluralaccessdisplay As above but for the shortpluralaccess replacement text.

```

9754 \DeclareRobustCommand*\glsshortpluralaccessdisplay}[2]{%
9755   \@gls@access@display{#1}{\glsentryshortpluralaccess{#2}}}%
9756 }

```

longaccessdisplay As above but for the longaccess replacement text.

```

9757 \DeclareRobustCommand*\glslongaccessdisplay}[2]{%
9758   \@gls@access@display{#1}{\glsentrylongaccess{#2}}}%
9759 }

```

longpluralaccessdisplay As above but for the longpluralaccess replacement text.

```

9760 \DeclareRobustCommand*\glslongpluralaccessdisplay}[2]{%
9761   \@gls@access@display{#1}{\glsentrylongpluralaccess{#2}}}%
9762 }

```

\glsaccessdisplay Gets the replacement text corresponding to the named key given by the first argument and calls the appropriate command defined above.

```

9763 \DeclareRobustCommand*\glsaccessdisplay}[3]{%
9764   \@ifundefined{gls#1accessdisplay}%
9765   {%
9766     \PackageError{glossaries-accsupp}{No accessibility support
9767       for key ‘#1’}{%
9768     }%
9769   }%
9770   \csname gls#1accessdisplay\endcsname{#2}{#3}%
9771 }%
9772 }

```

ls@default@entryfmt Redefine the default entry format to use accessibility information

```
9773 \renewcommand*{\@@gls@default@entryfmt}[2]{%
```

```
9774   \ifdefempty\glscustomtext
```

```
9775   {%
```

```
9776     \glsifplural
```

```
9777   {%
```

Plural form

```
9778     \glscapscase
```

```
9779   {%
```

Don't adjust case

```
9780     \ifglsused\glslabel
```

```
9781   {%
```

Subsequent use

```
9782         #2{\glspluralaccessdisplay
```

```
9783            {\glsentryplural{\glslabel}}{\glslabel}}%
```

```
9784         {\glsdescriptionpluralaccessdisplay
```

```
9785            {\glsentrydescplural{\glslabel}}{\glslabel}}%
```

```
9786         {\glssymbolpluralaccessdisplay
```

```
9787            {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
```

```
9788         {\glsinsert}}%
```

```
9789         }%
```

```
9790         {%
```

First use

```
9791         #1{\glsfirstpluralaccessdisplay
```

```
9792            {\glsentryfirstplural{\glslabel}}{\glslabel}}%
```

```
9793         {\glsdescriptionpluralaccessdisplay
```

```
9794            {\glsentrydescplural{\glslabel}}{\glslabel}}%
```

```
9795         {\glssymbolpluralaccessdisplay
```

```
9796            {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
```

```
9797         {\glsinsert}}%
```

```
9798         }%
```

```
9799         }%
```

```
9800         {%
```

Make first letter upper case

```
9801     \ifglsused\glslabel
```

```
9802   {%
```

Subsequent use.

```
9803         #2{\glspluralaccessdisplay
```

```
9804            {\Glsentryplural{\glslabel}}{\glslabel}}%
```

```
9805         {\glsdescriptionpluralaccessdisplay
```

```
9806            {\Glsentrydescplural{\glslabel}}{\glslabel}}%
```

```
9807         {\glssymbolpluralaccessdisplay
```

```
9808            {\Glsentrysymbolplural{\glslabel}}{\glslabel}}%
```

```
9809         {\glsinsert}}%
```

```
9810         }%
```

```
9811         {%
```

First use

```
9812      #1{\glsfirstpluralaccessdisplay
9813         {\Glsentryfirstplural{\glslabel}}{\glslabel}}%
9814         {\glsdescriptionpluralaccessdisplay
9815          {\glsentrydescplural{\glslabel}}{\glslabel}}%
9816         {\glssymbolpluralaccessdisplay
9817          {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
9818         {\glsinsert}}%
9819     }%
9820 }%
9821 {%
```

Make all upper case

```
9822     \ifglsused\glslabel
9823     {%
```

Subsequent use

```
9824     \MakeUppercase{%
9825     #2{\glspluralaccessdisplay
9826        {\glsentryplural{\glslabel}}{\glslabel}}%
9827        {\glsdescriptionpluralaccessdisplay
9828         {\glsentrydescplural{\glslabel}}{\glslabel}}%
9829        {\glssymbolpluralaccessdisplay
9830         {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
9831        {\glsinsert}}%
9832 }%
9833 {%
```

First use

```
9834     \MakeUppercase{%
9835     #1{\glsfirstpluralaccessdisplay
9836        {\glsentryfirstplural{\glslabel}}{\glslabel}}%
9837        {\glsdescriptionpluralaccessdisplay
9838         {\glsentrydescplural{\glslabel}}{\glslabel}}%
9839        {\glssymbolpluralaccessdisplay
9840         {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
9841        {\glsinsert}}%
9842 }%
9843 }%
9844 }%
9845 {%
```

Singular form

```
9846     \glscapscase
9847     {%
```

Don't adjust case

```
9848     \ifglsused\glslabel
9849     {%
```

Subsequent use

```

9850         #2{\glstextaccessdisplay
9851             {\glentrytext{\glslabel}}{\glslabel}}%
9852         {\glsdescriptionaccessdisplay
9853             {\glentrydesc{\glslabel}}{\glslabel}}%
9854         {\glssymbolaccessdisplay
9855             {\glentrysymbol{\glslabel}}{\glslabel}}%
9856         {\glsinsert}%
9857     }%
9858     {%

```

First use

```

9859         #1{\glsfirstaccessdisplay
9860             {\glentryfirst{\glslabel}}{\glslabel}}%
9861         {\glsdescriptionaccessdisplay
9862             {\glentrydesc{\glslabel}}{\glslabel}}%
9863         {\glssymbolaccessdisplay
9864             {\glentrysymbol{\glslabel}}{\glslabel}}%
9865         {\glsinsert}%
9866     }%
9867 }%
9868 {%

```

Make first letter upper case

```

9869     \ifglsused\glslabel
9870     {%

```

Subsequent use

```

9871         #2{\glstextaccessdisplay
9872             {\glentrytext{\glslabel}}{\glslabel}}%
9873         {\glsdescriptionaccessdisplay
9874             {\glentrydesc{\glslabel}}{\glslabel}}%
9875         {\glssymbolaccessdisplay
9876             {\glentrysymbol{\glslabel}}{\glslabel}}%
9877         {\glsinsert}%
9878     }%
9879     {%

```

First use

```

9880         #1{\glsfirstaccessdisplay
9881             {\glentryfirst{\glslabel}}{\glslabel}}%
9882         {\glsdescriptionaccessdisplay
9883             {\glentrydesc{\glslabel}}{\glslabel}}%
9884         {\glssymbolaccessdisplay
9885             {\glentrysymbol{\glslabel}}{\glslabel}}%
9886         {\glsinsert}%
9887     }%
9888 }%
9889 {%

```

Make all upper case

```

9890     \ifglsused\glslabel
9891     {%

```


Subsequent use

```
9892     \MakeUppercase{%
9893         #2{\glstextaccessdisplay
9894             {\glstext{\glslabel}}{\glslabel}}%
9895         {\glsdescriptionaccessdisplay
9896             {\glsentrydesc{\glslabel}}{\glslabel}}%
9897         {\glssymbolaccessdisplay
9898             {\glstentrysymbol{\glslabel}}{\glslabel}}%
9899         {\glsinsert}}%
9900     }%
9901     {%
```

First use

```
9902     \MakeUppercase{%
9903         #1{\glsfirstaccessdisplay
9904             {\glstentryfirst{\glslabel}}{\glslabel}}%
9905         {\glsdescriptionaccessdisplay
9906             {\glsentrydesc{\glslabel}}{\glslabel}}%
9907         {\glssymbolaccessdisplay
9908             {\glstentrysymbol{\glslabel}}{\glslabel}}%
9909         {\glsinsert}}%
9910     }%
9911     }%
9912     }%
9913     }%
9914     {%
```

Custom text provided in \glsdisp

```
9915     \ifglsused{\glslabel}%
9916     {%
```

Subsequent use

```
9917     #2{\glscustomtext}%
9918     {\glsdescriptionaccessdisplay
9919         {\glsentrydesc{\glslabel}}{\glslabel}}%
9920     {\glssymbolaccessdisplay
9921         {\glstentrysymbol{\glslabel}}{\glslabel}}%
9922     {\glsinsert}%
9923     }%
9924     {%
```

First use

```
9925     #1{\glscustomtext}%
9926     {\glsdescriptionaccessdisplay
9927         {\glsentrydesc{\glslabel}}{\glslabel}}%
9928     {\glssymbolaccessdisplay
9929         {\glstentrysymbol{\glslabel}}{\glslabel}}%
9930     {\glsinsert}%
9931     }%
9932     }%
9933 }
```

`\glsgenentryfmt` Redefine to use accessibility information.

```
9934 \renewcommand*{\glsgenentryfmt}{%
9935   \ifdefempty\glscustomtext
9936   {%
9937     \glusifplural
9938     {%
```

Plural form

```
9939     \glscapscase
9940     {%
```

Don't adjust case

```
9941     \ifglused\glslabel
9942     {%
```

Subsequent use

```
9943     \glspluralaccessdisplay
9944     {\glsglentryplural{\glslabel}}{\glslabel}%
9945     \glsinsert
9946     }%
9947     {%
```

First use

```
9948     \glsfirstpluralaccessdisplay
9949     {\glsglentryfirstplural{\glslabel}}{\glslabel}%
9950     \glsinsert
9951     }%
9952     }%
9953     {%
```

Make first letter upper case

```
9954     \ifglused\glslabel
9955     {%
```

Subsequent use.

```
9956     \glspluralaccessdisplay
9957     {\Glsentryplural{\glslabel}}{\glslabel}%
9958     \glsinsert
9959     }%
9960     {%
```

First use

```
9961     \glsfirstpluralaccessdisplay
9962     {\Glsentryfirstplural{\glslabel}}{\glslabel}%
9963     \glsinsert
9964     }%
9965     }%
9966     {%
```

Make all upper case

```
9967     \ifglused\glslabel
9968     {%
```

Subsequent use

```
9969      \glspluralaccessdisplay
9970      {\mfirstucMakeUppercase{\glsentryplural{\glslabel}}}%
9971      {\glslabel}%
9972      \mfirstucMakeUppercase{\glsinsert}%
9973      }%
9974      {%
```

First use

```
9975      \glsfirstpluralaccessdisplay
9976      {\mfirstucMakeUppercase{\glsentryfirstplural{\glslabel}}}%
9977      {\glslabel}%
9978      \mfirstucMakeUppercase{\glsinsert}%
9979      }%
9980      }%
9981      }%
9982      {%
```

Singular form

```
9983      \glscapscale
9984      {%
```

Don't adjust case

```
9985      \ifglsused\glslabel
9986      {%
```

Subsequent use

```
9987      \glstextaccessdisplay{\glsentrytext{\glslabel}}{\glslabel}%
9988      \glsinsert
9989      }%
9990      {%
```

First use

```
9991      \glsfirstaccessdisplay{\glsentryfirst{\glslabel}}{\glslabel}%
9992      \glsinsert
9993      }%
9994      }%
9995      {%
```

Make first letter upper case

```
9996      \ifglsused\glslabel
9997      {%
```

Subsequent use

```
9998      \glstextaccessdisplay{\Glsentrytext{\glslabel}}{\glslabel}%
9999      \glsinsert
10000     }%
10001     {%
```

First use

```
10002     \glsfirstaccessdisplay{\Glsentryfirst{\glslabel}}{\glslabel}%
10003     \glsinsert
```

```

10004     }%
10005     }%
10006     {%

```

Make all upper case

```

10007     \ifglsused\glslabel
10008     {%

```

Subsequent use

```

10009     \glstextaccessdisplay
10010     {\mfirstucMakeUppercase{\glsentrytext{\glslabel}}}{\glslabel}%
10011     \mfirstucMakeUppercase{\glsinsert}%
10012     }%
10013     {%

```

First use

```

10014     \glsfirstaccessdisplay
10015     {\mfirstucMakeUppercase{\glsentryfirst{\glslabel}}}{\glslabel}%
10016     \mfirstucMakeUppercase{\glsinsert}%
10017     }%
10018     }%
10019     }%
10020     }%
10021     {%

```

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.

```

10022     \glscustomtext\glsinsert
10023     }%
10024 }

```

`\glsгенacfmt` Redefine to include accessibility information.

```

10025 \renewcommand*{\glsгенacfmt}{%
10026   \ifdefempty\glscustomtext
10027   {%
10028     \ifglsused\glslabel
10029     {%

```

Subsequent use:

```

10030     \glsifplural
10031     {%

```

Subsequent plural form:

```

10032     \glscapscase
10033     {%

```

Subsequent plural form, don't adjust case:

```

10034     \acronymfont
10035     {\glsshortpluralaccessdisplay
10036     {\glsentryshortpl{\glslabel}}{\glslabel}}%
10037     \glsinsert

```

10038 }%
10039 {%

Subsequent plural form, make first letter upper case:

10040 \acronymfont
10041 {\glsshortpluralaccessdisplay
10042 {\Glsentryshortpl{\glslabel}}{\glslabel}}%
10043 \glsinsert
10044 }%
10045 {%

Subsequent plural form, all caps:

10046 \mfirstucMakeUppercase
10047 {\acronymfont
10048 {\glsshortpluralaccessdisplay
10049 {\Glsentryshortpl{\glslabel}}{\glslabel}}%
10050 \glsinsert}%
10051 }%
10052 }%
10053 {%

Subsequent singular form

10054 \glscapscase
10055 {%

Subsequent singular form, don't adjust case:

10056 \acronymfont
10057 {\glsshortaccessdisplay{\Glsentryshort{\glslabel}}{\glslabel}}%
10058 \glsinsert
10059 }%
10060 {%

Subsequent singular form, make first letter upper case:

10061 \acronymfont
10062 {\glsshortaccessdisplay{\Glsentryshort{\glslabel}}{\glslabel}}%
10063 \glsinsert
10064 }%
10065 {%

Subsequent singular form, all caps:

10066 \mfirstucMakeUppercase
10067 {\acronymfont{%
10068 \glsshortaccessdisplay{\Glsentryshort{\glslabel}}{\glslabel}}%
10069 \glsinsert}%
10070 }%
10071 }%
10072 }%
10073 {%

First use:

10074 \glsifplural
10075 {%

First use plural form:

```
10076      \glscapscase
10077      {%
```

First use plural form, don't adjust case:

```
10078      \genplacrfullformat{\glslabel}{\glsinsert}%
10079      }%
10080      {%
```

First use plural form, make first letter upper case:

```
10081      \Genplacrfullformat{\glslabel}{\glsinsert}%
10082      }%
10083      {%
```

First use plural form, all caps:

```
10084      \mfirstucMakeUppercase
10085      {\genplacrfullformat{\glslabel}{\glsinsert}}%
10086      }%
10087      }%
10088      {%
```

First use singular form

```
10089      \glscapscase
10090      {%
```

First use singular form, don't adjust case:

```
10091      \genacrfullformat{\glslabel}{\glsinsert}%
10092      }%
10093      {%
```

First use singular form, make first letter upper case:

```
10094      \Genacrfullformat{\glslabel}{\glsinsert}%
10095      }%
10096      {%
```

First use singular form, all caps:

```
10097      \mfirstucMakeUppercase
10098      {\genacrfullformat{\glslabel}{\glsinsert}}%
10099      }%
10100     }%
10101     }%
10102     }%
10103     {%
```

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.

```
10104     \glscustomtext
10105     }%
10106 }
```

`\genacrfullformat` Redefine to include accessibility information.

```
10107 \renewcommand*{\genacrfullformat}[2]{%
```

```

10108 \glslongaccessdisplay{\glsentrylong{#1}}{#1}#2\space
10109 (\glsshortaccessdisplay{\protect\firstacronymfont{\glsentryshort{#1}}}{#1})%
10110 }

```

`\Genacrfullformat` Redefine to include accessibility information.

```

10111 \renewcommand*{\Genacrfullformat}[2]{%
10112 \glslongaccessdisplay{\Glsentrylong{#1}}{#1}#2\space
10113 (\glsshortaccessdisplay{\protect\firstacronymfont{\Glsentryshort{#1}}}{#1})%
10114 }

```

`\genplacrfullformat` Redefine to include accessibility information.

```

10115 \renewcommand*{\genplacrfullformat}[2]{%
10116 \glslongpluralaccessdisplay{\glsentrylongpl{#1}}{#1}#2\space
10117 (\glsshortpluralaccessdisplay
10118 {\protect\firstacronymfont{\glsentryshortpl{#1}}}{#1})%
10119 }

```

`\Genplacrfullformat` Redefine to include accessibility information.

```

10120 \renewcommand*{\Genplacrfullformat}[2]{%
10121 \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}#2\space
10122 (\glsshortpluralaccessdisplay
10123 {\protect\firstacronymfont{\Glsentryshortpl{#1}}}{#1})%
10124 }

```

`\@acrshort`

```

10125 \def\@acrshort#1#2[#3]{%
10126 \glsdoifexists{#2}%
10127 {%
10128 \let\do@gls@link@checkfirsthyper\relax

10129 \let\glsifplural\@secondoftwo
10130 \let\glsifcaps\@firstofthree
10131 \let\glsinsert\@empty
10132 \def\glscustomtext{%
10133 \acronymfont{\glsshortaccessdisplay{\glsentryshort{#2}}{#2}}#3%
10134 }%

```

Call `\@gls@link`

```

10135 \@gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
10136 }%
10137 }

```

`\@Acrshort`

```

10138 \def\@Acrshort#1#2[#3]{%
10139 \glsdoifexists{#2}%
10140 {%
10141 \let\do@gls@link@checkfirsthyper\relax

```

```

10142 \let\glsifplural\@secondoftwo
10143 \let\glscapscase\@secondofthree
10144 \let\glsinsert\@empty
10145 \def\glscustomtext{%
10146 \acronymfont{\glsshortaccessdisplay{\Glsentryshort{#2}}{#2}}#3%
10147 }%

```

Call \@gls@link

```

10148 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10149 }%
10150 }

```

\@ACRshort

```

10151 \def\@ACRshort#1#2[#3]{%
10152 \glsdoifexists{#2}%
10153 {%
10154 \let\do@gls@link@checkfirsthyper\relax

10155 \let\glsifplural\@secondoftwo
10156 \let\glscapscase\@thirdofthree
10157 \let\glsinsert\@empty
10158 \def\glscustomtext{%
10159 \acronymfont{\glsshortaccessdisplay
10160 {\MakeUppercase{\glsentryshort{#2}}}{#2}}#3%
10161 }%

```

Call \@gls@link

```

10162 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10163 }%
10164 }

```

\@acrlong

```

10165 \def\@acrlong#1#2[#3]{%
10166 \glsdoifexists{#2}%
10167 {%
10168 \let\do@gls@link@checkfirsthyper\relax

10169 \let\glsifplural\@secondoftwo
10170 \let\glscapscase\@firstofthree
10171 \let\glsinsert\@empty
10172 \def\glscustomtext{%
10173 \acronymfont{\glslongaccessdisplay{\glsentrylong{#2}}{#2}}#3%
10174 }%

```

Call \@gls@link

```

10175 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10176 }%
10177 }

```


`\@Acrlong`

```
10178 \def\@Acrlong#1#2[#3]{%
10179   \glsdoifexists{#2}%
10180   {%
10181     \let\do@gls@link@checkfirsthyper\relax

10182     \let\glsifplural\@secondoftwo
10183     \let\glscapscase\@firstofthree
10184     \let\glsinsert\@empty
10185     \def\glscustomtext{%
10186       \acronymfont{\glslongaccessdisplay{\Glsentrylong{#2}}{#2}}#3%
10187     }%
```

Call `\@gls@link`

```
10188   \@gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
10189   }%
10190 }
```

`\@ACRlong`

```
10191 \def\@ACRlong#1#2[#3]{%
10192   \glsdoifexists{#2}%
10193   {%
10194     \let\do@gls@link@checkfirsthyper\relax

10195     \let\glsifplural\@secondoftwo
10196     \let\glscapscase\@firstofthree
10197     \let\glsinsert\@empty
10198     \def\glscustomtext{%
10199       \acronymfont{\glslongaccessdisplay{%
10200         \MakeUppercase{\glsentrylong{#2}}}{#2}}#3}%
10201     }%
```

Call `\@gls@link`

```
10202   \@gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
10203   }%
10204 }
```

7.3 Displaying the Glossary

We need to redefine the way the glossary entries are formatted to include the accessibility support. The predefined glossary styles use `\glossentryname`, `\glossentrydesc` and `\glossentrysymbol`, but we need to provide compatibility with earlier versions in case users have defined their own styles using `\accsuppglossaryentryfield` and `\accsuppglossarysubentryfield`.

