

Documented Code For glossaries v4.12

Nicola L.C. Talbot

Dickimaw Books

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

2014-11-22

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 v4.12: L^AT_EX2e Package to Assist Generating Glossaries”.

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

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

INSTALL Installation instructions.

CHANGES Change log.

README Package summary.

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

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

Contents

1 Main Package Code	4
1.1 Package Definition	4
1.2 Package Options	5
1.3 Predefined Text	30
1.4 Xindy	40
1.5 Loops and conditionals	49
1.6 Defining new glossaries	55
1.7 Defining new entries	59
1.8 Resetting and unsetting entry flags	80
1.9 Loading files containing glossary entries	82
1.10 Using glossary entries in the text	82
1.10.1 Links to glossary entries	93
1.10.2 Displaying entry details without adding information to the glossary	134
1.11 Adding an entry to the glossary without generating text	142
1.12 Creating associated files	144
1.13 Writing information to associated files	159
1.14 Glossary Entry Cross-References	165
1.15 Displaying the glossary	167
1.16 Acronyms	196
1.17 Predefined acronym styles	201
1.18 Predefined Glossary Styles	232
1.19 Debugging Commands	232
1.20 Compatibility with version 2.07 and below	238
2 Prefix Support (glossaries-prefix Code)	238
3 Mfirstuc Documented Code	244
4 Mfirstuc-english Documented Code	247
5 Glossary Styles	248
5.1 Glossary hyper-navigation definitions (glossary-hypernav package)	248
5.2 In-line Style (glossary-inline.sty)	250
5.3 List Style (glossary-list.sty)	252
5.4 Glossary Styles using longtable (the glossary-long package)	255
5.5 Glossary Styles using longtable (the glossary-longragged package)	262
5.6 Glossary Styles using multicol (glossary-mcols.sty)	267
5.7 Glossary Styles using supertabular environment (glossary-super package)	271
5.8 Glossary Styles using supertabular environment (glossary-superragged package)	278
5.9 Tree Styles (glossary-tree.sty)	284

6 glossaries-compatible-207	291
7 Accessibility Support (glossaries-accsupp Code)	311
7.1 Defining Replacement Text	312
7.2 Accessing Replacement Text	316
7.3 Displaying the Glossary	331
7.4 Acronyms	332
7.5 Debugging Commands	346
8 Multi-Lingual Support	348
8.1 Polyglossia Captions	348
Glossary	349
Change History	349
Index	373

1 Main Package Code

1.1 Package Definition

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

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries}[2014/11/22 v4.12 (NLCT)]
```

Required packages:

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

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

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

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

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

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

As from v3.0, now loading etoolbox:

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

Check if doc has been loaded.

```

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

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

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

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

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

End of doc stuff.
31 \fi

```

1.2 Package Options

- toc** The toc package option will add the glossaries to the table of contents. This is a boolean key, if the value is omitted it is taken to be true.
- ```
32 \define@boolkey{glossaries.sty}[gls]{toc}[true]{}

numberline The numberline package option adds \numberline to \addcontentsline. Note that this option only has an effect if used in with toc=true.
```
- ```
33 \define@boolkey{glossaries.sty}[gls]{numberline}[true]{}

```

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

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

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

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

Determine whether or not to use numbered sections.

`\@@glossarysecstar`

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

`\@@glossaryseclabel`

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

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

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

`numberedsection`

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

The default glossary style is stored in `\@glossary@default@style`. This is initialised to `list`. (The `list` style is defined in the accompanying package described in [subsection 1.18](#).)

`ssary@default@style`

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

`style` The default glossary style can be changed using the `style` package option. The value can be the name of any defined glossary style. The glossary style is set at the beginning of the document, so you can still use the `style` key to set a style that is defined in another package. This package comes with some predefined styles that are defined in [subsection 1.18](#).

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

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

`\@gls@declareoption`

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

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

`lossaryentrynumbers`

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

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

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

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

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

o@seeautonumberlist

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

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

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

\@gls@loadlong

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

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

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

\@gls@loadsuper

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

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

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

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

\@gls@loadlist

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

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

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

\@gls@loadtree

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

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

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

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

```
92 \@gls@declareoption{nostyles}{%  
93   \renewcommand*{\@gls@loadlong}{}}
```



```

94 \renewcommand*{\@gls@loadsuper}{}%
95 \renewcommand*{\@gls@loadlist}{}%
96 \renewcommand*{\@gls@loadtree}{}%
97 \let\@glossary@default@style\relax
98 }

```

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

```

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

```

`nopostdot` Boolean option to suppress post description dot

```

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

```

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

```

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

```

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

```

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

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

```

`entrycounter` Defines a counter that can be used in the standard glossary styles to number each (main) entry. If true, this will define a counter called `glossaryentry`.

```

114 \define@boolkey{glossaries.sty}[gls]{entrycounter}[true]{}
115 \glsentrycounterfalse

```

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

```

116 \define@key{glossaries.sty}{counterwithin}{%
117 \renewcommand*{\@gls@counterwithin}{#1}%
118 \glsentrycountertrue
119 }

```

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

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

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

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

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

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

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

`@setupsort@standard` Set up the macros for default sorting.
131 `\newcommand*\@gls@setupsort@standard}{%`
Store entry information when it's defined.
132 `\def\do@glo@storeentry{\@glo@storeentry}%`
No count register required for standard sort.
133 `\def\@gls@defsortcount##1{}%`
Sort according to sort key (`\@glo@sort`) if provided otherwise sort according to the entry's name (`\@glo@name`). (First argument glossary type, second argument entry label.)
134 `\def\@gls@defsort##1##2{%`
135 `\ifx\@glo@sort\@glsdefaultsort`
136 `\let\@glo@sort\@glo@name`
137 `\fi`
138 `\let\glsdosanitizesort\@gls@sanitizesort`
139 `\glsprestandardsort{\@glo@sort}{##1}{##2}%`
140 `\expandafter\protected@xdef\csname glo@##2@sort\endcsname{\@glo@sort}%`
141 `}`

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

```
142 \def\@gls@setsort##1{%  
143 }
```

Set standard sort as the default:

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

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

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

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

```
153 \newcommand*\@gls@setupsort@def{%
```

Store entry information when it's defined.

```
154 \def\do@glo@storeentry{\@glo@storeentry}%
```

Defined count register associated with the glossary.

```
155 \def\@gls@defsortcount##1{%  
156 \expandafter\global  
157 \expandafter\newcount\csname glossary@##1@sortcount\endcsname  
158 }%
```

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

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

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

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

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

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

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

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

Defined count register associated with the glossary.

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

Initialise the sort key to empty.

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

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

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

Get the parent, if one exists

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

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

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

Set index information for this entry

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

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

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

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

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

Keep track of the default glossary. This is initialised to the main glossary, but can be changed if for some reason you want to make a secondary glossary the main glossary. This affects any commands that can optionally take a glossary name as an argument (or as the value of the type key in a key-value list). This was mainly done so that `\loadglsentries` can temporarily change `\glsdefaulttype` while it loads a file containing new glossary entries (see [subsection 1.9](#)).

`\glsdefaulttype`

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

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

`\acronymtype`

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

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

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

`acronym` The `acronym` option sets an associated conditional which is used in [subsection 1.16](#) to determine whether or not to define a separate glossary for acronyms.

```
211 \define@boolkey{glossaries.sty}[gls]{acronym}[true]{%
212   \ifglsacronym
213     \renewcommand{\@gls@do@acronymsdef}{%
214       \DeclareAcronymList{acronym}%
215       \newglossary[alg]{acronym}{acr}{acn}{\acronymname}%
216       \renewcommand*{\acronymtype}{acronym}%
217     }%
218   }%
219 }
```

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

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

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

```

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

```

`@gls@do@acronymsdef` Set default value

```

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

```

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

```

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

```

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

```

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

```

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

```

247 \newcommand*\@glsacronymlists{}

```

`@addtoacronymlists`

```

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

```

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

```

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

```

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

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

```

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

```

Internal command requires label and list to be expanded:

```

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

```

Not found

```

268     #4%
269   \else

```

Found

```

270     #3%
271   \fi
272 }

```

`\if@glsisacronymlist` Convenient boolean.

```

273 \newif\if@glsisacronymlist

```

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

```

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

```

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

```

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

```

`acronymlists`

```

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

```

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

`\glscounter`

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

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

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

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

```
287 }
```

`\@gls@nohyperlist`

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

`sDeclareNoHyperList`

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

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

```
291   {%
```

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

```
293   }%
```

```
294   {%
```

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

```
296   }%
```

```
297 }
```

`nohypertypes`

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

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

```
300 }
```

`\GlossariesWarning` Prints a warning message.

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

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

```
303 }
```

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

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

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

```
306 }
```

`nowarn` Define package option to suppress warnings

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

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

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

```
310 }
```



```

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

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

norededefwarn Suppress warning on redefinition of \printglossary
317 \@gls@declareoption{norededefwarn}{%
318   \renewcommand*{\@gls@warnonglossdefined}{}%
319   \renewcommand*{\@gls@warnontheGLOSSdefined}{}%
320 }

```

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

```

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

```

```

\glssetexpandfield \glssetexpandfield{<field>}

```

Sets field to always expand.

```

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

```

```

\glssetnoexpandfield \glssetnoexpandfield{<field>}

```

Sets field to never expand.

```

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

```

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

```

333 \glssetexpandfield{type}

```

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

```

334 \glssetnoexpandfield{desc}

```

gn@descplural@field

```

335 \glssetnoexpandfield{descplural}

```

\@gls@sanitizename

```

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

```

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

```

337 \glssetnoexpandfield{name}

```

@gls@sanitizesymbol

```

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

```

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

```

339 \glssetnoexpandfield{symbol}

```

@symbolplural@field

```

340 \glssetnoexpandfield{symbolplural}

```

Sanitizing stuff:

\@gls@sanitizesort

```

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

```

\@@gls@sanitizesort

```

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

```

@gls@nosanitizesort

```

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

```

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

```

352 \newcommand*{\@gls@noidx@sanitizesort{%
353   \ifdefvoid\@glo@sort
354   }%
355   {%
356     \expandafter\@@gls@noidx@sanitizesort\@glo@sort\gls@end@sanitizesort
357   }%
358 }

```

```

359 \def\@@gls@noidx@sanitizesort#1#2\gls@end@sanitizesort{%
360   \def\@glo@sort{#1#2}%
361   \@onelevel@sanitize\@glo@sort
362 }

```

oidx@nosanitizesort

```

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

```

lsnoidxstripaccents

```

377 \newcommand*\glsnoidxstripaccents{%
378   \let\IeC\@firstofone
379   \let\''\@firstofone
380   \let\''\@firstofone
381   \let\^@\@firstofone
382   \let\""\@firstofone
383   \let\u\@firstofone
384   \let\t\@firstofone
385   \let\d\@firstofone
386   \let\r\@firstofone
387   \let\=\@firstofone
388   \let\.\@firstofone
389   \let\~\@firstofone
390   \let\v\@firstofone
391   \let\H\@firstofone
392   \let\c\@firstofone
393   \let\b\@firstofone
394   \def\AE{AE}%
395   \def\ae{ae}%
396   \def\OE{OE}%
397   \def\oe{oe}%
398   \def\AA{AA}%
399   \def\aa{aa}%
400   \def\L{L}%
401   \def\l{l}%
402   \def\O{O}%
403   \def\o{o}%

```

```

404 \def\SS{SS}%
405 \def\ss{ss}%
406 \def\th{th}%
407 }

```

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

```

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

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

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

```

sanitizesort

```

436 \define@boolkey{glossaries.sty}[glS]{sanitizesort}[true]{%
437   \ifglSSanitizesort
438     \glSsetnoexpandfield{sortvalue}%
439     \renewcommand*{\@glS@noidx@setsanitizesort}{%
440       \glSSanitizesorttrue
441       \glSsetnoexpandfield{sortvalue}%
442     }%
443   \else
444     \glSsetexpandfield{sortvalue}%
445     \renewcommand*{\@glS@noidx@setsanitizesort}{%

```

```

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

```

Default setting:

```

451 \glssanitizesorttrue
452 \glsssetnoexpandfield{sortvalue}%

```

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

```

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

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

```

`sanitize`

```

467 \define@key{glossaries.sty}{sanitize}[description=true,symbol=true,name=true]{%
468   \ifthenelse{\equal{#1}{none}}{%
469     {%
470       \GlossariesWarning{sanitize package option deprecated}%
471       \glsssetexpandfield{name}%
472       \glsssetexpandfield{symbol}%
473       \glsssetexpandfield{symbolplural}%
474       \glsssetexpandfield{desc}%
475       \glsssetexpandfield{descplural}%
476     }%
477   }%
478   \setkeys{gls}{sanitize}{#1}%
479 }%
480 }

```

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

```

481 \newif\ifglstranslate

```

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

\@gls@usetranslator

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

fusedtranslatordict Checks if given translator dictionary has been loaded.

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

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

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

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

```
510 \define@choicekey{glossaries.sty}{translate}[\val\nr]%
511 {true,false,babel}[true]%
512 {%
513   \ifcase\nr\relax
514     \glstranslatetrue
515     \renewcommand*\@gls@usetranslator{%
516       \@ifpackageloaded{polyglossia}%
517       {%
518         \let\glsifusetranslator\@secondoftwo
519       }%
520     }%
521   }
```

```

520      {%
521        \@ifpackageloaded{babel}%
522      {%
523        \IfFileExists{translator.sty}%
524      {%
525        \RequirePackage{translator}%
526        \let\glsifusetranslator\@firstoftwo
527      }%
528    }%
529  }%
530  {%
531  }%
532  }%
533  \or
534    \glstranslatefalse
535    \let\@gls@usetranslator\relax
536    \let\glsifusetranslator\@secondoftwo
537  \or
538    \glstranslatetrue
539    \let\@gls@usetranslator\relax
540    \let\glsifusetranslator\@secondoftwo
541  \fi
542  }

```

Set the default value:

```

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

```

indexonlyfirst Set whether to only index on first use.

```

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

```

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

```

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

```

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

```

565 \newcommand*{\@gls@setacrstyle}{}

```

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

```

566 \define@boolkey{glossaries.sty}[glsacr]{footnote}[true]{%
567   \ifbool{glsacrdescription}%
568   {}%
569   {%
570     \renewcommand*{\@gls@sanitizedesc}{}%
571   }%
572   \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
573 }

```

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

```

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

```

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

```

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

```

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

```

582 \define@boolkey{glossaries.sty}[glsacr]{smaller}[true]{%
583   \renewcommand*{\@gls@sanitizesymbol}{}%
584   \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
585 }

```

`dua` Define `\newacronym` to always use the long forms (i.e. don't use acronyms)

```

586 \define@boolkey{glossaries.sty}[glsacr]{dua}[true]{%
587   \renewcommand*{\@gls@sanitizesymbol}{}%
588   \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
589 }

```

`shortcuts` Define acronym shortcuts.

```

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

```

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

```

591 \newcommand*{\glsorder}{word}

```


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

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

order

```
593 \define@choicekey{glossaries.sty}{order}{word,letter}{%
```

```
594   \def\glsorder{#1}}
```

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

```
595 \newif\ifglxindy
```

The default is makeindex:

```
596 \glxindyfalse
```

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

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

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

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

```
599 \glx@xindy@glsnumberstrue
```

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

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

Define key to set the language

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

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

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

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

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

```
605 }
```

Define a key to set the code page.

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

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

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

```
608   \glxindytrue
```

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

```
610 }
```

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

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

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

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

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

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

\@gls@doautomake

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

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

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

Set default:

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

compatible-3.07

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

compatible-2.07

```
646 \define@boolkey{glossaries.sty}[gls]{compatible-2.07}[true]{%
  Also set 3.07 compatibility if this option is set.
647 \ifbool{glscompatible-2.07}%
648 {%
649 \booltrue{glscompatible-3.07}%
650 }%
651 {}%
652 }
653 \boolfalse{glscompatible-2.07}
```

symbols Create a “symbols” glossary type

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

Default is not to define the symbols glossary:

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

\@gls@symbolsdef

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

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

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

numbers Create a “symbols” glossary type

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

Default is not to define the numbers glossary:

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

\@gls@numbersdef

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

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

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

index Create an “index” glossary type

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

Default is not to define index glossary:

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

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

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

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

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

Now process options passed to the package:

```
698 \ProcessOptionsX
```

Load backward compatibility stuff:

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

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

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

Now define `\setupglossaries`:

```

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

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

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

```

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

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

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

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

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

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

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

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

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

1.3 Predefined Text

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

Main glossary title:

`\glossaryname`

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

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

`\acronymname`

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

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

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

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

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

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

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

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

Labels for makeindex's symbol and number groups:

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

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

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

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

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

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

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

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

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

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

dglossarytocaptions Does nothing if translator hasn't been loaded.

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

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

```
777 \ifglstranslate
```

Load tracklang

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

Load translator if required.

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

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

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

```
781 {%
```

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

```
782 \ifboolexpr
```

```
783 {
```

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

```
785 and not
```

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

```
787 }
```

```
788 {%
```

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

```
790 }%
```

```
791 {%
```

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

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

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

```
795 {%
```

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

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

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

```
799 }%
```

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

```
801 }%
```

```
802 }%
```

```
803 }%
```

```
804 }%
```

```
805 }%
```

Check for tracked languages


```

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

```

if using translator use translator interface.

```

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

```

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

```

850 \DeclareRobustCommand*\nopostdesc{}

```

`\@nopostdesc` Suppress next description terminator.

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

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

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

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

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

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

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

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

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

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

`\istfilename`

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

`\gl@istfilebase`

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

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

`\@istfilename`

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

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

`\glscompositor`

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

`\glsSetCompositor` Sets the compositor.

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

Only use before `\makeglossaries`

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

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

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

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

`\glsSetAlphaCompositor` Sets the alpha compositor.

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

Can only be used before `\makeglossaries`

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

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

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

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

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

Only has an effect when used before `\makeglossaries`

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

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

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

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

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

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

```
895 \ifcsundef{hyperlink}%
896 {%
897   \newcommand*{\glsnumberformat}[1]{#1}%
898 }%
899 {%
900   \newcommand*{\glsnumberformat}[1]{\glshypernumber{#1}}%
901 }
```

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

`\delimN`

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

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

`\delimR`

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

The glossary preamble is given by `\glossarypreamble`. This will appear after the glossary sectioning command, and before the `\theglossary` environment. It is designed to allow the user to add information pertaining to the glossary (e.g. “page numbers in italic indicate the primary definition”) therefore `\glossarypreamble` shouldn't be affected by the glossary style. (So if you define your own glossary style, don't have it change `\glossarypreamble`.) The preamble is empty by default. If you have multiple glossaries, and you want a different preamble for each glossary, you will need to use `\printglossary` for each glossary type, instead of `\printglossaries`, and redefine `\glossarypreamble` before each `\printglossary`.

```

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

```

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

```

```

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

```

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

```

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

926   \glsglossarymark{\glossarytoctitle}%
927 }

```

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

```

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

```

`\glossarymark` Provided for backward compatibility:

```

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

```

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

`\setglossarysection`

```

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

```

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

`\@glossarysection`

```

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

```

Do automatic labelling if required

```

971   \@glossaryseclabel
972 }

```

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

`\@p@glossarysection`

```

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

```

Do automatic labelling if required

```

984   \@glossaryseclabel
985 }

```

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

```

986 \newcommand*{\gls@doclearpage}{%
987   \ifthenelse{\equal{\@glossarysec}{chapter}}{%
988     {%
989       \ifcsundef{cleardoublepage}%
990       {%
991         \clearpage
992       }%
993     }%
994     \ifcsdef{if@openright}%

```

```

995      {%
996      \if@openright
997      \cleardoublepage
998      \else
999      \clearpage
1000      \fi
1001      }%
1002      {%
1003      \cleardoublepage
1004      }%
1005      }%
1006      }%
1007      {}%
1008      }

```

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

```

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

```

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

`\@gls@toc`

```

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

```

1.4 Xindy

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

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

```

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

```

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


```

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

```

\@xdyattributelist Comma-separated list of attributes.

```

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

```

\@xdylocref Define list of markup location references.

```

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

```

\@gls@ifinlist

```

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

```

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

```

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

```

Check if already in list before adding.

```

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

```

Only has an effect before \writeist:

```

1056   \@onlypremakeg\GlsAddXdyCounters

```

```

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

```

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

```

1062 \newcommand*\@disabled@glssaddxdycounters{%
1063   \PackageError{glossaries}{\string\GlsAddXdyCounters\space
1064     can't be used after \string\GlsAddXdyAttribute}{Move all
1065     occurrences of \string\GlsAddXdyCounters\space before the first
1066     instance of \string\GlsAddXdyAttribute}%
1067 }

```

`\GlsAddXdyAttribute` Adds an attribute.

```

1068 \ifglsxindy

```

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

```

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

```

Add to xindy attribute list

```

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

```

Add to xindy markup location.

```

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

```

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

```

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

```

High-level command:

```

1084 \newcommand*\GlsAddXdyAttribute[1]{%

```

Add to comma-separated attribute list

```

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

```

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

```

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

```

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

```

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

```

Only has an effect before `\writeist`:

```

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

```

redefinedattributes Add known attributes for all defined counters

```

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

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

```

`\@xdyuseralphabets` List of additional alphabets

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

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

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

This code is only required for xindy:

```
1142 \ifglsxindy
```

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

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

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

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

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

```

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

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

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

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

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

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

bic-section-numbers Section number style locations (e.g. 1.1, 1.2, ...). The compositor is given by
\glscompositor.
1186 \expandafter\def\csname @gls@xdy@Lclass@arabic-section-numbers\endcsname{%
1187     \string"arabic-numbers\string"
1188     :sep \string"\glscompositor\string"
1189     \string"arabic-numbers\string"%
1190 }%

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

dyuserlocationnames List of additional user location names
1192 \def\@xdyuserlocationnames{}

```

End of xindy-only block:

1193 \fi

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

```
1194 \ifglxindy
1195   \newcommand*{\GlsAddXdyLocation}[3][]{%
1196     \def\@gls@tmp{#1}%
1197     \ifx\@gls@tmp\@empty
1198       \edef\@xdyuserlocationdefs{%
1199         \@xdyuserlocationdefs ^^J%
1200         (define-location-class \string"#2\string"^^J\space\space
1201         \space(:sep \string"{}\glssopenbrace\string" #3
1202           :sep \string"\glsclosebrace\string"))
1203       }%
1204     \else
1205       \edef\@xdyuserlocationdefs{%
1206         \@xdyuserlocationdefs ^^J%
1207         (define-location-class \string"#2\string"^^J\space\space
1208         \space(:sep "\glssopenbrace"
1209           #1
1210           :sep "\glsclosebrace\glssopenbrace" #3
1211           :sep "\glsclosebrace"))
1212       }%
1213     \fi
1214     \edef\@xdyuserlocationnames{%
1215       \@xdyuserlocationnames^^J\space\space\space
1216       \string"#1\string"}%
1217   }
```

Only has an effect before \writeist:

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

ylocationclassorder Define location class order

```
1223 \ifglxindy
1224   \edef\@xdylocationclassorder{^^J\space\space\space
1225     \string"roman-page-numbers\string"^^J\space\space\space
1226     \string"arabic-page-numbers\string"^^J\space\space\space
1227     \string"arabic-section-numbers\string"^^J\space\space\space
1228     \string"alpha-page-numbers\string"^^J\space\space\space
1229     \string"Roman-page-numbers\string"^^J\space\space\space
1230     \string"Alpha-page-numbers\string"^^J\space\space\space
1231     \string"Appendix-page-numbers\string"
```

```

1232    \@xdyuserlocationnames^^J\space\space\space
1233    \string"see\string"
1234  }
1235 \fi

```

Change the location order.

yLocationClassOrder

```

1236 \ifglxindy
1237   \newcommand*\GlsSetXdyLocationClassOrder[1]{%
1238     \def\@xdylocationclassorder{#1}}
1239 \else
1240   \newcommand*\GlsSetXdyLocationClassOrder[1]{%
1241     \glsnnoxindywarning\GlsSetXdyLocationClassOrder}
1242 \fi

```

\@xdysortrules Define sort rules

```

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

```

\GlsAddSortRule Add a sort rule

```

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

```

\@xdyrequiredstyles Define list of required styles (this should be a comma-separated list of xindy styles)

```

1256 \ifglxindy
1257   \def\@xdyrequiredstyles{tex}
1258 \fi

```

\GlsAddXdyStyle Add a xindy style to the list of required styles

```

1259 \ifglxindy
1260   \newcommand*\GlsAddXdyStyle[1]{%
1261     \edef\@xdyrequiredstyles{\@xdyrequiredstyles,#1}}%
1262 \else
1263   \newcommand*\GlsAddXdyStyle[1]{%
1264     \glsnnoxindywarning\GlsAddXdyStyle}
1265 \fi

```

`\GlsSetXdyStyles` Reset the list of required styles

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

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

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

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

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

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

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

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

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

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

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



```

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

```

`\@xdylettergroups` Store letter group definitions.

```

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

```

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

```

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

```

1.5 Loops and conditionals

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

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

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

```

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

```

`\forallacronyms`

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

`\glsdetoklabel`

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```

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

```

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

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

```

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

```

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

```

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

```

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

```

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

```

```

1389 }%
1390 }

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

ifglsdessuppressed \ifglsdessuppressed{<label>}{<true part>}{<false part>} Does <true part>
if the description is just \nopostdesc otherwise does <false part>.
1396 \newcommand*{\ifglsdessuppressed}[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}{gls@detoklabel{#2}@#1}%

```

First check supplied field label is defined.

```

1435 \ifdef\@glo@thisvalue
1436 {%

```

Is defined, so now check if empty.

```

1437 \ifdefempty\@glo@thisvalue
1438 {%

```

Is empty, so doesn't have field set.

```

1439 #4%
1440 }%
1441 {%

```

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

```

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

```

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

```

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

```

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

```

1447 \ifdef\@gls@thisfield
1448 {%

```

Is defined, so now check if empty.

```

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

```

Is empty so field hasn't been set.

```

1452 #4%
1453 }%
1454 {%

```

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

```

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

```

Not defined.

```

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

```

1.6 Defining new glossaries

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

\@glo@types

```

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

```

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

```

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

```

Only need to do this once.

```

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

```

\defglsentryfmt Allow different glossaries to have different display styles.

```

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

```

\gls@doentryfmt

```

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

```

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

```

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

```

```

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

```

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

```

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

```

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

`\newglossary`

```

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

```

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

```

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

```

`\ns@newglossary` Define the unstarred version.

```

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

```

Check if default has been set

```

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

```


Add this to the list of glossary types:

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

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

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

Store the file extensions:

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

Store the title:

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

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

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

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

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

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

Define sort counter if required:

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

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

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

\altnewglossary

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

Only define new glossaries in the preamble:

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

Only define new glossaries before \makeglossaries

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

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

`\@newglossary`

```
1528 \newcommand*{\@newglossary}[4]{}

```

Store counter to be used for given glossary type (the first argument is the glossary label, the second argument is the name of the counter):

`\@gls@setcounter`

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

```

Add counter to xindy list, if not already added:

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

```

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

`\@gls@getcounter`

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

```

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

```
1538 \glsdefmain

```

Define the “acronym” glossaries if required.

```
1539 \@gls@do@acronymsdef

```

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

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

```

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

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

```

```

1550 }%
1551 \csgdef{glolist@#1}{,}%
1552 \ifcsundef{gls@#1@entryfmt}%
1553 {%
1554   \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 *<name>* *<description>*.

```
1580 \define@key{glossentry}{sort}{%
1581 \def\@glo@sort{#1}}
```

text The text key determines how the term should appear when used in the document (i.e. outside of the glossary). If omitted, the value of the name key is used instead.

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

plural The plural key determines how the plural form of the term should be displayed in the document. If omitted, the plural is constructed by appending `\glspluralsuffix` to the value of the text key.

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

first The first key determines how the entry should be displayed in the document when it is first used. If omitted, it is taken to be the same as the value of the text key.

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

firstplural The firstplural key is used to set the plural form for first use, in the event that the plural is required the first time the term is used. If omitted, it is constructed by appending `\glspluralsuffix` to the value of the first key.

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

`\@gls@default@value`

```
1594 \newcommand*{\@gls@default@value}{\relax}
```

symbol The symbol key is ignored by most of the predefined glossary styles, and defaults to `\relax` if omitted. It is provided for glossary styles that require an associated symbol, as well as a name and description. To make this value appear in the glossary, you need to redefine `\glossentry`. If you want this value to appear in the text when the term is used by commands like `\gls`, you will need to change `\glsentryfmt` (or use for `\defglsentryfmt` individual glossaries).

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

symbolplural

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

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

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

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

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

see The see key specifies a list of cross-references

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

\gls@checkseeallowed

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

```

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

```

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

```

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

```

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

```

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

```

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

user1

```

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

```

user2

```

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

```

user3

```

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

```

user4

```

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

```

user5

```

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

```

user6

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

short This key is provided for use by \newacronym. It's not designed for general purpose use, so isn't described in the user manual.

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

shortplural This key is provided for use by \newacronym.

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

long This key is provided for use by \newacronym.

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

longplural This key is provided for use by \newacronym.

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

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

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

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

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

\@glsdefaultplural Now obsolete. Don't use.

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

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

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

\@glsdefaultsort Define command to set default sort.

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

\gls@level Register to increment entry levels.

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

@gls@noexpand@field

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

gls@noexpand@fields

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



```

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

```

\@@gls@expand@field

```

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

```

@gls@expand@fields

```

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

```

startswitexpandonce

```

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

```

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

```

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

```

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

```

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

```

Write key value information:

```

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

```

Provide hook:

```

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

```

1834 }

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

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

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

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

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

Ensure user field name is fully expanded

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

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

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

Found it.

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

Break out of loop.

```
1867     \@endfortrue
```

```

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

```

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

Allow user to add their own custom keys.

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

Starred version switches on expansion for this key.

```

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

```

Unstarred version doesn't override default expansion.

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

Check the specified key doesn't already exist.

```

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

```

Set up the key.

```

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

```

Set the default value.

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

Assignment code.

```

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

```

Define the no-link commands.

```

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

```

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

```

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

```

```

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

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

```

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

```

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

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

```

Finally the all caps version:

```

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

```

```

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

```

\gls.writedefhook

```

1958 \newcommand*\gls.writedefhook{}

```

\gls@assign@desc

```

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

```

\longnewglossaryentry

```

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

```

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

```

1980 \@onlypreamble{\longnewglossaryentry}

```

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

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

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

Defines a new entry without checking if it already exists.

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

Store label

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

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

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

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

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

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

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

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

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

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

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

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

Using \let instead of \def to make later comparison avoid expansion issues.

(Thanks to Ulrich Diez for suggesting this.)

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

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

Set the default sort:

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

Set the default counter:

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

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

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

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



```

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

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

```

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

```
2014 \@newglossaryentryprehook
```

Extract key-val information from third parameter:

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

Check there is a default glossary.

```

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

```

Assign type. This must be fully expandable

```

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

```

Check to see if this glossary type has been defined, if it has, add this label to the relevant list, otherwise generate an error.

```

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

```

Check if it's an ignored glossary

```

2034 \ifignoredglossary\@glo@type
2035 {%

```

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

```

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

```

```

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

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

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

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

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

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

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

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

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

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

```

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

```

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

```

Set the level for this entry

```

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

```

Define commands associated with this entry:

```

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

```

If first has been specified, make the default by appending \glspluralsuffix, otherwise make the default the value of the plural key.

```

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

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

```

```

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

```

Set default numberlist if not defined:

```

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

```

The smaller and smallcaps options set the description to \@glo@first. Need to check for this, otherwise it won't get expanded if the description gets sanitized.

```

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

```

Set the sort key for this entry:

```

2143 \@gls@defsort{\@glo@type}{\@glo@label}%
2144 \def\@glo@@symbol{\@glo@text}%
2145 \ifx\@glo@symbol\@glo@@symbol
2146 \let\@glo@symbol\@glo@text
2147 \fi
2148 \gls@assign@field{\relax}{\@glo@label}{symbol}{\@glo@symbol}%
2149 \expandafter
2150 \gls@assign@field\expandafter
2151 {\csname glo@\@glo@label @symbol\endcsname}
2152 {\@glo@label}{symbolplural}{\@glo@symbolplural}%

```

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

```

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

```

Sort out any cross-referencing if required.

```

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

```

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

```

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

```

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

```

2180 \@newglossaryentryposthook
2181 }

```

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

```

2182 \newcommand*\@newglossaryentryprehook{}

```

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

```

2183 \newcommand*\@newglossaryentryposthook{}

```

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

```

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

```

```

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

```

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

```

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

```

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

```

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

```

\@glo@storeentry `\@glo@storeentry{<label>}`

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

```

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

```

Escape makeindex/xindy special characters in the label:

```

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

```

Get the sort string and escape any special characters

```

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

```

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

```

2213   \@gls@checkmkidxchars\@glo@prefix

```

Get the parent, if one exists

```

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

Write the information to the glossary file.

2215 \ifglxindy

Store using xindy syntax.

2216 \ifx\@glo@parent\@empty

Entry doesn't have a parent

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

Entry has a parent

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

Store using makeindex syntax.

2230 \ifx\@glo@parent\@empty

Sanitize \@glo@prefix

2231 \@onelevel@sanitize\@glo@prefix

Entry doesn't have a parent

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

Entry has a parent

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

```

1.8 Resetting and unsetting entry flags

Each glossary entry is assigned a conditional of the form `\ifglo@<label>@flag` which determines whether or not the entry has been used (see also `\ifglsused` defined below). These flags can be set and unset using the following macros, but first we need to know if we're in `amsmath`'s `align` environment's measuring pass.

`\gls@ifnotmeasuring`

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

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

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

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

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

`\glsunset` The command `\glsunset{<label>}` can be used to set the entry flag to indicate that it has been used. The required argument is the entry label.

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



```

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

```

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

```

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

```

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

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

`\glsresetall`

```

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

```

As above, but with only a local effect:

`\glslocalresetall`

```

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

```

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

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

`\glsunsetall`

```

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

```

As above, but with only a local effect:

```
\glslocalunsetall
```

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

1.9 Loading files containing glossary entries

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

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

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

```
\loadglsentries
```

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

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

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

1.10 Using glossary entries in the text

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

```
\glsformat
```

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

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

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

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

Format that provides backwards compatibility:

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

Plural form

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

Don't adjust case

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

Subsequent use

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

First use

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

Make first letter upper case

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

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

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

```

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

```

First use

```

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

```

Make all upper case

```

2381     \ifglsused\glslabel
2382     {%

```

Subsequent use

```

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

```

First use

```

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

```

Singular form

```

2395     \glscapscase
2396     {%

```

Don't adjust case

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

Subsequent use

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

First use

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

Make first letter upper case

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

Subsequent use

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

First use

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

```

2437         {\glentrydesc{\glslabel}}%
2438         {\glentrysymbol{\glslabel}}{\glinsert}%
2439     }%
2440 }%
2441 }%
2442 {%

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

        Subsequent use
2445         \mfirstucMakeUppercase{#2{\glentrytext{\glslabel}}%
2446         {\glentrydesc{\glslabel}}%
2447         {\glentrysymbol{\glslabel}}{\glinsert}}%
2448     }%
2449     {%

        First use
2450         \mfirstucMakeUppercase{#1{\glentryfirst{\glslabel}}%
2451         {\glentrydesc{\glslabel}}%
2452         {\glentrysymbol{\glslabel}}{\glinsert}}%
2453     }%
2454 }%
2455 }%
2456 }%
2457 {%

    Custom text provided in \glldisp
2458     \ifglused{\glslabel}%
2459     {%

        Subsequent use
2460         #2{\glscustomtext}%
2461         {\glentrydesc{\glslabel}}%
2462         {\glentrysymbol{\glslabel}}{%
2463     }%
2464     {%

        First use
2465         #1{\glscustomtext}%
2466         {\glentrydesc{\glslabel}}%
2467         {\glentrysymbol{\glslabel}}{%
2468     }%
2469     }%
2470 }

```

`\glsgenentryfmt` Define a generic format that just uses the first, text, plural or first plural keys (or the custom text) with the insert text appended.

```

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

```

```

2473  {%
2474      \glsifplural
2475  {%

    Plural form

2476      \glscapscase
2477  {%

    Don't adjust case

2478      \ifglsused\glslabel
2479  {%

    Subsequent use

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

    First use

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

    Make first letter upper case

2487      \ifglsused\glslabel
2488  {%

    Subsequent use.

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

    First use

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

    Make all upper case

2496      \ifglsused\glslabel
2497  {%

    Subsequent use

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

    First use

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

```

2506 }%
2507 {%

Singular form

2508 \glscapscase
2509 {%

Don't adjust case

2510 \ifglused\glslabel
2511 {%

Subsequent use

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

First use

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

Make first letter upper case

2519 \ifglused\glslabel
2520 {%

Subsequent use

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

First use

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

Make all upper case

2528 \ifglused\glslabel
2529 {%

Subsequent use

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

First use

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

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

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

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

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

Subsequent use:

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

Subsequent plural form:

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

Subsequent plural form, don't adjust case:

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

Subsequent plural form, make first letter upper case:

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

Subsequent plural form, all caps:

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

Subsequent singular form

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

Subsequent singular form, don't adjust case:

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

Subsequent singular form, make first letter upper case:

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

Subsequent singular form, all caps:

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

First use:

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

First use plural form:

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

First use plural form, don't adjust case:

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

First use plural form, make first letter upper case:

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

First use plural form, all caps:

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

First use singular form

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

First use singular form, don't adjust case:

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

First use singular form, make first letter upper case:

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

First use singular form, all caps:

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

```

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

```

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

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

```

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

```

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

As above but makes the first letter upper case.

```

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

```

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

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

```

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

```

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

As above but makes the first letter upper case.

```

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

```

`\glsdisplayfirst` Deprecated. Kept for backward compatibility.

```

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

```

`\glsdisplay` Deprecated. Kept for backward compatibility.

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

`\defglsdisplay` Deprecated. Kept for backward compatibility.

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

`\defglsdisplayfirst` Deprecated. Kept for backward compatibility.

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

1.10.1 Links to glossary entries

The links to glossary entries all have a first optional argument that can be used to change the format and counter of the associated entry number. Except for `\glslink` and `\glsdisp`, the commands like `\gls` have a final optional argument that can be used to insert additional text in the link (this will usually be appended, but can be redefined using `\defentryfmt`). It goes against the \TeX norm to have an optional argument after the mandatory arguments, but it makes more sense to write, say, `\gls{label}[s]` rather than, say, `\gls[append=s]{label}`. Since these control sequences are defined to include the final square bracket, spaces will be ignored after them. This is likely to lead to confusion as most users would not expect, say, `\gls{<label>}` to ignore following spaces, so `\new@ifnextchar` from the package is required.

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

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

The value of the format key should be the name of a command (without the initial backslash) that has a single mandatory argument which can be used to format the associated entry number.

```
2683 \define@key{glslink}{format}{%
2684   \def\@glsnumberformat{#1}}
```

The hyper key is a boolean key, it can either have the value true or false, and indicates whether or not to make a hyperlink to the relevant glossary entry. If hyper is false, an entry will still be made in the glossary, but the given text won't be a hyperlink.

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

Initialise hyper key.

```
2686 \ifdef{\hyperlink}{\KV@glslink@hypertrue}{\KV@glslink@hyperfalse}
```

The local key is a boolean key. If true this indicates that commands such as `\gls` should only do a local reset rather than a global one.

```
2687 \define@boolkey{glslink}{local}[true]{}
```

The original `\glsifhyper` command isn't particularly useful as it makes more sense to check the actual hyperlink setting rather than testing whether the starred or unstarred version has been used. Therefore, as from version 4.08, `\glsifhyper` is deprecated in favour of `\glsifhyperon`. In case there is a particular need to know whether the starred or unstarred version was used, provide a new command that determines whether the *-version, +-version or unmodified version was used.

```
\glslinkvar{<unmodified case>}{<star case>}{<plus case>}
```

`\glslinkvar` Initialise to unmodified case.

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

`\glsifhyper` Now deprecated.

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

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

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

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

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

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

```
2703 \newcommand*{\p@gls@hyp@opt}[1] []{%
2704 \let\glslinkvar\@thirdofthree
2705 \@gls@hyp@opt@cs[hyper=true,#1]}
```

Syntax:

```
\glslink[<options>]{<label>}{<text>}
```

Display `<text>` in the document, and add the entry information for `<label>` into the relevant glossary. The optional argument should be a key value list using the `\glslink` keys defined above.

There is also a starred version:

```
\glslink*[\<options>]{\<label>}{\<text>}
```

which is equivalent to `\glslink[hyper=false,\<options>]{\<label>}{\<text>}`

First determine which version is being used:

`\glslink`

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

`\@gls@link` The main part of the business is in `\@gls@link` which shouldn't check if the term is defined as it's called by `\gls` etc which also perform that check.

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

```

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

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

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

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

```

```

2735     \fi
2736   }%

   Allow user to hook into this
2737   \glslinkcheckfirsthyperhook
2738 }

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

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

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

   Macros must set this before calling \@gls@link. The commands that check
   the first use flag should set this to \@gls@link@checkfirsthyper otherwise it
   should be set to \relax.
2756   \do@gls@link@checkfirsthyper
2757   \setkeys{glslink}{#1}%

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

```


Store the entry's counter in \theglsentrycounter

```

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

```

\glolinkprefix

```

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

```

\glsentrycounter Set default value of entry counter

```

2774 \def\glsentrycounter{\glscounter}%

```

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

```

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

```

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

```

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

```

Check if hyperref version of this counter

```

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

```

Not using equation counter so no special measures:

```

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

```

Check if hyperref version of this counter

```

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

```

`\@set@glo@numformat` Set the formatting information in the format required by `makeindex`. The first argument is the format specified by the user (via the `format` key), the second argument is the name of the counter used to indicate the location, the third argument is a control sequence which stores the required format and the fourth argument (new to v3.0) is the hyper-prefix.

```

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

```

Check to see if the given string starts with a (or). If it does set `\@glo@prefix` to the starting character, and `\@glo@suffix` to the rest (or `glsnumberformat` if

there is nothing else), otherwise set \@glo@prefix to nothing and \@glo@suffix to all of it.

```

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

```

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

```

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

```

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

```

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

```

2868 }%

Assign to required control sequence

2869 \let#1=\gls@xdyststring

2870 }

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

\gls@checkmkidxchars

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

Update the control sequence and strip trailing \@nil:

\@gls@updatechecked

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

\@gls@tmpb Define temporary token

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

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

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

```

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

```

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

```

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

```

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

```

2947 \def\@gls@checkescactual#1\?#2\?#3\null{%
2948 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%

```

```

2949 \toks@={#1}%
2950 \ifx\null#2\null
2951   \ifx\null#3\null
2952     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
2953     \def\@@gls@checkescactual{\relax}%
2954   \else
2955     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
2956       \@gls@quotearchar\string"\@gls@actualchar
2957       \@gls@quotearchar\string"\@gls@actualchar}%
2958     \def\@@gls@checkescactual{\@gls@checkescactual#3\null}%
2959   \fi
2960 \else
2961   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
2962     \@gls@quotearchar\string"\@gls@actualchar}%
2963   \ifx\null#3\null
2964     \def\@@gls@checkescactual{\@gls@checkescactual#2\?\?\null}%
2965   \else
2966     \def\@@gls@checkescactual{\@gls@checkescactual#2\?#3\null}%
2967   \fi
2968 \fi
2969 \@@gls@checkescactual
2970 }

```

\@gls@checkescbar Similarly for \|:

```

2971 \def\@gls@checkescbar#1\|#2\|#3\null{%
2972   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
2973   \toks@={#1}%
2974   \ifx\null#2\null
2975     \ifx\null#3\null
2976       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
2977       \def\@@gls@checkescbar{\relax}%
2978     \else
2979       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
2980         \@gls@quotearchar\string"\@gls@encapchar
2981         \@gls@quotearchar\string"\@gls@encapchar}%
2982       \def\@@gls@checkescbar{\@gls@checkescbar#3\null}%
2983     \fi
2984   \else
2985     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
2986       \@gls@quotearchar\string"\@gls@encapchar}%
2987     \ifx\null#3\null
2988       \def\@@gls@checkescbar{\@gls@checkescbar#2\|\|\null}%
2989     \else
2990       \def\@@gls@checkescbar{\@gls@checkescbar#2\|#3\null}%
2991     \fi
2992   \fi
2993 \@@gls@checkescbar
2994 }

```

\@gls@checkesclevel Similarly for \!:

```
2995 \def\@gls@checkesclevel#1\!#2\!#3\null{%
2996   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
2997   \toks@={#1}%
2998   \ifx\null#2\null
2999     \ifx\null#3\null
3000       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3001       \def\@gls@checkesclevel{\relax}%
3002     \else
3003       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3004         \@gls@quotechar\string"\@gls@levelchar
3005         \@gls@quotechar\string"\@gls@levelchar}%
3006       \def\@gls@checkesclevel{\@gls@checkesclevel#3\null}%
3007     \fi
3008   \else
3009     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3010       \@gls@quotechar\string"\@gls@levelchar}%
3011     \ifx\null#3\null
3012       \def\@gls@checkesclevel{\@gls@checkesclevel#2\!\!\null}%
3013     \else
3014       \def\@gls@checkesclevel{\@gls@checkesclevel#2\!#3\null}%
3015     \fi
3016   \fi
3017 \@gls@checkesclevel
3018 }
```

\@gls@checkbar and for |:

```
3019 \def\@gls@checkbar#1|#2|#3\null{%
3020   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3021   \toks@={#1}%
3022   \ifx\null#2\null
3023     \ifx\null#3\null
3024       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3025       \def\@gls@checkbar{\relax}%
3026     \else
3027       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3028         \@gls@quotechar\@gls@encapchar\@gls@quotechar\@gls@encapchar}%
3029       \def\@gls@checkbar{\@gls@checkbar#3\null}%
3030     \fi
3031   \else
3032     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3033       \@gls@quotechar\@gls@encapchar}%
3034     \ifx\null#3\null
3035       \def\@gls@checkbar{\@gls@checkbar#2||\null}%
3036     \else
3037       \def\@gls@checkbar{\@gls@checkbar#2|#3\null}%
3038     \fi
3039   \fi
3040 \@gls@checkbar
```

3041 }

\@gls@checklevel and for !:

```
3042 \def\@gls@checklevel#1!#2!#3\null{%
3043   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3044   \toks@={#1}%
3045   \ifx\null#2\null
3046     \ifx\null#3\null
3047       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3048       \def\@gls@checklevel{\relax}%
3049     \else
3050       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3051         \@gls@quotechar\@gls@levelchar\@gls@quotechar\@gls@levelchar}%
3052       \def\@gls@checklevel{\@gls@checklevel#3\null}%
3053     \fi
3054   \else
3055     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3056       \@gls@quotechar\@gls@levelchar}%
3057     \ifx\null#3\null
3058       \def\@gls@checklevel{\@gls@checklevel#2!\null}%
3059     \else
3060       \def\@gls@checklevel{\@gls@checklevel#2!#3\null}%
3061     \fi
3062   \fi
3063   \@gls@checklevel
3064 }
```

\@gls@checkactual and for ?:

```
3065 \def\@gls@checkactual#1?#2?#3\null{%
3066   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3067   \toks@={#1}%
3068   \ifx\null#2\null
3069     \ifx\null#3\null
3070       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3071       \def\@gls@checkactual{\relax}%
3072     \else
3073       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3074         \@gls@quotechar\@gls@actualchar\@gls@quotechar\@gls@actualchar}%
3075       \def\@gls@checkactual{\@gls@checkactual#3\null}%
3076     \fi
3077   \else
3078     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3079       \@gls@quotechar\@gls@actualchar}%
3080     \ifx\null#3\null
3081       \def\@gls@checkactual{\@gls@checkactual#2??\null}%
3082     \else
3083       \def\@gls@checkactual{\@gls@checkactual#2?#3\null}%
3084     \fi
3085   \fi
```



```

3086 \@@gls@checkactual
3087 }

```

\@gls@xdycheckquote As before but for use with xindy

```

3088 \def\@gls@xdycheckquote#1"#2"#3\null{%
3089 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3090 \toks@={#1}%
3091 \ifx\null#2\null
3092 \ifx\null#3\null
3093 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3094 \def\@@gls@xdycheckquote{\relax}%
3095 \else
3096 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3097 \string\string\}%
3098 \def\@@gls@xdycheckquote{\@gls@xdycheckquote#3\null}%
3099 \fi
3100 \else
3101 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3102 \string\}%
3103 \ifx\null#3\null
3104 \def\@@gls@xdycheckquote{\@gls@xdycheckquote#2""\null}%
3105 \else
3106 \def\@@gls@xdycheckquote{\@gls@xdycheckquote#2"#3\null}%
3107 \fi
3108 \fi
3109 \@gls@xdycheckquote
3110 }

```

s@xdycheckbackslash Need to escape all backslashes for xindy. Define command that will define
\@gls@xdycheckbackslash

```

3111 \edef\def\@gls@xdycheckbackslash{%
3112 \noexpand\def\noexpand\@gls@xdycheckbackslash##1\@backslashchar
3113 ##2\@backslashchar##3\noexpand\null{%
3114 \noexpand\@gls@tmpb=\noexpand\expandafter
3115 {\noexpand\@gls@checkedmkidx}%
3116 \noexpand\toks@={##1}%
3117 \noexpand\ifx\noexpand\null##2\noexpand\null
3118 \noexpand\ifx\noexpand\null##3\noexpand\null
3119 \noexpand\edef\noexpand\@gls@checkedmkidx{%
3120 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
3121 \noexpand\def\noexpand\@@gls@xdycheckbackslash{\relax}%
3122 \noexpand\else
3123 \noexpand\edef\noexpand\@gls@checkedmkidx{%
3124 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
3125 \@backslashchar\@backslashchar\@backslashchar\@backslashchar}%
3126 \noexpand\def\noexpand\@@gls@xdycheckbackslash{%
3127 \noexpand\@gls@xdycheckbackslash##3\noexpand\null}%
3128 \noexpand\fi
3129 \noexpand\else

```

```

3130 \noexpand\edef\noexpand\@gls@checkedmkidx{%
3131 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
3132 \@backslashchar\@backslashchar}%
3133 \noexpand\ifx\noexpand\null##3\noexpand\null
3134 \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3135 \noexpand\@gls@xdycheckbackslash##2\@backslashchar
3136 \@backslashchar\noexpand\null}%
3137 \noexpand\else
3138 \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3139 \noexpand\@gls@xdycheckbackslash##2\@backslashchar
3140 ##3\noexpand\null}%
3141 \noexpand\fi
3142 \noexpand\fi
3143 \noexpand\@gls@xdycheckbackslash
3144 }%
3145 }

```

Now go ahead and define \@gls@xdycheckbackslash

```

3146 \def@gls@xdycheckbackslash

```

\glsdohypertarget

```

3147 \newlength@gls@tmplen
3148 \newcommand*\glsdohypertarget}[2]{%
3149 \settoheight{\gls@tmplen}{#2}%
3150 \raisebox{\gls@tmplen}{\hypertarget{#1}}{#2}%
3151 }

```

\glsdohyperlink

```

3152 \newcommand*\glsdohyperlink}[2]{\hyperlink{#1}{#2}}

```

\@glslink If \hyperlink is not defined \@glslink ignores its first argument and just does the second argument, otherwise it is equivalent to \hyperlink.

```

3153 \ifcsundef{hyperlink}%
3154 {%
3155 \let\@glslink\@secondoftwo
3156 }%
3157 {%
3158 \let\@glslink\glsdohyperlink
3159 }

```

\@glstarget If \hypertarget is not defined, \@glstarget ignores its first argument and just does the second argument, otherwise it is equivalent to \hypertarget.

```

3160 \ifcsundef{hypertarget}%
3161 {%
3162 \let\@glstarget\@secondoftwo
3163 }%
3164 {%
3165 \let\@glstarget\glsdohypertarget
3166 }

```

Glossary hyperlinks can be disabled using `\glsdisablehyper` (effect can be localised):

`\glsdisablehyper`

```
3167 \newcommand{\glsdisablehyper}{%
3168   \KV@glslink@hyperfalse
3169   \let\@glslink\@secondoftwo
3170   \let\@glstarget\@secondoftwo
3171 }
```

Glossary hyperlinks can be enabled using `\glsenablehyper` (effect can be localised):

`\glsenablehyper`

```
3172 \newcommand{\glsenablehyper}{%
3173   \KV@glslink@hypertrue
3174   \let\@glslink\glsdohyperlink
3175   \let\@glstarget\glsdohypertarget
3176 }
```

Provide some convenience commands if not already defined:

```
3177 \providecommand{\@firstofthree}[3]{#1}
3178 \providecommand{\@secondofthree}[3]{#2}
```

Syntax:

`\gls[<options>]{<label>}[<insert text>]`

Link to glossary entry using singular form. The link text is taken from the value of the text or first keys used when the entry was defined.

The first optional argument is a key-value list, the same as `\glslink`, the mandatory argument is the entry label. After the mandatory argument, there is another optional argument to insert extra text in the link text (the location of the inserted text is governed by `\glsdisplay` and `\glsdisplayfirst`). As with `\glslink` there is a starred version which is the same as the unstarred version but with the `hyper` key set to `false`. (Additional options can also be specified in the first optional argument.)

First determine which version is being used:

`\gls`

```
3179 \newrobustcmd*{\gls}{\@gls@hyp@opt\@gls}
```

Defined the un-starred form. Need to determine if there is a final optional argument

`\@gls`

```
3180 \newcommand*{\@gls}[2][ ]{%
3181   \new@ifnextchar[\@gls@{#1}{#2}]{\@gls@{#1}{#2}[ ]}%
3182 }
```

`\@gls@` Read in the final optional argument:

```
3183 \def\@gls@#1#2[#3]{%
3184   \glsdoifexists{#2}%
3185   {%
3186     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3187     \let\glsifplural\@secondoftwo
3188     \let\glscapscase\@firstofthree
3189     \let\glscustomtext\@empty
3190     \def\glsinsert{#3}%
```

Determine what the link text should be (this is stored in `\@glo@text`) Note that `\@gls@link` sets `\glstyle`.

```
3191   \def\@glo@text{\csname gls@\glstyle @entryfmt\endcsname}%
```

Call `\@gls@link`. If footnote package option has been used and the glossary type is `\acronymtype`, suppress hyperlink for first use. Likewise if the `hyperfirst=false` package option is used.

```
3192   \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3193   \ifKV@glslink@local
3194     \glslocalunset{#2}%
3195   \else
3196     \glsunset{#2}%
3197   \fi
3198 }%
3199 }
```

`\Gls` behaves like `\gls`, but the first letter of the link text is converted to uppercase (note that if the first letter has an accent, the accented letter will need to be grouped when you define the entry). It is mainly intended for terms that start a sentence:

`\Gls`

```
3200 \newrobustcmd*{\Gls}{\@gls@hyp@opt\@Gls}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3201 \newcommand*{\@Gls}[2] [] {%
3202   \new@ifnextchar[{\@Gls@{#1}{#2}}{\@Gls@{#1}{#2} []}%
3203 }
```

`\@Gls@` Read in the final optional argument:

```
3204 \def\@Gls@#1#2[#3]{%
3205   \glsdoifexists{#2}%
3206   {%
3207     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
```

```

3208 \let\glsifplural\@secondoftwo
3209 \let\glscapscase\@secondofthree
3210 \let\glscustomtext\@empty
3211 \def\glsinsert{#3}%

```

Determine what the link text should be (this is stored in \glo@text) Note that \gls@link sets \glstype.

```

3212 \def\glo@text{\csname gls@\glstype @entryfmt\endcsname}%

```

Call \gls@link If footnote package option has been used and the glossary type is \acronymtype, suppress hyperlink for first use. Likewise if the hyperfirst=false package option is used.

```

3213 \gls@link[#1]{#2}{\glo@text}%

```

Indicate that this entry has now been used

```

3214 \ifKV@glslink@local
3215 \glslocalunset{#2}%
3216 \else
3217 \glsunset{#2}%
3218 \fi
3219 }%
3220 }

```

\GLS behaves like \gls, but the link text is converted to uppercase:

\GLS

```

3221 \newrobustcmd*{\GLS}{\@gls@hyp@opt\@GLS}

```

Defined the un-starred form. Need to determine if there is a final optional argument

```

3222 \newcommand*{\@GLS}[2][ ]{%
3223 \new@ifnextchar[{\@GLS@{#1}{#2}}{\@GLS@{#1}{#2}[ ]}%
3224 }

```

\@GLS@ Read in the final optional argument:

```

3225 \def\@GLS@#1#2[#3]{%
3226 \glsdoifexists{#2}%
3227 {%
3228 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3229 \let\glsifplural\@secondoftwo
3230 \let\glscapscase\@thirdofthree
3231 \let\glscustomtext\@empty
3232 \def\glsinsert{#3}%

```

Determine what the link text should be (this is stored in \glo@text). Note that \gls@link sets \glstype.

```

3233 \def\glo@text{\csname gls@\glstype @entryfmt\endcsname}%

```

Call `\@gls@link` If footnote package option has been used and the glossary type is `\acronymtype`, suppress hyperlink for first use. Likewise if the `hyperfirst=false` package option is used.

```
3234 \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3235 \ifKV@glslink@local
3236 \glslocalunset{#2}%
3237 \else
3238 \glsunset{#2}%
3239 \fi
3240 }%
3241 }
```

`\glspl` behaves in the same way as `\gls` except it uses the plural form.

`\glspl`

```
3242 \newrobustcmd*{\glspl}{\@gls@hyp@opt\@glspl}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3243 \newcommand*{\@glspl}[2][{}]{%
3244 \new@ifnextchar[{\@glspl@{#1}{#2}}{\@glspl@{#1}{#2}[]}%
3245 }
```

`\@glspl@` Read in the final optional argument:

```
3246 \def\@glspl@#1#2[#3]{%
3247 \glsdoifexists{#2}%
3248 {%
3249 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3250 \let\glsifplural\@firstoftwo
3251 \let\glsupcase\@firstofthree
3252 \let\glsustomtext\@empty
3253 \def\glsinsert{#3}%
```

Determine what the link text should be (this is stored in `\@glo@text`) Note that `\@gls@link` sets `\glstyle`.

```
3254 \def\@glo@text{\csname gls@\glstyle @entryfmt\endcsname}%
```

Call `\@gls@link`. If footnote package option has been used and the glossary type is `\acronymtype`, suppress hyperlink for first use. Likewise if the `hyperfirst=false` package option is used.

```
3255 \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3256 \ifKV@glslink@local
3257 \glslocalunset{#2}%
3258 \else
3259 \glsunset{#2}%
```

```

3260   \fi
3261 }%
3262 }

```

`\Glspl` behaves in the same way as `\glsp1`, except that the first letter of the link text is converted to uppercase (as with `\Gls`, if the first letter has an accent, it will need to be grouped).

`\Glspl`

```

3263 \newrobustcmd*{\Glspl}{\@gls@hyp@opt\@Glspl}

```

Defined the un-starred form. Need to determine if there is a final optional argument

```

3264 \newcommand*{\@Glspl}[2][\@Glspl@#1]{%
3265   \new@ifnextchar[\@Glspl@#1]{#2}}{\@Glspl@#1}{#2}[]}%
3266 }

```

`\@Glspl@` Read in the final optional argument:

```

3267 \def\@Glspl@#1#2[#3]{%
3268   \glsdoifexists{#2}%
3269   {%
3270     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3271     \let\glsifplural\@firstoftwo
3272     \let\glsupcase\@secondofthree
3273     \let\glscustomtext\@empty
3274     \def\glsinsert{#3}%

```

Determine what the link text should be (this is stored in `\@glo@text`). This needs to be expanded so that the `\@glo@text` can be passed to `\xmakefirstuc`. Note that `\@gls@link` sets `\glstype`.

```

3275   \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%

```

Call `\@gls@link`. If footnote package option has been used and the glossary type is `\acronymtype`, suppress hyperlink for first use. Likewise if the `hyperfirst=false` package option is used.

```

3276   \@gls@link[#1]{#2}{\@glo@text}%

```

Indicate that this entry has now been used

```

3277   \ifKV@glslink@local
3278     \glslocalunset{#2}%
3279   \else
3280     \glsunset{#2}%
3281   \fi
3282 }%
3283 }

```

`\GLSp1` behaves like `\glsp1` except that all the link text is converted to uppercase.

\GLSp1

```
3284 \newrobustcmd*{\GLSp1}{\@gls@hyp@opt\GLSp1}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3285 \newcommand*{\GLSp1}[2] [] {%
```

```
3286   \new@ifnextchar[{\GLSp1@{#1}{#2}}{\GLSp1@{#1}{#2} []}%
```

```
3287 }
```

\@GLSp1 Read in the final optional argument:

```
3288 \def\GLSp1@#1#2[#3] {%
```

```
3289   \glsdoifexists{#2}%
```

```
3290   {%
```

```
3291     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
```

```
3292     \let\glsifplural\@firstoftwo
```

```
3293     \let\glscapscase\@thirdofthree
```

```
3294     \let\glscustomtext\@empty
```

```
3295     \def\glsinsert{#3}%
```

Determine what the link text should be (this is stored in \@glo@text) Note that \@gls@link sets \glstype.

```
3296   \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%
```

Call \@gls@link. If footnote package option has been used and the glossary type is \acronymtype, suppress hyperlink for first use. Likewise if the hyperfirst=false package option is used.

```
3297   \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3298   \ifKV@glslink@local
```

```
3299     \glslocalunset{#2}%
```

```
3300   \else
```

```
3301     \glsunset{#2}%
```

```
3302   \fi
```

```
3303   }%
```

```
3304 }
```

\glsdisp \glsdisp[<options>]{<label>}{<text>} This is like \gls except that the link text is provided. This differs from \glslink in that it uses \glsdisplay or \glsdisplayfirst and unsets the first use flag.

First determine if we are using the starred form:

```
3305 \newrobustcmd*{\glsdisp}{\@gls@hyp@opt\glsdisp}
```

Defined the un-starred form.

\@glsdisp

```
3306 \newcommand*{\@glsdisp}[3] [] {%
```

```
3307   \glsdoifexists{#2}{%
```



```

3308 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3309 \let\glsifplural\@secondoftwo
3310 \let\glscapscase\@firstofthree
3311 \def\glscustomtext{#3}%
3312 \def\glsinsert{}%

```

Determine what the link text should be (this is stored in \@glo@text) Note that \@gls@link sets \glstype.