Now redefine `\glossentryname`, `\glossentrydesc` and `\glossentrysymbol` etc so they use the accessibility stuff.

```
10205 \renewcommand*{\glossentryname}[1]{%
10206   \glsdoifexists{#1}%
10207   {%
```

```

10208 \glsnamefont{\glsnameaccessdisplay{\glsentryname{#1}}{#1}}%
10209 }%
10210 }

10211 \renewcommand*{\glossentryname}[1]{%
10212 \glsdoifexists{#1}%
10213 {%
10214 \glsnamefont{\glsnameaccessdisplay{\glsentryname{#1}}{#1}}%
10215 }%
10216 }

10217 \renewcommand*{\glossentrydesc}[1]{%
10218 \glsdoifexists{#1}%
10219 {%
10220 \glsdescriptionaccessdisplay{\glsentrydesc{#1}}{#1}%
10221 }%
10222 }

10223 \renewcommand*{\Glossentrydesc}[1]{%
10224 \glsdoifexists{#1}%
10225 {%
10226 \glsdescriptionaccessdisplay{\Glsentrydesc{#1}}{#1}%
10227 }%
10228 }

10229 \renewcommand*{\glossentrysymbol}[1]{%
10230 \glsdoifexists{#1}%
10231 {%
10232 \glsymbolaccessdisplay{\glsentrysymbol{#1}}{#1}%
10233 }%
10234 }

10235 \renewcommand*{\Glossentrysymbol}[1]{%
10236 \glsdoifexists{#1}%
10237 {%
10238 \glsymbolaccessdisplay{\Glsentrysymbol{#1}}{#1}%
10239 }%
10240 }

```

pglossaryentryfield

```

10241 \newcommand*{\accsuppglossaryentryfield}[5]{%
10242 \glossaryentryfield{#1}%
10243 {\glsnameaccessdisplay{#2}{#1}}%
10244 {\glsdescriptionaccessdisplay{#3}{#1}}%
10245 {\glsymbolaccessdisplay{#4}{#1}}{#5}%
10246 }

```

glossarysubentryfield

```

10247 \newcommand*{\accsuppglossarysubentryfield}[6]{%
10248 \glossarysubentryfield{#1}{#2}%
10249 {\glsnameaccessdisplay{#3}{#2}}%
10250 {\glsdescriptionaccessdisplay{#4}{#2}}%

```

```
10251 {\glssymbolaccessdisplay{#5}{#2}}{#6}%
10252 }
```

7.4 Acronyms

Redefine acronym styles provided by glossaries:

long-short *<long>* (*<short>*) acronym style.

```
10253 \renewacronymstyle{long-short}%
10254 {%
```

Check for long form in case this is a mixed glossary.

```
10255 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
10256 }%
10257 {%
10258 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
10259 \renewcommand*{\genacrfullformat}[2]{%
10260 \glslongaccessdisplay{\glsentrylong{##1}}{##1}##2\space
10261 (\glsshortaccessdisplay
10262 {\protect\firstacronymfont{\glsentryshort{##1}}}{##1})%
10263 }%
10264 \renewcommand*{\Genacrfullformat}[2]{%
10265 \glslongaccessdisplay{\Glsentrylong{##1}}{##1}##2\space
10266 (\glsshortaccessdisplay
10267 {\protect\firstacronymfont{\glsentryshort{##1}}}{##1})%
10268 }%
10269 \renewcommand*{\genplacrfullformat}[2]{%
10270 \glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}##2\space
10271 (\glsshortpluralaccessdisplay
10272 {\protect\firstacronymfont{\glsentryshortpl{##1}}}{##1})%
10273 }%
10274 \renewcommand*{\Genplacrfullformat}[2]{%
10275 \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}##2\space
10276 (\glsshortpluralaccessdisplay
10277 {\protect\firstacronymfont{\glsentryshortpl{##1}}}{##1})%
10278 }%
10279 \renewcommand*{\acronymentry}[1]{%
10280 \glsshortaccessdisplay{acronymfont{\glsentryshort{##1}}}{##1}}
10281 \renewcommand*{\acronymsort}[2]{##1}%
10282 \renewcommand*{\acronymfont}[1]{##1}%
10283 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
10284 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
10285 }
```

short-long *<short>* (*<long>*) acronym style.

```
10286 \renewacronymstyle{short-long}%
10287 {%
```

Check for long form in case this is a mixed glossary.

```
10288 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
```

```

10289 }%
10290 {%
10291 \renewcommand*\GenericAcronymFields{description={\the\glslongtok}}%
10292 \renewcommand*\genacrfullformat}[2]{%
10293 \glsshortaccessdisplay
10294 {\protect\firstacronymfont{\glsentryshort{##1}}{##1}##2\space
10295 (\glslongaccessdisplay{\glsentrylong{##1}}{##1})%
10296 }%
10297 \renewcommand*\Genacrfullformat}[2]{%
10298 \glsshortaccessdisplay
10299 {\protect\firstacronymfont{\Glsentryshort{##1}}{##1}##2\space
10300 (\glslongaccessdisplay{\glsentrylong{##1}}{##1})%
10301 }%
10302 \renewcommand*\genplacrfullformat}[2]{%
10303 \glsshortpluralaccessdisplay
10304 {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}##2\space
10305 (\glslongpluralaccessdisplay
10306 {\glsentrylongpl{##1}}{##1})%
10307 }%
10308 \renewcommand*\Genplacrfullformat}[2]{%
10309 \glsshortpluralaccessdisplay
10310 {\protect\firstacronymfont{\Glsentryshortpl{##1}}{##1}##2\space
10311 (\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1})%
10312 }%
10313 \renewcommand*\acronymentry}[1]{%
10314 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}{##1}}%
10315 \renewcommand*\acronymsort}[2]{##1}%
10316 \renewcommand*\acronymfont}[1]{##1}%
10317 \renewcommand*\firstacronymfont}[1]{\acronymfont{##1}}%
10318 \renewcommand*\acrpluralsuffix{\glspluralsuffix}%
10319 }

```

long-short-desc *⟨long⟩* (*{⟨short⟩}*) acronym style that has an accompanying description (which the user needs to supply).

```

10320 \renewacronymstyle{long-short-desc}%
10321 {%
10322 \GlsUseAcrEntryDispStyle{long-short}%
10323 }%
10324 {%
10325 \GlsUseAcrStyleDefs{long-short}%
10326 \renewcommand*\GenericAcronymFields{}%
10327 \renewcommand*\acronymsort}[2]{##2}%
10328 \renewcommand*\acronymentry}[1]{%
10329 \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10330 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}{##1}})%
10331 }

```

long-sc-short-desc *⟨long⟩* (\textsc{⟨short⟩}) acronym style that has an accompanying description (which the user needs to supply).

```

10332 \renewacronymstyle{long-sc-short-desc}%
10333 {%
10334   \GlsUseAcrEntryDispStyle{long-sc-short}%
10335 }%
10336 {%
10337   \GlsUseAcrStyleDefs{long-sc-short}%
10338   \renewcommand*\GenericAcronymFields{}%
10339   \renewcommand*\acronymsort}[2]{##2}%
10340   \renewcommand*\acronymentry}[1]{%
10341     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10342     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10343 }

```

long-sm-short-desc *long* (\textsmaller{*short*}) acronym style that has an accompanying description (which the user needs to supply).

```

10344 \renewacronymstyle{long-sm-short-desc}%
10345 {%
10346   \GlsUseAcrEntryDispStyle{long-sm-short}%
10347 }%
10348 {%
10349   \GlsUseAcrStyleDefs{long-sm-short}%
10350   \renewcommand*\GenericAcronymFields{}%
10351   \renewcommand*\acronymsort}[2]{##2}%
10352   \renewcommand*\acronymentry}[1]{%
10353     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10354     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10355 }

```

short-long-desc *short* ({*long*}) acronym style that has an accompanying description (which the user needs to supply).

```

10356 \renewacronymstyle{short-long-desc}%
10357 {%
10358   \GlsUseAcrEntryDispStyle{short-long}%
10359 }%
10360 {%
10361   \GlsUseAcrStyleDefs{short-long}%
10362   \renewcommand*\GenericAcronymFields{}%
10363   \renewcommand*\acronymsort}[2]{##2}%
10364   \renewcommand*\acronymentry}[1]{%
10365     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10366     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10367 }

```

sc-short-long-desc *long* (\textsc{*short*}) acronym style that has an accompanying description (which the user needs to supply).

```

10368 \renewacronymstyle{sc-short-long-desc}%
10369 {%
10370   \GlsUseAcrEntryDispStyle{sc-short-long}%
10371 }%

```

```

10372 {%
10373   \GlsUseAcrStyleDefs{sc-short-long}%
10374   \renewcommand*\GenericAcronymFields{}%
10375   \renewcommand*\acronymsort}[2]{##2}%
10376   \renewcommand*\acronymentry}[1]{%
10377     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10378     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10379 }

```

sm-short-long-desc *<long>* (`\textsmaller{<short>}`) acronym style that has an accompanying description (which the user needs to supply).

```

10380 \renewacronymstyle{sm-short-long-desc}%
10381 {%
10382   \GlsUseAcrEntryDispStyle{sm-short-long}%
10383 }%
10384 {%
10385   \GlsUseAcrStyleDefs{sm-short-long}%
10386   \renewcommand*\GenericAcronymFields{}%
10387   \renewcommand*\acronymsort}[2]{##2}%
10388   \renewcommand*\acronymentry}[1]{%
10389     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10390     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10391 }

```

dua *<long>* only acronym style.

```

10392 \renewacronymstyle{dua}%
10393 {%

```

Check for long form in case this is a mixed glossary.

```

10394   \ifdefempty\glscustomtext
10395   {%
10396     \ifglshaslong{\glslabel}%
10397     {%
10398       \glsifplural
10399       {%

```

Plural form:

```

10400         \glscapscase
10401         {%

```

Plural form, don't adjust case:

```

10402         \glslongpluralaccessdisplay{\glsentrylongpl{\glslabel}}{\glslabel}%
10403         \glsinsert
10404         }%
10405         {%

```

Plural form, make first letter upper case:

```

10406         \glslongpluralaccessdisplay{\Glsentrylongpl{\glslabel}}{\glslabel}%
10407         \glsinsert
10408         }%
10409         {%

```

Plural form, all caps:

```
10410      \glslongpluralaccessdisplay
10411      {\mfirstucMakeUppercase{\glsentrylongpl{\glslabel}}}{\glslabel}%
10412      \mfirstucMakeUppercase{\glsinsert}%
10413      }%
10414      }%
10415      {%
```

Singular form

```
10416      \glschapscase
10417      {%
```

Singular form, don't adjust case:

```
10418      \glslongaccessdisplay{\glsentrylong{\glslabel}}{\glslabel}\glsinsert
10419      }%
10420      {%
```

Subsequent singular form, make first letter upper case:

```
10421      \glslongaccessdisplay{\Glsentrylong{\glslabel}}{\glslabel}\glsinsert
10422      }%
10423      {%
```

Subsequent singular form, all caps:

```
10424      \glslongaccessdisplay
10425      {\mfirstucMakeUppercase
10426      {\glsentrylong{\glslabel}\glsinsert}}{\glslabel}%
10427      \mfirstucMakeUppercase{\glsinsert}%
10428      }%
10429      }%
10430      }%
10431      {%
```

Not an acronym:

```
10432      \glsgenentryfmt
10433      }%
10434      }%
10435      {\glscustomtext\glsinsert}%
10436      }%
10437      {%
10438      \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
10439      \renewcommand*{\acrfullfmt}[3]{%
10440      \glslink[##1]{##2}{%
10441      \glslongaccessdisplay{\glsentrylong{##2}}{##2}##3\space
10442      (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2)}}}%
10443      \renewcommand*{\Acrfullfmt}[3]{%
10444      \glslink[##1]{##2}{%
10445      \glslongaccessdisplay{\Glsentrylong{##2}}{##2}##3\space
10446      (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2)}}}%
10447      \renewcommand*{\ACRfullfmt}[3]{%
10448      \glslink[##1]{##2}{%
10449      \glslongaccessdisplay
```

```

10450     {\mfirstucMakeUppercase{\glsentrylong{##2}}{##2}##3\space
10451     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}{##2}})}%
10452 \renewcommand*{\acrfullplfmt}[3]{%
10453   \glslink[##1]{##2}{%
10454     \glslongpluralaccessdisplay
10455     {\glsentrylongpl{##2}}{##2}##3\space
10456     (\glsshortpluralaccessdisplay
10457     {\acronymfont{\glsentryshortpl{##2}}{##2}})}%
10458 \renewcommand*{\Acrfullplfmt}[3]{%
10459   \glslink[##1]{##2}{%
10460     \glslongpluralaccessdisplay
10461     {\Glsentrylongpl{##2}}{##2}##3\space
10462     (\glsshortpluralaccessdisplay
10463     {\acronymfont{\glsentryshortpl{##2}}{##2}})}%
10464 \renewcommand*{\ACRfullplfmt}[3]{%
10465   \glslink[##1]{##2}{%
10466     \glslongpluralaccessdisplay
10467     {\mfirstucMakeUppercase{\glsentrylongpl{##2}}{##2}##3\space
10468     (\glsshortpluralaccessdisplay
10469     {\acronymfont{\glsentryshortpl{##2}}{##2}})}%
10470 \renewcommand*{\glsentryfull}[1]{%
10471   \glslongaccessdisplay{\glsentrylong{##1}}\space
10472   (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}{##1}})%
10473 }%
10474 \renewcommand*{\Glsentryfull}[1]{%
10475   \glslongaccessdisplay{\Glsentrylong{##1}}{##1}\space
10476   (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}{##1}})%
10477 }%
10478 \renewcommand*{\glsentryfullpl}[1]{%
10479   \glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}\space
10480   (\glsshortpluralaccessdisplay{\acronymfont{\glsentryshortpl{##1}}{##1}})%
10481 }%
10482 \renewcommand*{\Glsentryfullpl}[1]{%
10483   \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}\space
10484   (\glsshortpluralaccessdisplay{\acronymfont{\glsentryshortpl{##1}}{##1}})%
10485 }%
10486 \renewcommand*{\acronymentry}[1]{%
10487   \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}{##1}}%
10488 \renewcommand*{\acronymsort}[2]{##1}%
10489 \renewcommand*{\acronymfont}[1]{##1}%
10490 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
10491 }

```

dua-desc *<long>* only acronym style with user-supplied description.

```

10492 \renewacronymstyle{dua-desc}%
10493 {%
10494   \GlsUseAcrEntryDispStyle{dua}%
10495 }%
10496 {%

```



```

10497 \GlsUseAcrStyleDefs{dua}%
10498 \renewcommand*\GenericAcronymFields{}%
10499 \renewcommand*\acronymentry}[1]{%
10500     \glslongaccessdisplay{\acronymfont{\glsentrylong{##1}}{##1}}%
10501 \renewcommand*\acronymsort}[2]{##2}%
10502 }%

```

footnote *<short>*\footnote{*<long>*} acronym style.

```

10503 \renewacronymstyle{footnote}%
10504 {%

```

Check for long form in case this is a mixed glossary.

```

10505 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
10506 }%
10507 {%
10508 \renewcommand*\GenericAcronymFields{description={\the\glslongtok}}%

```

Need to ensure hyperlinks are switched off on first use:

```

10509 \glshyperfirstfalse
10510 \renewcommand*\genacrfullformat}[2]{%
10511     \glsshortaccessdisplay
10512     {\protect\firstacronymfont{\glsentryshort{##1}}{##1}##2%
10513     \protect\footnote{\glslongaccessdisplay{\glsentrylong{##1}}{##1}}%
10514     }%
10515 \renewcommand*\Genacrfullformat}[2]{%
10516     \glsshortaccessdisplay
10517     {\firstacronymfont{\Glsentryshort{##1}}{##1}##2%
10518     \protect\footnote{\glslongaccessdisplay{\glsentrylong{##1}}{##1}}%
10519     }%
10520 \renewcommand*\genplacrfullformat}[2]{%
10521     \glsshortpluralaccessdisplay
10522     {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}##2%
10523     \protect\footnote{\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}}%
10524     }%
10525 \renewcommand*\Genplacrfullformat}[2]{%
10526     \glsshortpluralaccessdisplay
10527     {\protect\firstacronymfont{\Glsentryshortpl{##1}}{##1}##2%
10528     \protect\footnote{\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}}%
10529     }%
10530 \renewcommand*\acronymentry}[1]{%
10531     \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}{##1}}%
10532 \renewcommand*\acronymsort}[2]{##1}%
10533 \renewcommand*\acronymfont}[1]{##1}%
10534 \renewcommand*\acrpluralsuffix{\glspluralsuffix}%

```

Don't use footnotes for \acrfull:

```

10535 \renewcommand*\acrfullfmt}[3]{%
10536     \glslink[##1]{##2}{%
10537         \glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}{##2}##3\space
10538         (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}%

```

```

10539 \renewcommand*\Acrfullfmt}[3]{%
10540   \glslink[##1]{##2}{%
10541     \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##2}}}{##2}##3\space
10542     (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}%
10543 \renewcommand*\ACRfullfmt}[3]{%
10544   \glslink[##1]{##2}{%
10545     \glsshortaccessdisplay
10546     {\mfirstucMakeUppercase
10547     {\acronymfont{\glsentryshort{##2}}}{##2}##3\space
10548     (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}}%
10549 \renewcommand*\acrfullplfmt}[3]{%
10550   \glslink[##1]{##2}{%
10551     \glsshortpluralaccessdisplay
10552     {\acronymfont{\glsentryshortpl{##2}}}{##2}##3\space
10553     (\glslongpluralaccessdisplay{\glsentrylongpl{##2}}{##2})}}}%
10554 \renewcommand*\Acrfullplfmt}[3]{%
10555   \glslink[##1]{##2}{%
10556     \glsshortpluralaccessdisplay
10557     {\acronymfont{\Glsentryshortpl{##2}}}{##2}##3\space
10558     (\glslongpluralaccessdisplay{\glsentrylongpl{##2}})}%
10559 \renewcommand*\ACRfullplfmt}[3]{%
10560   \glslink[##1]{##2}{%
10561     \glsshortpluralaccessdisplay
10562     {\mfirstucMakeUppercase
10563     {\acronymfont{\glsentryshortpl{##2}}}{##2}##3\space
10564     (\glslongpluralaccessdisplay{\glsentrylongpl{##2}}{##2})}}}%

```

Similarly for \glsentryfull etc:

```

10565 \renewcommand*\glsentryfull}[1]{%
10566   \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}\space
10567   (\glslongaccessdisplay{\glsentrylong{##1}}{##1})}%
10568 \renewcommand*\Glsentryfull}[1]{%
10569   \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##1}}}{##1}\space
10570   (\glslongaccessdisplay{\glsentrylong{##1}}{##1})}%
10571 \renewcommand*\glsentryfullpl}[1]{%
10572   \glsshortpluralaccessdisplay
10573   {\acronymfont{\glsentryshortpl{##1}}}{##1}\space
10574   (\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1})}%
10575 \renewcommand*\Glsentryfullpl}[1]{%
10576   \glsshortpluralaccessdisplay
10577   {\acronymfont{\Glsentryshortpl{##1}}}{##1}\space
10578   (\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1})}%
10579 }

```

footnote-sc \textsc{<short>}\footnote{<long>} acronym style.

```

10580 \renewacronymstyle{footnote-sc}%
10581 {%
10582   \GlsUseAcrEntryDispStyle{footnote}%
10583 }%
10584 {%

```

```

10585 \GlsUseAcrStyleDefs{footnote}%
10586 \renewcommand{\acronymentry}[1]{%
10587     \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}
10588 \renewcommand{\acronymfont}[1]{\textsc{##1}}%
10589 \renewcommand*\{acrpluralsuffix}{\glstextup{\glspluralsuffix}}%
10590 }%

```

footnote-sm \textsmaller{<short>}\footnote{<long>} acronym style.

```

10591 \renewacronymstyle{footnote-sm}%
10592 {%
10593     \GlsUseAcrEntryDispStyle{footnote}%
10594 }%
10595 {%
10596     \GlsUseAcrStyleDefs{footnote}%
10597     \renewcommand{\acronymentry}[1]{%
10598         \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}
10599     \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
10600     \renewcommand*\{acrpluralsuffix}{\glspluralsuffix}%
10601 }%

```

footnote-desc <short>\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

10602 \renewacronymstyle{footnote-desc}%
10603 {%
10604     \GlsUseAcrEntryDispStyle{footnote}%
10605 }%
10606 {%
10607     \GlsUseAcrStyleDefs{footnote}%
10608     \renewcommand*\{GenericAcronymFields}{}%
10609     \renewcommand*\{acronymsort}[2]{##2}%
10610     \renewcommand*\{acronymentry}[1]{%
10611         \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10612         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10613 }

```

footnote-sc-desc \textsc{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

10614 \renewacronymstyle{footnote-sc-desc}%
10615 {%
10616     \GlsUseAcrEntryDispStyle{footnote-sc}%
10617 }%
10618 {%
10619     \GlsUseAcrStyleDefs{footnote-sc}%
10620     \renewcommand*\{GenericAcronymFields}{}%
10621     \renewcommand*\{acronymsort}[2]{##2}%
10622     \renewcommand*\{acronymentry}[1]{%
10623         \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10624         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10625 }

```

```

footnote-sm-desc \textsmaller{<short>}\footnote{<long>} acronym style that has an accom-
panying description (which the user needs to supply).
10626 \renewacronymstyle{footnote-sm-desc}%
10627 {%
10628 \GlsUseAcrEntryDispStyle{footnote-sm}%
10629 }%
10630 {%
10631 \GlsUseAcrStyleDefs{footnote-sm}%
10632 \renewcommand*{\GenericAcronymFields}{}%
10633 \renewcommand*{\acronymsort}[2]{##2}%
10634 \renewcommand*{\acronymentry}[1]{%
10635 \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10636 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10637 }

```

Use `\newacronymhook` to modify the key list to set the access text to the long version by default.

```

10638 \renewcommand*{\newacronymhook}{%
10639 \edef\@gls@keylist{shortaccess=\the\glslongtok,%
10640 \the\glskeylisttok}%
10641 \expandafter\glskeylisttok\expandafter{\@gls@keylist}%
10642 }

```

`defaultNewAcronymDef` Modify default style to use access text:

```

10643 \renewcommand*{\DefaultNewAcronymDef}{%
10644 \edef\@do@newglossaryentry{%
10645 \noexpand\newglossaryentry{\the\glslabeltok}%
10646 {%
10647 type=\acronymtype,%
10648 name={\the\glsshorttok},%
10649 description={\the\glslongtok},%
10650 descriptionaccess=\relax,%
10651 text={\the\glsshorttok},%
10652 access={\noexpand\@glo@textaccess},%
10653 sort={\the\glsshorttok},%
10654 short={\the\glsshorttok},%
10655 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10656 shortaccess={\the\glslongtok},%
10657 long={\the\glslongtok},%
10658 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10659 descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10660 first={\noexpand\glslongaccessdisplay
10661 {\the\glslongtok}{\the\glslabeltok}\space
10662 (\noexpand\glsshortaccessdisplay
10663 {\the\glsshorttok}{\the\glslabeltok})},%
10664 plural={\the\glsshorttok\acrpluralsuffix},%
10665 firstplural={\noexpand\glslongpluralaccessdisplay
10666 {\noexpand\@glo@longpl}{\the\glslabeltok}\space
10667 (\noexpand\glsshortpluralaccessdisplay

```

```

10668         {\noexpand\@glo@shortpl}\the\glslabeltok}},%
10669     firstaccess=\relax,
10670     firstpluralaccess=\relax,
10671     textaccess={\noexpand\@glo@shortaccess},%
10672     \the\glskeylisttok
10673 }%
10674 }%
10675 \let\@org@gls@assign@firstpl\gls@assign@firstpl
10676 \let\@org@gls@assign@plural\gls@assign@plural
10677 \let\@org@gls@assign@descplural\gls@assign@descplural
10678 \def\gls@assign@firstpl##1##2{%
10679     \@gls@expand@field{##1}{firstpl}{##2}%
10680 }%
10681 \def\gls@assign@plural##1##2{%
10682     \@gls@expand@field{##1}{plural}{##2}%
10683 }%
10684 \def\gls@assign@descplural##1##2{%
10685     \@gls@expand@field{##1}{descplural}{##2}%
10686 }%
10687 \@do@newglossaryentry
10688 \let\gls@assign@firstpl\@org@gls@assign@firstpl
10689 \let\gls@assign@plural\@org@gls@assign@plural
10690 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
10691 }

```

otnoteNewAcronymDef

```

10692 \renewcommand*{\DescriptionFootnoteNewAcronymDef}{%
10693     \edef\@do@newglossaryentry{%
10694         \noexpand\newglossaryentry{\the\glslabeltok}%
10695         {%
10696             type=\acronymtype,%
10697             name={\noexpand\acronymfont{\the\glsshorttok}},%
10698             sort={\the\glsshorttok},%
10699             text={\the\glsshorttok},%
10700             short={\the\glsshorttok},%
10701             shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10702             shortaccess={\the\glslongtok},%
10703             long={\the\glslongtok},%
10704             longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10705             access={\noexpand\@glo@textaccess},%
10706             plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10707             symbol={\the\glslongtok},%
10708             symbolplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10709             firstpluralaccess=\relax,
10710             textaccess={\noexpand\@glo@shortaccess},%
10711             \the\glskeylisttok
10712         }%
10713     }%
10714     \let\@org@gls@assign@firstpl\gls@assign@firstpl

```

```

10715 \let\@org@gls@assign@plural\gls@assign@plural
10716 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
10717 \def\gls@assign@firstpl##1##2{%
10718   \@@gls@expand@field{##1}{firstpl}{##2}%
10719 }%
10720 \def\gls@assign@plural##1##2{%
10721   \@@gls@expand@field{##1}{plural}{##2}%
10722 }%
10723 \def\gls@assign@symbolplural##1##2{%
10724   \@@gls@expand@field{##1}{symbolplural}{##2}%
10725 }%
10726 \@do@newglossaryentry
10727 \let\gls@assign@plural\@org@gls@assign@plural
10728 \let\gls@assign@firstpl\@org@gls@assign@firstpl
10729 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
10730 }

```

ptionNewAcronymDef

```

10731 \renewcommand*{\DescriptionNewAcronymDef}{%
10732   \edef\@do@newglossaryentry{%
10733     \noexpand\newglossaryentry{\the\glslabeltok}%
10734     {%
10735       type=\acronymtype,%
10736       name={\noexpand
10737         \acronymformat{\the\glsshorttok}{\the\glslongtok}},%
10738       access={\noexpand\@glo@textaccess},%
10739       sort={\the\glsshorttok},%
10740       short={\the\glsshorttok},%
10741       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10742       shortaccess={\the\glslongtok},%
10743       long={\the\glslongtok},%
10744       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10745       first={\the\glslongtok},%
10746       firstaccess=\relax,
10747       firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10748       text={\the\glsshorttok},%
10749       textaccess={\the\glslongtok},%
10750       plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10751       symbol={\noexpand\@glo@text},%
10752       symbolaccess={\noexpand\@glo@textaccess},%
10753       symbolplural={\noexpand\@glo@plural},%
10754       firstpluralaccess=\relax,
10755       textaccess={\noexpand\@glo@shortaccess},%
10756       \the\glskeylisttok}%
10757   }%
10758   \let\@org@gls@assign@firstpl\gls@assign@firstpl
10759   \let\@org@gls@assign@plural\gls@assign@plural
10760   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
10761   \def\gls@assign@firstpl##1##2{%