```

3313 \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%

```

Call \@gls@link. If footnote package option has been used and the glossary type is \acronymtype, suppress hyperlink for first use. Likewise if the hyperfirst=false package option is used.

```

3314 \@gls@link[#1]{#2}{\@glo@text}%

```

Indicate that this entry has now been used

```

3315 \ifKV@glslink@local
3316 \glslocalunset{#2}%
3317 \else
3318 \glsunset{#2}%
3319 \fi
3320 }%
3321 }

```

\@gls@field@link

```

3322 \newcommand{\@gls@field@link}[3]{%
3323 \glsdoifexists{#2}%
3324 {%
3325 \let\do@gls@link@checkfirsthyper\relax
3326 \@gls@link[#1]{#2}{#3}%
3327 }%
3328 }

```

\glstext behaves like \gls except it always uses the value given by the text key and it doesn't mark the entry as used.

\glstext

```

3329 \newrobustcmd*{\glstext}{\@gls@hyp@opt\@glstext}

```

Defined the un-starred form. Need to determine if there is a final optional argument

```

3330 \newcommand*{\@glstext}[2][ ]{%
3331 \new@ifnextchar[{\@glstext@{#1}{#2}}{\@glstext@{#1}{#2}[]}}

```

Read in the final optional argument:

```

3332 \def\@glstext@#1#2[#3]{%
3333 \@gls@field@link{#1}{#2}{\glstext{#2}#3}%
3334 }

```

\GLSText behaves like \glstext except the text is converted to uppercase.

`\GLStext`

```
3335 \newrobustcmd*{\GLStext}{\@gls@hyp@opt\GLStext}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3336 \newcommand*{\@GLStext}[2] [] {%
```

```
3337   \new@ifnextchar[{\@GLStext@{#1}{#2}}{\@GLStext@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3338 \def\@GLStext@#1#2[#3] {%
```

```
3339   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glstentrytext{#2}#3}}%
```

```
3340 }
```

`\Glstext` behaves like `\glstext` except that the first letter of the text is converted to uppercase.

`\Glstext`

```
3341 \newrobustcmd*{\Glstext}{\@gls@hyp@opt\Glstext}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3342 \newcommand*{\@Glstext}[2] [] {%
```

```
3343   \new@ifnextchar[{\@Glstext@{#1}{#2}}{\@Glstext@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3344 \def\@Glstext@#1#2[#3] {%
```

```
3345   \@gls@field@link{#1}{#2}{\Glstentrytext{#2}#3}%
```

```
3346 }
```

`\glsfirst` behaves like `\gls` except it always uses the value given by the first key and it doesn't mark the entry as used.

`\glsfirst`

```
3347 \newrobustcmd*{\glsfirst}{\@gls@hyp@opt\glsfirst}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3348 \newcommand*{\@glsfirst}[2] [] {%
```

```
3349   \new@ifnextchar[{\@glsfirst@{#1}{#2}}{\@glsfirst@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3350 \def\@glsfirst@#1#2[#3] {%
```

```
3351   \@gls@field@link{#1}{#2}{\glstentryfirst{#2}#3}%
```

```
3352 }
```

`\Glsfirst` behaves like `\glsfirst` except it displays the first letter in uppercase.

`\Glsfirst`

```
3353 \newrobustcmd*{\Glsfirst}{\@gls@hyp@opt\Glsfirst}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3354 \newcommand*{\@Glsfirst}[2] [] {%
3355   \new@ifnextchar[{\@Glsfirst@{#1}{#2}}{\@Glsfirst@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3356 \def\@Glsfirst@#1#2[#3] {%
3357   \@gls@field@link{#1}{#2}{\Glsentryfirst{#2}#3}%
3358 }
```

\GLSfirst behaves like \Glsfirst except it displays the text in uppercase.

\GLSfirst

```
3359 \newrobustcmd*{\GLSfirst}{\@gls@hyp@opt\@GLSfirst}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3360 \newcommand*{\@GLSfirst}[2] [] {%
3361   \new@ifnextchar[{\@GLSfirst@{#1}{#2}}{\@GLSfirst@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3362 \def\@GLSfirst@#1#2[#3] {%
3363   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentryfirst{#2}#3}}%
3364 }
```

\glsplural behaves like \gls except it always uses the value given by the plural key and it doesn't mark the entry as used.

\glsplural

```
3365 \newrobustcmd*{\glsplural}{\@gls@hyp@opt\@glsplural}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3366 \newcommand*{\@glsplural}[2] [] {%
3367   \new@ifnextchar[{\@glsplural@{#1}{#2}}{\@glsplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3368 \def\@glsplural@#1#2[#3] {%
3369   \@gls@field@link{#1}{#2}{\Glsentryplural{#2}#3}%
3370 }
```

\Glsplural behaves like \glsplural except that the first letter is converted to uppercase.

\Glsplural

```
3371 \newrobustcmd*{\Glsplural}{\@gls@hyp@opt\@Glsplural}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3372 \newcommand*{\@Glsplural}[2] [] {%
3373   \new@ifnextchar[{\@Glsplural@{#1}{#2}}{\@Glsplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3374 \def\@Glsplural@#1#2[#3]{%
3375   \@gls@field@link{#1}{#2}{\Glsentryplural{#2}#3}%
3376 }
```

\Glsplural behaves like \glsplural except that the text is converted to uppercase.

\Glsplural

```
3377 \newrobustcmd*{\Glsplural}{\@gls@hyp@opt\@Glsplural}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3378 \newcommand*{\@Glsplural}[2][\%
3379   \new@ifnextchar[{\@Glsplural@{#1}{#2}}{\@Glsplural@{#1}{#2}[]}]
```

Read in the final optional argument:

```
3380 \def\@Glsplural@#1#2[#3]{%
3381   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentryplural{#2}#3}}%
3382 }
```

\glsfirstplural behaves like \gls except it always uses the value given by the firstplural key and it doesn't mark the entry as used.

\glsfirstplural

```
3383 \newrobustcmd*{\glsfirstplural}{\@gls@hyp@opt\@glsfirstplural}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3384 \newcommand*{\@glsfirstplural}[2][\%
3385   \new@ifnextchar[{\@glsfirstplural@{#1}{#2}}{\@glsfirstplural@{#1}{#2}[]}]
```

Read in the final optional argument:

```
3386 \def\@glsfirstplural@#1#2[#3]{%
3387   \@gls@field@link{#1}{#2}{\Glsentryfirstplural{#2}#3}%
3388 }
```

\Glsfirstplural behaves like \glsfirstplural except that the first letter is converted to uppercase.

\Glsfirstplural

```
3389 \newrobustcmd*{\Glsfirstplural}{\@gls@hyp@opt\@Glsfirstplural}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3390 \newcommand*{\@Glsfirstplural}[2][\%
3391   \new@ifnextchar[{\@Glsfirstplural@{#1}{#2}}{\@Glsfirstplural@{#1}{#2}[]}]
```

Read in the final optional argument:

```
3392 \def\@Glsfirstplural@#1#2[#3]{%
3393   \@gls@field@link{#1}{#2}{\Glsentryfirstplural{#2}#3}%
3394 }
```

`\GLSfirstplural` behaves like `\glsfirstplural` except that the link text is converted to uppercase.

`\GLSfirstplural`

```
3395 \newrobustcmd*{\GLSfirstplural}{\@gls@hyp@opt\@GLSfirstplural}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3396 \newcommand*{\@GLSfirstplural}[2][\@GLSfirstplural@{#1}{#2}[]]{%
```

```
3397 \new@ifnextchar[\@GLSfirstplural@{#1}{#2}]{\@GLSfirstplural@{#1}{#2}[]}}
```

Read in the final optional argument:

```
3398 \def\@GLSfirstplural@#1#2[#3]{%
```

```
3399 \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryfirstplural{#2}{#3}}}%
```

```
3400 }
```

`\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`

```
3401 \newrobustcmd*{\glsname}{\@gls@hyp@opt\@glsname}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3402 \newcommand*{\@glsname}[2][\@glsname@{#1}{#2}[]]{%
```

```
3403 \new@ifnextchar[\@glsname@{#1}{#2}]{\@glsname@{#1}{#2}[]}}
```

Read in the final optional argument:

```
3404 \def\@glsname@#1#2[#3]{%
```

```
3405 \@gls@field@link{#1}{#2}{\glsentryname{#2}{#3}}%
```

```
3406 }
```

`\Glsname` behaves like `\glsname` except that the first letter is converted to uppercase.

`\Glsname`

```
3407 \newrobustcmd*{\Glsname}{\@gls@hyp@opt\@Glsname}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3408 \newcommand*{\@Glsname}[2][\@Glsname@{#1}{#2}[]]{%
```

```
3409 \new@ifnextchar[\@Glsname@{#1}{#2}]{\@Glsname@{#1}{#2}[]}}
```

Read in the final optional argument:

```
3410 \def\@Glsname@#1#2[#3]{%
```

```
3411 \@gls@field@link{#1}{#2}{\Glsentryname{#2}{#3}}%
```

```
3412 }
```

`\GLSname` behaves like `\glsname` except that the link text is converted to uppercase.

`\GLSname`

```
3413 \newrobustcmd*{\GLSname}{\@gls@hyp@opt\@GLSname}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3414 \newcommand*{\@GLSname}[2] [] {%
3415   \new@ifnextchar[{\@GLSname@{#1}{#2}}{\@GLSname@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3416 \def\@GLSname@#1#2[#3] {%
3417   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryname{#2}#3}}%
3418 }
```

\glsdesc behaves like \gls except it always uses the value given by the description key and it doesn't mark the entry as used.

\glsdesc

```
3419 \newrobustcmd*{\glsdesc}{\@gls@hyp@opt\@glsdesc}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3420 \newcommand*{\@glsdesc}[2] [] {%
3421   \new@ifnextchar[{\@glsdesc@{#1}{#2}}{\@glsdesc@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3422 \def\@glsdesc@#1#2[#3] {%
3423   \@gls@field@link{#1}{#2}{\glsentrydesc{#2}#3}%
3424 }
```

\Glsdesc behaves like \glsdesc except that the first letter is converted to uppercase.

\Glsdesc

```
3425 \newrobustcmd*{\Glsdesc}{\@gls@hyp@opt\@Glsdesc}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3426 \newcommand*{\@Glsdesc}[2] [] {%
3427   \new@ifnextchar[{\@Glsdesc@{#1}{#2}}{\@Glsdesc@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3428 \def\@Glsdesc@#1#2[#3] {%
3429   \@gls@field@link{#1}{#2}{\Glsentrydesc{#2}#3}%
3430 }
```

\GLSdesc behaves like \glsdesc except that the link text is converted to uppercase.

\GLSdesc

```
3431 \newrobustcmd*{\GLSdesc}{\@gls@hyp@opt\@GLSdesc}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3432 \newcommand*{\@GLSdesc}[2] [] {%
3433   \new@ifnextchar[{\@GLSdesc@{#1}{#2}}{\@GLSdesc@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3434 \def\@GLSdesc@#1#2[#3]{%
3435   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrydesc{#2}#3}}%
3436 }
```

`\glsdescplural` behaves like `\gls` except it always uses the value given by the descriptionplural key and it doesn't mark the entry as used.

`\glsdescplural`

```
3437 \newrobustcmd*{\glsdescplural}{\@gls@hyp@opt\@glsdescplural}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3438 \newcommand*{\@glsdescplural}[2][\@glsdescplural@{#1}{#2}[]]{%
3439   \new@ifnextchar[{\@glsdescplural@{#1}{#2}}{\@glsdescplural@{#1}{#2}[]}]}
```

Read in the final optional argument:

```
3440 \def\@glsdescplural@#1#2[#3]{%
3441   \@gls@field@link{#1}{#2}{\glsentrydescplural{#2}#3}%
3442 }
```

`\Glsdescplural` behaves like `\glsdescplural` except that the first letter is converted to uppercase.

`\Glsdescplural`

```
3443 \newrobustcmd*{\Glsdescplural}{\@gls@hyp@opt\@Glsdescplural}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3444 \newcommand*{\@Glsdescplural}[2][\@Glsdescplural@{#1}{#2}[]]{%
3445   \new@ifnextchar[{\@Glsdescplural@{#1}{#2}}{\@Glsdescplural@{#1}{#2}[]}]}
```

Read in the final optional argument:

```
3446 \def\@Glsdescplural@#1#2[#3]{%
3447   \@gls@field@link{#1}{#2}{\Glsentrydescplural{#2}#3}%
3448 }
```

`\GLSdescplural` behaves like `\glsdescplural` except that the link text is converted to uppercase.

`\GLSdescplural`

```
3449 \newrobustcmd*{\GLSdescplural}{\@gls@hyp@opt\@GLSdescplural}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3450 \newcommand*{\@GLSdescplural}[2][\@GLSdescplural@{#1}{#2}[]]{%
3451   \new@ifnextchar[{\@GLSdescplural@{#1}{#2}}{\@GLSdescplural@{#1}{#2}[]}]}
```

Read in the final optional argument:

```
3452 \def\@GLSdescplural@#1#2[#3]{%
3453   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrydescplural{#2}#3}}%
3454 }
```

`\glssymbol` behaves like `\gls` except it always uses the value given by the symbol key and it doesn't mark the entry as used.

`\glssymbol`

```
3455 \newrobustcmd*{\glssymbol}{\@gls@hyp@opt\@glssymbol}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3456 \newcommand*{\@glssymbol}[2] [] {%
```

```
3457   \new@ifnextchar[{\@glssymbol@{#1}{#2}}{\@glssymbol@{#1}{#2} []}]
```

Read in the final optional argument:

```
3458 \def\@glssymbol@#1#2[#3] {%
```

```
3459   \@gls@field@link{#1}{#2}{\glstentrysymbol{#2}#3}%
```

```
3460 }
```

`\Glsymbol` behaves like `\glssymbol` except that the first letter is converted to uppercase.

`\Glsymbol`

```
3461 \newrobustcmd*{\Glsymbol}{\@gls@hyp@opt\@Glsymbol}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3462 \newcommand*{\@Glsymbol}[2] [] {%
```

```
3463   \new@ifnextchar[{\@Glsymbol@{#1}{#2}}{\@Glsymbol@{#1}{#2} []}]
```

Read in the final optional argument:

```
3464 \def\@Glsymbol@#1#2[#3] {%
```

```
3465   \@gls@field@link{#1}{#2}{\Glstentrysymbol{#2}#3}%
```

```
3466 }
```

`\GLSsymbol` behaves like `\glssymbol` except that the link text is converted to uppercase.

`\GLSsymbol`

```
3467 \newrobustcmd*{\GLSsymbol}{\@gls@hyp@opt\@GLSsymbol}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3468 \newcommand*{\@GLSsymbol}[2] [] {%
```

```
3469   \new@ifnextchar[{\@GLSsymbol@{#1}{#2}}{\@GLSsymbol@{#1}{#2} []}]
```

Read in the final optional argument:

```
3470 \def\@GLSsymbol@#1#2[#3] {%
```

```
3471   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glstentrysymbol{#2}#3}}%
```

```
3472 }
```

`\glssymbolplural` behaves like `\gls` except it always uses the value given by the `symbolplural` key and it doesn't mark the entry as used.

`\glssymbolplural`

```
3473 \newrobustcmd*{\glssymbolplural}{\@gls@hyp@opt\@glssymbolplural}
```


Define the un-starred form. Need to determine if there is a final optional argument

```
3474 \newcommand*{\@glssymbolplural}[2] [] {%
3475   \new@ifnextchar[{\@glssymbolplural@{#1}{#2}}{\@glssymbolplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3476 \def\@glssymbolplural@#1#2[#3] {%
3477   \@gls@field@link{#1}{#2}{\glsentrysymbolplural{#2}#3}%
3478 }
```

\Glsymbolplural behaves like \glssymbolplural except that the first letter is converted to uppercase.

\Glsymbolplural

```
3479 \newrobustcmd*{\Glsymbolplural}{\@gls@hyp@opt\@Glsymbolplural}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3480 \newcommand*{\@GLssymbolplural}[2] [] {%
3481   \new@ifnextchar[{\@GLssymbolplural@{#1}{#2}}{\@GLssymbolplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3482 \def\@GLssymbolplural@#1#2[#3] {%
3483   \@gls@field@link{#1}{#2}{\Glsentrysymbolplural{#2}#3}%
3484 }
```

\GLSsymbolplural behaves like \glssymbolplural except that the link text is converted to uppercase.

\GLSsymbolplural

```
3485 \newrobustcmd*{\GLSsymbolplural}{\@gls@hyp@opt\@GLSsymbolplural}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3486 \newcommand*{\@GLSsymbolplural}[2] [] {%
3487   \new@ifnextchar[{\@GLSsymbolplural@{#1}{#2}}{\@GLSsymbolplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3488 \def\@GLSsymbolplural@#1#2[#3] {%
3489   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrysymbolplural{#2}#3}}%
3490 }
```

\glsuseri behaves like \gls except it always uses the value given by the user1 key and it doesn't mark the entry as used.

\glsuseri

```
3491 \newrobustcmd*{\glsuseri}{\@gls@hyp@opt\@glsuseri}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3492 \newcommand*{\@glsuseri}[2] [] {%
3493   \new@ifnextchar[{\@glsuseri@{#1}{#2}}{\@glsuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3494 \def\@glsuseri@#1#2[#3]{%
3495   \@gls@field@link{#1}{#2}{\glsentryuseri{#2}#3}%
3496 }
```

\Glsuseri behaves like \glsuseri except that the first letter is converted to uppercase.

\Glsuseri

```
3497 \newrobustcmd*{\Glsuseri}{\@gls@hyp@opt\@Glsuseri}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3498 \newcommand*{\@Glsuseri}[2][]{%
3499   \new@ifnextchar[{\@Glsuseri@{#1}{#2}}{\@Glsuseri@{#1}{#2}[]}]}
```

Read in the final optional argument:

```
3500 \def\@Glsuseri@#1#2[#3]{%
3501   \@gls@field@link{#1}{#2}{\glsentryuseri{#2}#3}%
3502 }
```

\GLSuseri behaves like \glsuseri except that the link text is converted to uppercase.

\GLSuseri

```
3503 \newrobustcmd*{\GLSuseri}{\@gls@hyp@opt\@GLSuseri}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3504 \newcommand*{\@GLSuseri}[2][]{%
3505   \new@ifnextchar[{\@GLSuseri@{#1}{#2}}{\@GLSuseri@{#1}{#2}[]}]}
```

Read in the final optional argument:

```
3506 \def\@GLSuseri@#1#2[#3]{%
3507   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseri{#2}#3}}%
3508 }
```

\glsuserii behaves like \gls except it always uses the value given by the user2 key and it doesn't mark the entry as used.

\glsuserii

```
3509 \newrobustcmd*{\glsuserii}{\@gls@hyp@opt\@glsuserii}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3510 \newcommand*{\@glsuserii}[2][]{%
3511   \new@ifnextchar[{\@glsuserii@{#1}{#2}}{\@glsuserii@{#1}{#2}[]}]}
```

Read in the final optional argument:

```
3512 \def\@glsuserii@#1#2[#3]{%
3513   \@gls@field@link{#1}{#2}{\glsentryuserii{#2}#3}%
3514 }
```

\Glsuserii behaves like \glsuserii except that the first letter is converted to uppercase.

\Glsuserii

```
3515 \newrobustcmd*{\Glsuserii}{\@gls@hyp@opt\@Glsuserii}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3516 \newcommand*{\@Glsuserii}[2] [] {%
```

```
3517   \new@ifnextchar[{\@Glsuserii@{#1}{#2}}{\@Glsuserii@{#1}{#2} []}]
```

Read in the final optional argument:

```
3518 \def\@Glsuserii@#1#2[#3] {%
```

```
3519   \@gls@field@link{#1}{#2}{\Glsentryuserii{#2}#3}%
```

```
3520 }
```

\GLSuserii behaves like \glsuserii except that the link text is converted to uppercase.

\GLSuserii

```
3521 \newrobustcmd*{\GLSuserii}{\@gls@hyp@opt\@GLSuserii}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3522 \newcommand*{\@GLSuserii}[2] [] {%
```

```
3523   \new@ifnextchar[{\@GLSuserii@{#1}{#2}}{\@GLSuserii@{#1}{#2} []}]
```

Read in the final optional argument:

```
3524 \def\@GLSuserii@#1#2[#3] {%
```

```
3525   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentryuserii{#2}#3}}%
```

```
3526 }
```

\glsuseriii behaves like \gls except it always uses the value given by the user3 key and it doesn't mark the entry as used.

\glsuseriii

```
3527 \newrobustcmd*{\glsuseriii}{\@gls@hyp@opt\@glsuseriii}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3528 \newcommand*{\@glsuseriii}[2] [] {%
```

```
3529   \new@ifnextchar[{\@glsuseriii@{#1}{#2}}{\@glsuseriii@{#1}{#2} []}]
```

Read in the final optional argument:

```
3530 \def\@glsuseriii@#1#2[#3] {%
```

```
3531   \@gls@field@link{#1}{#2}{\Glsentryuseriii{#2}#3}%
```

```
3532 }
```

\Glsuseriii behaves like \glsuseriii except that the first letter is converted to uppercase.

\Glsuseriii

```
3533 \newrobustcmd*{\Glsuseriii}{\@gls@hyp@opt\@Glsuseriii}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3534 \newcommand*{\@Glsuseriii}[2] [] {%
3535   \new@ifnextchar[{\@Glsuseriii@{#1}{#2}}{\@Glsuseriii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3536 \def\@Glsuseriii@#1#2[#3] {%
3537   \@gls@field@link{#1}{#2}{\Glsentryuseriii{#2}#3}%
3538 }
```

\Glsuseriii behaves like \glsuseriii except that the link text is converted to uppercase.

\Glsuseriii

```
3539 \newrobustcmd*{\Glsuseriii}{\@gls@hyp@opt\@Glsuseriii}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3540 \newcommand*{\@Glsuseriii}[2] [] {%
3541   \new@ifnextchar[{\@Glsuseriii@{#1}{#2}}{\@Glsuseriii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3542 \def\@Glsuseriii@#1#2[#3] {%
3543   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentryuseriii{#2}#3}}%
3544 }
```

\glsuseriv behaves like \gls except it always uses the value given by the user4 key and it doesn't mark the entry as used.

\glsuseriv

```
3545 \newrobustcmd*{\glsuseriv}{\@gls@hyp@opt\@glsuseriv}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3546 \newcommand*{\@glsuseriv}[2] [] {%
3547   \new@ifnextchar[{\@glsuseriv@{#1}{#2}}{\@glsuseriv@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3548 \def\@glsuseriv@#1#2[#3] {%
3549   \@gls@field@link{#1}{#2}{\Glsentryuseriv{#2}#3}%
3550 }
```

\Glsuseriv behaves like \glsuseriv except that the first letter is converted to uppercase.

\Glsuseriv

```
3551 \newrobustcmd*{\Glsuseriv}{\@gls@hyp@opt\@Glsuseriv}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3552 \newcommand*{\@Glsuseriv}[2] [] {%
3553   \new@ifnextchar[{\@Glsuseriv@{#1}{#2}}{\@Glsuseriv@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3554 \def\@Glsuseriv@#1#2[#3]{%
3555   \@gls@field@link{#1}{#2}{\Glsentryuseriv{#2}#3}%
3556 }
```

\Glsuseriv behaves like \glsuseriv except that the link text is converted to uppercase.

\Glsuseriv

```
3557 \newrobustcmd*{\Glsuseriv}{\@gls@hyp@opt\@Glsuseriv}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3558 \newcommand*{\@Glsuseriv}[2][\%]
3559   \new@ifnextchar[{\@Glsuseriv@{#1}{#2}}{\@Glsuseriv@{#1}{#2}[]}]
```

Read in the final optional argument:

```
3560 \def\@Glsuseriv@#1#2[#3]{%
3561   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentryuseriv{#2}#3}}%
3562 }
```

\glsuserv behaves like \gls except it always uses the value given by the user5 key and it doesn't mark the entry as used.

\glsuserv

```
3563 \newrobustcmd*{\glsuserv}{\@gls@hyp@opt\@glsuserv}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3564 \newcommand*{\@glsuserv}[2][\%]
3565   \new@ifnextchar[{\@glsuserv@{#1}{#2}}{\@glsuserv@{#1}{#2}[]}]
```

Read in the final optional argument:

```
3566 \def\@glsuserv@#1#2[#3]{%
3567   \@gls@field@link{#1}{#2}{\Glsentryuserv{#2}#3}%
3568 }
```

\Glsuserv behaves like \glsuserv except that the first letter is converted to uppercase.

\Glsuserv

```
3569 \newrobustcmd*{\Glsuserv}{\@gls@hyp@opt\@Glsuserv}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3570 \newcommand*{\@Glsuserv}[2][\%]
3571   \new@ifnextchar[{\@Glsuserv@{#1}{#2}}{\@Glsuserv@{#1}{#2}[]}]
```

Read in the final optional argument:

```
3572 \def\@Glsuserv@#1#2[#3]{%
3573   \@gls@field@link{#1}{#2}{\Glsentryuserv{#2}#3}%
3574 }
```

`\GLSuserv` behaves like `\glsuserv` except that the link text is converted to uppercase.

`\GLSuserv`

```
3575 \newrobustcmd*{\GLSuserv}{\@gls@hyp@opt\@GLSuserv}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3576 \newcommand*{\@GLSuserv}[2] [] {%
```

```
3577 \new@ifnextchar[{\@GLSuserv@{#1}{#2}}{\@GLSuserv@{#1}{#2} []}]
```

Read in the final optional argument:

```
3578 \def\@GLSuserv@#1#2[#3] {%
```

```
3579 \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuserv{#2}#3}}%
```

```
3580 }
```

`\glsuservi` behaves like `\gls` except it always uses the value given by the `user6` key and it doesn't mark the entry as used.

`\glsuservi`

```
3581 \newrobustcmd*{\glsuservi}{\@gls@hyp@opt\@glsuservi}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3582 \newcommand*{\@glsuservi}[2] [] {%
```

```
3583 \new@ifnextchar[{\@glsuservi@{#1}{#2}}{\@glsuservi@{#1}{#2} []}]
```

Read in the final optional argument:

```
3584 \def\@glsuservi@#1#2[#3] {%
```

```
3585 \@gls@field@link{#1}{#2}{\glsentryuservi{#2}#3}}%
```

```
3586 }
```

`\Glsuservi` behaves like `\glsuservi` except that the first letter is converted to uppercase.

`\Glsuservi`

```
3587 \newrobustcmd*{\Glsuservi}{\@gls@hyp@opt\@Glsuservi}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3588 \newcommand*{\@Glsuservi}[2] [] {%
```

```
3589 \new@ifnextchar[{\@Glsuservi@{#1}{#2}}{\@Glsuservi@{#1}{#2} []}]
```

Read in the final optional argument:

```
3590 \def\@Glsuservi@#1#2[#3] {%
```

```
3591 \@gls@field@link{#1}{#2}{\Glsentryuservi{#2}#3}}%
```

```
3592 }
```

`\GLSuservi` behaves like `\glsuservi` except that the link text is converted to uppercase.

`\GLSuservi`

```
3593 \newrobustcmd*{\GLSuservi}{\@gls@hyp@opt\@GLSuservi}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3594 \newcommand*{\@GLSuservi}[2] [] {%
3595   \new@ifnextchar[{\@GLSuservi@{#1}{#2}}{\@GLSuservi@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3596 \def\@GLSuservi@#1#2[#3] {%
3597   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuservi{#2}#3}}%
3598 }
```

Now deal with acronym related keys. First the short form:

\acrshort

```
3599 \newrobustcmd*{\acrshort}{\@gls@hyp@opt\@ns@acrshort}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3600 \newcommand*{\ns@acrshort}[2] [] {%
3601   \new@ifnextchar[{\@acrshort{#1}{#2}}{\@acrshort{#1}{#2} []}]%
3602 }
```

Read in the final optional argument:

```
3603 \def\@acrshort#1#2[#3] {%
3604   \glsdoifexists{#2}%
3605   {%
3606     \let\do@gls@link@checkfirsthyper\relax
3607     \let\glsifplural\@secondoftwo
3608     \let\glsapscase\@firstofthree
3609     \let\glsinsert\@empty
3610     \def\glscustomtext{%
3611       \acronymfont{\glsentryshort{#2}}#3%
3612     }%
```

Call \@gls@link Note that \@gls@link sets \glsstyle.

```
3613   \@gls@link{#1}{#2}{\csname gls@\glsstyle @entryfmt\endcsname}%
3614   }%
3615 }
```

\Acrshort

```
3616 \newrobustcmd*{\Acrshort}{\@gls@hyp@opt\@ns@Acrshort}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3617 \newcommand*{\ns@Acrshort}[2] [] {%
3618   \new@ifnextchar[{\@Acrshort{#1}{#2}}{\@Acrshort{#1}{#2} []}]%
3619 }
```

Read in the final optional argument:

```
3620 \def\@Acrshort#1#2[#3] {%
3621   \glsdoifexists{#2}%
```

```

3622  {%
3623    \let\do@gl@link@checkfirsthyper\relax

3624    \def\glslabel{#2}%
3625    \let\gl@ifplural\@secondoftwo
3626    \let\glscapscase\@secondofthree
3627    \let\gl@insert\@empty
3628    \def\glscustomtext{%
3629      \acronymfont{\gl@entryshort{#2}}#3%
3630    }%

    Call \@gl@link Note that \@gl@link sets \glstype.
3631    \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
3632  }%
3633 }

```

\ACRshort

```

3634 \newrobustcmd*{\ACRshort}{\@gl@hyp@opt\ns@ACRshort}

```

Define the un-starred form. Need to determine if there is a final optional argument

```

3635 \newcommand*{\ns@ACRshort}[2][{}]{%
3636   \new@ifnextchar[\@ACRshort{#1}{#2}]{\@ACRshort{#1}{#2}[]}%
3637 }

```

Read in the final optional argument:

```

3638 \def\@ACRshort#1#2[#3]{%
3639   \gl@ifexists{#2}%
3640   {%
3641     \let\do@gl@link@checkfirsthyper\relax

3642     \def\glslabel{#2}%
3643     \let\gl@ifplural\@secondoftwo
3644     \let\glscapscase\@thirdofthree
3645     \let\gl@insert\@empty
3646     \def\glscustomtext{%
3647       \mfirstucMakeUppercase{\acronymfont{\gl@entryshort{#2}}#3}%
3648     }%

```

Call \@gl@link Note that \@gl@link sets \glstype.

```

3649   \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
3650 }%
3651 }

```

Short plural:

\acrshortpl

```

3652 \newrobustcmd*{\acrshortpl}{\@gl@hyp@opt\ns@acrshortpl}

```


Define the un-starred form. Need to determine if there is a final optional argument

```
3653 \newcommand*{\ns@acrshortpl}[2] [] {%
3654   \new@ifnextchar[{\@acrshortpl{#1}{#2}}{\@acrshortpl{#1}{#2} []}%
3655 }
```

Read in the final optional argument:

```
3656 \def\@acrshortpl#1#2[#3] {%
3657   \glsdoifexists{#2}%
3658   {%
3659     \let\do@gl@link@checkfirsthyper\relax

3660     \def\glslabel{#2}%
3661     \let\glsifplural\@firstoftwo
3662     \let\glscapscase\@firstofthree
3663     \let\glsinsert\@empty
3664     \def\glscustomtext{%
3665       \acronymfont{\glsentryshortpl{#2}}#3%
3666     }%
```

Call \@gl@link Note that \@gl@link sets \glstype.

```
3667   \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
3668 }%
3669 }
```

\Acrshortpl

```
3670 \newrobustcmd*{\Acrshortpl}{\@gl@hyp@opt\ns@Acrshortpl}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3671 \newcommand*{\ns@Acrshortpl}[2] [] {%
3672   \new@ifnextchar[{\@Acrshortpl{#1}{#2}}{\@Acrshortpl{#1}{#2} []}%
3673 }
```

Read in the final optional argument:

```
3674 \def\@Acrshortpl#1#2[#3] {%
3675   \glsdoifexists{#2}%
3676   {%
3677     \let\do@gl@link@checkfirsthyper\relax

3678     \def\glslabel{#2}%
3679     \let\glsifplural\@firstoftwo
3680     \let\glscapscase\@secondofthree
3681     \let\glsinsert\@empty
3682     \def\glscustomtext{%
3683       \acronymfont{\Glsentryshortpl{#2}}#3%
3684     }%
```

Call \@gl@link Note that \@gl@link sets \glstype.

```
3685   \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
3686 }
```

```

3686 }%
3687 }

```

\ACRshortpl

```

3688 \newrobustcmd*{\ACRshortpl}{\@gls@hyp@opt\ns@ACRshortpl}

```

Define the un-starred form. Need to determine if there is a final optional argument

```

3689 \newcommand*{\ns@ACRshortpl}[2][{}]{%
3690   \new@ifnextchar[{\@ACRshortpl{#1}{#2}}{\@ACRshortpl{#1}{#2}[]}%
3691 }

```

Read in the final optional argument:

```

3692 \def\@ACRshortpl#1#2[#3]{%
3693   \glsdoifexists{#2}%
3694   {%
3695     \let\do@gls@link@checkfirsthyper\relax

3696     \def\glslabel{#2}%
3697     \let\glsifplural\@firstoftwo
3698     \let\glscapscase\@thirdofthree
3699     \let\glsinsert\@empty
3700     \def\glscustomtext{%
3701       \mfirstucMakeUppercase{\acronymfont{\glsentryshortpl{#2}}#3}%
3702     }%

```

Call \@gls@link Note that \@gls@link sets \glstype.

```

3703   \@gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
3704 }%
3705 }

```

\acrlong

```

3706 \newrobustcmd*{\acrlong}{\@gls@hyp@opt\ns@acrlong}

```

Define the un-starred form. Need to determine if there is a final optional argument

```

3707 \newcommand*{\ns@acrlong}[2][{}]{%
3708   \new@ifnextchar[{\@acrlong{#1}{#2}}{\@acrlong{#1}{#2}[]}%
3709 }

```

Read in the final optional argument:

```

3710 \def\@acrlong#1#2[#3]{%
3711   \glsdoifexists{#2}%
3712   {%
3713     \let\do@gls@link@checkfirsthyper\relax

3714     \def\glslabel{#2}%
3715     \let\glsifplural\@secondoftwo
3716     \let\glscapscase\@firstofthree
3717     \let\glsinsert\@empty

```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```
3718 \def\glscustomtext{%
3719 \glentrylong{#2}#3%
3720 }%
```

Call \@gls@link Note that \@gls@link sets \glstype.

```
3721 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3722 }%
3723 }
```

\Acrlong

```
3724 \newrobustcmd*{\Acrlong}{\@gls@hyp@opt\ns@Acrlong}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3725 \newcommand*{\ns@Acrlong}[2][{}]{%
3726 \new@ifnextchar[{\@Acrlong{#1}{#2}}{\@Acrlong{#1}{#2}[]}%
3727 }
```

Read in the final optional argument:

```
3728 \def\@Acrlong#1#2[#3]{%
3729 \glsdoifexists{#2}%
3730 {%
3731 \let\do@gls@link@checkfirsthyper\relax

3732 \def\glslabel{#2}%
3733 \let\glsifplural\@secondoftwo
3734 \let\glscapscase\@secondofthree
3735 \let\glsinsert\@empty
```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```
3736 \def\glscustomtext{%
3737 \Glsentrylong{#2}#3%
3738 }%
```

Call \@gls@link. Note that \@gls@link sets \glstype.

```
3739 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3740 }%
3741 }
```

\ACRlong

```
3742 \newrobustcmd*{\ACRlong}{\@gls@hyp@opt\ns@ACRlong}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3743 \newcommand*{\ns@ACRlong}[2][{}]{%
3744 \new@ifnextchar[{\@ACRlong{#1}{#2}}{\@ACRlong{#1}{#2}[]}%
3745 }
```

Read in the final optional argument:

```

3746 \def\@ACRlong#1#2[#3]{%
3747   \glsdoifexists{#2}%
3748   {%
3749     \let\do@gl@link@checkfirsthyper\relax

3750   \def\glslabel{#2}%
3751   \let\glsifplural\@secondoftwo
3752   \let\glscapscase\@thirdofthree
3753   \let\glsinsert\@empty

```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```

3754   \def\glscustomtext{%
3755     \mfirstucMakeUppercase{\glentrylong{#2}#3}%
3756   }%

```

Call \@gl@link. Note that \@gl@link sets \glstype.

```

3757   \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
3758   }%
3759 }

```

Short plural:

\acrlongpl

```

3760 \newrobustcmd*{\acrlongpl}{\@gl@hyp@opt\@ns@acrlongpl}

```

Define the un-starred form. Need to determine if there is a final optional argument

```

3761 \newcommand*{\ns@acrlongpl}[2][ ]{%
3762   \new@ifnextchar[{\@acrlongpl{#1}{#2}}{\@acrlongpl{#1}{#2}[ ]}%
3763 }

```

Read in the final optional argument:

```

3764 \def\@acrlongpl#1#2[#3]{%
3765   \glsdoifexists{#2}%
3766   {%
3767     \let\do@gl@link@checkfirsthyper\relax

3768   \def\glslabel{#2}%
3769   \let\glsifplural\@firstoftwo
3770   \let\glscapscase\@firstofthree
3771   \let\glsinsert\@empty

```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```

3772   \def\glscustomtext{%
3773     \glentrylongpl{#2}#3%
3774   }%

```

Call \@gls@link. Note that \@gls@link sets \glstype.

```
3775 \gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
3776 }%
3777 }
```

\Acrlongpl

```
3778 \newrobustcmd*{\Acrlongpl}{\@gls@hyp@opt\@ns@Acrlongpl}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3779 \newcommand*{\ns@Acrlongpl}[2] [] {%
3780 \new@ifnextchar[{\@Acrlongpl{#1}{#2}}{\@Acrlongpl{#1}{#2} []}%
3781 }
```

Read in the final optional argument:

```
3782 \def\@Acrlongpl#1#2[#3] {%
3783 \glsdoifexists{#2}%
3784 {%
3785 \let\do@gls@link@checkfirsthyper\relax

3786 \def\glslabel{#2}%
3787 \let\glsifplural\@firstoftwo
3788 \let\glscapscase\@secondofthree
3789 \let\glsinsert\@empty
```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```
3790 \def\glscustomtext{%
3791 \Glsentrylongpl{#2}#3%
3792 }%
```

Call \@gls@link. Note that \@gls@link sets \glstype.

```
3793 \gls@link[#1]{#2}{\csname gls@@glo@type @entryfmt\endcsname}%
3794 }%
3795 }
```

\ACRlongpl

```
3796 \newrobustcmd*{\ACRlongpl}{\@gls@hyp@opt\@ns@ACRlongpl}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3797 \newcommand*{\ns@ACRlongpl}[2] [] {%
3798 \new@ifnextchar[{\@ACRlongpl{#1}{#2}}{\@ACRlongpl{#1}{#2} []}%
3799 }
```

Read in the final optional argument:

```
3800 \def\@ACRlongpl#1#2[#3] {%
3801 \glsdoifexists{#2}%
3802 {%
3803 \let\do@gls@link@checkfirsthyper\relax
```

```

3804 \def\glslabel{#2}%
3805 \let\glsifplural\@firstoftwo
3806 \let\glscapscase\@thirdofthree
3807 \let\glsinsert\@empty

```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```

3808 \def\glscustomtext{%
3809 \mfirstucMakeUppercase{\glsentrylongpl{#2}#3}%
3810 }%

```

Call \@gls@link. Note that \@gls@link sets \glstype.

```

3811 \@gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
3812 }%
3813 }

```

1.10.2 Displaying entry details without adding information to the glossary

These commands merely display entry information without adding entries in the associated file or having hyperlinks.

\@gls@entry@field Generic version.

```
\@gls@entry@field{<label>}{<field>}
```

```

3814 \newcommand*{\@gls@entry@field}[2]{%
3815 \csname glo@\glsdetoklabel{#1}@#2\endcsname
3816 }

```

\glsletentryfield \glsletentryfield{<cs>}{<label>}{<field>}

```

3817 \newcommand*{\glsletentryfield}[3]{%
3818 \letcs{#1}{glo@\glsdetoklabel{#2}@#3}%
3819 }

```

\@Gls@entry@field Generic first letter uppercase version.

```
\@Gls@entry@field{<label>}{<field>}
```

```

3820 \newcommand*{\@Gls@entry@field}[2]{%
3821 \letcs\@glo@text{glo@\glsdetoklabel{#1}@#2}%
3822 \ifdef\@glo@text
3823 {%
3824 \xmakefirstuc{\@glo@text}%
3825 }%
3826 {%

```

```

3827 \PackageError{glossaries}{Either glossary entry
3828   '\glsdetoklabel{#1}' doesn't exist or the field '#2'
3829   doesn't exist}{Check you have correctly spelt the entry
3830   label and the field name}%
3831 }%
3832 }

```

Get the entry name (as specified by the name key when the entry was defined). The argument is the label associated with the entry. Note that unless you used `name=false` in the `sanitize` package option you may get unexpected results if the name key contains any commands.

`\glsentryname`

```

3833 \newcommand*{\glsentryname}[1]{\@gls@entry@field{#1}{name}}

```

`\Glsentryname`

```

3834 \newrobustcmd*{\Glsentryname}[1]{%
3835   \@Gls@entryname{#1}%
3836 }

```

`\@Gls@entryname` This is a workaround in the event that the user defies the warning in the manual about not using `\Glsname` or `\Glsentryname` with acronyms. First the default behaviour:

```

3837 \newcommand*{\@Gls@entryname}[1]{%
3838   \@Gls@entry@field{#1}{name}%
3839 }

```

`\@Gls@acrentryname` Now the behaviour when `\setacronymstyle` is used:

```

3840 \newcommand*{\@Gls@acrentryname}[1]{%
3841   \ifglshaslong{#1}%
3842   {%
3843     \letcs\@glo@text{glo@\glsdetoklabel{#1}@name}%
3844     \expandafter\@gls@getbody\@glo@text{}\@nil
3845     \expandafter\ifx\@gls@body\glsentrylong\relax
3846     \expandafter\Glsentrylong\@gls@rest
3847   }else
3848     \expandafter\ifx\@gls@body\glsentryshort\relax
3849     \expandafter\Glsentryshort\@gls@rest
3850   }else
3851     \expandafter\ifx\@gls@body\acronymfont\relax

```

Temporarily make `\glsentryshort` behave like `\Glsentryshort`. (This is on the assumption that the argument of `\acronymfont` is `\glsentryshort{<label>}`, as that's the behaviour of the predefined acronym styles.) This is scoped to localise the effect of the assignment.

```

3852     {%
3853       \let\glsentryshort\Glsentryshort
3854       \@glo@text

```

```

3855         }%
3856     \else
3857         \xmakefirstuc{\@gls@text}%
3858     \fi
3859 \fi
3860 \fi
3861 }%
3862 {%
    Not an acronym
3863     \@Gls@entry@field{#1}{name}%
3864 }%
3865 }

```

Get the entry description (as specified by the description when the entry was defined). The argument is the label associated with the entry. Note that unless you used `description=false` in the `sanitize` package option you may get unexpected results if the description key contained any commands.

```

\glsentrydesc
3866 \newcommand*{\glsentrydesc}[1]{\@Gls@entry@field{#1}{desc}}

\Glsentrydesc
3867 \newrobustcmd*{\Glsentrydesc}[1]{%
3868     \@Gls@entry@field{#1}{desc}%
3869 }

```

Plural form:

```

\glsentrydescplural
3870 \newcommand*{\glsentrydescplural}[1]{%
3871     \@Gls@entry@field{#1}{descplural}%
3872 }

\Glsentrydescplural
3873 \newrobustcmd*{\Glsentrydescplural}[1]{%
3874     \@Gls@entry@field{#1}{descplural}%
3875 }

```

Get the entry text, as specified by the text key when the entry was defined. The argument is the label associated with the entry:

```

\glsentrytext
3876 \newcommand*{\glsentrytext}[1]{\@Gls@entry@field{#1}{text}}

\Glsentrytext
3877 \newrobustcmd*{\Glsentrytext}[1]{%
3878     \@Gls@entry@field{#1}{text}%
3879 }

```


Get the plural form:

`\glsentryplural`

```
3880 \newcommand*{\glsentryplural}[1]{%
3881   \@gls@entry@field{#1}{plural}%
3882 }
```

`\Glsentryplural`

```
3883 \newrobustcmd*{\Glsentryplural}[1]{%
3884   \@Gls@entry@field{#1}{plural}%
3885 }
```

Get the symbol associated with this entry. The argument is the label associated with the entry.

`\glsentrysymbol`

```
3886 \newcommand*{\glsentrysymbol}[1]{%
3887   \@gls@entry@field{#1}{symbol}%
3888 }
```

`\Glsentrysymbol`

```
3889 \newrobustcmd*{\Glsentrysymbol}[1]{%
3890   \@Gls@entry@field{#1}{symbol}%
3891 }
```

Plural form:

`\glsentrysymbolplural`

```
3892 \newcommand*{\glsentrysymbolplural}[1]{%
3893   \@gls@entry@field{#1}{symbolplural}%
3894 }
```

`\Glsentrysymbolplural`

```
3895 \newrobustcmd*{\Glsentrysymbolplural}[1]{%
3896   \@Gls@entry@field{#1}{symbolplural}%
3897 }
```

Get the entry text to be used when the entry is first used in the document (as specified by the first key when the entry was defined).

`\glsentryfirst`

```
3898 \newcommand*{\glsentryfirst}[1]{%
3899   \@gls@entry@field{#1}{first}%
3900 }
```

`\Glsentryfirst`

```
3901 \newrobustcmd*{\Glsentryfirst}[1]{%
3902   \@Gls@entry@field{#1}{first}%
3903 }
```

Get the plural form (as specified by the firstplural key when the entry was defined).

glsentryfirstplural

```
3904 \newcommand*{\glsentryfirstplural}[1]{%
3905   \@gls@entry@field{#1}{firstpl}%
3906 }
```

Glsentryfirstplural

```
3907 \newrobustcmd*{\Glsentryfirstplural}[1]{%
3908   \@Gls@entry@field{#1}{firstpl}%
3909 }
```

Display the glossary type with which this entry is associated (as specified by the type key used when the entry was defined)

\glsentrytype

```
3910 \newcommand*{\glsentrytype}[1]{\@gls@entry@field{#1}{type}}
```

Display the sort text used for this entry. Note that the sort key is sanitize, so unexpected results may occur if the sort key contained commands.

\glsentrysort

```
3911 \newcommand*{\glsentrysort}[1]{%
3912   \@gls@entry@field{#1}{sort}%
3913 }
```

\glsentryuseri Get the first user key (as specified by the user1 when the entry was defined).
The argument is the label associated with the entry.

```
3914 \newcommand*{\glsentryuseri}[1]{%
3915   \@gls@entry@field{#1}{useri}%
3916 }
```

\Glsentryuseri

```
3917 \newrobustcmd*{\Glsentryuseri}[1]{%
3918   \@Gls@entry@field{#1}{useri}%
3919 }
```

\glsentryuserii Get the second user key (as specified by the user2 when the entry was defined).
The argument is the label associated with the entry.

```
3920 \newcommand*{\glsentryuserii}[1]{%
3921   \@gls@entry@field{#1}{userii}%
3922 }
```

\Glsentryuserii

```
3923 \newrobustcmd*{\Glsentryuserii}[1]{%
3924   \@Gls@entry@field{#1}{userii}%
3925 }
```

`\glentryuseriii` Get the third user key (as specified by the user3 when the entry was defined).
The argument is the label associated with the entry.

```
3926 \newcommand*{\glentryuseriii}[1]{%  
3927   \@gls@entry@field{#1}{useriii}%  
3928 }
```

`\Glsentryuseriii`

```
3929 \newrobustcmd*{\Glsentryuseriii}[1]{%  
3930   \@Gls@entry@field{#1}{useriii}%  
3931 }
```

`\glentryuseriv` Get the fourth user key (as specified by the user4 when the entry was defined).
The argument is the label associated with the entry.

```
3932 \newcommand*{\glentryuseriv}[1]{%  
3933   \@gls@entry@field{#1}{useriv}%  
3934 }
```

`\Glsentryuseriv`

```
3935 \newrobustcmd*{\Glsentryuseriv}[1]{%  
3936   \@Gls@entry@field{#1}{useriv}%  
3937 }
```

`\glentryuserv` Get the fifth user key (as specified by the user5 when the entry was defined).
The argument is the label associated with the entry.

```
3938 \newcommand*{\glentryuserv}[1]{%  
3939   \@gls@entry@field{#1}{userv}%  
3940 }
```

`\Glsentryuserv`

```
3941 \newrobustcmd*{\Glsentryuserv}[1]{%  
3942   \@Gls@entry@field{#1}{userv}%  
3943 }
```

`\glentryuservi` Get the sixth user key (as specified by the user6 when the entry was defined).
The argument is the label associated with the entry.

```
3944 \newcommand*{\glentryuservi}[1]{%  
3945   \@gls@entry@field{#1}{uservi}%  
3946 }
```

`\Glsentryuservi`

```
3947 \newrobustcmd*{\Glsentryuservi}[1]{%  
3948   \@Gls@entry@field{#1}{uservi}%  
3949 }
```

`\glentryshort` Get the short key (as specified by the short the entry was defined). The argument is the label associated with the entry.

```
3950 \newcommand*{\glentryshort}[1]{\@gls@entry@field{#1}{short}}
```

```

\Glsentryshort
3951 \newrobustcmd*{\Glsentryshort}[1]{%
3952   \@Gls@entry@field{#1}{short}%
3953 }

\glsentryshortpl  Get the short plural key (as specified by the shortplural the entry was defined).
                  The argument is the label associated with the entry.
3954 \newcommand*{\glsentryshortpl}[1]{\@Gls@entry@field{#1}{shortpl}}

\Glsentryshortpl
3955 \newrobustcmd*{\Glsentryshortpl}[1]{%
3956   \@Gls@entry@field{#1}{shortpl}%
3957 }

\glsentrylong  Get the long key (as specified by the long the entry was defined). The argument
               is the label associated with the entry.
3958 \newcommand*{\glsentrylong}[1]{\@Gls@entry@field{#1}{long}}

\Glsentrylong
3959 \newrobustcmd*{\Glsentrylong}[1]{%
3960   \@Gls@entry@field{#1}{long}%
3961 }

\glsentrylongpl  Get the long plural key (as specified by the longplural the entry was defined).
                 The argument is the label associated with the entry.
3962 \newcommand*{\glsentrylongpl}[1]{\@Gls@entry@field{#1}{longpl}}

\Glsentrylongpl
3963 \newrobustcmd*{\Glsentrylongpl}[1]{%
3964   \@Gls@entry@field{#1}{longpl}%
3965 }

Short cut macros to access full form:

\glsentryfull
3966 \newcommand*{\glsentryfull}[1]{%
3967   \acrfullformat{\glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%
3968 }

\Glsentryfull
3969 \newrobustcmd*{\Glsentryfull}[1]{%
3970   \acrfullformat{\Glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%
3971 }

\glsentryfullpl
3972 \newcommand*{\glsentryfullpl}[1]{%
3973   \acrfullformat{\glsentrylongpl{#1}}{\acronymfont{\glsentryshortpl{#1}}}%
3974 }

```

\Glsentryfullpl

```
3975 \newrobustcmd*{\Glsentryfullpl}[1]{%
3976   \acrfullformat{\Glsentrylongpl{#1}}{\acronymfont{\Glsentryshortpl{#1}}}%
3977 }
```

\glsentrynumberlist Displays the number list as is.

```
3978 \newcommand*{\glsentrynumberlist}[1]{%
3979   \glsdoifexists{#1}%
3980   {%
3981     \@gls@entry@field{#1}{numberlist}%
3982   }%
3983 }
```

\glsdisplaynumberlist Formats the number list for the given entry label. Doesn't work with hyperref.

```
3984 \@ifpackageloaded{hyperref} {%
3985   \newcommand*{\glsdisplaynumberlist}[1]{%
3986     \GlossariesWarning
3987     {%
3988       \string\glsdisplaynumberlist\space
3989       doesn't work with hyperref.^^JUsing
3990       \string\glsentrynumberlist\space instead%
3991     }%
3992     \glsentrynumberlist{#1}%
3993   }%
3994 }%
3995 {%
3996   \newcommand*{\glsdisplaynumberlist}[1]{%
3997     \glsdoifexists{#1}%
3998     {%
3999       \bgroup
4000
4001       \edef\@glo@label{\glsdetoklabel{#1}}%
4002       \let\@org@glsnnumberformat\glsnnumberformat
4003       \def\glsnnumberformat##1{##1}%
4004       \protected@edef\the@numberlist{%
4005         \csname glo@\@glo@label @numberlist\endcsname}%
4006       \def\@gls@numlist@sep{}%
4007       \def\@gls@numlist@nextsep{}%
4008       \def\@gls@numlist@lastsep{}%
4009       \def\@gls@thislist{}%
4010       \def\@gls@donext@def{}%
4011       \renewcommand\do[1]{%
4012         \protected@edef\@gls@thislist{%
4013           \@gls@thislist
4014           \noexpand\@gls@numlist@sep
4015           ##1%
4016         }%
4017         \let\@gls@numlist@sep\@gls@numlist@nextsep
4018         \def\@gls@numlist@nextsep{\glsnnumlistsep}%

```

```

4018         \@gls@donext@def
4019         \def\@gls@donext@def{%
4020             \def\@gls@numlist@lastsep{\glsnumlistlastsep}%
4021         }%
4022     }%
4023     \expandafter \glsnumlistparser \expandafter{\the@numberlist}%
4024     \let\@gls@numlist@sep\@gls@numlist@lastsep
4025     \@gls@thislist
4026 \egroup
4027 }%
4028 }
4029 }

```

`\glsnumlistsep`

```
4030 \newcommand*{\glsnumlistsep}{, }
```

`\glsnumlistlastsep`

```
4031 \newcommand*{\glsnumlistlastsep}{ \& }
```

`\glshyperlink` Provide a hyperlink to a glossary entry without adding information to the glossary file. The entry needs to be added using a command like `\glslink` or `\glsadd` to ensure that the target is defined. The first (optional) argument specifies the link text. The entry name is used by default. The second argument is the entry label.

```

4032 \newcommand*{\glshyperlink}[2][\glsentrytext{\@glo@label}]{%
4033   \def\@glo@label{#2}%
4034   \@glslink{\glo@linkprefix\glsdetoklabel{#2}}{#1}}

```

1.11 Adding an entry to the glossary without generating text

The following keys are provided for `\glsadd` and `\glsaddall`:

```
4035 \define@key{glossadd}{counter}{\def\@gls@counter{#1}}
```

```
4036 \define@key{glossadd}{format}{\def\@glsnumberformat{#1}}
```

This key is only used by `\glsaddall`:

```
4037 \define@key{glossadd}{types}{\def\@glo@type{#1}}
```

`\glsadd[<options>]{<label>}`

Add a term to the glossary without generating any link text. The optional argument indicates which counter to use, and how to format it (using a key-value list) the second argument is the entry label. Note that *<options>* only has two keys: counter and format (the types key will be ignored).

`\glsadd`

```
4038 \newrobustcmd*{\glsadd}[2][ ]{%
```

Need to move to horizontal mode if not already in it, but only if not in preamble.

```

4039 \@gls@adjustmode
4040 \glsdoifexists{#2}%
4041 {%
4042 \def\@glsnumberformat{glsnumberformat}%
4043 \edef\@gls@counter{\csname glo@glsetoklabel{#2}@counter\endcsname}%
4044 \setkeys{glossadd}{#1}%

```

Store the entry's counter in \theglsentrycounter

```

4045 \@gls@saveentrycounter
4046 \@do@wrglossary{#2}%
4047 }%
4048 }

```

\@gls@adjustmode

```

4049 \newcommand*{\@gls@adjustmode}{}
4050 \AtBeginDocument{\renewcommand*{\@gls@adjustmode}{\ifvmode\mbox{}\fi}}

```

\glsaddall[*<option list>*]

Add all terms defined for the listed glossaries (without displaying any text). If types key is omitted, apply to all glossary types.

\glsaddall

```

4051 \newrobustcmd*{\glsaddall}[1][]{%
4052 \edef\@glo@type{\@glo@types}%
4053 \setkeys{glossadd}{#1}%
4054 \forallglsentries[\@glo@type]{\@glo@entry}{%
4055 \glsadd[#1]{\@glo@entry}%
4056 }%
4057 }

```

\glsaddallunused **\glsaddallunused[*<glossary type>*]**

Add all used terms defined for the listed glossaries (without displaying any text). If optional argument is omitted, apply to all glossary types. This should typically go at the end of the document.

```

4058 \newrobustcmd*{\glsaddallunused}[1][\@glo@types]{%
4059 \forallglsentries[#1]{\@glo@entry}%
4060 {%
4061 \ifglsused{\@glo@entry}{\glsadd[format=glsignore]{\@glo@entry}}%
4062 }%
4063 }

```

\glsignore

```

4064 \newcommand*{\glsignore}[1]{}

```

1.12 Creating associated files

The `\writeist` command creates the associated customized `.ist` `makeindex` style file. While defining this command, some characters have their catcodes temporarily changed to ensure they get written to the `.ist` file correctly. The `makeindex` actual character (usually `@`) is redefined to be a `?`, to allow internal commands to be written to the glossary file output file.

The special characters are stored in `\@gls@actualchar`, `\@gls@encapchar`, `\@gls@levelchar` and `\@gls@quotechar` to make them easier to use later, but don't change these values, because the characters are encoded in the command definitions that are used to escape the special characters (which means that the user no longer needs to worry about `makeindex` special characters).

The symbols and numbers label for group headings are hardwired into the `.ist` file as `glssymbols` and `glsnumbers`, the group titles can be translated (so that `\glssymbolsgroupname` replaces `glssymbols` and `\glsnumbersgroupname` replaces `glsnumbers`) using the command `\glssetgrouptitle` which is defined in `.` This is done to prevent any problem characters in `\glssymbolsgroupname` and `\glsnumbersgroupname` from breaking hyperlinks.

`\glsopenbrace` Define `\glsopenbrace` to make it easier to write an opening brace to a file.

```
4065 \edef\glsopenbrace{\expandafter\@gobble\string\{}
```

`\glsclosebrace` Define `\glsclosebrace` to make it easier to write an opening brace to a file.

```
4066 \edef\glsclosebrace{\expandafter\@gobble\string\}}
```

`\glsbackslash` Define `\glsbackslash` to make it easier to write a backslash to a file.

```
4067 \edef\glsbackslash{\expandafter\@gobble\string\\}
```

`\glsquote` Define command that makes it easier to write quote marks to a file in the event that the double quote character has been made active.

```
4068 \edef\glsquote#1{\string"#1\string"}
```

`\glspercentchar` Define `\glspercentchar` to make it easier to write a percent character to a file.

```
4069 \edef\glspercentchar{\expandafter\@gobble\string\%}
```

`\glstildechar` Define `\glstildechar` to make it easier to write a tilde character to a file.

```
4070 \edef\glstildechar{\string~}
```

`\@glsfirstletter` Define the first letter to come after the digits 0,...,9. Only required for `xindy`.

```
4071 \ifglsxindy
```

```
4072 \newcommand*{\@glsfirstletter}{A}
```

```
4073 \fi
```

`stLetterAfterDigits` Sets the first letter to come after the digits 0,...,9.

```
4074 \ifglsxindy
```

```
4075 \newcommand*{\GlsSetXdyFirstLetterAfterDigits}[1]{%
```



```

4076     \renewcommand*{\@glsfirstletter}{#1}}
4077 \else
4078     \newcommand*{\GlsSetXdyFirstLetterAfterDigits}[1]{%
4079         \glsnoxywarning\GlsSetXdyFirstLetterAfterDigits}
4080 \fi

\@glsminrange Define the minimum number of successive location references to merge into a
range.
4081 \newcommand*{\@glsminrange}{2}

etXdyMinRangeLength Set the minimum range length. The value must either be none or a positive
integer. The glossaries package doesn't check if the argument is valid, that is left
to xindy.
4082 \ifglxindy
4083     \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
4084         \renewcommand*{\@glsminrange}{#1}}
4085 \else
4086     \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
4087         \glsnoxywarning\GlsSetXdyMinRangeLength}
4088 \fi

\writeist
4089 \ifglxindy
    Code to use if xindy is required.
4090 \def\writeist{%
    Define write register if not already defined
4091     \ifundef{\glswrite}{\newwrite\glswrite}{}%
    Update attributes list
4092     \@gls@addpredefinedattributes
    Open the file.
4093     \openout\glswrite=\istfilename
    Write header comment at the start of the file
4094     \write\glswrite{;; xindy style file created by the glossaries
4095         package}%
4096     \write\glswrite{;; for document '\jobname' on
4097         \the\year-\the\month-\the\day}%
    Specify the required styles
4098     \write\glswrite{^^J; required styles^^J}
4099     \@for\@xdystyle:=\@xdyrequiredstyles\do{%
4100         \ifx\@xdystyle\@empty
4101         \else
4102             \protected@write\glswrite{{(require
4103                 \string"\@xdystyle.xdy\string")}}%
4104         \fi
4105     }%

```

List the allowed attributes (possible values used by the format key)

```
4106 \write\glswrite{^^J%
4107 ; list of allowed attributes (number formats)^^J}%
4108 \write\glswrite{(define-attributes ((\@xdyattributes)))}%
```

Define any additional alphabets

```
4109 \write\glswrite{^^J; user defined alphabets^^J}%
4110 \write\glswrite{\@xdyuseralphabets}%
```

Define location classes.

```
4111 \write\glswrite{^^J; location class definitions^^J}%
```

As from version 3.0, locations are now specified as $\{\langle Hprefix \rangle\}\{\langle number \rangle\}$, so need to add all possible combinations of location types.