```

```

10762 \@@gls@expand@field{##1}{firstpl}{##2}%
10763 }%
10764 \def\gls@assign@plural##1##2{%
10765 \@@gls@expand@field{##1}{plural}{##2}%
10766 }%
10767 \def\gls@assign@symbolplural##1##2{%
10768 \@@gls@expand@field{##1}{symbolplural}{##2}%
10769 }%
10770 \@do@newglossaryentry
10771 \let\gls@assign@firstpl\@org@gls@assign@firstpl
10772 \let\gls@assign@plural\@org@gls@assign@plural
10773 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
10774 }

```

otnoteNewAcronymDef

```

10775 \renewcommand*{\FootnoteNewAcronymDef}{%
10776 \edef\@do@newglossaryentry{%
10777 \noexpand\newglossaryentry{\the\glslabeltok}%
10778 {%
10779 type=\acronymtype,%
10780 name={\noexpand\acronymfont{\the\glsshorttok}},%
10781 sort={\the\glsshorttok},%
10782 text={\the\glsshorttok},%
10783 textaccess={\the\glslongtok},%
10784 access={\noexpand\@glo@textaccess},%
10785 plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10786 short={\the\glsshorttok},%
10787 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10788 long={\the\glslongtok},%
10789 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10790 description={\the\glslongtok},%
10791 descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10792 \the\glskeylisttok
10793 }%
10794 }%
10795 \let\@org@gls@assign@plural\gls@assign@plural
10796 \let\@org@gls@assign@firstpl\gls@assign@firstpl
10797 \let\@org@gls@assign@descplural\gls@assign@descplural
10798 \def\gls@assign@firstpl##1##2{%
10799 \@@gls@expand@field{##1}{firstpl}{##2}%
10800 }%
10801 \def\gls@assign@plural##1##2{%
10802 \@@gls@expand@field{##1}{plural}{##2}%
10803 }%
10804 \def\gls@assign@descplural##1##2{%
10805 \@@gls@expand@field{##1}{descplural}{##2}%
10806 }%
10807 \@do@newglossaryentry
10808 \let\gls@assign@plural\@org@gls@assign@plural

```

```

10809 \let\gls@assign@firstpl\@org@gls@assign@firstpl
10810 \let\gls@assign@descplural\@org@gls@assign@descplural
10811 }

```

\SmallNewAcronymDef

```

10812 \renewcommand*{\SmallNewAcronymDef}{%
10813 \edef\@do@newglossaryentry{%
10814 \noexpand\newglossaryentry{\the\glslabeltok}%
10815 {%
10816 type=\acronymtype,%
10817 name={\noexpand\acronymfont{\the\glsshorttok}},%
10818 access={\noexpand\@glo@symbolaccess},%
10819 sort={\the\glsshorttok},%
10820 short={\the\glsshorttok},%
10821 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10822 shortaccess={\the\glslongtok},%
10823 long={\the\glslongtok},%
10824 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10825 text={\noexpand\@glo@short},%
10826 textaccess={\noexpand\@glo@shortaccess},%
10827 plural={\noexpand\@glo@shortpl},%
10828 first={\the\glslongtok},%
10829 firstaccess=\relax,
10830 firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10831 description={\noexpand\@glo@first},%
10832 descriptionplural={\noexpand\@glo@firstplural},%
10833 symbol={\the\glsshorttok},%
10834 symbolaccess={\the\glslongtok},%
10835 symbolplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10836 \the\glskeylisttok
10837 }%
10838 }%
10839 \let\@org@gls@assign@firstpl\gls@assign@firstpl
10840 \let\@org@gls@assign@plural\gls@assign@plural
10841 \let\@org@gls@assign@descplural\gls@assign@descplural
10842 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
10843 \def\gls@assign@firstpl##1##2{%
10844 \@@gls@expand@field{##1}{firstpl}{##2}%
10845 }%
10846 \def\gls@assign@plural##1##2{%
10847 \@@gls@expand@field{##1}{plural}{##2}%
10848 }%
10849 \def\gls@assign@descplural##1##2{%
10850 \@@gls@expand@field{##1}{descplural}{##2}%
10851 }%
10852 \def\gls@assign@symbolplural##1##2{%
10853 \@@gls@expand@field{##1}{symbolplural}{##2}%
10854 }%
10855 \@do@newglossaryentry

```



```

10856 \let\gls@assign@firstpl\@org@gls@assign@firstpl
10857 \let\gls@assign@plural\@org@gls@assign@plural
10858 \let\gls@assign@descplural\@org@gls@assign@descplural
10859 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
10860 }

```

The following are kept for compatibility with versions before 3.0:

`\glsshortaccesskey`

```

10861 \newcommand*\glsshortaccesskey{\glsshortkey access}%

```

`\glsshortpluralaccesskey`

```

10862 \newcommand*\glsshortpluralaccesskey{\glsshortpluralkey access}%

```

`\glslongaccesskey`

```

10863 \newcommand*\glslongaccesskey{\glslongkey access}%

```

`\glslongpluralaccesskey`

```

10864 \newcommand*\glslongpluralaccesskey{\glslongpluralkey access}%

```

7.5 Debugging Commands

`\showglongnameaccess`

```

10865 \newcommand*\showglongnameaccess}[1]{%
10866 \expandafter\show\csname glo@\glsdetoklabel{#1}@textaccess\endcsname
10867 }

```

`\showglotextaccess`

```

10868 \newcommand*\showglotextaccess}[1]{%
10869 \expandafter\show\csname glo@\glsdetoklabel{#1}@textaccess\endcsname
10870 }

```

`\showglopluralaccess`

```

10871 \newcommand*\showglopluralaccess}[1]{%
10872 \expandafter\show\csname glo@\glsdetoklabel{#1}@pluralaccess\endcsname
10873 }

```

`\showglofirstaccess`

```

10874 \newcommand*\showglofirstaccess}[1]{%
10875 \expandafter\show\csname glo@\glsdetoklabel{#1}@firstaccess\endcsname
10876 }

```

`\showglofirstpluralaccess`

```

10877 \newcommand*\showglofirstpluralaccess}[1]{%
10878 \expandafter\show\csname glo@\glsdetoklabel{#1}@firstpluralaccess\endcsname
10879 }

```

showglosymbolaccess

```
10880 \newcommand*{\showglosymbolaccess}[1]{%
10881   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolaccess\endcsname
10882 }
```

osymbolpluralaccess

```
10883 \newcommand*{\showglosymbolpluralaccess}[1]{%
10884   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolpluralaccess\endcsname
10885 }
```

\showglodescaccess

```
10886 \newcommand*{\showglodescaccess}[1]{%
10887   \expandafter\show\csname glo@\glsdetoklabel{#1}@descaccess\endcsname
10888 }
```

glodescpluralaccess

```
10889 \newcommand*{\showglodescpluralaccess}[1]{%
10890   \expandafter\show\csname glo@\glsdetoklabel{#1}@descpluralaccess\endcsname
10891 }
```

\showgloshortaccess

```
10892 \newcommand*{\showgloshortaccess}[1]{%
10893   \expandafter\show\csname glo@\glsdetoklabel{#1}@shortaccess\endcsname
10894 }
```

loshortpluralaccess

```
10895 \newcommand*{\showgloshortpluralaccess}[1]{%
10896   \expandafter\show\csname glo@\glsdetoklabel{#1}@shortpluralaccess\endcsname
10897 }
```

\showglolongaccess

```
10898 \newcommand*{\showglolongaccess}[1]{%
10899   \expandafter\show\csname glo@\glsdetoklabel{#1}@longaccess\endcsname
10900 }
```

glolongpluralaccess

```
10901 \newcommand*{\showglolongpluralaccess}[1]{%
10902   \expandafter\show\csname glo@\glsdetoklabel{#1}@longpluralaccess\endcsname
10903 }
```

8 Multi-Lingual Support

Many thanks to everyone who contributed to the translations both via email and on comp.text.tex. Language support has now been split off into independent language modules.

```
10904 \NeedsTeXFormat{LaTeX2e}
10905 \ProvidesPackage{glossaries-babel}[2014/11/22 v4.12 (NLCT)]
```

Load tracklang to obtain language settings.

```
10906 \RequirePackage{tracklang}
10907 \let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
10908 \AnyTrackedLanguages
10909 {%
10910   \ForEachTrackedDialect{\this@dialect}{%
10911     \IfTrackedLanguageFileExists{\this@dialect}%
10912     {glossaries-}% prefix
10913     {.ldf}%
10914     {%
10915       \RequireGlossariesLang{\CurrentTrackedTag}%
10916     }%
10917     {%
10918       \PackageWarningNoLine{glossaries}%
10919       {No language module detected for ‘\this@dialect’.\MessageBreak
10920       Language modules need to be installed separately.\MessageBreak
10921       Please check on CTAN for a bundle called\MessageBreak
10922       ‘glossaries-\CurrentTrackedLanguage’ or similar}%
10923     }%
10924   }%
10925 }%
10926 }
```

8.1 Polyglossia Captions

Language support has now been split off into independent language modules.

```
10927 \NeedsTeXFormat{LaTeX2e}
10928 \ProvidesPackage{glossaries-polyglossia}[2014/11/22 v4.12 (NLCT)]
```

Load tracklang to obtain language settings.

```
10929 \RequirePackage{tracklang}
10930 \let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
10931 \AnyTrackedLanguages
10932 {%
10933   \ForEachTrackedDialect{\this@dialect}{%
10934     \IfTrackedLanguageFileExists{\this@dialect}%
10935     {glossaries-}% prefix
10936     {.ldf}%
10937     {%
10938       \RequireGlossariesLang{\CurrentTrackedTag}%
10939     }%
10940     {%
10941       \PackageWarningNoLine{glossaries}%
10942       {No language module detected for ‘\this@dialect’.\MessageBreak
10943       Language modules need to be installed separately.\MessageBreak
10944       Please check on CTAN for a bundle called\MessageBreak
```

10945 ‘glossaries-`\CurrentTrackedLanguage`’ or similar}%
 10946 }%
 10947 }%
 10948 }%
 10949 {}%

Glossary

`makeindex` An indexing application. [10](#), [25](#), [26](#), [166](#)

`xindy` An flexible indexing application with multilingual support written in Perl. [10](#), [25](#), [26](#), [166](#)

Change History

??		ting the page number too soon 102
	<code>super</code> : fixed typo in <code>\subglossentry</code> <code>(\glossentrydesc)</code> 278	<code>\glsadd</code> : fixed bug caused by <code>\theglentrycounter</code> set- ting the page number too soon 149
1.01	General: Added range facility in format key 104	1.08
	<code>\writeist</code> : Added spaces after <code>\delimN</code> and <code>\delimR</code> in ist file 151	General: Added babel support ... 31
1.03	<code>\makefirstuc</code> : changed 'pro- tected@edef to 'def 251	<code>\capitalisewords</code> : made robust 252
1.04	General: Added <code>\glstextformat</code> 89	<code>listgroup</code> : changed listgroup style to use <code>\glsgetgrouptitle</code> 259
1.05	<code>\glossarysection</code> : added <code>\@mkboth to \glossarysection</code> 37	<code>altlistgroup</code> : changed al- tlistgroup style to use <code>\glsgetgrouptitle</code> 260
	<code>\gls@defglossaryentry</code> : Changed the default value of the sort key to just the value of the name key 72	<code>\makefirstuc</code> : made robust ... 251
	<code>\glsmakefirstuc</code> : new 252	1.09
1.06	General: now requires <code>etoolbox</code> . 250	<code>\@mfu@nocaplist</code> : new 253
	<code>\capitalisewords</code> : new 252	<code>\capitalisewords</code> : added check for words that shouldn't be capitalised 252
	<code>\xcapitalisewords</code> : new 253	<code>\gMFUnocap</code> : new 253
1.07	<code>\@gls@link</code> : fixed bug caused by <code>\theglentrycounter</code> set-	<code>\mfu@checkword</code> : new 253
		<code>\MFUclear</code> : new 253
		1.1
		<code>\@glossarysection</code> : numbered sections and auto label added 39
		<code>\@gls@tmpb</code> : changed <code>\toksdef</code> to <code>\newtoks</code> 107

<code>\@gls@toc: numberline added</code> .. 40	<code>\SetFootnoteAcronymStyle:</code>
<code>\@p@glossarysection: numbered sections and auto label added</code> 39	Added <code>\protect</code> before <code>\footnote</code> and <code>\glslink</code> . 231
General: <code>amsgen</code> now loaded (<code>\new@ifnextchar</code> needed) .. 4	symbolplural: new 61
<code>translate: translate</code> option added 22	1.13
<code>\setglossarysection: new</code> ... 38	General: fixed bug that ignored 3rd parameter 120–128
<code>numberedsection: numbered-section package option added</code> . 6	<code>\ACRfullpl: new</code> 206
<code>numberline: numberline</code> option added 6	<code>\Acrfullpl: new</code> 206
1.10	<code>\acrfullpl: new</code> 205
<code>\ecapitalisewords: new</code> 253	<code>\acrpluralsuffix: New</code> 203
<code>\emakefirstuc: new</code> 252	<code>\gls@defglossaryentry:</code>
1.12	Changed default first value .. 72
<code>\@GLSpl: now uses \glsentrydescplural and \glsentrysymbolplural instead of \glsentrydesc and \glsentrysymbol</code> 118	Changed default firstplural value 72
<code>\@Glspl: now uses \glsentrydescplural and \glsentrysymbolplural instead of \glsentrydesc and \glsentrysymbol</code> 117	Removed restriction on only using <code>\newglossaryentry</code> in the preamble 77
<code>\@glspl: now uses \glsentrydescplural and \glsentrysymbolplural instead of \glsentrydesc and \glsentrysymbol</code> 116	<code>\newacronym: Removed restriction on only using \newacronym in the preamble</code> 203
General: added check for <code>\hypertarget</code> separate to <code>\hyperlink</code> (memoir defines <code>\hyperlink</code> but not <code>\hypertarget</code>) 112	<code>\@gls@hypergroup: new</code> 255
<code>descriptionplural: new</code> 60	General: added nonnumberlist key to <code>\printglossary</code> 189
<code>\gls@defglossaryentry:</code>	added <code>numberedsection</code> key to <code>\printglossary</code> 188
Changed default first plural to be first key with <code>s</code> appended (was text key with <code>s</code> appended) 72	<code>\firstacronymfont: new</code> 207
<code>descriptionplural</code> support added 72	<code>\glsautoprefix: new</code> 6
symbolplural support added .. 72	<code>\glsnavhyperlink: changed 'edef to 'protected@edef</code> ... 254
<code>\Glsentrydescplural: New</code> .. 143	<code>\glsnavhypertarget: added write to aux file</code> 254
<code>\glsentrydescplural: New</code> .. 142	<code>\glsnavigation: changed to only use labels for groups that are present</code> 256
<code>\Glsentrysymbolplural: New</code> 144	1.15
<code>\glsentrysymbolplural: New</code> 143	<code>\@gls@link: added \glslabel</code> 102
<code>\SetDescriptionFootnoteAcronymStyle</code>	<code>\gls@defglossaryentry: check for \@glo@first in description</code> 76
Added <code>\protect</code> before <code>\footnote</code> and <code>\glslink</code> . 224	check for <code>\@glo@text</code> in symbol 76
	<code>\gls@hypergroup: new</code> . 255
	<code>\glsnavhypertarget: added check if rerun required</code> 254
	<code>\glssettoctitle: new</code> 30
	<code>\printglossary: changed the way the TOC title is set</code> 174

1.16			
	\@GLS@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	116	
	\@GLSpl: Test glossary type is \acronymtype in addition to checking if footnote option has been used	118	
	\@GLs@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	115	
	\@GLspl@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	117	
	\@gls@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	114	
	\@glsdisp: Test glossary type is \acronymtype in addition to checking if footnote option has been used	119	
	\@glspl@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	117	
	\@glstarget: raised the hyper- target so the target text doesn't scroll off the top of the page	113	
	\gls@defglossaryentry: Changed def to let	72	
1.17			
	\@do@wrglossary: new	169	
	\@do@seeglossary: new	171	
	\@glo@storeentry: new	78	
	\@gls@glossary: changed defi- nition to use \index instead of \@index	167	
	\@glsdefaultplural: new	63	
	\@glsdefaultsort: new	64	
	\@glshypernumber: new	200	
	\@glsnoname: new	63	
	\@glsnonextpages: new	190	
	General: added xindy support ...	25	
	parent: new	62	
	see: new	61	
	\gls@defglossaryentry: added nonumberlist key	73	
	added parent key	73	
	added see key	72	
	Stored main part of entry format when entry is defined	77	
	\gls@suffixF: new	35	
	\gls@suffixFF: new	36	
	\gls@wrglossary: modified to allow for xindy support	167	
	\glshyperlink: new	148	
	\glshypernumber: modified to allow material to be attached to location	200	
	\glsnavhyperlink: replaced 'hy- perlink to '@glslink	254	
	\glsnavhypertarget: replaced 'hypertarget to '@glstarget .	254	
	\glssee: new	172	
	\glsseeformat: new	172	
	\glsSetSuffixF: new	35	
	\glsSetSuffixFF: new	36	
	\ifglsxindy: new	25	
	\istfilename: added xindy sup- port	34	
	\newglossarystyle: made \newglossarystyle long .	199	
	\nopostdesc: new	33	
	nonumberlist: new	62	
	\printglossary: added check to determine if \printglossary is already defined	174	
	added print language to aux file	174	
	order: order package option added	25	
	\writeist: added xindy support	151	
1.18			
	\@gls@loadlist: new	8	
	\@gls@loadlong: new	8	
	\@gls@loadsuper: new	8	
	\@gls@loadtree: new	8	
	\gls@defglossaryentry: Changed default value of sort to \@glsdefaultsort	72	
	moved sort sanitization to \newglossaryentry	76	
	\glstarget: new	193	
	\oldacronym: new	203	
	nolist: new	8	

nolong: new	8	\glossarysection: changed	
sort: moved sanitization to		\@mkboth to \glossarymark	37
\newglossaryentry	60	\glsglossarymark: New	38
nostyles: new	8		
nosuper: new	8	2.03	
notree: new	8	\@GLS@: Added check for hyper-	
1.19		first	116
\glsclearpage: new	40	\@GLSp1: Added check for hyper-	
\glsdisp: new	119	first	118
\SetDescriptionAcronymStyle:		\@Gls@: Added check for hyper-	
changed \acronymfont to		first	115
use \textsmaller instead of		\@Glspl@: Added check for hyper-	
\smaller	229	first	117
\SetDescriptionFootnoteAcronymStyle:		\@gls@: Added check for hyper-	
changed \acronymfont to		first	114
use \textsmaller instead of		\@gls@@link: new	101
\smaller	224	\@gls@link: added \leavevmode	
\SetFootnoteAcronymStyle:		102
changed \acronymfont to		Moved entry existence check to	
use \textsmaller instead of		avoid duplicate code	102
\smaller	231	\@glsdisp: Added check for hyper-	
\SetSmallAcronymStyle:		first	119
changed \acronymfont to		\@glsp1@: Added check for hyper-	
use \textsmaller instead of		first	117
\smaller	234	\glsglossarymark: Added check	
2.01		to see if it's already defined ..	38
\@gls@link: moved \@do@wrglossary		hyperfirst: new	23
before term is displayed to pre-		2.04	
vent unwanted whatsit	103	\@GLS@: Changed test to check if	
\forallglossaries: replaced		glossary type has been identi-	
\ifthenelse with \ifx	49	fied as a list of acronyms ...	116
\forallglsentries: replaced		\@GLSp1: Changed test to check if	
\ifthenelse with \ifx	50	glossary type has been identi-	
\glsdefmain: new	12	fied as a list of acronyms ...	118
\glsdescwidth: changed		\@Gls@: Changed test to check if	
\linewidth to \hsize . 262, 277		glossary type has been identi-	
\glslistdottedwidth: changed		fied as a list of acronyms ...	115
\linewidth to \hsize	261	\@Glspl@: Changed test to check	
\glspagelistwidth: changed		if glossary type has been identi-	
\linewidth to \hsize . 262, 278		fied as a list of acronyms ..	117
nomain: added nomain package		\@glossaryentryfield: new ..	78
option	13	\@glossarysubentryfield:	
\writeist: removed item_02 - no		new	78
such makeindex key	155	\@gls@: Changed test to check if	
2.02		glossary type has been identi-	
\@printglossary: suppressed		fied as a list of acronyms ...	114
warning globally rather than		\@glsacronymlists: new	14
locally	176	\@glsdisp: Changed test to check	
		if glossary type has been identi-	
		fied as a list of acronyms ..	119

<code>\@glsp1@</code> : Changed test to check if glossary type has been identified as a list of acronyms ..	117		
<code>\@newglossaryentryposthook</code> : new	77		
<code>\@newglossaryentryprehook</code> : new	77		
<code>acronymlists:new</code>	15		
<code>\DeclareAcronymList:new</code> ...	14		
<code>\DefineAcronymSynonyms:new</code>	219		
<code>\gls@defglossaryentry</code> : added user1-6 keys	73		
<code>\glsadd</code> : fixed bug that ignored counter	149		
<code>\Glsentryuseri</code> : new	145		
<code>\glsentryuseri</code> : new	144		
<code>\Glsentryuserii</code> : new	145		
<code>\glsentryuserii</code> : new	145		
<code>\Glsentryuseriii</code> : new	145		
<code>\glsentryuseriii</code> : new	145		
<code>\Glsentryuseriv</code> : new	145		
<code>\glsentryuseriv</code> : new	145		
<code>\Glsentryuserv</code> : new	145		
<code>\glsentryuserv</code> : new	145		
<code>\Glsentryuservi</code> : new	146		
<code>\glsentryuservi</code> : new	146		
<code>\ns@newglossary</code> : added check to determine if <code>\gls@<type>@display</code> and <code>\gls@<type>@displayfirst</code> have been defined.	57		
<code>\SetAcronymLists</code> : new	15		
<code>\SetDefaultAcronymDisplayStyle</code> : new	221		
<code>\SetDefaultAcronymStyle</code> : new	222		
<code>\SetDescriptionAcronymDisplayStyle</code> : new	226		
<code>\SetDescriptionDUAAcronymDisplayStyle</code> : new	225		
<code>\SetDescriptionFootnoteAcronymDisplayStyle</code> : new	223		
<code>\SetDUADisplayStyle</code> : new ..	234		
<code>\SetFootnoteAcronymDisplayStyle</code> : new	229		
<code>\SetSmallAcronymDisplayStyle</code> : new	231		
		2.05	
			<code>\@glsdisp</code> : Added closing brace. Patch provided by Sergiu Dotenco
			119
			Removed spurious brace. Patch provided by Sergiu Dotenco
			119
			<code>\writeist</code> : Added <code>\string</code> before opening and closing braces. Patch provided by Segiu Dotenco
			156
		2.06	
			<code>\altnewglossary</code> : new
			57
			<code>\CustomAcronymFields</code> : new .
			237
			<code>\CustomNewAcronymDef</code> : new .
			237
			<code>\SetCustomDisplayStyle</code> : new
			236
			<code>\SetCustomStyle</code> : new
			237
		2.07	
			General: <code>glsadd</code> format key stored in <code>\@glsnumberformat</code> (was mistakenly stored in <code>\@glo@format</code>)
			149
		3.0	
			<code>\@do@wrglossary</code> : added check for hyper location prefix ...
			170
			modified to use new format ..
			169
			<code>\@@glossarysec</code> : replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>
			6
			<code>\@do@seeglossary</code> : Sanitize and escape cross-referencing information
			172
			<code>\@gls@counterwithin</code> : new ...
			10
			<code>\@gls@ifinlist</code> : new
			41
			<code>\@gls@link</code> : added <code>\@gls@saveentrycounter</code>
			103
			added <code>\@gls@setsort</code>
			103
			<code>\@gls@saveentrycounter</code> : new
			103
			<code>\@gls@setupsort@def</code> : new ...
			11
			<code>\@gls@setupsort@standard</code> : new
			10
			<code>\@gls@setupsort@use</code> : new ...
			11
			<code>\@gls@xdy@locationlist</code> : new
			44
			<code>\@gls@link</code> : replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>
			112
			<code>\@glsnextpages</code> : new
			190
			<code>\@makeglossary</code> : Added check for <code>savewrites</code>
			157
			<code>\@print@glossary</code> : replaced <code>\@ifundefined</code> with

<code>\ifcsundef</code>	177	<code>\glsadd:added \@gls@saveentrycounter</code>	149
<code>\@printglossary:</code>	added	<code>\GlsAddXdyCounters:new</code>	41
<code>\currentglossary</code>	175	<code>\glsentrycounterlabel:new</code>	192
added <code>\glsnextpages</code>	176	<code>\glsentryitem:new</code>	192
make toctitle default to title ..	175	<code>\Glsentrylong:new</code>	146
<code>\@set@glo@numformat:</code>	added	<code>\glsentrylong:new</code>	146
4th argument	104	<code>\Glsentrylongpl:new</code>	146
<code>\@xdyattributelist:new</code>	41	<code>\glsentrylongpl:new</code>	146
General: added prefix to hyperlink		<code>\Glsentryshort:new</code>	146
.....	201	<code>\glsentryshort:new</code>	146
etoolbox now loaded	5	<code>\Glsentryshortpl:new</code>	146
replaced <code>\@ifundefined</code> with		<code>\glsentryshortpl:new</code>	146
<code>\ifcsundef</code>	29, 32, 99, 187	<code>\glsgetgrouptitle:</code>	re-
<code>\acrfootnote:new</code>	222	placed <code>\@ifundefined</code> with	
<code>\ACRfull:added starred version</code>	205	<code>\ifcsundef</code>	197
<code>\Acrfull:added starred version</code>	205	<code>\gls glossarymark:</code>	replaced
<code>\acrfull:added starred version</code>	204	<code>\@ifundefined</code>	with
<code>\ACRfullpl:added starred version</code>	206	<code>\ifcsundef</code>	38
<code>\Acrfullpl:added starred version</code>	206	<code>\gls hyperlink:</code>	changed de-
<code>\acrfullpl:added starred version</code>	206	fault from <code>\glsentryname</code> to	
<code>\acrlinkfootnote:new</code>	222	<code>\glsentrytext</code>	148
<code>\acrnolinkfootnote:new</code> ...	222	<code>\gls hypernumber:</code>	replaced
<code>savewrites:new</code>	26	<code>\@ifundefined</code>	with
<code>see:added \@glo@seeautonumberlist</code>		<code>\ifcsundef</code>	200
.....	61	<code>\gls numberformat:</code>	replaced
<code>seeautonumberlist:new</code>	8	<code>\@ifundefined</code>	with
<code>\glossarysection:</code>	replaced	<code>\ifcsundef</code>	36
<code>\@ifundefined</code>	with	<code>\gls rrefentry:new</code>	192
<code>\ifcsundef</code>	37	<code>\gls resetsubentrycounter:</code>	
<code>\glossarystyle:</code>	replaced	new	191
<code>\@ifundefined</code>	with	<code>\gls seeitem:</code>	hyperlink uses
<code>\ifcsundef</code>	198	<code>\gls seeitemformat</code>	instead
<code>\gls@codepage:</code>	replaced	of <code>\glsentryname</code>	173
<code>\@ifundefined</code>	with	<code>\gls seeitemformat:new</code>	173
<code>\ifcsundef</code>	25	<code>\gls sortnumberfmt:new</code>	11
<code>\gls@defglossaryentry:added</code>		<code>\gls stepentry:new</code>	191
<code>\@gls@defsort</code>	76	<code>\gls stepsubentry:new</code>	192
added short and long keys	73	<code>\gls subentrycounterlabel:</code>	
replaced <code>\@ifundefined</code> with		new	192
<code>\ifcsundef</code>	73	<code>\gls subentryitem:new</code>	193
<code>\gls@docclearpage:</code>	replaced	<code>theglossary:replaced \@ifundefined</code>	
<code>\@ifundefined</code>	with	with <code>\ifcsundef</code>	193
<code>\ifcsundef</code>	39	short:new	63
<code>\gls@wrglossary:</code>	modified to	shortplural:new	63
take into account <code>savewrites</code>	167	<code>\if glossaryexists:</code>	re-
		placed <code>\@ifundefined</code> with	
		<code>\ifcsundef</code>	50

<code>\ifglentryexists:</code>	replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	51	<code>\showglossaries:new</code>	243
<code>\istfile:</code>	deprecated	165	<code>\showglossarycounter:new</code> .	243
<code>glossaryentry:new</code>	191	<code>\showglossaryentries:new</code> .	244	
<code>glossarysubentry:new</code>	191	<code>\showglossaryin:new</code>	243	
<code>\newglossaryentry:</code>	replaced <code>\DeclareRobustCommand</code> with <code>\newrobustcmd</code>	66	<code>\showglossaryout:new</code>	243
<code>\newglossarystyle:</code>	replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	199	<code>\showglossarytitle:new</code> ...	243
<code>\ns@newglossary:</code>	added <code>\@gls@defsortcount</code>	57	<code>\showglosymbol:new</code>	241
	replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	57	<code>\showglosymbolplural:new</code> .	241
<code>entrycounter:new</code>	9	<code>\showglotext:new</code>	239	
<code>entrycounterwithin:new</code>	9	<code>\showglotype:new</code>	239	
<code>\oldacronym:</code>	replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	203	<code>\showglouserii:new</code>	240
<code>compatible-2.07:</code>	compatible-2.07 option added	27	<code>\showglouseriii:new</code>	240
<code>long:new</code>	63	<code>\showglouseriv:new</code>	240	
<code>longplural:new</code>	63	<code>\showglouservi:new</code>	240	
<code>nonumberlist:</code>	now boolean ...	62	<code>subentrycounter:new</code>	10
<code>sort:new</code>	10	<code>\writeist:</code>	added xindy-only macro definitions to glossary open tag	153
<code>counter:</code>	replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	61	modified to support new format	151
<code>\printglossary:</code>	replaced <code>\@ifundefined</code> with <code>\ifcsundef</code>	174		
<code>\SetDescriptionFootnoteAcronymDisplayStyle:</code>	expanded options link options	223	<code>\@glswritefiles:</code>	added check for empty glossaries
<code>\setentrycounter:</code>	added optional argument	198	General: made robust	115
<code>\showacronymlists:new</code> ...	242	<code>\ACRfull:</code>	made robust	205
<code>\showglocounter:new</code>	239	<code>\Acrfull:</code>	made robust	205
<code>\showglodesc:new</code>	241	<code>\acrfull:</code>	made robust	204
<code>\showglodescplural:new</code> ...	241	<code>\acrfullformat:</code>	removed <code>\acronymfont</code> as it should always be set in the second argument	204
<code>\showglofirst:new</code>	239	<code>\ACRfullpl:</code>	made robust	206
<code>\showglofirstpl:new</code>	239	<code>\Acrfullpl:</code>	made robust	206
<code>\showgloflag:new</code>	242	<code>\acrfullpl:</code>	made robust	205
<code>\showgloindex:new</code>	242	<code>\ACRlong:</code>	made robust	138
<code>\showglolevel:new</code>	238	<code>\Acrlong:</code>	made robust	137
<code>\showglongame:new</code>	241	<code>\acrlong:</code>	made robust	136
<code>\showgloparent:new</code>	238	<code>\ACRlongpl:</code>	made robust	140
<code>\showgloplural:new</code>	239	<code>\Acrlongpl:</code>	made robust	139
<code>\showglosort:new</code>	241	<code>\ACRshort:</code>	made robust	134
		<code>\Acrshort:</code>	made robust	134
		<code>\acrshort:</code>	made robust	133
		<code>\ACRshortpl:</code>	made robust	136
		<code>\Acrshortpl:</code>	made robust	135
		<code>\acrshortpl:</code>	made robust	135
		<code>\Gls:</code>	made robust	114

<code>\glsadd</code> : made robust	149	3.02	
<code>\glsaddall</code> : made robust	149		<code>\@do@wrglossary</code> : changed
<code>\GLSdesc</code> : made robust	125		<code>\@glslocref</code> to <code>\theglentrycounter</code>
<code>\Glsdesc</code> : made robust	124	 170
<code>\glsdesc</code> : made robust	124		<code>\do@wrglossary</code> : changed
<code>\GLSdescplural</code> : made robust	126		<code>\do@wr@glossary</code> to test for
<code>\Glsdescplural</code> : made robust	125		<code>indexonlyfirst</code> option; put old
<code>\glsdescplural</code> : made robust	125		<code>\do@wr@glossary</code> code into
<code>\glsfirst</code> : made robust	120		<code>\@do@wrglossary</code> 168
<code>\GLSfirstplural</code> : made robust	123		<code>\@gls@missingnumberlist</code> :
<code>\Glsfirstplural</code> : made robust	123		new 64
<code>\glsfirstplural</code> : made robust	122		<code>\@glswritefiles</code> : added check
<code>\glslink</code> : made robust	101		for existence of token in case
<code>\GLSname</code> : made robust	124		<code>\makeglossaries</code> has been
<code>\Glsname</code> : made robust	123		omitted 165
<code>\glsname</code> : made robust	123		<code>\@printglossary</code> : add a way to
<code>\GLSpl</code> : made robust	118		fetch current entry label ... 176
<code>\Glspl</code> : made robust	117		<code>savenumberlist</code> : new 7
<code>\glspl</code> : made robust	116		<code>ucmark</code> : new 9
<code>\GLSplural</code> : made robust	122		<code>\gls@defglossaryentry</code> : added
<code>\GLSsymbol</code> : made robust	126		numberlist element 76
<code>\Glsymbol</code> : made robust	126		<code>\gls@save@numberlist</code> : new . 174
<code>\glsymbol</code> : made robust	126		<code>\gls@wrglossary</code> : added check
<code>\GLSsymbolplural</code> : made robust			for glossary file defined ... 167
.....	127		<code>\glsdisplaynumberlist</code> : new 147
<code>\Glsymbolplural</code> : made robust			<code>\glentrycounter</code> : set default
.....	127		value 103
<code>\glsymbolplural</code> : made robust			<code>\Glsentryfull</code> : fixed bug (re-
.....	127		placed <code>\glentryshortpl</code>
<code>\Glstext</code> : made robust	120		with <code>\glentryshort</code>) ... 147
<code>\glstext</code> : made robust	120		<code>\glentryfullpl</code> : fixed bug (re-
<code>\GLSuseri</code> : made robust	128		placed <code>\glentryshort</code> with
<code>\Glsuseri</code> : made robust	128		<code>\glentryshortpl</code>) 147
<code>\glsuseri</code> : made robust	128		<code>\glentrynumberlist</code> : new .. 147
<code>\GLSuserii</code> : made robust	129		<code>\glsmoveentry</code> : new 78
<code>\Glsuserii</code> : made robust	129		<code>\glsnumlistlastsep</code> : new ... 148
<code>\glsuserii</code> : made robust	129		<code>\glsnumlistsep</code> : new 148
<code>\GLSuseriii</code> : made robust	130		<code>\glsresetsubentrycounter</code> :
<code>\Glsuseriii</code> : made robust	130		new 191
<code>\glsuseriii</code> : made robust	129		<code>\ifglshaschildren</code> : new 52
<code>\GLSuseriv</code> : made robust	131		<code>\ifglshasparent</code> : new 52
<code>\Glsuseriv</code> : made robust	131		<code>\makeglossaries</code> : added list
<code>\glsuseriv</code> : made robust	130		parser 160
<code>\GLSuserv</code> : made robust	132		<code>indexonlyfirst</code> : new 23
<code>\Glsuserv</code> : made robust	132		<code>\renewglossarystyle</code> : new .. 199
<code>\glsuserv</code> : made robust	131		<code>\showglossaryentries</code> : fixed
<code>\GLSuservi</code> : made robust	133		misspelt command 244
<code>\Glsuservi</code> : made robust	132		<code>\SmallNewAcronymDef</code> : fixed
<code>\glsuservi</code> : made robust	132		broken short and long plural 232

3.03		
	\@gls@sanitizesort:new	18
	\@gls@setupsort@standard: used \@gls@sanitizesort .	10
	\@printglossary: allow title to override default toctitle	175
	General: allow title to set toctitle	187
	\glsinlinedescformat:new .	258
	\glsinlineemptydescformat: new	258
	\glsinlinenameformat:new .	258
	\glsinlinepostchild:new ..	258
	\glsinlinesubdescformat: new	258
	\glsinlinesubnameformat: new	258
	\glspostinline: replaced “.” with \glspostdescription	258
	altlongragged4col: added check for glsnogroupskip ..	272
	altsuperragged4col: added check for glsnogroupskip ..	289
	alttree: added check for glsnogroupskip	297
	index: added check for glsnogroupskip	291
	nogroupskip:new	9
	long: added check for glsnogroupskip	263
	long3col: added check for glsnogroupskip	264
	long4col: added check for glsnogroupskip	266
	longragged: added check for glsnogroupskip	269
	longragged3col: added check for glsnogroupskip	270
	nopostdot:new	9
	tree: added check for glsnogroupskip	293
	treenoname: added check for glsnogroupskip	294
	super: added check for glsnogroupskip	278
	super3col: added check for glsnogroupskip	280
	super4col: added check for glsnogroupskip	282
	superragged: added check for glsnogroupskip	285
	superragged3col: added check for glsnogroupskip	287
	3.04	
	\@@do@wrglossary: changed \theglsentrycounter back to \@glslocref	170
	\@@do@wrglossary: modified to compensate for possible incor- rect page number	169
	\@gls@escbsdq: unsani- tize \gls@numberpage, \gls@alphpage, \gls@Alphpage and \gls@romanpage	105
	\@print@glossary: Moved aux write to end of document to prevent unwanted whatsit oc- curring here.	177
	General: Added check for doc package	5
	added datatool-base as a re- quired package	4
	added local key	100
	\gls@Alphpage:new	168
	\gls@alphpage:new	168
	\gls@disablepagerefexpansion: new	168
	\gls@numberpage:new	168
	\gls@protected@pagefmts: new	168
	\gls@romanpage:new	168
	\glsdefmain: added check for doc package	12
	\glsorg@endtheglossary:new .	5
	\glsorg@theglossary:new	5
	altlist: replaced \newline with paragraph break	260
	\PrintChanges:new	5
	3.05	
	\@@do@wrglossary: add Roman case. Fixed bugs in the else statements	169
	\@gls@link: added check for “no- hypertypes”	102
	\@gls@nohyperlist:new	16
	mcolalttree: replaced ‘2’ with \glscols	276

mcolindex: replaced '2' with \glsmcols 274	removed definition of \@glossarysubentryfield 337
mcoltree: replaced '2' with \glsmcols 275	\compatibleglossentry: new 193
mcoltreename: replaced '2' with \glsmcols 276	\compatiblesubglossentry: new 195
\gls@protected@pagefmts: added Roman to list 168	\glossaryentryfield: depre- cated 195
\gls@Romanpage: new 168	\Glossentrydesc: new 194
\GlsDeclareNoHyperList: new 16	\glossentrydesc: new 194
\glsgetgrouplabel: fixed bug (typo in \equal) 198	\Glossentryname: new 194
\nopostdesc: made robust 33	\glossentryname: new 194
nohypertypes: new 16	\Glossentrysymbol: new 195
3.06	\glossentrysymbol: new 194
\@xdy@main@language: Changed back to using \languagename 25	\gls@assign@desc@field: new 18
\findrootlanguage: Obsoleted 48	\gls@assign@descplural@field: new 18
3.07	\gls@assign@field: new 66
\@gls@link: fixed bug that failed to find entry in list 102	\gls@ifnotmeasuring: new ... 80
\glossarypreamble: modified to work with \setglossarypreamble 37	\glsaddallunused: new 150
\gls@docclearpage: added check for openright 39	\glsexpandfields: new 66
\glspostdescription: Added spacefactor code 9	\glsnoexpandfields: new 66
\GlsSetXdyCodePage: Added check for fontspec 49	\glssee: made robust 172
\SetDescriptionAcronymDisplayStyle: now using \glsdoparenifnotempty 226	\glsseeformat: made robust .. 172
\setglossarypreamble: new .. 37	\glsseeitem: made robust 173
3.08a	\glsseeelist: made robust 172
\@glo@storeentry: no longer need to check for special char- acters in any of the fields other than sort 79	\ifglsdescsuppressed: new .. 53
updated for \glossentry 79	\ifglsdesc: new 53
\@glossaryentryfield: switched to \glossentry 78	\ifglsdescsymbol: new 53
\@glossarysubentryfield: switched to \subglossentry 78	list: updated list style to use \glossentry and \subglossentry 259
General: added nogroupskip key to \printglossary 188	listdotted: updated listdotted style to use \glossentry and \subglossentry 261
removed definition of \@glossaryentryfield .. 337	altlist: updated altlist style to use \glossentry and \subglossentry 260
	altlongragged4col: updated to use \glossentry and \subglossentry 272
	alttree: updated to use \glossentry and \subglossentry 295
	index: added paragraph break at end of environment 291
	updated to use \glossentry and \subglossentry 291

inline: updated inline style to use <code>\glossentry</code> and <code>\subglossentry</code>	257	<code>\glstextup:new</code>	204
long: updated to use <code>\glossentry</code> and <code>\subglossentry</code>	263	<code>\ifglshassymbol: changed test to check for <code>\@gls@default@symbol</code></code>	53
longragged: updated to use <code>\glossentry</code> and <code>\subglossentry</code>	269	3.10a	
longragged3col: updated to use <code>\glossentry</code> and <code>\subglossentry</code>	270	<code>\@gls@keymap:new</code>	68
tree: updated to use <code>\glossentry</code> and <code>\subglossentry</code>	292	<code>\@gls@provide@newglossary:new</code>	55
<code>\setglossarystyle:new</code>	198	<code>\@gls@writedef:new</code>	67
<code>\setglossentrycompatibility:new</code>	195	<code>\@gls@defaultplural: Obsolete</code> .	63
superragged: updated to use <code>\glossentry</code> and <code>\subglossentry</code>	285	<code>\@gls@nodesc:new</code>	63
3.09a		<code>\@print@glossary: Added providecommand code to aux file</code>	177
<code>\@gls@assign@symbolplural@field:new</code>	18	<code>\gls@assign@type@field:new</code>	17
<code>\@gls@default@value:new</code> ...	60	<code>\gls@defglossaryentry: Changed to using <code>\@gls@default@value</code></code>	72
<code>\Glsentrydesc: made robust</code> ..	142	new	72
<code>\Glsentrydescplural: made robust</code>	143	<code>\glswritedefhook:new</code>	71
<code>\Glsentryfirst: made robust</code> .	144	<code>\makeglossaries: Added providecommand code to aux file</code>	159
<code>\Glsentryfirstplural: made robust</code>	144	<code>\new@glossaryentry:new</code>	67
<code>\Glsentryfull: made robust</code> ..	147	<code>\ns@newglossary: added <code>\@gls@provide@newglossary</code></code>	57
<code>\Glsentryfullpl: made robust</code> ..	147	3.11a	
<code>\Glsentrylong: made robust</code> ..	146	<code>\@ACRlong: added <code>\glslabel</code>, <code>\glsifplural</code>, <code>\glscapscase</code>, <code>\glsinsert</code> and <code>\glscustomtext</code></code>	337
<code>\Glsentrylongpl: made robust</code> ..	146	<code>\@ACRshort: added <code>\glslabel</code>, <code>\glsifplural</code>, <code>\glscapscase</code>, <code>\glsinsert</code> and <code>\glscustomtext</code></code>	336
<code>\Glsentryname: made robust</code> ..	141	<code>\@Acrlong: added <code>\glslabel</code>, <code>\glsifplural</code>, <code>\glscapscase</code>, <code>\glsinsert</code> and <code>\glscustomtext</code></code>	337
<code>\Glsentryplural: made robust</code> ..	143	<code>\@Acrshort: added <code>\glslabel</code>, <code>\glsifplural</code>, <code>\glscapscase</code>, <code>\glsinsert</code> and <code>\glscustomtext</code></code>	336
<code>\Glsentryshort: made robust</code> .	146	<code>\@GLS@: add <code>\glslabel</code>, <code>\glsifplural</code>, <code>\glscapscase</code>, <code>\glscustomtext</code> and <code>\glsinsert</code></code>	116
<code>\Glsentryshortpl: made robust</code>	146		
<code>\Glsentrysymbol: made robust</code> ..	143		
<code>\Glsentrysymbolplural: made robust</code>	144		
<code>\Glsentrytext: made robust</code> ..	143		
<code>\Glsentryuseri: made robust</code> .	145		
<code>\Glsentryuserii: made robust</code> ..	145		
<code>\Glsentryuseriii: made robust</code>	145		
<code>\Glsentryuseriv: made robust</code> ..	145		
<code>\Glsentryuserv: made robust</code> .	145		
<code>\Glsentryuservi: made robust</code> ..	146		

change to using `\glentryfmt`
style commands 116

removed `\MakeUppercase`
(now moved to `\glentryfmt`)
..... 116

`\@GLSpl:` add `\glslabel`,
`\glsifplural`, `\glscapscase`,
`\glscustomtext` and
`\glsinsert` 118

change to using `\glentryfmt`
style commands 118

removed `\MakeUppercase`
as now dealt with in
`\glentryfmt` 118

`\@Gls@:` add `\glsifplural`,
`\glscapscase`, `\glscustomtext`
and `\glsinsert` 115

change to using `\glentryfmt`
style commands 115

removed `\makefirstuc` (now
dealt with in `\glentryfmt`) 115

`\@Glspl@:` add `\glsifplural`,
`\glscapscase`, `\glscustomtext`
and `\glsinsert` 117

change to using `\glentryfmt`
style commands 117

removed `\makefirstuc` (now
dealt with in `\glentryfmt`) 117

`\@acrlong:` added `\glslabel`,
`\glsifplural`, `\glscapscase`,
`\glsinsert` and `\glscustomtext`
..... 336

`\@acrshort:` added `\glslabel`,
`\glsifplural`, `\glscapscase`,
`\glsinsert` and `\glscustomtext`
..... 335