```
4112 \@for\@gls@classI:=\@gls@xdy@locationlist\do{%
```

Case were $\langle Hprefix \rangle$ is empty:

```
4113 \protected@write\glswrite{}{(define-location-class
4114 \string"\@gls@classI\string"^^J\space\space\space
4115 (
4116 :sep "{{"
4117 \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
4118 :sep "}"
4119 )
4120 ^^J\space\space\space
4121 :min-range-length \@glsminrange^^J%
4122 )
4123 }%
```

Nested iteration over all classes:

```
4124 {%
4125 \@for\@gls@classII:=\@gls@xdy@locationlist\do{%
4126 \protected@write\glswrite{}{(define-location-class
4127 \string"\@gls@classII-\@gls@classI\string"
4128 ^^J\space\space\space
4129 (
4130 :sep "{"
4131 \csname @gls@xdy@Lclass@\@gls@classII\endcsname\space
4132 :sep "{{"
4133 \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
4134 :sep "}"
4135 )
4136 ^^J\space\space\space
4137 :min-range-length \@glsminrange^^J%
4138 )
4139 }%
4140 }%
4141 }%
4142 }%
```

User defined location classes (needs checking for new location format).

```

4143 \write\glswrite{^^J; user defined location classes}%
4144 \write\glswrite{\@xdyuserlocationdefs}%

```

Cross-reference class. (The unverified option is used as the cross-references are supplied using the list of labels along with the optional argument for `\glsseeformat` which xindy won't recognise.)

```

4145 \write\glswrite{^^J; define cross-reference class^^J}%
4146 \write\glswrite{(define-crossref-class \string"see\string"
4147 :unverified )}%

```

Define how cross-references should be displayed. This adds an empty set of braces after the cross-referencing information allowing for the final argument of `\glsseeformat` which gets ignored. (When using `makeindex` this final argument contains the location information which is not required.)

```

4148 \write\glswrite{(markup-crossref-list
4149 :class \string"see\string"^^J\space\space\space
4150 :open \string"\string\glsseeformat\string"
4151 :close \string"{}\string")}%

```

List the order to sort the classes.

```

4152 \write\glswrite{^^J; define the order of the location classes}%
4153 \write\glswrite{(define-location-class-order
4154 (\@xdylocationclassorder))}%

```

Specify what to write to the start and end of the glossary file.

```

4155 \write\glswrite{^^J; define the glossary markup^^J}%

4156 \write\glswrite{(markup-index^^J\space\space\space
4157 :open \string"\string
4158 \glossarysection[\string\glossarytoctitle]{\string
4159 \glossarytitle}\string\glossarypreamble}%

```

Add all the xindy-only macro definitions (needed to prevent errors in the event that the user changes from xindy to `makeindex`)

```

4160 \@for\@this@ctr:=\@xdycounters\do{%
4161   {%
4162     \@for\@this@attr:=\@xdyattributelist\do{%
4163       \protected\write\glswrite{}\string\providecommand*%
4164       \expandafter\string
4165       \csname glsX\@this@ctr X\@this@attr\endcsname[2]%
4166       {%
4167         \string\setentrycounter
4168         [\expandafter\@gobble\string\#1]{\@this@ctr}%
4169         \expandafter\string
4170         \csname\@this@attr\endcsname
4171         {\expandafter\@gobble\string\#2}%
4172       }%
4173     }%
4174   }%
4175 }%
4176 }%

```

Add the end part of the open tag and the rest of the markup-index information:

```
4177 \write\glswrite{%
4178   \string\begin
4179   {theglossary}\string\glossaryheader\glstildechar n\string" ^^J\space
4180   \space\space:close \string"\glpercentchar\glstildechar n\string
4181   \end{theglossary}\string\glossarypostamble
4182   \glstildechar n\string" ^^J\space\space\space
4183   :tree}}%
```

Specify what to put between letter groups

```
4184 \write\glswrite{(markup-letter-group-list
4185   :sep \string"\string\glsgroupskip\glstildechar n\string"}}%
```

Specify what to put between entries

```
4186 \write\glswrite{(markup-indexentry
4187   :open \string"\string\relax \string\glsresetentrylist
4188   \glstildechar n\string"}}%
```

Specify how to format entries

```
4189 \write\glswrite{(markup-locclass-list :open
4190   \string"\glsopenbrace\string\glossaryentrynumbers
4191   \glsopenbrace\string\relax\space \string"^^J\space\space\space
4192   :sep \string", \string"
4193   :close \string"\glsclosebrace\glsclosebrace\string"}}%
```

Specify how to separate location numbers

```
4194 \write\glswrite{(markup-locref-list
4195   :sep \string"\string\delimN\space\string"}}%
```

Specify how to indicate location ranges

```
4196 \write\glswrite{(markup-range
4197   :sep \string"\string\delimR\space\string"}}%
```

Specify 2-page and 3-page suffixes, if defined. First, the values must be sanitized to write them explicitly.

```
4198 \@onelevel@sanitize\gls@suffixF
4199 \@onelevel@sanitize\gls@suffixFF
4200 \ifx\gls@suffixF\@empty
4201 \else
4202   \write\glswrite{(markup-range
4203     :close "\gls@suffixF" :length 1 :ignore-end)}}%
4204 \fi
4205 \ifx\gls@suffixFF\@empty
4206 \else
4207   \write\glswrite{(markup-range
4208     :close "\gls@suffixFF" :length 2 :ignore-end)}}%
4209 \fi
```

Specify how to format locations.

```
4210 \write\glswrite{^^J; define format to use for locations^^J}%
4211 \write\glswrite{\@xdylocref}%
```

Specify how to separate letter groups.

```
4212 \write\glswrite{^^J; define letter group list format^^J}%
4213 \write\glswrite{(markup-letter-group-list
4214 :sep \string"\string\glsgroupskip\glstildechar n\string")}%
```

Define letter group headings.

```
4215 \write\glswrite{^^J; letter group headings^^J}%
4216 \write\glswrite{(markup-letter-group
4217 :open-head \string"\string\glsgroupheading
4218 \glsoopenbrace\string"^^J\space\space\space
4219 :close-head \string"\glsclosebrace\string")}%
```

Define additional letter groups.

```
4220 \write\glswrite{^^J; additional letter groups^^J}%
4221 \write\glswrite{\@xdylettergroups}%
```

Define additional sort rules

```
4222 \write\glswrite{^^J; additional sort rules^^J}
4223 \write\glswrite{\@xdysortrules}%
```

Close the style file

```
4224 \closeout\glswrite
```

Suppress any further calls.

```
4225 \let\writeist\relax
4226 }
4227 \else
```

Code to use if makeindex is required.

```
4228 \edef\@gls@actualchar{\string?}
4229 \edef\@gls@encapchar{\string|}
4230 \edef\@gls@levelchar{\string!}
4231 \edef\@gls@quotechar{\string"}
4232 \def\writeist{\relax
4233 \ifundef\glswrite{\newwrite\glswrite}{\relax
4234 \openout\glswrite=\istfilename
4235 \write\glswrite{\glspersentchar\space makeindex style file
4236 created by the glossaries package}
4237 \write\glswrite{\glspersentchar\space for document
4238 '\jobname' on \the\year-\the\month-\the\day}
4239 \write\glswrite{actual '\@gls@actualchar'}
4240 \write\glswrite{encap '\@gls@encapchar'}
4241 \write\glswrite{level '\@gls@levelchar'}
4242 \write\glswrite{quote '\@gls@quotechar'}
4243 \write\glswrite{keyword \string"\string\glossaryentry\string"}
4244 \write\glswrite{preamble \string"\string\glossarysection[\string
4245 \glossarytoctitle]{\string\glossarytitle}\string
4246 \glossarypreamble\string\n\string\begin{theglossary}\string
4247 \glossaryheader\string\n\string"}
4248 \write\glswrite{postamble \string"\string%\string\n\string
4249 \end{theglossary}\string\glossarypostamble\string\n
4250 \string"}
```

```

4251 \write\glswrite{group_skip \string"\string\glsgroupskip\string\n
4252 \string"}
4253 \write\glswrite{item_0 \string"\string%\string\n\string"}
4254 \write\glswrite{item_1 \string"\string%\string\n\string"}
4255 \write\glswrite{item_2 \string"\string%\string\n\string"}
4256 \write\glswrite{item_01 \string"\string%\string\n\string"}
4257 \write\glswrite{item_x1
4258 \string"\string\relax \string\glresetentrylist\string\n
4259 \string"}
4260 \write\glswrite{item_12 \string"\string%\string\n\string"}
4261 \write\glswrite{item_x2
4262 \string"\string\relax \string\glresetentrylist\string\n
4263 \string"}

4264 \write\glswrite{delim_0 \string"\string\{\string
4265 \glssymbols\string\{\string\relax \string"}
4266 \write\glswrite{delim_1 \string"\string\{\string
4267 \glssymbols\string\{\string\relax \string"}
4268 \write\glswrite{delim_2 \string"\string\{\string
4269 \glssymbols\string\{\string\relax \string"}
4270 \write\glswrite{delim_t \string"\string\}\string\}\string"}
4271 \write\glswrite{delim_n \string"\string\delimN \string"}
4272 \write\glswrite{delim_r \string"\string\delimR \string"}
4273 \write\glswrite{headings_flag 1}
4274 \write\glswrite{heading_prefix
4275 \string"\string\glsgroupheading\string\{\string"}
4276 \write\glswrite{heading_suffix
4277 \string"\string\}\string\relax
4278 \string\glresetentrylist \string"}
4279 \write\glswrite{symhead_positive \string"glssymbols\string"}
4280 \write\glswrite{numhead_positive \string"glssymbols\string"}
4281 \write\glswrite{page_compositor \string"glscpositor\string"}
4282 \@gls@escbsdq\gls@suffixF
4283 \@gls@escbsdq\gls@suffixFF
4284 \ifx\gls@suffixF\@empty
4285 \else
4286 \write\glswrite{suffix_2p \string"\gls@suffixF\string"}
4287 \fi
4288 \ifx\gls@suffixFF\@empty
4289 \else
4290 \write\glswrite{suffix_3p \string"\gls@suffixFF\string"}
4291 \fi
4292 \closeout\glswrite
4293 \let\writeist\relax
4294 }
4295 \fi

```

The command `\noist` will suppress the creation of the `.ist` file. Obviously you need to use this command before `\writeist` to have any effect.

`\noist`

```
4296 \newcommand{\noist}{%
```

Update attributes list

```
4297 \@gls@addpredefinedattributes
```

```
4298 \let\writeist\relax
```

```
4299 }
```

`\@makeglossary` is an internal command that takes an argument indicating the glossary type. This command will create the glossary file required by `makeindex` for the given glossary type, using the extension supplied by the `<out-ext>` parameter used in `\newglossary` (and it will also activate the `\glossary` command, and create the customized `.ist` `makeindex` style file).

Note that you can't use `\@makeglossary` for only some of the defined glossaries. You either need to have a `\makeglossary` for all glossaries or none (otherwise you will end up with a situation where \TeX is trying to write to a non-existent file). The relevant glossary must be defined prior to using `\@makeglossary`.

`\@makeglossary`

```
4300 \newcommand*{\@makeglossary}[1]{%
```

```
4301 \ifglossaryexists{#1}%
```

```
4302 {%
```

Only create a new write if `savewrites=false` otherwise create a token to collect the information.

```
4303 \ifglssavewrites
```

```
4304 \expandafter\newtoks\csname glo@#1@filetok\endcsname
```

```
4305 \else
```

```
4306 \expandafter\newwrite\csname glo@#1@file\endcsname
```

```
4307 \expandafter\@glsopenfile\csname glo@#1@file\endcsname{#1}%
```

```
4308 \fi
```

```
4309 \@gls@renewglossary
```

```
4310 \writeist
```

```
4311 }%
```

```
4312 {%
```

```
4313 \PackageError{glossaries}%
```

```
4314 {Glossary type ‘#1’ not defined}%
```

```
4315 {New glossaries must be defined before using \string\makeglossary}%
```

```
4316 }%
```

```
4317 }
```

`\@glsopenfile` Open write file associated with the given glossary.

```
4318 \newcommand*{\@glsopenfile}[2]{%
```

```
4319 \immediate\openout#1=\jobname.\csname @glotype@#2@out\endcsname
```

```
4320 \PackageInfo{glossaries}{Writing glossary file
```

```
4321 \jobname.\csname @glotype@#2@out\endcsname}%
```

```
4322 }
```

\@closegls

```

4323 \newcommand*{\@closegls}[1]{%
4324   \closeout\csname glo@#1@file\endcsname
4325 }
4326 %   \end{macrocode}
4327 %\end{macro}
4328 %
4329 %\begin{macro}{\@gls@automake}
4330 %\changes{4.08}{2014-07-30}{new}
4331 %   \begin{macrocode}
4332 \ifglxindy
4333   \newcommand*{\@gls@automake}[1]{%
4334     \ifglossaryexists{#1}
4335     {%
4336       \@closegls{#1}%
4337       \ifdefstring{glsorder}{letter}%
4338         {\def\@gls@order{-M ord/letorder }}%
4339         {\let\@gls@order\@empty}%
4340       \ifcsundef{xdy@#1@language}%
4341         {\let\@gls@langmod\@xdy@main@language}%
4342         {\letcs\@gls@langmod{xdy@#1@language}}%
4343       \edef\@gls@dothiswrite{\noexpand\write18{xindy
4344         -I xindy
4345         \@gls@order
4346         -L \@gls@langmod\space
4347         -M \@gls@istfilebase\space
4348         -C \@gls@codepage\space
4349         -t \jobname.\csuse{@glotype@#1@log}
4350         -o \jobname.\csuse{@glotype@#1@in}
4351         \jobname.\csuse{@glotype@#1@out}}}%
4352     }%
4353     \@gls@dothiswrite
4354   }%
4355   {%
4356     \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
4357   }%
4358 }
4359 \else
4360   \newcommand*{\@gls@automake}[1]{%
4361     \ifglossaryexists{#1}
4362     {%
4363       \@closegls{#1}%
4364       \ifdefstring{glsorder}{letter}%
4365         {\def\@gls@order{-l }}%
4366         {\let\@gls@order\@empty}%
4367       \edef\@gls@dothiswrite{\noexpand\write18{makeindex \@gls@order
4368         -s \istfilename\space
4369         -t \jobname.\csuse{@glotype@#1@log}
4370         -o \jobname.\csuse{@glotype@#1@in}

```



```

4371      \jobname.\csuse{@glotype@#1@out}}%
4372    }%
4373    \@gls@dothiswrite
4374  }%
4375  {%
4376    \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
4377  }%
4378 }
4379 \fi

```

`\nomakeglossaries` Issue warning that `\makeglossaries` hasn't been used.

```

4380 \newcommand*{\@warn@nomakeglossaries}{%

```

Only use this if warning if `\printglossary` has been used without `\makeglossaries`

```

4381 \newcommand*{\@warn@nomakeglossaries}{\@warn@nomakeglossaries}

```

`\makeglossaries` will use `\makeglossary` for each glossary type that has been defined. New glossaries need to be defined before using `\makeglossary`, so have `\makeglossaries` redefine `\newglossary` to prevent it being used afterwards.

`\makeglossaries`

```

4382 \newcommand*{\makeglossaries}{%

```

Define the write used for style file also used for all other output files if `savewrites=true`.

```

4383 \ifundef{glswrite}{\newwrite\glswrite}{}%

```

If the user removes the glossary package from their document, ensure the next run doesn't throw a load of undefined control sequence errors when the aux file is parsed.

```

4384 \protected@write\@auxout{}{\string\providecommand\string\@glsorder[1]{}%

```

```

4385 \protected@write\@auxout{}{\string\providecommand\string\@istfilename[1]{}%

```

Write the name of the style file to the aux file (needed by `makeglossaries`)

```

4386 \protected@write\@auxout{}{\string\@istfilename{\istfilename}}%

```

```

4387 \protected@write\@auxout{}{\string\@glsorder{\glsorder}}

```

Iterate through each glossary type and activate it.

```

4388 \@for\@glo@type:=\@glo@types\do{%

```

```

4389   \ifthenelse{\equal{\@glo@type}{}}{}{}%

```

```

4390   \@makeglossary{\@glo@type}}%

```

```

4391 }%

```

New glossaries must be created before `\makeglossaries` so disable `\newglossary`.

```

4392 \renewcommand*\newglossary[4][]{%

```

```

4393 \PackageError{glossaries}{New glossaries

```

```

4394 must be created before \string\makeglossaries}{You need

```

```

4395 to move \string\makeglossaries\space after all your

```

```

4396 \string\newglossary\space commands}}%

```

Any subsequence instances of this command should have no effect

```
4397 \let\@makeglossary\relax
4398 \let\makeglossary\relax
4399 \let\makeglossaries\relax
```

Disable all commands that have no effect after \makeglossaries

```
4400 \@disable@onlypremakeg
```

Allow see key:

```
4401 \let\gls@checkseeallowed\relax
```

Suppress warning about no \makeglossaries

```
4402 \let\warn@nomakeglossaries\relax
```

Activate warning about missing \printglossary

```
4403 \def\warn@noprntglossary{%
4404   \GlossariesWarningNoLine{No \string\printglossary\space
4405     or \string\printglossaries\space
4406     found.^^J(Remove \string\makeglossaries\space if you don't want
4407     any glossaries.)^^JThis document will not have a glossary}%
4408 }%
```

Declare list parser for \glsdisplaynumberlist

```
4409 \ifglssavenumberlist
4410   \edef\@gls@dodolistparser{\noexpand\DeclareListParser
4411     {\noexpand\glsnumlistparser}{\delimN}}%
4412   \@gls@dodolistparser
4413 \fi
```

Prevent user from also using \makenoidxglossaries

```
4414 \let\makenoidxglossaries\@no@makeglossaries
```

Prohibit sort key in printgloss family:

```
4415 \renewcommand*{\@printgloss@setsort}{%
4416   \let\@glo@assign@sortkey\@glo@no@assign@sortkey
4417 }%
```

Check the automake setting:

```
4418 \ifglssautomake
4419   \renewcommand*{\@gls@doautomake}{%
4420     \@for\@gls@type:=\@glo@types\do{%
4421       \ifdefempty{\@gls@type}{}%
4422       {\@gls@automake{\@gls@type}}%
4423     }%
4424   }%
4425 \fi
4426 }
```

Must occur in the preamble:

```
4427 \@onlypreamble{\makeglossaries}
```

\glswrite The definition of \glswrite has now been moved to \makeglossaries so that it's only defined if needed.

The `\makeglossary` command is redefined to be identical to `\makeglossaries`.
 (This is done to reinforce the message that you must either use `\@makeglossary`
 for all the glossaries or for none of them.)

`\makeglossary`

```
4428 \let\makeglossary\makeglossaries
```

If `\makeglossaries` hasn't been used, issue a warning. Also issue a warning
 if neither `\printglossaries` nor `\printglossary` have been used.

```
4429 \AtEndDocument{%
4430   \warn@nomakeglossaries
4431   \warn@noprintglossary
4432 }
```

`\makenoidxglossaries` Analogous to `\makeglossaries` this activates the commands needed for `\printnoidxglossary`

```
4433 \newcommand*\makenoidxglossaries{%
```

Redefine empty glossary warning:

```
4434   \renewcommand{\@gls@noref@warn}[1]{%
4435     \GlossariesWarning{Empty glossary for
4436     \string\printnoidxglossary[type={##1}].
4437     Rerun may be required (or you may have forgotten to use
4438     commands like \string\gls).}%
4439   }%
```

Don't escape makeindex/xindy characters

```
4440   \let\@gls@checkmkidxchars\@gobble
```

Write glossary information to aux instead of glossary files

```
4441   \let\@@do@@wrglossary\gls@noidxglossary
```

Switch on group headings that use the character code:

```
4442   \let\@gls@getgrouptitle\@gls@noidx@getgrouptitle
```

Allow see key:

```
4443   \let\gls@checkseeallowed\relax
```

Redefine cross-referencing macro:

```
4444   \renewcommand{\@do@seeglossary}[2]{%
4445     \edef\@gls@label{\glsdetoklabel{##1}}%
4446     \protected@write\@auxout{%
4447       \string\@gls@reference
4448       {\csname glo@\@gls@label @type\endcsname}%
4449       {\@gls@label}%
4450       {%
4451         \string\glsseeformat##2}%
4452       }%
4453     }%
4454   }%
```

If user removes the glossaries package from their document, ensure the next run doesn't throw a load of undefined control sequence errors when the aux file is parsed.

```

4455 \AtBeginDocument
4456 {%
4457   \write\@auxout{\string\providecommand\string\@gls@reference[3]{}}%
4458 }%
```

Change warning about no glossares

```

4459 \def\warn@noprintglossary{%
4460   \GlossariesWarningNoLine{No \string\printnoidxglossary\space
4461     or \string\printnoidxglossaries ^^J
4462     found. (Remove \string\makenoidxglossaries\space if you
4463     don't want any glossaries.)^^JThis document will not have a glossary}%
4464 }%
```

Suppress warning about no \makeglossaries

```

4465 \let\warn@nomakeglossaries\relax
```

Prevent user from also using \makeglossaries

```

4466 \let\makeglossaries\@no@makeglossaries
```

Allow sort key in printgloss family:

```

4467 \renewcommand*{\@printgloss@setsort}{%
4468   \let\@glo@assign@sortkey\@glo@assign@sortkey
```

Initialise default sort order:

```

4469 \def\@glo@sorttype{\@glo@default@sorttype}%
4470 }%
```

All entries must be defined in the preamble:

```

4471 \renewcommand*\new@glossaryentry[2]{%
4472   \PackageError{glossaries}{Glossary entries must be
4473     defined in the preamble^^Jwhen you use
4474     \string\makenoidxglossaries}%
4475   {Either move your definitions to the preamble or use
4476     \string\makeglossaries}%
4477 }%
```

Redefine \glsentrynumberlist

```

4478 \renewcommand*{\glsentrynumberlist}[1]{%
4479   \letcs{\@gls@loclist}{\glo@\glsdetoklabel{##1}@loclist}%
4480   \ifdef\@gls@loclist
4481     {%
4482       \glsnoidxloclist{\@gls@loclist}%
4483     }%
4484     {%
4485       \ifglsentryexists{##1}%
4486         {%
4487           \GlossariesWarning{Missing location list for ‘##1’. Either
4488             a rerun is required or you haven't referenced the entry.}%
4489         }%
```

```

4490     {%
4491         \PackageError{glossaries}{Glossary entry ‘##1’ has not been
4492             defined.}{}%
4493     }%
4494 }%
4495 }%

Redefine \glsdisplaynumberlist
4496 \renewcommand*{\glsdisplaynumberlist}[1]{%
4497     \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
4498     \ifdef\@gls@loclist
4499     {%
4500         \def\@gls@noidxloclist@sep{%
4501             \def\@gls@noidxloclist@sep{%
4502                 \def\@gls@noidxloclist@sep{%
4503                     \glsnumlistsep
4504                 }%
4505                 \def\@gls@noidxloclist@finalsep{\glsnumlistlastsep}%
4506             }%
4507         }%
4508         \def\@gls@noidxloclist@finalsep{}%
4509         \def\@gls@noidxloclist@prev{}%
4510         \forlistloop{\glsnoidxdisplayloclisthandler}{\@gls@loclist}%
4511         \@gls@noidxloclist@finalsep
4512         \@gls@noidxloclist@prev
4513     }%
4514     {%
4515         ??\ifglsentryexists{##1}%
4516         {%
4517             \GlossariesWarning{Missing location list for ‘##1’. Either
4518                 a rerun is required or you haven’t referenced the entry.}%
4519         }%
4520         {%
4521             \PackageError{glossaries}{Glossary entry ‘##1’ has not been
4522                 defined.}{}%
4523         }%
4524     }%
4525 }%

```

Provide a generic way of iterating through the number list:

```

4526 \renewcommand*{\glsnumberlistloop}[3]{%
4527     \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
4528     \let\@gls@org@glsnoidxdisplayloc\glsnoidxdisplayloc
4529     \let\@gls@org@glsseeformat\glsseeformat
4530     \let\glsnoidxdisplayloc##2\relax
4531     \let\glsseeformat##3\relax
4532     \ifdef\@gls@loclist
4533     {%
4534         \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
4535     }%

```

```

4536   {%
4537       \ifglsentryexists{##1}%
4538       {%
4539           \GlossariesWarning{Missing location list for ‘##1’. Either
4540               a rerun is required or you haven’t referenced the entry.}%
4541       }%
4542       {%
4543           \PackageError{glossaries}{Glossary entry ‘##1’ has not been
4544               defined.}{}%
4545       }%
4546   }%
4547   \let\glsnoidxdisplayloc\@gls@org\glsnoidxdisplayloc
4548   \let\glsseeformat\@gls@org\glsseeformat
4549 }%

```

Modify sanitize sort function

```

4550 \let\@gls@sanitizesort\@gls@noidx@sanitizesort
4551 \let\@gls@nosanitizesort\@gls@noidx@nosanitizesort
4552 \@gls@noidx@setsanitizesort
4553 }

```

Preamble-only command:

```

4554 \onlypreamble{\makenoidxglossaries}

```

`\glsnumberlistloop` `\glsnumberlistloop{<label>}{<handler>}`

```

4555 \newcommand*{\glsnumberlistloop}[2]{%
4556     \PackageError{glossaries}{\string\glsnumberlistloop\space
4557         only works with \string\makenoidxglossaries}{}%
4558 }

```

`numberlistloophandler` Handler macro for `\glsnumberlistloop`. (The argument should be in the form `\glsnoidxdisplayloc{<prefix>}{<counter>}{<format>}{<n>}`)

```

4559 \newcommand*{\glsnoidxnumberlistloophandler}[1]{%
4560     #1%
4561 }

```

`\@no@makeglossaries` Can’t use both `\makeglossaries` and `\makenoidxglossaries`

```

4562 \newcommand*{\@no@makeglossaries}{%
4563     \PackageError{glossaries}{You can’t use both
4564         \string\makeglossaries\space and \string\makenoidxglossaries}%
4565     {Either use one or other (or none) of those commands but not both
4566         together.}%
4567 }

```

`\@gls@noref@warn` Warning when no instances of `\@gls@reference` found.

```

4568 \newcommand{\@gls@noref@warn}[1]{%
4569     \GlossariesWarning{\string\makenoidxglossaries\space

```

```

4570   is required to make \string\printnoidxglossary[type={#1}] work}%
4571 }

```

`\gls@noidxglossary` Write the glossary information to the aux file:

```

4572 \newcommand*{\gls@noidxglossary}{%
4573   \protected@write\@auxout{}{%
4574     \string\@gls@reference
4575     {\csname glo@\@gls@label @type\endcsname}%
4576     {\@gls@label}%
4577     {\string\glsnoidxdisplayloc
4578      {\@glo@counterprefix}%
4579      {\@gls@counter}%
4580      {\@glsnumberformat}%
4581      {\@glslocref}%
4582     }%
4583   }%
4584 }

```

1.13 Writing information to associated files

`\istfile` Deprecated.

```

4585 \def\istfile{\glswrite}

```

At the end of the document, the files should be created if `savewrites=true`.

```

4586 \AtEndDocument{%
4587   \glswritefiles
4588 }

```

`\@glswritefiles` Only write the files if `savewrites=true`

```

4589 \newcommand*{\@glswritefiles}{%

```

Iterate through all the glossaries

```

4590   \foralllglossaries{\@glo@type}{%

```

Check for empty glossaries (patch provided by Patrick Häcker)

```

4591     \ifcsundef{glo@\@glo@type @filetok}%
4592     {%
4593       \def\gls@tmp{}%
4594     }%
4595     {%
4596       \edef\gls@tmp{\expandafter\the
4597         \csname glo@\@glo@type @filetok\endcsname}%
4598     }%
4599     \ifx\gls@tmp\@empty
4600       \ifx\@glo@type\glsdefaulttype
4601         \GlossariesWarningNoLine{Glossary '\@glo@type' has no
4602           entries.^^JRemember to use package option 'nomain' if
4603 you
4604           don't want to^^Juse the main glossary}%

```

```

4605         \else
4606             \GlossariesWarningNoLine{Glossary ‘\@glo@type’ has no
4607                 entries}%
4608         \fi
4609     \else
4610         \@glsopenfile{\glswrite}{\@glo@type}%
4611         \immediate\write\glswrite{%
4612             \expandafter\the
4613             \csname glo@\@glo@type @filetok\endcsname}%
4614         \immediate\closeout\glswrite
4615     \fi
4616 }%
4617 }

```

As from v4.10, the `\glossary` command is used by the `glossaries` package. Since the user isn't expected to use this command (as `glossaries` takes care of the particular format required for `makeindex/xindy`) there's no need for a user level command. Using a custom internal command prevents any conflict with other packages (and with the `\mark` mechanism).

In v4.10, the redefinition of `\glossary` was removed since it wasn't intended as a user level command, however it seems there are packages that have hacked the internal macros used by `glossaries` and no longer work with this redefinition removed, so it's been restored in v4.11 but is not used at all by `glossaries`. (This may be removed or moved to a compatibility mode in future.)

`\glossary`

```

4618 \if@gls@docloaded
4619 \else
4620   \renewcommand*{\glossary}[1][main]{\gls@glossary{#1}}
4621 \fi

```

The associated number should be stored in `\theglsentrycounter` before using `\gls@glossary`.

`\gls@glossary`

```

4622 \newcommand*{\gls@glossary}[1]{%
4623   \@gls@glossary{#1}%
4624 }

```

`\@gls@glossary` (In v4.10, `\@glossary` was redefined to `\@gls@glossary` to avoid conflict with other packages.) Define internal `\@gls@glossary` to ignore its argument. This gets redefined in `\@makeglossary`. This is defined to just `\index` as `memoir` changes the definition of `\@index`. (Thanks to Dan Luecking for pointing this out.) The argument #1 is the glossary type.

```

4625 \newcommand*{\@gls@glossary}[1]{\index}

```

This is a convenience command to set `\@gls@glossary`. It's used by `\@makeglossary` and then redefined to do nothing, as it only needs to be done once.

\@gls@renewglossary

```

4626 \newcommand{\@gls@renewglossary}{%
4627   \gdef\@gls@glossary##1{\@bsphack\begingroup\gls@wrglossary{##1}}%
4628   \let\@gls@renewglossary\@empty
4629 }
```

The \gls@wrglossary command is defined to have two arguments. The first argument is the glossary type, the second argument is the glossary entry (the format of which is set in \glslink).

\gls@wrglossary

```

4630 \newcommand*{\gls@wrglossary}[2]{%
4631   \ifglssavewrites
4632     \protected@edef\@gls@tmp{\the\csname glo@#1@filetok\endcsname#2}%
4633     \expandafter\global\expandafter\csname glo@#1@filetok\endcsname
4634       \expandafter{\@gls@tmp^^J}%
4635   \else
4636     \ifcsdef{glo@#1@file}%
4637     {%
4638       \expandafter\protected@write\csname glo@#1@file\endcsname{%
4639         \gls@disablepagerefexpansion}{#2}%
4640     }%
4641     {%
4642       \ifignoredglossary{#1}{}%
4643       {%
4644         \GlossariesWarning{No file defined for glossary ‘#1’}%
4645       }%
4646     }%
4647   \fi
4648   \endgroup\@esphack
4649 }
```

\@do@wrglossary

```

4650 \newcommand*{\@do@wrglossary}[1]{%
4651   \ifglsindexonlyfirst
4652     \ifglsused{#1}{\@do@wrglossary{#1}}%
4653   \else
4654     \@do@wrglossary{#1}%
4655   \fi
4656 }
```

@protected@pagefmts List of page formats to be protected against expansion.

```

4657 \newcommand{\gls@protected@pagefmts}{%
4658   \gls@numberpage,\gls@alphpage,\gls@Alphpage,\gls@romanpage,\gls@Romanpage%
4659 }
```

blepagerefexpansion

```

4660 \newcommand*{\gls@disablepagerefexpansion}{%
```

```

4661 \@for\@gls@this:=\gls@protected@pagefmts\do
4662 {%
4663     \expandafter\let\@gls@this\relax
4664 }%
4665 }

```

`\gls@alphpage`

```
4666 \newcommand*\gls@alphpage{\@alph\c@page}
```

`\gls@Alphpage`

```
4667 \newcommand*\gls@Alphpage{\@Alph\c@page}
```

`\gls@numberpage`

```
4668 \newcommand*\gls@numberpage{\number\c@page}
```

`\gls@romanpage`

```
4669 \newcommand*\gls@romanpage{\romannumeral\c@page}
```

`\gls@Romanpage`

```
4670 \newcommand*\gls@Romanpage{\@Roman\c@page}
```

`\glsaddprotectedpagefmt`

```
\glsaddprotectedpagefmt{<cs name>}
```

Added a page format to the list of protected page formats. The argument should be the name (without a backslash) of the command that takes a \TeX register as the argument (`\<csname>\c@page` must be valid).

```

4671 \newcommand*\glsaddprotectedpagefmt[1]{%
4672     \eappto\gls@protected@pagefmts{\expandonce{\csname gls#1page\endcsname}}%
4673     \csedef{gls#1page}{\expandonce{\csname#1\endcsname}\noexpand\c@page}%
4674     \eappto\@wrglossarynumberhook{%
4675         \noexpand\let\expandonce{\csname org@gls#1\endcsname}%
4676         \expandonce{\csname#1\endcsname}%
4677         \noexpand\def\expandonce{\csname#1\endcsname}{%
4678             \noexpand\@wrglossary@pageformat
4679             \expandonce{\csname gls#1page\endcsname}%
4680             \expandonce{\csname org@gls#1\endcsname}%
4681         }%
4682     }%
4683 }

```

`\@wrglossarynumberhook` Hook used by `\@do@wrglossary`

```
4684 \newcommand*\@wrglossarynumberhook{}
```

`\@wrglossary@pageformat`

```

4685 \newcommand{\@wrglossary@pageformat}[3]{%
4686     \ifx#3\c@page #1\else #2#3\fi
4687 }

```

`\@do@wrglossary` Write the glossary entry in the appropriate format. (Need to set `\@glsnumberformat` and `\@gls@counter` prior to use.) The argument is the entry's label.

```
4688 \newcommand*{\@do@wrglossary}[1]{%
4689   \begingroup
```

First a bit of hackery to prevent premature expansion of `\c@page`. Store original definitions:

```
4690   \let\orgthe\the
4691   \let\orgnumber\number
4692   \let\orgromannumeral\romannumeral
4693   \let\orgalph\@alph
4694   \let\orgAlph\@Alph
4695   \let\orgRoman\@Roman
```

Redefine:

```
4696   \def\the##1{%
4697     \ifx##1\c@page \gls@numberpage\else\orgthe##1\fi}%
4698   \def\number##1{%
4699     \ifx##1\c@page \gls@numberpage\else\orgnumber##1\fi}%
4700   \def\romannumeral##1{%
4701     \ifx##1\c@page \gls@romanpage\else\orgromannumeral##1\fi}%
4702   \def\@Roman##1{%
4703     \ifx##1\c@page \gls@Romanpage\else\orgRoman##1\fi}%
4704   \def\@alph##1{%
4705     \ifx##1\c@page \gls@alphpage\else\orgalph##1\fi}%
4706   \def\@Alph##1{%
4707     \ifx##1\c@page \gls@Alphpage\else\orgAlph##1\fi}%
```

Add hook to allow for other number formats:

```
4708   \@wrglossarynumberhook
```

Prevent expansion:

```
4709   \gls@disablepagerefexpansion
```

Now store location in `\@glslocref`:

```
4710   \protected\xdef\@glslocref{\theHglentrycounter}%
4711   \endgroup
```

Escape any special characters

```
4712   \@gls@checkmkidxchars\@glslocref
```

Check if the hyper-location is the same as the location and set the hyper prefix.

```
4713   \expandafter\ifx\theHglentrycounter\theHglentrycounter\relax
4714   \def\@glo@counterprefix{%
4715   \else
4716     \protected\edef\@glsHlocref{\theHglentrycounter}%
4717     \@gls@checkmkidxchars\@glsHlocref
4718     \edef\@do@gls@getcounterprefix{\noexpand\@gls@getcounterprefix
4719       {\@glslocref}{\@glsHlocref}%
4720     }%
```

```

4721 \do@glsglscounterprefix
4722 \fi

De-tok label if required
4723 \edef\@glsglscounterlabel{\glsglscounterlabel{#1}}%

Write the information to file:
4724 \do@do@wrglossary
4725 }

```

\do@do@wrglossary

```

4726 \newcommand*\do@do@wrglossary{%

Determine whether to use xindy or makeindex syntax
4727 \ifglsglscounterxindy

Need to determine if the formatting information starts with a ( or ) indicating a
range.
4728 \expandafter\@glsglscountercheck@mkidxrangearchar\@glsglscounterformat\@nil
4729 \def\@glsglscounter@range{%
4730 \expandafter\if\@glsglscounter@prefix(\relax
4731 \def\@glsglscounter@range{:open-range}%
4732 \else
4733 \expandafter\if\@glsglscounter@prefix)\relax
4734 \def\@glsglscounter@range{:close-range}%
4735 \fi
4736 \fi

Write to the glossary file using xindy syntax.
4737 \glsglscounter{csname gls@glsglscounterlabel @type\endcsname}{%
4738 (indexentry :tkey (\csname gls@glsglscounterlabel @index\endcsname)

4739 :locoref \string"\@glsglscounter@counterprefix}\@glsglscounterlocoref}\string" %
4740 :attr \string"\@glsglscounter\@glsglscounter@suffix\string"
4741 \@glsglscounter@range
4742 )
4743 }%
4744 \else

```

Convert the format information into the format required for makeindex

```

4745 \set@glsglscounter@numformat{\@glsglscounter@numfmt}\@glsglscounter\@glsglscounterformat}%
4746 {\@glsglscounter@counterprefix}%

```

Write to the glossary file using makeindex syntax.

```

4747 \glsglscounter{csname gls@glsglscounterlabel @type\endcsname}{%
4748 \string\glossaryentry{\csname gls@glsglscounterlabel @index\endcsname
4749 \@glsglscounter@encapchar\@glsglscounter@numfmt}\@glsglscounterlocoref}}%
4750 \fi
4751 }

```

ls@glsglscounterprefix Get the prefix that needs to be prepended to counter in order to get the hyper counter. (For example, with the standard article class and hyperref,

\theequation needs to be prefixed with `\section num` to get the equivalent \theHequation.) NB this assumes that the prefix ends with a dot, which is the standard. (Otherwise it makes the xindy location classes more complicated.)

```

4752 \newcommand*\@gls@getcounterprefix[2]{%
4753   \edef\@gls@thisloc{#1}\edef\@gls@thisHloc{#2}%
4754   \ifx\@gls@thisloc\@gls@thisHloc
4755     \def\@glo@counterprefix{}%
4756   \else
4757     \def\@gls@get@counterprefix##1.#1##2\end@getprefix{%
4758       \def\@glo@tmp{##2}%
4759       \ifx\@glo@tmp\@empty
4760         \def\@glo@counterprefix{}%
4761       \else
4762         \def\@glo@counterprefix{##1}%
4763       \fi
4764     }%
4765     \@gls@get@counterprefix#2.#1\end@getprefix

Warn if no prefix can be formed.

4766   \ifx\@glo@counterprefix\@empty
4767     \GlossariesWarning{Hyper target ‘#2’ can’t be formed by
4768       prefixing^^Jlocation ‘#1’. You need to modify the
4769       definition of \string\theH\@gls@counter^^Jotherwise you
4770       will get the warning: “‘name{\@gls@counter.#1}’ has been^^J
4771       referenced but does not exist”}%
4772   \fi
4773 \fi
4774 }
```

1.14 Glossary Entry Cross-References

`\do@seeglossary` Write the glossary entry with a cross reference. The first argument is the entry's label, the second must be in the form `[\langle tag \rangle]{\langle list \rangle}`, where `\langle tag \rangle` is a tag such as “see” and `\langle list \rangle` is a list of labels.

```

4775 \newcommand{\@do@seeglossary}[2]{%
4776 \def\@gls@xref{#2}%
4777 \@onelevel@sanitize\@gls@xref
4778 \@gls@checkmkidxchars\@gls@xref
4779 \ifglsxindy
4780   \gls@glossary{\csname glo@#1@type\endcsname}{%
4781     (indexentry
4782       :key (\csname glo@#1@index\endcsname)
4783       :xref (\string"\@gls@xref\string")
4784       :attr \string"see\string"
4785     )
4786   }%
4787 \else
4788   \gls@glossary{\csname glo@#1@type\endcsname}{%
4789     \string\glossaryentry{\csname glo@#1@index\endcsname

```

```

4790 \@gls@encapchar glsseeformat\@gls@xref}{Z}}%
4791 \fi
4792 }

```

`\@gls@fixbraces` If no optional argument is specified, list needs to be enclosed in a set of braces.

```

4793 \def\@gls@fixbraces#1#2#3\@nil{%
4794   \ifx#2[\relax
4795     \@gls@fixbraces#1#2#3\@end@fixbraces
4796   \else
4797     \def#1{{#2#3}}%
4798   \fi
4799 }

```

`\@gls@fixbraces`

```

4800 \def\@gls@fixbraces#1[#2]#3\@end@fixbraces{%
4801   \def#1{[#2]{#3}}%
4802 }

```

`\glssee` `\glssee{<label>}{<cross-ref list>}`

```

4803 \DeclareRobustCommand*\glssee[3][\seename]{%
4804   \@do@seeglossary{#2}{[#1]{#3}}}
4805 \newcommand*\@glssee[3][\seename]{%
4806   \glssee[#1]{#3}{#2}}

```

`\glsseeformat` The first argument specifies what tag to use (e.g. “see”), the second argument is a comma-separated list of labels. The final argument (the location) is ignored.

```

4807 \DeclareRobustCommand*\glsseeformat[3][\seename]{%
4808   \emph{#1} \glsseelist{#2}}

```

`\glsseelist` `\glsseelist{<list>}` formats list of entry labels.

```

4809 \DeclareRobustCommand*\glsseelist[1]{%

```

If there is only one item in the list, set the last separator to do nothing.

```

4810 \let\@gls@dolast\relax

```

Don’t display separator on the first iteration of the loop

```

4811 \let\@gls@donext\relax

```

Iterate through the labels

```

4812 \@for\@gls@thislabel:=#1\do{%

```

Check if on last iteration of loop

```

4813   \ifx\@xfor@nextelement\@nnil

```

```

4814     \@gls@dolast

```

```

4815   \else

```

```

4816     \@gls@donext

```

```

4817   \fi

```

Display the entry for this label. (Expanding label as it’s a temporary control sequence that’s used elsewhere.)

```

4818   \expandafter\glsseeitem\expandafter{\@gls@thislabel}%

```

Update separators

```
4819 \let\@gls@dolast\glsseelastsep
4820 \let\@gls@donext\glsseesep
4821 }%
4822 }
```

`\glsseelastsep` Separator to use between penultimate and ultimate entries in a cross-referencing list.

```
4823 \newcommand*{\glsseelastsep}{\space\andname\space}
```

`\glsseesep` Separator to use between entires in a cross-referencing list.

```
4824 \newcommand*{\glsseesep}{, }
```

`\glsseeitem` `\glsseeitem{<label>}` formats individual entry in a cross-referencing list.

```
4825 \DeclareRobustCommand*{\glsseeitem}[1]{\gls hyperlink[\glsseeitemformat{#1}]{#1}}
```

`\glsseeitemformat` As from v3.0, default is to use `\glsentrytext` instead of `\glsentryname`. (To avoid problems with the name key being sanitized.)

```
4826 \newcommand*{\glsseeitemformat}[1]{\glsentrytext{#1}}
```

1.15 Displaying the glossary

An individual glossary is displayed in the text using `\printglossary[<key-val list>]`. If the type key is omitted, the default glossary is displayed. The optional argument can be used to specify an alternative glossary, and can also be used to set the style, title and entry in the table of contents. Available keys are defined below.

`\gls@save@numberlist` Provide command to store number list.

```
4827 \newcommand*{\gls@save@numberlist}[1]{%
4828   \ifglssavenumberlist
4829     \toks@{#1}%
4830     \edef\@do@writeaux@info{%
4831       \noexpand\csgdef{glo@\glscurrententrylabel @numberlist}{\the\toks@}%
4832     }%
4833     \@onelevel@sanitize\@do@writeaux@info
4834     \protected@write\@auxout{}\@do@writeaux@info}%
4835   \fi
4836 }
```

`\warn@noprintglossary` Warn the user if they have forgotten `\printglossaries` or `\printglossary`. (Will be suppressed if there is at least one occurrence of `\printglossary`. There is no check to ensure that there is a `\printglossary` for each defined glossary.)

```
4837 \newcommand*{\warn@noprintglossary}{}%
```

`\printglossary` The TOC title needs to be processed in a different manner to the main title in case the translator and hyperref packages are both being used.

```
4838 \ifcsundef{printglossary}{}%  
4839 {%
```

If `\printglossary` is already defined, issue a warning and undefine it.

```
4840 \@gls@warnonglossdefined  
4841 \undef\printglossary  
4842 }
```

`\printglossary` has an optional argument. The default value is to set the glossary type to the main glossary.

```
4843 \newcommand*{\printglossary}[1][type=\glsdefaulttype]{%  
4844 \@printglossary{#1}{\@print@glossary}%  
4845 }
```

The `\printglossaries` command will do `\printglossary` for each glossary type that has been defined. It is better to use `\printglossaries` rather than individual `\printglossary` commands to ensure that you don't forget any new glossaries you may have created. It also makes it easier to chop and change the value of the acronym package option. However, if you want to list the glossaries in a different order, or if you want to set the title or table of contents entry, or if you want to use different glossary styles for each glossary, you will need to use `\printglossary` explicitly for each glossary type.

`\printglossaries`

```
4846 \newcommand*{\printglossaries}{%  
4847 \forallglossaries{\@glo@type}{\printglossary[type=\@glo@type]}}%  
4848 }
```

`\printnoidxglossary` Provide an alternative to `\printglossary` that doesn't require an external indexing application. Entries won't be sorted and the location list will be empty.

```
4849 \newcommand*{\printnoidxglossary}[1][type=\glsdefaulttype]{%  
4850 \@printglossary{#1}{\@print@noidx@glossary}%  
4851 }
```

`\printnoidxglossaries` Analogous to `\printglossaries`

```
4852 \newcommand*{\printnoidxglossaries}{%  
4853 \forallglossaries{\@glo@type}{\printnoidxglossary[type=\@glo@type]}}%  
4854 }
```

`@printgloss@setsort` Initialise to do nothing.

```
4855 \newcommand*{\@printgloss@setsort}{}
```

`\@printglossary` Sets up the glossary for either `\printglossary` or `\printnoidxglossary`. The first argument is the options list, the second argument is the handler macro that deals with the actual glossary.

```
4856 \newcommand{\@printglossary}[2]{%
```


Set up defaults.

```
4857 \def\@glo@type{\glsdefaulttype}%
4858 \def\glossarytitle{\csname @glo@type\@glo@type @title\endcsname}%

4859 \def\glossarytoctitle{\glossarytitle}%
4860 \let\org@glossarytitle\glossarytitle
4861 \def\@glossarystyle{}%
4862 \def\gls@dotoc@title{\gls@dotoc@title{\@glo@type}}%
```

Store current value of `\glossaryentrynumbers`. (This may be changed via the optional argument)

```
4863 \let\@org@glossaryentrynumbers\glossaryentrynumbers
```

Localise the effects of the optional argument

```
4864 \bgroup
```

Activate or deactivate sort key:

```
4865 \@printgloss@setsort
```

Determine settings specified in the optional argument.

```
4866 \setkeys{printgloss}{#1}%
```

If title has been set, but toctitle hasn't, make toctitle the same as given title (rather than the title used when the glossary was defined)

```
4867 \ifx\glossarytitle\org@glossarytitle
4868 \else
4869 \expandafter\let\csname @glo@type\@glo@type @title\endcsname
4870 \glossarytitle
4871 \fi
```

Allow a high-level user command to indicate the current glossary

```
4872 \let\currentglossary\@glo@type
```

Enable individual number lists to be suppressed.

```
4873 \let\org@glossaryentrynumbers\glossaryentrynumbers
4874 \let\glsnonextpages\@glsnonextpages
```

Enable individual number list to be activated:

```
4875 \let\glsnextpages\@glsnextpages
```

Enable suppression of description terminators.

```
4876 \let\nopostdesc\@nopostdesc
```

Set up the entry for the TOC

```
4877 \gls@dotoc@title
```

Set the glossary style

```
4878 \@glossarystyle
```

Added a way to fetch the current entry label (v3.08 updated for new `\glossentry` and `\subglossentry`, but this is now only needed for backward compatibility):

```
4879 \let\gls@org@glossaryentryfield\glossentry
```

```

4880 \let\gls@org@glossarysubentryfield\subglossentry
4881 \renewcommand{\glossentry}[1]{%
4882   \xdef\glscurrententrylabel{\glsdetoklabel{##1}}%
4883   \gls@org@glossaryentryfield{##1}%
4884 }%
4885 \renewcommand{\subglossentry}[2]{%
4886   \xdef\glscurrententrylabel{\glsdetoklabel{##2}}%
4887   \gls@org@glossarysubentryfield{##1}{##2}%
4888 }%

```

Now do the handler macro that deals with the actual glossary:

```

4889   #2%

End the current scope
4890 \egroup

Reset \glossaryentrynumbers
4891 \global\let\glossaryentrynumbers\@org@glossaryentrynumbers

Suppress warning about no \printglossary
4892 \global\let\warn@noprintglossary\relax
4893 }

```

\@print@glossary Internal workings of \printglossary dealing with reading the external file.

```

4894 \newcommand{\@print@glossary}{%

```

Some macros may end up being expanded into internals in the glossary, so need to make @ a letter. (Unlikely to be a problem since v3.08a but kept for backward compatibility.)

```

4895 \makeatletter

```

Input the glossary file, if it exists.

```

4896 \@input{\jobname.\csname @glotype@\@glo@type @in\endcsname}%

```

If the glossary file doesn't exist, do \null. (This ensures that the page is shipped out and all write commands are done.) This might produce an empty page, but at this point the document isn't complete, so it shouldn't matter.

```

4897 \IfFileExists{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
4898 {}%
4899 {\null}%

```

If xindy is being used, need to write the language dependent information to the .aux file for makeglossaries.

```

4900 \ifglxindy
4901   \ifcsundef{@xdy@\@glo@type @language}%
4902   {%
4903     \edef\do@auxoutstuff{%
4904       \noexpand\AtEndDocument{%

```

If the user removes the glossary package from their document, ensure the next run doesn't throw a load of undefined control sequence errors when the aux file is parsed.

```

4905         \noexpand\immediate\noexpand\write\@auxout{%
4906         \string\providecommand\string\@xdylanguage[2]{}%
4907         \noexpand\immediate\noexpand\write\@auxout{%
4908         \string\@xdylanguage{\@glo@type}{\@xdy@main@language}}%
4909     }%
4910 }%
4911 }%
4912 {%
4913     \edef\@do@auxoutstuff{%
4914     \noexpand\AtEndDocument{%
4915     \noexpand\immediate\noexpand\write\@auxout{%
4916     \string\providecommand\string\@xdylanguage[2]{}%
4917     \noexpand\immediate\noexpand\write\@auxout{%
4918     \string\@xdylanguage{\@glo@type}{\csname @xdy@\@glo@type
4919     @language\endcsname}}%
4920     }%
4921     }%
4922     }%
4923     \@do@auxoutstuff
4924     \edef\@do@auxoutstuff{%
4925     \noexpand\AtEndDocument{%

```

If the user removes the glossaries package from their document, ensure the next run doesn't throw a load of undefined control sequence errors when the aux file is parsed.

```

4926     \noexpand\immediate\noexpand\write\@auxout{%
4927     \string\providecommand\string\@gls@codepage[2]{}%
4928     \noexpand\immediate\noexpand\write\@auxout{%
4929     \string\@gls@codepage{\@glo@type}{\gls@codepage}}%
4930     }%
4931     }%
4932     \@do@auxoutstuff
4933 \fi

```

Activate warning if \makeglossaries hasn't been used.

```

4934 \renewcommand*{\@warn@nomakeglossaries}{%
4935 \GlossariesWarningNoLine{\string\makeglossaries\space
4936 hasn't been used,^^Jthe glossaries will not be updated}%
4937 }%
4938 }

```

The sort macros all have the syntax:

`\@glo@sortmacro@<order>{<type>}`

where <order> is the sort order as specified by the sort key and <type> is the glossary type. (The referenced entry list is stored in \@glsref@<type>). The actual sorting is done by \@glo@sortentries{<handler>}{<type>}.

\@glo@sortentries

```
4939 \newcommand*{\@glo@sortentries}[2]{%
4940   \def\@glo@sortinglist{}%
4941   \def\@glo@sortinghandler{#1}%
4942   \edef\@glo@type{#2}%
4943   \forlistcsloop{\@glo@do@sortentries}{\@glsref@#2}%
4944   \csdef{\@glsref@#2}{}%
4945   \@for\@this@label:=\@glo@sortinglist\do{%
      Has this entry already been added?
4946     \xifinlistcs{\@this@label}{\@glsref@#2}%
4947     {}%
4948     {%
4949       \listcsxadd{\@glsref@#2}{\@this@label}%
4950     }%
4951     \ifcsdef{\@glo@sortingchildren@\@this@label}%
4952     {%
4953       \@glo@addchildren{#2}{\@this@label}%
4954     }%
4955     {}%
4956   }%
4957 }
```

\@glo@addchildren

\@glo@addchildren{<type>}{<parent>}

```
4958 \newcommand*{\@glo@addchildren}[2]{%
      Scope to allow nesting.
4959   \bgroup
4960   \letcs{\@glo@childlist}{\@glo@sortingchildren@#2}%
4961   \@for\@this@childlabel:=\@glo@childlist\do
4962   {%
      Check this label hasn't already been added.
4963     \xifinlistcs{\@this@childlabel}{\@glsref@#1}%
4964     {}%
4965     {%
4966       \listcsxadd{\@glsref@#1}{\@this@childlabel}%
4967     }%
      Does this child have children?
4968     \ifcsdef{\@glo@sortingchildren@\@this@childlabel}%
4969     {%
4970       \@glo@addchildren{#1}{\@this@childlabel}%
4971     }%
4972     {%
4973     }%
4974   }%
4975   \egroup
4976 }
```

@glo@do@sortentries

```
4977 \newcommand*{\@glo@do@sortentries}[1]{%
4978   \ifglshasparent{#1}%
4979   {%
```

This entry has a parent, so add it to the child list

```
4980   \edef\@glo@parent{\csuse{glo@glstetoklabel{#1}@parent}}%
4981   \ifcsundef{glo@sortingchildren@\@glo@parent}%
4982   {%
4983     \csdef{glo@sortingchildren@\@glo@parent}{}%
4984   }%
4985   {}%
4986   \expandafter\@glo@sortedinsert
4987     \csname glo@sortingchildren@\@glo@parent\endcsname{#1}%
```

Has the parent been added?

```
4988   \xifinlistcs{\@glo@parent}{glsref@\@glo@type}%
4989   {%
```

Yes, it has so do nothing.

```
4990   }%
4991   {%
```

No, it hasn't so add it now.

```
4992     \expandafter\@glo@do@sortentries\expandafter{\@glo@parent}%
4993   }%
4994   }%
4995   {%
4996     \@glo@sortedinsert{\@glo@sortinglist}{#1}%
4997   }%
4998 }
```

\@glo@sortedinsert `\@glo@sortedinsert{<list>}{<entry label>}`

Insert into list.

```
4999 \newcommand*{\@glo@sortedinsert}[2]{%
5000   \dtl@insertinto{#2}{#1}{\@glo@sortinghandler}%
5001 }
```

The sort handlers need to be in the form required by datatool's \dtl@sortlist macro. These must set the count register \dtl@sortresult to either -1 (#1 less than #2), 0 (#1 = #2) or +1 (#1 greater than #2).

lo@sorthandler@word

```
5002 \newcommand*{\@glo@sorthandler@word}[2]{%
5003   \letcs\@gls@sortA{glo@glstetoklabel{#1}@sort}%
5004   \letcs\@gls@sortB{glo@glstetoklabel{#2}@sort}%
5005   \edef\@glo@do@compare{%
5006     \noexpand\dtlwordindexcompare{\noexpand\dtl@sortresult}%
```

```

5007     {\expandonce\@gls@sort@B}%
5008     {\expandonce\@gls@sort@A}%
5009 }%
5010 \glo@do@compare
5011 }

```

@sorthandler@letter

```

5012 \newcommand*{\@glo@sorthandler@letter}[2]{%
5013   \letcs\@gls@sort@A{glo\@glsdetoklabel{#1}@sort}%
5014   \letcs\@gls@sort@B{glo\@glsdetoklabel{#2}@sort}%
5015   \edef\glo@do@compare{%
5016     \noexpand\dtlletterindexcompare{\noexpand\dtl@sortresult}%
5017     {\expandonce\@gls@sort@B}%
5018     {\expandonce\@gls@sort@A}%
5019   }%
5020   \glo@do@compare
5021 }

```

lo@sorthandler@case Case-sensitive sort.

```

5022 \newcommand*{\@glo@sorthandler@case}[2]{%
5023   \letcs\@gls@sort@A{glo\@glsdetoklabel{#1}@sort}%
5024   \letcs\@gls@sort@B{glo\@glsdetoklabel{#2}@sort}%
5025   \edef\glo@do@compare{%
5026     \noexpand\dtlcompare{\noexpand\dtl@sortresult}%
5027     {\expandonce\@gls@sort@B}%
5028     {\expandonce\@gls@sort@A}%
5029   }%
5030   \glo@do@compare
5031 }

```

@sorthandler@nocase Case-insensitive sort.

```

5032 \newcommand*{\@glo@sorthandler@nocase}[2]{%
5033   \letcs\@gls@sort@A{glo\@glsdetoklabel{#1}@sort}%
5034   \letcs\@gls@sort@B{glo\@glsdetoklabel{#2}@sort}%
5035   \edef\glo@do@compare{%
5036     \noexpand\dtlicompare{\noexpand\dtl@sortresult}%
5037     {\expandonce\@gls@sort@B}%
5038     {\expandonce\@gls@sort@A}%
5039   }%
5040   \glo@do@compare
5041 }

```

@glo@sortmacro@word Sort macro for ‘word’

```

5042 \newcommand*{\@glo@sortmacro@word}[1]{%
5043   \ifdefstring{\@glo@default@sorttype}{standard}%
5044   {%
5045     \@glo@sortentries{\@glo@sorthandler@word}{#1}%
5046   }%
5047   {%

```

```

5048 \PackageError{glossaries}{Conflicting sort options:^^J
5049 \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5050 \string\printnoidxglossary[sort=word]}{}%
5051 }%
5052 }

\lo@sortmacro@letter Sort macro for 'letter'

5053 \newcommand*\@glo@sortmacro@letter[1]{%
5054 \ifdefstring{\@glo@default@sorttype}{standard}%
5055 {%
5056 \@glo@sortentries{\@glo@sorthandler@letter}{#1}%
5057 }%
5058 {%
5059 \PackageError{glossaries}{Conflicting sort options:^^J
5060 \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5061 \string\printnoidxglossary[sort=letter]}{}%
5062 }%
5063 }

\sortmacro@standard Sort macro for 'standard'. (Use either 'word' or 'letter' order.)

5064 \newcommand*\@glo@sortmacro@standard[1]{%
5065 \ifdefstring{\@glo@default@sorttype}{standard}%
5066 {%
5067 \ifcsdef{@glo@sorthandler@\glsorder}%
5068 {%
5069 \@glo@sortentries{\csuse{@glo@sorthandler@\glsorder}}{#1}%
5070 }%
5071 {%
5072 \PackageError{glossaries}{Unknown sort handler '@glsorder'}{}%
5073 }%
5074 }%
5075 {%
5076 \PackageError{glossaries}{Conflicting sort options:^^J
5077 \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5078 \string\printnoidxglossary[sort=standard]}{}%
5079 }%
5080 }

\glo@sortmacro@case Sort macro for 'case'

5081 \newcommand*\@glo@sortmacro@case[1]{%
5082 \ifdefstring{\@glo@default@sorttype}{standard}%
5083 {%
5084 \@glo@sortentries{\@glo@sorthandler@case}{#1}%
5085 }%
5086 {%
5087 \PackageError{glossaries}{Conflicting sort options:^^J
5088 \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5089 \string\printnoidxglossary[sort=case]}{}%
5090 }%

```

```

5091 }

\lo@sortmacro@nocase Sort macro for ‘nocase’

5092 \newcommand*{\@glo@sortmacro@nocase}[1]{%
5093   \ifdefstring{\@glo@default@sorttype}{standard}%
5094   {%
5095     \@glo@sortentries{\@glo@sorthandler@nocase}{#1}%
5096   }%
5097   {%
5098     \PackageError{glossaries}{Conflicting sort options:^^J
5099       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5100       \string\printnoidxglossary[sort=nocase]}{}%
5101   }%
5102 }

\@glo@sortmacro@def Sort macro for ‘def’. The order of definition is given in \glo@list@<type>.

5103 \newcommand*{\@glo@sortmacro@def}[1]{%
5104   \def\@glo@sortinglist{}%
5105   \for@gl@sentries[#1]{\@gls@thislabel}%
5106   {%
5107     \xifinlistcs{\@gls@thislabel}{\@glsref@#1}%
5108     {%
5109       \listcsadd{\@glo@sortinglist}{\@gls@thislabel}%
5110     }%
5111     {%
5112       }%
5113     }%
5114     \cslet{\@glsref@#1}{\@glo@sortinglist}%
5115   }

\lo@sortmacro@def@do This won’t include parent entries that haven’t been referenced.

5116 \newcommand*{\@glo@sortmacro@def@do}[1]{%
5117   \ifinlistcs{#1}{\@glsref@\@glo@type}%
5118   {}%
5119   {%
5120     \listcsadd{\@glsref@\@glo@type}{#1}%
5121   }%
5122   \ifcsdef{\@glo@sortingchildren@#1}%
5123   {%
5124     \@glo@addchildren{\@glo@type}{#1}%
5125   }%
5126   {}%
5127 }

\@glo@sortmacro@use Sort macro for ‘use’. (No sorting is required, as the entries are already in order
of use, so do nothing.)

5128 \newcommand*{\@glo@sortmacro@use}[1]{%

```


`\print@noidx@glossary` Glossary handler for `\printnoidxglossary` which doesn't use an indexing application. Since `\printnoidxglossary` may occur at the start of the document, we can't just check if an entry has been used. Instead, the first pass needs to write information to the aux file every time an entry is referenced. This needs to be read in on the second run and stored in a list corresponding to the appropriate glossary.