`\@gls@:` add `\glslabel`,
`\glsifplural`, `\glscapscase`,
`\glscustomtext` and
`\glsinsert` 114

change to using `\glentryfmt`
style commands 114

`\@gls@noexpand@fields:` Fixed
bug `expand` replaced with
`noexpand` 64

`\@glsdisp:` add `\glslabel`,
`\glsifplural`, `\glscapscase`,
`\glscustomtext` and
`\glsinsert` 119

change to using `\glentryfmt`
style commands 119

`\@glspl@:` add `\glslabel`,
`\glsifplural`, `\glscapscase`,
`\glscustomtext` and
`\glsinsert` 116

change to using `\glentryfmt`
style commands 116

General: added `\glslabel`,
`\glsifplural`, `\glscapscase`,
`\glsinsert` and `\glscustomtext`
..... 133–140

changed to just use `\Glsentrydescplural`
..... 125

changed to just use `\glentrydescplural`
..... 125, 126

changed to just use `\Glsentrydesc`
..... 125

changed to just use `\glentrydesc`
..... 124, 125

changed to just use `\Glsentryfirstplural`
..... 123

changed to just use `\glentryfirstplural`
..... 122, 123

changed to just use `\Glsentryfirst`
..... 121

changed to just use `\glentryfirst`
..... 121

changed to just use `\Glsentryname`
..... 124

changed to just use `\glentryname`
..... 123, 124

changed to just use `\Glsentryplural`
..... 122

changed to just use `\glentryplural`
..... 122

changed to just use `\Glsentrysymbolplural`
..... 127

changed to just use `\glentrysymbolplural`
..... 127, 128

changed to just use `\Glsentrysymbol`
..... 126

changed to just use `\glentrysymbol`
..... 126, 127

Changed to just use
`\Glsentrytext` 120

changed to just use `\glentrytext`
..... 120

changed to just use <code>\Glsentryuseriii</code>	<code>\glsnavigation</code> : switched to using <code>\@gls@getgrouptitle</code>	256
..... 130		
changed to just use <code>\Glsentryuseriii</code>	<code>\ifglshasdesc</code> : replaced	
..... 130	<code>\ifdefempty</code> with <code>\ifcsemtyp</code>	
changed to just use <code>\Glsentryuserii</code>	53
..... 129	<code>\ifglshaslong</code> : new	53
changed to just use <code>\Glsentryuserii</code>	<code>\ifglshasshort</code> : new	53
..... 129	<code>\ifglshassymbol</code> : replaced	
changed to just use <code>\Glsentryuseriv</code>	<code>\ifdefempty</code> with <code>\ifcsemtyp</code>	
..... 131	53
changed to just use <code>\Glsentryuseriv</code>	<code>\ifglused</code> : replaced <code>\ifthenelse</code>	
..... 131	with <code>\ifbool</code>	51
changed to just use <code>\Glsentryuseri</code>	<code>\longnewglossaryentry</code> : new	71
..... 128	<code>\ns@newglossary</code> : replaced	
changed to just use <code>\Glsentryuseri</code>	<code>\glsdisplay</code> and <code>\glsdisplayfirst</code>	
..... 128	with <code>\glsentryfmt</code>	57
changed to just use <code>\Glsentryuservi</code>	compatible-3.07: cnew	27
..... 133	<code>\SetCustomDisplayStyle</code> : updated to use <code>\defglentryfmt</code>	
changed to just use <code>\Glsentryuservi</code>	236
..... 132, 133	<code>\SetDefaultAcronymDisplayStyle</code> :	
changed to just use <code>\Glsentryuserv</code>	changed to use <code>\defglentryfmt</code>	
..... 132	221
changed to just use <code>\Glsentryuserv</code>	<code>\SetDescriptionAcronymDisplayStyle</code> :	
..... 131, 132	updated to use <code>\defglentryfmt</code>	
Now requires <code>textcase</code>	226
..... 4	<code>\SetDescriptionDUAAcronymDisplayStyle</code> :	
acronymlists: replaced	updated to use <code>\defglentryfmt</code>	
<code>\@addtoacronymlists</code> with	225
<code>\DeclareAcronymList</code>	<code>\SetDescriptionFootnoteAcronymDisplayStyle</code> :	
....	updated to use <code>\defglentryfmt</code>	
15	223
<code>\defgldisplay</code> : obsoleted	<code>\SetDUADisplayStyle</code> : updated	
....	to use <code>\defglentryfmt</code>	234
98	<code>\SetFootnoteAcronymDisplayStyle</code> :	
<code>\defgldisplayfirst</code> : obsoleted	updated to use <code>\defglentryfmt</code>	
.....	229
98	<code>\SetSmallAcronymDisplayStyle</code> :	
<code>\defglentryfmt</code> : new	updated to use <code>\defglentryfmt</code>	
.....	231
55	<code>\setupglossaries</code> : new	28
<code>\forglsentries</code> : replaced <code>\ifx</code>	<code>\showglolong</code> : new	242
with <code>\ifdefempty</code>	<code>\showgloshort</code> : new	242
.....	numbers: new	27
50	symbols: new	27
<code>\gls@assign@desc</code> : new	
.....		
71	<code>\gls@defglossaryentry</code> : Fixed	
<code>\gls@defglossaryentry</code> : Fixed	default counter if none supplied	75
.....		
75	<code>\gls@doentryfmt</code> : new	55
<code>\gls@doentryfmt</code> : new	
.....		
55	<code>\glsdisplay</code> : obsoleted	98
<code>\glsdisplay</code> : obsoleted	
.....		
98	<code>\glsdisplayfirst</code> : obsoleted	98
<code>\glsdisplayfirst</code> : obsoleted	..	
.....		
98	<code>\glsngenentryfmt</code> : new	93
<code>\glsngenentryfmt</code> : new	
.....		
93	<code>\glsgetgrouptitle</code> : Added	
<code>\glsgetgrouptitle</code> : Added	check in case non-Latin alphabet in use	197
.....	3.12a
197	<code>\gls glossarymark</code> : replaced	
3.12a	<code>\MakeUppercase</code> with	
<code>\gls glossarymark</code> : replaced	<code>\mfirstucMakeUppercase</code>	38
.....		
38	<code>\gls@defglossaryentry</code> : added	
<code>\mfirstucMakeUppercase</code>	<code>\glslabel</code>	72
.....	<code>\glsaddkey</code> : new	69
38	

3.13a		\glsetnoexpandfield: new .. 17
\@gls@assign@symbol@field:	altsuper4colheader: switched	
changed to use \glsetnoexpandfield	to \tabularnewline	283
..... 18	altsuper4colheaderborder:	
\@gls@assign@symbolplural@field:	switched to \tabularnewline	
changed to use \glsetnoexpandfield 284	
..... 18	long: switched to \tabularnewline	
\@gls@link: removed \relax . 103 263	
\@gls@notranslatorhook: new 21	long3col: switched to \tabularnewline	
\@gls@setupsort@standard: 264	
moved \@gls@santizesort	long3colheader: switched to	
to \glsprestandardsort .. 10	\tabularnewline	265
ucmark: added check for memoir . 9	long3colheaderborder: switched	
see: added \gls@checkseeallowed	to \tabularnewline	265
..... 61	long4col: switched to \tabularnewline	
\glossarysection: changed 265	
\glossarymark to \glsglossarymark	long4colheader: switched to	
..... 37	\tabularnewline	266
\glossarystyle: fixed bug	longheader: switched to	
caused by using \ifdef in-	\tabularnewline	263
stead of \ifcsdef	longheaderborder: switched to	
199	\tabularnewline	264
\gls@assign@desc@field:	\SetFootnoteAcronymDisplayStyle:	
changed to use \glsetnoexpandfield	fixed missing argument bug 229	
..... 18	super: switched to \tabularnewline	
\gls@assign@descplural@field: 278	
changed to use \glsetnoexpandfield	super3col: switched to	
..... 18	\tabularnewline	280
\gls@assign@name@field:	super3colheader: switched to	
changed to use \glsetnoexpandfield	\tabularnewline	280
..... 18	super4col: switched to	
\gls@assign@type@field:	\tabularnewline	281
changed to use \glsetexpandfield	super4colheader: switched to	
..... 17	\tabularnewline	282
\gls@checkseeallowed: new .. 61	super4colheaderborder:	
\glsaddallunused: set default to	switched to \tabularnewline	
\@glo@types 283	
150	superheader: switched to	
\Glsentryfull: changed to use	\tabularnewline	279
\acrfullformat	superheaderborder: switched to	
147	\tabularnewline	279
\glsentryfull: changed to use		
\acrfullformat		
147		
\Glsentryfullpl: changed to		
use \acrfullformat		
147		
\glsentryfullpl: changed to	3.14a	
use \acrfullformat	\@glswritefiles: renamed	
147	\glswritefiles to \@glswritefiles	
\glsglossarymark: renamed	and used "savewrites" option	
\glossarymark to \glsglossarymark	to set \glswritefiles 165	
to avoid conflict with memoir 38	General: new	244
\glsprestandardsort: new ... 10	acronyms: new	14
\glsetexpandfield: new 17		

<ul style="list-style-type: none"> <code>\gls@defglossaryentry</code>: added <ul style="list-style-type: none"> check for existence of default glossary 73 set the default for firstplural to be the value of plural 75 <code>xindygloss</code>: new 26 <code>\longprovideglossaryentry</code>: <ul style="list-style-type: none"> new 72 compatible-2.07: added check for 2.07 before setting 3.07 compatibility 27 <code>notranslate</code>: new 22 <code>\provideglossaryentry</code>: new . 66 	<ul style="list-style-type: none"> <code>\glsentryfull</code>: bug fix: added <ul style="list-style-type: none"> missing <code>\acronymfont</code> 147 <code>\Glsentryfullpl</code>: bug fix: added <ul style="list-style-type: none"> missing <code>\acronymfont</code> 147 <code>\glsentryfullpl</code>: bug fix: added <ul style="list-style-type: none"> missing <code>\acronymfont</code> 147 <code>\glsgenacfmt</code>: new 95 <code>\GlsUseAcrEntryDispStyle</code>: <ul style="list-style-type: none"> new 210 <code>\GlsUseAcrStyleDefs</code>: new .. 210 <code>short-long</code>: new 211 <code>short-long-desc</code>: new 213 <code>xindynoglsnumbers</code>: new 26 <code>sm-short-long</code>: new 212 <code>sm-short-long-desc</code>: new ... 214 <code>\makeglossaries</code>: made preamble only 161 <code>index</code>: new 28 <code>\newacronymstyle</code>: new 209 <code>long-sc-short</code>: new 211 <code>long-sc-short-desc</code>: new ... 213 <code>long-short</code>: new 210 <code>long-short-desc</code>: new 212 <code>long-sm-short</code>: new 212 <code>long-sm-short-desc</code>: new ... 213 <code>footnote</code>: new 216 <code>footnote-desc</code>: new 218 <code>footnote-sc</code>: new 218 <code>footnote-sc-desc</code>: new 218 <code>footnote-sm</code>: new 218 <code>footnote-sm-desc</code>: new 219 <code>\setacronymstyle</code>: new 209 <code>\SetDescriptionAcronymDisplayStyle</code>: <ul style="list-style-type: none"> Moved check for empty custom text to prevent unwanted parenthetical material 226 <code>\SetDescriptionFootnoteAcronymDisplayStyle</code>: <ul style="list-style-type: none"> Moved check for empty custom text to prevent unwanted parenthetical material 223 <code>\SetFootnoteAcronymDisplayStyle</code>: <ul style="list-style-type: none"> Moved check for empty custom text to prevent unwanted parenthetical material 229 <code>\SetGenericNewAcronym</code>: new 207 <code>\SetSmallAcronymDisplayStyle</code>: <ul style="list-style-type: none"> Moved check for empty custom text to prevent unwanted parenthetical material 232
<ul style="list-style-type: none"> 4.0 <ul style="list-style-type: none"> <code>\gls@defglossaryentry</code>: added <ul style="list-style-type: none"> check for first key 75 4.01 <ul style="list-style-type: none"> General: fixed non-value options so that they can be passed to document class 7 <code>\CustomAcronymFields</code>: inserted missing comma 237 4.02 <ul style="list-style-type: none"> <code>\@acrfull</code>: now using <code>\acrfullfmt</code> 204 <code>\@gls@indexdef</code>: new 28 <code>\@gls@numbersdef</code>: new 27 <code>\@gls@symbolsdef</code>: new 27 General: Removed <code>\acronymfont</code> 137–140 <code>\ACRfullfmt</code>: new 205 <code>\Acrfullfmt</code>: new 205 <code>\acrfullfmt</code>: new 204 <code>\ACRfullplfmt</code>: new 207 <code>\Acrfullplfmt</code>: new 206 <code>\acrfullplfmt</code>: new 206 <code>\acronymentry</code>: new 209 sanitize: fixed bug that caused an error here 21 <code>sc-short-long</code>: new 212 <code>sc-short-long-desc</code>: new ... 213 <code>\Genacrfullformat</code>: new 97 <code>\genacrfullformat</code>: new 97 <code>\GenericAcronymFields</code>: new 208 <code>\Genplacrfullformat</code>: new ... 97 <code>\genplacrfullformat</code>: new ... 97 <code>\Glsentryfull</code>: bug fix: added <ul style="list-style-type: none"> missing <code>\acronymfont</code> 147 	

dua: new	214	removed \glslabel (defined in	
dua-desc: new	216	\@gls@link)	133
numberedsection:	added	sc-short-long-desc: redefined	
nameref option	6	to use accessibility informa-	
4.03		tion	341
\@do@wrglossary:	added	\compatibleglossentry: added	
\glsdetoklabel	170	\glsdetoklabel	318
\@ACRlong: removed \glslabel		\compatiblesubglossentry:	
(defined in \@gls@link) ...	337	added \glsdetoklabel ...	318
\@ACRshort: removed \glslabel		\Genacrfullformat: redefined	
(defined in \@gls@link) ...	336	to use accessibility informa-	
\@ACrlong: removed \glslabel		tion	335
(defined in \@gls@link) ...	337	\genacrfullformat: redefined	
\@ACrshort: removed \glslabel		to use accessibility informa-	
(defined in \@gls@link) ...	336	tion	334
\@GLS@: removed \glslabel (de-		\Genplacrfullformat: rede-	
defined in \@gls@link)	116	defined to use accessibility in-	
\@GLSpl: removed \glslabel		formation	335
(defined in \@gls@link) ...	118	\genplacrfullformat: rede-	
\@Gls@: removed \glslabel (de-		defined to use accessibility in-	
defined in \@gls@link)	115	formation	335
\@Gls@entry@field: new ...	141	\glossentryname: added	
\@Glspl@: removed \glslabel		\glsdetoklabel	194
(defined in \@gls@link) ...	117	\gls@defglossaryentry: added	
\@acrlong: removed \glslabel		\glsdetoklabel	72
(defined in \@gls@link) ...	336	replaced #1 with \@glo@label	73
\@acrshort: removed \glslabel		replaced \ifthenelse with	
(defined in \@gls@link) ...	335	\ifdefequal	74
\@gls@: removed \glslabel (de-		\glsadd: added \glsdetoklabel	
defined in \@gls@link)	114	149
\@gls@access@display: new .	324	\glsaddkey: switched to using	
\@gls@entry@field: new ...	140	\@gls@field@link	70
\@gls@fetchfield: new	68	\glsdetoklabel: new	51
\@gls@field@link: new	119	\glsdisplaynumberlist: added	
\@gls@link: added \glsdetoklabel		\glsdetoklabel	148
.....	102	\glsdoifexistsorwarn: new ..	52
moved \@gls@link@opts		\glsentryaccess: switched to	
and \@gls@link@label to		using \@gls@entry@field .	322
\@gls@link	102	\glsentrydescaccess: switched	
\@gls@writedef: added		to using \@gls@entry@field	323
\glsdetoklabel	67	\glsentrydescpluralaccess:	
\@glsdisp: removed \glslabel		switched to using \@gls@entry@field	
(defined in \@gls@link) ...	119	323
\@glspl@: removed \glslabel		\glsentryfirstaccess: switched	
(defined in \@gls@link) ...	116	to using \@gls@entry@field	322
\@printglossary: added		\glsentryfirstplural: added	
\glsdetoklabel	176	\glsdetoklabel	144
General: changed default to		\glsentrylongaccess: switched	
\@empty instead of \relax ..	26	to using \@gls@entry@field	323

<code>\glsentrylongpluralaccess:</code>	short-long-desc: redefined to use accessibility information	341
switched to using <code>\@gls@entry@field</code>		
..... 323	<code>\ifglsdescsuppressed:</code> added	
<code>\glsentrypluralaccess:</code>	<code>\glsdetoklabel</code>	53
switched to using <code>\@gls@entry@field</code>	fixed typo	53
..... 322	<code>\ifglsentryexists:</code> added	
<code>\glsentryshortaccess:switched</code>	<code>\glsdetoklabel</code>	51
to using <code>\@gls@entry@field</code>	<code>\ifglschaschildren:</code> added	
323	<code>\glsdetoklabel</code>	52
<code>\glsentryshortpluralaccess:</code>	<code>\ifglschasdesc:added\glsdetoklabel</code>	
switched to using <code>\@gls@entry@field</code> 53	
..... 323	<code>\ifglschasfield:new</code>	54
<code>\glsentrysymbolaccess:</code>	<code>\ifglschaslong:added\glsdetoklabel</code>	
switched to using <code>\@gls@entry@field</code> 53	
..... 322	<code>\ifglschasparent:</code> added	
<code>\glsentrysymbolpluralaccess:</code>	<code>\glsdetoklabel</code>	52
switched to using <code>\@gls@entry@field</code>	<code>\ifglschasshort:</code> added	
..... 323	<code>\glsdetoklabel</code>	53
<code>\glsentrytextaccess:switched</code>	<code>\ifglschassymbol:</code> added	
to using <code>\@gls@entry@field</code>	<code>\glsdetoklabel</code>	53
322	replaced <code>\ifcsemtyp</code> with	
<code>\glsгенаcfmt:</code> redefined to use accessibility information ...	<code>\ifdefempty</code> and replaced	
332	<code>\ifx</code> with <code>\ifdefequal</code>	53
<code>\glsгенаentryfmt:</code> redefined to use accessibility information	<code>\ifglsused:added\glsdetoklabel</code>	
330 51	
<code>\gls hyperlink:added\glsdetoklabel</code>	<code>sm-short-long-desc: redefined to use accessibility information</code>	
..... 148 342	
<code>\glslocalreset: added</code>	<code>long-sc-short-desc: redefined to use accessibility information</code>	
<code>\glsdetoklabel</code> 340	
80	<code>long-short: redefined to use accessibility information</code>	
<code>\glslocalunset: added</code> 339	
<code>\glsdetoklabel</code>	<code>long-short-desc: redefined to use accessibility information</code>	
81	340	
<code>\glsmoveentry:added\glsdetoklabel</code>	<code>long-sm-short-desc: redefined to use accessibility information</code>	
..... 78 341	
replaced <code>\ifthenelse</code> with	<code>footnote: redefined to use accessibility information</code>	
<code>\ifdefequal</code> 345	
78	<code>footnote-desc: redefined to use accessibility information</code>	
<code>\glsrefentry:added\glsdetoklabel</code>	... 347	
..... 192	<code>footnote-sc: redefined to use accessibility information</code>	
<code>\glsreset:added\glsdetoklabel</code> 346	
..... 80	<code>footnote-sc-desc: redefined to use accessibility information</code>	
<code>\glsseelist:added\expandafter</code>	347	
commands	<code>footnote-sm: redefined to use accessibility information</code>	
173 347	
<code>\glsstepentry:added\glsdetoklabel</code>		
..... 191		
<code>\glsstepsubentry: added</code>		
<code>\glsdetoklabel</code>		
192		
<code>\glsunset:added\glsdetoklabel</code>		
..... 81		
short-long: commented spurious EOL		
211		
redefined to use accessibility information		
339		

footnote-sm-desc: redefined to use accessibility information	348		
\renewacronymstyle:new	...	210	
\showglocounter:	added		
\glsdetoklabel	240	
\showgloDESC:added	\glsdetoklabel	241
\showgloDESCaccess:	added		
\glsdetoklabel	354	
\showgloDESCplural:	added		
\glsdetoklabel	241	
\showgloDESCpluralaccess:	added		
\glsdetoklabel	354	
\showgloFIRST:added	\glsdetoklabel	239
\showgloFIRSTaccess:	added		
\glsdetoklabel	353	
\showgloFIRSTpl:	added		
\glsdetoklabel	239	
\showgloFIRSTpluralaccess:	added		
\glsdetoklabel	353	
\showgloFLAG:added	\glsdetoklabel	242
\showgloINDEX:added	\glsdetoklabel	242
\showgloLEVEL:added	\glsdetoklabel	239
\showgloLONG:added	\glsdetoklabel	242
\showgloLONGaccess:	added		
\glsdetoklabel	354	
\showgloLONGpluralaccess:	added		
\glsdetoklabel	354	
\showgloNAME:added	\glsdetoklabel	241
\showgloNAMEaccess:	added		
\glsdetoklabel	353	
\showgloPARENT:added		4.04	
\glsdetoklabel	238	
\showgloPLURAL:	added		
\glsdetoklabel	239	
\showgloPLURALaccess:	added		
\glsdetoklabel	353	
\showgloSHORT:added	\glsdetoklabel	242
\showgloSHORTaccess:	added		
\glsdetoklabel	354	
\showgloSHORTpluralaccess:	added		
\glsdetoklabel	354	
\showgloSORT:added	\glsdetoklabel	241
\showgloSYMBOL:	added		
\glsdetoklabel	241	
\showgloSYMBOLaccess:	added		
\glsdetoklabel	354	
\showgloSYMBOLplural:	added		
\glsdetoklabel	242	
\showgloSYMBOLpluralaccess:	added		
\glsdetoklabel	354	
\showgloTEXT:added	\glsdetoklabel	239
\showgloTEXTaccess:	added		
\glsdetoklabel	353	
\showgloTYPE:added	\glsdetoklabel	239
\showgloUSERI:added	\glsdetoklabel	240
\showgloUSERII:	added		
\glsdetoklabel	240	
\showgloUSERIII:	added		
\glsdetoklabel	240	
\showgloUSERIV:	added		
\glsdetoklabel	240	
\showgloUSERV:added	\glsdetoklabel	240
\showgloUSERVI:	added		
\glsdetoklabel	241	
dua: fixed bug in \acrfullfmt	.	215	
fixed bug in \Acrfullplfmt	.	215	
fixed bug in \acrfullplfmt	.	215	
redefined to use accessibility information	342	
dua-desc: commented spurious EOL	216	
redefined to use accessibility information	344	
\@@gls@noidx@nosanitizesort:	new	19
\@@gls@noidx@sanitizesort:	new	18
\@@gls@nosanitizesort:new	.	18	
\@@gls@sanitizesort:new	...	18	
\@glo@addchildren:new	178	
\@glo@do@sortentries:new	.	179	
\@glo@grabfirst:new	184	
\@glo@sortedinsert:new	...	180	
\@glo@sortentries:new	178	

<code>\@glo@sorthandler@case</code> : new	180	<code>\glsnoidxdisplayloclisthandler</code> :	
<code>\@glo@sorthandler@letter</code> :		new	186
new	180	<code>\glsnoidxloclist</code> : new	186
<code>\@glo@sorthandler@nocase</code> :		<code>\glsnoidxloclisthandler</code> :	
new	180	new	186
<code>\@glo@sorthandler@word</code> : new	180	<code>\glsnoidxstripaccents</code> : new	19
<code>\@glo@sortmacro@case</code> : new	182	alttree: moved hangindent and	
<code>\@glo@sortmacro@def</code> : new	182	parindent assignments out-	
<code>\@glo@sortmacro@def@do</code> : new	183	side level test	295
<code>\@glo@sortmacro@letter</code> : new	181	<code>\makeglossaries</code> : Moved def-	
<code>\@glo@sortmacro@nocase</code> : new	182	inition of <code>\glswrite</code> to	
<code>\@glo@sortmacro@standard</code> :		<code>\makeglossaries</code>	159
new	181	<code>\makenoidxglossaries</code> : new	161
<code>\@glo@sortmacro@use</code> : new	183	<code>\printglossary</code> : changed to use	
<code>\@glo@sortmacro@word</code> : new	181	new <code>\@printglossary</code>	174
<code>\@gls@getcounterprefix</code> :		<code>\printnoidxglossaries</code> : new	175
added warning if no prefix can		<code>\printnoidxglossary</code> : new	175
be formed	171	<code>\showgloclist</code> : new	242
<code>\@gls@getothergrouptitle</code> :		<code>\warn@noprintglossary</code> : Acti-	
new	197	vate warning in <code>\makeglossaries</code>	
<code>\@gls@noidx@do</code> : new	184	174
<code>\@gls@noref@warn</code> : new	165	<code>\writeist</code> : checked for definition	
<code>\@gls@reference</code> : new	186	of <code>\glswrite</code>	151, 155
<code>\@gls@warnonglossdefined</code> :			
new	17		
<code>\@gls@warnontheglossdefined</code> :		4.06	
new	17	<code>\@GLS@</code> : added <code>\glsifhyper</code>	116
<code>\@no@makeglossaries</code> : new	165	<code>\@GLSpl</code> : added <code>\glsifhyper</code>	118
<code>\@print@glossary</code> : new	176	<code>\@Gls@</code> : added <code>\glsifhyper</code>	115
<code>\@print@noidx@glossary</code> : new	183	<code>\@Glspl@</code> : added <code>\glsifhyper</code>	117
<code>\@printgloss@setsort</code> : new	175	<code>\@gls@</code> : added <code>\glsifhyper</code>	114
<code>\@printglossary</code> : new	175	<code>\@gls@numbersdef</code> : added hook	
General: added sort key to print-		to set toc title	28
gloss group	190	<code>\@gls@symbolsdef</code> : added hook	
<code>\compatibleglossentry</code> :		to set toc title	27
changed <code>\newcommand</code> to		<code>\@glsdisp</code> : added <code>\glsifhyper</code>	119
<code>\def</code> as is may or may not be		<code>\@glspl@</code> : added <code>\glsifhyper</code>	116
defined	318	General: added <code>\glsifhyper</code>	
<code>\compatiblesubglossentry</code> :		134–140
changed <code>\newcommand</code> to		acronym: added hook to set toc ti-	
<code>\def</code> as is may or may not be		tle	13
defined	318	acronyms: added hook to set toc	
<code>\defglsdisplayfirst</code> : fixed un-		title	14
wanted space	98	<code>\glsdefmain</code> : added hook to set	
<code>\glo@grabfirst</code> : new	184	toc title	13
<code>\gls@defglossaryentry</code> : re-			
placed <code>\ifx</code> with <code>\ifdefvoid</code>	77	<code>\@glossarysection</code> : added op-	
<code>\glsnoidxdisplayloc</code> : new	186	tional argument when using	
		unstarred version	39

\@gls@noidx@do: added \global in case it's used in a tabular- like style	185	moved check for first use to \@gls@link	119
\Acrfullplfmt: fixed no case change bug	206	\@glspl@: moved \glsifhyper moved check for first use to \@gls@link	116 116
\glsletentryfield: new	140	\@ignored@glossaries: new ..	59
4.08		General: added entrycounter op- tion to printgloss family .	188
\@ACRlong: added \do@gls@link@checkfirsthyper	337	added nopostdot option to printgloss family	188
\@ACRshort: added \do@gls@link@checkfirsthyper	336	added subentrycounter option to printgloss family	189
\@Acrlong: added \do@gls@link@checkfirsthyper	337	explicitly initialise hyper key .	100
\@Acrshort: added \do@gls@link@checkfirsthyper	335	moved \glsifhyper ...	134-140
\@GLS@: moved \glsifhyper ..	116	removed \@sACRlongpl	140
moved check for first use to \@gls@link	116	removed \@sAcrlongpl	139
\@GLSpl: moved \glsifhyper .	118	removed \@sacrlongpl	138
moved check for first use to \@gls@link	118	removed \@sACRlong	138
\@GLs@: moved \glsifhyper ..	115	removed \@sAcrlong	137
moved check for first use to \@gls@link	115	removed \@sacrlong	137
\@GLspl@: moved \glsifhyper	117	removed \@sACRshortpl ...	136
moved check for first use to \@gls@link	117	removed \@sAcrshortpl ...	135
\@acrlong: added \do@gls@link@checkfirsthyper	336	removed \@sacrshortpl ...	135
\@acrshort: added \do@gls@link@checkfirsthyper	335	removed \@sACRshort	134
\@closegls: new	158	removed \@sAcrshort	134
\@gls@: moved \glsifhyper ..	114	removed \@sacrshort	133
moved check for first use to \@gls@link	114	removed \@sgls@link	101
\@gls@doautomake: new	26	removed \@sGLSdescplural	126
\@gls@field@link: added as- signment of \do@gls@link@checkfirsthyper	119	removed \@sGLSdescplural	125
\@gls@forbidtextext: new	55	removed \@sglsdescplural	125
\@gls@hyp@opt: new	100	removed \@sGLSdesc	125
\@gls@link: removed redun- dancy	102	removed \@sGLSdesc	124
renamed \gls@type to \glstype	102	removed \@sglsdesc	124
\@gls@link@checkfirsthyper: new	101	removed \@sglsdisp	119
\@glsdisp: moved \glsifhyper	119	removed \@sGLSfirstplural	123
		removed \@sGlsfirstplural	123
		removed \@sglsfirstplural	122
		removed \@sGLSfirst	121
		removed \@sGlsfirst	121
		removed \@sglsfirst	121
		removed \@sGLSname	124
		removed \@sGlsname	124
		removed \@sglsname	123
		removed \@sGLSplural	122
		removed \@sGlsplural	122
		removed \@sglsplural	121
		removed \@sGLSpl	118
		removed \@sGlspl	117
		removed \@sglspl	116

removed \@sGLSsymbolplural	127	switched to using \@gls@hyp@opt	204
removed \@sGLssymbolplural	127	\ACRfullpl: removed \@s@ACRfullpl	206
removed \@sglssymbolplural	127	switched to using \@gls@hyp@opt	206
removed \@sGLSsymbol	127	\Acrfullpl: removed \@s@Acrfullpl	206
removed \@sGLssymbol	126	switched to using \@gls@hyp@opt	206
removed \@sglssymbol	126	\acrfullpl: removed \@s@acrfullpl	206
removed \@sGLStext	120	switched to using \@gls@hyp@opt	205
removed \@sglStext	120	\ACRlong: switched to using \@gls@hyp@opt	138
removed \@sGLSuseriii ...	130	\Acrlong: switched to using \@gls@hyp@opt	137
removed \@sGlsuseriii ...	130	\acrlong: switched to using \@gls@hyp@opt	136
removed \@sglsuseriii ...	130	\ACRlongpl: switched to using \@gls@hyp@opt	140
removed \@sGLSuserii	129	\Acrlongpl: switched to using \@gls@hyp@opt	139
removed \@sGlsuserii	129	\acrlongpl: switched to using \@gls@hyp@opt	138
removed \@sglsuserii	129	\ACRshort: switched to using \@gls@hyp@opt	134
removed \@sGLSuseriv	131	\Acrshort: switched to using \@gls@hyp@opt	134
removed \@sGlsuseriv	131	\acrshort: switched to using \@gls@hyp@opt	133
removed \@sglsuseriv	130	\ACRshortpl: switched to using \@gls@hyp@opt	136
removed \@sGLSuseri	128	\Acrshortpl: switched to using \@gls@hyp@opt	135
removed \@sGlsuseri	128	\acrshortpl: switched to using \@gls@hyp@opt	135
removed \@sglsuseri	128	\forallacronyms: new	50
removed \@sGLSuservi	133	\GLS: switched to using \@gls@hyp@opt	115
removed \@sGlsuservi	133	\Gls: switched to using \@gls@hyp@opt	114
removed \@sglsuservi	132	\gls: switched to using \@gls@hyp@opt	114
removed \@sGLSuserv	132	\gls@defglossaryentry: added check for ignored glossary ...	73
removed \@sGlsuserv	132	\gls@istfilebase: new	34
removed \@sglsuserv	131		
removed \@sGLS	115		
removed \@sGls	115		
removed \@sgls	114		
removed \@thirdofthree (de- fined in kernel)	113		
removed sPGLS	249		
removed sPgls	248		
removed spgls	247		
removed sPGLSpl	250		
removed sPglspl	248		
removed spglspl	247		
\ACRfull: removed \@s@ACRfull	205		
switched to using \@gls@hyp@opt	205		
\Acrfull: removed \@s@Acrfull	205		
switched to using \@gls@hyp@opt	205		
\acrfull: removed \@s@acrfull	204		

<code>\glsaddkey: removed \sGLS@user@⟨key⟩</code>	<code>\Glsname: switched to using</code>
..... 71	<code>\@gls@hyp@opt</code> 123
removed <code>\sGls@user@⟨key⟩</code> . 70	<code>\glsname: switched to using</code>
removed <code>\sgls@user@⟨key⟩</code> . 70	<code>\@gls@hyp@opt</code> 123
switched to using <code>\@gls@hyp@opt</code>	<code>\GLSpl: switched to using</code>
..... 70, 71	<code>\@gls@hyp@opt</code> 118
<code>\GLSdesc: switched to using</code>	<code>\Glspl: switched to using</code>
<code>\@gls@hyp@opt</code> 125	<code>\@gls@hyp@opt</code> 117
<code>\Glsdesc: switched to using</code>	<code>\glspl: switched to using</code>
<code>\@gls@hyp@opt</code> 124	<code>\@gls@hyp@opt</code> 116
<code>\glsdesc: switched to using</code>	<code>\GLSplural: switched to using</code>
<code>\@gls@hyp@opt</code> 124	<code>\@gls@hyp@opt</code> 122
<code>\GLSdescplural: switched to using</code>	<code>\Glsplural: switched to using</code>
<code>\@gls@hyp@opt</code> 126	<code>\@gls@hyp@opt</code> 122
<code>\Glsdescplural: switched to using</code>	<code>\glsplural: switched to using</code>
<code>\@gls@hyp@opt</code> 125	<code>\@gls@hyp@opt</code> 121
<code>\glsdescplural: switched to using</code>	<code>\glsSPACE: new</code> 205
<code>\@gls@hyp@opt</code> 125	<code>\GLSsymbol: switched to using</code>
<code>\glsdisablehyper: added</code>	<code>\@gls@hyp@opt</code> 126
<code>\KV@glslink@hyperfalse</code> to	<code>\Glsymbol: switched to using</code>
definition 113	<code>\@gls@hyp@opt</code> 126
<code>\glsdisp: switched to using</code>	<code>\glsymbol: switched to using</code>
<code>\@gls@hyp@opt</code> 119	<code>\@gls@hyp@opt</code> 126
<code>\glsdohyperlink: new</code> 112	<code>\GLSsymbolplural: switched to</code>
<code>\glsdohypertarget: new</code> 112	using <code>\@gls@hyp@opt</code> 127
<code>\glsenablehyper: added</code>	<code>\Glsymbolplural: switched to</code>
<code>\KV@glslink@hypertrue</code> to	using <code>\@gls@hyp@opt</code> 127
definition 113	<code>\glsymbolplural: switched to</code>
<code>\GLSfirst: switched to using</code>	using <code>\@gls@hyp@opt</code> 127
<code>\@gls@hyp@opt</code> 121	<code>\GLStext: switched to using</code>
<code>\Glsfirst: switched to using</code>	<code>\@gls@hyp@opt</code> 120
<code>\@gls@hyp@opt</code> 121	<code>\Glstext: switched to using</code>
<code>\glsfirst: switched to using</code>	<code>\@gls@hyp@opt</code> 120
<code>\@gls@hyp@opt</code> 120	<code>\glstext: switched to using</code>
<code>\GLSfirstplural: switched to</code>	<code>\@gls@hyp@opt</code> 120
using <code>\@gls@hyp@opt</code> 123	<code>\glstreenamfmt: new</code> 290
<code>\Glsfirstplural: switched to</code>	<code>\GLSuseri: switched to using</code>
using <code>\@gls@hyp@opt</code> 123	<code>\@gls@hyp@opt</code> 128
<code>\glsfirstplural: switched to</code>	<code>\Glsuseri: switched to using</code>
using <code>\@gls@hyp@opt</code> 122	<code>\@gls@hyp@opt</code> 128
<code>\glsifhyper: deprecated</code> 100	<code>\glsuseri: switched to using</code>
<code>\glslink: switched to using</code>	<code>\@gls@hyp@opt</code> 128
<code>\@gls@hyp@opt</code> 101	<code>\GLSuserii: switched to using</code>
<code>\glslinkcheckfirsthyperhook:</code>	<code>\@gls@hyp@opt</code> 129
new 102	<code>\Glsuserii: switched to using</code>
<code>\glslinkvar: new</code> 100	<code>\@gls@hyp@opt</code> 129
<code>\GLSname: switched to using</code>	<code>\glsuserii: switched to using</code>
<code>\@gls@hyp@opt</code> 124	<code>\@gls@hyp@opt</code> 129

\GLSuseriii: switched to using \@gls@hyp@opt	130	4.09	automake: new	26
\Glsuseriii: switched to using \@gls@hyp@opt	130		\glsaddkey: fixed bug in user commands	70
\glsuseriii: switched to using \@gls@hyp@opt	129	4.10	\@Gls@acentryname: new ...	141
\GLSuseriv: switched to using \@gls@hyp@opt	131		\@Gls@entryname: new	141
\Glsuseriv: switched to using \@gls@hyp@opt	131		\@gls@glossary: Renamed \@glossary to \@gls@glossary	167
\glsuseriv: switched to using \@gls@hyp@opt	130		\glspercentchar: new	151
\GLSuserv: switched to using \@gls@hyp@opt	132		\glstildechar: new	151
\Glsuserv: switched to using \@gls@hyp@opt	132	4.11	almtree: moved space after sym- bol	296, 297
\glsuserv: switched to using \@gls@hyp@opt	131		\@@do@wrglossary: added hook	169
\GLSuservi: switched to using \@gls@hyp@opt	133		sanitize: none option	21
\Glsuservi: switched to using \@gls@hyp@opt	132		\gls@wrglossary: renamed from \@wrglossary to \gls@wrglossary	167
\glsuservi: switched to using \@gls@hyp@opt	132	4.12	\glsaddprotectedpagefmt: new	168
\ifignoredglossary: new	59		\glsbackslash: new	150
altlongragged4col: fixed bug that displayed description in- stead of symbol	272		\@gls@addpredefinedattributes: Added glsignore attribute ...	43
\newglossary: added starred ver- sion	56		\@gls@adjustmode: new	149
\newignoredglossary: new ...	58		\@gls@notranslatorhook: re- moved	21
\ns@newglossary: added \@glotype@<name>@log ...	57		\@gls@toc: added \protect to \numberline	40
new	56		\@gls@usetranslator: new ...	22
\p@gls@hyp@opt: new	100		\glsacrpluralsuffix: new ...	31
\PGLS: changed to use \@gls@hyp@opt	249		\glsadd: added check for vertical mode	149
\PglS: changed to use \@gls@hyp@opt	248		\glsaddallunused: replaced @gobble with glsignore ...	150
\pgls: changed to use \@gls@hyp@opt	247		\glsifusedtranslatordict: new	22
\PGLSpl: changed to use \@gls@hyp@opt	250		\glsignore: new	150
\PglSpl: changed to use \@gls@hyp@opt	248	4.13	\glsupacrpluralsuffix: new .	31
\pglSpl: changed to use \@gls@hyp@opt	247	4.14	\ProvidesGlossariesLang: new	31
\s@gls@hyp@opt: new	100		\RequireGlossariesLang: new	31
\s@newglossary: new	56		\indexspace: new ...	259, 274, 290
			\@@glslocalreset: new	82
			\@@glslocalunset: new	81

\@glsreset:new	82	\cGls:new	86
\@glsunset:new	81	\cglS:new	86
\@newglossaryentry@defcounters:		\cGlsformat:new	87
new	83	\cglSformat:new	86
\@cGls:new	86	\cGlspl:new	87
\@cGls@:new	86	\cglSpl:new	87
\@cGlspl@:new	88	\cGlsplformat:new	88
\@cglS:new	86	\cglSplformat:new	87
\@cglS@:new	86	\gls@defdocnewglossaryentry:	
\@cglSpl:new	87	new	66
\@cglSpl@:new	87	\glsenableentrycount:new ..	83
\@gls@entry@count:new	86	\glslocalreset: switched to	
\@gls@increment@currcount:		\@glslocalreset	80
new	85	\glslocalunset: switched to	
\@gls@local@increment@currcount:		\@glslocalunset	81
new	85	\glsreset: switched to	
\@gls@write@entrycounts:		\@glsreset	80
new	85	\glsunset: switched to	
\@glslocalreset:new	81	\@glsunset	81
\@glslocalunset:new	81		
\@glsreset:new	82		
\@glsunset:new	81		
\@newglossaryentry@defcounters:		4.15	
new	78	General: bug fix replaced	
		\@glo@type with \glstype	139

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>	170
<code>\@do@wrglossary</code>	169
<code>\@glo@assign@sortkey</code>	190
<code>\@glossarysec</code>	6
<code>\@glossaryseclabel</code>	6
<code>\@glossarysecstar</code>	6
<code>\@gls@default@entryfmt</code>	326
<code>\@gls@expand@field</code>	65
<code>\@gls@fixbraces</code>	172
<code>\@gls@noidx@nosanitizesort</code> .	19
<code>\@gls@noidx@sanitizesort</code> ...	18
<code>\@gls@nosanitizesort</code>	18
<code>\@gls@sanitizesort</code>	18
<code>\@glslocalreset</code>	82
<code>\@glslocalunset</code>	81
<code>\@glsreset</code>	82
<code>\@glsunset</code>	81
<code>\@newglossaryentry@defcounters</code>	83
<code>\@ACRlong</code>	337
<code>\@ACRshort</code>	336
<code>\@Acrlong</code>	337
<code>\@Acrshort</code>	335
<code>\@GLS@</code>	115
<code>\@GLSpl</code>	118
<code>\@Gls@</code>	115
<code>\@Gls@acrentryname</code>	141
<code>\@Gls@entry@field</code>	141
<code>\@Gls@entryname</code>	141
<code>\@Glspl@</code>	117
<code>\@PGLS</code>	249
<code>\@PGLS@</code>	249
<code>\@PGLSpl</code>	250
<code>\@PGLSpl@</code>	250
<code>\@Pgls</code>	248
<code>\@Pgls@</code>	248
<code>\@Pglspl</code>	248
<code>\@Pglspl@</code>	249
<code>\@acrfull</code>	204
<code>\@acrlong</code>	336
<code>\@acrshort</code>	335
<code>\@addtoacronymlists</code>	14
<code>\@cGls</code>	86
<code>\@cGls@</code>	86
<code>\@cGlspl@</code>	88
<code>\@cgls</code>	86
<code>\@cgls@</code>	86
<code>\@cglspl</code>	87
<code>\@cglspl@</code>	87
<code>\@closegls</code>	158
<code>\@delimN</code>	201
<code>\@delimR</code>	200
<code>\@disable@onlypremakeg</code>	30
<code>\@disable@premakecs</code>	30
<code>\@disabled@glsaddxdycounters</code>	42
<code>\@do@seeglossary</code>	171
<code>\@do@wrglossary</code>	168, 299
<code>\@glo@addchildren</code>	178
<code>\@glo@default@sorttype</code>	10
<code>\@glo@do@sortentries</code>	179
<code>\@glo@grabfirst</code>	184
<code>\@glo@no@assign@sortkey</code>	190
<code>\@glo@seeautonumberlist</code>	8
<code>\@glo@sortedinsert</code>	180
<code>\@glo@sortentries</code>	178
<code>\@glo@sorthandler@case</code>	180
<code>\@glo@sorthandler@letter</code> ...	180
<code>\@glo@sorthandler@nocase</code> ...	180
<code>\@glo@sorthandler@word</code>	180
<code>\@glo@sortmacro@case</code>	182
<code>\@glo@sortmacro@def</code>	182
<code>\@glo@sortmacro@def@do</code>	183
<code>\@glo@sortmacro@letter</code>	181
<code>\@glo@sortmacro@nocase</code>	182
<code>\@glo@sortmacro@standard</code> ...	181
<code>\@glo@sortmacro@use</code>	183
<code>\@glo@sortmacro@word</code>	181
<code>\@glo@storeentry</code>	78
<code>\@glo@types</code>	55
<code>\@glossary@default@style</code>	7
<code>\@glossaryentryfield</code>	78
<code>\@glossarysection</code>	39
<code>\@glossarysubentryfield</code>	78
<code>\@gls</code>	114
<code>\@gls@</code>	114

<code>\@gls@link</code>	101	<code>\@gls@local@increment@currcount</code>	85
<code>\@gls@access@display</code>	324	85
<code>\@gls@addpredefinedattributes</code>		<code>\@gls@makefirstuc</code>	252
.....	43	<code>\@gls@missingnumberlist</code>	64
<code>\@gls@adjustmode</code>	149	<code>\@gls@noaccess</code>	320
<code>\@gls@assign@symbol@field</code> ...	18	<code>\@gls@noexpand@field</code>	64
<code>\@gls@assign@symbolplural@field</code>		<code>\@gls@noexpand@fields</code>	64
.....	18	<code>\@gls@nohyperlist</code>	16
<code>\@gls@checkactual</code>	110	<code>\@gls@noidx@do</code>	184
<code>\@gls@checkbar</code>	109	<code>\@gls@noidx@setsanitizesort</code> .	21
<code>\@gls@checkescactual</code>	108	<code>\@gls@noref@warn</code>	165
<code>\@gls@checkescbar</code>	108	<code>\@gls@notranslatorhook</code>	21
<code>\@gls@checkesclevel</code>	109	<code>\@gls@numbersdef</code>	27
<code>\@gls@checkescquote</code>	107	<code>\@gls@onlypremakeg</code>	29
<code>\@gls@checklevel</code>	110	<code>\@gls@provide@newglossary</code> ...	55
<code>\@gls@checkmkidxchars</code>	106	<code>\@gls@reference</code>	186
<code>\@gls@checkquote</code>	107	<code>\@gls@renewglossary</code>	167
<code>\@gls@codepage</code>	48	<code>\@gls@sanitizedesc</code>	17
<code>\@gls@counterwithin</code>	10	<code>\@gls@sanitizename</code>	18
<code>\@gls@declareoption</code>	7	<code>\@gls@sanitizesort</code>	18
<code>\@gls@default@value</code>	60	<code>\@gls@sanitizesymbol</code>	18
<code>\@gls@do@acronymsdef</code>	14	<code>\@gls@saveentrycounter</code>	103
<code>\@gls@doautomake</code>	26	<code>\@gls@setacrstyle</code>	24
<code>\@gls@entry@count</code>	86	<code>\@gls@setcounter</code>	58
<code>\@gls@entry@field</code>	140	<code>\@gls@setupsort@def</code>	11
<code>\@gls@escbsdq</code>	105	<code>\@gls@setupsort@standard</code>	10
<code>\@gls@expand@fields</code>	65	<code>\@gls@setupsort@use</code>	11
<code>\@gls@fetchfield</code>	68	<code>\@gls@startswithexpandonce</code> ..	65
<code>\@gls@field@link</code>	119	<code>\@gls@symbolsdef</code>	27
<code>\@gls@fixbraces</code>	172	<code>\@gls@tmpb</code>	107
<code>\@gls@forbidtexext</code>	55	<code>\@gls@toc</code>	40
<code>\@gls@getcounter</code>	58	<code>\@gls@updatechecked</code>	106
<code>\@gls@getcounterprefix</code>	171	<code>\@gls@usetranslator</code>	22
<code>\@gls@getgrouptitle</code>	197	<code>\@gls@warnonglossdefined</code>	17
<code>\@gls@getothergrouptitle</code> ...	197	<code>\@gls@warnonthe glossdefined</code> .	17
<code>\@gls@glossary</code>	167	<code>\@gls@write@entrycounts</code>	85
<code>\@gls@hyp@opt</code>	100	<code>\@gls@writedef</code>	67
<code>\@gls@hypergroup</code>	255	<code>\@gls@xdy@Lclass@Alpha-page-numbers</code>	
<code>\@gls@ifinlist</code>	41	45
<code>\@gls@increment@currcount</code> ...	85	<code>\@gls@xdy@Lclass@Appendix-page-numbers</code>	
<code>\@gls@indexdef</code>	28	45
<code>\@gls@keymap</code>	68, 245	<code>\@gls@xdy@Lclass@Roman-page-numbers</code>	
<code>\@gls@link</code>	102	45
<code>\@gls@link@checkfirsthyper</code> .	101	<code>\@gls@xdy@Lclass@alpha-page-numbers</code>	
<code>\@gls@loadlist</code>	8	45
<code>\@gls@loadlong</code>	8	<code>\@gls@xdy@Lclass@arabic-page-numbers</code>	
<code>\@gls@loadsUPER</code>	8	45
<code>\@gls@loadtree</code>	8	<code>\@gls@xdy@Lclass@arabic-section-numbers</code>	
		45