```

5129 \newcommand*{\@print@noidx@glossary}{%
5130   \ifcsdef{@glsref@{\glo@type}}%
5131   {%
      Sort the entries:
5132     \ifcsdef{@glo@sortmacro@{\glo@sorttype}}%
5133     {%
5134       \csuse{@glo@sortmacro@{\glo@sorttype}}{\@glo@type}%
5135     }%
5136     {%
5137       \PackageError{glossaries}{Unknown sort handler ‘\@glo@sorttype’}{}%
5138     }%

```

Do the glossary heading and preamble

```

5139   \glossarysection[\glossarytoctitle]{\glossarytitle}%
5140   \glossarypreamble
5141   \begin{theglossary}%
5142   \glossaryheader
5143   \glsresetentrylist
5144   \def\@gls@currentlettergroup{}%

```

Iterate through the entries.

```

5145   \forlistcsloop{\@gls@noidx@do}{@glsref@{\glo@type}}%

```

Finally end the glossary and do the postamble:

```

5146   \end{theglossary}%
5147   \glossarypostamble
5148 }%
5149 {%
5150   \@gls@noref@warn{\@glo@type}%
5151 }%
5152 }

```

`\glo@grabfirst`

```

5153 \def\glo@grabfirst#1#2\@nil{%
5154   \def\@gls@firsttok{#1}%
5155   \ifdefempty\@gls@firsttok
5156   {%
5157     \def\@glo@thislettergrp{0}%
5158   }%
5159   {%

```

Sanitize it:

```

5160   \@onelevel@sanitize\@gls@firsttok

```

Fetch the first letter:

```

5161 \expandafter\@glo@grabfirst\@gls@firsttok{}\}\@nil
5162 }%
5163 }

```

\@glo@grabfirst

```

5164 \def\@glo@grabfirst#1#2\@nil{%
5165 \ifdefempty\@glo@thislettergrp
5166 {%
5167 \def\@glo@thislettergrp{glssymbols}%
5168 }%
5169 {%
5170 \count@=\uccode'#1\relax
5171 \ifnum\count@=0\relax
5172 \def\@glo@thislettergrp{glssymbols}%
5173 \else
5174 \ifdefstring\@glo@sorttype{case}%
5175 {%
5176 \count@='#1\relax
5177 }%
5178 {%
5179 }%
5180 \edef\@glo@thislettergrp{\the\count@}%
5181 \fi
5182 }%
5183 }

```

\@gls@noidx@do Handler for list iteration used by \@print@noidx@glossary. The argument is the entry label. This only allows one sublevel.

```

5184 \newcommand{\@gls@noidx@do}[1]{%

```

Get this entry's location list

```

5185 \global\letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%

```

Does this entry have a parent?

```

5186 \ifglshasparent{#1}%
5187 {%

```

Has a parent.

```

5188 \gls@level=\csuse{glo@\glsdetoklabel{#1}@level}\relax
5189 \ifdefvoid{\@gls@loclist}
5190 {%
5191 \subglossentry{\gls@level}{#1}{}%
5192 }%
5193 {%
5194 \subglossentry{\gls@level}{#1}%
5195 {%
5196 \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
5197 }%
5198 }%

```

```

5199 }%
5200 {%
    Doesn't have a parent Get this entry's sort key
5201     \letcs{\@gls@sort}{glo@glsdetoklabel{#1}@sort}%
    Fetch the first letter:
5202     \expandafter\glo@grabfirst\@gls@sort{}{}\@nil
5203     \ifdefequal{\@glo@thislettergrp}{\@gls@currentlettergroup}%
5204     {}%
5205     {%
        Do the group header:
5206         \ifdefempty{\@gls@currentlettergroup}{\@gls@groupskip}%
5207         \gls@groupheading{\@glo@thislettergrp}%
5208     }%
5209     \let\@gls@currentlettergroup\@glo@thislettergrp
        Do this entry:
5210     \ifdefvoid{\@gls@loclist}
5211     {%
5212         \glossentry{#1}{}%
5213     }%
5214     {%
5215         \glossentry{#1}%
5216     }%
5217     \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
5218     }%
5219     }%
5220 }%
5221 }

```

`\glsnoidxloclist` `\glsnoidxloclist{<list cs>}`

Display location list.

```

5222 \newcommand*{\glsnoidxloclist}[1]{%
5223     \def\@gls@noidxloclist@sep{}%
5224     \def\@gls@noidxloclist@prev{}%
5225     \forlistloop{\glsnoidxloclisthandler}{#1}%
5226 }

```

`noidxloclisthandler` Handler for location list iterator.

```

5227 \newcommand*{\glsnoidxloclisthandler}[1]{%
5228     \ifdefstring{\@gls@noidxloclist@prev}{#1}%
5229     {%
        Same as previous location so skip.
5230     }%
5231     {%
5232         \@gls@noidxloclist@sep

```

```

5233    #1%
5234    \def\@gls@noidxloclist@sep{\delimN}%
5235    \def\@gls@noidxloclist@prev{#1}%
5236  }%
5237 }

```

`\displayloclisthandler` Handler for location list iterator when used with `\glsdisplaynumberlist`.

```

5238 \newcommand*\@glsnoidxdisplayloclisthandler[1]{%
5239   \ifdefstring{\@gls@noidxloclist@prev}{#1}%
5240   {%
    Same as previous location so skip.
5241   }%
5242   {%
5243     \@gls@noidxloclist@sep
5244     \@gls@noidxloclist@prev
5245     \def\@gls@noidxloclist@prev{#1}%
5246   }%
5247 }

```

`\glsnoidxdisplayloc` `\glsnoidxdisplayloc{<prefix>}{<counter>}{<format>}{<location>}`

Display a location in the location list.

```

5248 \newcommand*\glsnoidxdisplayloc[4]{%
5249   \setentrycounter[#1]{#2}%
5250   \csuse{#3}{#4}%
5251 }

```

`\@gls@reference` `\@gls@reference{<type>}{<label>}{<loc>}`

Identifies that a reference has been used (for use in the aux file). All entries must be defined in the preamble.

```

5252 \newcommand*\@gls@reference[3]{%

```

Add to label list

```

5253   \glsdoifexistsorwarn{#2}%
5254   {%
5255     \ifcsundef{\@glsref@#1}{\csgdef{\@glsref@#1}{}}{}%
5256     \ifinlistcs{#2}{\@glsref@#1}%
5257     {}%
5258     {\listcsgadd{\@glsref@#1}{#2}}%

```

Add to location list

```

5259   \ifcsundef{glo@\glsdetoklabel{#2}@loclist}%
5260   {\csgdef{glo@\glsdetoklabel{#2}@loclist}{}}{}%
5261   {}%
5262   \listcsgadd{glo@\glsdetoklabel{#2}@loclist}{#3}%

```

```

5263 }%
5264 }

```

The keys that can be used in the optional argument to `\printglossary` or `\printnoidxglossary` are as follows: The `type` key sets the glossary type.

```

5265 \define@key{printgloss}{type}{\def\@glo@type{#1}}

```

The `title` key sets the title used in the glossary section header. This overrides the title used in `\newglossary`.

```

5266 \define@key{printgloss}{title}{%
5267 \def\glossarytitle{#1}%
5268 \let\gls@dotoc@title\relax
5269 }

```

The `toctitle` sets the text used for the relevant entry in the table of contents.

```

5270 \define@key{printgloss}{toctitle}{%
5271 \def\glossarytoctitle{#1}%
5272 \let\gls@dotoc@title\relax
5273 }

```

The `style` key sets the glossary style (but only for the given glossary).

```

5274 \define@key{printgloss}{style}{%
5275 \ifcsundef{\glsstyle@#1}%
5276 {%
5277 \PackageError{glossaries}%
5278 {Glossary style ‘#1’ undefined}{}%
5279 }%
5280 {%
5281 \def\@glossarystyle{\setglossentrycompatibility
5282 \csname \glsstyle@#1\endcsname}%
5283 }%
5284 }

```

The `numberedsection` key determines if this glossary should be in a numbered section.

```

5285 \define@choicekey{printgloss}{numberedsection}[\val\nr]{%
5286 false,nolabel,autolabel,nameref}[nolabel]{%
5287 \ifcase\nr\relax
5288 \renewcommand*{\@glossarysecstar}{*}%
5289 \renewcommand*{\@glossaryseclabel}{}%
5290 \or
5291 \renewcommand*{\@glossarysecstar}{}%
5292 \renewcommand*{\@glossaryseclabel}{}%
5293 \or
5294 \renewcommand*{\@glossarysecstar}{}%
5295 \renewcommand*{\@glossaryseclabel}{\label{\glsautoprefix\@glo@type}}%
5296 \or
5297 \renewcommand*{\@glossarysecstar}{*}%
5298 \renewcommand*{\@glossaryseclabel}{%
5299 \protected@edef\@currentlabelname{\glossarytoctitle}%

```

```

5300     \label{\glsautoprefix\@glo@type}}}%
5301   \fi
5302 }

```

The `nogroupskip` key determines whether or not there should be a vertical gap between glossary groups.

```

5303 \define@choicekey{printgloss}{nogroupskip}{true,false}[true]{%
5304   \csuse{glsnogroupskip#1}}%
5305 }

```

The `nopostdot` key has the same effect as the package option of the same name.

```

5306 \define@choicekey{printgloss}{nopostdot}{true,false}[true]{%
5307   \csuse{glsnopostdot#1}}%
5308 }

```

The `entrycounter` key is the same as the package option but localised to the current glossary.

```

5309 \define@choicekey{printgloss}{entrycounter}{true,false}[true]{%
5310   \csuse{glsentrycounter#1}}%
5311   \ifglsentrycounter
5312     \ifx\@gls@counterwithin\@empty
5313       \newcounter{glossaryentry}%
5314     \else
5315       \newcounter{glossaryentry}[\@gls@counterwithin]%
5316     \fi
5317     \def\theHglossaryentry{\currentglossary.\theglossaryentry}%
5318     \renewcommand*\{glsresetentrycounter}{%
5319       \setcounter{glossaryentry}{0}}%
5320   }%
5321   \renewcommand*\{glsstepentry}[1]{%
5322     \refstepcounter{glossaryentry}%
5323     \label{glsentry-\glsdetoklabel{##1}}}%
5324   }%
5325   \renewcommand*\{glsentrycounterlabel}{\theglossaryentry.\space}%
5326   \renewcommand*\{glsentryitem}[1]{%
5327     \glsstepentry{##1}\glsentrycounterlabel
5328   }%
5329 \else
5330   \renewcommand*\{glsresetentrycounter}{}%
5331   \renewcommand*\{glsstepentry}[1]{}%
5332   \renewcommand*\{glsentrycounterlabel}{}%
5333   \renewcommand*\{glsentryitem}[1]{\glsresetsubentrycounter}
5334 \fi
5335 }

```

The `subentrycounter` key is the same as the package option but localised to the current glossary. Note that this doesn't affect the master/slave counter attributes, which occurs if `subentrycounter` and `entrycounter` package options are set to true.

```

5336 \define@choicekey{printgloss}{subentrycounter}{true,false}[true]{%
5337   \csuse{glssubentrycounter#1}%
5338   \ifglssubentrycounter
5339     \ifundef\c@glossarysubentry
5340     {%
5341       \ifglsubentrycounter
5342         \newcounter{glossarysubentry}[glossaryentry]%
5343       \else
5344         \newcounter{glossarysubentry}
5345       \fi
5346     }{}%
5347   \renewcommand*{\glstepsubentry}[1]{%
5348     \edef\currentglssubentry{\glsetoklabel{##1}}%
5349     \refstepcounter{glossarysubentry}%
5350     \label{glsubentry-\currentglssubentry}%
5351   }%
5352   \renewcommand*{\glsubentrycounter}{%
5353     \setcounter{glossarysubentry}{0}%
5354   }%
5355   \renewcommand*{\glssubentryitem}[1]{%
5356     \glstepsubentry{##1}\glssubentrycounterlabel
5357   }%
5358   \renewcommand*{\glssubentrycounterlabel}{\theglossarysubentry\space}%
5359   \def\theHglossarysubentry{\currentglssubentry.\theglossarysubentry}
5360 \else
5361   \renewcommand*{\glssubentryitem}[1]{}%
5362   \renewcommand*{\glstepsubentry}[1]{}%
5363   \renewcommand*{\glsubentrycounter}{}%
5364   \renewcommand*{\glssubentrycounterlabel}{}%
5365 \fi
5366 }

```

The nonumberlist key determines if this glossary should have a number list.

```

5367 \define@boolkey{printgloss}[gls]{nonumberlist}[true]{%
5368 \ifglnonumberlist
5369   \def\glossaryentrynumbers##1{}%
5370 \else
5371   \def\glossaryentrynumbers##1{##1}%
5372 \fi}

```

The sort key sets the glossary sort handler (\printnoidxglossary only).

```

5373 \define@key{printgloss}{sort}{\@glo@assign@sortkey{#1}}

```

\no@assign@sortkey Issue error if used with \printglossary

```

5374 \newcommand*{\@glo@no@assign@sortkey}[1]{%
5375   \PackageError{glossaries}{‘sort’ key not permitted with
5376     \string\printglossary}%
5377   {The ‘sort’ key may only be used with \string\printnoidxglossary}%
5378 }

```

`@glo@assign@sortkey` For use with `\printnoidxglossary`

```

5379 \newcommand*{\@glo@assign@sortkey}[1]{%
5380   \def\@glo@sorttype{#1}%
5381 }

```

`\@glsnonetopages` Suppresses the next number list only. Global assignments required as it may not occur in the same level of grouping as the next numberlist. (For example, if `\glsnonetopages` is place in the entry's description and 3 column tabular style glossary is used.) `\org@glossaryentrynumbers` needs to be set at the start of each glossary, in the event that `\glossaryentrynumber` is redefined.

```

5382 \newcommand*{\@glsnonetopages}{%
5383   \gdef\glossaryentrynumbers##1{%
5384     \glsresetentrylist
5385   }%
5386 }

```

`\@glsnextpages` Activate the next number list only. Global assignments required as it may not occur in the same level of grouping as the next numberlist. (For example, if `\glsnextpages` is place in the entry's description and 3 column tabular style glossary is used.) `\org@glossaryentrynumbers` needs to be set at the start of each glossary, in the event that `\glossaryentrynumber` is redefined.

```

5387 \newcommand*{\@glsnextpages}{%
5388   \gdef\glossaryentrynumbers##1{%
5389     ##1\glsresetentrylist}}

```

`\glsresetentrylist` Resets `\glossaryentrynumbers`

```

5390 \newcommand*{\glsresetentrylist}{%
5391   \global\let\glossaryentrynumbers\org@glossaryentrynumbers}

```

`\glsnonetopages` Outside of `\printglossary` this does nothing.

```

5392 \newcommand*{\glsnonetopages}{}

```

`\glsnextpages` Outside of `\printglossary` this does nothing.

```

5393 \newcommand*{\glsnextpages}{}

```

`glossaryentry` If the `entrycounter` package option has been used, define a counter to number each level 0 entry.

```

5394 \ifglstentrycounter
5395   \ifx\@gls@counterwithin\@empty
5396     \newcounter{glossaryentry}
5397   \else
5398     \newcounter{glossaryentry}[\@gls@counterwithin]
5399   \fi
5400   \def\theHglossaryentry{\currentglossary.\theglossaryentry}
5401 \fi

```


`glossarysubentry` If the `subentrycounter` package option has been used, define a counter to number each level 1 entry.

```

5402 \ifglssubentrycounter
5403   \ifglsentrycounter
5404     \newcounter{glossarysubentry}[glossaryentry]
5405   \else
5406     \newcounter{glossarysubentry}
5407   \fi
5408   \def\theHglossarysubentry{\currentglssubentry.\theglossarysubentry}
5409 \fi

```

`esetsubentrycounter` Resets the `glossarysubentry` counter.

```

5410 \ifglssubentrycounter
5411   \newcommand*{\glsresetsubentrycounter}{%
5412     \setcounter{glossarysubentry}{0}%
5413   }
5414 \else
5415   \newcommand*{\glsresetsubentrycounter}{}
5416 \fi

```

`esetsubentrycounter` Resets the `glossareentry` counter.

```

5417 \ifglsentrycounter
5418   \newcommand*{\glsresetentrycounter}{%
5419     \setcounter{glossaryentry}{0}%
5420   }
5421 \else
5422   \newcommand*{\glsresetentrycounter}{}
5423 \fi

```

`\glsstepentry` Advance the `glossaryentry` counter if in use. The argument is the label associated with the entry.

```

5424 \ifglsentrycounter
5425   \newcommand*{\glsstepentry}[1]{%
5426     \refstepcounter{glossaryentry}%
5427     \label{glsentry-\glsdetoklabel{#1}}%
5428   }
5429 \else
5430   \newcommand*{\glsstepentry}[1]{}
5431 \fi

```

`\glsstepsubentry` Advance the `glossarysubentry` counter if in use. The argument is the label associated with the subentry.

```

5432 \ifglssubentrycounter
5433   \newcommand*{\glsstepsubentry}[1]{%
5434     \edef\currentglssubentry{\glsdetoklabel{#1}}%
5435     \refstepcounter{glossarysubentry}%
5436     \label{glsentry-\currentglssubentry}%
5437   }

```

```

5438 \else
5439   \newcommand*{\glsstepsubentry}[1]{%
5440 \fi

```

`\glsrefentry` Reference the entry or sub-entry counter if in use, otherwise just do `\gls`.

```

5441 \ifglsentrycounter
5442   \newcommand*{\glsrefentry}[1]{\ref{glsentry-\glsdetoklabel{#1}}}
5443 \else
5444   \ifglssubentrycounter
5445     \newcommand*{\glsrefentry}[1]{\ref{glsentry-\glsdetoklabel{#1}}}
5446   \else
5447     \newcommand*{\glsrefentry}[1]{\gls{#1}}
5448   \fi
5449 \fi

```

`glsentrycounterlabel` Defines how to display the glossaryentry counter.

```

5450 \ifglsentrycounter
5451   \newcommand*{\glsentrycounterlabel}{\theglossaryentry.\space}
5452 \else
5453   \newcommand*{\glsentrycounterlabel}{}
5454 \fi

```

`glsentrycounterlabel` Defines how to display the glossarysubentry counter.

```

5455 \ifglssubentrycounter
5456   \newcommand*{\glsentrycounterlabel}{\theglossarysubentry)\space}
5457 \else
5458   \newcommand*{\glsentrycounterlabel}{}
5459 \fi

```

`\glsentryitem` Step and display glossaryentry counter, if appropriate.

```

5460 \ifglsentrycounter
5461   \newcommand*{\glsentryitem}[1]{%
5462     \glsstepentry{#1}\glsentrycounterlabel
5463   }
5464 \else
5465   \newcommand*{\glsentryitem}[1]{\glsresetsubentrycounter}
5466 \fi

```

`\glssubentryitem` Step and display glossarysubentry counter, if appropriate.

```

5467 \ifglssubentrycounter
5468   \newcommand*{\glssubentryitem}[1]{%
5469     \glsstepsubentry{#1}\glssubentrycounterlabel
5470   }
5471 \else
5472   \newcommand*{\glssubentryitem}[1]{}
5473 \fi

```

`theglossary` If the `theglossary` environment has already been defined, a warning will be issued. This environment should be redefined by glossary styles.

```

5474 \ifcsundef{theglossary}%
5475 {%
5476   \newenvironment{theglossary}{}{}%
5477 }%
5478 {%
5479   \@gls@warnontheGLOSSdefined
5480   \renewenvironment{theglossary}{}{}%
5481 }

```

The glossary header is given by `\glossaryheader`. This forms part of the glossary style, and must indicate what should appear immediately after the start of the `theglossary` environment. (For example, if the glossary uses a tabular-like environment, it may be used to set the header row.) Note that if you don't want a header row, the glossary style must redefine `\glossaryheader` to do nothing.

`\glossaryheader`

```

5482 \newcommand*{\glossaryheader}{}

```

`\glstarget` `\glstarget{<label>}{<name>}`

Provide user interface to `\@glstarget` to make it easier to modify the glossary style in the document.

```

5483 \newcommand*{\glstarget}[2]{\@glstarget{\glo@linkprefix#1}{#2}}

```

As from version 3.08, glossary information is now written to the external files using `\glossentry` and `\subglossentry` instead of `\glossaryentryfield` and `\glossarysubentryfield`. The default definition provides backward compatibility for glossary styles that use the old forms.

`compatibleglossentry`

`\glossentry{<label>}{<page-list>}`

```

5484 \providecommand*{\compatibleglossentry}[2]{%
5485   \toks@{#2}%
5486   \protected@edef\@do@glossentry{\noexpand\glossaryentryfield{#1}%
5487     {\noexpand\glsnamefont
5488       {\expandafter\expandonce\csname glo@#1@name\endcsname}}%
5489     {\expandafter\expandonce\csname glo@#1@desc\endcsname}%
5490     {\expandafter\expandonce\csname glo@#1@symbol\endcsname}%
5491     {\the\toks@}}%
5492   }%
5493   \@do@glossentry
5494 }

```

`\glossentryname`

```

5495 \newcommand*{\glossentryname}[1]{%
5496   \glsdoifexistsorwarn{#1}%

```

```

5497  {%
5498    \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
5499    \expandafter\glsnamefont\expandafter{\glo@name}%
5500  }%
5501 }

```

\Glossentryname

```

5502 \newcommand*{\Glossentryname}[1]{%
5503   \glsdoifexistsorwarn{#1}%
5504   {%
5505     \glsnamefont{\Glsentryname{#1}}%
5506   }%
5507 }

```

\glossentrydesc

```

5508 \newcommand*{\glossentrydesc}[1]{%
5509   \glsdoifexistsorwarn{#1}%
5510   {%
5511     \glentrydesc{#1}%
5512   }%
5513 }

```

\Glossentrydesc

```

5514 \newcommand*{\Glossentrydesc}[1]{%
5515   \glsdoifexistsorwarn{#1}%
5516   {%
5517     \Glsentrydesc{#1}%
5518   }%
5519 }

```

\glossentrysymbol

```

5520 \newcommand*{\glossentrysymbol}[1]{%
5521   \glsdoifexistsorwarn{#1}%
5522   {%
5523     \glentrysymbol{#1}%
5524   }%
5525 }

```

\Glossentrysymbol

```

5526 \newcommand*{\Glossentrysymbol}[1]{%
5527   \glsdoifexistsorwarn{#1}%
5528   {%
5529     \Glsentrysymbol{#1}%
5530   }%
5531 }

```

patiblesubglossentry `\subglossentry{<level>}{<label>}{<page-list>}`

```

5532 \providecommand*\compatiblesubglossentry}[3]{%
5533   \toks@{#3}%
5534   \protected@edef\@do@subglossentry{\noexpand\glossarysubentryfield{\number#1}%
5535     {#2}%
5536     {\noexpand\glsnamefont
5537       {\expandafter\expandonce\csname glo@#2@name\endcsname}}}%
5538     {\expandafter\expandonce\csname glo@#2@desc\endcsname}}%
5539     {\expandafter\expandonce\csname glo@#2@symbol\endcsname}}%
5540     {\the\toks@}%
5541   }%
5542   \@do@subglossentry
5543 }

```

sentrycompatibility

```

5544 \newcommand*\setglossentrycompatibility{%
5545   \let\glossentry\compatibleglossentry
5546   \let\subglossentry\compatiblesubglossentry
5547 }
5548 \setglossentrycompatibility

```

\glossaryentryfield

```
\glossaryentryfield{<label>}{<name>}{<description>}{<symbol>}{<page-list>}
```

This command formerly governed how each entry row should be formatted in the glossary. Now deprecated.

```

5549 \newcommand{\glossaryentryfield}[5]{%
5550   \GlossariesWarning
5551   {Deprecated use of \string\glossaryentryfield.^^J
5552     I recommend you change to \string\glossentry.^^J
5553     If you've just upgraded, try removing your gls auxiliary
5554     files^^J and recompile}%
5555   \noindent\textbf{\glstarget{#1}{#2}} #4 #3. #5\par}

```

lossarysubentryfield

```
\glossarysubentryfield{<level>}{<label>}{<name>}{<description>}{<symbol>}{<page-list>}
```

This command governs how each subentry should be formatted in the glossary. Glossary styles need to redefine this command. Most of the predefined styles ignore *<symbol>*. The first argument is a number indicating the level. (The level should be greater than or equal to 1.)

```

5556 \newcommand*\glossarysubentryfield}[6]{%
5557   \GlossariesWarning
5558   {Deprecated use of \string\glossarysubentryfield.^^J
5559     I recommend you change to \string\subglossentry.^^J
5560     If you've just upgraded, try removing your gls auxiliary

```

```

5561   files^^J and recompile}%
5562   \glstarget{#2}{\strut}#4. #6\par}

```

Within each glossary, the entries form distinct groups which are determined by the first character of the sort key. When using `makeindex`, there will be a maximum of 28 groups: symbols, numbers, and the 26 alphabetical groups A, ..., Z. If you use `xindy` the groups will depend on whatever alphabet is used. This is determined by the language or custom alphabets can be created in the `xindy` style file. The command `\glsgroupskip` specifies what to do between glossary groups. Glossary styles must redefine this command. (Note that `\glsgroupskip` only occurs between groups, not at the start or end of the glossary.)

`\glsgroupskip`

```

5563 \newcommand*{\glsgroupskip}{}

```

Each of the 28 glossary groups described above is preceded by a group heading. This is formatted by the command `\glsgroupheading` which takes one argument which is the *label* assigned to that group (not the title). The corresponding labels are: `glssymbols`, `glsnumbers`, A, ..., Z. Glossary styles must redefine this command. (In between groups, `\glsgroupheading` comes immediately after `\glsgroupskip`.)

`\glsgroupheading`

```

5564 \newcommand*{\glsgroupheading}[1]{}

```

It is possible to “trick” `makeindex` into treating entries as though they belong to the same group, even if the terms don’t start with the same letter, by modifying the sort key. For example, all entries belonging to one group could be defined so that the sort key starts with an a, while entries belonging to another group could be defined so that the sort key starts with a b, and so on. If you want each group to have a heading, you would then need to modify the translation control sequences `\glsgrouptitle` and `\glsgrouplabel` so that the label is translated into the required title (and vice-versa).

`\glsgrouptitle{<label>}`

This command produces the title for the glossary group whose label is given by `<label>`. By default, the group labelled `glssymbols` produces `\glssymbolsgroupname`, the group labelled `glsnumbers` produces `\glsnumbersgroupname` and all the other groups simply produce their label. As mentioned above, the group labels are: `glssymbols`, `glsnumbers`, A, ..., Z. If you want to redefine the group titles, you will need to redefine this command. Languages other than English may produce labels that are non-expandable, so we need to check for that otherwise it will create a “missing `\endcsname` inserted” error.

```

\glsgetgrouptitle
5565 \newcommand*{\glsgetgrouptitle}[1]{%
5566   \@gls@getgrouptitle{#1}{\@gls@grptitle}%
5567   \@gls@grptitle
5568 }

\@gls@getgrouptitle  Gets the group title specified by the label (first argument) and stores in the sec-
                      ond argument, which must be a control sequence.
5569 \newcommand*{\@gls@getgrouptitle}[2]{%
                      Even if the argument appears to be a single letter, it won't be considered a single
                      letter by \dtl@ifsingle if it's an active character.
5570   \dtl@ifsingle{#1}%
5571   {%
5572     \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
5573   }%
5574   {%
5575     \ifboolexpr{test{\ifstrequal{#1}{glssymbols}}
5576                 or test{\ifstrequal{#1}{glsnumbers}}}%
5577     {%
5578       \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
5579     }%
5580     {%
5581       \def#2{#1}%
5582     }%
5583   }%
5584 }

@getothergrouptitle  Version for the no-indexing app option:
5585 \newcommand*{\@gls@noidx@getgrouptitle}[2]{%
5586   \DTLifint{#1}%
5587   {\edef#2{\char#1\relax}}%
5588   {%
5589     \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
5590   }%
5591 }

```

`\glsgetgrouplabel{<title>}`

This command does the reverse to the previous command. The argument is the group title, and it produces the group label. Note that if you redefine `\glsgetgrouptitle`, you will also need to redefine `\glsgetgrouplabel`.

```

\glsgetgrouplabel
5592 \newcommand*{\glsgetgrouplabel}[1]{%
5593 \ifthenelse{\equal{#1}{\glssymbolsgroupname}}{\glssymbols}{%
5594 \ifthenelse{\equal{#1}{\glsnumbersgroupname}}{\glsnumbers}{#1}}

```

The command `\setentrycounter` sets the entry's associated counter (required by `\glshypernumber` etc.) `\glslink` and `\glsadd` encode the `\glossary` argument so that the relevant counter is set prior to the formatting command.

`\setentrycounter`

```
5595 \newcommand*{\setentrycounter}[2][1]{%
5596   \def\@glo@counterprefix{#1}%
5597   \ifx\@glo@counterprefix\@empty
5598     \def\@glo@counterprefix{.}%
5599   \else
5600     \def\@glo@counterprefix{.#1.}%
5601   \fi
5602   \def\glsentrycounter{#2}%
5603 }
```

The current glossary style can be set using `\setglossarystyle{<style>}`.

`\setglossarystyle`

```
5604 \newcommand*{\setglossarystyle}[1]{%
5605   \ifcsundef{@glsstyle@#1}%
5606   {%
5607     \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
5608   }%
5609   {%
5610     \csname @glsstyle@#1\endcsname
5611   }%
5612 }
```

`\glossarystyle`

```
5613 \newcommand*{\glossarystyle}[1]{%
5614   \ifcsundef{@glsstyle@#1}%
5615   {%
5616     \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
5617   }%
5618   {%
5619     \GlossariesWarning
5620     {Deprecated command \string\glossarystyle.^^J
5621     I recommend you switch to \string\setglossarystyle\space unless
5622     you want to maintain backward compatibility}%
5623     \setglossentrycompatibility
5624     \csname @glsstyle@#1\endcsname

5625     \ifcsdef{@glscompstyle@#1}%
5626     {\setglossentrycompatibility\csuse{@glscompstyle@#1}}%
5627     {}%
5628   }%
5629 }
```

`\newglossarystyle` New glossary styles can be defined using:

`\newglossarystyle{<name>}{<definition>}`

The `<definition>` argument should redefine `\theglossary`, `\glossaryheader`, `\glsgroupheading`, `\glossaryentryfield` and `\glsgroupskip` (see [subsection 1.18](#) for the definitions of predefined styles). Glossary styles should not redefine `\glossarypreamble` and `\glossarypostamble`, as the user should be able to switch between styles without affecting the pre- and postambles.

```
5630 \newcommand{\newglossarystyle}[2]{%
5631   \ifcsundef{@glsstyle@#1}%
5632   {%
5633     \expandafter\def\csname @glsstyle@#1\endcsname{#2}%
5634   }%
5635   {%
5636     \PackageError{glossaries}{Glossary style ‘#1’ is already defined}{}%
5637   }%
5638 }
```

`\renewglossarystyle` Code for this macro supplied by Marco Daniel.

```
5639 \newcommand{\renewglossarystyle}[2]{%
5640   \ifcsundef{@glsstyle@#1}%
5641   {%
5642     \PackageError{glossaries}{Glossary style ‘#1’ isn’t already defined}{}%
5643   }%
5644   {%
5645     \csdef{@glsstyle@#1}{#2}%
5646   }%
5647 }
```

Glossary entries are encoded so that the second argument to `\glossaryentryfield` is always specified as `\glsnamefont{<name>}`. This allows the user to change the font used to display the name term without having to redefine `\glossaryentryfield`. The default uses the surrounding font, so in the list type styles (which place the name in the optional argument to `\item`) the name will appear in bold.

`\glsnamefont`

```
5648 \newcommand*{\glsnamefont}[1]{#1}
```

Each glossary entry has an associated number list (usually page numbers) that indicate where in the document the entry has been used. The format for these number lists can be changed using the format key in commands like `\glslink`. The default format is given by `\glshypernumber`. This takes a single argument which may be a single number, a number range or a number list. The number ranges are delimited with `\delimR`, the number lists are delimited with `\delimN`.

If the document doesn’t have hyperlinks, the numbers can be displayed just as they are, but if the document supports hyperlinks, the numbers should link to the relevant location. This means extracting the individual numbers from

the list or ranges. The package does this with the `\hyperpage` command, but this is encoded for comma and dash delimiters and only for the page counter, but this code needs to be more general. So I have adapted the code used in the package.

`\glshypernumber`

```
5649 \ifcsundef{hyperlink}%
5650 {%
5651   \def\glshypernumber#1{#1}%
5652 }%
5653 {%
5654   \def\glshypernumber#1{\@glshypernumber#1\nohyperpage{}\@nil}
5655 }
```

`\@glshypernumber` This code was provided by Heiko Oberdiek to allow material to be attached to the location.

```
5656 \def\@glshypernumber#1\nohyperpage#2#3\@nil{%
5657   \ifx\#1\%
5658   \else
5659     \@delimR#1\delimR\delimR\%
5660   \fi
5661   \ifx\#2\%
5662   \else
5663     #2%
5664   \fi
5665   \ifx\#3\%
5666   \else
5667     \@glshypernumber#3\@nil
5668   \fi
5669 }
```

`\@delimR` displays a range of numbers for the counter whose name is given by `\@gls@counter` (which must be set prior to using `\glshypernumber`).

`\@delimR`

```
5670 \def\@delimR#1\delimR #2\delimR #3\%
5671 \ifx\#2\%
5672   \@delimN{#1}%
5673 \else
5674   \@gls@numberlink{#1}\delimR\@gls@numberlink{#2}%
5675 \fi}
```

`\@delimN` displays a list of individual numbers, instead of a range:

`\@delimN`

```
5676 \def\@delimN#1{\@@delimN#1\delimN \delimN\%
5677 \def\@@delimN#1\delimN #2\delimN#3\%
5678 \ifx\#3\%
5679   \@gls@numberlink{#1}%
5680 }
```

```

5680 \else
5681   \@gls@numberlink{#1}\delimN\@gls@numberlink{#2}%
5682 \fi
5683 }

```

The following code is modified from hyperref's \HyInd@pagelink where the name of the counter being used is given by \@gls@counter.

```

5684 \def\@gls@numberlink#1{%
5685 \begingroup
5686 \toks@={}%
5687 \@gls@removespaces#1 \@nil
5688 \endgroup}

5689 \def\@gls@removespaces#1 #2\@nil{%
5690 \toks@=\expandafter{\the\toks@#1}%
5691 \ifx\#2\%
5692   \edef\x{\the\toks@}%
5693   \ifx\x\empty
5694     \else

5695     \hyperlink{\glstentrycounter\@gls@counterprefix\the\toks@}%
5696               {\the\toks@}%
5697   \fi
5698 \else
5699   \@gls@ReturnAfterFi{%
5700     \@gls@removespaces#2\@nil
5701   }%
5702 \fi
5703 }
5704 \long\def\@gls@ReturnAfterFi#1\fi{\fi#1}

```

The following commands will switch to the appropriate font, and create a hyperlink, if hyperlinks are supported. If hyperlinks are not supported, they will just display their argument in the appropriate font.

```

\hyperrm
5705 \newcommand*{\hyperrm}[1]{\textrm{\glshypernumber{#1}}}

\hypersf
5706 \newcommand*{\hypersf}[1]{\textsf{\glshypernumber{#1}}}

\hypertt
5707 \newcommand*{\hypertt}[1]{\texttt{\glshypernumber{#1}}}

\hyperbf
5708 \newcommand*{\hyperbf}[1]{\textbf{\glshypernumber{#1}}}

\hypermd
5709 \newcommand*{\hypermd}[1]{\textmd{\glshypernumber{#1}}}

```

```

\hyperit
5710 \newcommand*{\hyperit}[1]{\textit{\glshypernumber{#1}}}

\hypersl
5711 \newcommand*{\hypersl}[1]{\textsl{\glshypernumber{#1}}}

\hyperup
5712 \newcommand*{\hyperup}[1]{\textup{\glshypernumber{#1}}}

\hypersc
5713 \newcommand*{\hypersc}[1]{\textsc{\glshypernumber{#1}}}

\hyperemph
5714 \newcommand*{\hyperemph}[1]{\emph{\glshypernumber{#1}}}

```

1.16 Acronyms

```

\oldacronym \oldacronym[⟨label⟩]{⟨abbrv⟩}{⟨long⟩}{⟨key-val list⟩}

```

This emulates the way the old package defined acronyms. It is equivalent to `\newacronym[⟨key-val list⟩]{⟨label⟩}{⟨abbrv⟩}{⟨long⟩}` and it additionally defines the command `\⟨label⟩` which is equivalent to `\gls{⟨label⟩}` (thus `⟨label⟩` must only contain alphabetical characters). If `⟨label⟩` is omitted, `⟨abbrv⟩` is used. This only emulates the syntax of the old package. The way the acronyms appear in the list of acronyms is determined by the definition of `\newacronym` and the glossary style.

Note that `\⟨label⟩` can't have an optional argument if the package is loaded. If hasn't been loaded then you can do `\⟨label⟩[⟨insert⟩]` but you can't do `\⟨label⟩[⟨key-val list⟩]`. For example if you define the acronym `svm`, then you can do `\svm['s]` but you can't do `\svm[format=textbf]`. If the package is loaded, `\svm['s]` will appear as `svm ['s]` which is unlikely to be the desired result. In this case, you will need to use `\gls` explicitly, e.g. `\gls{svm}['s]`.

Note that it is up to the user to load if desired.

```

5715 \newcommand{\oldacronym}[4][\gls@label]{%
5716   \def\gls@label{#2}%
5717   \newacronym[#4]{#1}{#2}{#3}%
5718   \ifcsundef{xspace}%
5719     {%
5720       \expandafter\edef\csname#1\endcsname{%
5721         \noexpand\@ifstar{\noexpand\Gls{#1}}{\noexpand\gls{#1}}}%
5722       }%
5723     }%
5724     {%
5725       \expandafter\edef\csname#1\endcsname{%
5726         \noexpand\@ifstar{\noexpand\Gls{#1}\noexpand\xspace}{%

```

```

5727     \noexpand\gls{#1}\noexpand\xspace}%
5728   }%
5729 }%
5730 }

```

```
\newacronym[⟨key-val list⟩]{⟨label⟩}{⟨abbrev⟩}{⟨long⟩}
```

This is a quick way of defining acronyms, using `\newglossaryentry` with the appropriate values. It sets the glossary type to `\acronymtype` which will be `acronym` if the package option `acronym` has been used, otherwise it will be the default glossary. Since `\newacronym` merely calls `\newglossaryentry`, the acronym is treated like any other glossary entry.

If you prefer a different format, you can redefine `\newacronym` as required. The optional argument can be used to override any of the settings.

This is just a stub. It's redefined by commands like `\SetDefaultAcronymStyle`.

`\newacronym`

```
5731 \newcommand{\newacronym}[4] [] {}
```

Set up some convenient short cuts. These need to be changed if `\newacronym` is changed (or if the description key is changed).

`\acrpluralsuffix` Plural suffix used by `\newacronym`. This just defaults to `\glspluralsuffix` but is changed to include `\textup` if the `smallcaps` option is used, so that the suffix doesn't appear in small caps as it doesn't look right. For example, `ABCS` looks as though the “s” is part of the acronym, but `ABCS` looks as though the “s” is a plural suffix. Since the entire text `abcs` is set in `\textsc`, `\textup` is needed to cancel it out.

```
5732 \newcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}
```

If `garamondx` has been loaded, need to use `\textulc` instead of `\textup`.

`\glstextup`

```
5733 \newrobustcmd*{\glstextup}[1]{\ifdef\textulc{\textulc{#1}}{\textup{#1}}}
```

The following are defined for compatibility with version 2.07 and earlier.

`\glsshortkey`

```
5734 \newcommand*{\glsshortkey}{short}
```

`\glsshortpluralkey`

```
5735 \newcommand*{\glsshortpluralkey}{shortplural}
```

`\glslongkey`

```
5736 \newcommand*{\glslongkey}{long}
```

`\glslongpluralkey`

```
5737 \newcommand*{\glslongpluralkey}{longplural}
```

`\acrfull` Full form of the acronym.

```
5738 \newrobustcmd*{\acrfull}{\@gls@hyp@opt\@ns@acrfull}
```

```
5739 \newcommand*\ns@acrfull[2][]{%
5740   \new@ifnextchar[{\@acrfull{#1}{#2}}}%
5741   {\@acrfull{#1}{#2}[]}%
5742 }
```

`\@acrfull` Low-level macro:

```
5743 \def\@acrfull#1#2[#3]{%
    Make it easier for acronym styles to change this:
5744   \acrfullfmt{#1}{#2}{#3}%
5745 }
```

Using `\acrlinkfullformat` and `\acrfullformat` is now deprecated as it can cause complications with the first letter upper case variants, but the package needs to provide backward compatibility support.

`\acrfullfmt` No case change full format.

```
5746 \newcommand*{\acrfullfmt}[3]{%
5747   \acrlinkfullformat{\@acrlong}{\@acrshort}{#1}{#2}{#3}%
5748 }
```

`\acrlinkfullformat` Format for full links like `\acrfull`. Syntax: `\acrlinkfullformat{<long cs>}{<short cs>}{<options>}{<label>}{<insert>}`

```
5749 \newcommand{\acrlinkfullformat}[5]{%
5750   \acrfullformat{#1}{#3}{#4}[#5]{#2}{#3}{#4}[]}%
5751 }
```

`\acrfullformat` Default full form is `<long>` (`<short>`).

```
5752 \newcommand{\acrfullformat}[2]{#1\glsspace(#2)}
```

`\glsspace` Robust space to ensure it's written to the `.glsdefs` file.

```
5753 \newrobustcmd{\glsspace}{\space}
```

Default format for full acronym

`\Acrfull`

```
5754 \newrobustcmd*{\Acrfull}{\@gls@hyp@opt\@ns@Acrfull}

5755 \newcommand*\ns@Acrfull[2][]{%
5756   \new@ifnextchar[{\@Acrfull{#1}{#2}}}%
5757   {\@Acrfull{#1}{#2}[]}%
5758 }
```

Low-level macro:

```
5759 \def\@Acrfull#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
5760 \Acrfullfmt{#1}{#2}{#3}%  
5761 }
```

`\Acrfullfmt` First letter upper case full format.

```
5762 \newcommand*\Acrfullfmt}[3]{%  
5763 \acrlinkfullformat{\@Acrlong}{\@acrshort}{#1}{#2}{#3}%  
5764 }
```

`\ACRfull`

```
5765 \newrobustcmd*\ACRfull{\@gls@hyp@opt\@ns@ACRfull}  
  
5766 \newcommand*\ns@ACRfull[2][]{%  
5767 \new@ifnextchar[\@ACRfull{#1}{#2}}%  
5768 {\@ACRfull{#1}{#2}[]}%  
5769 }
```

Low-level macro:

```
5770 \def\@ACRfull#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
5771 \ACRfullfmt{#1}{#2}{#3}%  
5772 }
```

`\ACRfullfmt` All upper case full format.

```
5773 \newcommand*\ACRfullfmt}[3]{%  
5774 \acrlinkfullformat{\@ACRlong}{\@ACRshort}{#1}{#2}{#3}%  
5775 }
```

Plural:

`\acrfullpl`

```
5776 \newrobustcmd*\acrfullpl{\@gls@hyp@opt\@ns@acrfullpl}  
  
5777 \newcommand*\ns@acrfullpl[2][]{%  
5778 \new@ifnextchar[\@acrfullpl{#1}{#2}}%  
5779 {\@acrfullpl{#1}{#2}[]}%  
5780 }
```

Low-level macro:

```
5781 \def\@acrfullpl#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
5782 \acrfullplfmt{#1}{#2}{#3}%  
5783 }
```

```

\acrfullplfmt  No case change plural full format.
5784 \newcommand*\acrfullplfmt}[3]{%
5785   \acrlinkfullformat{\@acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%
5786 }

\Acrfullpl
5787 \newrobustcmd*\Acrfullpl{\@gls@hyp@opt\ns@Acrfullpl}

5788 \newcommand*\ns@Acrfullpl[2][{}]{%
5789   \new@ifnextchar[{\@Acrfullpl{#1}{#2}}%
5790     {\@Acrfullpl{#1}{#2}[]}%
5791 }

    Low-level macro:
5792 \def\@Acrfullpl#1#2[#3]{%

    Make it easier for acronym styles to change this:
5793   \Acrfullplfmt{#1}{#2}{#3}%
5794 }

\Acrfullplfmt  First letter upper case plural full format.
5795 \newcommand*\Acrfullplfmt}[3]{%
5796   \acrlinkfullformat{\@Acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%
5797 }

\ACRfullpl
5798 \newrobustcmd*\ACRfullpl{\@gls@hyp@opt\ns@ACRfullpl}

5799 \newcommand*\ns@ACRfullpl[2][{}]{%
5800   \new@ifnextchar[{\@ACRfullpl{#1}{#2}}%
5801     {\@ACRfullpl{#1}{#2}[]}%
5802 }

    Low-level macro:
5803 \def\@ACRfullpl#1#2[#3]{%

    Make it easier for acronym styles to change this:
5804   \ACRfullplfmt{#1}{#2}{#3}%
5805 }

\ACRfullplfmt  All upper case plural full format.
5806 \newcommand*\ACRfullplfmt}[3]{%
5807   \acrlinkfullformat{\@ACRlongpl}{\@ACRshortpl}{#1}{#2}{#3}%
5808 }

```


1.17 Predefined acronym styles

`\acronymfont` This is only used with the additional acronym styles:

```
5809 \newcommand{\acronymfont}[1]{#1}
```

`\firstacronymfont` This is only used with the additional acronym styles:

```
5810 \newcommand{\firstacronymfont}[1]{\acronymfont{#1}}
```

`\acrnameformat` The styles that allow an additional description use `\acrnameformat{<short>}{<long>}` to determine what information is displayed in the name.

```
5811 \newcommand*{\acrnameformat}[2]{\acronymfont{#1}}
```

Define some tokens used by `\newacronym`:

`\glskeylisttok`

```
5812 \newtoks\glskeylisttok
```

`\glslabeltok`

```
5813 \newtoks\glslabeltok
```

`\glsshorttok`

```
5814 \newtoks\glsshorttok
```

`\glslongtok`

```
5815 \newtoks\glslongtok
```

`\newacronymhook` Provide a hook for `\newacronym`:

```
5816 \newcommand*{\newacronymhook}{}
```

`\setGenericNewAcronym` New improved version of setting the acronym style.

```
5817 \newcommand*{\SetGenericNewAcronym}{%
```

Change the behaviour of `\Glsentryname` to workaround expansion issues that cause a problem for `\makefirstuc`

```
5818 \let\@Gls@entryname\@Gls@acentryname
```

Change the way acronyms are defined:

```
5819 \renewcommand{\newacronym}[4][\]{%
5820 \ifdefempty{\@glsacronymlists}%
5821 {%
5822 \def\@glo@type{\acronymtype}%
5823 \setkeys{glossentry}{##1}%
5824 \DeclareAcronymList{\@glo@type}%
5825 }%
5826 }%
5827 \glskeylisttok{##1}%
5828 \glslabeltok{##2}%
5829 \glsshorttok{##3}%
5830 \glslongtok{##4}%
```

```

5831 \newacronymhook
5832 \protected@edef\@do@newglossaryentry{%
5833 \noexpand\newglossaryentry{\the\glslabeltok}%
5834 {%
5835 type=\acronymtype,%
5836 name={\expandonce{\acronymentry{##2}}},%
5837 sort={\acronymsort{\the\glsshorttok}{\the\glslongtok}},%
5838 text={\the\glsshorttok},%
5839 short={\the\glsshorttok},%
5840 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
5841 long={\the\glslongtok},%
5842 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
5843 \GenericAcronymFields,%
5844 \the\glskeylisttok
5845 }%
5846 }%
5847 \@do@newglossaryentry
5848 }%

```

Make sure that \acrfull etc reflects the new style:

```

5849 \renewcommand*\acrfullfmt}[3]{%
5850 \glslink[##1]{##2}{\genacrfullformat{##2}{##3}}}%
5851 \renewcommand*\Acrfullfmt}[3]{%
5852 \glslink[##1]{##2}{\Genacrfullformat{##2}{##3}}}%
5853 \renewcommand*\ACRfullfmt}[3]{%
5854 \glslink[##1]{##2}{%
5855 \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}}%
5856 \renewcommand*\acrfullplfmt}[3]{%
5857 \glslink[##1]{##2}{\genplacrfullformat{##2}{##3}}}%
5858 \renewcommand*\Acrfullplfmt}[3]{%
5859 \glslink[##1]{##2}{\Genplacrfullformat{##2}{##3}}}%
5860 \renewcommand*\ACRfullplfmt}[3]{%
5861 \glslink[##1]{##2}{%
5862 \mfirstucMakeUppercase{\genplacrfullformat{##2}{##3}}}}%

```

Make sure that \glsentryfull etc reflects the new style:

```

5863 \renewcommand*\glsentryfull}[1]{\genacrfullformat{##1}{}}%
5864 \renewcommand*\Glsentryfull}[1]{\Genacrfullformat{##1}{}}%
5865 \renewcommand*\glsentryfullpl}[1]{\genplacrfullformat{##1}{}}%
5866 \renewcommand*\Glsentryfullpl}[1]{\Genplacrfullformat{##1}{}}%
5867 }

```

`\GenericAcronymFields` Fields used by \SetGenericNewAcronym that can be changed by the acronym style.

```

5868 \newcommand*\GenericAcronymFields{description={\the\glslongtok}}

```

`\acronymentry` `\acronymentry{<label>}`

Display style for the name field in the list of acronyms.

```
5869 \newcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{#1}}}
```

`\acronymsort` `\acronymsort{<short>}{<long>}`

Default sort format for acronyms.

```
5870 \newcommand*{\acronymsort}[2]{#1}
```

`\setacronymstyle` `\setacronymstyle{<style name>}`

```
5871 \newcommand*{\setacronymstyle}[1]{%
5872   \ifcsundef{@glsacr@dispstyle@#1}%
5873   {%
5874     \PackageError{glossaries}{Undefined acronym style ‘#1’}{%
5875     }%
5876   }%
5877   \ifdefempty{@glsacronymlists}%
5878   {%
5879     \DeclareAcronymList{\acronymtype}%
5880   }%
5881   {%
5882     \SetGenericNewAcronym
5883     \GlsUseAcrStyleDefs{#1}%
5884     \@for\@gls@type:=\@glsacronymlists\do{%
5885       \defglsentryfmt[\@gls@type]{\GlsUseAcrEntryDispStyle{#1}}%
5886     }%
5887   }%
5888 }
```

`\newacronymstyle` `\newacronymstyle{<style name>}{<entry format definition>}{<display definitions>}`

Defines a new acronym style called *<style name>*.

```
5889 \newcommand*{\newacronymstyle}[3]{%
5890   \ifcsdef{@glsacr@dispstyle@#1}%
5891   {%
5892     \PackageError{glossaries}{Acronym style ‘#1’ already exists}{%
5893     }%
5894   }%
5895   \csdef{@glsacr@dispstyle@#1}{#2}%
5896   \csdef{@glsacr@styledefs@#1}{#3}%
5897   }%
5898 }
```

`\renewacronymstyle` Redefines the given acronym style.

```

5899 \newcommand*{\renewacronymstyle}[3]{%
5900   \ifcsdef{@glsacr@dispstyle@#1}%
5901   {%
5902     \csdef{@glsacr@dispstyle@#1}{#2}%
5903     \csdef{@glsacr@styledefs@#1}{#3}%
5904   }%
5905   {%
5906     \PackageError{glossaries}{Acronym style ‘#1’ doesn’t exist}{}%
5907   }%
5908 }

```

seAcrEntryDispStyle

```

5909 \newcommand*{\GlsUseAcrEntryDispStyle}[1]{\csuse{@glsacr@dispstyle@#1}}

```

\GlsUseAcrStyleDefs

```

5910 \newcommand*{\GlsUseAcrStyleDefs}[1]{\csuse{@glsacr@styledefs@#1}}

```

Predefined acronym styles:

long-short *(long)* (*short*) acronym style.

```

5911 \newacronymstyle{long-short}%
5912 {%

```

Check for long form in case this is a mixed glossary.

```

5913   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
5914 }%
5915 {%
5916   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
5917   \renewcommand*{\genacrfullformat}[2]{%
5918     \glsentrylong{##1}##2\space
5919     (\protect\firstacronymfont{\glsentryshort{##1}})}%
5920   }%
5921   \renewcommand*{\Genacrfullformat}[2]{%
5922     \Glsentrylong{##1}##2\space
5923     (\protect\firstacronymfont{\glsentryshort{##1}})}%
5924   }%
5925   \renewcommand*{\genplacrfullformat}[2]{%
5926     \glsentrylongpl{##1}##2\space
5927     (\protect\firstacronymfont{\glsentryshortpl{##1}})}%
5928   }%
5929   \renewcommand*{\Genplacrfullformat}[2]{%
5930     \Glsentrylongpl{##1}##2\space
5931     (\protect\firstacronymfont{\glsentryshortpl{##1}})}%
5932   }%
5933   \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}
5934   \renewcommand*{\acronymsort}[2]{##1}%
5935   \renewcommand*{\acronymfont}[1]{##1}%
5936   \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
5937   \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
5938 }

```

short-long *<short>* (*<long>*) acronym style.

```
5939 \newacronymstyle{short-long}%
5940 {%
    Check for long form in case this is a mixed glossary.
5941   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
5942 }%
5943 {%
5944   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
5945   \renewcommand*{\genacrfullformat}[2]{%
5946     \protect\firstacronymfont{\glsentryshort{##1}}##2\space
5947     (\glsentrylong{##1})%
5948   }%
5949   \renewcommand*{\Genacrfullformat}[2]{%
5950     \protect\firstacronymfont{\Glsentryshort{##1}}##2\space
5951     (\glsentrylong{##1})%
5952   }%
5953   \renewcommand*{\genplacrfullformat}[2]{%
5954     \protect\firstacronymfont{\glsentryshortpl{##1}}##2\space
5955     (\glsentrylongpl{##1})%
5956   }%
5957   \renewcommand*{\Genplacrfullformat}[2]{%
5958     \protect\firstacronymfont{\Glsentryshortpl{##1}}##2\space
5959     (\glsentrylongpl{##1})%
5960   }%

5961   \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
5962   \renewcommand*{\acronymsort}[2]{##1}%
5963   \renewcommand*{\acronymfont}[1]{##1}%
5964   \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
5965   \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
5966 }
```

long-sc-short *<long>* (\textsc{<short>}) acronym style.

```
5967 \newacronymstyle{long-sc-short}%
5968 {%
5969   \GlsUseAcrEntryDisplayStyle{long-short}%
5970 }%
5971 {%
5972   \GlsUseAcrStyleDefs{long-short}%
5973   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
5974   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
5975 }
```

long-sm-short *<long>* (\textsmaller{<short>}) acronym style.

```
5976 \newacronymstyle{long-sm-short}%
5977 {%
5978   \GlsUseAcrEntryDisplayStyle{long-short}%
5979 }%
5980 {%
```

```

5981 \GlsUseAcrStyleDefs{long-short}%
5982 \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
5983 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
5984 }

```

sc-short-long *<short>* (*\textsc{<long>}*) acronym style.

```

5985 \newacronymstyle{sc-short-long}%
5986 {%
5987   \GlsUseAcrEntryDisplayStyle{short-long}%
5988 }%
5989 {%
5990   \GlsUseAcrStyleDefs{short-long}%
5991   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
5992   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
5993 }

```

sm-short-long *<short>* (*\textsmaller{<long>}*) acronym style.

```

5994 \newacronymstyle{sm-short-long}%
5995 {%
5996   \GlsUseAcrEntryDisplayStyle{short-long}%
5997 }%
5998 {%
5999   \GlsUseAcrStyleDefs{short-long}%
6000   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6001   \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6002 }

```

long-short-desc *<long>* (*{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

6003 \newacronymstyle{long-short-desc}%
6004 {%
6005   \GlsUseAcrEntryDisplayStyle{long-short}%
6006 }%
6007 {%
6008   \GlsUseAcrStyleDefs{long-short}%
6009   \renewcommand*{\GenericAcronymFields}{}%
6010   \renewcommand*{\acronymsort}[2]{##2}%
6011   \renewcommand*{\acronymentry}[1]{%
6012     \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6013 }

```

long-sc-short-desc *<long>* (*\textsc{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

6014 \newacronymstyle{long-sc-short-desc}%
6015 {%
6016   \GlsUseAcrEntryDisplayStyle{long-sc-short}%
6017 }%
6018 {%

```

```

6019 \GlsUseAcrStyleDefs{long-sm-short}%
6020 \renewcommand*{\GenericAcronymFields}{}%
6021 \renewcommand*{\acronymsort}[2]{##2}%
6022 \renewcommand*{\acronymentry}[1]{%
6023     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6024 }

```

long-sm-short-desc *⟨long⟩* (\textsmaller{⟨short⟩}) acronym style that has an accompanying description (which the user needs to supply).

```

6025 \newacronymstyle{long-sm-short-desc}%
6026 {%
6027     \GlsUseAcrEntryDisplayStyle{long-sm-short}%
6028 }%
6029 {%
6030     \GlsUseAcrStyleDefs{long-sm-short}%
6031     \renewcommand*{\GenericAcronymFields}{}%
6032     \renewcommand*{\acronymsort}[2]{##2}%
6033     \renewcommand*{\acronymentry}[1]{%
6034         \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6035 }

```

short-long-desc *⟨short⟩* ({⟨long⟩}) acronym style that has an accompanying description (which the user needs to supply).

```

6036 \newacronymstyle{short-long-desc}%
6037 {%
6038     \GlsUseAcrEntryDisplayStyle{short-long}%
6039 }%
6040 {%
6041     \GlsUseAcrStyleDefs{short-long}%
6042     \renewcommand*{\GenericAcronymFields}{}%
6043     \renewcommand*{\acronymsort}[2]{##2}%
6044     \renewcommand*{\acronymentry}[1]{%
6045         \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6046 }

```

sc-short-long-desc *⟨long⟩* (\textsc{⟨short⟩}) acronym style that has an accompanying description (which the user needs to supply).

```

6047 \newacronymstyle{sc-short-long-desc}%
6048 {%
6049     \GlsUseAcrEntryDisplayStyle{sc-short-long}%
6050 }%
6051 {%
6052     \GlsUseAcrStyleDefs{sc-short-long}%
6053     \renewcommand*{\GenericAcronymFields}{}%
6054     \renewcommand*{\acronymsort}[2]{##2}%
6055     \renewcommand*{\acronymentry}[1]{%
6056         \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6057 }

```

sm-short-long-desc *<long>* (\textsmaller{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

6058 \newacronymstyle{sm-short-long-desc}%
6059 {%
6060   \GlsUseAcrEntryDispStyle{sm-short-long}%
6061 }%
6062 {%
6063   \GlsUseAcrStyleDefs{sm-short-long}%
6064   \renewcommand*{\GenericAcronymFields}{}%
6065   \renewcommand*{\acronymsort}[2]{##2}%
6066   \renewcommand*{\acronymentry}[1]{%
6067     \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6068 }

```

dua *<long>* only acronym style.

```

6069 \newacronymstyle{dua}%
6070 {%

```

Check for long form in case this is a mixed glossary.

```

6071   \ifdefempty\glscustomtext
6072   {%
6073     \ifglshaslong{\glslabel}%
6074     {%
6075       \glsifplural
6076       {%

```

Plural form:

```

6077         \glscapscase
6078         {%

```

Plural form, don't adjust case:

```

6079         \glsentrylongpl{\glslabel}\glsinsert
6080         }%
6081         {%

```

Plural form, make first letter upper case:

```

6082         \Glsentrylongpl{\glslabel}\glsinsert
6083         }%
6084         {%

```

Plural form, all caps:

```

6085         \mfirstucMakeUppercase
6086         {\glsentrylongpl{\glslabel}\glsinsert}%
6087         }%
6088         }%
6089         {%

```

Singular form

```

6090         \glscapscase
6091         {%

```


Singular form, don't adjust case:

```
6092      \glsentrylong{\glslabel}\glsinsert
6093      }%
6094      {%
```

Subsequent singular form, make first letter upper case:

```
6095      \Glsentrylong{\glslabel}\glsinsert
6096      }%
6097      {%
```

Subsequent singular form, all caps:

```
6098      \mfirstucMakeUppercase
6099      {\glsentrylong{\glslabel}\glsinsert}%
6100      }%
6101      }%
6102      }%
6103      {%
```

Not an acronym:

```
6104      \glsgenentryfmt
6105      }%
6106      }%
6107      {\glscustomtext\glsinsert}%
6108      }%
6109      {%
6110      \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%

6111      \renewcommand*{\acrfullfmt}[3]{%
6112          \glslink[##1]{##2}{\glsentrylong{##2}##3\space
6113              (\acronymfont{\glsentryshort{##2}})}}%
6114      \renewcommand*{\Acrfullfmt}[3]{%
6115          \glslink[##1]{##2}{\Glsentrylong{##2}##3\space
6116              (\acronymfont{\glsentryshort{##2}})}}%
6117      \renewcommand*{\ACRfullfmt}[3]{%
6118          \glslink[##1]{##2}{%
6119              \mfirstucMakeUppercase{\glsentrylong{##2}##3\space
6120                  (\acronymfont{\glsentryshort{##2}})}}}%

6121      \renewcommand*{\acrfullplfmt}[3]{%
6122          \glslink[##1]{##2}{\glsentrylongpl{##2}##3\space
6123              (\acronymfont{\glsentryshortpl{##2}})}}%

6124      \renewcommand*{\Acrfullplfmt}[3]{%
6125          \glslink[##1]{##2}{\Glsentrylongpl{##2}##3\space
6126              (\acronymfont{\glsentryshortpl{##2}})}}%
6127      \renewcommand*{\ACRfullplfmt}[3]{%
6128          \glslink[##1]{##2}{%
6129              \mfirstucMakeUppercase{\glsentrylongpl{##2}##3\space
6130                  (\acronymfont{\glsentryshortpl{##2}})}}}%
6131      \renewcommand*{\glsentryfull}[1]{%
6132          \glsentrylong{##1}\space(\acronymfont{\glsentryshort{##1}})%
```

```

6133 }%
6134 \renewcommand*{\Glsentryfull}[1]{%
6135   \Glsentrylong{##1}\space(\acronymfont{\glsentryshort{##1}})%
6136 }%
6137 \renewcommand*{\glsentryfullpl}[1]{%
6138   \glsentrylongpl{##1}\space(\acronymfont{\glsentryshortpl{##1}})%
6139 }%
6140 \renewcommand*{\Glsentryfullpl}[1]{%
6141   \Glsentrylongpl{##1}\space(\acronymfont{\glsentryshortpl{##1}})%
6142 }%
6143 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6144 \renewcommand*{\acronymsort}[2]{##1}%
6145 \renewcommand*{\acronymfont}[1]{##1}%
6146 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6147 }

```

dua-desc *<long>* only acronym style with user-supplied description.

```

6148 \newacronymstyle{dua-desc}%
6149 {%
6150   \GlsUseAcrEntryDispStyle{dua}%
6151 }%
6152 {%
6153   \GlsUseAcrStyleDefs{dua}%
6154   \renewcommand*{\GenericAcronymFields}{}%
6155   \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentrylong{##1}}}%
6156   \renewcommand*{\acronymsort}[2]{##2}%
6157 }%

```

footnote *<short>*\footnote{*<long>*} acronym style.

```

6158 \newacronymstyle{footnote}%
6159 {%
6160   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6161 }%
6162 {%
6163   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6164   \glshyperfirstfalse
6165   \renewcommand*{\genacrfullformat}[2]{%
6166     \protect\firstacronymfont{\glsentryshort{##1}}##2%
6167     \protect\footnote{\glsentrylong{##1}}%
6168   }%
6169   \renewcommand*{\Genacrfullformat}[2]{%
6170     \firstacronymfont{\glsentryshort{##1}}##2%
6171     \protect\footnote{\glsentrylong{##1}}%
6172   }%
6173   \renewcommand*{\genplacrfullformat}[2]{%

```

```

6174 \protect\firstacronymfont{\glentryshortpl{##1}}##2%
6175 \protect\footnote{\glentrylongpl{##1}}%
6176 }%
6177 \renewcommand*{\Genplacrfullformat}[2]{%
6178 \protect\firstacronymfont{\glentryshortpl{##1}}##2%
6179 \protect\footnote{\glentrylongpl{##1}}%
6180 }%
6181 \renewcommand*{\acronymentry}[1]{\acronymfont{\glentryshort{##1}}}%
6182 \renewcommand*{\acronymsort}[2]{##1}%
6183 \renewcommand*{\acronymfont}[1]{##1}%
6184 \renewcommand*{\acrpluralsuffix}{\glacrpluralsuffix}%

```

Don't use footnotes for \acrfull:

```

6185 \renewcommand*{\acrfullfmt}[3]{%
6186 \glslink[##1]{##2}{\acronymfont{\glentryshort{##2}}##3\space
6187 (\glentrylong{##2})}%
6188 \renewcommand*{\Acrfullfmt}[3]{%
6189 \glslink[##1]{##2}{\acronymfont{\glentryshort{##2}}##3\space
6190 (\glentrylong{##2})}%
6191 \renewcommand*{\ACRfullfmt}[3]{%
6192 \glslink[##1]{##2}{%
6193 \mfirstucMakeUppercase{\acronymfont{\glentryshort{##2}}##3\space
6194 (\glentrylong{##2})}}}%
6195 \renewcommand*{\acrfullplfmt}[3]{%
6196 \glslink[##1]{##2}{\acronymfont{\glentryshortpl{##2}}##3\space
6197 (\glentrylongpl{##2})}%
6198 \renewcommand*{\Acrfullplfmt}[3]{%
6199 \glslink[##1]{##2}{\acronymfont{\glentryshortpl{##2}}##3\space
6200 (\glentrylongpl{##2})}%
6201 \renewcommand*{\ACRfullplfmt}[3]{%
6202 \glslink[##1]{##2}{%
6203 \mfirstucMakeUppercase{\acronymfont{\glentryshortpl{##2}}##3\space
6204 (\glentrylongpl{##2})}}}%

```

Similarly for \glentryfull etc:

```

6205 \renewcommand*{\glentryfull}[1]{%
6206 \acronymfont{\glentryshort{##1}}\space(\glentrylong{##1})}%
6207 \renewcommand*{\Glsentryfull}[1]{%
6208 \acronymfont{\Glsentryshort{##1}}\space(\glentrylong{##1})}%
6209 \renewcommand*{\glentryfullpl}[1]{%
6210 \acronymfont{\glentryshortpl{##1}}\space(\glentrylongpl{##1})}%
6211 \renewcommand*{\Glsentryfullpl}[1]{%
6212 \acronymfont{\Glsentryshortpl{##1}}\space(\glentrylongpl{##1})}%
6213 }

```

footnote-sc \textsc{<short>}\footnote{<long>} acronym style.

```

6214 \newacronymstyle{footnote-sc}%
6215 {%
6216 \GlsUseAcrEntryDisplayStyle{footnote}%
6217 }%

```

```

6218 {%
6219   \GlsUseAcrStyleDefs{footnote}%
6220   \renewcommand{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}
6221   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6222   \renewcommand*\{acrpluralsuffix}{\glsupacrpluralsuffix}%
6223 }%

```

footnote-sm \textsmaller{<short>}\footnote{<long>} acronym style.

```

6224 \newacronymstyle{footnote-sm}%
6225 {%
6226   \GlsUseAcrEntryDispStyle{footnote}%
6227 }%
6228 {%
6229   \GlsUseAcrStyleDefs{footnote}%
6230   \renewcommand{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}
6231   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6232   \renewcommand*\{acrpluralsuffix}{\glsacrpluralsuffix}%
6233 }%

```

footnote-desc <short>\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

6234 \newacronymstyle{footnote-desc}%
6235 {%
6236   \GlsUseAcrEntryDispStyle{footnote}%
6237 }%
6238 {%
6239   \GlsUseAcrStyleDefs{footnote}%
6240   \renewcommand*\{GenericAcronymFields}{}%
6241   \renewcommand*\{acronymsort}[2]{##2}%
6242   \renewcommand*\{acronymentry}[1]{%
6243     \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6244 }

```

footnote-sc-desc \textsc{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

6245 \newacronymstyle{footnote-sc-desc}%
6246 {%
6247   \GlsUseAcrEntryDispStyle{footnote-sc}%
6248 }%
6249 {%
6250   \GlsUseAcrStyleDefs{footnote-sc}%
6251   \renewcommand*\{GenericAcronymFields}{}%
6252   \renewcommand*\{acronymsort}[2]{##2}%
6253   \renewcommand*\{acronymentry}[1]{%
6254     \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6255 }

```

footnote-sm-desc \textsmaller{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

6256 \newacronymstyle{footnote-sm-desc}%
6257 {%
6258   \GlsUseAcrEntryDispStyle{footnote-sm}%
6259 }%
6260 {%
6261   \GlsUseAcrStyleDefs{footnote-sm}%
6262   \renewcommand*{\GenericAcronymFields}{}%
6263   \renewcommand*{\acronymsort}[2]{##2}%
6264   \renewcommand*{\acronymentry}[1]{%
6265     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6266 }

```

fineAcronymSynonyms

```

6267 \newcommand*{\DefineAcronymSynonyms}{%

```

Short form

\acs

```

6268   \let\acs\acrshort

```

First letter uppercase short form

\Acs

```

6269   \let\Acs\Acrshort

```

Plural short form

\acsp

```

6270   \let\acsp\acrshortpl

```

First letter uppercase plural short form

\Acsp

```

6271   \let\Acsp\Acrshortpl

```

Long form

\acl

```

6272   \let\acl\aclong

```

Plural long form

\aclp

```

6273   \let\aclp\aclongpl

```

First letter upper case long form

\Acl

```

6274   \let\Acl\Aclong

```

First letter upper case plural long form

```

\Aclp
6275 \let\Aclp\Acrlongpl

Full form

\acf
6276 \let\acf\acrfull

Plural full form

\acfp
6277 \let\acfp\acrfullpl

First letter upper case full form

\Acf
6278 \let\Acf\Acrfull

First letter upper case plural full form

\Acfp
6279 \let\Acfp\Acrfullpl

Standard form

\ac
6280 \let\ac\gls

First upper case standard form

\Ac
6281 \let\Ac\Gls

Standard plural form

\acp
6282 \let\acp\glspl

Standard first letter upper case plural form

\Acp
6283 \let\Acp\Glspl
6284 }

Define synonyms if required
6285 \ifglsacrshortcuts
6286 \DefineAcronymSynonyms
6287 \fi

```

These commands for setting the style are now deprecated but are kept for backward compatibility.

AcronymDisplayStyle Sets the default acronym display style for given glossary.

```
6288 \newcommand*{\SetDefaultAcronymDisplayStyle}[1]{%
6289   \def\glsentryfmt[#1]{\glsentryfmt}%
6290 }
```

DefaultNewAcronymDef Sets up the acronym definition for the default style. The information is provided by the tokens `\glslabeltok`, `\glsshorttok`, `\glslongtok` and `\glskeylisttok`.

```
6291 \newcommand*{\DefaultNewAcronymDef}{%
6292   \edef\@do@newglossaryentry{%
6293     \noexpand\newglossaryentry{\the\glslabeltok}%
6294     {%
6295       type=\acronymtype,%
6296       name={\the\glsshorttok},%
6297       sort={\the\glsshorttok},%
6298       text={\the\glsshorttok},%
6299       first={\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
6300       plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6301       firstplural={\acrfullformat{\noexpand\expandonce\noexpand\@glo@longpl}%
6302                    {\noexpand\expandonce\noexpand\@glo@shortpl}},%
6303       short={\the\glsshorttok},%
6304       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6305       long={\the\glslongtok},%
6306       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6307       description={\the\glslongtok},%
6308       descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
```

Remaining options specified by the user:

```
6309     \the\glskeylisttok
6310   }%
6311 }%
6312 \let\@org@gls@assign@firstpl\gls@assign@firstpl
6313 \let\@org@gls@assign@plural\gls@assign@plural
6314 \let\@org@gls@assign@descplural\gls@assign@descplural
6315 \def\gls@assign@firstpl##1##2{%
6316   \@@gls@expand@field{##1}{firstpl}{##2}%
6317 }%
6318 \def\gls@assign@plural##1##2{%
6319   \@@gls@expand@field{##1}{plural}{##2}%
6320 }%
6321 \def\gls@assign@descplural##1##2{%
6322   \@@gls@expand@field{##1}{descplural}{##2}%
6323 }%
6324 \@do@newglossaryentry
6325 \let\gls@assign@firstpl\@org@gls@assign@firstpl
6326 \let\gls@assign@plural\@org@gls@assign@plural
6327 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
6328 }
```

DefaultAcronymStyle Set up the default acronym style:

```
6329 \newcommand*\SetDefaultAcronymStyle{%
```

Set the display style:

```
6330 \@for\@gls@type:=\@glsacronymlists\do{%
6331 \SetDefaultAcronymDisplayStyle{\@gls@type}%
6332 }%
```

Set up the definition of `\newacronym`:

```
6333 \renewcommand{\newacronym}[4][\]{%
```

If user is just using the main glossary and hasn't identified it as a list of acronyms, then update. (This is done to ensure backwards compatibility with versions prior to 2.04).

```
6334 \ifx\@glsacronymlists\@empty
6335 \def\@glo@type{\acronymtype}%
6336 \setkeys{glossentry}{##1}%
6337 \DeclareAcronymList{\@glo@type}%
6338 \SetDefaultAcronymDisplayStyle{\@glo@type}%
6339 \fi
6340 \glskeylisttok{##1}%
6341 \glslabeltok{##2}%
6342 \glsshorttok{##3}%
6343 \glslongtok{##4}%
6344 \newacronymhook
6345 \DefaultNewAcronymDef
6346 }%
6347 \renewcommand*\acrpluralsuffix{\glsacrpluralsuffix}%
6348 }
```

\acrfootnote Used by the footnote acronym styles.

```
6349 \newcommand*\acrfootnote[3]{\acrlinkfootnote{#1}{#2}{#3}}
```

\acrlinkfootnote

```
6350 \newcommand*\acrlinkfootnote[3]{%
6351 \footnote{\glslink{#1}{#2}{#3}}%
6352 }
```

\acrnoflinkfootnote

```
6353 \newcommand*\acrnoflinkfootnote[3]{%
6354 \footnote{#3}%
6355 }
```

AcronymDisplayStyle Sets the acronym display style for given glossary for the description and footnote combination.

```
6356 \newcommand*\SetDescriptionFootnoteAcronymDisplayStyle[1]{%
6357 \def\glsentryfmt[1]{%
```



```

6358 \ifdefempty\glscustomtext
6359 {%
6360 \ifglssused{\glslabel}%
6361 {%
6362 \acronymfont{\glsgenentryfmt}%
6363 }%
6364 {%
6365 \firstacronymfont{\glsgenentryfmt}%
6366 \ifglsshassymbol{\glslabel}%
6367 {%
6368 \expandafter\protect\expandafter\acrfootnote\expandafter
6369 {\@gls@link@opts}{\@gls@link@label}%
6370 {%
6371 \glsifplural
6372 {\glsentrysymbolplural{\glslabel}}%
6373 {\glsentrysymbol{\glslabel}}%
6374 }%
6375 }%
6376 }%
6377 }%
6378 {\glscustomtext\glsinsert}%
6379 }%
6380 }

```

otnoteNewAcronymDef

```

6381 \newcommand*{\DescriptionFootnoteNewAcronymDef}{%
6382 \edef\@do@newglossaryentry{%
6383 \noexpand\newglossaryentry{\the\glslabelltok}%
6384 {%
6385 type=\acronymtype,%
6386 name={\noexpand\acronymfont{\the\glsshorttok}},%
6387 sort={\the\glsshorttok},%
6388 first={\the\glsshorttok},%
6389 firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6390 text={\the\glsshorttok},%
6391 plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6392 short={\the\glsshorttok},%
6393 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6394 long={\the\glslongtok},%
6395 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6396 symbol={\the\glslongtok},%
6397 symbolplural={\noexpand\expandonce\noexpand\@glo@longpl},%
6398 \the\glskeylisttok
6399 }%
6400 }%
6401 \let\@org@gls@assign@firstpl\gls@assign@firstpl
6402 \let\@org@gls@assign@plural\gls@assign@plural
6403 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
6404 \def\gls@assign@firstpl##1##2{%

```

```

6405 \@@gls@expand@field{##1}{firstpl}{##2}%
6406 }%
6407 \def\gls@assign@plural##1##2{%
6408 \@@gls@expand@field{##1}{plural}{##2}%
6409 }%
6410 \def\gls@assign@symbolplural##1##2{%
6411 \@@gls@expand@field{##1}{symbolplural}{##2}%
6412 }%
6413 \do@newglossaryentry
6414 \let\gls@assign@plural\@org@gls@assign@plural
6415 \let\gls@assign@firstpl\@org@gls@assign@firstpl
6416 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
6417 }

```

footnoteAcronymStyle If a description and footnote are both required, store the long form in the symbol key. Store the short form in text key. Note that since the long form is stored in the symbol key, if you want the long form to appear in the list of acronyms, you need to use a glossary style that displays the symbol key.

```

6418 \newcommand*\SetDescriptionFootnoteAcronymStyle{%
6419 \renewcommand{\newacronym}[4][\]{%
6420 \ifx\@glsacronymlists\@empty
6421 \def\@glo@type{\acronymtype}%
6422 \setkeys{glossentry}{##1}%
6423 \DeclareAcronymList{\@glo@type}%
6424 \SetDescriptionFootnoteAcronymDisplayStyle{\@glo@type}%
6425 \fi
6426 \glskeylisttok{##1}%
6427 \glslabeltok{##2}%
6428 \glsshorttok{##3}%
6429 \glslongtok{##4}%
6430 \newacronymhook
6431 \DescriptionFootnoteNewAcronymDef
6432 }%

```

If footnote package option is specified, set the first use to append the long form (stored in symbol) as a footnote.

```

6433 \@for\@gls@type:=\@glsacronymlists\do{%
6434 \SetDescriptionFootnoteAcronymDisplayStyle{\@gls@type}%
6435 }%

```

Redefine `\acronymfont` if small caps required. The plural suffix is set in an upright font so that it remains in normal lower case, otherwise it looks as though it's part of the acronym.

```

6436 \ifglsacrsmallcaps
6437 \renewcommand*\acronymfont{[1]{\textsc{##1}}}%
6438 \renewcommand*\acrpluralsuffix{\glsupacrpluralsuffix}%
6439 \else
6440 \ifglsacrsmaller
6441 \renewcommand*\acronymfont{[1]{\textsmaller{##1}}}%

```

```

6442 \fi
6443 \fi

```

Check for package option clash

```

6444 \ifglsacrdua
6445 \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’
6446 can’t both be set}{}%
6447 \fi
6448 }%

```

AcronymDisplayStyle Sets the acronym display style for given glossary with description and dua combination.

```

6449 \newcommand*{\SetDescriptionDUAAcronymDisplayStyle}[1]{%
6450 \defglsentryfmt[#1]{\glsentryfmt}%
6451 }

```

ionDUANewAcronymDef

```

6452 \newcommand*{\DescriptionDUANewAcronymDef}{%
6453 \edef\@do@newglossaryentry{%
6454 \noexpand\newglossaryentry{\the\glslabeltok}%
6455 {%
6456 type=\acronymtype,%
6457 name={\the\glslongtok},%
6458 sort={\the\glslongtok},%
6459 text={\the\glslongtok},%
6460 first={\the\glslongtok},%
6461 plural={\noexpand\expandonce\noexpand\@glo@longpl},%
6462 firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
6463 short={\the\glsshorttok},%
6464 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6465 long={\the\glslongtok},%
6466 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6467 symbol={\the\glsshorttok},%
6468 symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6469 \the\glskeylisttok
6470 }%
6471 }%
6472 \let\@org@gls@assign@firstpl\gls@assign@firstpl
6473 \let\@org@gls@assign@plural\gls@assign@plural
6474 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
6475 \def\gls@assign@firstpl##1##2{%
6476 \@@gls@expand@field{##1}{firstpl}{##2}%
6477 }%
6478 \def\gls@assign@plural##1##2{%
6479 \@@gls@expand@field{##1}{plural}{##2}%
6480 }%
6481 \def\gls@assign@symbolplural##1##2{%
6482 \@@gls@expand@field{##1}{symbolplural}{##2}%
6483 }%

```

```

6484 \do@newglossaryentry
6485 \let\gls@assign@firstpl\@org@gls@assign@firstpl
6486 \let\gls@assign@plural\@org@gls@assign@plural
6487 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
6488 }

```

tionDUAAcronymStyle Description, don't use acronym and no footnote. Note that the short form is stored in the symbol key, so if the short form needs to be displayed in the glossary, use a style the displays the symbol.

```

6489 \newcommand*\SetDescriptionDUAAcronymStyle{%
6490   \ifglsacrsmallcaps
6491     \PackageError{glossaries}{Option clash: 'smallcaps' and 'dua'
6492       can't both be set}{}%
6493   \else
6494     \ifglsacrsmaller
6495       \PackageError{glossaries}{Option clash: 'smaller' and 'dua'
6496         can't both be set}{}%
6497     \fi
6498   \fi
6499   \renewcommand{\newacronym}[4][{}]{%
6500     \ifx\@glsacronymlists\@empty
6501       \def\@glo@type{\acronymtype}%
6502       \setkeys{glossentry}{##1}%
6503       \DeclareAcronymList{\@glo@type}%
6504       \SetDescriptionDUAAcronymDisplayStyle{\@glo@type}%
6505     \fi
6506     \glskeylisttok{##1}%
6507     \glslabeltok{##2}%
6508     \glsshorttok{##3}%
6509     \glslongtok{##4}%
6510     \newacronymhook
6511     \DescriptionDUANewAcronymDef
6512   }%

```

Set display.

```

6513 \@for\@gls@type:=\@glsacronymlists\do{%
6514   \SetDescriptionDUAAcronymDisplayStyle{\@gls@type}%
6515 }%
6516 }%

```

AcronymDisplayStyle Sets the acronym display style for given glossary using the description setting (but not footnote or dua).

```

6517 \newcommand*\SetDescriptionAcronymDisplayStyle}[1]{%
6518   \defglsentryfmt[#1]{%
6519     \ifdefempty\glscustomtext
6520     {%
6521       \ifglsused{\glslabel}%
6522       {%

```

Move the inserted text outside of \acronymfont

```

6523     \let\gls@org@insert\glsinsert
6524     \let\glsinsert\@empty
6525     \acronymfont{\gls@org@insert
6526 }%
6527 {%
6528     \gls@entryfmt
6529     \ifgls@has@symbol{\gls@label}%
6530     {%
6531         \gls@ifplural
6532         {%
6533             \def\@glo@symbol{\gls@entrysymbolplural{\gls@label}}%
6534         }%
6535         {%
6536             \def\@glo@symbol{\gls@entrysymbol{\gls@label}}%
6537         }%
6538         \space(\protect\firstacronymfont
6539         {\gls@caps@case
6540         {\@glo@symbol}
6541         {\@glo@symbol}
6542         {\mfirstucMakeUppercase{\@glo@symbol}}})%
6543     }%
6544 }%
6545 }%
6546 }%
6547 {\gls@customtext\glsinsert}%
6548 }%
6549 }

```

ip tionNewAcronymDef

```

6550 \newcommand*{\DescriptionNewAcronymDef}{%
6551     \edef\@do@newglossaryentry{%
6552         \noexpand\newglossaryentry{\the\gls@labeltok}%
6553         {%
6554             type=\acronymtype,%
6555             name={\noexpand
6556                 \acrnameformat{\the\glsshorttok}{\the\gls@longtok}},%
6557             sort={\the\glsshorttok},%
6558             first={\the\gls@longtok},%
6559             firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
6560             text={\the\glsshorttok},%
6561             plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6562             short={\the\glsshorttok},%
6563             shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6564             long={\the\gls@longtok},%
6565             longplural={\the\gls@longtok\noexpand\acrpluralsuffix},%
6566             symbol={\noexpand\@glo@text},%
6567             symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6568             \the\gls@keylisttok}%

```

```

6569 }%
6570 \let\@org@gl@s@assign@firstpl\gl@s@assign@firstpl
6571 \let\@org@gl@s@assign@plural\gl@s@assign@plural
6572 \let\@org@gl@s@assign@symbolplural\gl@s@assign@symbolplural
6573 \def\gl@s@assign@firstpl##1##2{%
6574   \@@gl@s@expand@field{##1}{firstpl}{##2}%
6575 }%
6576 \def\gl@s@assign@plural##1##2{%
6577   \@@gl@s@expand@field{##1}{plural}{##2}%
6578 }%
6579 \def\gl@s@assign@symbolplural##1##2{%
6580   \@@gl@s@expand@field{##1}{symbolplural}{##2}%
6581 }%
6582 \do@newglossaryentry
6583 \let\gl@s@assign@firstpl\@org@gl@s@assign@firstpl
6584 \let\gl@s@assign@plural\@org@gl@s@assign@plural
6585 \let\gl@s@assign@symbolplural\@org@gl@s@assign@symbolplural
6586 }

```

DescriptionAcronymStyle Option description is used, but not dua or footnote. Store long form in first key and short form in text and symbol key. The name is stored using `\acrnameformat` to allow the user to override the way the name is displayed in the list of acronyms.

```

6587 \newcommand*{\SetDescriptionAcronymStyle}{%
6588   \renewcommand{\newacronym}[4][[]]{%
6589     \ifx\@gl@sacronymlists\@empty
6590       \def\@glo@type{\acronymtype}%
6591       \setkeys{glossentry}{##1}%
6592       \DeclareAcronymList{\@glo@type}%
6593       \SetDescriptionAcronymDisplayStyle{\@glo@type}%
6594     \fi
6595     \gl@keylisttok{##1}%
6596     \gl@labeltok{##2}%
6597     \gl@shorttok{##3}%
6598     \gl@longtok{##4}%
6599     \newacronymhook
6600     \DescriptionNewAcronymDef
6601   }%

```

Set display.

```

6602 \@for\@gl@s@type:=\@gl@sacronymlists\do{%
6603   \SetDescriptionAcronymDisplayStyle{\@gl@s@type}%
6604 }%

```

Redefine `\acronymfont` if small caps required. The plural suffix is set in an upright font so that it remains in normal lower case, otherwise it looks as though it's part of the acronym.

```

6605 \ifgl@smallcaps
6606   \renewcommand{\acronymfont}[1]{\textsc{##1}}

```

```

6607 \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6608 \else
6609 \ifglsmaller
6610 \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
6611 \fi
6612 \fi
6613 }%

```

AcronymDisplayStyle Sets the acronym display style for given glossary with footnote setting (but not description or dua).

```

6614 \newcommand*{\SetFootnoteAcronymDisplayStyle}[1]{%
6615 \defglentryfmt[#1]{%

```

```

6616 \ifdefempty\glscustomtext
6617 {%

```

Move the inserted text outside of \acronymfont

```

6618 \let\gls@org@insert\glsinsert
6619 \let\glsinsert\@empty
6620 \ifglused{\glslabel}%
6621 {%
6622 \acronymfont{\glsgenentryfmt}\gls@org@insert
6623 }%
6624 {%
6625 \firstacronymfont{\glsgenentryfmt}\gls@org@insert
6626 \ifglshaslong{\glslabel}%
6627 {%
6628 \expandafter\protect\expandafter\acrfootnote\expandafter
6629 {\@gls@link@opts}{\@gls@link@label}%
6630 {%
6631 \glsifplural
6632 {\glsentrylongpl{\glslabel}}%
6633 {\glsentrylong{\glslabel}}%
6634 }%
6635 }%
6636 }%
6637 }%
6638 }%
6639 {\glscustomtext\glsinsert}%
6640 }%
6641 }

```

FootnoteNewAcronymDef

```

6642 \newcommand*{\FootnoteNewAcronymDef}{%
6643 \edef\@do@newglossaryentry{%
6644 \noexpand\newglossaryentry{\the\glslabeltok}%
6645 {%
6646 type=\acronymtype,%
6647 name={\noexpand\acronymfont{\the\glsshorttok}},%

```

```

6648     sort={\the\glsshorttok},%
6649     text={\the\glsshorttok},%
6650     plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6651     first={\the\glsshorttok},%
6652     firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6653     short={\the\glsshorttok},%
6654     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6655     long={\the\glslongtok},%
6656     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6657     description={\the\glslongtok},%
6658     descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
6659     \the\glskeylisttok
6660 }%
6661 }%
6662 \let\@org@gls@assign@plural\gls@assign@plural
6663 \let\@org@gls@assign@firstpl\gls@assign@firstpl
6664 \let\@org@gls@assign@descplural\gls@assign@descplural
6665 \def\gls@assign@firstpl##1##2{%
6666   \@@gls@expand@field{##1}{firstpl}{##2}%
6667 }%
6668 \def\gls@assign@plural##1##2{%
6669   \@@gls@expand@field{##1}{plural}{##2}%
6670 }%
6671 \def\gls@assign@descplural##1##2{%
6672   \@@gls@expand@field{##1}{descplural}{##2}%
6673 }%
6674 \do@newglossaryentry
6675 \let\gls@assign@plural\@org@gls@assign@plural
6676 \let\gls@assign@firstpl\@org@gls@assign@firstpl
6677 \let\gls@assign@descplural\@org@gls@assign@descplural
6678 }

```

FootnoteAcronymStyle If footnote package option is specified, set the first use to append the long form (stored in description) as a footnote. Use the description key to store the long form.

```

6679 \newcommand*\SetFootnoteAcronymStyle{%
6680   \renewcommand{\newacronym}[4][\]{%
6681     \ifx\@glsacronymlists\@empty
6682       \def\@glo@type{\acronymtype}%
6683       \setkeys{glossentry}{##1}%
6684       \DeclareAcronymList{\@glo@type}%
6685       \SetFootnoteAcronymDisplayStyle{\@glo@type}%
6686     \fi
6687     \glskeylisttok{##1}%
6688     \glslabeltok{##2}%
6689     \glsshorttok{##3}%
6690     \glslongtok{##4}%
6691     \newacronymhook
6692     \FootnoteNewAcronymDef

```



```
6693 }%
```

Set display

```
6694 \@for\@gls@type:=\@gls@acronymlists\do{%
6695   \SetFootnoteAcronymDisplayStyle{\@gls@type}%
6696 }%
```

Redefine `\acronymfont` if small caps required. The plural suffix is set in an up-right font so that it remains in normal lower case, otherwise it looks as though it's part of the acronym.

```
6697 \ifglsacrsmallcaps
6698   \renewcommand*{\acronymfont}[1]{\textsc{##1}}%
6699   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6700 \else
6701   \ifglsacrsmaller
6702     \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
6703   \fi
6704 \fi
```

Check for option clash

```
6705 \ifglsacrdua
6706   \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’
6707     can’t both be set}{}%
6708 \fi
6709 }%
```

`\glsdoparenifnotempty` Do a space followed by the argument if the argument doesn't expand to empty or `\relax`. If argument isn't empty (or `\relax`), apply the macro to it given in the second argument.

```
6710 \DeclareRobustCommand*{\glsdoparenifnotempty}[2]{%
6711   \protected@edef\gls@tmp{#1}%
6712   \ifdefempty\gls@tmp
6713     {}%
6714   {%
6715     \ifx\gls@tmp\@gls@default@value
6716       \else
6717         \space (#2{#1})%
6718       \fi
6719     }%
6720 }
```

`AcronymDisplayStyle` Sets the acronym display style for given glossary where neither footnote nor description is required, but smallcaps or smaller specified.

```
6721 \newcommand*{\SetSmallAcronymDisplayStyle}[1]{%
6722   \defglsentryfmt[#1]{%
6723     \ifdefempty\glscustomtext
6724     {%
```

Move the inserted text outside of \acronymfont

```

6725 \let\gls@org@insert\glsinsert
6726 \let\glsinsert\@empty
6727 \ifglsused{\glslabel}%
6728 {%
6729 \acronymfont{\glsgetentryfmt}\gls@org@insert
6730 }%
6731 {%
6732 \glsgetentryfmt
6733 \ifglshassymbol{\glslabel}%
6734 {%
6735 \glsifplural
6736 {%
6737 \def\@glo@symbol{\glsentrysymbolplural{\glslabel}}%
6738 }%
6739 {%
6740 \def\@glo@symbol{\glsentrysymbol{\glslabel}}%
6741 }%
6742 \space
6743 (\glscapscase
6744 {\firstacronymfont{\@glo@symbol}}%
6745 {\firstacronymfont{\@glo@symbol}}%
6746 {\firstacronymfont{\mfirstucMakeUppercase{\@glo@symbol}}})%
6747 }%
6748 {}%
6749 }%
6750 }%
6751 {\glscustomtext\glsinsert}%
6752 }%
6753 }

```

\SmallNewAcronymDef

```

6754 \newcommand*{\SmallNewAcronymDef}{%
6755 \edef\@do@newglossaryentry{%
6756 \noexpand\newglossaryentry{\the\glslabeltok}%
6757 {%
6758 type=\acronymtype,%
6759 name={\noexpand\acronymfont{\the\glsshorttok}},%
6760 sort={\the\glsshorttok},%
6761 text={\the\glsshorttok},%

```

Default to the short plural.

```

6762 plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6763 first={\the\glslongtok},%

```

Default to the long plural.

```

6764 firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
6765 short={\the\glsshorttok},%
6766 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6767 long={\the\glslongtok},%

```

```

6768     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6769     description={\noexpand\@glo@first},%
6770     descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
6771     symbol={\the\glsshorttok},%

```

Default to the short plural.

```

6772     symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6773     \the\glskeylisttok
6774 }%
6775 }%
6776 \let\@org@gls@assign@firstpl\gls@assign@firstpl
6777 \let\@org@gls@assign@plural\gls@assign@plural
6778 \let\@org@gls@assign@descplural\gls@assign@descplural
6779 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
6780 \def\gls@assign@firstpl##1##2{%
6781     \@@gls@expand@field{##1}{firstpl}{##2}%
6782 }%
6783 \def\gls@assign@plural##1##2{%
6784     \@@gls@expand@field{##1}{plural}{##2}%
6785 }%
6786 \def\gls@assign@descplural##1##2{%
6787     \@@gls@expand@field{##1}{descplural}{##2}%
6788 }%
6789 \def\gls@assign@symbolplural##1##2{%
6790     \@@gls@expand@field{##1}{symbolplural}{##2}%
6791 }%
6792 \do@newglossaryentry
6793 \let\gls@assign@firstpl\@org@gls@assign@firstpl
6794 \let\gls@assign@plural\@org@gls@assign@plural
6795 \let\gls@assign@descplural\@org@gls@assign@descplural
6796 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
6797 }

```

etSmallAcronymStyle Neither footnote nor description required, but smallcaps or smaller specified.
Use the symbol key to store the short form and first to store the long form.

```

6798 \newcommand*\SetSmallAcronymStyle{%
6799     \renewcommand{\newacronym}[4][\]{%
6800         \ifx\@glsacronymlists\@empty
6801             \def\@glo@type{\acronymtype}%
6802             \setkeys{glossentry}{##1}%
6803             \DeclareAcronymList{\@glo@type}%
6804             \SetSmallAcronymDisplayStyle{\@glo@type}%
6805         \fi
6806         \glskeylisttok{##1}%
6807         \glslabeltok{##2}%
6808         \glsshorttok{##3}%
6809         \glslongtok{##4}%
6810         \newacronymhook
6811         \SmallNewAcronymDef
6812     }%

```

Change the display since first only contains long form.

```
6813 \@for\@gls@type:=\@glsacronymlists\do{%
6814   \SetSmallAcronymDisplayStyle{\@gls@type}%
6815 }%
```

Redefine \acronymfont if small caps required. The plural suffix is set in an up-right font so that it remains in normal lower case, otherwise it looks as though it's part of the acronym.

```
6816 \ifglsacrsmallcaps
6817   \renewcommand*{\acronymfont}[1]{\textsc{##1}}
6818   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6819 \else
6820   \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}
6821 \fi
```

check for option clash

```
6822 \ifglsacrdua
6823   \ifglsacrsmallcaps
6824     \PackageError{glossaries}{Option clash: ‘smallcaps’ and ‘dua’
6825       can’t both be set}{}%
6826   \else
6827     \PackageError{glossaries}{Option clash: ‘smaller’ and ‘dua’
6828       can’t both be set}{}%
6829   \fi
6830 \fi
6831 }%
```

\SetDUADisplayStyle Sets the acronym display style for given glossary with dua setting.

```
6832 \newcommand*{\SetDUADisplayStyle}[1]{%
6833   \defglsentryfmt[##1]{\glsngenentryfmt}%
6834 }
```

\DUANewAcronymDef

```
6835 \newcommand*{\DUANewAcronymDef}{%
6836   \edef\@do@newglossaryentry{%
6837     \noexpand\newglossaryentry{\the\glslabeltok}%
6838     {%
6839       type=\acronymtype,%
6840       name={\the\glsshorttok},%
6841       text={\the\glslongtok},%
6842       first={\the\glslongtok},%
6843       plural={\noexpand\expandonce\noexpand\@glo@longpl},%
6844       firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
6845       short={\the\glsshorttok},%
6846       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6847       long={\the\glslongtok},%
6848       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6849       description={\the\glslongtok},%
6850       descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
```

```

6851     symbol={\the\glsshorttok},%
6852     symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6853     \the\glskeylisttok
6854   }%
6855 }%
6856 \let\@org@gls@assign@firstpl\gls@assign@firstpl
6857 \let\@org@gls@assign@plural\gls@assign@plural
6858 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
6859 \let\@org@gls@assign@descplural\gls@assign@descplural
6860 \def\gls@assign@firstpl##1##2{%
6861   \@@gls@expand@field{##1}{firstpl}{##2}%
6862 }%
6863 \def\gls@assign@plural##1##2{%
6864   \@@gls@expand@field{##1}{plural}{##2}%
6865 }%
6866 \def\gls@assign@symbolplural##1##2{%
6867   \@@gls@expand@field{##1}{symbolplural}{##2}%
6868 }%
6869 \def\gls@assign@descplural##1##2{%
6870   \@@gls@expand@field{##1}{descplural}{##2}%
6871 }%
6872 \do@newglossaryentry
6873 \let\gls@assign@firstpl\@org@gls@assign@firstpl
6874 \let\gls@assign@plural\@org@gls@assign@plural
6875 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
6876 \let\gls@assign@descplural\@org@gls@assign@descplural
6877 }

```

\SetDUASyle Always expand acronyms.

```

6878 \newcommand*\SetDUASyle{%
6879   \renewcommand{\newacronym}[4][[]]{%
6880     \ifx\@glsacronymlists\@empty
6881       \def\@glo@type{\acronymtype}%
6882       \setkeys{glossentry}{##1}%
6883       \DeclareAcronymList{\@glo@type}%
6884       \SetDUADisplayStyle{\@glo@type}%
6885     \fi
6886     \glskeylisttok{##1}%
6887     \glslabeltok{##2}%
6888     \glsshorttok{##3}%
6889     \glslongtok{##4}%
6890     \newacronymhook
6891     \DUANewAcronymDef
6892   }%
6893   \Set the display
6894   \@for\@gls@type:=\@glsacronymlists\do{%
6895     \SetDUADisplayStyle{\@gls@type}%
6896   }%
6897 }

```

\SetAcronymStyle

```
6897 \newcommand*{\SetAcronymStyle}{%
6898   \SetDefaultAcronymStyle
6899   \ifglacrdescription
6900     \ifglacrfootnote
6901       \SetDescriptionFootnoteAcronymStyle
6902     \else
6903       \ifglacrdua
6904         \SetDescriptionDUAAcronymStyle
6905       \else
6906         \SetDescriptionAcronymStyle
6907     \fi
6908   \fi
6909 \else
6910   \ifglacrfootnote
6911     \SetFootnoteAcronymStyle
6912   \else
6913     \ifthenelse{\boolean{glacrsmalldcaps}}{\OR
6914       \boolean{glacrsmalld}}{%
6915       {%
6916         \SetSmallAcronymStyle
6917       }%
6918     {%
6919       \ifglacrdua
6920         \SetDUASStyle
6921     \fi
6922   }%
6923 \fi
6924 \fi
6925 }
```

Set the acronym style according to the package options

6926 \SetAcronymStyle

Allow user to define their own custom acronyms. (For compatibility with versions before v3.0, the short form is stored in the user1 key, the plural short form is stored in the user2 key, the long form is stored in the user3 key and the plural long form is stored in the user4 key.) Defaults to displaying only the acronym with the long form as the description.

tCustomDisplayStyle Sets the acronym display style.

```
6927 \newcommand*{\SetCustomDisplayStyle}[1]{%
6928   \defglsentryfmt[#1]{\glsgenentryfmt}%
6929 }
```

CustomAcronymFields

```
6930 \newcommand*{\CustomAcronymFields}{%
6931   name={\the\glsshorttok},%
6932   description={\the\glslongtok},%
```

```

6933 first={\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
6934 firstplural={\acrfullformat
6935   {\noexpand\glsentrylongpl{\the\glslabeltok}}}%
6936   {\noexpand\glsentryshortpl{\the\glslabeltok}}},%

6937 text={\the\glsshorttok},%
6938 plural={\the\glsshorttok\noexpand\acrpluralsuffix}%
6939 }

```

CustomNewAcronymDef

```

6940 \newcommand*{\CustomNewAcronymDef}{%
6941   \protected@edef\@do@newglossaryentry{%
6942     \noexpand\newglossaryentry{\the\glslabeltok}%
6943     {%
6944       type=\acronymtype,%
6945       short={\the\glsshorttok},%
6946       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6947       long={\the\glslongtok},%
6948       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6949       user1={\the\glsshorttok},%
6950       user2={\the\glsshorttok\noexpand\acrpluralsuffix},%
6951       user3={\the\glslongtok},%
6952       user4={\the\glslongtok\noexpand\acrpluralsuffix},%
6953       \CustomAcronymFields,%
6954       \the\glskeylisttok
6955     }%
6956   }%
6957   \@do@newglossaryentry
6958 }

```

\SetCustomStyle

```

6959 \newcommand*{\SetCustomStyle}{%
6960   \renewcommand{\newacronym}[4][\]{%
6961     \ifx\@glsacronymlists\@empty
6962       \def\@glo@type{\acronymtype}%
6963       \setkeys{glossentry}{##1}%
6964       \DeclareAcronymList{\@glo@type}%
6965       \SetCustomDisplayStyle{\@glo@type}%
6966     \fi
6967     \glskeylisttok{##1}%
6968     \glslabeltok{##2}%
6969     \glsshorttok{##3}%
6970     \glslongtok{##4}%
6971     \newacronymhook
6972     \CustomNewAcronymDef
6973   }%

```

Set the display

```

6974   \@for\@gls@type:=\@glsacronymlists\do{%
6975     \SetCustomDisplayStyle{\@gls@type}%

```

```
6976 }%
6977 }
```

1.18 Predefined Glossary Styles

The glossaries bundle comes with some predefined glossary styles. These need to be loaded now for the style option to use them.

First, the glossary hyper-navigation commands need to be loaded.

```
6978 \RequirePackage{glossary-hypernav}
```

The styles that use list-like environments. These are not loaded if the `nolist` option is used:

```
6979 \@gls@loadlist
```

The styles that use the `longtable` environment. These are not loaded if the `no-long package` option is used.

```
6980 \@gls@loadlong
```

The styles that use the `supertabular` environment. These are not loaded if the `nosuper` package option is used or if the package isn't installed.

```
6981 \@gls@loadsuper
```

The tree-like styles. These are not loaded if the `notree` package option is used.

```
6982 \@gls@loadtree
```

The default glossary style is set according to the `style` package option, but can be overridden by `\glossarystyle`. The required style must be defined at this point.

```
6983 \ifx\@glossary@default@style\relax
6984 \else
6985   \setglossarystyle{\@glossary@default@style}
6986 \fi
```

1.19 Debugging Commands

`\showgloparent` `\showgloparent{<label>}`

```
6987 \newcommand*{\showgloparent}[1]{%
6988   \expandafter\show\csname glo@\glsdetoklabel{#1}@parent\endcsname
6989 }
```

`\showglolevel` `\showglolevel{<label>}`

```
6990 \newcommand*{\showglolevel}[1]{%
6991   \expandafter\show\csname glo@\glsdetoklabel{#1}@level\endcsname
6992 }
```


`\showglotext` `\showglotext{<label>}`

```
6993 \newcommand*{\showglotext}[1]{%
6994   \expandafter\show\csname glo@\glsdetoklabel{#1}@text\endcsname
6995 }
```

`\showgloplural` `\showgloplural{<label>}`

```
6996 \newcommand*{\showgloplural}[1]{%
6997   \expandafter\show\csname glo@\glsdetoklabel{#1}@plural\endcsname
6998 }
```

`\showglofirst` `\showglofirst{<label>}`

```
6999 \newcommand*{\showglofirst}[1]{%
7000   \expandafter\show\csname glo@\glsdetoklabel{#1}@first\endcsname
7001 }
```

`\showglofirstpl` `\showglofirstpl{<label>}`

```
7002 \newcommand*{\showglofirstpl}[1]{%
7003   \expandafter\show\csname glo@\glsdetoklabel{#1}@firstpl\endcsname
7004 }
```

`\showglotype` `\showglotype{<label>}`

```
7005 \newcommand*{\showglotype}[1]{%
7006   \expandafter\show\csname glo@\glsdetoklabel{#1}@type\endcsname
7007 }
```

`\showglocounter` `\showglocounter{<label>}`

```
7008 \newcommand*{\showglocounter}[1]{%
7009   \expandafter\show\csname glo@\glsdetoklabel{#1}@counter\endcsname
7010 }
```

\showglouserii \showglouserii{\label{}}

```
7011 \newcommand*{\showglouserii}[1]{%
7012   \expandafter\show\csname glo@\glsdetoklabel{#1}@userii\endcsname
7013 }
```

\showglouseriii \showglouseriii{\label{}}

```
7014 \newcommand*{\showglouseriii}[1]{%
7015   \expandafter\show\csname glo@\glsdetoklabel{#1}@useriii\endcsname
7016 }
```

\showglouseriv \showglouseriv{\label{}}

```
7017 \newcommand*{\showglouseriv}[1]{%
7018   \expandafter\show\csname glo@\glsdetoklabel{#1}@useriv\endcsname
7019 }
```

\showglouserv \showglouserv{\label{}}

```
7020 \newcommand*{\showglouserv}[1]{%
7021   \expandafter\show\csname glo@\glsdetoklabel{#1}@userv\endcsname
7022 }
```

\showglouservi \showglouservi{\label{}}

```
7023 \newcommand*{\showglouservi}[1]{%
7024   \expandafter\show\csname glo@\glsdetoklabel{#1}@uservi\endcsname
7025 }
```

\showglouservi \showglouservi{\label{}}

```
7026 \newcommand*{\showglouservi}[1]{%
7027   \expandafter\show\csname glo@\glsdetoklabel{#1}@uservi\endcsname
7028 }
```

`\showglo`name `\showglo`name{*\label*}

```
7029 \newcommand*{\showglo
```

name}[1]{%
7030 \expandafter\show\csname glo@\glsdetoklabel{#1}@name\endcsname
7031 }

`\showglo`desc `\showglo`desc{*\label*}

```
7032 \newcommand*{\showglo
```

desc}[1]{%
7033 \expandafter\show\csname glo@\glsdetoklabel{#1}@desc\endcsname
7034 }

`\showglo`descplural `\showglo`descplural{*\label*}

```
7035 \newcommand*{\showglo
```

descplural}[1]{%
7036 \expandafter\show\csname glo@\glsdetoklabel{#1}@descplural\endcsname
7037 }

`\showglo`sort `\showglo`sort{*\label*}

```
7038 \newcommand*{\showglo
```

sort}[1]{%
7039 \expandafter\show\csname glo@\glsdetoklabel{#1}@sort\endcsname
7040 }

`\showglo`symbol `\showglo`symbol{*\label*}

```
7041 \newcommand*{\showglo
```

symbol}[1]{%
7042 \expandafter\show\csname glo@\glsdetoklabel{#1}@symbol\endcsname
7043 }

`\showglo`symbolplural `\showglo`symbolplural{*\label*}

```
7044 \newcommand*{\showglo
```

symbolplural}[1]{%
7045 \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolplural\endcsname
7046 }

`\showgloshort` `\showgloshort{<label>}`

```
7047 \newcommand*{\showgloshort}[1]{%
7048   \expandafter\show\csname glo@\glsdetoklabel{#1}@short\endcsname
7049 }
```

`\showglolong` `\showglolong{<label>}`

```
7050 \newcommand*{\showglolong}[1]{%
7051   \expandafter\show\csname glo@\glsdetoklabel{#1}@long\endcsname
7052 }
```

`\showgloindex` `\showgloindex{<label>}`

```
7053 \newcommand*{\showgloindex}[1]{%
7054   \expandafter\show\csname glo@\glsdetoklabel{#1}@index\endcsname
7055 }
```

`\showgloflag` `\showgloflag{<label>}`

```
7056 \newcommand*{\showgloflag}[1]{%
7057   \expandafter\show\csname ifglo@\glsdetoklabel{#1}@flag\endcsname
7058 }
```

`\showgloloclist` `\showgloloclist{<label>}`

```
7059 \newcommand*{\showgloloclist}[1]{%
7060   \expandafter\show\csname glo@\glsdetoklabel{#1}@loclist\endcsname
7061 }
```

`\showacronymlists` `\showacronymlists`

Show list of glossaries that have been flagged as a list of acronyms.

```
7062 \newcommand*{\showacronymlists}{%
7063   \show\@glsacronymlists
7064 }
```

`\showglossaries` `\showglossaries`

Show list of defined glossaries.

```
7065 \newcommand*{\showglossaries}{%  
7066   \show\@glo@types  
7067 }
```

`\showglossaryin` `\showglossaryin{<glossary-label>}`

Show the ‘in’ extension for the given glossary.

```
7068 \newcommand*{\showglossaryin}[1]{%  
7069   \expandafter\show\csname @glotype@#1@in\endcsname  
7070 }
```

`\showglossaryout` `\showglossaryout{<glossary-label>}`

Show the ‘out’ extension for the given glossary.

```
7071 \newcommand*{\showglossaryout}[1]{%  
7072   \expandafter\show\csname @glotype@#1@out\endcsname  
7073 }
```

`\showglossarytitle` `\showglossarytitle{<glossary-label>}`

Show the title for the given glossary.

```
7074 \newcommand*{\showglossarytitle}[1]{%  
7075   \expandafter\show\csname @glotype@#1@title\endcsname  
7076 }
```

`\showglossarycounter` `\showglossarycounter{<glossary-label>}`

Show the counter for the given glossary.

```
7077 \newcommand*{\showglossarycounter}[1]{%  
7078   \expandafter\show\csname @glotype@#1@counter\endcsname  
7079 }
```

`\showglossaryentries` `\showglossaryentries{<glossary-label>}`

Show the list of entry labels for the given glossary.

```
7080 \newcommand*{\showglossaryentries}[1]{%  
7081   \expandafter\show\csname glolist@#1\endcsname  
7082 }
```

1.20 Compatibility with version 2.07 and below

In order to fix some bugs in v3.0, it was necessary to change the way information is written to the `glo` file, which also meant a change in the format of the Xindy style file. The compatibility option is meant for documents that use a customised Xindy style file with `\noist`. With the compatibility option, hopefully xindy will still be able to process the old document, but the bugs will remain. The issues in versions 2.07 and below:

- With xindy, the counter used by the entry was hard-coded into the Xindy style file. This meant that you couldn't use the counter to swap counters.
- With both xindy and makeindex, if used with hyperref and `\theH{counter}` was different to `\thecounter`, the link in the location number would be undefined.

```
7083 \csname ifglscompatible-2.07\endcsname
7084 \RequirePackage{glossaries-compatible-207}
7085 \fi
```

2 Prefix Support (glossaries-prefix Code)

This package provides a means of adding prefixes to your glossary entries. For example, you may want to use “a `\gls{<label>}`” on first use but use “an `\gls{<label>}`” on subsequent use.

```
7086 \NeedsTeXFormat{LaTeX2e}
7087 \ProvidesPackage{glossaries-prefix}[2014/07/30 v4.08 (NLCT)]

Pass all options to glossaries:
7088 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}

Process options:
7089 \ProcessOptions

Load glossaries:
7090 \RequirePackage{glossaries}

Add the new keys:
7091 \define@key{glossentry}{prefixfirst}{\def\@glo@entryprefixfirst{#1}}%
7092 \define@key{glossentry}{prefixfirstplural}{\def\@glo@entryprefixfirstplural{#1}}%
7093 \define@key{glossentry}{prefix}{\def\@glo@entryprefix{#1}}%
7094 \define@key{glossentry}{prefixplural}{\def\@glo@entryprefixplural{#1}}%

Add them to \@gls@keymap:
7095 \appto\@gls@keymap{,%
7096   {prefixfirst}{prefixfirst},%
7097   {prefixfirstplural}{prefixfirstplural},%
7098   {prefix}{prefix},%
7099   {prefixplural}{prefixplural}}%
7100 }
```

Set the default values:

```

7101 \appto\@newglossaryentryprehook{%
7102   \def\@glo@entryprefix{}%
7103   \def\@glo@entryprefixplural{}%
7104   \let\@glo@entryprefixfirst\@gls@default@value
7105   \let\@glo@entryprefixfirstplural\@gls@default@value
7106 }

```

Set the assignment code:

```

7107 \appto\@newglossaryentryposthook{%
7108   \gls@assign@field{\@glo@label}{prefix}{\@glo@entryprefix}%
7109   \gls@assign@field{\@glo@label}{prefixplural}{\@glo@entryprefixplural}%

```

If prefixfirst has not been supplied, make it the same as prefix.

```

7110   \expandafter\gls@assign@field\expandafter
7111     {\csname glo@\@glo@label @prefix\endcsname}{\@glo@label}{prefixfirst}%
7112     {\@glo@entryprefixfirst}%

```

If prefixfirstplural has not been supplied, make it the same as prefixplural.

```

7113   \expandafter\gls@assign@field\expandafter
7114     {\csname glo@\@glo@label @prefixplural\endcsname}{\@glo@label}%
7115     {prefixfirstplural}{\@glo@entryprefixfirstplural}%
7116 }

```

Define commands to access these fields:

glsentryprefixfirst

```

7117 \newcommand*\{glsentryprefixfirst}[1]{\csuse{glo@#1@prefixfirst}}

```

ryprefixfirstplural

```

7118 \newcommand*\{glsentryprefixfirstplural}[1]{\csuse{glo@#1@prefixfirstplural}}

```

\glsentryprefix

```

7119 \newcommand*\{glsentryprefix}[1]{\csuse{glo@#1@prefix}}

```

lsentryprefixplural

```

7120 \newcommand*\{glsentryprefixplural}[1]{\csuse{glo@#1@prefixplural}}

```

Now for the initial upper case variants:

Glsentryprefixfirst

```

7121 \newrobustcmd*\{Glsentryprefixfirst}[1]{%
7122   \protected@edef\@glo@text{\csname glo@#1@prefixfirst\endcsname}%
7123   \xmakefirstuc\@glo@text
7124 }

```

ryprefixfirstplural

```

7125 \newrobustcmd*\{Glsentryprefixfirstplural}[1]{%
7126   \protected@edef\@glo@text{\csname glo@#1@prefixfirstplural\endcsname}%
7127   \xmakefirstuc\@glo@text
7128 }

```

\Glsentryprefix

```
7129 \newrobustcmd*{\Glsentryprefix}[1]{%
7130   \protected@edef\@glo@text{\csname glo@#1@prefix\endcsname}%
7131   \xmakefirstuc\@glo@text
7132 }
```

\Glsentryprefixplural

```
7133 \newrobustcmd*{\Glsentryprefixplural}[1]{%
7134   \protected@edef\@glo@text{\csname glo@#1@prefixplural\endcsname}%
7135   \xmakefirstuc\@glo@text
7136 }
```

Define commands to determine if the prefix keys have been set:

\ifglshasprefix

```
7137 \newcommand*{\ifglshasprefix}[3]{%
7138   \ifcempty{glo@#1@prefix}%
7139   {#3}%
7140   {#2}%
7141 }
```

\ifglshasprefixplural

```
7142 \newcommand*{\ifglshasprefixplural}[3]{%
7143   \ifcempty{glo@#1@prefixplural}%
7144   {#3}%
7145   {#2}%
7146 }
```

\ifglshasprefixfirst

```
7147 \newcommand*{\ifglshasprefixfirst}[3]{%
7148   \ifcempty{glo@#1@prefixfirst}%
7149   {#3}%
7150   {#2}%
7151 }
```

\ifglshasprefixfirstplural

```
7152 \newcommand*{\ifglshasprefixfirstplural}[3]{%
7153   \ifcempty{glo@#1@prefixfirstplural}%
7154   {#3}%
7155   {#2}%
7156 }
```

Define commands that insert the prefix before commands like \gls:

\pgls

```
7157 \newrobustcmd{\pgls}{\@gls@hyp@opt\@pgls}
```


\@pgls Unstarred version.

```
7158 \newcommand*{\@pgls}[2] [] {%
7159   \new@ifnextchar [%
7160     {\@pgls@{#1}{#2}}%
7161     {\@pgls@{#1}{#2} []}%
7162 }
```

\@pgls@ Read in the final optional argument:

```
7163 \def\@pgls@#1#2[#3] {%
7164   \glsdoifexists{#2}%
7165   {%
7166     \ifglsused{#2}%
7167     {%
7168       \glsentryprefix{#2}%
7169     }%
7170     {%
7171       \glsentryprefixfirst{#2}%
7172     }%
7173     \@gls@{#1}{#2}[#3]%
7174   }%
7175 }
```

Similarly for the plural version:

\pglsp1

```
7176 \newrobustcmd{\pglsp1}{\@gls@hyp@opt\@pglsp1}
```

\@pglsp1 Unstarred version.

```
7177 \newcommand*{\@pglsp1}[2] [] {%
7178   \new@ifnextchar [%
7179     {\@pglsp1@{#1}{#2}}%
7180     {\@pglsp1@{#1}{#2} []}%
7181 }
```

\@pglsp1@ Read in the final optional argument:

```
7182 \def\@pglsp1@#1#2[#3] {%
7183   \glsdoifexists{#2}%
7184   {%
7185     \ifglsused{#2}%
7186     {%
7187       \glsentryprefixplural{#2}%
7188     }%
7189     {%
7190       \glsentryprefixfirstplural{#2}%
7191     }%
7192     \@glspl@{#1}{#2}[#3]%
7193   }%
7194 }
```

Now for the first letter upper case versions:

\PglS

```
7195 \newrobustcmd{\PglS}{\@gls@hyp@opt\@PglS}
```

\@PglS Unstarred version.

```
7196 \newcommand*{\@PglS}[2][\]{%
7197   \new@ifnextchar[%
7198     {\@PglS@{#1}{#2}}%
7199     {\@PglS@{#1}{#2}[]}%
7200 }
```

\@PglS@ Read in the final optional argument:

```
7201 \def\@PglS@#1#2[#3]{%
7202   \glsdoifexists{#2}%
7203   {%
7204     \ifglSused{#2}%
7205     {%
7206       \ifglshasprefix{#2}%
7207       {%
7208         \Glsentryprefix{#2}%
7209         \@gls@{#1}{#2}[#3]%
7210       }%
7211       {\@Gls@{#1}{#2}[#3]}%
7212     }%
7213   }%
7214   \ifglshasprefixfirst{#2}%
7215   {%
7216     \Glsentryprefixfirst{#2}%
7217     \@gls@{#1}{#2}[#3]%
7218   }%
7219   {\@Gls@{#1}{#2}[#3]}%
7220 }%
7221 }%
7222 }
```

Similarly for the plural version:

\PglSpl

```
7223 \newrobustcmd{\PglSpl}{\@gls@hyp@opt\@PglSpl}
```

\@PglSpl Unstarred version.

```
7224 \newcommand*{\@PglSpl}[2][\]{%
7225   \new@ifnextchar[%
7226     {\@PglSpl@{#1}{#2}}%
7227     {\@PglSpl@{#1}{#2}[]}%
7228 }
```

\@Pglsp1@ Read in the final optional argument:

```
7229 \def\@Pglsp1@#1#2[#3]{%
7230   \glsdoifexists{#2}%
7231   {%
7232     \ifglsused{#2}%
7233     {%
7234       \ifglshasprefixplural{#2}%
7235       {%
7236         \Glsentryprefixplural{#2}%
7237         \@glsp1@{#1}{#2}[#3]%
7238       }%
7239       {\@Glspl@{#1}{#2}[#3]}%
7240     }%
7241     {%
7242       \ifglshasprefixfirstplural{#2}%
7243       {%
7244         \Glsentryprefixfirstplural{#2}%
7245         \@glsp1@{#1}{#2}[#3]%
7246       }%
7247       {\@Glspl@{#1}{#2}[#3]}%
7248     }%
7249   }%
7250 }
```

Finally the all upper case versions:

\PGLS

```
7251 \newrobustcmd{\PGLS}{\@gls@hyp@opt\@PGLS}
```

\@PGLS Unstarred version.

```
7252 \newcommand*{\@PGLS}[2][ ]{%
7253   \new@ifnextchar[%
7254   {\@PGLS@{#1}{#2}}%
7255   {\@PGLS@{#1}{#2}[ ]}%
7256 }
```

\@PGLS@ Read in the final optional argument:

```
7257 \def\@PGLS@#1#2[#3]{%
7258   \glsdoifexists{#2}%
7259   {%
7260     \ifglsused{#2}%
7261     {%
7262       \mfirstucMakeUppercase{\glsentryprefix{#2}}%
7263     }%
7264     {%
7265       \mfirstucMakeUppercase{\glsentryprefixfirst{#2}}%
7266     }%
7267     \@GLS@{#1}{#2}[#3]%
7268   }
```

```

7268 }%
7269 }

```

Plural version:

`\PGLSp1`

```

7270 \newrobustcmd{\PGLSp1}{\@gls@hyp@opt\PGLSp1}

```

`\@PGLSp1` Unstarred version.

```

7271 \newcommand*{\@PGLSp1}[2][{}]{%
7272   \new@ifnextchar[%
7273     {\@PGLSp1@{#1}{#2}}%
7274     {\@PGLSp1@{#1}{#2}[]}%
7275 }

```

`\@PGLSp1@` Read in the final optional argument:

```

7276 \def\@PGLSp1@#1#2[#3]{%
7277   \glsdoifexists{#2}%
7278   {%
7279     \ifglsused{#2}%
7280     {%
7281       \mfirstucMakeUppercase{\glsentryprefixplural{#2}}%
7282     }%
7283     {%
7284       \mfirstucMakeUppercase{\glsentryprefixfirstplural{#2}}%
7285     }%
7286     \@GLSp1@{#1}{#2}[#3]%
7287   }%
7288 }

```

3 Mfirstuc Documented Code

```

7289 \NeedsTeXFormat{LaTeX2e}
7290 \ProvidesPackage{mfirstuc}[2014/07/30 v1.09 (NLCT)]

```

Requires etoolbox:

```

7291 \RequirePackage{etoolbox}

```

`\makefirstuc` Syntax:

`\makefirstuc{<text>}`

Makes the first letter uppercase, but will skip initial control sequences if they are followed by a group and make the first thing in the group uppercase, unless the group is empty. Thus `\makefirstuc{abc}` will produce: `Abc`, `\makefirstuc{\ae bc}` will produce: `Æbc`, but `\makefirstuc{\emph{abc}}` will produce `Abc`. This is required by `\Gls` and `\Glspl`.