<code>\@gls@xdy@Lclass@roman-page-numbers</code>	<code>\@printglossary</code>	175
.....	<code>\@set@glo@numformat</code>	104, 300
<code>\@gls@xdy@locationlist</code>	<code>\@wrglossary@pageformat</code>	169
.....	<code>\@wrglossarynumberhook</code>	169
<code>\@gls@xdy@checkbackslash</code>	<code>\@xdy@main@language</code>	25
.....	<code>\@xdy@attributelist</code>	41
<code>\@gls@xdy@checkquote</code>	<code>\@xdy@attributes</code>	40
.....	<code>\@xdy@language</code>	48
<code>\@gls@Alphacompositor</code>	<code>\@xdy@lettergroups</code>	49
.....	<code>\@xdy@locationclassorder</code>	46
<code>\@gls@acronymlists</code>	<code>\@xdy@locref</code>	41
.....	<code>\@xdy@requiredstyles</code>	47
<code>\@gls@defaultplural</code>	<code>\@xdy@sortrules</code>	47
.....	<code>\@xdy@useralphabets</code>	44
<code>\@gls@defaultsort</code>	<code>\@xdy@userlocationdefs</code>	45
.....	<code>\@xdy@userlocationnames</code>	45
<code>\@gls@disp</code>		
.....		
<code>\@gls@firstletter</code>		
.....		
<code>\@gls@hypernumber</code>		
.....		
<code>\@gls@link</code>		
.....		
<code>\@gls@localreset</code>		
.....		
<code>\@gls@localunset</code>		
.....		
<code>\@gls@minrange</code>		
.....		
<code>\@gls@nextpages</code>		
.....		
<code>\@gls@nodesc</code>		
.....		
<code>\@gls@noname</code>		
.....		
<code>\@gls@nonextpages</code>		
.....		
<code>\@gls@openfile</code>		
.....		
<code>\@gls@order</code>		
.....		
<code>\@gls@spl@</code>		
.....		
<code>\@gls@reset</code>		
.....		
<code>\@glstarget</code>		
.....		
<code>\@gls@unset</code>		
.....		
<code>\@gls@widestname</code>		
.....		
<code>\@gls@writefiles</code>		
.....		
<code>\@ignored@glossaries</code>		
.....		
<code>\@istfilename</code>		
.....		
<code>\@makeglossary</code>		
.....		
<code>\@mfu@nocaplist</code>		
.....		
<code>\@newglossary</code>		
.....		
<code>\@newglossaryentry@defcounters</code>		
.....		
<code>\@newglossaryentryposthook</code>		
..		
<code>\@newglossaryentryprehook</code>		
...		
<code>\@no@makeglossaries</code>		
.....		
<code>\@no@post@desc</code>		
.....		
<code>\@nopostdesc</code>		
.....		
<code>\@onlypremakeg</code>		
.....		
<code>\@p@glossarysection</code>		
.....		
<code>\@pgls</code>		
.....		
<code>\@pgls@</code>		
.....		
<code>\@pglspl</code>		
.....		
<code>\@pglspl@</code>		
.....		
<code>\@print@glossary</code>		
.....		
<code>\@print@noidx@glossary</code>		
.....		
<code>\@printgloss@setsort</code>		
.....		
	A	
<code>\Ac</code>	220
<code>\ac</code>	220
<code>access (key)</code>	319
<code>accsupp package</code>	318
<code>\accsuppglossaryentryfield</code>	..	338
<code>\accsuppglossarysubentryfield</code>	338
<code>\Acf</code>	220
<code>\acf</code>	220
<code>\Acfp</code>	220
<code>\acfp</code>	220
<code>\Acl</code>	220
<code>\acl</code>	219
<code>\Aclp</code>	220
<code>\aclp</code>	220
<code>\Acp</code>	220
<code>\acp</code>	220
<code>\acrfootnote</code>	222
<code>\ACRfull</code>	205
<code>\Acrfull</code>	205
<code>\acrfull</code>	204, 208, 217, 345
<code>\ACRfullfmt</code>	205
<code>\Acrfullfmt</code>	205
<code>\acrfullfmt</code>	204
<code>\acrfullformat</code>	95, 204
<code>\ACRfullpl</code>	206
<code>\Acrfullpl</code>	206
<code>\acrfullpl</code>	205
<code>\ACRfullplfmt</code>	207
<code>\Acrfullplfmt</code>	206
<code>\acrfullplfmt</code>	206

<code>\acrlinkfootnote</code>	222	<code>\Acsp</code>	219
<code>\acrlinkfullformat</code>	204	<code>\acsp</code>	219
<code>\ACRlong</code>	138	<code>\addglossarytocaptions</code>	32
<code>\Acrlong</code>	137	<code>\addto</code>	32
<code>\acrlong</code>	136	<code>align (environment)</code>	80, 103
<code>\ACRlongpl</code>	140	<code>altlist (style)</code>	260
<code>\Acrlongpl</code>	139	<code>altlistgroup (style)</code>	260
<code>\acrlongpl</code>	138	<code>altlisthypergroup (style)</code>	261
<code>\acrnameformat</code>	207, 228	<code>altlong4col (style)</code>	267
<code>\acrnolinkfootnote</code>	222	<code>altlong4colborder (style)</code>	267
<code>acronym (option)</code>	13	<code>altlong4colheader (style)</code>	267
acronym styles:		<code>altlong4colheaderborder (style)</code>	267
<code>dua</code>	214, 342	<code>altlongragged4col (style)</code>	271
<code>dua-desc</code>	216, 344	<code>altlongragged4colborder (style)</code>	273
<code>footnote</code>	216, 345	<code>altlongragged4colheader (style)</code>	272
<code>footnote-desc</code>	218, 347	<code>altlongragged4colheaderborder</code>	
<code>footnote-sc</code>	218, 346	<code>(style)</code>	273
<code>footnote-sc-desc</code>	218, 347	<code>\altnewglossary</code>	57
<code>footnote-sm</code>	218, 347	<code>altsuper4col (style)</code>	283
<code>footnote-sm-desc</code>	219, 348	<code>altsuper4colborder (style)</code>	283
<code>long-sc-short</code>	211	<code>altsuper4colheader (style)</code>	283
<code>long-sc-short-desc</code>	213, 340	<code>altsuper4colheaderborder</code>	
<code>long-short</code>	210, 339	<code>(style)</code>	284
<code>long-short-desc</code>	212, 340	<code>altsuperragged4col (style)</code>	288
<code>long-sm-short</code>	212	<code>altsuperragged4colborder</code>	
<code>long-sm-short-desc</code>	213, 341	<code>(style)</code>	289
<code>sc-short-long</code>	212	<code>altsuperragged4colheader</code>	
<code>sc-short-long-desc</code>	213, 341	<code>(style)</code>	289
<code>short-long</code>	211, 339	<code>altsuperragged4colheaderborder</code>	
<code>short-long-desc</code>	213, 341	<code>(style)</code>	290
<code>sm-short-long</code>	212	<code>almtree (style)</code>	295
<code>sm-short-long-desc</code>	214, 342	<code>almtreegroup (style)</code>	297
<code>\acronymentry</code>	209	<code>almtreehypergroup (style)</code>	297
<code>\acronymfont</code>		<code>amsgen package</code>	4, 99
.....	95, 207, 224, 228, 231, 234	<code>amsmath package</code>	80
<code>acronymlists (option)</code>	15	<code>\andname</code>	31
<code>\acronymname</code>	30	<code>array package</code>	268, 284
<code>acronyms (option)</code>	14	<code>article class</code>	171
<code>\acronymsort</code>	209	<code>automake (option)</code>	26
<code>\acronymtype</code>	13, 203		
<code>\acrpluralsuffix</code>	203	B	
<code>\ACRshort</code>	134	<code>babel package</code>	22, 30, 32, 48
<code>\Acrshort</code>	134		
<code>\acrshort</code>	133	C	
<code>\ACRshortpl</code>	136	<code>\capitalisewords</code>	252
<code>\Acrshortpl</code>	135	<code>\cGls</code>	86
<code>\acrshortpl</code>	135	<code>\cgls</code>	86
<code>\acrshortpl</code>	135	<code>\cGlsformat</code>	87
<code>\Acs</code>	219	<code>\cglsformat</code>	86
<code>\acs</code>	219	<code>\cGlspl</code>	87

`\cglsp` 87
`\cGlsplformat` 88
`\cglspformat` 87
`\compatglossarystyle` 304
`compatible-2.07` (option) 27
`compatible-3.07` (option) 27
`\compatibleglossentry` .. 193, 318
`\compatiblesubglossentry` 195, 318
`counter` (key) 61
`counter` (option) 16
`\CustomAcronymFields` 237
`\CustomNewAcronymDef` 237

D

`datatool` package 180
`\DeclareAcronymList` 14
`\DefaultNewAcronymDef` .. 221, 348
`\defentryfmt` 99
`\defgldisplay` 98
`\defgldisplayfirst` 98
`\defglentry` 57
`\defglentryfmt` 55, 59, 61, 90
`\DefineAcronymSynonyms` 219
`\delimN` 36, 200
`\delimR` 36, 200
`description` (environment) 259
`description` (key) 59
`description` (option) 24
`descriptionaccess` (key) 319
`\DescriptionDUANewAcronymDef` 225
`\DescriptionFootnoteNewAcronymDef`
..... 223, 349
`\descriptionname` 31
`\DescriptionNewAcronymDef` ..
..... 227, 350
`descriptionplural` (key) 60
`descriptionpluralaccess` (key) 320
`\detokenize` 51
`doc` package 5, 12
`document` (environment) 66, 83
`dua` (acrstyle) 214, 342
`dua` (option) 24
`dua-desc` (acrstyle) 216, 344
`\DUANewAcronymDef` 234

E

`\ecapitalisewords` 253
`\emakefirstuc` 252
`entrycounter` (option) 9
`entrycounterwithin` (option) 9

`\entryname` 31
environments:
 `align` 80, 103
 `description` 259
 `document` 66, 83
 `longtable` 8, 238, 262–273
 `multicols` 274
 `supertabular` ... 8, 238, 277–290
 `theglossary` ... 5, 17, 36, 37,
 193, 199, 275, 276, 292, 293, 295
 `theindex` 290
`equation` (counter) 103, 104
`etoolbox` package 5, 250

F

file types
 `.aux` 177
 `.glo` 78
 `.ist` 150, 157
 `.toc` 40
 `.xdy` 34
 `glo` 244
`\findrootlanguage` 48
`first` (key) 60
`firstaccess` (key) 319
`\firstacronymfont` 95, 207
`firstplural` (key) 60
`firstpluralaccess` (key) 319
`footnote` (acrstyle) 216, 345
`footnote` (option) 24
`footnote-desc` (acrstyle) .. 218, 347
`footnote-sc` (acrstyle) 218, 346
`footnote-sc-desc` (acrstyle) 218, 347
`footnote-sm` (acrstyle) 218, 347
`footnote-sm-desc` (acrstyle) 219, 348
`\FootnoteNewAcronymDef` . 229, 351
`\forallacronyms` 50
`\forallglossaries` 49
`\forallglsentries` 50
`\forglsentries` 50

G

`garamondx` package 203
`\Genacrfullformat` 97, 335
`\genacrfullformat` 97, 334
`\GenericAcronymFields` 208
`\Genplacrfullformat` 97, 335
`\genplacrfullformat` 97, 335
`\glo@grabfirst` 184
`\glolinkprefix` 103

glossareentry (counter)	191	glossary package	1, 202
glossaries package	28,	glossary styles:	
48, 151, 238, 244, 259, 298, 318		altlist	260, 261, 306
glossaries-accsupp package ...	78, 318	altlist	260
\GlossariesWarning	16	altlistgroup	260, 261, 306
\GlossariesWarningNoLine	16	altlistgroup	260
\glossary	56, 157, 166, 198	altlisthypergroup ...	261, 306
glossary counters:		altlisthypergroup	261
glossaryentry	191	altlong4col	267, 271, 308
glossarysubentry	191	altlong4col	267
glossary keys:		altlong4colborder ...	267, 308
access	319	altlong4colborder	267
counter	61	altlong4colheader ...	267, 308
description	59	altlong4colheader	267
descriptionaccess	319	altlong4colheaderborder .	
descriptionplural	60	267, 308
descriptionpluralaccess .	320	altlong4colheaderborder .	267
first	60	altlongragged4col 271–273, 309	
firstaccess	319	altlongragged4col	271
firstplural	60	altlongragged4colborder .	
firstpluralaccess	319	273, 310
long	63	altlongragged4colborder .	273
longaccess	320	altlongragged4colheader .	
longplural	63	272, 310
longpluralaccess	320	altlongragged4colheader .	272
name	59	altlongragged4colheaderborder	
nonumberlist	62	273, 310
parent	62	altlongragged4colheaderborder	
plural	60	273
pluralaccess	319	altsuper4col . 283, 284, 288, 317	
see	61	altsuper4col	283
short	63	altsuper4colborder ..	283, 317
shortaccess	320	altsuper4colborder	283
shortplural	63	altsuper4colheader ..	283, 317
shortpluralaccess	320	altsuper4colheader	283
sort	60	altsuper4colheaderborder	
symbol	61	284, 317
symbolaccess	319	altsuper4colheaderborder 284	
symbolplural	61	altsuperragged4col 288–290, 315	
symbolpluralaccess	319	altsuperragged4col	288
text	60	altsuperragged4colborder	
textaccess	319	289, 316
type	61	altsuperragged4colborder 289	
user1	62	altsuperragged4colheader	
user2	62	289, 315
user3	62	altsuperragged4colheader 289	
user4	62	altsuperragged4colheaderborder	
user5	62	290, 316
user6	63		

altsuperragged4colheaderborder	290
alttree	276, 295, 297, 312
alttree	295
alttreegroup	297, 313
alttreegroup	297
alttreehypergroup	297, 313
alttreehypergroup	297
index	274, 290–292, 310
index	290
indexgroup	291, 292, 310
indexgroup	291
indexhypergroup	292, 311
indexhypergroup	292
inline	305
inline	256
list	7, 259–261, 305
list	259
listdotted	261, 262, 306
listdotted	261
listgroup	259, 260, 306
listgroup	259
listhypergroup	260, 306
listhypergroup	260
long	262–264, 268, 307, 308
long	262
long3col	264, 265, 307
long3col	264
long3colborder	264, 307
long3colborder	264
long3colheader	265, 307
long3colheader	265
long3colheaderborder	265, 307
long3colheaderborder	265
long4col	265–267, 308
long4col	265
long4colborder	266, 308
long4colborder	266
long4colheader	266, 308
long4colheader	266
long4colheaderborder	266, 308
long4colheaderborder	266
longborder	263, 307
longborder	263
longheader	263, 307
longheader	263
longheaderborder	263, 307
longheaderborder	263
longragged	268–270
longragged	268
longragged3col	270, 271, 309
longragged3col	270
longragged3colborder	271, 309
longragged3colborder	271
longragged3colheader	271, 309
longragged3colheader	271
longragged3colheaderborder	271, 309
longragged3colheaderborder	271
longraggedborder	269, 309
longraggedborder	269
longraggedheader	269, 309
longraggedheader	269
longraggedheaderborder	270, 309
longraggedheaderborder	270
mcolalttree	277, 314
mcolalttree	276
mcolalttreegroup	277, 314
mcolalttreegroup	277
mcolalttreehypergroup	277, 314
mcolalttreehypergroup	277
mcolindex	274, 313
mcolindex	274
mcolindexgroup	274, 313
mcolindexgroup	274
mcolindexhypergroup	274, 313
mcolindexhypergroup	274
mcoltree	275, 313
mcoltree	275
mcoltreegroup	314
mcoltreegroup	275
mcoltreehypergroup	275, 314
mcoltreehypergroup	275
mcoltreenoname	276, 314
mcoltreenoname	275
mcoltreenonamegroup	276, 314
mcoltreenonamegroup	276
mcoltreenonamehypergroup	276, 314
mcoltreenonamehypergroup	276
sublistdotted	306
sublistdotted	262
super	278, 279, 286, 316
super	278
super3col	279–281, 316
super3col	279
super3colborder	280, 316

super3colborder	280	treenonamehypergroup	294
super3colheader	280, 317	glossary-hypernav package	150
super3colheader	280	glossary-list package	7, 8, 259
super3colheaderborder	281, 317	glossary-long package	8, 262, 271, 277, 278
super3colheaderborder	281	glossary-longragged package	268
super4col	281–283, 317	glossary-mcols package	273
super4col	281	glossary-super package	8, 262, 277, 284, 288
super4colborder	282, 317	glossary-superragged package	284
super4colborder	282	glossary-tree package	8, 290
super4colheader	282, 317	glossaryentry (counter)	9, 191, 192
super4colheader	282	glossaryentry (counter)	191
super4colheaderborder	282, 317	\glossaryentryfield	195, 199
super4colheaderborder	282	\glossaryentrynumber	190
superborder	278, 316	\glossaryentrynumbers	7, 36, 175, 176
superborder	278	\glossaryheader	193, 199
superheader	279, 316	\glossarymark	38
superheader	279	\glossaryname	30, 32
superheaderborder	279, 316	\glossarypostamble	37, 199
superheaderborder	279	\glossary preamble	37, 199
superragged	285, 286, 314	\glossarysection	6, 37, 56
superragged	285	\glossarystyle	198, 238
superragged3col	286–288, 315	glossarysubentry (counter)	10, 191–193
superragged3col	286	glossarysubentry (counter)	191
superragged3colborder	287, 315	\glossarysubentryfield	196
superragged3colborder	287	\glossentry	61, 193
superragged3colheader	287, 315	\Glossentrydesc	194
superragged3colheader	287	\glossentrydesc	194, 337
superragged3colheaderborder	288, 315	\Glossentryname	194
superraggedborder	285, 315	\glossentryname	194, 337
superraggedborder	285	\Glossentrysymbol	195
superraggedheader	286, 315	\glossentrysymbol	194, 337
superraggedheader	286	\GLS	115
superraggedheaderborder	286, 315	\Gls	114, 117, 251
superraggedheaderborder	286	\gls	4, 61, 88, 100, 114–116, 119–132, 192, 246
superraggedright3colheaderborder	288	\gls@Alphpage	168
tree	275, 292, 293, 295, 311	\gls@alphpage	168
tree	292	\gls@assign@desc	71
treegroup	275, 293, 311	\gls@assign@desc@field	18
treegroup	293	\gls@assign@desc@plural@field	18
treehypergroup	293, 311	\gls@assign@field	66
treehypergroup	293	\gls@assign@name@field	18
treenoname	275, 293, 294, 311	\gls@assign@type@field	17
treenoname	293	\gls@checkisacronymlist	15
treenonamegroup	294, 312	\gls@checkseeallowed	61
treenonamegroup	294		
treenonamehypergroup	294, 312		

<code>\gls@codepage</code>	25	<code>\GLSdesc</code>	125
<code>\gls@defdocnewglossaryentry</code>	66	<code>\Glsdesc</code>	124
<code>\gls@defglossaryentry</code>	72	<code>\glsdesc</code>	124, 125
<code>\gls@disablepagerefexpansion</code>	168	<code>\GLSdescplural</code>	126
<code>\gls@docclearpage</code>	39	<code>\Glsdescplural</code>	125
<code>\gls@doentryfmt</code>	55	<code>\glsdescplural</code>	125, 126
<code>\gls@glossary</code>	167	<code>\glsdescriptionaccessdisplay</code>	325
<code>\gls@hypergrouprrerun</code>	255	<code>\glsdescriptionpluralaccessdisplay</code>	325
<code>\gls@ifnotmeasuring</code>	80	<code>\glsdescwidth</code>	262, 268, 277, 284
<code>\gls@istfilebase</code>	34	<code>\glsdetoklabel</code>	51
<code>\gls@level</code>	64	<code>\glsdisablehyper</code>	113
<code>\gls@noidxglossary</code>	165	<code>\glsdisp</code>	119
<code>\gls@numberpage</code>	168	<code>\glsdisplay</code>	88, 98, 113
<code>\gls@protected@pagefmts</code>	168	<code>\glsdisplayfirst</code>	88, 98, 113
<code>\gls@Romanpage</code>	168	<code>\glsdisplaynumberlist</code>	147, 163, 186
<code>\gls@romanpage</code>	168	<code>\glsdohyperlink</code>	112
<code>\gls@save@numberlist</code>	174	<code>\glsdohypertarget</code>	112
<code>\gls@suffixF</code>	35	<code>\glsdoifexists</code>	51
<code>\gls@suffixFF</code>	36	<code>\glsdoifexistsorwarn</code>	52
<code>\gls@wrglossary</code>	167	<code>\glsdoifnoexists</code>	51
<code>\glsaccessdisplay</code>	325	<code>\glsdoparenifnotempty</code>	231
<code>\glsaccsupp</code>	323	<code>\glsenableentrycount</code>	83
<code>\glsacrpluralsuffix</code>	31	<code>\glsenablehyper</code>	113
<code>\glsadd</code>	88, 148, 149, 198	<code>\glsentryaccess</code>	322
<code>\glsadd options</code>		<code>\glsentrycounter</code>	103
<code>counter</code>	149	<code>\glsentrycounterlabel</code>	192
<code>format</code>	149, 200	<code>\glsentrycurrcount</code>	83
<code>\glsaddall</code>	51, 88, 149	<code>\Glsentrydesc</code>	142
<code>\glsaddall options</code>		<code>\glsentrydesc</code>	142
<code>types</code>	149	<code>\glsentrydescaccess</code>	323
<code>\glsaddallunused</code>	150	<code>\Glsentrydescplural</code>	143
<code>\glsaddkey</code>	69	<code>\glsentrydescplural</code>	142
<code>\GlsAddLetterGroup</code>	49	<code>\glsentrydescpluralaccess</code>	323
<code>\glsaddprotectedpagefmt</code>	168	<code>\Glsentryfirst</code>	144
<code>\GlsAddSortRule</code>	47	<code>\glsentryfirst</code>	144
<code>\GlsAddXdyAlphabet</code>	44	<code>\glsentryfirstaccess</code>	322
<code>\GlsAddXdyAttribute</code>	42, 298	<code>\Glsentryfirstplural</code>	144
<code>\GlsAddXdyCounters</code>	41, 298	<code>\glsentryfirstplural</code>	144
<code>\GlsAddXdyLocation</code>	46, 299	<code>\glsentryfirstpluralaccess</code>	322
<code>\GlsAddXdyStyle</code>	47	<code>\glsentryfmt</code>	59, 61, 89
<code>\glsautoprefix</code>	6	<code>\Glsentryfull</code>	147
<code>\glsbackslash</code>	150	<code>\glsentryfull</code>	147, 208, 217, 346
<code>\glsclearpage</code>	40	<code>\Glsentryfullpl</code>	147
<code>\glsclosebrace</code>	150	<code>\glsentryfullpl</code>	147
<code>\glscompositor</code>	35, 45	<code>\glsentryitem</code>	192
<code>\glscounter</code>	16, 57	<code>\Glsentrylong</code>	146
<code>\GlsDeclareNoHyperList</code>	16	<code>\glsentrylong</code>	146
<code>\glsdefaultttype</code>	13	<code>\glsentrylongaccess</code>	323
<code>\glsdefmain</code>	12		