```

7292 \newif\if@glscs

```

```

7293 \newtoks\@glsmfirst
7294 \newtoks\@glsmrest
7295 \newrobustcmd*{\makefirstuc}[1]{%
7296   \def\@gls@argi{#1}%
7297   \ifx\@gls@argi\@empty
       If the argument is empty, do nothing.
7298   \else

7299     \def\@gls@tmp{\ #1}%
7300     \@onelevel@sanitize\@gls@tmp
7301     \expandafter\@gls@checkcs\@gls@tmp\relax\relax
7302     \if@glscs
7303       \@gls@getbody #1{}\@nil
7304       \ifx\@gls@rest\@empty
7305         \glsmakefirstuc{#1}%
7306       \else
7307         \expandafter\@gls@split\@gls@rest\@nil
7308         \ifx\@gls@first\@empty
7309           \glsmakefirstuc{#1}%
7310         \else
7311           \expandafter\@glsmfirst\expandafter{\@gls@first}%
7312           \expandafter\@glsmrest\expandafter{\@gls@rest}%
7313           \edef\@gls@domfirstuc{\noexpand\@gls@body
7314             {\noexpand\glsmakefirstuc\the\@glsmfirst}%
7315             \the\@glsmrest}%
7316           \@gls@domfirstuc
7317         \fi
7318       \fi
7319     \else
7320       \glsmakefirstuc{#1}%
7321     \fi
7322   \fi
7323 }

       Put first argument in \@gls@first and second argument in \@gls@rest:
7324 \def\@gls@split#1#2\@nil{%
7325   \def\@gls@first{#1}\def\@gls@rest{#2}%
7326 }

7327 \def\@gls@checkcs#1 #2#3\relax{%
7328   \def\@gls@argi{#1}\def\@gls@argii{#2}%
7329   \ifx\@gls@argi\@gls@argii
7330     \@glscstrue
7331   \else
7332     \@glscsfalse
7333   \fi
7334 }

```

\@gls@makefirstuc Make first thing upper case:

```

7335 \def\@gls@makefirstuc#1{\mfirstucMakeUppercase #1}

```

irstucMakeUppercase Allow user to replace \MakeUppercase with another case changing command.

```
7336 \newcommand*{\mfirstucMakeUppercase}{\MakeUppercase}
```

\glsmakefirstuc Provide a user command to make it easier to customise.

```
7337 \newcommand*{\glsmakefirstuc}[1]{\@gls@makefirstuc{#1}}
```

Get the first grouped argument and store in \@gls@body.

```
7338 \def\@gls@getbody#1#\def\@gls@body{#1}\@gls@gobbletonil}
```

Scoop up everything to \@nil and store in \@gls@rest:

```
7339 \def\@gls@gobbletonil#1\@nil{\def\@gls@rest{#1}}
```

\xmakefirstuc Expand argument once before applying \makefirstuc (added v1.01).

```
7340 \newcommand*{\xmakefirstuc}[1]{%
```

```
7341 \expandafter\makefirstuc\expandafter{#1}}
```

\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).

```
7342 \newrobustcmd*{\capitalisewords}[1]{%
```

```
7343 \def\gls@add@space{}%
```

```
7344 \let\@mfu@domakefirstuc\makefirstuc
```

```
7345 \let\@mfu@checkword\@gobble
```

```
7346 \mfu@capitalisewords#1 \@nil\mfu@endcap
```

```
7347 }
```

```
7348 \def\mfu@capitalisewords#1 #2\mfu@endcap{%
```

```
7349 \def\mfu@cap@first{#1}%
```

```
7350 \def\mfu@cap@second{#2}%
```

```
7351 \gls@add@space
```

```
7352 \@mfu@checkword{#1}%
```

```
7353 \@mfu@domakefirstuc{#1}%
```

```
7354 \def\gls@add@space{ }%
```

```
7355 \ifx\mfu@cap@second\@nnil
```

```
7356 \let\next\mfu@cap\mfu@noop
```

```
7357 \else
```

```
7358 \let\next\mfu@cap\mfu@capitalisewords
```

```
7359 \let\@mfu@checkword\mfu@checkword
```

```
7360 \fi
```

```
7361 \next\mfu@cap#2\mfu@endcap
```

```
7362 }
```

```
7363 \def\mfu@noop#1\mfu@endcap{}
```

\mfu@checkword Check if word should be capitalised.

```
7364 \newcommand*{\mfu@checkword}[1]{%
```

```
7365 \ifinlist{#1}{\@mfu@nocaplist}%
```

```
7366 {%
```

```
7367 \let\@mfu@domakefirstuc\@firstofone
```

```
7368 }%
```

```
7369 {%
```

```

7370 \let\@mfu@domakefirstuc\makefirstuc
7371 }%
7372 }

```

\@mfu@nocaplist List of words that shouldn't be capitalised.

```

7373 \newcommand*\@mfu@nocaplist{}

```

\MFUnocap Provide the user with a means to add a word to the list.

```

7374 \newcommand*\MFUnocap[1]{\listadd{\@mfu@nocaplist}{#1}}

```

\gMFUnocap Global version.

```

7375 \newcommand*\gMFUnocap[1]{\listgadd{\@mfu@nocaplist}{#1}}

```

\MFUclear Clear the list

```

7376 \newcommand*\MFUclear{}\renewcommand*\@mfu@nocaplist{}}

```

\xcapitalisewords Short-cut command:

```

7377 \newcommand*\xcapitalisewords[1]{%
7378 \expandafter\capitalisewords\expandafter{#1}%
7379 }

```

4 Mfirstuc-english Documented Code

```

7380 \NeedsTeXFormat{LaTeX2e}
7381 \ProvidesPackage{mfirstuc-english}[2014/07/30 v1.0 (NLCT)]

```

Load mfirstuc if not already loaded:

```

7382 \RequirePackage{mfirstuc}

```

Add no-cap words. (List isn't a complete list.)

```

7383 \MFUnocap{a}
7384 \MFUnocap{an}
7385 \MFUnocap{and}
7386 \MFUnocap{but}
7387 \MFUnocap{for}
7388 \MFUnocap{in}
7389 \MFUnocap{of}
7390 \MFUnocap{or}
7391 \MFUnocap{no}
7392 \MFUnocap{nor}
7393 \MFUnocap{so}
7394 \MFUnocap{some}
7395 \MFUnocap{the}
7396 \MFUnocap{with}
7397 \MFUnocap{yet}

```

5 Glossary Styles

5.1 Glossary hyper-navigation definitions (glossary-hypernav package)

Package Definition:

```
7398 \ProvidesPackage{glossary-hypernav}[2013/11/14 v4.0 (NLCT)]
```

The commands defined in this package are provided to help navigate around the groups within a glossary (see [subsection 1.15.](#)) `\printglossary` (and `\printglossaries`) set `\@glo@type` to the label of the current glossary. This is used to create a unique hypertarget in the event of multiple glossaries.

```
\glsnavhyperlink[<type>]{<label>}{<text>}
```

This command makes `<text>` a hyperlink to the glossary group whose label is given by `<label>` for the glossary given by `<type>`.

```
\glsnavhyperlink
```

```
7399 \newcommand*{\glsnavhyperlink}[3][\@glo@type]{%
7400   \edef\gls@grplabel{#2}\protected@edef\gls@grptitle{#3}%
7401   \@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
```

```
7402 \newcommand*{\glsnavhypertarget}[3][\@glo@type]{%
  Add this group to the aux file for re-run check.
7403   \protected@write\@auxout{}{\string\@gls@hypergroup{#1}{#2}}%
  Add the target.
7404   \@glstarget{glsn:#1@#2}{#3}%
  Check list of know groups to determine if a re-run is required.
7405   \expandafter\let
7406     \expandafter\@gls@list\csname @gls@hypergroup@#1\endcsname
  Iterate through list and terminate loop if this group is found.
7407   \@for\@gls@elem:=\@gls@list\do{%
7408     \ifthenelse{\equal{\@gls@elem}{#2}}{\@endfortrue}{}%
  Check if list terminated prematurely.
7409   \if@endfor
7410   \else
```


This group was not included in the list, so issue a warning.

```

7411 \GlossariesWarningNoLine{Navigation panel
7412   for glossary type ‘#1’^^Jmissing group ‘#2’}%
7413 \gdef\gls@hypergroupprerun{%
7414   \GlossariesWarningNoLine{Navigation panel
7415   has changed. Rerun LaTeX}}%
7416 \fi
7417 }

```

`\gls@hypergroupprerun` Give a warning at the end if re-run required

```

7418 \let\gls@hypergroupprerun\relax
7419 \AtEndDocument{\gls@hypergroupprerun}

```

`\@gls@hypergroup` This adds to (or creates) the command `\@gls@hypergrouplist@<glossary type>` which lists all groups for a given glossary, so that the navigation bar only contains those groups that are present. However it requires at least 2 runs to ensure the information is up-to-date.

```

7420 \newcommand*{\@gls@hypergroup}[2]{%
7421 \ifundefined{\@gls@hypergrouplist@#1}{%
7422   \expandafter\xdef\csname @gls@hypergrouplist@#1\endcsname{#2}%
7423 }{%
7424   \expandafter\let\expandafter\@gls@tmp
7425     \csname @gls@hypergrouplist@#1\endcsname
7426   \expandafter\xdef\csname @gls@hypergrouplist@#1\endcsname{%
7427     \@gls@tmp,#2}%
7428 }%
7429 }

```

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`

```

7430 \newcommand*{\glsnavigation}{%
7431 \def\@gls@between{}%
7432 \@ifundefined{\@gls@hypergrouplist@\@glo@type}{%
7433   \def\@gls@list{}%
7434 }{%
7435   \expandafter\let\expandafter\@gls@list
7436     \csname @gls@hypergrouplist@\@glo@type\endcsname
7437 }%
7438 \@for\@gls@tmp:=\@gls@list\do{%
7439   \@gls@between
7440   \@gls@getgrouptitle{\@gls@tmp}{\@gls@grptitle}%

```

```

7441 \glsnavhyperlink{\@gls@tmp}{\@gls@grptitle}%
7442 \let\@gls@between\glshypernavsep%
7443 }%
7444 }

```

`\glshypernavsep` Separator for the hyper navigation bar.

```

7445 \newcommand*{\glshypernavsep}{\space\textbar\space}

```

The `\glssymbolnav` produces a simple navigation set of links for just the symbol and number groups. This used to be used at the start of `\glsnavigation`. This command is no longer needed.

`\glssymbolnav`

```

7446 \newcommand*{\glssymbolnav}{%
7447 \glsnavhyperlink{glssymbols}{\glsgetgrouptitle{glssymbols}}%
7448 \glshypernavsep
7449 \glsnavhyperlink{glsnumbers}{\glsgetgrouptitle{glsnumbers}}%
7450 \glshypernavsep
7451 }

```

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.

```

7452 \ProvidesPackage{glossary-inline}[2013/11/14 v4.0 (NLCT)]

```

`inline` Define the inline style.

```

7453 \newglossarystyle{inline}{%

```

Start of glossary sets up first empty separator between entries. (This is then changed by `\glossentry`)

```

7454 \renewenvironment{theglossary}%
7455 {%
7456 \def\gls@inlinesep{}%
7457 \def\gls@inlinesubsep{}%
7458 \def\gls@inlinepostchild{}%
7459 }%
7460 {\glspostinline}%

```

No header:

```

7461 \renewcommand*{\glossaryheader}{}%

```

No group headings (if heading is required, add `\glsinlinedopostchild` to start definition in case heading follows a child entry):

```

7462 \renewcommand*{\glsgroupheading}[1]{}%

```

Just display separator followed by name and description:

```

7463 \renewcommand{\glossentry}[2]{%
7464 \glsinlinedopostchild
7465 \gls@inlinesep

```

```

7466 \glsentryitem{##1}%
7467 \glsinlinenameformat{##1}{%
7468 \glossentryname{##1}%
7469 }%
7470 \ifglstdescsuppressed{##1}%
7471 {%
7472 \glsinlineemptydescformat
7473 {%
7474 \glossentrysymbol{##1}%
7475 }%
7476 {%
7477 ##2%
7478 }%
7479 }%
7480 {%
7481 \ifglshasdesc{##1}%
7482 {\glsinlinedescformat{\glossentrydesc{##1}}{\glossentrysymbol{##1}}{##2}}%
7483 {\glsinlineemptydescformat{\glossentrysymbol{##1}}{##2}}%
7484 }%
7485 \ifglshaschildren{##1}%
7486 {%
7487 \glsresetsubentrycounter
7488 \glsinlineparentchildseparator
7489 \def\gls@inlinesubsep{}%
7490 \def\gls@inlinepostchild{\glsinlinepostchild}%
7491 }%
7492 {}%
7493 \def\gls@inlinesep{\glsinlineseparator}%
7494 }%

```

Sub-entries display description:

```

7495 \renewcommand{\subglossentry}[3]{%
7496 \gls@inlinesubsep%
7497 \glsinlinesubnameformat{##2}{%
7498 \glossentryname{##2}}%
7499 \glssubentryitem{##2}%
7500 \glsinlinesubdescformat{\glossentrydesc{##2}}{\glossentrysymbol{##2}}{##3}%
7501 \def\gls@inlinesubsep{\glsinlinesubseparator}%
7502 }%

```

Nothing special between groups:

```

7503 \renewcommand*{\glsgroupskip}{}%
7504 }

```

lsinlinedopostchild

```

7505 \newcommand*{\glsinlinedopostchild}{%
7506 \gls@inlinepostchild
7507 \def\gls@inlinepostchild{}%
7508 }

```

`\glsinlineseparator` Separator to use between entries.
7509 `\newcommand*{\glsinlineseparator}{;\space}`

`\glsinlinesubseparator` Separator to use between sub-entries.
7510 `\newcommand*{\glsinlinesubseparator}{,\space}`

`\glsinlineparentchildseparator` Separator to use between parent and children.
7511 `\newcommand*{\glsinlineparentchildseparator}{:\space}`

`\glsinlinepostchild` Hook to use between child and next entry
7512 `\newcommand*{\glsinlinepostchild}{}`

`\glspostinline` Terminator for inline glossary.
7513 `\newcommand*{\glspostinline}{\glspostdescription\space}`

`\glsinlinenameformat` Formats the name of the entry (first argument label, second argument name):
7514 `\newcommand*{\glsinlinenameformat}[2]{\glstarget{#1}{#2}}`

`\glsinlinedescformat` Formats the entry's description, symbol and location list:
7515 `\newcommand*{\glsinlinedescformat}[3]{\space#1}`

`\glsinlineemptydescformat` Formats the entry's symbol and location list when the description is empty:
7516 `\newcommand*{\glsinlineemptydescformat}[2]{}`

`\glsinlinesubnameformat` Formats the name of the subentry (first argument label, second argument name):
7517 `\newcommand*{\glsinlinesubnameformat}[2]{\glstarget{#1}{}}`

`\glsinlinesubdescformat` Formats the subentry's description, symbol and location list:
7518 `\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.

7519 `\ProvidesPackage{glossary-list}[2013/11/14 v4.0 (NLCT)]`

`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.

7520 `\newglossarystyle{list}{%`

Use description environment:

```
7521 \renewenvironment{theglossary}%  
7522   {\begin{description}}{\end{description}}%
```

No header at the start of the environment:

```
7523 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
7524 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries start a new item in the list:

```
7525 \renewcommand*{\glossentry}[2]{%  
7526   \item[\glsentryitem{##1}]%  
7527     \glstarget{##1}{\glossentryname{##1}}]  
7528     \glossentrydesc{##1}\glspostdescription\space ##2}%
```

Sub-entries continue on the same line:

```
7529 \renewcommand*{\subglossentry}[3]{%  
7530   \glssubentryitem{##2}%  
7531   \glstarget{##2}{\strut}%  
7532   \glossentrydesc{##2}\glspostdescription\space ##3.}%  
7533 % \end{macrocode}  
7534 % Add vertical space between groups:  
7535 %\changes{3.03}{2012/09/21}{added check for glsnogroupskip}  
7536 % \begin{macrocode}  
7537 \renewcommand*{\glsgroupskip}{\ifglsgroupskip\else\indexspace\fi}%  
7538 }
```

listgroup The listgroup style is like the list style, but the glossary groups have headings.

```
7539 \newglossarystyle{listgroup}{%
```

Base it on the list style:

```
7540 \setglossarystyle{list}%
```

Each group has a heading:

```
7541 \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.

```
7542 \newglossarystyle{listhypergroup}{%
```

Base it on the list style:

```
7543 \setglossarystyle{list}%
```

Add navigation links at the start of the environment:

```
7544 \renewcommand*{\glossaryheader}{%  
7545   \item[\glsnavigation]}%
```

Each group has a heading with a hypertarget:

```
7546 \renewcommand*{\glsgroupheading}[1]{%  
7547   \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.

```
7548 \newglossarystyle{altlist}{%
```

Base it on the list style:

```
7549 \setglossarystyle{list}%
```

Main (level 0) entries start a new item in the list with a line break after the entry name:

```
7550 \renewcommand*{\glossentry}[2]{%
7551 \item[\glsentryitem{##1}%
7552 \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.

```
7553 \mbox{} \par \nobreak \afterheading
7554 \glossentrydesc{##1} \glspostdescription \space ##2}%
```

Sub-entries start a new paragraph:

```
7555 \renewcommand{\subglossentry}[3]{%
7556 \par
7557 \glssubentryitem{##2}%
7558 \glstarget{##2}{\strut} \glossentrydesc{##2} \glspostdescription \space ##3}%
7559 }
```

altlistgroup The altlistgroup glossary style is like the altlist style, but the glossary groups have headings.

```
7560 \newglossarystyle{altlistgroup}{%
```

Base it on the altlist style:

```
7561 \setglossarystyle{altlist}%
```

Each group has a heading:

```
7562 \renewcommand*{\glsgroupheading}[1]{\item[\glsgrouptitle{##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.

```
7563 \newglossarystyle{altlisthypergroup}{%
```

Base it on the altlist style:

```
7564 \setglossarystyle{altlist}%
```

Add navigation links at the start of the environment:

```
7565 \renewcommand*{\glossaryheader}{%
7566 \item[\glsnavigation]}%
```

Each group has a heading with a hypertarget:

```
7567 \renewcommand*{\glsgroupheading}[1]{%
7568 \item[\glsnavhypertarget{##1}{\glsgrouptitle{##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.

```
7569 \newglossarystyle{listdotted}{%
```

Base it on the list style:

```
7570 \setglossarystyle{list}{%
```

Each main (level 0) entry starts a new item:

```
7571 \renewcommand*{\glossentry}[2]{%
7572 \item[]\makebox[\glslistdottedwidth][l]{%
7573 \glentryitem{##1}%
7574 \glstarget{##1}{\glossentryname{##1}}%
7575 \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##1}}%
```

Sub entries have the same format as main entries:

```
7576 \renewcommand*{\subglossentry}[3]{%
7577 \item[]\makebox[\glslistdottedwidth][l]{%
7578 \glssubentryitem{##2}%
7579 \glstarget{##2}{\glossentryname{##2}}%
7580 \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##2}}%
7581 }
```

`\glslistdottedwidth`

```
7582 \newlength\glslistdottedwidth
7583 \setlength{\glslistdottedwidth}{.5\hsize}
```

`sublistdotted` This style is similar to the `glostylelistdotted` style, except that the main entries just have the name displayed.

```
7584 \newglossarystyle{sublistdotted}{%
```

Base it on the `listdotted` style:

```
7585 \setglossarystyle{listdotted}{%
```

Main (level 0) entries just display the name:

```
7586 \renewcommand*{\glossentry}[2]{%
7587 \item[\glentryitem{##1}\glstarget{##1}{\glossentryname{##1}}}%
7588 }
```

5.4 Glossary Styles using `longtable` (the `glossary-long` package)

The glossary styles defined in the package used the `longtable` environment in the glossary.

```
7589 \ProvidesPackage{glossary-long}[2013/11/14 v4.0 (NLCT)]
```

Requires the package:

```
7590 \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`.)

```
7591 \ifundefined{glsdescwidth}{%
7592   \newlength{glsdescwidth
7593   \setlength{glsdescwidth}{0.6\hsize}
7594 }{}
```

`\glspagelistwidth` This is a length that governs the width of the page list column.

```
7595 \ifundefined{glspagelistwidth}{%
7596   \newlength{glspagelistwidth
7597   \setlength{glspagelistwidth}{0.1\hsize}
7598 }{}
```

`long` The long glossary style command which uses the `longtable` environment:

```
7599 \newglossarystyle{long}{%
```

Use `longtable` with two columns:

```
7600   \renewenvironment{theglossary}%
7601     {\begin{longtable}{lp{glsdescwidth}}}%
7602     {\end{longtable}}%
```

Do nothing at the start of the environment:

```
7603   \renewcommand*{\glossaryheader}{}%
```

No heading between groups:

```
7604   \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries displayed in a row:

```
7605   \renewcommand{\glossentry}[2]{%
7606     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
7607     \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline
7608   }%
```

Sub entries displayed on the following row without the name:

```
7609   \renewcommand{\subglossentry}[3]{%
7610     &
7611     \glssubentryitem{##2}%
7612     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
7613     ##3\tabularnewline
7614   }%
```

Blank row between groups:

```
7615   \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else &
7616   \tabularnewline\fi}%
7617 }
```

`longborder` The `longborder` style is like the above, but with horizontal and vertical lines:

```
7618 \newglossarystyle{longborder}{%
```


Base it on the `glostylelong` style:

```
7619 \setglossarystyle{long}%
```

Use `longtable` with two columns with vertical lines between each column:

```
7620 \renewenvironment{theglossary}{%  
7621 \begin{longtable}{|l|p{\glstdescwidth}|}{\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
7622 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%  
7623 }
```

`longheader` The `longheader` style is like the `long` style but with a header:

```
7624 \newglossarystyle{longheader}{%
```

Base it on the `glostylelong` style:

```
7625 \setglossarystyle{long}%
```

Set the table's header:

```
7626 \renewcommand*{\glossaryheader}{%  
7627 \bfseries \entryname & \bfseries \descriptionname\tabularnewline\endhead}%  
7628 }
```

`longheaderborder` The `longheaderborder` style is like the `long` style but with a header and border:

```
7629 \newglossarystyle{longheaderborder}{%
```

Base it on the `glostylelongborder` style:

```
7630 \setglossarystyle{longborder}%
```

Set the table's header and add horizontal line to table's foot:

```
7631 \renewcommand*{\glossaryheader}{%  
7632 \hline\bfseries \entryname & \bfseries  
7633 \descriptionname\tabularnewline\hline  
7634 \endhead  
7635 \hline\endfoot}%  
7636 }
```

`long3col` The `long3col` style is like `long` but with 3 columns

```
7637 \newglossarystyle{long3col}{%
```

Use a `longtable` with 3 columns:

```
7638 \renewenvironment{theglossary}{%  
7639 {\begin{longtable}{lp{\glstdescwidth}p{\glspagelistwidth}}}%  
7640 {\end{longtable}}%
```

No table header:

```
7641 \renewcommand*{\glossaryheader}{}%
```

No headings between groups:

```
7642 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```

7643 \renewcommand{\glossentry}[2]{%
7644   \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
7645   \glossentrydesc{##1} & ##2\tabularnewline
7646 }%
```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```

7647 \renewcommand{\subglossentry}[3]{%
7648   &
7649   \glssubentryitem{##2}%
7650   \glstarget{##2}{\strut}\glossentrydesc{##2} &
7651   ##3\tabularnewline
7652 }%
```

Blank row between groups:

```

7653 \renewcommand*{\glsgroupskip}{%
7654   \ifglsnogroupskip\else & &\tabularnewline\fi}%
7655 }
```

long3colborder The long3colborder style is like the long3col style but with a border:

```
7656 \newglossarystyle{long3colborder}{%
```

Base it on the glostylelong3col style:

```
7657 \setglossarystyle{long3col}%
```

Use a longtable with 3 columns with vertical lines around them:

```

7658 \renewenvironment{theglossary}%
7659   {\begin{longtable}{|l|p{\glsdscwidth}|p{\glspagelistwidth}|}}%
7660   {\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```

7661 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
7662 }
```

long3colheader The long3colheader style is like long3col but with a header row:

```
7663 \newglossarystyle{long3colheader}{%
```

Base it on the glostylelong3col style:

```
7664 \setglossarystyle{long3col}%
```

Set the table's header:

```

7665 \renewcommand*{\glossaryheader}{%
7666   \bfseries\entryname&\bfseries\descriptionname&
7667   \bfseries\pagelistname\tabularnewline\endhead}%
7668 }
```

long3colheaderborder The long3colheaderborder style is like the above but with a border

```
7669 \newglossarystyle{long3colheaderborder}{%
```

Base it on the `glostylelong3colborder` style:

```
7670 \setglossarystyle{long3colborder}{%
```

Set the table's header and add horizontal line at table's foot:

```
7671 \renewcommand*{\glossaryheader}{{%  
7672 \hline  
7673 \bfseries\entryname&\bfseries\descriptionname&  
7674 \bfseries\pagelistname\tabularnewline\hline\endhead  
7675 \hline\endfoot}%  
7676 }
```

`long4col` The `long4col` style has four columns where the third column contains the value of the associated symbol key.

```
7677 \newglossarystyle{long4col}{%
```

Use a `longtable` with 4 columns:

```
7678 \renewenvironment{theglossary}{%  
7679 {\begin{longtable}{llll}}%  
7680 {\end{longtable}}%
```

No table header:

```
7681 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
7682 \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):

```
7683 \renewcommand{\glossentry}[2]{%  
7684 \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &  
7685 \glossentrydesc{##1} &  
7686 \glossentrysymbol{##1} &  
7687 ##2\tabularnewline  
7688 }%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
7689 \renewcommand{\subglossentry}[3]{%  
7690 &  
7691 \glssubentryitem{##2}%  
7692 \glstarget{##2}{\strut}\glossentrydesc{##2} &  
7693 \glossentrysymbol{##2} & ##3\tabularnewline  
7694 }%
```

Blank row between groups:

```
7695 \renewcommand*{\glsgroupskip}{%  
7696 \ifglsgnোগroupskip\else & & \tabularnewline\fi}%  
7697 }
```

`long4colheader` The `long4colheader` style is like `long4col` but with a header row.

```
7698 \newglossarystyle{long4colheader}{%
```

Base it on the `glostylelong4col` style:

```
7699 \setglossarystyle{long4col}%
```

Table has a header:

```
7700 \renewcommand*{\glossaryheader}{%  
7701   \bfseries\entryname&\bfseries\descriptionname&  
7702   \bfseries \symbolname&  
7703   \bfseries\pagelistname\tabularnewline\endhead}%  
7704 }
```

`long4colborder` The `long4colborder` style is like `long4col` but with a border.

```
7705 \newglossarystyle{long4colborder}{%
```

Base it on the `glostylelong4col` style:

```
7706 \setglossarystyle{long4col}%
```

Use a longtable with 4 columns surrounded by vertical lines:

```
7707 \renewenvironment{theglossary}%  
7708   {\begin{longtable}{|l|l|l|l|}}%  
7709   {\end{longtable}}%
```

Add horizontal lines to the head and foot of the table:

```
7710 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%  
7711 }
```

`long4colheaderborder` The `long4colheaderborder` style is like the above but with a border.

```
7712 \newglossarystyle{long4colheaderborder}{%
```

Base it on the `glostylelong4col` style:

```
7713 \setglossarystyle{long4col}%
```

Use a longtable with 4 columns surrounded by vertical lines:

```
7714 \renewenvironment{theglossary}%  
7715   {\begin{longtable}{|l|l|l|l|}}%  
7716   {\end{longtable}}%
```

Add table header and horizontal line at the table's foot:

```
7717 \renewcommand*{\glossaryheader}{%  
7718   \hline\bfseries\entryname&\bfseries\descriptionname&  
7719   \bfseries \symbolname&  
7720   \bfseries\pagelistname\tabularnewline\hline\endhead  
7721   \hline\endfoot}%  
7722 }
```

`altlong4col` The `altlong4col` style is like the `long4col` style but can have multiline descriptions and page lists.

```
7723 \newglossarystyle{altlong4col}{%
```

Base it on the `glostylelong4col` style:

```
7724 \setglossarystyle{long4col}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
7725 \renewenvironment{theglossary}%
7726   {\begin{longtable}{lp{\glsdescwidth}lp{\glspagelistwidth}}}%
7727   {\end{longtable}}%
7728 }
```

altlong4colheader The altlong4colheader style is like altlong4col but with a header row.

```
7729 \newglossarystyle{altlong4colheader}{%
```

Base it on the glostylelong4colheader style:

```
7730 \setglossarystyle{long4colheader}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
7731 \renewenvironment{theglossary}%
7732   {\begin{longtable}{lp{\glsdescwidth}lp{\glspagelistwidth}}}%
7733   {\end{longtable}}%
7734 }
```

altlong4colborder The altlong4colborder style is like altlong4col but with a border.

```
7735 \newglossarystyle{altlong4colborder}{%
```

Base it on the glostylelong4colborder style:

```
7736 \setglossarystyle{long4colborder}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
7737 \renewenvironment{theglossary}%
7738   {\begin{longtable}{|lp{\glsdescwidth}|lp{\glspagelistwidth}|}}%
7739   {\end{longtable}}%
7740 }
```

ong4colheaderborder The altlong4colheaderborder style is like the above but with a header as well as a border.

```
7741 \newglossarystyle{altlong4colheaderborder}{%
```

Base it on the glostylelong4colheaderborder style:

```
7742 \setglossarystyle{long4colheaderborder}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
7743 \renewenvironment{theglossary}%
7744   {\begin{longtable}{|lp{\glsdescwidth}|lp{\glspagelistwidth}|}}%
7745   {\end{longtable}}%
7746 }
```

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.

7747 \ProvidesPackage{glossary-longragged}[2014/07/30 v4.08 (NLCT)]

Requires the package:

7748 \RequirePackage{array}

Requires the package:

7749 \RequirePackage{longtable}

`\glsdescwidth` This is a length that governs the width of the description column. This may have already been defined.

```
7750 \@ifundefined{glsdescwidth}{%
7751   \newlength{glsdescwidth
7752   \setlength{glsdescwidth}{0.6\hsize}
7753 }{}
```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```
7754 \@ifundefined{glspagelistwidth}{%
7755   \newlength{glspagelistwidth
7756   \setlength{glspagelistwidth}{0.1\hsize}
7757 }{}
```

`longragged` The longragged glossary style is like the long but uses ragged right formatting for the description column.

7758 \newglossarystyle{longragged}{%

Use longtable with two columns:

```
7759   \renewenvironment{theglossary}{%
7760     {\begin{longtable}{l>{\raggedright}p{glsdescwidth}}}%
7761     {\end{longtable}}%
```

Do nothing at the start of the environment:

7762 \renewcommand*{\glossaryheader}{}%

No heading between groups:

7763 \renewcommand*{\glsgroupheading}[1]{}%

Main (level 0) entries displayed in a row:

```
7764 \renewcommand{\glossentry}[2]{%
7765   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
7766   \glossentrydesc{##1}\glspostdescription\space ##2%
7767   \tabularnewline
7768 }%
```

Sub entries displayed on the following row without the name:

```

7769 \renewcommand{\subglossentry}[3]{%
7770     &
7771     \glssubentryitem{##2}%
7772     \glstarget{##2}{\strut}\glossentrydesc{##2}%
7773     \glspostdescription\space ##3%
7774     \tabularnewline
7775 }%
```

Blank row between groups:

```

7776 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else & \tabularnewline\fi}%
7777 }
```

longraggedborder The longraggedborder style is like the above, but with horizontal and vertical lines:

```

7778 \newglossarystyle{longraggedborder}{%
```

Base it on the glostylelongragged style:

```

7779 \setglossarystyle{longragged}%
```

Use longtable with two columns with vertical lines between each column:

```

7780 \renewenvironment{theglossary}{%
7781     \begin{longtable}{|l|>{\raggedright}p{\glsdescwidth}|}%
7782     {\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```

7783 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
7784 }
```

longraggedheader The longraggedheader style is like the longragged style but with a header:

```

7785 \newglossarystyle{longraggedheader}{%
```

Base it on the glostylelongragged style:

```

7786 \setglossarystyle{longragged}%
```

Set the table's header:

```

7787 \renewcommand*{\glossaryheader}{%
7788     \bfseries \entryname & \bfseries \descriptionname
7789     \tabularnewline\endhead}%
7790 }
```

longraggedheaderborder The longraggedheaderborder style is like the longragged style but with a header and border:

```

7791 \newglossarystyle{longraggedheaderborder}{%
```

Base it on the glostylelongraggedborder style:

```

7792 \setglossarystyle{longraggedborder}%
```

Set the table's header and add horizontal line to table's foot:

```

7793 \renewcommand*{\glossaryheader}{%
7794     \hline\bfseries \entryname & \bfseries \descriptionname
```

```

7795 \tabularnewline\hline
7796 \endhead
7797 \hline\endfoot}%
7798 }

```

longragged3col The longragged3col style is like longragged but with 3 columns

```
7799 \newglossarystyle{longragged3col}{%
```

Use a longtable with 3 columns:

```

7800 \renewenvironment{theglossary}%
7801 {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}%
7802 >{\raggedright}p{\glspagelistwidth}}}%
7803 {\end{longtable}}%

```

No table header:

```
7804 \renewcommand*{\glossaryheader}{}%
```

No headings between groups:

```
7805 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```

7806 \renewcommand{\glossentry}[2]{%
7807 \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
7808 \glossentrydesc{##1} & ##2\tabularnewline
7809 }%

```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```

7810 \renewcommand{\subglossentry}[3]{%
7811 &
7812 \glssubentryitem{##2}%
7813 \glstarget{##2}{\strut}\glossentrydesc{##2} &
7814 ##3\tabularnewline
7815 }%

```

Blank row between groups:

```

7816 \renewcommand*{\glsgroupskip}{%
7817 \ifglsgnoglobskip\else & &\tabularnewline\fi}%
7818 }

```

longragged3colborder The longragged3colborder style is like the longragged3col style but with a border:

```
7819 \newglossarystyle{longragged3colborder}{%
```

Base it on the glostylelongragged3col style:

```
7820 \setglossarystyle{longragged3col}%
```

Use a longtable with 3 columns with vertical lines around them:

```

7821 \renewenvironment{theglossary}%
7822 {\begin{longtable}{|l|>{\raggedright}p{\glsgdescwidth}|}%
7823 >{\raggedright}p{\glspagelistwidth}|}%
7824 {\end{longtable}}%

```


Place horizontal lines at the head and foot of the table:

```
7825 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
7826 }
```

`longragged3colheader` The `longragged3colheader` style is like `longragged3col` but with a header row:

```
7827 \newglossarystyle{longragged3colheader}{%
```

Base it on the `glostylelongragged3col` style:

```
7828 \setglossarystyle{longragged3col}%
```

Set the table's header:

```
7829 \renewcommand*{\glossaryheader}{%
7830 \bfseries\entryname&\bfseries\descriptionname&
7831 \bfseries\pagelistname\tabularnewline\endhead}%
7832 }
```

`longragged3colheaderborder` The `longragged3colheaderborder` style is like the above but with a border

```
7833 \newglossarystyle{longragged3colheaderborder}{%
```

Base it on the `glostylelongragged3colborder` style:

```
7834 \setglossarystyle{longragged3colborder}%
```

Set the table's header and add horizontal line at table's foot:

```
7835 \renewcommand*{\glossaryheader}{%
7836 \hline
7837 \bfseries\entryname&\bfseries\descriptionname&
7838 \bfseries\pagelistname\tabularnewline\hline\endhead
7839 \hline\endfoot}%
7840 }
```

`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.

```
7841 \newglossarystyle{altlongragged4col}{%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
7842 \renewenvironment{theglossary}%
7843 {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}l%
7844 >{\raggedright}p{\glspagelistwidth}}}%
7845 {\end{longtable}}%
```

No table header:

```
7846 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
7847 \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):

```
7848 \renewcommand{\glossentry}[2]{}%
```

```

7849 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
7850 \glossentrydesc{##1} & \glossentrysymbol{##1} &
7851 ##2\tabularnewline
7852 }%

```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```

7853 \renewcommand{\subglossentry}[3]{%
7854 &
7855 \glssubentryitem{##2}%
7856 \glstarget{##2}{\strut}\glossentrydesc{##2} &
7857 \glossentrysymbol{##2} & ##3\tabularnewline
7858 }%

```

Blank row between groups:

```

7859 \renewcommand*\glsgroupskip{%
7860 \ifglsgroupskip\else & & \tabularnewline\fi}%
7861 }

```

`longragged4colheader` The `altlongragged4colheader` style is like `altlongragged4col` but with a header row.

```

7862 \newglossarystyle{altlongragged4colheader}{%

```

Base it on the `glostylealtlongragged4col` style:

```

7863 \setglossarystyle{altlongragged4col}%

```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```

7864 \renewenvironment{theglossary}%
7865 {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}l%
7866 >{\raggedright}p{\glspagelistwidth}}}%
7867 {\end{longtable}}%

```

Table has a header:

```

7868 \renewcommand*\glossaryheader{%
7869 \bfseries\entryname&\bfseries\descriptionname&
7870 \bfseries \symbolname&
7871 \bfseries\pagelistname\tabularnewline\endhead}%
7872 }

```

`longragged4colborder` The `altlongragged4colborder` style is like `altlongragged4col` but with a border.

```

7873 \newglossarystyle{altlongragged4colborder}{%

```

Base it on the `glostylealtlongragged4col` style:

```

7874 \setglossarystyle{altlongragged4col}%

```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```

7875 \renewenvironment{theglossary}%
7876 {\begin{longtable}{|l|>{\raggedright}p{\glsdescwidth}|l|%
7877 >{\raggedright}p{\glspagelistwidth}|}%
7878 {\end{longtable}}%

```

Add horizontal lines to the head and foot of the table:

```
7879 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
7880 }
```

`ged4colheaderborder` The `altlongragged4colheaderborder` style is like the above but with a header as well as a border.

```
7881 \newglossarystyle{altlongragged4colheaderborder}{%
```

Base it on the `glostylealtlongragged4col` style:

```
7882 \setglossarystyle{altlongragged4col}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
7883 \renewenvironment{theglossary}%
7884 {\begin{longtable}{|l|>{\raggedright}p{\glsdescwidth}|l|}%
7885 >{\raggedright}p{\glspagelistwidth}|}%
7886 {\end{longtable}}%
```

Add table header and horizontal line at the table's foot:

```
7887 \renewcommand*{\glossaryheader}{%
7888 \hline\bfseries\entryname&\bfseries\descriptionname&
7889 \bfseries \symbolname&
7890 \bfseries\pagelistname\tabularnewline\hline\endhead
7891 \hline\endfoot}%
7892 }
```

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.

```
7893 \ProvidesPackage{glossary-mcols}[2013/11/14 v4.0 (NLCT)]
```

Required packages:

```
7894 \RequirePackage{multicol}
7895 \RequirePackage{glossary-tree}
```

`\glscols` Define macro in which to store the number of columns. (Defaults to 2.)

```
7896 \newcommand*{\glscols}{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.)

```
7897 \newglossarystyle{mcolindex}{%
7898 \setglossarystyle{index}%
7899 \renewenvironment{theglossary}%
7900 {%
```

```

7901 \begin{multicols}{\glsmcols}
7902 \setlength{\parindent}{0pt}%
7903 \setlength{\parskip}{0pt plus 0.3pt}%
7904 \let\item\@idxitem}%
7905 {\end{multicols}}%
7906 }

```

mcolindexgroup As mcolindex but has headings:

```

7907 \newglossarystyle{mcolindexgroup}{%
7908 \setglossarystyle{mcolindex}%
7909 \renewcommand*{\glsgroupheading}[1]{%
7910 \item\textbf{\glsgroupheading{##1}}\indexspace}%
7911 }

```

mcolindexhypergroup The mcolindexhypergroup style is like the mcolindexgroup style but has hyper navigation.

```

7912 \newglossarystyle{mcolindexhypergroup}{%
    Base it on the glostylemcolindex style:
7913 \setglossarystyle{mcolindex}%
    Put navigation links to the groups at the start of the glossary:
7914 \renewcommand*{\glossaryheader}{%
7915 \item\textbf{\glshypernavigation}\indexspace}%
    Add a heading for each group (with a target). The group's title is in bold followed
    by a vertical gap.
7916 \renewcommand*{\glsgroupheading}[1]{%
7917 \item\textbf{\glshypernavhypertarget{##1}}{\glsgroupheading{##1}}}%
7918 \indexspace}%
7919 }

```

mcoltree Multi-column index style. Same as the tree, but puts the glossary in multiple columns.

```

7920 \newglossarystyle{mcoltree}{%
7921 \setglossarystyle{tree}%
7922 \renewenvironment{theglossary}%
7923 {%
7924 \begin{multicols}{\glsmcols}
7925 \setlength{\parindent}{0pt}%
7926 \setlength{\parskip}{0pt plus 0.3pt}%
7927 }%
7928 {\end{multicols}}%
7929 }

```

mcoltreegroup Like the mcoltree style but the glossary groups have headings.

```

7930 \newglossarystyle{mcoltreegroup}{%
    Base it on the glostylemcoltree style:
7931 \setglossarystyle{mcoltree}%

```

Each group has a heading (in bold) followed by a vertical gap):

```
7932 \renewcommand{\glsgroupheading}[1]{\par
7933 \noindent\textbf{\glsgrouptitle{##1}}\par\indexspace}%
7934 }
```

mcoltreehypergroup The mcoltreehypergroup style is like the treegroup style, but has a set of links to the groups at the start of the glossary.

```
7935 \newglossarystyle{mcoltreehypergroup}{%
```

Base it on the glostylemcoltree style:

```
7936 \setglossarystyle{mcoltree}%
```

Put navigation links to the groups at the start of the theglossary environment:

```
7937 \renewcommand*\glossaryheader{%
7938 \par\noindent\textbf{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
7939 \renewcommand*\glsgroupheading[1]{%
7940 \par\noindent
7941 \textbf{\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}\par
7942 \indexspace}%
7943 }
```

mcoltreenoname Multi-column index style. Same as the treenoname, but puts the glossary in multiple columns.

```
7944 \newglossarystyle{mcoltreenoname}{%
7945 \setglossarystyle{treenoname}%
7946 \renewenvironment{theglossary}%
7947 {%
7948 \begin{multicols}{\glsmcols}
7949 \setlength{\parindent}{0pt}%
7950 \setlength{\parskip}{0pt plus 0.3pt}%
7951 }%
7952 {\end{multicols}}}%
7953 }
```

mcoltreenonamegroup Like the mcoltreenoname style but the glossary groups have headings.

```
7954 \newglossarystyle{mcoltreenonamegroup}{%
```

Base it on the glostylemcoltreenoname style:

```
7955 \setglossarystyle{mcoltreenoname}%
```

Give each group a heading:

```
7956 \renewcommand{\glsgroupheading}[1]{\par
7957 \noindent\textbf{\glsgrouptitle{##1}}\par\indexspace}%
7958 }
```

treenonamehypergroup The mcoltreenonamehypergroup style is like the mcoltreenonamegroup style, but has a set of links to the groups at the start of the glossary.

```
7959 \newglossarystyle{mcoltreenonamehypergroup}{%
```

Base it on the `glostylemcoltreename` style:

```
7960 \setglossarystyle{mcoltreename}%
```

Put navigation links to the groups at the start of the `theglossary` environment:

```
7961 \renewcommand*{\glossaryheader}{%
```

```
7962 \par\noindent\textbf{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
7963 \renewcommand*{\glsgroupheading}[1]{%
```

```
7964 \par\noindent
```

```
7965 \textbf{\glsnavigationhypertarget{##1}{\glsgroupheading{##1}}}\par
```

```
7966 \indexspace}%
```

```
7967 }
```

`mcolalttree` Multi-column index style. Same as the `alttree`, but puts the glossary in multiple columns.

```
7968 \newglossarystyle{mcolalttree}{%
```

```
7969 \setglossarystyle{alttree}%
```

```
7970 \renewenvironment{theglossary}%
```

```
7971 {%
```

```
7972 \begin{multicols}{\glsmcols}
```

```
7973 \def\@gls@prevlevel{-1}%
```

```
7974 \mbox{}\par
```

```
7975 }%
```

```
7976 {\par\end{multicols}}%
```

```
7977 }
```

`mcolalttreegroup` Like the `mcolalttree` style but the glossary groups have headings.

```
7978 \newglossarystyle{mcolalttreegroup}{%
```

Base it on the `glostylemcolalttree` style:

```
7979 \setglossarystyle{mcolalttree}%
```

Give each group a heading.

```
7980 \renewcommand{\glsgroupheading}[1]{\par
```

```
7981 \def\@gls@prevlevel{-1}%
```

```
7982 \hangindent0pt\relax
```

```
7983 \parindent0pt\relax
```

```
7984 \textbf{\glsgroupheading{##1}}\par\indexspace}%
```

```
7985 }
```

`olalttreehypergroup` The `mcolalttreehypergroup` style is like the `mcolalttreegroup` style, but has a set of links to the groups at the start of the glossary.

```
7986 \newglossarystyle{mcolalttreehypergroup}{%
```

Base it on the `glostylemcolalttree` style:

```
7987 \setglossarystyle{mcolalttree}%
```

Put the navigation links in the header

```
7988 \renewcommand*{\glossaryheader}{%
7989 \par
7990 \def\@gls@prevlevel{-1}%
7991 \hangindent0pt\relax
7992 \parindent0pt\relax
7993 \textbf{\glsnavigation}\par\indexspace}%
```

Put a hypertarget at the start of each group

```
7994 \renewcommand*{\glsgroupheading}[1]{%
7995 \par
7996 \def\@gls@prevlevel{-1}%
7997 \hangindent0pt\relax
7998 \parindent0pt\relax
7999 \textbf{\glsnavhypertarget{##1}\glsgetgrouptitle{##1}}\par
8000 \indexspace}}
```

5.7 Glossary Styles using supertabular environment (glossary-super package)

The glossary styles defined in the package use the supertabular environment.

```
8001 \ProvidesPackage{glossary-super}[2013/11/14 v4.0 (NLCT)]
```

Requires the package:

```
8002 \RequirePackage{supertabular}
```

`\glsdescwidth` This is a length that governs the width of the description column. This may already have been defined if has been loaded.

```
8003 \@ifundefined{glsdescwidth}{%
8004 \newlength{glsdescwidth
8005 \setlength{glsdescwidth}{0.6\hsize}
8006 }{}}
```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined if has been loaded.

```
8007 \@ifundefined{glspagelistwidth}{%
8008 \newlength{glspagelistwidth
8009 \setlength{glspagelistwidth}{0.1\hsize}
8010 }{}}
```

`super` The super glossary style uses the supertabular environment (it uses lengths defined in the package.)

```
8011 \newglossarystyle{super}{%
```

Put the glossary in a supertabular environment with two columns and no head or tail:

```
8012 \renewenvironment{theglossary}{%
8013 {\tablehead{}\tabletail}%
8014 \begin{supertabular}{lp{glsdescwidth}}%
8015 {\end{supertabular}}%
```

Do nothing at the start of the table:

```
8016 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8017 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```
8018 \renewcommand{\glossentry}[2]{%
8019   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8020   \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline
8021 }%
```

Sub entries put in a row (no name, description and page list in second column):

```
8022 \renewcommand{\subglossentry}[3]{%
8023   &
8024   \glssubentryitem{##2}%
8025   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
8026   ##3\tabularnewline
8027 }%
```

Blank row between groups:

```
8028 \renewcommand*{\glsgroupskip}{}%
8029 \ifglsgnoglobskip\else & \tabularnewline\fi}%
8030 }
```

superborder The superborder style is like the above, but with horizontal and vertical lines:

```
8031 \newglossarystyle{superborder}{}%
```

Base it on the `glostylesuper` style:

```
8032 \setglossarystyle{super}{}%
```

Put the glossary in a `supertabular` environment with two columns and a horizontal line in the head and tail:

```
8033 \renewenvironment{theglossary}%
8034   {\tablehead{\hline}\tabletail{\hline}}%
8035   \begin{supertabular}{|l|p{\glsdescwidth}|}%
8036   {\end{supertabular}}%
8037 }
```

superheader The superheader style is like the super style, but with a header:

```
8038 \newglossarystyle{superheader}{}%
```

Base it on the `glostylesuper` style:

```
8039 \setglossarystyle{super}{}%
```

Put the glossary in a `supertabular` environment with two columns, a header and no tail:

```
8040 \renewenvironment{theglossary}%
8041   {\tablehead{\bfseries \entryname &
```



```

8042 \bfseries\descriptionname\tabularnewline}%
8043 \tabletail{}%
8044 \begin{supertabular}{lp{\glsgdescwidth}}}%
8045 {\end{supertabular}}%
8046 }

```

superheaderborder The superheaderborder style is like the super style but with a header and border:

```
8047 \newglossarystyle{superheaderborder}{%
```

Base it on the glostylesuper style:

```
8048 \setglossarystyle{super}%
```

Put the glossary in a supertabular environment with two columns, a header and horizontal lines above and below the table:

```

8049 \renewenvironment{theglossary}%
8050 {\tablehead{\hline\bfseries \entryname &
8051 \bfseries \descriptionname\tabularnewline\hline}%
8052 \tabletail{\hline}
8053 \begin{supertabular}{|lp{\glsgdescwidth}|}%
8054 {\end{supertabular}}%
8055 }

```

super3col The super3col style is like the super style, but with 3 columns:

```
8056 \newglossarystyle{super3col}{%
```

Put the glossary in a supertabular environment with three columns and no head or tail:

```

8057 \renewenvironment{theglossary}%
8058 {\tablehead{}\tabletail{}%
8059 \begin{supertabular}{lp{\glsgdescwidth}p{\glspagelistwidth}}}%
8060 {\end{supertabular}}%

```

Do nothing at the start of the table:

```
8061 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8062 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```

8063 \renewcommand{\glossentry}[2]{%
8064 \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8065 \glossentrydesc{##1} & ##2\tabularnewline
8066 }%

```

Sub entries on a row (no name, description in second column, page list in last column):

```

8067 \renewcommand{\subglossentry}[3]{%
8068 &
8069 \glssubentryitem{##2}%
8070 \glstarget{##2}{\strut}\glossentrydesc{##2} &

```

```
8071      ##3\tabularnewline
```

```
8072  }%
```

Blank row between groups:

```
8073  \renewcommand*{\glsgroupskip}{%
```

```
8074    \ifglsgnigroupskip\else & &\tabularnewline\fi}%
```

```
8075 }
```

super3colborder The **super3colborder** style is like the **super3col** style, but with a border:

```
8076 \newglossarystyle{super3colborder}{%
```

Base it on the **glostylesuper3col** style:

```
8077  \setglossarystyle{super3col}%
```

Put the glossary in a supertabular environment with three columns and a horizontal line in the head and tail:

```
8078  \renewenvironment{theglossary}%
```

```
8079    {\tablehead{\hline}\tabletail{\hline}%
```

```
8080     \begin{supertabular}{|l|p{\glsgdescwidth}|p{\glspagelistwidth}|}%
```

```
8081     {\end{supertabular}}%
```

```
8082 }
```

super3colheader The **super3colheader** style is like the **super3col** style but with a header row:

```
8083 \newglossarystyle{super3colheader}{%
```

Base it on the **glostylesuper3col** style:

```
8084  \setglossarystyle{super3col}%
```

Put the glossary in a supertabular environment with three columns, a header and no tail:

```
8085  \renewenvironment{theglossary}%
```

```
8086    {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
```

```
8087     \bfseries\pagelistname\tabularnewline}\tabletail{}}%
```

```
8088     \begin{supertabular}{lp{\glsgdescwidth}p{\glspagelistwidth}}%
```

```
8089     {\end{supertabular}}%
```

```
8090 }
```

super3colheaderborder The **super3colheaderborder** style is like the **super3col** style but with a header and border:

```
8091 \newglossarystyle{super3colheaderborder}{%
```

Base it on the **glostylesuper3colborder** style:

```
8092  \setglossarystyle{super3colborder}%
```

Put the glossary in a supertabular environment with three columns, a header with horizontal lines and a horizontal line in the tail:

```
8093  \renewenvironment{theglossary}%
```

```
8094    {\tablehead{\hline
```

```
8095     \bfseries\entryname&\bfseries\descriptionname&
```

```
8096     \bfseries\pagelistname\tabularnewline\hline}%
```

```
8097     \tabletail{\hline}%
```

```

8098     \begin{supertabular}{|l|p{\glstdescwidth}|p{\glspagelistwidth}|}%
8099     {\end{supertabular}}}%
8100 }

```

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.

```
8101 \newglossarystyle{super4col}{%
```

Put the glossary in a `supertabular` environment with four columns and no head or tail:

```

8102   \renewenvironment{theglossary}%
8103   {\tablehead{}\tabletail}%
8104   \begin{supertabular}{|l|l|l|l|}%
8105   \end{supertabular}}%

```

Do nothing at the start of the table:

```
8106   \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8107   \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:

```

8108   \renewcommand{\glossentry}[2]{%
8109     \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8110     \glossentrydesc{##1} &
8111     \glossentrysymbol{##1} & ##3\tabularnewline
8112   }%

```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```

8113   \renewcommand{\subglossentry}[3]{%
8114     &
8115     \glssubentryitem{##2}%
8116     \glstarget{##2}{\strut}\glossentrydesc{##2} &
8117     \glossentrysymbol{##2} & ##3\tabularnewline
8118   }%

```

Blank row between groups:

```

8119   \renewcommand*{\glsgroupskip}{%
8120     \ifglsnogroupskip\else & & \tabularnewline\fi}%
8121 }

```

super4colheader The `super4colheader` style is like the `super4col` but with a header row.

```
8122 \newglossarystyle{super4colheader}{%
```

Base it on the `glostylesuper4col` style:

```
8123   \setglossarystyle{super4col}%
```

Put the glossary in a `supertabular` environment with four columns, a header and no tail:

```

8124 \renewenvironment{theglossary}%
8125   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8126     \bfseries\symbolname &
8127     \bfseries\pagelistname\tabularnewline}%
8128   \tabletail{}}%
8129   \begin{supertabular}{|l|l|l|l|}%
8130   {\end{supertabular}}}%
8131 }

```

super4colborder The super4colborder style is like the super4col but with a border.

```

8132 \newglossarystyle{super4colborder}{%
      Base it on the glostylesuper4col style:
8133   \setglossarystyle{super4col}%
      Put the glossary in a supertabular environment with four columns and a horizontal line in the head and tail:
8134   \renewenvironment{theglossary}%
8135     {\tablehead{\hline}\tabletail{\hline}%
8136     \begin{supertabular}{|l|l|l|l|}%
8137     {\end{supertabular}}}%
8138 }

```

super4colheaderborder The super4colheaderborder style is like the super4col but with a header and border.

```

8139 \newglossarystyle{super4colheaderborder}{%
      Base it on the glostylesuper4col style:
8140   \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:
8141   \renewenvironment{theglossary}%
8142     {\tablehead{\hline\bfseries\entryname&\bfseries\descriptionname&
8143       \bfseries\symbolname &
8144       \bfseries\pagelistname\tabularnewline\hline}%
8145     \tabletail{\hline}%
8146     \begin{supertabular}{|l|l|l|l|}%
8147     {\end{supertabular}}}%
8148 }

```

altsuper4col The altsuper4col glossary style is like super4col but has provision for multiline descriptions.

```

8149 \newglossarystyle{altsuper4col}{%
      Base it on the glostylesuper4col style:
8150   \setglossarystyle{super4col}%
      Put the glossary in a supertabular environment with four columns and no head or tail:

```

```

8151 \renewenvironment{theglossary}%
8152   {\tablehead{}\tabletail{}}%
8153   \begin{supertabular}{lp{\glstdescwidth}lp{\glspagelistwidth}}}%
8154   {\end{supertabular}}}%
8155 }

```

altsuper4colheader The altsuper4colheader style is like the altsuper4col but with a header row.

```

8156 \newglossarystyle{altsuper4colheader}{%
      Base it on the glostylesuper4colheader style:
8157   \setglossarystyle{super4colheader}%
      Put the glossary in a supertabular environment with four columns, a header and
      no tail:
8158   \renewenvironment{theglossary}%
8159   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8160     \bfseries\symbolname &
8161     \bfseries\pagelistname\tabularnewline}\tabletail{}}%
8162   \begin{supertabular}{lp{\glstdescwidth}lp{\glspagelistwidth}}}%
8163   {\end{supertabular}}}%
8164 }

```

altsuper4colborder The altsuper4colborder style is like the altsuper4col but with a border.

```

8165 \newglossarystyle{altsuper4colborder}{%
      Base it on the glostylesuper4colborder style:
8166   \setglossarystyle{super4colborder}%
      Put the glossary in a supertabular environment with four columns and a hori-
      zontal line in the head and tail:
8167   \renewenvironment{theglossary}%
8168   {\tablehead{\hline}\tabletail{\hline}%
8169   \begin{supertabular}%
8170     {\lllp{\glstdescwidth}lllp{\glspagelistwidth}ll}%
8171   {\end{supertabular}}}%
8172 }

```

per4colheaderborder The altsuper4colheaderborder style is like the altsuper4col but with a header and border.

```

8173 \newglossarystyle{altsuper4colheaderborder}{%
      Base it on the glostylesuper4colheaderborder style:
8174   \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:
8175   \renewenvironment{theglossary}%
8176   {\tablehead{\hline
8177     \bfseries\entryname &
8178     \bfseries\descriptionname &
8179     \bfseries\symbolname &

```

```

8180      \bfseries\pagelistname\tabularnewline\hline}%
8181      \tabletail{\hline}%
8182      \begin{supertabular}%
8183        {\lllp{\glsdescwidth}\lllp{\glspagelistwidth}}}%
8184      {\end{supertabular}}%
8185 }

```

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.

```
8186 \ProvidesPackage{glossary-superragged}[2013/11/14 v4.0 (NLCT)]
```

Requires the package:

```
8187 \RequirePackage{array}
```

Requires the package:

```
8188 \RequirePackage{supertabular}
```

`\glsdescwidth` This is a length that governs the width of the description column. This may already have been defined.

```

8189 \@ifundefined{glsdescwidth}{%
8190   \newlength\glsdescwidth
8191   \setlength{\glsdescwidth}{0.6\hsize}
8192 }{}

```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```

8193 \@ifundefined{glspagelistwidth}{%
8194   \newlength\glspagelistwidth
8195   \setlength{\glspagelistwidth}{0.1\hsize}
8196 }{}

```

`superragged` The superragged glossary style uses the supertabular environment.

```
8197 \newglossarystyle{superragged}{%
```

Put the glossary in a supertabular environment with two columns and no head or tail:

```

8198   \renewenvironment{theglossary}%
8199     {\tablehead{}\tabletail{}}%
8200     \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}}}%
8201     {\end{supertabular}}%

```

Do nothing at the start of the table:

```
8202   \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8203   \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```
8204 \renewcommand{\glossentry}[2]{%
8205   \glentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8206   \glossentrydesc{##1}\glspostdescription\space ##2%
8207   \tabularnewline
8208 }
```

Sub entries put in a row (no name, description and page list in second column):

```
8209 \renewcommand{\subglossentry}[3]{%
8210   &
8211   \glssubentryitem{##2}%
8212   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
8213   ##3%
8214   \tabularnewline
8215 }
```

Blank row between groups:

```
8216 \renewcommand*{\glsgroupskip}{\ifglsgnোগroupskip\else & \tabularnewline\fi}%
8217 }
```

superraggedborder The superraggedborder style is like the above, but with horizontal and vertical lines:

```
8218 \newglossarystyle{superraggedborder}{%
```

Base it on the glostylessuperragged style:

```
8219 \setglossarystyle{superragged}%
```

Put the glossary in a supertabular environment with two columns and a horizontal line in the head and tail:

```
8220 \renewenvironment{theglossary}%
8221   {\tablehead{\hline}\tabletail{\hline}%
8222   \begin{supertabular}{|l|>\raggedright}p{\glsgdescwidth}}}%
8223   {\end{supertabular}}%
8224 }
```

superraggedheader The superraggedheader style is like the super style, but with a header:

```
8225 \newglossarystyle{superraggedheader}{%
```

Base it on the glostylessuperragged style:

```
8226 \setglossarystyle{superragged}%
```

Put the glossary in a supertabular environment with two columns, a header and no tail:

```
8227 \renewenvironment{theglossary}%
8228   {\tablehead{\bfseries \entryname & \bfseries \descriptionname
8229   \tabularnewline}%
8230   \tabletail{}}%
8231   \begin{supertabular}{|l|>\raggedright}p{\glsgdescwidth}}}%
8232   {\end{supertabular}}%
8233 }
```

`rraggedheaderborder` The `superraggedheaderborder` style is like the `superragged` style but with a header and border:

```
8234 \newglossarystyle{superraggedheaderborder}{%
```

Base it on the `glostylesuper` style:

```
8235 \setglossarystyle{superragged}%
```

Put the glossary in a `supertabular` environment with two columns, a header and horizontal lines above and below the table:

```
8236 \renewenvironment{theglossary}%
8237 {\tablehead{\hline\bfseries \entryname &
8238 \bfseries \descriptionname\tabularnewline\hline}%
8239 \tabletail{\hline}
8240 \begin{supertabular}{|l|>\raggedright}p{\glsgdescwidth}|}%
8241 {\end{supertabular}}%
8242 }
```

`superragged3col` The `superragged3col` style is like the `superragged` style, but with 3 columns:

```
8243 \newglossarystyle{superragged3col}{%
```

Put the glossary in a `supertabular` environment with three columns and no head or tail:

```
8244 \renewenvironment{theglossary}%
8245 {\tablehead{}\tabletail{}}%
8246 \begin{supertabular}{|l>\raggedright}p{\glsgdescwidth}%
8247 >\raggedright}p{\glspagelistwidth}}}%
8248 {\end{supertabular}}%
```

Do nothing at the start of the table:

```
8249 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8250 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
8251 \renewcommand{\glossentry}[2]{%
8252 \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8253 \glossentrydesc{##1} &
8254 ##2\tabularnewline
8255 }%
```

Sub entries on a row (no name, description in second column, page list in last column):

```
8256 \renewcommand{\subglossentry}[3]{%
8257 &
8258 \glssubentryitem{##2}%
8259 \glstarget{##2}{\strut}\glossentrydesc{##2} &
8260 ##3\tabularnewline
8261 }%
```


Blank row between groups:

```
8262 \renewcommand*{\glsgroupskip}{\ifglsgroupskip\else & &\tabularnewline\fi}%  
8263 }
```

superragged3colborder The `superragged3colborder` style is like the `superragged3col` style, but with a border:

```
8264 \newglossarystyle{superragged3colborder}{%
```

Base it on the `glostylesuperragged3col` style:

```
8265 \setglossarystyle{superragged3col}%
```

Put the glossary in a `supertabular` environment with three columns and a horizontal line in the head and tail:

```
8266 \renewenvironment{theglossary}%  
8267 {\tablehead{\hline}\tabletail{\hline}%  
8268 \begin{supertabular}{|l|>{\raggedright}p{\glsgdescwidth}|%  
8269 >{\raggedright}p{\glspagelistwidth}|}%  
8270 {\end{supertabular}}%  
8271 }
```

superragged3colheader The `superragged3colheader` style is like the `superragged3col` style but with a header row:

```
8272 \newglossarystyle{superragged3colheader}{%
```

Base it on the `glostylesuperragged3col` style:

```
8273 \setglossarystyle{superragged3col}%
```

Put the glossary in a `supertabular` environment with three columns, a header and no tail:

```
8274 \renewenvironment{theglossary}%  
8275 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&  
8276 \bfseries\pagelistname\tabularnewline}\tabletail{}}%  
8277 \begin{supertabular}{|l>{\raggedright}p{\glsgdescwidth}%  
8278 >{\raggedright}p{\glspagelistwidth}|}%  
8279 {\end{supertabular}}%  
8280 }
```

superragged3colheaderborder The `superragged3colheaderborder` style is like the `superragged3col` style but with a header and border:

```
8281 \newglossarystyle{superragged3colheaderborder}{%
```

Base it on the `glostylesuperragged3colborder` style:

```
8282 \setglossarystyle{superragged3colborder}%
```

Put the glossary in a `supertabular` environment with three columns, a header with horizontal lines and a horizontal line in the tail:

```
8283 \renewenvironment{theglossary}%  
8284 {\tablehead{\hline  
8285 \bfseries\entryname&\bfseries\descriptionname&  
8286 \bfseries\pagelistname\tabularnewline\hline}%
```

```

8287     \tabletail{\hline}%
8288     \begin{supertabular}{|l|>{\raggedright}p{\glsgdescwidth}|%
8289         >{\raggedright}p{\glspagelistwidth}|}%
8290     {\end{supertabular}}%
8291 }

```

`altsuperragged4col` The `altsuperragged4col` glossary style is like `altsuper4col` style in the package but uses ragged right formatting in the description and page list columns.

```

8292 \newglossarystyle{altsuperragged4col}{%

```

Put the glossary in a `supertabular` environment with four columns and no head or tail:

```

8293     \renewenvironment{theglossary}%
8294     {\tablehead{}\tabletail{}}%
8295     \begin{supertabular}{|l>{\raggedright}p{\glsgdescwidth}l%
8296         >{\raggedright}p{\glspagelistwidth}|}%
8297     {\end{supertabular}}%

```

Do nothing at the start of the table:

```

8298     \renewcommand*{\glossaryheader}{}%

```

No group headings:

```

8299     \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:

```

8300     \renewcommand{\glossentry}[2]{%
8301         \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8302         \glossentrydesc{##1} &
8303         \glossentrysymbol{##1} & ##2\tabularnewline
8304     }%

```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```

8305     \renewcommand{\subglossentry}[3]{%
8306         &
8307         \glssubentryitem{##2}%
8308         \glstarget{##2}{\strut}\glossentrydesc{##2} &
8309         \glossentrysymbol{##2} & ##3\tabularnewline
8310     }%

```

Blank row between groups:

```

8311     \renewcommand*{\glsgroupskip}{\ifglsgnোগroupskip\else & & \tabularnewline\fi}%
8312 }

```

`perragged4colheader` The `altsuperragged4colheader` style is like the `altsuperragged4col` style but with a header row.

```

8313 \newglossarystyle{altsuperragged4colheader}{%

```

Base it on the `glostylealtsuperragged4col` style:

```

8314     \setglossarystyle{altsuperragged4col}%

```

Put the glossary in a supertabular environment with four columns, a header and no tail:

```

8315 \renewenvironment{theglossary}%
8316   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8317     \bfseries\symbolname &
8318     \bfseries\pagelistname\tabularnewline}\tabletail{}}%
8319   \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}l%
8320     >{\raggedright}p{\glspagelistwidth}}}%
8321   {\end{supertabular}}%
8322 }

```

`altperragged4colborder` The `altperragged4colborder` style is like the `altperragged4col` style but with a border.

```

8323 \newglossarystyle{altperragged4colborder}{%

```

Base it on the `glostylealtperragged4col` style:

```

8324 \setglossarystyle{altperragged4col}%

```

Put the glossary in a supertabular environment with four columns and a horizontal line in the head and tail:

```

8325 \renewenvironment{theglossary}%
8326   {\tablehead{\hline}\tabletail{\hline}%
8327   \begin{supertabular}%
8328     {\l|>{\raggedright}p{\glsgdescwidth}l|}%
8329     >{\raggedright}p{\glspagelistwidth}l}%
8330   {\end{supertabular}}%
8331 }

```

`altperragged4colheaderborder` The `altperragged4colheaderborder` style is like the `altperragged4col` style but with a header and border.

```

8332 \newglossarystyle{altperragged4colheaderborder}{%

```

Base it on the `glostylealtperragged4col` style:

```

8333 \setglossarystyle{altperragged4col}%

```

Put the glossary in a supertabular environment with four columns and a header bordered by horizontal lines and a horizontal line in the tail:

```

8334 \renewenvironment{theglossary}%
8335   {\tablehead{\hline
8336     \bfseries\entryname &
8337     \bfseries\descriptionname &
8338     \bfseries\symbolname &
8339     \bfseries\pagelistname\tabularnewline\hline}%
8340   \tabletail{\hline}%
8341   \begin{supertabular}%
8342     {\l|>{\raggedright}p{\glsgdescwidth}l|}%
8343     >{\raggedright}p{\glspagelistwidth}l}%
8344   {\end{supertabular}}%
8345 }

```

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.

8346 \ProvidesPackage{glossary-tree}[2014/08/27 v4.10 (NLCT)]

\glstreenamefmt Format used to display the name in the tree styles. (This may be counteracted by \glsnamefont.) This command is also used to format the group headings.

8347 \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.

8348 \newglossarystyle{index}{%

Set the paragraph indentation and skip and define \item to be the same as that used by theindex:

8349 \renewenvironment{theglossary}{%
8350 {\setlength{\parindent}{0pt}}%
8351 \setlength{\parskip}{0pt plus 0.3pt}}%
8352 \let\item\@idxitem}%

8353 {\par}%

Do nothing at the start of the environment:

8354 \renewcommand*{\glossaryheader}{}%

No group headers:

8355 \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.

8356 \renewcommand*{\glossentry}[2]{%
8357 \item\glstentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
8358 \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
8359 \space \glossentrydesc{##1}\glspostdescription\space ##2%
8360 }%

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.

8361 \renewcommand{\subglossentry}[3]{%
8362 \ifcase##1\relax
8363 % level 0
8364 \item
8365 \or
8366 % level 1
8367 \subitem

```

8368     \glssubentryitem{##2}%
8369     \else
8370     % all other levels
8371     \subsubitem
8372     \fi
8373     \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
8374     \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
8375     \space\glossentrydesc{##2}\glspostdescription\space ##3%
8376 }%

```

Vertical gap between groups is the same as that used by indices:

```

8377 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}

```

indexgroup The indexgroup style is like the index style but has headings.

```

8378 \newglossarystyle{indexgroup}{%

```

Base it on the glostyleindex style:

```

8379 \setglossarystyle{index}%

```

Add a heading for each group. This puts the group's title in bold followed by a vertical gap.

```

8380 \renewcommand*{\glsgroupheading}[1]{%
8381     \item\glstreenamefmt{\glsgrouptitle{##1}}\indexspace}%
8382 }

```

indexhypergroup The indexhypergroup style is like the indexgroup style but has hyper navigation.

```

8383 \newglossarystyle{indexhypergroup}{%

```

Base it on the glostyleindex style:

```

8384 \setglossarystyle{index}%

```

Put navigation links to the groups at the start of the glossary:

```

8385 \renewcommand*{\glossaryheader}{%
8386     \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.

```

8387 \renewcommand*{\glsgroupheading}[1]{%
8388     \item\glstreenamefmt{\glsnavigation\hypertarget{##1}{\glsgrouptitle{##1}}}%
8389     \indexspace}%
8390 }

```

tree The tree glossary style is similar in style to the index style, but can have arbitrary levels.

```

8391 \newglossarystyle{tree}{%

```

Set the paragraph indentation and skip:

```

8392 \renewenvironment{theglossary}%
8393     {\setlength{\parindent}{0pt}%
8394     \setlength{\parskip}{0pt plus 0.3pt}}%
8395 {}%

```

Do nothing at the start of the theglossary environment:

```
8396 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8397 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries: name in bold, followed by symbol in brackets (if it exists), the description and the page list:

```
8398 \renewcommand{\glossentry}[2]{%
8399 \hangindent0pt\relax
8400 \parindent0pt\relax
8401 \glstryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
8402 \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
8403 \space\glossentrydesc{##1}\glspostdescription\space##2\par
8404 }%
```

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.

```
8405 \renewcommand{\subglossentry}[3]{%
8406 \hangindent##1\glstreeindent\relax
8407 \parindent##1\glstreeindent\relax
8408 \ifnum##1=1\relax
8409 \glssubentryitem{##2}%
8410 \fi
8411 \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
8412 \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
8413 \space\glossentrydesc{##2}\glspostdescription\space ##3\par
8414 }%
```

Vertical gap between groups is the same as that used by indices:

```
8415 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}}
```

treegroup Like the tree style but the glossary groups have headings.

```
8416 \newglossarystyle{treegroup}{%
```

Base it on the glostyletree style:

```
8417 \setglossarystyle{tree}%
```

Each group has a heading (in bold) followed by a vertical gap):

```
8418 \renewcommand{\glsgroupheading}[1]{\par
8419 \noindent\glstreenamefmt{\glsgrouptitle{##1}}\par\indexspace}%
8420 }
```

treehypergroup The treehypergroup style is like the treegroup style, but has a set of links to the groups at the start of the glossary.

```
8421 \newglossarystyle{treehypergroup}{%
```

Base it on the glostyletree style:

```
8422 \setglossarystyle{tree}%
```

Put navigation links to the groups at the start of the theglossary environment:

```
8423 \renewcommand*{\glossaryheader}{%
8424   \par\noindent\glstreenamefmt{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
8425 \renewcommand*{\glsgroupheading}[1]{%
8426   \par\noindent
8427   \glstreenamefmt{\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}\par
8428   \indexspace}%
8429 }
```

`\glstreeindent` Length governing left indent for each level of the tree style.

```
8430 \newlength\glstreeindent
8431 \setlength{\glstreeindent}{10pt}
```

`treenoname` The `treenoname` glossary style is like the tree style, but doesn't print the name or symbol for sub-levels.

```
8432 \newglossarystyle{treenoname}{%
```

Set the paragraph indentation and skip:

```
8433 \renewenvironment{theglossary}%
8434   {\setlength{\parindent}{0pt}}%
8435   \setlength{\parskip}{0pt plus 0.3pt}}%
8436   {}%
```

No header:

```
8437 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8438 \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.

```
8439 \renewcommand{\glossentry}[2]{%
8440   \hangindent0pt\relax
8441   \parindent0pt\relax
8442   \glstentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
8443   \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
8444   \space\glossentrydesc{##1}\glspostdescription\space##2\par
8445   }%
```

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.

```
8446 \renewcommand{\subglossentry}[3]{%
8447   \hangindent##1\glstreeindent\relax
8448   \parindent##1\glstreeindent\relax
8449   \ifnum##1=1\relax
8450     \glssubentryitem{##2}%
8451     \fi
8452     \glstarget{##2}{\strut}%
8453     \glossentrydesc{##2}\glspostdescription\space##3\par
8454   }%
```

Vertical gap between groups is the same as that used by indices:

```
8455 \renewcommand*{\glsgroupskip}{\ifglsgroupskip\else\indexspace\fi}%
8456 }
```

treenonamegroup Like the `treenoname` style but the glossary groups have headings.

```
8457 \newglossarystyle{treenonamegroup}{%
```

Base it on the `glostyletreenoname` style:

```
8458 \setglossarystyle{treenoname}%
```

Give each group a heading:

```
8459 \renewcommand{\glsgroupheading}[1]{\par
```

```
8460 \noindent\glstreenamefmt{\glsgrouptitle{##1}}\par\indexspace}%
```

```
8461 }
```

treenonamehypergroup The `treenonamehypergroup` style is like the `treenonamegroup` style, but has a set of links to the groups at the start of the glossary.

```
8462 \newglossarystyle{treenonamehypergroup}{%
```

Base it on the `glostyletreenoname` style:

```
8463 \setglossarystyle{treenoname}%
```

Put navigation links to the groups at the start of the `theglossary` environment:

```
8464 \renewcommand*{\glossaryheader}{%
```

```
8465 \par\noindent\glstreenamefmt{\glshnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
8466 \renewcommand*{\glsgroupheading}[1]{%
```

```
8467 \par\noindent
```

```
8468 \glstreenamefmt{\glshnavhypertarget{##1}{\glsgrouptitle{##1}}}\par
```

```
8469 \indexspace}%
```

```
8470 }
```

\glissetwidest `\glissetwidest[level]{text}` sets the widest text for the given level. It is used by the `alttree` glossary styles to determine the indentation of each level.

```
8471 \newcommand*{\glissetwidest}[2][0]{%
```

```
8472 \expandafter\def\csname @glswidestname\romannumeral#1\endcsname{%
```

```
8473 #2}%
```

```
8474 }
```

\@glswidestname Initialise `\@glswidestname`.

```
8475 \newcommand*{\@glswidestname}{}
```

alttree The `alttree` glossary style is similar in style to the `tree` style, but the indentation is obtained from the width of `\@glswidestname` which is set using `\glissetwidest`.

```
8476 \newglossarystyle{alttree}{%
```


Redefine theglossary environment.

```
8477 \renewenvironment{theglossary}%  
8478     {\def\@gls@prevlevel{-1}%  
8479      \mbox{}\par}%  
8480     {\par}%
```

Set the header and group headers to nothing.

```
8481 \renewcommand*\glossaryheader{}%  
8482 \renewcommand*\glsgroupheading[1]{}%
```

Redefine the way that the level 0 entries are displayed.

```
8483 \renewcommand{\glosentry}[2]{%  
8484     \ifnum\@gls@prevlevel=0\relax  
8485     \else
```

Find out how big the indentation should be by measuring the widest entry.

```
8486         \settowidth{\glstreeindent}{\glstreenamfmt{\@glswidestname\space}}%  
8487     \fi
```

Set the hangindent and paragraph indent.

```
8488     \hangindent\glstreeindent  
8489     \parindent\glstreeindent
```

Put the name to the left of the paragraph block.

```
8490     \makebox[0pt][r]{\makebox[\glstreeindent][l]{%  
8491         \glsentryitem{##1}\glstreenamfmt{\glstarget{##1}{\glosentryname{##1}}}}}%
```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
8492     \ifglshassymbol{##1}{(\glosentrysymbol{##1})\space}{}%
```

Do the description followed by the description terminator and location list.

```
8493     \glosentrydesc{##1}\glspostdescription \space ##2\par
```

Set the previous level to 0.

```
8494     \def\@gls@prevlevel{0}%  
8495 }%
```

Redefine the way sub-entries are displayed.

```
8496 \renewcommand{\subglosentry}[3]{%
```

Increment and display the sub-entry counter if this is a level 1 entry and the sub-entry counter is in use.

```
8497     \ifnum##1=1\relax  
8498         \glssubentryitem{##2}%  
8499     \fi
```

If the level hasn't changed, keep the same settings, otherwise adjust \glstreeindent accordingly.

```
8500     \ifnum\@gls@prevlevel=##1\relax  
8501     \else
```

Compute the widest entry for this level, or for level 0 if not defined for this level.
Store in \gls@tmplen

```

8502     \@ifundefined{glswidestname\romannumeral##1}{%
8503         \settowidth{gls@tmplen}{\glstreenamfmt{\glswidestname\space}}{%
8504         \settowidth{gls@tmplen}{\glstreenamfmt{%
8505             \csname glswidestname\romannumeral##1\endcsname\space}}}%

```

Determine if going up or down a level

```

8506     \ifnum\@gls@prevlevel<##1\relax

```

Depth has increased, so add the width of the widest entry to \glstreeindent.

```

8507         \setlength\glstreeindent\gls@tmplen
8508         \addtolength\glstreeindent\parindent
8509         \parindent\glstreeindent
8510     \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.

```

8511         \@ifundefined{glswidestname\romannumeral\@gls@prevlevel}{%
8512             \settowidth{\glstreeindent}{\glstreenamfmt{%
8513                 \glswidestname\space}}{%
8514             \settowidth{\glstreeindent}{\glstreenamfmt{%
8515                 \csname glswidestname\romannumeral\@gls@prevlevel
8516                     \endcsname\space}}}%

```

Subtract this length from the previous level's paragraph indent and set to \glstreeindent.

```

8517         \addtolength\parindent{-\glstreeindent}%
8518         \setlength\glstreeindent\parindent
8519     \fi
8520 \fi

```

Set the hanging indentation.

```

8521     \hangindent\glstreeindent

```

Put the name to the left of the paragraph block

```

8522     \makebox[0pt][r]{\makebox[\gls@tmplen][l]{%
8523         \glstreenamfmt{\glstarget{##2}{\glossentryname{##2}}}}}%

```

If the symbol is missing, ignore it, otherwise put it in brackets.

```

8524     \ifglshassymbol{##2}{(\glossentrysymbol{##2})\space}{}%

```

Do the description followed by the description terminator and location list.

```

8525     \glossentrydesc{##2}\glspostdescription\space ##3\par

```

Set the previous level macro to the current level.

```

8526     \def\@gls@prevlevel{##1}%
8527 }%

```

Vertical gap between groups is the same as that used by indices:

```

8528 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
8529 }

```

almtreegroup Like the almtree style but the glossary groups have headings.

```
8530 \newglossarystyle{almtreegroup}{%
      Base it on the glosstylealmtree style:
8531   \setglossarystyle{almtree}%
      Give each group a heading.
8532   \renewcommand{\glsgroupheading}[1]{\par
8533     \def\@gls@prevlevel{-1}%
8534     \hangindent0pt\relax
8535     \parindent0pt\relax
8536     \glstreenamfmt{\glsgetgrouptitle{##1}}\par\indexspace}%
8537 }
```

almtreehypergroup The almtreehypergroup style is like the almtreegroup style, but has a set of links to the groups at the start of the glossary.

```
8538 \newglossarystyle{almtreehypergroup}{%
      Base it on the glosstylealmtree style:
8539   \setglossarystyle{almtree}%
      Put the navigation links in the header
8540   \renewcommand*{\glossaryheader}{%
8541     \par
8542     \def\@gls@prevlevel{-1}%
8543     \hangindent0pt\relax
8544     \parindent0pt\relax
8545     \glstreenamfmt{\glsnavigation}\par\indexspace}%
      Put a hypertarget at the start of each group
8546   \renewcommand*{\glsgroupheading}[1]{%
8547     \par
8548     \def\@gls@prevlevel{-1}%
8549     \hangindent0pt\relax
8550     \parindent0pt\relax
8551     \glstreenamfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8552     \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.

```
8553 \NeedsTeXFormat{LaTeX2e}
8554 \ProvidesPackage{glossaries-compatible-207}[2011/04/02 v1.0 (NLCT)]
```

\GlsAddXdyAttribute Adds an attribute in old format.

```
8555 \ifglsxindy
8556   \renewcommand*\GlsAddXdyAttribute[1]{%
```

```

8557 \edef\@xdyattributes{\@xdyattributes ^^J \string"#1\string"}%
8558 \expandafter\toks@\expandafter{\@xdylocref}%
8559 \edef\@xdylocref{\the\toks@ ^^J%
8560 (markup-locref
8561 :open \string"\string~n\string\setentrycounter
8562 {\noexpand\glscounter}%
8563 \expandafter\string\csname#1\endcsname
8564 \expandafter\@gobble\string\{\string" ^^J
8565 :close \string"\expandafter\@gobble\string\}\string" ^^J
8566 :attr \string"#1\string")}}

```

Only has an effect before \writeist:

```

8567 \fi

```

\GlsAddXdyCounters

```

8568 \renewcommand*\GlsAddXdyCounters[1]{%
8569 \GlossariesWarning{\string\GlsAddXdyCounters\space not available
8570 in compatibility mode.}%
8571 }

```

Add predefined attributes

```

8572 \GlsAddXdyAttribute{glnumberformat}
8573 \GlsAddXdyAttribute{textrm}
8574 \GlsAddXdyAttribute{textsf}
8575 \GlsAddXdyAttribute{texttt}
8576 \GlsAddXdyAttribute{textbf}
8577 \GlsAddXdyAttribute{textmd}
8578 \GlsAddXdyAttribute{textit}
8579 \GlsAddXdyAttribute{textup}
8580 \GlsAddXdyAttribute{textsl}
8581 \GlsAddXdyAttribute{textsc}
8582 \GlsAddXdyAttribute{emph}
8583 \GlsAddXdyAttribute{glshypernumber}
8584 \GlsAddXdyAttribute{hyperrrm}
8585 \GlsAddXdyAttribute{hypersf}
8586 \GlsAddXdyAttribute{hypertt}
8587 \GlsAddXdyAttribute{hyperbf}
8588 \GlsAddXdyAttribute{hypermd}
8589 \GlsAddXdyAttribute{hyperit}
8590 \GlsAddXdyAttribute{hyperup}
8591 \GlsAddXdyAttribute{hypersl}
8592 \GlsAddXdyAttribute{hypersc}
8593 \GlsAddXdyAttribute{hyperemph}

```

\GlsAddXdyLocation Restore v2.07 definition:

```

8594 \ifglxindy
8595 \renewcommand*\GlsAddXdyLocation[2]{%
8596 \edef\@xdyuserlocationdefs{%
8597 \@xdyuserlocationdefs ^^J%

```

```

8598      (define-location-class \string"#1\string"^^J\space\space
8599      \space(#2))
8600    }%
8601    \edef\xdyuserlocationnames{%
8602      \xdyuserlocationnames^^J\space\space\space
8603      \string"#1\string"}%
8604  }
8605 \fi

```

\@do@wrglossary

```
8606 \renewcommand{\@do@wrglossary}[1]{%
```

Determine whether to use xindy or makeindex syntax

```
8607 \ifglxindy
```

Need to determine if the formatting information starts with a (or) indicating a range.

```

8608 \expandafter\@glo@check@mkidxrangechar\@glsnumberformat\@nil
8609 \def\@glo@range{%
8610 \expandafter\if\@glo@prefix(\relax
8611 \def\@glo@range{:open-range}%
8612 \else
8613 \expandafter\if\@glo@prefix)\relax
8614 \def\@glo@range{:close-range}%
8615 \fi
8616 \fi

```

Get the location and escape any special characters

```

8617 \protected@edef\@glslocref{\theglentrycounter}%
8618 \@gls@checkmkidxchars\@glslocref

```

Write to the glossary file using xindy syntax.

```

8619 \glossary[\csname glo@#1@type\endcsname]{%
8620 (indexentry :key (\csname glo@#1@index\endcsname)
8621 :locref \string"\@glslocref\string" %
8622 :attr \string"\@glo@suffix\string" \@glo@range
8623 )
8624 }%
8625 \else

```

Convert the format information into the format required for makeindex

```
8626 \@set@glo@numformat\@glo@numfmt\@gls@counter\@glsnumberformat
```

Write to the glossary file using makeindex syntax.

```

8627 \glossary[\csname glo@#1@type\endcsname]{%
8628 \string\glossaryentry{\csname glo@#1@index\endcsname
8629 \@gls@encapchar\@glo@numfmt}\theglentrycounter}}%
8630 \fi
8631 }

```

\@set@glo@numformat Only had 3 arguments in v2.07

```

8632 \def\@set@glo@numformat#1#2#3{%
8633   \expandafter\@glo@check@mkidxrangechar#3\@nil
8634   \protected@edef#1{%
8635     \@glo@prefix setentrycounter[]{#2}%
8636     \expandafter\string\csname\@glo@suffix\endcsname
8637   }%
8638   \@gls@checkmkidxchars#1%
8639 }

```

\writeist Redefine \writeist back to the way it was in v2.07, but change \istfile to
\glswrite.

```

8640 \ifglxindy
8641   \def\writeist{%
8642     \openout\glswrite=\istfilename
8643     \write\glswrite{;; xindy style file created by the glossaries
8644       package in compatible-2.07 mode}%
8645     \write\glswrite{;; for document '\jobname' on
8646       \the\year-\the\month-\the\day}%
8647     \write\glswrite{^^J; required styles^^J}
8648     \for\@xdystyle:=\@xdyrequiredstyles\do{%
8649       \ifx\@xdystyle\@empty
8650         \else
8651           \protected@write\glswrite{{(require
8652             \string"\@xdystyle.xdy\string")}}%
8653         \fi
8654       }%
8655     \write\glswrite{^^J%
8656       ; list of allowed attributes (number formats)^^J}%
8657     \write\glswrite{(define-attributes ((\@xdyattributes)))}%
8658     \write\glswrite{^^J; user defined alphabets^^J}%
8659     \write\glswrite{\@xdyuseralphabets}%
8660     \write\glswrite{^^J; location class definitions^^J}%
8661     \protected@edef\@gls@roman{\@roman{0}\string"
8662       \string"roman-numbers-lowercase\string" :sep \string"}}%
8663     \@onelevel@sanitize\@gls@roman
8664     \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
8665       :sep \string"}%
8666     \@onelevel@sanitize\@tmp
8667     \ifx\@tmp\@gls@roman
8668       \write\glswrite{(define-location-class
8669         \string"roman-page-numbers\string"^^J\space\space\space
8670         (\string"roman-numbers-lowercase\string")
8671         :min-range-length \@glsminrange)}}%
8672     \else
8673       \write\glswrite{(define-location-class
8674         \string"roman-page-numbers\string"^^J\space\space\space
8675         (:sep "\@gls@roman")
8676         :min-range-length \@glsminrange)}}%
8677     \fi

```

```

8678 \write\glswrite{(define-location-class
8679   \string"Roman-page-numbers\string"^^J\space\space\space
8680   (\string"roman-numbers-uppercase\string")
8681   :min-range-length \@glxminrange)}}%
8682 \write\glswrite{(define-location-class
8683   \string"arabic-page-numbers\string"^^J\space\space\space
8684   (\string"arabic-numbers\string")
8685   :min-range-length \@glxminrange)}}%
8686 \write\glswrite{(define-location-class
8687   \string"alpha-page-numbers\string"^^J\space\space\space
8688   (\string"alpha\string")
8689   :min-range-length \@glxminrange)}}%
8690 \write\glswrite{(define-location-class
8691   \string"Alpha-page-numbers\string"^^J\space\space\space
8692   (\string"ALPHA\string")
8693   :min-range-length \@glxminrange)}}%
8694 \write\glswrite{(define-location-class
8695   \string"Appendix-page-numbers\string"^^J\space\space\space
8696   (\string"ALPHA\string"
8697   :sep \string"@glsAlphacompositor\string"
8698   \string"arabic-numbers\string")
8699   :min-range-length \@glxminrange)}}%
8700 \write\glswrite{(define-location-class
8701   \string"arabic-section-numbers\string"^^J\space\space\space
8702   (\string"arabic-numbers\string"
8703   :sep \string"glscompositor\string"
8704   \string"arabic-numbers\string")
8705   :min-range-length \@glxminrange)}}%
8706 \write\glswrite{^^J; user defined location classes}%
8707 \write\glswrite{@xdyuserlocationdefs}%
8708 \write\glswrite{^^J; define cross-reference class^^J}%
8709 \write\glswrite{(define-crossref-class \string"see\string"
8710   :unverified )}%
8711 \write\glswrite{(markup-crossref-list
8712   :class \string"see\string"^^J\space\space\space
8713   :open \string"\string\glsseeformat\string"
8714   :close \string"{}\string")}%
8715 \write\glswrite{^^J; define the order of the location classes}%
8716 \write\glswrite{(define-location-class-order
8717   (\@xdylocationclassorder))}%
8718 \write\glswrite{^^J; define the glossary markup^^J}%
8719 \write\glswrite{(markup-index^^J\space\space\space
8720   :open \string"\string
8721   \glossarysection[\string\glossarytoctitle]{\string
8722   \glossarytitle}\string\glossarypreamble\string~n\string\begin
8723   {theglossary}\string\glossaryheader\string~n\string" ^^J\space
8724   \space\space:close \string"\expandafter\@gobble
8725   \string\%\string~n\string
8726   \end{theglossary}\string\glossarypostamble

```

```

8727     \string~n\string" ^^J\space\space\space
8728     :tree)}}%
8729     \write\glswrite{(markup-letter-group-list
8730     :sep \string"\string\glsgroupskip\string~n\string"))}%
8731     \write\glswrite{(markup-indexentry
8732     :open \string"\string\relax \string\glsresetentrylist
8733     \string~n\string"))}%
8734     \write\glswrite{(markup-locclass-list :open
8735     \string"\glsopenbrace\string\glossaryentrynumbers
8736     \glsopenbrace\string\relax\space \string"^^J\space\space\space
8737     :sep \string", \string"
8738     :close \string"\glsclosebrace\glsclosebrace\string"))}%
8739     \write\glswrite{(markup-locref-list
8740     :sep \string"\string\delimN\space\string"))}%
8741     \write\glswrite{(markup-range
8742     :sep \string"\string\delimR\space\string"))}%
8743     \@onelevel@sanitize\gls@suffixF
8744     \@onelevel@sanitize\gls@suffixFF
8745     \ifx\gls@suffixF\@empty
8746     \else
8747     \write\glswrite{(markup-range
8748     :close "\gls@suffixF" :length 1 :ignore-end)}}%
8749     \fi
8750     \ifx\gls@suffixFF\@empty
8751     \else
8752     \write\glswrite{(markup-range
8753     :close "\gls@suffixFF" :length 2 :ignore-end)}}%
8754     \fi
8755     \write\glswrite{^^J; define format to use for locations^^J}%
8756     \write\glswrite{\@xdylocref}%
8757     \write\glswrite{^^J; define letter group list format^^J}%
8758     \write\glswrite{(markup-letter-group-list
8759     :sep \string"\string\glsgroupskip\string~n\string"))}%
8760     \write\glswrite{^^J; letter group headings^^J}%
8761     \write\glswrite{(markup-letter-group
8762     :open-head \string"\string\glsgroupheading
8763     \glsopenbrace\string"^^J\space\space\space
8764     :close-head \string"\glsclosebrace\string"))}%
8765     \write\glswrite{^^J; additional letter groups^^J}%
8766     \write\glswrite{\@xdylettergroups}%
8767     \write\glswrite{^^J; additional sort rules^^J}%
8768     \write\glswrite{\@dysortrules}%
8769     \noist}
8770 \else
8771     \edef\@gls@actualchar{\string?}
8772     \edef\@gls@encapchar{\string|}
8773     \edef\@gls@levelchar{\string!}
8774     \edef\@gls@quotechar{\string"}
8775     \def\writeist{\relax

```



```

8776 \openout\glswrite=\istfilename
8777 \write\glswrite{\expandafter\@gobble\string\% makeindex style file
8778   created by the glossaries package}
8779 \write\glswrite{\expandafter\@gobble\string\% for document
8780   '\jobname' on \the\year-\the\month-\the\day}
8781 \write\glswrite{actual '\@gls@actualchar'}
8782 \write\glswrite{encap '\@gls@encapchar'}
8783 \write\glswrite{level '\@gls@levelchar'}
8784 \write\glswrite{quote '\@gls@quotechar'}
8785 \write\glswrite{keyword \string"\string\glossaryentry\string"}
8786 \write\glswrite{preamble \string"\string\glossarysection[\string
8787   \glossarytoctitle]{\string\glossarytitle}\string
8788   \glossarypreamble\string\n\string\begin{theglossary}\string
8789   \glossaryheader\string\n\string"}
8790 \write\glswrite{postamble \string"\string%\string\n\string
8791   \end{theglossary}\string\glossarypostamble\string\n
8792   \string"}
8793 \write\glswrite{group_skip \string"\string\glsgroupskip\string\n
8794   \string"}
8795 \write\glswrite{item_0 \string"\string%\string\n\string"}
8796 \write\glswrite{item_1 \string"\string%\string\n\string"}
8797 \write\glswrite{item_2 \string"\string%\string\n\string"}
8798 \write\glswrite{item_01 \string"\string%\string\n\string"}
8799 \write\glswrite{item_x1
8800   \string"\string\relax \string\glsresetentrylist\string\n
8801   \string"}
8802 \write\glswrite{item_12 \string"\string%\string\n\string"}
8803 \write\glswrite{item_x2
8804   \string"\string\relax \string\glsresetentrylist\string\n
8805   \string"}
8806 \write\glswrite{delim_0 \string"\string\{\string
8807   \glossaryentrynumbers\string\{\string\relax \string"}
8808 \write\glswrite{delim_1 \string"\string\{\string
8809   \glossaryentrynumbers\string\{\string\relax \string"}
8810 \write\glswrite{delim_2 \string"\string\{\string
8811   \glossaryentrynumbers\string\{\string\relax \string"}
8812 \write\glswrite{delim_t \string"\string\}\string\}\string"}
8813 \write\glswrite{delim_n \string"\string\delimN \string"}
8814 \write\glswrite{delim_r \string"\string\delimR \string"}
8815 \write\glswrite{headings_flag 1}
8816 \write\glswrite{heading_prefix
8817   \string"\string\glsgroupheading\string\{\string"}
8818 \write\glswrite{heading_suffix
8819   \string"\string\}\string\relax
8820   \string\glsresetentrylist \string"}
8821 \write\glswrite{symhead_positive \string"glssymbols\string"}
8822 \write\glswrite{numhead_positive \string"glslnumbers\string"}
8823 \write\glswrite{page_compositor \string"glscpositor\string"}
8824 \@gls@escbsdq\gls@suffixF

```

```

8825 \@gls@escbsdq\gls@suffixFF
8826 \ifx\gls@suffixF\@empty
8827 \else
8828 \write\glswrite{suffix_2p \string"\gls@suffixF\string"}
8829 \fi
8830 \ifx\gls@suffixFF\@empty
8831 \else
8832 \write\glswrite{suffix_3p \string"\gls@suffixFF\string"}
8833 \fi
8834 \noist
8835 }
8836 \fi

```

\noist

```
8837 \renewcommand*{\noist}{\let\writeist\relax}
```

Compatibility macros.

```

8838 \NeedsTeXFormat{LaTeX2e}
8839 \ProvidesPackage{glossaries-compatible-307}[2013/11/14 v4.0 (NLCT)]

```

Compatibility macros for predefined glossary styles:

compatglossarystyle Defines a compatibility glossary style.

```

8840 \newcommand{\compatglossarystyle}[2]{%
8841 \ifcsundef{@glscompstyle@#1}%
8842 {%
8843 \csdef{@glscompstyle@#1}{#2}%
8844 }%
8845 {%
8846 \PackageError{glossaries}{Glossary compatibility style ‘#1’ is already defined}{}%
8847 }%
8848 }

```

Backward compatible inline style.

```

8849 \compatglossarystyle{inline}{%
8850 \renewcommand{\glossaryentryfield}[5]{%
8851 \glsinlinedopostchild
8852 \gls@inlinesep
8853 \def\glo@desc{##3}%
8854 \def\@no@post@desc{\nopostdesc}%
8855 \glsentryitem{##1}\glsinlinenameformat{##1}{##2}%
8856 \ifx\glo@desc\@no@post@desc
8857 \glsinlineemptydescformat{##4}{##5}%
8858 \else
8859 \ifstrempy{##3}%
8860 {\glsinlineemptydescformat{##4}{##5}}%
8861 {\glsinlinedescformat{##3}{##4}{##5}}%
8862 \fi
8863 \ifglshaschildren{##1}%
8864 {%

```

```

8865      \glsresetsubentrycounter
8866      \glsinlineparentchildseparator
8867      \def\gls@inlinesubsep{%
8868      \def\gls@inlinepostchild{\glsinlinepostchild}%
8869      }%
8870      {}%
8871      \def\gls@inlinesep{\glsinlineseparator}%
8872      }%

Sub-entries display description:
8873      \renewcommand{\glossarysubentryfield}[6]{%
8874      \gls@inlinesubsep%
8875      \glsinlinesubnameformat{##2}{##3}%
8876      \glssubentryitem{##2}\glsinlinesubdescformat{##4}{##5}{##6}%
8877      \def\gls@inlinesubsep{\glsinlinesubseparator}%
8878      }%
8879      }

Backward compatible list style.
8880 \compatglossarystyle{list}{%
8881      \renewcommand*\glossaryentryfield[5]{%
8882      \item[\glsentryitem{##1}\glstarget{##1}{##2}]
8883      ##3\glspostdescription\space ##5}%

Sub-entries continue on the same line:
8884      \renewcommand*\glossarysubentryfield[6]{%
8885      \glssubentryitem{##2}%
8886      \glstarget{##2}{\strut}##4\glspostdescription\space ##6.}%
8887      }

Backward compatible listgroup style.
8888 \compatglossarystyle{listgroup}{%
8889      \csuse{@glscompstyle@list}%
8890      }%

Backward compatible listhypergroup style.
8891 \compatglossarystyle{listhypergroup}{%
8892      \csuse{@glscompstyle@list}%
8893      }%

Backward compatible altlist style.
8894 \compatglossarystyle{altlist}{%
8895      \renewcommand*\glossaryentryfield[5]{%
8896      \item[\glsentryitem{##1}\glstarget{##1}{##2}]%
8897      \mbox{}\par\nobreak\@afterheading
8898      ##3\glspostdescription\space ##5}%
8899      \renewcommand{\glossarysubentryfield}[6]{%
8900      \par
8901      \glssubentryitem{##2}%
8902      \glstarget{##2}{\strut}##4\glspostdescription\space ##6}%
8903      }%

```

Backward compatible altlistgroup style.

```
8904 \compatglossarystyle{altlistgroup}{%  
8905 \csuse{@glscompstyle@altlist}%  
8906 }%
```

Backward compatible altlisthypergroup style.

```
8907 \compatglossarystyle{altlisthypergroup}{%  
8908 \csuse{@glscompstyle@altlist}%  
8909 }%
```

Backward compatible listdotted style.

```
8910 \compatglossarystyle{listdotted}{%  
8911 \renewcommand*{\glossaryentryfield}[5]{%  
8912 \item[]\makebox[\glslistdottedwidth][l]{%  
8913 \glentryitem{##1}\glstarget{##1}{##2}%  
8914 \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##3}%  
8915 \renewcommand*{\glossarysubentryfield}[6]{%  
8916 \item[]\makebox[\glslistdottedwidth][l]{%  
8917 \glssubentryitem{##2}%  
8918 \glstarget{##2}{##3}%  
8919 \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##4}%  
8920 }%
```

Backward compatible sublistdotted style.

```
8921 \compatglossarystyle{sublistdotted}{%  
8922 \csuse{@glscompstyle@listdotted}%  
8923 \renewcommand*{\glossaryentryfield}[5]{%  
8924 \item[\glentryitem{##1}\glstarget{##1}{##2}]}%  
8925 }%
```

Backward compatible long style.

```
8926 \compatglossarystyle{long}{%  
8927 \renewcommand*{\glossaryentryfield}[5]{%  
8928 \glentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%  
8929 \renewcommand*{\glossarysubentryfield}[6]{%  
8930 &  
8931 \glssubentryitem{##2}%  
8932 \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%  
8933 }%
```

Backward compatible longborder style.

```
8934 \compatglossarystyle{longborder}{%  
8935 \csuse{@glscompstyle@long}%  
8936 }%
```

Backward compatible longheader style.

```
8937 \compatglossarystyle{longheader}{%  
8938 \csuse{@glscompstyle@long}%  
8939 }%
```

Backward compatible longheaderborder style.

```
8940 \compatglossarystyle{longheaderborder}{%
```

```

8941 \csuse{@glscompstyle@long}%
8942 }%

```

Backward compatible long3col style.

```

8943 \compatglossarystyle{long3col}{%
8944   \renewcommand*{\glossaryentryfield}[5]{%
8945     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\\}%
8946   \renewcommand*{\glossarysubentryfield}[6]{%
8947     &
8948     \glssubentryitem{##2}%
8949     \glstarget{##2}{\strut}##4 & ##6\\}%
8950 }%

```

Backward compatible long3colborder style.

```

8951 \compatglossarystyle{long3colborder}{%
8952   \csuse{@glscompstyle@long3col}%
8953 }%

```

Backward compatible long3colheader style.

```

8954 \compatglossarystyle{long3colheader}{%
8955   \csuse{@glscompstyle@long3col}%
8956 }%

```

Backward compatible long3colheaderborder style.

```

8957 \compatglossarystyle{long3colheaderborder}{%
8958   \csuse{@glscompstyle@long3col}%
8959 }%

```

Backward compatible long4col style.

```

8960 \compatglossarystyle{long4col}{%
8961   \renewcommand*{\glossaryentryfield}[5]{%
8962     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%
8963   \renewcommand*{\glossarysubentryfield}[6]{%
8964     &
8965     \glssubentryitem{##2}%
8966     \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%
8967 }%

```

Backward compatible long4colheader style.

```

8968 \compatglossarystyle{long4colheader}{%
8969   \csuse{@glscompstyle@long4col}%
8970 }%

```

Backward compatible long4colborder style.

```

8971 \compatglossarystyle{long4colborder}{%
8972   \csuse{@glscompstyle@long4col}%
8973 }%

```

Backward compatible long4colheaderborder style.

```

8974 \compatglossarystyle{long4colheaderborder}{%
8975   \csuse{@glscompstyle@long4col}%
8976 }%

```

Backward compatible altlong4col style.

```
8977 \compatglossarystyle{altlong4col}{%  
8978 \csuse{@glscompstyle@long4col}%  
8979 }%
```

Backward compatible altlong4colheader style.

```
8980 \compatglossarystyle{altlong4colheader}{%  
8981 \csuse{@glscompstyle@long4col}%  
8982 }%
```

Backward compatible altlong4colborder style.

```
8983 \compatglossarystyle{altlong4colborder}{%  
8984 \csuse{@glscompstyle@long4col}%  
8985 }%
```

Backward compatible altlong4colheaderborder style.

```
8986 \compatglossarystyle{altlong4colheaderborder}{%  
8987 \csuse{@glscompstyle@long4col}%  
8988 }%
```

Backward compatible long style.

```
8989 \compatglossarystyle{longragged}{%  
8990 \renewcommand*{\glossaryentryfield}[5]{%  
8991 \glstryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5%  
8992 \tabularnewline}%  
8993 \renewcommand*{\glossarysubentryfield}[6]{%  
8994 &  
8995 \glssubentryitem{##2}%  
8996 \glstarget{##2}{\strut}##4\glspostdescription\space ##6%  
8997 \tabularnewline}%  
8998 }%
```

Backward compatible longraggedborder style.

```
8999 \compatglossarystyle{longraggedborder}{%  
9000 \csuse{@glscompstyle@longragged}%  
9001 }%
```

Backward compatible longraggedheader style.

```
9002 \compatglossarystyle{longraggedheader}{%  
9003 \csuse{@glscompstyle@longragged}%  
9004 }%
```

Backward compatible longraggedheaderborder style.

```
9005 \compatglossarystyle{longraggedheaderborder}{%  
9006 \csuse{@glscompstyle@longragged}%  
9007 }%
```

Backward compatible longragged3col style.

```
9008 \compatglossarystyle{longragged3col}{%  
9009 \renewcommand*{\glossaryentryfield}[5]{%  
9010 \glstryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%  
9011 \renewcommand*{\glossarysubentryfield}[6]{%
```

```

9012      &
9013      \glssubentryitem{##2}%
9014      \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%
9015 }%

Backward compatible longragged3colborder style.
9016 \compatglossarystyle{longragged3colborder}{%
9017 \csuse{@glscmpstyle@longragged3col}%
9018 }%

Backward compatible longragged3colheader style.
9019 \compatglossarystyle{longragged3colheader}{%
9020 \csuse{@glscmpstyle@longragged3col}%
9021 }%

Backward compatible longragged3colheaderborder style.
9022 \compatglossarystyle{longragged3colheaderborder}{%
9023 \csuse{@glscmpstyle@longragged3col}%
9024 }%

Backward compatible altlongragged4col style.
9025 \compatglossarystyle{altlongragged4col}{%
9026 \renewcommand*{\glossaryentryfield}[5]{%
9027 \glssentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%
9028 \renewcommand*{\glossarysubentryfield}[6]{%
9029 &
9030 \glssubentryitem{##2}%
9031 \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%
9032 }%

Backward compatible altlongragged4colheader style.
9033 \compatglossarystyle{altlongragged4colheader}{%
9034 \csuse{@glscmpstyle@altlong4col}%
9035 }%

Backward compatible altlongragged4colborder style.
9036 \compatglossarystyle{altlongragged4colborder}{%
9037 \csuse{@glscmpstyle@altlong4col}%
9038 }%

Backward compatible altlongragged4colheaderborder style.
9039 \compatglossarystyle{altlongragged4colheaderborder}{%
9040 \csuse{@glscmpstyle@altlong4col}%
9041 }%

Backward compatible index style.
9042 \compatglossarystyle{index}{%
9043 \renewcommand*{\glossaryentryfield}[5]{%
9044 \item\glssentryitem{##1}\textbf{\glstarget{##1}{##2}}%
9045 \ifx\relax##4\relax
9046 \else
9047 \space(##4)%

```

```

9048     \fi
9049     \space ##3\glspostdescription \space ##5}%
9050 \renewcommand*{\glossarysubentryfield}[6]{%
9051     \ifcase##1\relax
9052     % level 0
9053     \item
9054     \or
9055     % level 1
9056     \subitem
9057     \glssubentryitem{##2}%
9058     \else
9059     % all other levels
9060     \subsubitem
9061     \fi
9062     \textbf{\glstarget{##2}{##3}}%
9063     \ifx\relax##5\relax
9064     \else
9065     \space(##5)%
9066     \fi
9067     \space##4\glspostdescription\space ##6}%
9068 }%

```

Backward compatible indexgroup style.

```

9069 \compatglossarystyle{indexgroup}{%
9070 \csuse{@glscmpstyle@index}%
9071 }%

```

Backward compatible indexhypergroup style.

```

9072 \compatglossarystyle{indexhypergroup}{%
9073 \csuse{@glscmpstyle@index}%
9074 }%

```

Backward compatible tree style.

```

9075 \compatglossarystyle{tree}{%
9076 \renewcommand{\glossaryentryfield}[5]{%
9077     \hangindent0pt\relax
9078     \parindent0pt\relax
9079     \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
9080     \ifx\relax##4\relax
9081     \else
9082     \space(##4)%
9083     \fi
9084     \space ##3\glspostdescription \space ##5\par}%
9085 \renewcommand{\glossarysubentryfield}[6]{%
9086     \hangindent##1\glstreeindent\relax
9087     \parindent##1\glstreeindent\relax
9088     \ifnum##1=1\relax
9089     \glssubentryitem{##2}%
9090     \fi
9091     \textbf{\glstarget{##2}{##3}}%
9092     \ifx\relax##5\relax

```



```

9093 \else
9094 \space(##5)%
9095 \fi
9096 \space##4\glspostdescription\space ##6\par}%
9097 }%

Backward compatible treegroup style.
9098 \compatglossarystyle{treegroup}{%
9099 \csuse{@glscmpstyle@tree}%
9100 }%

Backward compatible treehypergroup style.
9101 \compatglossarystyle{treehypergroup}{%
9102 \csuse{@glscmpstyle@tree}%
9103 }%

Backward compatible treenoname style.
9104 \compatglossarystyle{treenoname}{%
9105 \renewcommand{\glossaryentryfield}[5]{%
9106 \hangindent0pt\relax
9107 \parindent0pt\relax
9108 \glstryitem{##1}\textbf{\glstarget{##1}{##2}}%
9109 \ifx\relax##4\relax
9110 \else
9111 \space(##4)%
9112 \fi
9113 \space ##3\glspostdescription \space ##5\par}%
9114 \renewcommand{\glossarysubentryfield}[6]{%
9115 \hangindent##1\glstreeindent\relax
9116 \parindent##1\glstreeindent\relax
9117 \ifnum##1=1\relax
9118 \glssubentryitem{##2}%
9119 \fi
9120 \glstarget{##2}{\strut}%
9121 ##4\glspostdescription\space ##6\par}%
9122 }%

Backward compatible treenonamegroup style.
9123 \compatglossarystyle{treenonamegroup}{%
9124 \csuse{@glscmpstyle@treenoname}%
9125 }%

Backward compatible treenonamehypergroup style.
9126 \compatglossarystyle{treenonamehypergroup}{%
9127 \csuse{@glscmpstyle@treenoname}%
9128 }%

Backward compatible alttree style.
9129 \compatglossarystyle{alttree}{%
9130 \renewcommand{\glossaryentryfield}[5]{%
9131 \ifnum \@gls@prevlevel=0\relax
9132 \else

```

```

9133     \settowidth{\glstreeindent}{\textbf{\@glswidestname\space}}%
9134     \hangindent\glstreeindent
9135     \parindent\glstreeindent
9136     \fi
9137     \makebox[0pt][r]{\makebox[\glstreeindent][l]{%
9138       \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}}%
9139     \ifx\relax##4\relax
9140     \else
9141       (##4)\space
9142     \fi
9143     ##3\glspostdescription \space ##5\par
9144     \def\@gls@prevlevel{0}%
9145   }%
9146   \renewcommand{\glossarysubentryfield}[6]{%
9147     \ifnum##1=1\relax
9148       \glssubentryitem{##2}%
9149     \fi
9150     \ifnum\@gls@prevlevel=##1\relax
9151     \else
9152       \@ifundefined{\@glswidestname\romannumeral##1}{%
9153         \settowidth{\gls@tmplen}{\textbf{\@glswidestname\space}}{%
9154         \settowidth{\gls@tmplen}{\textbf{%
9155           \csname \@glswidestname\romannumeral##1\endcsname\space}}}%
9156       \ifnum\@gls@prevlevel<##1\relax
9157         \setlength\glstreeindent\gls@tmplen
9158         \addtolength\glstreeindent\parindent
9159         \parindent\glstreeindent
9160       \else
9161         \@ifundefined{\@glswidestname\romannumeral\@gls@prevlevel}{%
9162           \settowidth{\glstreeindent}{\textbf{%
9163             \@glswidestname\space}}{%
9164           \settowidth{\glstreeindent}{\textbf{%
9165             \csname \@glswidestname\romannumeral\@gls@prevlevel
9166             \endcsname\space}}}%
9167         \addtolength\parindent{-\glstreeindent}%
9168         \setlength\glstreeindent\parindent
9169       \fi
9170     \fi
9171     \hangindent\glstreeindent
9172     \makebox[0pt][r]{\makebox[\gls@tmplen][l]{%
9173       \textbf{\glstarget{##2}{##3}}}%
9174     \ifx##5\relax\relax
9175     \else
9176       (##5)\space
9177     \fi
9178     ##4\glspostdescription\space ##6\par
9179     \def\@gls@prevlevel{##1}%
9180   }%
9181 }%

```

Backward compatible alttreegroup style.

```
9182 \compatglossarystyle{alttreegroup}{%  
9183 \csuse{@glscompstyle@almtree}%  
9184 }%
```

Backward compatible alttreehypergroup style.

```
9185 \compatglossarystyle{alttreehypergroup}{%  
9186 \csuse{@glscompstyle@almtree}%  
9187 }%
```

Backward compatible mcolindex style.

```
9188 \compatglossarystyle{mcolindex}{%  
9189 \csuse{@glscompstyle@index}%  
9190 }%
```

Backward compatible mcolindexgroup style.

```
9191 \compatglossarystyle{mcolindexgroup}{%  
9192 \csuse{@glscompstyle@index}%  
9193 }%
```

Backward compatible mcolindexhypergroup style.

```
9194 \compatglossarystyle{mcolindexhypergroup}{%  
9195 \csuse{@glscompstyle@index}%  
9196 }%
```

Backward compatible mcoltree style.

```
9197 \compatglossarystyle{mcoltree}{%  
9198 \csuse{@glscompstyle@tree}%  
9199 }%
```

Backward compatible mcoltreegroup style.

```
9200 \compatglossarystyle{mcolindextreegroup}{%  
9201 \csuse{@glscompstyle@tree}%  
9202 }%
```

Backward compatible mcoltreehypergroup style.

```
9203 \compatglossarystyle{mcolindextreehypergroup}{%  
9204 \csuse{@glscompstyle@tree}%  
9205 }%
```

Backward compatible mcoltreenoname style.

```
9206 \compatglossarystyle{mcoltreenoname}{%  
9207 \csuse{@glscompstyle@tree}%  
9208 }%
```

Backward compatible mcoltreenonamegroup style.

```
9209 \compatglossarystyle{mcoltreenonamegroup}{%  
9210 \csuse{@glscompstyle@tree}%  
9211 }%
```

Backward compatible mcoltreenonamehypergroup style.

```
9212 \compatglossarystyle{mcoltreenonamehypergroup}{%  
9213 \csuse{@glscompstyle@tree}%  
9214 }%
```

Backward compatible mcolalmtree style.

```
9215 \compatglossarystyle{mcolalmtree}{%
9216 \csuse{@glscmpstyle@almtree}%
9217 }%
```

Backward compatible mcolalmtreegroup style.

```
9218 \compatglossarystyle{mcolalmtreegroup}{%
9219 \csuse{@glscmpstyle@almtree}%
9220 }%
```

Backward compatible mcolalmtreehypergroup style.

```
9221 \compatglossarystyle{mcolalmtreehypergroup}{%
9222 \csuse{@glscmpstyle@almtree}%
9223 }%
```

Backward compatible superragged style.

```
9224 \compatglossarystyle{superragged}{%
9225 \renewcommand*{\glossaryentryfield}[5]{%
9226 \glstentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
9227 \tabularnewline}%
9228 \renewcommand*{\glossarysubentryfield}[6]{%
9229 &
9230 \glssubentryitem{##2}%
9231 \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
9232 \tabularnewline}%
9233 }%
```

Backward compatible superraggedborder style.

```
9234 \compatglossarystyle{superraggedborder}{%
9235 \csuse{@glscmpstyle@superragged}%
9236 }%
```

Backward compatible superraggedheader style.

```
9237 \compatglossarystyle{superraggedheader}{%
9238 \csuse{@glscmpstyle@superragged}%
9239 }%
```

Backward compatible superraggedheaderborder style.

```
9240 \compatglossarystyle{superraggedheaderborder}{%
9241 \csuse{@glscmpstyle@superragged}%
9242 }%
```

Backward compatible superragged3col style.

```
9243 \compatglossarystyle{superragged3col}{%
9244 \renewcommand*{\glossaryentryfield}[5]{%
9245 \glstentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%
9246 \renewcommand*{\glossarysubentryfield}[6]{%
9247 &
9248 \glssubentryitem{##2}%
9249 \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%
9250 }%
```

Backward compatible superragged3colborder style.

```
9251 \compatglossarystyle{superragged3colborder}{%
9252 \csuse{@glscmpstyle@superragged3col}%
9253 }%
```

Backward compatible superragged3colheader style.

```
9254 \compatglossarystyle{superragged3colheader}{%
9255 \csuse{@glscmpstyle@superragged3col}%
9256 }%
```

Backward compatible superragged3colheaderborder style.

```
9257 \compatglossarystyle{superragged3colheaderborder}{%
9258 \csuse{@glscmpstyle@superragged3col}%
9259 }%
```

Backward compatible altsuperragged4col style.

```
9260 \compatglossarystyle{altsuperragged4col}{%
9261 \renewcommand*{\glossaryentryfield}[5]{%
9262 \glstarget{##1}{\glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%
9263 \renewcommand*{\glossarysubentryfield}[6]{%
9264 &
9265 \glssubentryitem{##2}%
9266 \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%
9267 }%
```

Backward compatible altsuperragged4colheader style.

```
9268 \compatglossarystyle{altsuperragged4colheader}{%
9269 \csuse{@glscmpstyle@altsuperragged4col}%
9270 }%
```

Backward compatible altsuperragged4colborder style.

```
9271 \compatglossarystyle{altsuperragged4colborder}{%
9272 \csuse{@glscmpstyle@altsuperragged4col}%
9273 }%
```

Backward compatible altsuperragged4colheaderborder style.

```
9274 \compatglossarystyle{altsuperragged4colheaderborder}{%
9275 \csuse{@glscmpstyle@altsuperragged4col}%
9276 }%
```

Backward compatible super style.

```
9277 \compatglossarystyle{super}{%
9278 \renewcommand*{\glossaryentryfield}[5]{%
9279 \glstarget{##1}{\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
9280 \renewcommand*{\glossarysubentryfield}[6]{%
9281 &
9282 \glssubentryitem{##2}%
9283 \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
9284 }%
```

Backward compatible superborder style.

```
9285 \compatglossarystyle{superborder}{%
```

```

9286 \csuse{@glscompstyle@super}%
9287 }%

```

Backward compatible superheader style.

```

9288 \compatglossarystyle{superheader}{%
9289 \csuse{@glscompstyle@super}%
9290 }%

```

Backward compatible superheaderborder style.

```

9291 \compatglossarystyle{superheaderborder}{%
9292 \csuse{@glscompstyle@super}%
9293 }%

```

Backward compatible super3col style.

```

9294 \compatglossarystyle{super3col}{%
9295 \renewcommand*{\glossaryentryfield}[5]{%
9296 \glstentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\\}%
9297 \renewcommand*{\glossarysubentryfield}[6]{%
9298 &
9299 \glssubentryitem{##2}%
9300 \glstarget{##2}{\strut}##4 & ##6\\}%
9301 }%

```

Backward compatible super3colborder style.

```

9302 \compatglossarystyle{super3colborder}{%
9303 \csuse{@glscompstyle@super3col}%
9304 }%

```

Backward compatible super3colheader style.

```

9305 \compatglossarystyle{super3colheader}{%
9306 \csuse{@glscompstyle@super3col}%
9307 }%

```

Backward compatible super3colheaderborder style.

```

9308 \compatglossarystyle{super3colheaderborder}{%
9309 \csuse{@glscompstyle@super3col}%
9310 }%

```

Backward compatible super4col style.

```

9311 \compatglossarystyle{super4col}{%
9312 \renewcommand*{\glossaryentryfield}[5]{%
9313 \glstentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%
9314 \renewcommand*{\glossarysubentryfield}[6]{%
9315 &
9316 \glssubentryitem{##2}%
9317 \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%
9318 }%

```

Backward compatible super4colheader style.

```

9319 \compatglossarystyle{super4colheader}{%
9320 \csuse{@glscompstyle@super4col}%
9321 }%

```

Backward compatible super4colborder style.

```
9322 \compatglossarystyle{super4colborder}{%
9323 \csuse{@glscompstyle@super4col}%
9324 }%
```

Backward compatible super4colheaderborder style.

```
9325 \compatglossarystyle{super4colheaderborder}{%
9326 \csuse{@glscompstyle@super4col}%
9327 }%
```

Backward compatible altsuper4col style.

```
9328 \compatglossarystyle{altsuper4col}{%
9329 \csuse{@glscompstyle@super4col}%
9330 }%
```

Backward compatible altsuper4colheader style.

```
9331 \compatglossarystyle{altsuper4colheader}{%
9332 \csuse{@glscompstyle@super4col}%
9333 }%
```

Backward compatible altsuper4colborder style.

```
9334 \compatglossarystyle{altsuper4colborder}{%
9335 \csuse{@glscompstyle@super4col}%
9336 }%
```

Backward compatible altsuper4colheaderborder style.

```
9337 \compatglossarystyle{altsuper4colheaderborder}{%
9338 \csuse{@glscompstyle@super4col}%
9339 }%
```

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.

```
9340 \NeedsTeXFormat{LaTeX2e}
```

Package version number now in line with main glossaries package number but will only be updated when glossaries-accsupp.sty is modified.

```
9341 \ProvidesPackage{glossaries-accsupp}[2014/07/30 v4.08 (NLCT)]
9342 Experimental glossaries accessibility]
```

Pass all options to glossaries:

```
9343 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
9344 \ProcessOptions
```

ompatibleglossentry Override style compatibility macros:

```
9345 \def\compatibleglossentry#1#2{%
```

```

9346 \toks@{#2}%
9347 \protected@edef\@do@glossentry{%
9348   \noexpand\accsuppglossaryentryfield{#1}%
9349   {\noexpand\glsnamefont
9350     {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@name\endcsname}}}%
9351   {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@desc\endcsname}}}%
9352   {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@symbol\endcsname}}}%
9353   {\the\toks@}%
9354 }%
9355 \@do@glossentry
9356 }

```

atiblessubglossentry

```

9357 \def\compatiblessubglossentry#1#2#3{%
9358   \toks@{#3}%
9359   \protected@edef\@do@subglossentry{%
9360     \noexpand\accsuppglossarysubentryfield{\number#1}%
9361     {#2}%
9362     {\noexpand\glsnamefont
9363       {\expandafter\expandonce\csname glo@\glsdetoklabel{#2}@name\endcsname}}}%
9364     {\expandafter\expandonce\csname glo@\glsdetoklabel{#2}@desc\endcsname}}}%
9365     {\expandafter\expandonce\csname glo@\glsdetoklabel{#2}@symbol\endcsname}}}%
9366     {\the\toks@}%
9367   }%
9368   \@do@subglossentry
9369 }

```

Required packages:

```

9370 \RequirePackage{glossaries}
9371 \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:

```

9372 \define@key{glossentry}{access}{%
9373   \def\@glo@access{#1}%
9374 }

```

textaccess The replacement text corresponding to the text key:

```

9375 \define@key{glossentry}{textaccess}{%
9376   \def\@glo@textaccess{#1}%
9377 }

```


firstaccess The replacement text corresponding to the first key:

```
9378 \define@key{glossentry}{firstaccess}{%
9379   \def\@glo@firstaccess{#1}%
9380 }
```

pluralaccess The replacement text corresponding to the plural key:

```
9381 \define@key{glossentry}{pluralaccess}{%
9382   \def\@glo@pluralaccess{#1}%
9383 }
```

firstpluralaccess The replacement text corresponding to the firstplural key:

```
9384 \define@key{glossentry}{firstpluralaccess}{%
9385   \def\@glo@firstpluralaccess{#1}%
9386 }
```

symbolaccess The replacement text corresponding to the symbol key:

```
9387 \define@key{glossentry}{symbolaccess}{%
9388   \def\@glo@symbolaccess{#1}%
9389 }
```

symbolpluralaccess The replacement text corresponding to the symbolplural key:

```
9390 \define@key{glossentry}{symbolpluralaccess}{%
9391   \def\@glo@symbolpluralaccess{#1}%
9392 }
```

descriptionaccess The replacement text corresponding to the description key:

```
9393 \define@key{glossentry}{descriptionaccess}{%
9394   \def\@glo@descaccess{#1}%
9395 }
```

descriptionpluralaccess The replacement text corresponding to the descriptionplural key:

```
9396 \define@key{glossentry}{descriptionpluralaccess}{%
9397   \def\@glo@descpluralaccess{#1}%
9398 }
```

shortaccess The replacement text corresponding to the short key:

```
9399 \define@key{glossentry}{shortaccess}{%
9400   \def\@glo@shortaccess{#1}%
9401 }
```

shortpluralaccess The replacement text corresponding to the shortplural key:

```
9402 \define@key{glossentry}{shortpluralaccess}{%
9403   \def\@glo@shortpluralaccess{#1}%
9404 }
```

longaccess The replacement text corresponding to the long key:

```
9405 \define@key{glossentry}{longaccess}{%
9406   \def\@glo@longaccess{#1}%
9407 }
```

longpluralaccess The replacement text corresponding to the longplural key:

```
9408 \define@key{glossentry}{longpluralaccess}{%
9409   \def\@glo@longpluralaccess{#1}%
9410 }
```

There are no equivalent keys for the user1...user6 keys. The replacement text would have to be explicitly put in the value, e.g., user1={\glsacccsupp{inches}{in}}.