<code>\Glsentrylongpl</code>	146	<code>\Glsfirst</code>	121
<code>\glsentrylongpl</code>	146	<code>\glsfirst</code>	120, 121
<code>\glsentrylongpluralaccess</code>	323	<code>\glsfirstaccessdisplay</code>	324
<code>\Glsentryname</code>	141	<code>\GLSfirstplural</code>	123
<code>\glsentryname</code>	141, 173	<code>\Glsfirstplural</code>	123
<code>\glsentrynumberlist</code>	147, 163	<code>\glsfirstplural</code>	122, 123
<code>\Glsentryplural</code>	143	<code>\glsfirstpluralaccessdisplay</code>	324
<code>\glsentryplural</code>	143	<code>\glsgenacfmt</code>	95, 332
<code>\glsentrypluralaccess</code>	322	<code>\glsgenentryfmt</code>	93, 330
<code>\Glsentryprefix</code>	246	<code>\glsgetgrouplabel</code>	198
<code>\glsentryprefix</code>	245	<code>\glsgetgrouptitle</code>	150, 197
<code>\Glsentryprefixfirst</code>	245	<code>\gls glossarymark</code>	9, 38
<code>\glsentryprefixfirst</code>	245	<code>\gls groupheading</code>	196, 199
<code>\Glsentryprefixfirstplural</code>	246	<code>\gls groupskip</code>	196, 199, 259
<code>\glsentryprefixfirstplural</code>	245	<code>\gls hyperlink</code>	148
<code>\Glsentryprefixplural</code>	246	<code>\gls hypernavsep</code>	256
<code>\glsentryprefixplural</code>	245	<code>\gls hypernumber</code>	36, 200
<code>\glsentryprevcount</code>	83	<code>\gls ifhyper</code>	100
<code>\Glsentryshort</code>	146	<code>\gls ifhyperon</code>	100
<code>\glsentryshort</code>	146	<code>\gls IfListOfAcronyms</code>	15
<code>\glsentryshortaccess</code>	323	<code>\gls ifusedtranslatordict</code>	22
<code>\Glsentryshortpl</code>	146	<code>\gls ignore</code>	150
<code>\glsentryshortpl</code>	146	<code>\gls inlinedescformat</code>	258
<code>\glsentryshortpluralaccess</code>	323	<code>\gls inlinedopostchild</code>	257, 258
<code>\glsentrysort</code>	144	<code>\gls inlineemptydescformat</code>	258
<code>\Glsentrysymbol</code>	143	<code>\gls inlinenameformat</code>	258
<code>\glsentrysymbol</code>	143	<code>\gls inlineparentchildseparator</code>	258
<code>\glsentrysymbolaccess</code>	322	<code>\gls inlinepostchild</code>	258
<code>\Glsentrysymbolplural</code>	144	<code>\gls inlineseparator</code>	258
<code>\glsentrysymbolplural</code>	143	<code>\gls inlinesubdescformat</code>	258
<code>\glsentrysymbolpluralaccess</code>	323	<code>\gls inlinesubnameformat</code>	258
<code>\Glsentrytext</code>	143	<code>\gls inlinesubseparator</code>	258
<code>\glsentrytext</code>	50, 143, 173	<code>\gls keylisttok</code>	207
<code>\glsentrytextaccess</code>	322	<code>\gls labeltok</code>	207
<code>\glsentrytype</code>	144	<code>\gls letentryfield</code>	140
<code>\Glsentryuseri</code>	145	<code>\gls link</code>	88, 101, 113, 148, 198, 200
<code>\glsentryuseri</code>	144	<code>\gls link options</code>	
<code>\Glsentryuserii</code>	145	counter	99, 113, 244
<code>\glsentryuserii</code>	145	format	99, 113, 200
<code>\Glsentryuseriii</code>	145	hyper	99, 101, 113
<code>\glsentryuseriii</code>	145	local	100
<code>\Glsentryuseriv</code>	145	<code>\gls linkcheckfirsthyperhook</code>	102
<code>\glsentryuseriv</code>	145	<code>\gls linkvar</code>	100
<code>\Glsentryuserv</code>	145	<code>\gls listdottedwidth</code>	261
<code>\glsentryuserv</code>	145	<code>\gls localreset</code>	80
<code>\Glsentryuservi</code>	146	<code>\gls localresetall</code>	82
<code>\glsentryuservi</code>	146	<code>\gls localunset</code>	81
<code>\glsexpandfields</code>	66	<code>\gls localunsetall</code>	83
<code>\GLSfirst</code>	121		

<code>\glslongaccessdisplay</code>	325	<code>\glspluralsuffix</code>	31, 60
<code>\glslongaccesskey</code>	353	<code>\glspostdescription</code>	9
<code>\glslongkey</code>	204	<code>\glspostinline</code>	258
<code>\glslongpluralaccessdisplay</code>	325	<code>\glsprestandardsort</code>	10
<code>\glslongpluralaccesskey</code>	353	<code>\glsquote</code>	150
<code>\glslongpluralkey</code>	204	<code>\glsrefentry</code>	192
<code>\glslongtok</code>	207	<code>\glsreset</code>	80
<code>\glsmakefirstuc</code>	252	<code>\glsresetall</code>	82
<code>\glsmcols</code>	274	<code>\glsresetentrylist</code>	190
<code>\glsmoveentry</code>	78	<code>\glsresetsubentrycounter</code>	191
<code>\GLSname</code>	124	<code>\glssee</code>	172
<code>\Glsname</code>	123	<code>\glsseeformat</code>	153, 172
<code>\glsname</code>	123, 124	<code>\glsseeitem</code>	173
<code>\glsnameaccessdisplay</code>	324	<code>\glsseeitemformat</code>	173
<code>\glsnamefont</code>	199, 290	<code>\glsseeelastsep</code>	173
<code>\glsnavhyperlink</code>	254	<code>\glsseeelist</code>	172
<code>\glsnavhypertarget</code>	254	<code>\glsseeesep</code>	173
<code>\glsnavigation</code>	256	<code>\glsSetAlphaCompositor</code>	35
<code>\glsnextpages</code>	191	<code>\glsSetCompositor</code>	35
<code>\glsnoexpandfields</code>	66	<code>\glssetexpandfield</code>	17
<code>\glsnoidxdisplayloc</code>	186	<code>\glssetnoexpandfield</code>	17
<code>\glsnoidxdisplayloclisthandler</code>	186	<code>\glsSetSuffixF</code>	35
		<code>\glsSetSuffixFF</code>	36
<code>\glsnoidxloclist</code>	186	<code>\glssettoctitle</code>	30
<code>\glsnoidxloclisthandler</code>	186	<code>\glssetwidest</code>	295
<code>\glsnoidxnumberlistloophandler</code>	164	<code>\GlsSetXdyCodePage</code>	48
		<code>\GlsSetXdyFirstLetterAfterDigits</code>	151
<code>\glsnoidxstripaccents</code>	19		
<code>\glsnonextpages</code>	190	<code>\GlsSetXdyLanguage</code>	48
<code>\glsnoxindywarning</code>	40	<code>\GlsSetXdyLocationClassOrder</code>	47
<code>\glsnumberformat</code>	36	<code>\GlsSetXdyMinRangeLength</code>	151
<code>\glsnumberlistloop</code>	164	<code>\GlsSetXdyStyles</code>	48
<code>\glsnumbersgroupname</code>	31, 150, 197	<code>\glsshortaccessdisplay</code>	325
<code>\glsnumlistlastsep</code>	148	<code>\glsshortaccesskey</code>	353
<code>\glsnumlistsep</code>	148	<code>\glsshortkey</code>	204
<code>\glsopenbrace</code>	150	<code>\glsshortpluralaccessdisplay</code>	325
<code>\glsorder</code>	24	<code>\glsshortpluralaccesskey</code>	353
<code>\glsorg@endtheglossary</code>	5	<code>\glsshortpluralkey</code>	204
<code>\glsorg@theglossary</code>	5	<code>\glsshorttok</code>	207
<code>\glspagelistwidth</code>	262, 268, 278, 285	<code>\glssortnumberfmt</code>	11
<code>\glspar</code>	34	<code>\glsspace</code>	205
<code>\glspercentchar</code>	151	<code>\glsstepentry</code>	191
<code>\GLSpl</code>	118	<code>\glsstepsubentry</code>	192
<code>\Glspl</code>	117, 251	<code>\glssubentrycounterlabel</code>	192
<code>\glspl</code>	88, 116–118	<code>\glssubentryitem</code>	193
<code>\GLSplural</code>	122	<code>\GLSsymbol</code>	126
<code>\Glsplural</code>	122	<code>\Glsymbol</code>	126
<code>\glsplural</code>	121, 122	<code>\glssymbol</code>	126
<code>\glspluralaccessdisplay</code>	324	<code>\glssymbolaccessdisplay</code>	324

<code>\glssymbolnav</code>	256
<code>\GLSsymbolplural</code>	127
<code>\Glsymbolplural</code>	127
<code>\glssymbolplural</code>	127
<code>\glssymbolpluralaccessdisplay</code>	
.....	325
<code>\glssymbolsgroupname</code>	31, 150, 197
<code>\glstarget</code>	193
<code>\GLStext</code>	120
<code>\Glstext</code>	120
<code>\glstext</code>	120
<code>\glstextaccessdisplay</code>	324
<code>\glstextformat</code>	89
<code>\glstextup</code>	204
<code>\glstildechar</code>	151
<code>\glstreeindent</code>	292–294
<code>\glstreenamefmt</code>	290
<code>\glsunset</code>	81
<code>\glsunsetall</code>	82
<code>\glsupacrpluralsuffix</code>	31
<code>\GlsUseAcrEntryDispStyle</code>	210
<code>\GlsUseAcrStyleDefs</code>	210
<code>\GLSuseri</code>	128
<code>\Glsuseri</code>	128
<code>\glsuseri</code>	128
<code>\GLSuserii</code>	129
<code>\Glsuserii</code>	129
<code>\glsuserii</code>	129
<code>\GLSuseriii</code>	130
<code>\Glsuseriii</code>	130
<code>\glsuseriii</code>	129, 130
<code>\GLSuseriv</code>	131
<code>\Glsuseriv</code>	131
<code>\glsuseriv</code>	130, 131
<code>\GLSuseriv</code>	132
<code>\Glsuseriv</code>	132
<code>\glsuseriv</code>	131, 132
<code>\GLSuserivi</code>	133
<code>\Glsuserivi</code>	132
<code>\glsuserivi</code>	132, 133
<code>\glswrite</code>	161
<code>\glswritedefhook</code>	71
<code>\gMFUnocap</code>	253

H

<code>\hyperbf</code>	202
<code>\hyperemph</code>	202
<code>hyperfirst (option)</code>	23
<code>\hyperit</code>	202

<code>\hyperlink</code>	112
<code>\hypermd</code>	202
<code>\hyperpage</code>	200
<code>hyperref package</code>	171, 174, 200, 244
<code>\hyperrm</code>	201
<code>\hypersc</code>	202
<code>\hypersf</code>	202
<code>\hypersl</code>	202
<code>\hypertarget</code>	113
<code>\hypertt</code>	202
<code>\hyperup</code>	202

I

<code>\if@gls@docloaded</code>	5
<code>\if@gls@isacronymlist</code>	15
<code>\ifglossaryexists</code>	50
<code>\ifglsdescsuppressed</code>	53
<code>\ifglsentryexists</code>	51
<code>\ifglschildhaschildren</code>	52
<code>\ifglschildhasdesc</code>	53
<code>\ifglschildhasfield</code>	54
<code>\ifglschildhaslong</code>	53
<code>\ifglschildhasparent</code>	52
<code>\ifglschildhasprefix</code>	246
<code>\ifglschildhasprefixfirst</code>	246
<code>\ifglschildhasprefixfirstplural</code>	246
<code>\ifglschildhasprefixplural</code>	246
<code>\ifglschildhasshort</code>	53
<code>\ifglschildhasymbol</code>	53
<code>\ifglschildtranslate</code>	21
<code>\ifglsused</code>	51, 80
<code>\ifglsxindy</code>	25
<code>\ifignoredglossary</code>	59
<code>index (option)</code>	28
<code>index (style)</code>	290
<code>indexgroup (style)</code>	291
<code>indexhypergroup (style)</code>	292
<code>indexonlyfirst (option)</code>	23
<code>\indexspace</code>	259, 274, 290
<code>inline (style)</code>	256
<code>\inputencodingname</code>	25
<code>\listfile</code>	165
<code>\listfilename</code>	34
<code>\item</code>	199, 259, 290, 291

L

<code>link text</code>	89
<code>list (style)</code>	259
<code>listdotted (style)</code>	261
<code>listgroup (style)</code>	259

listhypergroup (style) 260
\loadglsentries 13, 88
long (key) 63
long (style) 262
long-sc-short (acrstyle) 211
long-sc-short-desc (acrstyle) ..
..... 213, 340
long-short (acrstyle) 210, 339
long-short-desc (acrstyle) . 212, 340
long-sm-short (acrstyle) 212
long-sm-short-desc (acrstyle) ..
..... 213, 341
long3col (style) 264
long3colborder (style) 264
long3colheader (style) 265
long3colheaderborder (style) .. 265
long4col (style) 265
long4colborder (style) 266
long4colheader (style) 266
long4colheaderborder (style) .. 266
longaccess (key) 320
longborder (style) 263
longheader (style) 263
longheaderborder (style) 263
\longnewglossaryentry 60, 71
longplural (key) 63
longpluralaccess (key) 320
\longprovideglossaryentry ... 72
longragged (style) 268
longragged3col (style) 270
longragged3colborder (style) .. 271
longragged3colheader (style) .. 271
longragged3colheaderborder
(style) 271
longraggedborder (style) 269
longraggedheader (style) 269
longraggedheaderborder (style) 270
longtable (environment)
..... 8, 238, 262–273
longtable package 262, 268

M

\makefirsttuc 250
makeglossaries
..... 24, 25, 34, 48, 56, 160, 177
\makeglossaries 26,
29, 30, 34, 35, 55, 57, 159, 161
\makeglossary 161
makeindex 356

makeindex
. 10, 25, 26, 31, 34–36, 56, 58,
60, 79, 104, 108, 150, 153, 155,
157, 166, 170, 171, 196, 299, 300
delim_n 36
delim_r 36
page_compositor 35
special characters ... 106, 107, 150
makeindex (option) 25
\makenoidxglossaries 21, 161
mcolalmtree (style) 276
mcolalmtreegroup (style) 277
mcolalmtreehypergroup (style) . 277
mcolindex (style) 274
mcolindexgroup (style) 274
mcolindexhypergroup (style) ... 274
mcoltree (style) 275
mcoltreegroup (style) 275
mcoltreehypergroup (style) 275
mcoltreenoname (style) 275
mcoltreenonamegroup (style) ... 276
mcoltreenonamehypergroup
(style) 276
memoir class 167
mfirsttuc package 1, 253
\mfirsttucMakeUppercase 252
\mfu@checkword 253
\MFUclear 253
\MFUocap 253
multicol package 273
multicols (environment) 274

N

name (key) 59
\new@glossaryentry 67
\newacronym 24, 63, 88, 202, 203
\newacronymhook 207
\newacronymstyle 209
\newglossary 16, 56, 58, 157, 160, 187
\newglossaryentry .. 60, 66, 88, 203
\newglossaryentry options
access 321, 322
counter 61
description 24, 59, 60,
63, 66, 72, 124, 142, 203, 230, 319
descriptionaccess 323, 325
descriptionplural 125, 320
descriptionpluralaccess .. 323, 325
first 60, 75,
113, 120, 144, 228, 233, 234, 319

firstaccess	322, 324
firstplural	60, 122, 144, 319
firstpluralaccess	322, 324
format	152
long	95, 146, 320
longaccess	323, 325
longplural	146, 320
longpluralaccess	323, 325
name	59, 60, 63, 66, 72, 123, 141, 173, 319
nonumberlist	62
parent	62, 66
plural	60, 75, 121, 319
pluralaccess	322, 324
prefix	245
prefixfirst	245
prefixfirstplural	245
prefixplural	245
see	8, 61, 160, 162
short	95, 146, 320
shortaccess	323, 325
shortplural	146, 320
shortpluralaccess	323, 325
sort	60, 144, 196
symbol	59, 61, 126, 224, 226, 228, 233, 265, 281, 319, 321
symbolaccess	322, 324
symbolplural	127, 319
symbolpluralaccess	323, 325
text	60, 113, 119, 143, 224, 228, 319
textaccess	322, 324
type	13, 61, 88, 144
user1	128, 144, 320
user2	129, 145
user3	129, 145
user4	130, 145
user5	131, 145
user6	132, 146, 320
\newglossarystyle	199
\newignoredglossary	58
nogroupskip (option)	9
nohypertypes (option)	16
\noist	157, 244, 304
nolist (option)	8
nolong (option)	8
nomain (option)	13
nonumberlist (key)	62
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)	6
numbers (option)	27
O	
\oldacronym	202
order (option)	25
P	
\p@gl@s@hyp@opt	100
package options:	
acronym	13, 14, 30, 174, 203
true	14
acronym	13
acronymlists	15
acronyms	14
automake	26
compatible-2.07	27
compatible-3.07	27
counter	16
counter	16
description	228, 229
description	24
dua	226, 228, 229
dua	24
entrycounter	189, 191
true	9
entrycounter	9
entrycounterwithin	9
footnote	114–119, 224, 226, 228, 230
footnote	24
hyperfirst	
false	114–119
hyperfirst	23
index	28
indexonlyfirst	363
indexonlyfirst	23
makeindex	153, 244
makeindex	25
nogroupskip	9
nohypertypes	16
nolist	238

nolist	8
nolong	238, 262
nolong	8
nomain	12, 13
nomain	13
nonumberlist	7
nonumberlist	7
nopostdot	9
noredefwarn	17
nostyles	8
nosuper	238
nosuper	8
notranslate	22
notree	238
notree	8
nowarn	16
numberedsection	6
numberline	6
numberline	6
numbers	27
order	25
sanitize	20, 59, 141, 142
sanitize	21
sanitizesort	20
savenumberlist	7
savewrites	26, 360, 361
false	157
true	159, 165
savewrites	26
section	6, 38
section	6
seeautonumberlist	8
shotcuts	24
smallcaps	24
smaller	24
sort	
def	10, 11
standard	10
use	10, 11
sort	10
style	7, 238
style	7
subentrycounter	189, 191
subentrycounter	10
symbols	27
toc	5
true	6
toc	5
translate	22
false	22
translate	22
translator	21
ucmark	9
xindy	25, 26, 153, 244
xindy	25
xindygloss	26
xindynoglsnumbers	26
\pagelistname	31
parent (key)	62
\PGLS	249
\PglS	248
\pgls	247
\PGLSpl	250
\PglSpl	248
\pglspl	247
\phantomsection	37–39
plural (key)	60
pluralaccess (key)	319
polyglossia package	22, 32
\printacronyms	14
\PrintChanges	5
\printglossaries	
...	12, 36, 55, 58, 161, 174, 254
\printglossary	36,
37, 56, 58, 161, 174, 176, 187, 254	
\printglossary options	
entrycounter	188
nogroupskip	188
nonumberlist	189
nopostdot	188
numberedsection	188
style	187
subentrycounter	189
title	187
toctitle	187
type	13, 173, 187
\printnoidxglossaries	175
\printnoidxglossary	175, 187
\printnoidxglossary options	
sort	190
\provideglossaryentry	66
\ProvidesGlossariesLang	31
R	
\renewacronymstyle	210
\renewglossarystyle	199
\RequireGlossariesLang	31
\roman	44

S	
\s@glshyp@opt	100
\s@newglossary	56
sanitize (option)	21
sanitizesort (option)	20
savenumberlist (option)	7
savewrites (option)	26
sc-short-long (acrstyle)	212
sc-short-long-desc (acrstyle) ..	213, 341
\scantokens	50
\section	50
section (option)	6
see (key)	61
seeautonumberlist (option)	8
\seename	31
\SetAcronymLists	15
\SetAcronymStyle	14, 236
\setacronymstyle	209
\SetCustomDisplayStyle	236
\SetCustomStyle	237
\SetDefaultAcronymDisplayStyle	221
\SetDefaultAcronymStyle	222
\SetDescriptionAcronymDisplayStyle	226
\SetDescriptionAcronymStyle	228
\SetDescriptionDUAAcronymDisplayStyle	225
\SetDescriptionDUAAcronymStyle	226
\SetDescriptionFootnoteAcronymDisplayStyle	223
\SetDescriptionFootnoteAcronymStyle	224
\SetDUADisplayStyle	234
\SetDUASyle	235
\setentrycounter	198
\SetFootnoteAcronymDisplayStyle	229
\SetFootnoteAcronymStyle	230
\SetGenericNewAcronym	207
\setglossarypreamble	37
\setglossarysection	38
\setglossarystyle	198
\setglossentrycompatibility	195
\SetSmallAcronymDisplayStyle	231
\SetSmallAcronymStyle	233
\setStyleFile	34
\setupglossaries	28
short (key)	63
short-long (acrstyle)	211, 339
short-long-desc (acrstyle)	213, 341
shortaccess (key)	320
shortplural (key)	63
shortpluralaccess (key)	320
shotcuts (option)	24
\showacronymlists	242
\showglocounter	239
\showglodesc	241
\showglodescaccess	354
\showglodescplural	241
\showglodescpluralaccess	354
\showglofirst	239
\showglofirstaccess	353
\showglofirstpl	239
\showglofirstpluralaccess	353
\showgloflag	242
\showgloindex	242
\showglolevel	238
\showgloloclist	242
\showglolong	242
\showglolongaccess	354
\showglolongpluralaccess	354
\showgloname	241
\showglonameaccess	353
\showgloparent	238
\showgloplural	239
\showglopluralaccess	353
\showgloshort	242
\showgloshortaccess	354
\showgloshortpluralaccess	354
\showglosort	241
\showglossaries	243
\showglossarycounter	243
\showglossaryentries	244
\showglossaryin	243
\showglossaryout	243
\showglossarytitle	243
\showglosymbol	241
\showglosymbolaccess	354
\showglosymbolplural	241
\showglosymbolpluralaccess	354
\showglotext	239
\showglotextaccess	353
\showglotype	239
\showglouseri	240
\showglouserii	240

<code>\showglouseriii</code>	240
<code>\showglouseriv</code>	240
<code>\showglouserve</code>	240
<code>\showglouservei</code>	240
<code>sm-short-long (acrstyle)</code>	212
<code>sm-short-long-desc (acrstyle)</code>	214, 342
<code>smallcaps (option)</code>	24
<code>smaller (option)</code>	24
<code>\SmallNewAcronymDef</code>	232, 352
<code>sort (key)</code>	60
<code>sort (option)</code>	10
<code>style (option)</code>	7
<code>subentrycounter (option)</code>	10
<code>\subglossentry</code>	193
<code>\subitem</code>	291
<code>sublistdotted (style)</code>	262
<code>\subsubitem</code>	291
<code>super (style)</code>	278
<code>super3col (style)</code>	279
<code>super3colborder (style)</code>	280
<code>super3colheader (style)</code>	280
<code>super3colheaderborder (style)</code>	281
<code>super4col (style)</code>	281
<code>super4colborder (style)</code>	282
<code>super4colheader (style)</code>	282
<code>super4colheaderborder (style)</code>	282
<code>superborder (style)</code>	278
<code>superheader (style)</code>	279
<code>superheaderborder (style)</code>	279
<code>superragged (style)</code>	285
<code>superragged3col (style)</code>	286
<code>superragged3colborder (style)</code>	287
<code>superragged3colheader (style)</code>	287
<code>superraggedborder (style)</code>	285
<code>superraggedheader (style)</code>	286
<code>superraggedheaderborder (style)</code>	286
<code>superraggedright3colheaderborder (style)</code>	288
<code>supertabular (environment)</code>	8, 238, 277–290
<code>supertabular package</code>	8, 238, 277, 284
<code>symbol (key)</code>	61
<code>symbolaccess (key)</code>	319
<code>\symbolname</code>	31
<code>symbolplural (key)</code>	61
<code>symbolpluralaccess (key)</code>	319
<code>symbols (option)</code>	27

T

<code>text (key)</code>	60
<code>textaccess (key)</code>	319
<code>textcase package</code>	4
<code>\theequation</code>	171
<code>theglossary (environment)</code>	5, 17, 36, 37, 193, 199, 275, 276, 292, 293, 295
<code>\theHequation</code>	171
<code>theindex (environment)</code>	290
<code>toc (option)</code>	5
<code>tracklang package</code>	32, 355
<code>\translate</code>	32
<code>translate (option)</code>	22
<code>translator package</code>	13, 14, 22, 27, 28, 32, 33, 174
<code>tree (style)</code>	292
<code>treegroup (style)</code>	293
<code>treehypergroup (style)</code>	293
<code>treenoname (style)</code>	293
<code>treenonamegroup (style)</code>	294
<code>treenonamehypergroup (style)</code>	294
<code>type (key)</code>	61

U

<code>ucmark (option)</code>	9
<code>user1 (key)</code>	62
<code>user2 (key)</code>	62
<code>user3 (key)</code>	62
<code>user4 (key)</code>	62
<code>user5 (key)</code>	62
<code>user6 (key)</code>	63

W

<code>\warn@nomakeglossaries</code>	159
<code>\warn@noprintglossary</code>	174
<code>\writeist</code>	34, 41, 43, 46, 151, 298, 300

X

<code>\xcapitalisewords</code>	253
<code>\xglsaccsupp</code>	324
<code>xindy</code>	10, 25, 26, 34, 35, 40, 44, 46–49, 79, 111, 151, 153, 166, 170, 177, 196, 244, 299, 300
<code>xindy (option)</code>	25
<code>xindygloss (option)</code>	26
<code>xindynoglsnumbers (option)</code>	26
<code>\xmakefirststuc</code>	252
<code>xspace package</code>	4, 202, 203