Append these new keys to \@gls@keymap:

```
9411 \appto\@gls@keymap{,%
9412   {access}{access},%
9413   {textaccess}{textaccess},%
9414   {firstaccess}{firstaccess},%
9415   {pluralaccess}{pluralaccess},%
9416   {firstpluralaccess}{firstpluralaccess},%
9417   {symbolaccess}{symbolaccess},%
9418   {symbolpluralaccess}{symbolpluralaccess},%
9419   {descaccess}{descaccess},%
9420   {descpluralaccess}{descpluralaccess},%
9421   {shortaccess}{shortaccess},%
9422   {shortpluralaccess}{shortpluralaccess},%
9423   {longaccess}{longaccess},%
9424   {longpluralaccess}{longpluralaccess}%
9425 }
```

\@gls@noaccess Indicates that no replacement text has been provided.

```
9426 \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):

```
9427 \let\@gls@oldnewglossaryentryprehook\@newglossaryentryprehook
9428 \renewcommand*{\@newglossaryentryprehook}{%
9429   \@gls@oldnewglossaryentryprehook
9430   \def\@glo@access{\@glo@symbol}%
9431 }
```

Initialise the other keys:

```
9431 \def\@glo@textaccess{\@glo@access}%
9432 \def\@glo@firstaccess{\@glo@access}%
9433 \def\@glo@pluralaccess{\@glo@textaccess}%
9434 \def\@glo@firstpluralaccess{\@glo@pluralaccess}%
9435 \def\@glo@symbolaccess{\relax}%
9436 \def\@glo@symbolpluralaccess{\@glo@symbolaccess}%
9437 \def\@glo@descaccess{\relax}%
9438 \def\@glo@descpluralaccess{\@glo@descaccess}%
9439 \def\@glo@shortaccess{\relax}%
9440 \def\@glo@shortpluralaccess{\@glo@shortaccess}%
9441 \def\@glo@longaccess{\relax}%
9442 \def\@glo@longpluralaccess{\@glo@longaccess}%
9443 }
```

Add to the end hook:

```
9444 \let\@gls@oldnewglossaryentryposthook\@newglossaryentryposthook
9445 \renewcommand*{\@newglossaryentryposthook}{%
9446   \@gls@oldnewglossaryentryposthook
```

Store the access information:

```
9447   \expandafter
9448     \protected@xdef\csname glo@\@glo@label @access\endcsname{%
9449       \@glo@access}%
9450   \expandafter
9451     \protected@xdef\csname glo@\@glo@label @textaccess\endcsname{%
9452       \@glo@textaccess}%
9453   \expandafter
9454     \protected@xdef\csname glo@\@glo@label @firstaccess\endcsname{%
9455       \@glo@firstaccess}%
9456   \expandafter
9457     \protected@xdef\csname glo@\@glo@label @pluralaccess\endcsname{%
9458       \@glo@pluralaccess}%
9459   \expandafter
9460     \protected@xdef\csname glo@\@glo@label @firstpluralaccess\endcsname{%
9461       \@glo@firstpluralaccess}%
9462   \expandafter
9463     \protected@xdef\csname glo@\@glo@label @symbolaccess\endcsname{%
9464       \@glo@symbolaccess}%
9465   \expandafter
9466     \protected@xdef\csname glo@\@glo@label @symbolpluralaccess\endcsname{%
9467       \@glo@symbolpluralaccess}%
9468   \expandafter
9469     \protected@xdef\csname glo@\@glo@label @descaccess\endcsname{%
9470       \@glo@descaccess}%
9471   \expandafter
9472     \protected@xdef\csname glo@\@glo@label @descpluralaccess\endcsname{%
9473       \@glo@descpluralaccess}%
9474   \expandafter
9475     \protected@xdef\csname glo@\@glo@label @shortaccess\endcsname{%
9476       \@glo@shortaccess}%
9477   \expandafter
9478     \protected@xdef\csname glo@\@glo@label @shortpluralaccess\endcsname{%
9479       \@glo@shortpluralaccess}%
9480   \expandafter
9481     \protected@xdef\csname glo@\@glo@label @longaccess\endcsname{%
9482       \@glo@longaccess}%
9483   \expandafter
9484     \protected@xdef\csname glo@\@glo@label @longpluralaccess\endcsname{%
9485       \@glo@longpluralaccess}%
9486 }
```

7.2 Accessing Replacement Text

`\glsentryaccess` Get the value of the access key for the entry with the given label:

```
9487 \newcommand*{\glsentryaccess}[1]{%
9488   \@gls@entry@field{#1}{access}%
9489 }
```

`\glsentrytextaccess` Get the value of the textaccess key for the entry with the given label:

```
9490 \newcommand*{\glsentrytextaccess}[1]{%
9491   \@gls@entry@field{#1}{textaccess}%
9492 }
```

`\glsentryfirstaccess` Get the value of the firstaccess key for the entry with the given label:

```
9493 \newcommand*{\glsentryfirstaccess}[1]{%
9494   \@gls@entry@field{#1}{firstaccess}%
9495 }
```

`\glsentrypluralaccess` Get the value of the pluralaccess key for the entry with the given label:

```
9496 \newcommand*{\glsentrypluralaccess}[1]{%
9497   \@gls@entry@field{#1}{pluralaccess}%
9498 }
```

`\glsentryfirstpluralaccess` Get the value of the firstpluralaccess key for the entry with the given label:

```
9499 \newcommand*{\glsentryfirstpluralaccess}[1]{%
9500   \csname glo@#1@firstpluralaccess\endcsname
9501 }
```

`\glsentrysymbolaccess` Get the value of the symbolaccess key for the entry with the given label:

```
9502 \newcommand*{\glsentrysymbolaccess}[1]{%
9503   \@gls@entry@field{#1}{symbolaccess}%
9504 }
```

`\glsentrysymbolpluralaccess` Get the value of the symbolpluralaccess key for the entry with the given label:

```
9505 \newcommand*{\glsentrysymbolpluralaccess}[1]{%
9506   \@gls@entry@field{#1}{symbolpluralaccess}%
9507 }
```

`\glsentrydescaccess` Get the value of the descriptionaccess key for the entry with the given label:

```
9508 \newcommand*{\glsentrydescaccess}[1]{%
9509   \@gls@entry@field{#1}{descaccess}%
9510 }
```

`\glsentrydescpluralaccess` Get the value of the descriptionpluralaccess key for the entry with the given label:

```
9511 \newcommand*{\glsentrydescpluralaccess}[1]{%
9512   \@gls@entry@field{#1}{descaccess}%
9513 }
```

`\glsentryshortaccess` Get the value of the shortaccess key for the entry with the given label:

```
9514 \newcommand*{\glsentryshortaccess}[1]{%
9515   \@gls@entry@field{#1}{shortaccess}%
9516 }
```

`\glsentryshortpluralaccess` Get the value of the shortpluralaccess key for the entry with the given label:

```
9517 \newcommand*{\glsentryshortpluralaccess}[1]{%
9518   \@gls@entry@field{#1}{shortpluralaccess}%
9519 }
```

`\glsentrylongaccess` Get the value of the longaccess key for the entry with the given label:

```
9520 \newcommand*{\glsentrylongaccess}[1]{%
9521   \@gls@entry@field{#1}{longaccess}%
9522 }
```

`\glsentrylongpluralaccess` Get the value of the longpluralaccess key for the entry with the given label:

```
9523 \newcommand*{\glsentrylongpluralaccess}[1]{%
9524   \@gls@entry@field{#1}{longpluralaccess}%
9525 }
```

`\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.)

```
9526 \newcommand*{\glsaccsupp}[2]{%
9527   \BeginAccSupp{ActualText=#1}#2\EndAccSupp{}%
9528 }
```

`\xglsaccsupp` Fully expands replacement text before calling `\glsaccsupp`

```
9529 \newcommand*{\xglsaccsupp}[2]{%
9530   \protected@edef\@gls@replacementtext{#1}%
9531   \expandafter\glsaccsupp\expandafter{\@gls@replacementtext}{#2}%
9532 }
```

`@gls@access@display`

```
9533 \newcommand*{\@gls@access@display}[2]{%
9534   \protected@edef\@glo@access{#2}%
9535   \ifx\@glo@access\@gls@noaccess
9536     #1%
9537   \else
9538     \xglsaccsupp{\@glo@access}{#1}%
9539   \fi
9540 }
```

`\glsnameaccessdisplay` Displays the first argument with the accessibility text for the entry with the label given by the second argument (if set).

```
9541 \DeclareRobustCommand*{\glsnameaccessdisplay}[2]{%
9542   \@gls@access@display{#1}{\glsentryaccess{#2}}%
9543 }
```

lstextaccessdisplay As above but for the textaccess replacement text.

```

9544 \DeclareRobustCommand*\glstextaccessdisplay}[2]{%
9545   \@gls@access@display{#1}{\glsentrytextaccess{#2}}}%
9546 }

```

pluralaccessdisplay As above but for the pluralaccess replacement text.

```

9547 \DeclareRobustCommand*\glspluralaccessdisplay}[2]{%
9548   \@gls@access@display{#1}{\glsentrypluralaccess{#2}}}%
9549 }

```

sfirstaccessdisplay As above but for the firstaccess replacement text.

```

9550 \DeclareRobustCommand*\glsfirstaccessdisplay}[2]{%
9551   \@gls@access@display{#1}{\glsentryfirstaccess{#2}}}%
9552 }

```

pluralaccessdisplay As above but for the firstpluralaccess replacement text.

```

9553 \DeclareRobustCommand*\glsfirstpluralaccessdisplay}[2]{%
9554   \@gls@access@display{#1}{\glsentryfirstpluralaccess{#2}}}%
9555 }

```

symbolaccessdisplay As above but for the symbolaccess replacement text.

```

9556 \DeclareRobustCommand*\glssymbolaccessdisplay}[2]{%
9557   \@gls@access@display{#1}{\glsentrysymbolaccess{#2}}}%
9558 }

```

pluralaccessdisplay As above but for the symbolpluralaccess replacement text.

```

9559 \DeclareRobustCommand*\glssymbolpluralaccessdisplay}[2]{%
9560   \@gls@access@display{#1}{\glsentrysymbolpluralaccess{#2}}}%
9561 }

```

ptionaccessdisplay As above but for the descriptionaccess replacement text.

```

9562 \DeclareRobustCommand*\glsdescriptionaccessdisplay}[2]{%
9563   \@gls@access@display{#1}{\glsentrydescaccess{#2}}}%
9564 }

```

pluralaccessdisplay As above but for the descriptionpluralaccess replacement text.

```

9565 \DeclareRobustCommand*\glsdescriptionpluralaccessdisplay}[2]{%
9566   \@gls@access@display{#1}{\glsentrydescpluralaccess{#2}}}%
9567 }

```

sshortaccessdisplay As above but for the shortaccess replacement text.

```

9568 \DeclareRobustCommand*\glsshortaccessdisplay}[2]{%
9569   \@gls@access@display{#1}{\glsentryshortaccess{#2}}}%
9570 }

```

pluralaccessdisplay As above but for the shortpluralaccess replacement text.

```

9571 \DeclareRobustCommand*\glsshortpluralaccessdisplay}[2]{%
9572   \@gls@access@display{#1}{\glsentryshortpluralaccess{#2}}}%
9573 }

```

`longaccessdisplay` As above but for the `longaccess` replacement text.

```

9574 \DeclareRobustCommand*\glslongaccessdisplay}[2]{%
9575   \@gls@access@display{#1}{\glsentrylongaccess{#2}}}%
9576 }

```

`pluralaccessdisplay` As above but for the `longpluralaccess` replacement text.

```

9577 \DeclareRobustCommand*\glslongpluralaccessdisplay}[2]{%
9578   \@gls@access@display{#1}{\glsentrylongpluralaccess{#2}}}%
9579 }

```

`\glsaccessdisplay` Gets the replacement text corresponding to the named key given by the first argument and calls the appropriate command defined above.

```

9580 \DeclareRobustCommand*\glsaccessdisplay}[3]{%
9581   \@ifundefined{gls#1accessdisplay}%
9582   {%
9583     \PackageError{glossaries-accsupp}{No accessibility support
9584       for key ‘#1’}{}%
9585   }%
9586   {%
9587     \csname gls#1accessdisplay\endcsname{#2}{#3}%
9588   }%
9589 }

```

`ls@default@entryfmt` Redefine the default entry format to use accessibility information

```

9590 \renewcommand*\@@gls@default@entryfmt}[2]{%
9591   \ifdefempty\glscustomtext
9592   {%
9593     \glsifplural
9594     {%

```

Plural form

```

9595       \glscapscase
9596       {%

```

Don't adjust case

```

9597       \ifglsused\glslabel
9598       {%

```

Subsequent use

```

9599         #2{\glspluralaccessdisplay
9600           {\glsentryplural{\glslabel}}{\glslabel}}%
9601         {\glsdescriptionpluralaccessdisplay
9602           {\glsentrydescplural{\glslabel}}{\glslabel}}%
9603         {\glssymbolpluralaccessdisplay
9604           {\glsentrysymbolplural{\glslabel}}{\glslabel}}
9605         {\glsinsert}%
9606       }%
9607     }%

```

First use

```

9608      #1{\glsfirstpluralaccessdisplay
9609          {\glsentryfirstplural{\glslabel}}{\glslabel}}%
9610          {\glsdescriptionpluralaccessdisplay
9611              {\glsentrydescplural{\glslabel}}{\glslabel}}%
9612          {\glssymbolpluralaccessdisplay
9613              {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
9614          {\glsinsert}}%
9615      }%
9616  }%
9617  {%

```

Make first letter upper case

```

9618      \ifglsused\glslabel
9619      {%

```

Subsequent use.

```

9620      #2{\glspluralaccessdisplay
9621          {\Glsentryplural{\glslabel}}{\glslabel}}%
9622          {\glsdescriptionpluralaccessdisplay
9623              {\glsentrydescplural{\glslabel}}{\glslabel}}%
9624          {\glssymbolpluralaccessdisplay
9625              {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
9626          {\glsinsert}}%
9627      }%
9628  {%

```

First use

```

9629      #1{\glsfirstpluralaccessdisplay
9630          {\Glsentryfirstplural{\glslabel}}{\glslabel}}%
9631          {\glsdescriptionpluralaccessdisplay
9632              {\glsentrydescplural{\glslabel}}{\glslabel}}%
9633          {\glssymbolpluralaccessdisplay
9634              {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
9635          {\glsinsert}}%
9636      }%
9637  }%
9638  {%

```

Make all upper case

```

9639      \ifglsused\glslabel
9640      {%

```

Subsequent use

```

9641      \MakeUppercase{%
9642          #2{\glspluralaccessdisplay
9643              {\glsentryplural{\glslabel}}{\glslabel}}%
9644              {\glsdescriptionpluralaccessdisplay
9645                  {\glsentrydescplural{\glslabel}}{\glslabel}}%
9646              {\glssymbolpluralaccessdisplay
9647                  {\glsentrysymbolplural{\glslabel}}{\glslabel}}%

```



```

9648             {\glsinsert}}}%
9649         }%
9650     {%

```

First use

```

9651         \MakeUppercase{%
9652             #1{\glsfirstpluralaccessdisplay
9653                 {\glsentryfirstplural{\glslabel}}{\glslabel}}}%
9654             {\glsdescriptionpluralaccessdisplay
9655                 {\glsentrydescplural{\glslabel}}{\glslabel}}}%
9656             {\glsymbolpluralaccessdisplay
9657                 {\glsentrysymbolplural{\glslabel}}{\glslabel}}}%
9658             {\glsinsert}}}%
9659     }%
9660 }%
9661 }%
9662 {%

```

Singular form

```

9663     \glscapscase
9664     {%

```

Don't adjust case

```

9665     \ifglsused\glslabel
9666     {%

```

Subsequent use

```

9667         #2{\glstextaccessdisplay
9668             {\glsentrytext{\glslabel}}{\glslabel}}}%
9669         {\glsdescriptionaccessdisplay
9670             {\glsentrydesc{\glslabel}}{\glslabel}}}%
9671         {\glsymbolaccessdisplay
9672             {\glsentrysymbol{\glslabel}}{\glslabel}}}%
9673         {\glsinsert}}}%
9674     }%
9675     {%

```

First use

```

9676         #1{\glsfirstaccessdisplay
9677             {\glsentryfirst{\glslabel}}{\glslabel}}}%
9678         {\glsdescriptionaccessdisplay
9679             {\glsentrydesc{\glslabel}}{\glslabel}}}%
9680         {\glsymbolaccessdisplay
9681             {\glsentrysymbol{\glslabel}}{\glslabel}}}%
9682         {\glsinsert}}}%
9683     }%
9684     }%
9685     {%

```

Make first letter upper case

```

9686     \ifglsused\glslabel
9687     {%

```

Subsequent use

```

9688      #2{\glstextaccessdisplay
9689          {\Glsentrytext{\glslabel}}{\glslabel}}%
9690      {\glsdescriptionaccessdisplay
9691          {\Glsentrydesc{\glslabel}}{\glslabel}}%
9692      {\glssymbolaccessdisplay
9693          {\Glsentrysymbol{\glslabel}}{\glslabel}}%
9694      {\glsinsert}}%
9695  }%
9696  {%

```

First use

```

9697      #1{\glsfirstaccessdisplay
9698          {\Glsentryfirst{\glslabel}}{\glslabel}}%
9699      {\glsdescriptionaccessdisplay
9700          {\Glsentrydesc{\glslabel}}{\glslabel}}%
9701      {\glssymbolaccessdisplay
9702          {\Glsentrysymbol{\glslabel}}{\glslabel}}%
9703      {\glsinsert}}%
9704  }%
9705  }%
9706  {%

```

Make all upper case

```

9707      \ifglsused\glslabel
9708      {%

```

Subsequent use

```

9709      \MakeUppercase{%
9710      #2{\glstextaccessdisplay
9711          {\glstextentry{\glslabel}}{\glslabel}}%
9712      {\glsdescriptionaccessdisplay
9713          {\glstextentrydesc{\glslabel}}{\glslabel}}%
9714      {\glssymbolaccessdisplay
9715          {\glstentrysymbol{\glslabel}}{\glslabel}}%
9716      {\glsinsert}}%
9717  }%
9718  {%

```

First use

```

9719      \MakeUppercase{%
9720      #1{\glsfirstaccessdisplay
9721          {\glstentryfirst{\glslabel}}{\glslabel}}%
9722      {\glsdescriptionaccessdisplay
9723          {\glstentrydesc{\glslabel}}{\glslabel}}%
9724      {\glssymbolaccessdisplay
9725          {\glstentrysymbol{\glslabel}}{\glslabel}}%
9726      {\glsinsert}}%
9727  }%
9728  }%
9729  }%

```

```

9730 }%
9731 {%

Custom text provided in \glsdisp
9732 \ifglsused{\glslabel}%
9733 {%

Subsequent use
9734 #2{\glscustomtext}%
9735 {\glsdescriptionaccessdisplay
9736 {\glsentrydesc{\glslabel}}{\glslabel}}%
9737 {\glssymbolaccessdisplay
9738 {\glsentrysymbol{\glslabel}}{\glslabel}}%
9739 {\glsinsert}%
9740 }%
9741 {%

First use
9742 #1{\glscustomtext}%
9743 {\glsdescriptionaccessdisplay
9744 {\glsentrydesc{\glslabel}}{\glslabel}}%
9745 {\glssymbolaccessdisplay
9746 {\glsentrysymbol{\glslabel}}{\glslabel}}%
9747 {\glsinsert}%
9748 }%
9749 }%
9750 }

```

`\glsgenentryfmt` Redefine to use accessibility information.

```

9751 \renewcommand*{\glsgenentryfmt}{%
9752 \ifdefempty\glscustomtext
9753 {%
9754 \glsifplural
9755 {%

Plural form
9756 \glscapscase
9757 {%

Don't adjust case
9758 \ifglsused\glslabel
9759 {%

Subsequent use
9760 \glspluralaccessdisplay
9761 {\glsentryplural{\glslabel}}{\glslabel}%
9762 \glsinsert
9763 }%
9764 {%

First use
9765 \glsfirstpluralaccessdisplay

```

```

9766             {\glsentryfirstplural{\glslabel}}{\glslabel}%
9767         \glsinsert
9768     }%
9769 }%
9770 {%

```

Make first letter upper case

```

9771     \ifglsused\glslabel
9772     {%

```

Subsequent use.

```

9773         \glspluralaccessdisplay
9774         {\Glsentryplural{\glslabel}}{\glslabel}%
9775     \glsinsert
9776 }%
9777 {%

```

First use

```

9778     \glsfirstpluralaccessdisplay
9779     {\Glsentryfirstplural{\glslabel}}{\glslabel}%
9780     \glsinsert
9781 }%
9782 }%
9783 {%

```

Make all upper case

```

9784     \ifglsused\glslabel
9785     {%

```

Subsequent use

```

9786         \glspluralaccessdisplay
9787         {\mfirstucMakeUppercase{\glsentryplural{\glslabel}}}%
9788         {\glslabel}%
9789         \mfirstucMakeUppercase{\glsinsert}%
9790     }%
9791     {%

```

First use

```

9792     \glsfirstpluralaccessdisplay
9793     {\mfirstucMakeUppercase{\glsentryfirstplural{\glslabel}}}%
9794     {\glslabel}%
9795     \mfirstucMakeUppercase{\glsinsert}%
9796 }%
9797 }%
9798 }%
9799 {%

```

Singular form

```

9800     \glscapscase
9801     {%

```

Don't adjust case

```

9802      \ifglsused\glslabel
9803      {%

```

Subsequent use

```

9804      \glstextaccessdisplay{\glsentrytext{\glslabel}}{\glslabel}%
9805      \glsinsert
9806      }%
9807      {%

```

First use

```

9808      \glsfirstaccessdisplay{\glsentryfirst{\glslabel}}{\glslabel}%
9809      \glsinsert
9810      }%
9811      }%
9812      {%

```

Make first letter upper case

```

9813      \ifglsused\glslabel
9814      {%

```

Subsequent use

```

9815      \glstextaccessdisplay{\Glsentrytext{\glslabel}}{\glslabel}%
9816      \glsinsert
9817      }%
9818      {%

```

First use

```

9819      \glsfirstaccessdisplay{\Glsentryfirst{\glslabel}}{\glslabel}%
9820      \glsinsert
9821      }%
9822      }%
9823      {%

```

Make all upper case

```

9824      \ifglsused\glslabel
9825      {%

```

Subsequent use

```

9826      \glstextaccessdisplay
9827      {\mfirstucMakeUppercase{\glsentrytext{\glslabel}}}{\glslabel}%
9828      \mfirstucMakeUppercase{\glsinsert}%
9829      }%
9830      {%

```

First use

```

9831      \glsfirstaccessdisplay
9832      {\mfirstucMakeUppercase{\glsentryfirst{\glslabel}}}{\glslabel}%
9833      \mfirstucMakeUppercase{\glsinsert}%
9834      }%
9835      }%
9836      }%
9837      }%
9838      {%

```

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.

```
9839 \glscustomtext\glsinsert
9840 }%
9841 }
```

`\glsngenacfmt` Redefine to include accessibility information.

```
9842 \renewcommand*{\glsngenacfmt}{%
9843 \ifdefempty\glscustomtext
9844 {%
9845 \ifglused\glslabel
9846 {%
```

Subsequent use:

```
9847 \glsifplural
9848 {%
```

Subsequent plural form:

```
9849 \glscapscase
9850 {%
```

Subsequent plural form, don't adjust case:

```
9851 \acronymfont
9852 {\glsshortpluralaccessdisplay
9853 {\glstryshortpl{\glslabel}}{\glslabel}}%
9854 \glsinsert
9855 }%
9856 {%
```

Subsequent plural form, make first letter upper case:

```
9857 \acronymfont
9858 {\glsshortpluralaccessdisplay
9859 {\Glsentryshortpl{\glslabel}}{\glslabel}}%
9860 \glsinsert
9861 }%
9862 {%
```

Subsequent plural form, all caps:

```
9863 \mfirstucMakeUppercase
9864 {\acronymfont
9865 {\glsshortpluralaccessdisplay
9866 {\glstryshortpl{\glslabel}}{\glslabel}}%
9867 \glsinsert}%
9868 }%
9869 }%
9870 {%
```

Subsequent singular form

```
9871 \glscapscase
9872 {%
```

Subsequent singular form, don't adjust case:

```

9873      \acronymfont
9874      {\glsshortaccessdisplay{\glsentryshort{\glslabel}}{\glslabel}}%
9875      \glsinsert
9876      }%
9877      {%

```

Subsequent singular form, make first letter upper case:

```

9878      \acronymfont
9879      {\glsshortaccessdisplay{\Glsentryshort{\glslabel}}{\glslabel}}%
9880      \glsinsert
9881      }%
9882      {%

```

Subsequent singular form, all caps:

```

9883      \mfirstucMakeUppercase
9884      {\acronymfont{%
9885      \glsshortaccessdisplay{\glsentryshort{\glslabel}}{\glslabel}}%
9886      \glsinsert}%
9887      }%
9888      }%
9889      }%
9890      {%

```

First use:

```

9891      \glsifplural
9892      {%

```

First use plural form:

```

9893      \glscapscase
9894      {%

```

First use plural form, don't adjust case:

```

9895      \genplacrfullformat{\glslabel}{\glsinsert}%
9896      }%
9897      {%

```

First use plural form, make first letter upper case:

```

9898      \Genplacrfullformat{\glslabel}{\glsinsert}%
9899      }%
9900      {%

```

First use plural form, all caps:

```

9901      \mfirstucMakeUppercase
9902      {\genplacrfullformat{\glslabel}{\glsinsert}}%
9903      }%
9904      }%
9905      {%

```

First use singular form

```

9906      \glscapscase
9907      {%

```

First use singular form, don't adjust case:

```
9908      \genacrfullformat{\glslabel}{\glsinsert}%
9909      }%
9910      {%
```

First use singular form, make first letter upper case:

```
9911      \Genacrfullformat{\glslabel}{\glsinsert}%
9912      }%
9913      {%
```

First use singular form, all caps:

```
9914      \mfirstucMakeUppercase
9915      {\genacrfullformat{\glslabel}{\glsinsert}}%
9916      }%
9917      }%
9918      }%
9919      }%
9920      {%
```

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.

```
9921      \glscustomtext
9922      }%
9923 }
```

`\genacrfullformat` Redefine to include accessibility information.

```
9924 \renewcommand*{\genacrfullformat}[2]{%
9925   \glslongaccessdisplay{\glsentrylong{#1}}{#1}#2\space
9926   (\glsshortaccessdisplay{\protect\firstacronymfont{\glsentryshort{#1}}}{#1})%
9927 }
```

`\Genacrfullformat` Redefine to include accessibility information.

```
9928 \renewcommand*{\Genacrfullformat}[2]{%
9929   \glslongaccessdisplay{\Glsentrylong{#1}}{#1}#2\space
9930   (\glsshortaccessdisplay{\protect\firstacronymfont{\Glsentryshort{#1}}}{#1})%
9931 }
```

`\genplacrfullformat` Redefine to include accessibility information.

```
9932 \renewcommand*{\genplacrfullformat}[2]{%
9933   \glslongpluralaccessdisplay{\glsentrylongpl{#1}}{#1}#2\space
9934   (\glsshortpluralaccessdisplay
9935     {\protect\firstacronymfont{\glsentryshortpl{#1}}}{#1})%
9936 }
```

`\Genplacrfullformat` Redefine to include accessibility information.

```
9937 \renewcommand*{\Genplacrfullformat}[2]{%
9938   \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}#2\space
9939   (\glsshortpluralaccessdisplay
9940     {\protect\firstacronymfont{\Glsentryshortpl{#1}}}{#1})%
9941 }
```


\@acrshort

```
9942 \def\@acrshort#1#2[#3]{%
9943   \glsdoifexists{#2}%
9944   {%
9945     \let\do@gl@link@checkfirsthyper\relax

9946     \let\glsifplural\@secondoftwo
9947     \let\glscapscase\@firstofthree
9948     \let\glsinsert\@empty
9949     \def\glscustomtext{%
9950       \acronymfont{\glsshortaccessdisplay{\glsentryshort{#2}}{#2}}#3%
9951     }%

    Call \@gl@link
9952     \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
9953   }%
9954 }
```

\@Acrshort

```
9955 \def\@Acrshort#1#2[#3]{%
9956   \glsdoifexists{#2}%
9957   {%
9958     \let\do@gl@link@checkfirsthyper\relax

9959     \let\glsifplural\@secondoftwo
9960     \let\glscapscase\@secondofthree
9961     \let\glsinsert\@empty
9962     \def\glscustomtext{%
9963       \acronymfont{\glsshortaccessdisplay{\Glsentryshort{#2}}{#2}}#3%
9964     }%

    Call \@gl@link
9965     \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
9966   }%
9967 }
```

\@ACRshort

```
9968 \def\@ACRshort#1#2[#3]{%
9969   \glsdoifexists{#2}%
9970   {%
9971     \let\do@gl@link@checkfirsthyper\relax

9972     \let\glsifplural\@secondoftwo
9973     \let\glscapscase\@thirdofthree
9974     \let\glsinsert\@empty
9975     \def\glscustomtext{%
9976       \acronymfont{\glsshortaccessdisplay
9977         {\MakeUppercase{\glsentryshort{#2}}}{#2}}#3%
9978     }%
9979 }
```

```

Call \@gls@link
9979   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
9980   }%
9981 }

```

\@acrlong

```

9982 \def\@acrlong#1#2[#3]{%
9983   \glsdoifexists{#2}%
9984   {%
9985     \let\do@gls@link@checkfirsthyper\relax

9986     \let\glsifplural\@secondoftwo
9987     \let\glscapscase\@firstofthree
9988     \let\glsinsert\@empty
9989     \def\glscustomtext{%
9990       \acronymfont{\glslongaccessdisplay{\glsentrylong{#2}}{#2}}#3%
9991     }%

```

```

Call \@gls@link
9992   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
9993   }%
9994 }

```

\@Acrlong

```

9995 \def\@Acrlong#1#2[#3]{%
9996   \glsdoifexists{#2}%
9997   {%
9998     \let\do@gls@link@checkfirsthyper\relax

9999     \let\glsifplural\@secondoftwo
10000    \let\glscapscase\@firstofthree
10001    \let\glsinsert\@empty
10002    \def\glscustomtext{%
10003      \acronymfont{\glslongaccessdisplay{\Glsentrylong{#2}}{#2}}#3%
10004    }%

```

```

Call \@gls@link
10005   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10006   }%
10007 }

```

\@ACRlong

```

10008 \def\@ACRlong#1#2[#3]{%
10009   \glsdoifexists{#2}%
10010   {%
10011     \let\do@gls@link@checkfirsthyper\relax

10012     \let\glsifplural\@secondoftwo
10013     \let\glscapscase\@firstofthree
10014     \let\glsinsert\@empty

```

```

10015 \def\glscustomtext{%
10016 \acronymfont{\glslongaccessdisplay{%
10017 \MakeUppercase{\glsentrylong{#2}}}{#2}#3}%
10018 }%

Call \@gls@link
10019 \@gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
10020 }%
10021 }

```

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.

```

10022 \renewcommand*{\glossentryname}[1]{%
10023 \glsdoifexists{#1}%
10024 {%
10025 \glsnamefont{\glsnameaccessdisplay{\glsentryname{#1}}{#1}}%
10026 }%
10027 }

10028 \renewcommand*{\glossentryname}[1]{%
10029 \glsdoifexists{#1}%
10030 {%
10031 \glsnamefont{\glsnameaccessdisplay{\Glsentryname{#1}}{#1}}%
10032 }%
10033 }

10034 \renewcommand*{\glossentrydesc}[1]{%
10035 \glsdoifexists{#1}%
10036 {%
10037 \glsdescriptionaccessdisplay{\glsentrydesc{#1}}{#1}%
10038 }%
10039 }

10040 \renewcommand*{\Glossentrydesc}[1]{%
10041 \glsdoifexists{#1}%
10042 {%
10043 \glsdescriptionaccessdisplay{\Glsentrydesc{#1}}{#1}%
10044 }%
10045 }

10046 \renewcommand*{\glossentrysymbol}[1]{%
10047 \glsdoifexists{#1}%
10048 {%

```

```

10049 \glssymbolaccessdisplay{\glentrysymbol{#1}}{#1}%
10050 }%
10051 }

10052 \renewcommand*{\Glossentrysymbol}[1]{%
10053 \glsoifexists{#1}%
10054 {%
10055 \glssymbolaccessdisplay{\Glsentrysymbol{#1}}{#1}%
10056 }%
10057 }

```

pglossaryentryfield

```

10058 \newcommand*{\accsuppglossaryentryfield}[5]{%
10059 \glossaryentryfield{#1}%
10060 {\glslnameaccessdisplay{#2}{#1}}%
10061 {\glslsdescriptionaccessdisplay{#3}{#1}}%
10062 {\glssymbolaccessdisplay{#4}{#1}}{#5}%
10063 }

```

glossarysubentryfield

```

10064 \newcommand*{\accsuppglossarysubentryfield}[6]{%
10065 \glossarysubentryfield{#1}{#2}%
10066 {\glslnameaccessdisplay{#3}{#2}}%
10067 {\glslsdescriptionaccessdisplay{#4}{#2}}%
10068 {\glssymbolaccessdisplay{#5}{#2}}{#6}%
10069 }

```

7.4 Acronyms

Redefine acronym styles provided by glossaries:

long-short *<long>* (*<short>*) acronym style.

```

10070 \renewacronymstyle{long-short}%
10071 {%

```

Check for long form in case this is a mixed glossary.

```

10072 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
10073 }%
10074 {%
10075 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
10076 \renewcommand*{\genacrfullformat}[2]{%
10077 \glslongaccessdisplay{\glslentrylong{##1}}{##1}##2\space
10078 (\glsshortaccessdisplay
10079 {\protect\firstacronymfont{\glslentryshort{##1}}}{##1})%
10080 }%
10081 \renewcommand*{\Genacrfullformat}[2]{%
10082 \glslongaccessdisplay{\Glsentrylong{##1}}{##1}##2\space
10083 (\glsshortaccessdisplay
10084 {\protect\firstacronymfont{\Glsentryshort{##1}}}{##1})%
10085 }%

```

```

10086 \renewcommand*{\genplacrfullformat}[2]{%
10087   \glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}##2\space
10088   (\glsshortpluralaccessdisplay
10089     {\protect\firstacronymfont{\glsentryshortpl{##1}}}{##1})%
10090 }%
10091 \renewcommand*{\Genplacrfullformat}[2]{%
10092   \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}##2\space
10093   (\glsshortpluralaccessdisplay
10094     {\protect\firstacronymfont{\Glsentryshortpl{##1}}}{##1})%
10095 }%
10096 \renewcommand*{\acronymentry}[1]{%
10097   \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}
10098 \renewcommand*{\acronymsort}[2]{##1}%
10099 \renewcommand*{\acronymfont}[1]{##1}%
10100 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
10101 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
10102 }

```

short-long *<short>* (*<long>*) acronym style.

```

10103 \renewacronymstyle{short-long}%
10104 {%
10105   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
10106 }%
10107 {%
10108   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
10109   \renewcommand*{\genacrfullformat}[2]{%
10110     \glsshortaccessdisplay
10111     {\protect\firstacronymfont{\glsentryshort{##1}}}{##1}##2\space
10112     (\glslongaccessdisplay{\glsentrylong{##1}}{##1})%
10113   }%
10114   \renewcommand*{\Genacrfullformat}[2]{%
10115     \glsshortaccessdisplay
10116     {\protect\firstacronymfont{\Glsentryshort{##1}}}{##1}##2\space
10117     (\glslongaccessdisplay{\Glsentrylong{##1}}{##1})%
10118   }%
10119   \renewcommand*{\genplacrfullformat}[2]{%
10120     \glsshortpluralaccessdisplay
10121     {\protect\firstacronymfont{\glsentryshortpl{##1}}}{##1}##2\space
10122     (\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1})%
10123   }%
10124   \renewcommand*{\Genplacrfullformat}[2]{%
10125     \glsshortpluralaccessdisplay
10126     {\protect\firstacronymfont{\Glsentryshortpl{##1}}}{##1}##2\space
10127     (\glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1})%
10128   }%
10129   \renewcommand*{\acronymentry}[1]{%
10130     \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}%
10131   }

```

```

10132 \renewcommand*\acronymsort}[2]{##1}%
10133 \renewcommand*\acronymfont}[1]{##1}%
10134 \renewcommand*\firstacronymfont}[1]{\acronymfont{##1}}%
10135 \renewcommand*\acrpluralsuffix{\glspluralsuffix}%
10136 }

```

long-short-desc *<long>* (*<short>*) acronym style that has an accompanying description (which the user needs to supply).

```

10137 \renewacronymstyle{long-short-desc}%
10138 {%
10139   \GlsUseAcrEntryDisplayStyle{long-short}%
10140 }%
10141 {%
10142   \GlsUseAcrStyleDefs{long-short}%
10143   \renewcommand*\GenericAcronymFields{}%
10144   \renewcommand*\acronymsort}[2]{##2}%
10145   \renewcommand*\acronymentry}[1]{%
10146     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10147     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10148 }

```

long-sc-short-desc *<long>* (*\textsc{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

10149 \renewacronymstyle{long-sc-short-desc}%
10150 {%
10151   \GlsUseAcrEntryDisplayStyle{long-sc-short}%
10152 }%
10153 {%
10154   \GlsUseAcrStyleDefs{long-sc-short}%
10155   \renewcommand*\GenericAcronymFields{}%
10156   \renewcommand*\acronymsort}[2]{##2}%
10157   \renewcommand*\acronymentry}[1]{%
10158     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10159     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10160 }

```

long-sm-short-desc *<long>* (*\textsmaller{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

10161 \renewacronymstyle{long-sm-short-desc}%
10162 {%
10163   \GlsUseAcrEntryDisplayStyle{long-sm-short}%
10164 }%
10165 {%
10166   \GlsUseAcrStyleDefs{long-sm-short}%
10167   \renewcommand*\GenericAcronymFields{}%
10168   \renewcommand*\acronymsort}[2]{##2}%
10169   \renewcommand*\acronymentry}[1]{%
10170     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10171     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%

```

10172 }

short-long-desc *<short>* (*{<long>}*) acronym style that has an accompanying description (which the user needs to supply).

```
10173 \renewacronymstyle{short-long-desc}%
10174 {%
10175   \GlsUseAcrEntryDispStyle{short-long}%
10176 }%
10177 {%
10178   \GlsUseAcrStyleDefs{short-long}%
10179   \renewcommand*{\GenericAcronymFields}{}%
10180   \renewcommand*{\acronymsort}[2]{##2}%
10181   \renewcommand*{\acronymentry}[1]{%
10182     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10183     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10184 }
```

sc-short-long-desc *<long>* (*\textsc{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```
10185 \renewacronymstyle{sc-short-long-desc}%
10186 {%
10187   \GlsUseAcrEntryDispStyle{sc-short-long}%
10188 }%
10189 {%
10190   \GlsUseAcrStyleDefs{sc-short-long}%
10191   \renewcommand*{\GenericAcronymFields}{}%
10192   \renewcommand*{\acronymsort}[2]{##2}%
10193   \renewcommand*{\acronymentry}[1]{%
10194     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10195     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10196 }
```

sm-short-long-desc *<long>* (*\textsmaller{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```
10197 \renewacronymstyle{sm-short-long-desc}%
10198 {%
10199   \GlsUseAcrEntryDispStyle{sm-short-long}%
10200 }%
10201 {%
10202   \GlsUseAcrStyleDefs{sm-short-long}%
10203   \renewcommand*{\GenericAcronymFields}{}%
10204   \renewcommand*{\acronymsort}[2]{##2}%
10205   \renewcommand*{\acronymentry}[1]{%
10206     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10207     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10208 }
```

dua *<long>* only acronym style.

10209 \renewacronymstyle{dua}%

10210 {%

Check for long form in case this is a mixed glossary.

10211 \ifdefempty\glscustomtext

10212 {%

10213 \ifglshaslong{\glslabel}%

10214 {%

10215 \glsifplural

10216 {%

Plural form:

10217 \glscapscase

10218 {%

Plural form, don't adjust case:

10219 \glslongpluralaccessdisplay{\glentrylongpl{\glslabel}}{\glslabel}%

10220 \glsinsert

10221 }%

10222 {%

Plural form, make first letter upper case:

10223 \glslongpluralaccessdisplay{\Glsentrylongpl{\glslabel}}{\glslabel}%

10224 \glsinsert

10225 }%

10226 {%

Plural form, all caps:

10227 \glslongpluralaccessdisplay

10228 {\mfirstucMakeUppercase{\glentrylongpl{\glslabel}}}{\glslabel}%

10229 \mfirstucMakeUppercase{\glsinsert}%

10230 }%

10231 }%

10232 {%

Singular form

10233 \glscapscase

10234 {%

Singular form, don't adjust case:

10235 \glslongaccessdisplay{\glentrylong{\glslabel}}{\glslabel}\glsinsert

10236 }%

10237 {%

Subsequent singular form, make first letter upper case:

10238 \glslongaccessdisplay{\Glsentrylong{\glslabel}}{\glslabel}\glsinsert

10239 }%

10240 {%

Subsequent singular form, all caps:

10241 \glslongaccessdisplay

10242 {\mfirstucMakeUppercase

10243 {\glentrylong{\glslabel}\glsinsert}}{\glslabel}%


```

10244         \mfirstucMakeUppercase{\glsinsert}%
10245     }%
10246 }%
10247 }%
10248 {%

    Not an acronym:

10249     \glsgenentryfmt
10250 }%
10251 }%
10252 {\glscustomtext\glsinsert}%
10253 }%
10254 {%
10255 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
10256 \renewcommand*{\acrfullfmt}[3]{%
10257     \glslink[##1]{##2}{%
10258         \glslongaccessdisplay{\glsentrylong{##2}}{##2}##3\space
10259         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}}}%
10260 \renewcommand*{\Acrfullfmt}[3]{%
10261     \glslink[##1]{##2}{%
10262         \glslongaccessdisplay{\Glsentrylong{##2}}{##2}##3\space
10263         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}}}%
10264 \renewcommand*{\ACRfullfmt}[3]{%
10265     \glslink[##1]{##2}{%
10266         \glslongaccessdisplay
10267         {\mfirstucMakeUppercase{\glsentrylong{##2}}{##2}##3\space
10268         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}}}}}%
10269 \renewcommand*{\acrfullplfmt}[3]{%
10270     \glslink[##1]{##2}{%
10271         \glslongpluralaccessdisplay
10272         {\glsentrylongpl{##2}}{##2}##3\space
10273         (\glsshortpluralaccessdisplay
10274         {\acronymfont{\glsentryshortpl{##2}}}{##2}}}%
10275 \renewcommand*{\Acrfullplfmt}[3]{%
10276     \glslink[##1]{##2}{%
10277         \glslongpluralaccessdisplay
10278         {\Glsentrylongpl{##2}}{##2}##3\space
10279         (\glsshortpluralaccessdisplay
10280         {\acronymfont{\glsentryshortpl{##2}}}{##2}}}%
10281 \renewcommand*{\ACRfullplfmt}[3]{%
10282     \glslink[##1]{##2}{%
10283         \glslongpluralaccessdisplay
10284         {\mfirstucMakeUppercase{\glsentrylongpl{##2}}{##2}##3\space
10285         (\glsshortpluralaccessdisplay
10286         {\acronymfont{\glsentryshortpl{##2}}}{##2}}}}}%
10287 \renewcommand*{\glsentryfull}[1]{%
10288     \glslongaccessdisplay{\glsentrylong{##1}}\space
10289     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})%
10290 }%
10291 \renewcommand*{\Glsentryfull}[1]{%

```

```

10292 \glslongaccessdisplay{\Glsentrylong{##1}}{##1}\space
10293 (\glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}}{##1})%
10294 }%
10295 \renewcommand*{\glsentryfullpl}[1]{%
10296 \glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}\space
10297 (\glsshortpluralaccessdisplay{\acronymfont{\glentryshortpl{##1}}}{##1})%
10298 }%
10299 \renewcommand*{\Glsentryfullpl}[1]{%
10300 \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}\space
10301 (\glsshortpluralaccessdisplay{\acronymfont{\glentryshortpl{##1}}}{##1})%
10302 }%
10303 \renewcommand*{\acronymentry}[1]{%
10304 \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}}{##1}%
10305 \renewcommand*{\acronymsort}[2]{##1}%
10306 \renewcommand*{\acronymfont}[1]{##1}%
10307 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
10308 }

```

dua-desc *<long>* only acronym style with user-supplied description.

```

10309 \renewacronymstyle{dua-desc}%
10310 {%
10311 \GlsUseAcrEntryDispStyle{dua}%
10312 }%
10313 {%
10314 \GlsUseAcrStyleDefs{dua}%
10315 \renewcommand*{\GenericAcronymFields}{}%
10316 \renewcommand*{\acronymentry}[1]{%
10317 \glslongaccessdisplay{\acronymfont{\glentrylong{##1}}}{##1}%
10318 \renewcommand*{\acronymsort}[2]{##2}%
10319 }%

```

footnote *<short>*\footnote{*<long>*} acronym style.

```

10320 \renewacronymstyle{footnote}%
10321 {%

```

Check for long form in case this is a mixed glossary.

```

10322 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
10323 }%
10324 {%
10325 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%

```

Need to ensure hyperlinks are switched off on first use:

```

10326 \glshyperfirstfalse
10327 \renewcommand*{\genacrfullformat}[2]{%
10328 \glsshortaccessdisplay
10329 {\protect\firstacronymfont{\glentryshort{##1}}}{##1}##2%
10330 \protect\footnote{\glslongaccessdisplay{\glentrylong{##1}}{##1}}%
10331 }%
10332 \renewcommand*{\Genacrfullformat}[2]{%
10333 \glsshortaccessdisplay

```

```

10334     {\firstacronymfont{\Glsentryshort{##1}}}{##1}##2%
10335     \protect\footnote{\glslongaccessdisplay{\glssentrylong{##1}}}{##1}}%
10336 }%
10337 \renewcommand*{\genplacrfullformat}[2]{%
10338     \glsshortpluralaccessdisplay
10339     {\protect\firstacronymfont{\glssentryshortpl{##1}}}{##1}##2%
10340     \protect\footnote{\glslongpluralaccessdisplay{\glssentrylongpl{##1}}}{##1}}%
10341 }%
10342 \renewcommand*{\Genplacrfullformat}[2]{%
10343     \glsshortpluralaccessdisplay
10344     {\protect\firstacronymfont{\Glsentryshortpl{##1}}}{##1}##2%
10345     \protect\footnote{\glslongpluralaccessdisplay{\glssentrylongpl{##1}}}{##1}}%
10346 }%
10347 \renewcommand*{\acronymentry}[1]{%
10348     \glsshortaccessdisplay{\acronymfont{\glssentryshort{##1}}}{##1}}%
10349 \renewcommand*{\acronymsort}[2]{##1}%
10350 \renewcommand*{\acronymfont}[1]{##1}%
10351 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%

```

Don't use footnotes for \acrfull:

```

10352 \renewcommand*{\acrfullfmt}[3]{%
10353     \glslink[##1]{##2}{%
10354         \glsshortaccessdisplay{\acronymfont{\glssentryshort{##2}}}{##2}##3\space
10355         (\glslongaccessdisplay{\glssentrylong{##2}}{##2})}%
10356 \renewcommand*{\Acrfullfmt}[3]{%
10357     \glslink[##1]{##2}{%
10358         \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##2}}}{##2}##3\space
10359         (\glslongaccessdisplay{\glssentrylong{##2}}{##2})}%
10360 \renewcommand*{\ACRfullfmt}[3]{%
10361     \glslink[##1]{##2}{%
10362         \glsshortaccessdisplay
10363         {\mfirstucMakeUppercase
10364         {\acronymfont{\glssentryshort{##2}}}{##2}##3\space
10365         (\glslongaccessdisplay{\glssentrylong{##2}}{##2})}%
10366 \renewcommand*{\acrfullplfmt}[3]{%
10367     \glslink[##1]{##2}{%
10368         \glsshortpluralaccessdisplay
10369         {\acronymfont{\glssentryshortpl{##2}}}{##2}##3\space
10370         (\glslongpluralaccessdisplay{\glssentrylongpl{##2}}{##2})}%
10371 \renewcommand*{\Acrfullplfmt}[3]{%
10372     \glslink[##1]{##2}{%
10373         \glsshortpluralaccessdisplay
10374         {\acronymfont{\Glsentryshortpl{##2}}}{##2}##3\space
10375         (\glslongpluralaccessdisplay{\glssentrylongpl{##2}}{##2})}%
10376 \renewcommand*{\ACRfullplfmt}[3]{%
10377     \glslink[##1]{##2}{%
10378         \glsshortpluralaccessdisplay
10379         {\mfirstucMakeUppercase
10380         {\acronymfont{\glssentryshortpl{##2}}}{##2}##3\space
10381         (\glslongpluralaccessdisplay{\glssentrylongpl{##2}}{##2})}%

```

Similarly for \glsentryfull etc:

```

10382 \renewcommand*{\glsentryfull}[1]{%
10383   \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}\space
10384   (\glslongaccessdisplay{\glsentrylong{##1}}{##1})}%
10385 \renewcommand*{\Glsentryfull}[1]{%
10386   \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##1}}}{##1}\space
10387   (\glslongaccessdisplay{\glsentrylong{##1}}{##1})}%
10388 \renewcommand*{\glsentryfullpl}[1]{%
10389   \glsshortpluralaccessdisplay
10390   {\acronymfont{\glsentryshortpl{##1}}}{##1}\space
10391   (\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1})}%
10392 \renewcommand*{\Glsentryfullpl}[1]{%
10393   \glsshortpluralaccessdisplay
10394   {\acronymfont{\Glsentryshortpl{##1}}}{##1}\space
10395   (\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1})}%
10396 }

```

footnote-sc \textsc{<short>}\footnote{<long>} acronym style.

```

10397 \renewacronymstyle{footnote-sc}%
10398 {%
10399   \GlsUseAcrEntryDispStyle{footnote}%
10400 }%
10401 {%
10402   \GlsUseAcrStyleDefs{footnote}%
10403   \renewcommand{\acronymentry}[1]{%
10404     \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}
10405   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
10406   \renewcommand*{\acrpluralsuffix}{\glstextup{\glspluralsuffix}}%
10407 }%

```

footnote-sm \textsmaller{<short>}\footnote{<long>} acronym style.

```

10408 \renewacronymstyle{footnote-sm}%
10409 {%
10410   \GlsUseAcrEntryDispStyle{footnote}%
10411 }%
10412 {%
10413   \GlsUseAcrStyleDefs{footnote}%
10414   \renewcommand{\acronymentry}[1]{%
10415     \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}
10416   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
10417   \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
10418 }%

```

footnote-desc <short>\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

10419 \renewacronymstyle{footnote-desc}%
10420 {%
10421   \GlsUseAcrEntryDispStyle{footnote}%
10422 }%

```

```

10423 {%
10424   \GlsUseAcrStyleDefs{footnote}%
10425   \renewcommand*{\GenericAcronymFields}{}%
10426   \renewcommand*{\acronymsort}[2]{##2}%
10427   \renewcommand*{\acronymentry}[1]{%
10428     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10429     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10430 }

```

footnote-sc-desc \textsc{\<short>}\footnote{\<long>} acronym style that has an accompanying description (which the user needs to supply).

```

10431 \renewacronymstyle{footnote-sc-desc}%
10432 {%
10433   \GlsUseAcrEntryDispStyle{footnote-sc}%
10434 }%
10435 {%
10436   \GlsUseAcrStyleDefs{footnote-sc}%
10437   \renewcommand*{\GenericAcronymFields}{}%
10438   \renewcommand*{\acronymsort}[2]{##2}%
10439   \renewcommand*{\acronymentry}[1]{%
10440     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10441     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10442 }

```

footnote-sm-desc \textsmaller{\<short>}\footnote{\<long>} acronym style that has an accompanying description (which the user needs to supply).

```

10443 \renewacronymstyle{footnote-sm-desc}%
10444 {%
10445   \GlsUseAcrEntryDispStyle{footnote-sm}%
10446 }%
10447 {%
10448   \GlsUseAcrStyleDefs{footnote-sm}%
10449   \renewcommand*{\GenericAcronymFields}{}%
10450   \renewcommand*{\acronymsort}[2]{##2}%
10451   \renewcommand*{\acronymentry}[1]{%
10452     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
10453     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
10454 }

```

Use \newacronymhook to modify the key list to set the access text to the long version by default.

```

10455 \renewcommand*{\newacronymhook}{%
10456   \edef\@gls@keylist{shortaccess=\the\glslongtok,%
10457     \the\glskeylisttok}%
10458   \expandafter\glskeylisttok\expandafter{\@gls@keylist}%
10459 }

```

defaultNewAcronymDef Modify default style to use access text:

```

10460 \renewcommand*{\DefaultNewAcronymDef}{%
10461   \edef\@do@newglossaryentry{%
10462     \noexpand\newglossaryentry{\the\glslabeltok}%
10463     {%
10464       type=\acronymtype,%
10465       name={\the\glsshorttok},%
10466       description={\the\glslongtok},%
10467       descriptionaccess=\relax,
10468       text={\the\glsshorttok},%
10469       access={\noexpand\@glo@textaccess},%
10470       sort={\the\glsshorttok},%
10471       short={\the\glsshorttok},%
10472       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10473       shortaccess={\the\glslongtok},%
10474       long={\the\glslongtok},%
10475       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10476       descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10477       first={\noexpand\glslongaccessdisplay
10478         {\the\glslongtok}{\the\glslabeltok}\space
10479         (\noexpand\glsshortaccessdisplay
10480         {\the\glsshorttok}{\the\glslabeltok})},%
10481       plural={\the\glsshorttok\acrpluralsuffix},%
10482       firstplural={\noexpand\glslongpluralaccessdisplay
10483         {\noexpand\@glo@longpl}{\the\glslabeltok}\space
10484         (\noexpand\glsshortpluralaccessdisplay
10485         {\noexpand\@glo@shortpl}{\the\glslabeltok})},%
10486       firstaccess=\relax,
10487       firstpluralaccess=\relax,
10488       textaccess={\noexpand\@glo@shortaccess},%
10489       \the\glskeylisttok
10490     }%
10491   }%
10492   \let\@org@gls@assign@firstpl\gls@assign@firstpl
10493   \let\@org@gls@assign@plural\gls@assign@plural
10494   \let\@org@gls@assign@descplural\gls@assign@descplural
10495   \def\gls@assign@firstpl##1##2{%
10496     \@@gls@expand@field{##1}{firstpl}{##2}%
10497   }%
10498   \def\gls@assign@plural##1##2{%
10499     \@@gls@expand@field{##1}{plural}{##2}%
10500   }%
10501   \def\gls@assign@descplural##1##2{%
10502     \@@gls@expand@field{##1}{descplural}{##2}%
10503   }%
10504   \@do@newglossaryentry
10505   \let\gls@assign@firstpl\@org@gls@assign@firstpl
10506   \let\gls@assign@plural\@org@gls@assign@plural
10507   \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
10508 }

```

otnoteNewAcronymDef

```

10509 \renewcommand*{\DescriptionFootnoteNewAcronymDef}{%
10510   \edef\@do@newglossaryentry{%
10511     \noexpand\newglossaryentry{\the\glslabeltok}%
10512     {%
10513       type=\acronymtype,%
10514       name={\noexpand\acronymfont{\the\glsshorttok}},%
10515       sort={\the\glsshorttok},%
10516       text={\the\glsshorttok},%
10517       short={\the\glsshorttok},%
10518       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10519       shortaccess={\the\glslongtok},%
10520       long={\the\glslongtok},%
10521       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10522       access={\noexpand\@glo@textaccess},%
10523       plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10524       symbol={\the\glslongtok},%
10525       symbolplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10526       firstpluralaccess=\relax,
10527       textaccess={\noexpand\@glo@shortaccess},%
10528       \the\glskeylisttok
10529     }%
10530   }%
10531   \let\@org@gls@assign@firstpl\gls@assign@firstpl
10532   \let\@org@gls@assign@plural\gls@assign@plural
10533   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
10534   \def\gls@assign@firstpl##1##2{%
10535     \@@gls@expand@field{##1}{firstpl}{##2}%
10536   }%
10537   \def\gls@assign@plural##1##2{%
10538     \@@gls@expand@field{##1}{plural}{##2}%
10539   }%
10540   \def\gls@assign@symbolplural##1##2{%
10541     \@@gls@expand@field{##1}{symbolplural}{##2}%
10542   }%
10543   \@do@newglossaryentry
10544   \let\gls@assign@plural\@org@gls@assign@plural
10545   \let\gls@assign@firstpl\@org@gls@assign@firstpl
10546   \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
10547 }

```

ipitionNewAcronymDef

```

10548 \renewcommand*{\DescriptionNewAcronymDef}{%
10549   \edef\@do@newglossaryentry{%
10550     \noexpand\newglossaryentry{\the\glslabeltok}%
10551     {%
10552       type=\acronymtype,%
10553       name={\noexpand
10554         \acrnameformat{\the\glsshorttok}{\the\glslongtok}},%

```

```

10555     access={\noexpand\@glo@textaccess},%
10556     sort={\the\glsshorttok},%
10557     short={\the\glsshorttok},%
10558     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10559     shortaccess={\the\glslongtok},%
10560     long={\the\glslongtok},%
10561     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10562     first={\the\glslongtok},%
10563     firstaccess=\relax,
10564     firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10565     text={\the\glsshorttok},%
10566     textaccess={\the\glslongtok},%
10567     plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10568     symbol={\noexpand\@glo@text},%
10569     symbolaccess={\noexpand\@glo@textaccess},%
10570     symbolplural={\noexpand\@glo@plural},%
10571     firstpluralaccess=\relax,
10572     textaccess={\noexpand\@glo@shortaccess},%
10573     \the\glskeylisttok}%
10574 }%
10575 \let\@org@gls@assign@firstpl\gls@assign@firstpl
10576 \let\@org@gls@assign@plural\gls@assign@plural
10577 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
10578 \def\gls@assign@firstpl##1##2{%
10579   \@@gls@expand@field{##1}{firstpl}{##2}%
10580 }%
10581 \def\gls@assign@plural##1##2{%
10582   \@@gls@expand@field{##1}{plural}{##2}%
10583 }%
10584 \def\gls@assign@symbolplural##1##2{%
10585   \@@gls@expand@field{##1}{symbolplural}{##2}%
10586 }%
10587 \do@newglossaryentry
10588 \let\gls@assign@firstpl\@org@gls@assign@firstpl
10589 \let\gls@assign@plural\@org@gls@assign@plural
10590 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
10591 }

```

otnoteNewAcronymDef

```

10592 \renewcommand*{\FootnoteNewAcronymDef}{%
10593   \edef\@do@newglossaryentry{%
10594     \noexpand\newglossaryentry{\the\glslabeltok}%
10595     {%
10596       type=\acronymtype,%
10597       name={\noexpand\acronymfont{\the\glsshorttok}},%
10598       sort={\the\glsshorttok},%
10599       text={\the\glsshorttok},%
10600       textaccess={\the\glslongtok},%
10601       access={\noexpand\@glo@textaccess},%

```



```

10602 plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10603 short={\the\glsshorttok},%
10604 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10605 long={\the\glslongtok},%
10606 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10607 description={\the\glslongtok},%
10608 descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10609 \the\glskeylisttok
10610 }%
10611 }%
10612 \let\@org@gls@assign@plural\gls@assign@plural
10613 \let\@org@gls@assign@firstpl\gls@assign@firstpl
10614 \let\@org@gls@assign@descplural\gls@assign@descplural
10615 \def\gls@assign@firstpl##1##2{%
10616   \@@gls@expand@field{##1}{firstpl}{##2}%
10617 }%
10618 \def\gls@assign@plural##1##2{%
10619   \@@gls@expand@field{##1}{plural}{##2}%
10620 }%
10621 \def\gls@assign@descplural##1##2{%
10622   \@@gls@expand@field{##1}{descplural}{##2}%
10623 }%
10624 \@do@newglossaryentry
10625 \let\gls@assign@plural\@org@gls@assign@plural
10626 \let\gls@assign@firstpl\@org@gls@assign@firstpl
10627 \let\gls@assign@descplural\@org@gls@assign@descplural
10628 }

```

\SmallNewAcronymDef

```

10629 \renewcommand*{\SmallNewAcronymDef}{%
10630   \edef\@do@newglossaryentry{%
10631     \noexpand\newglossaryentry{\the\glslabeltok}%
10632     {%
10633       type=\acronymtype,%
10634       name={\noexpand\acronymfont{\the\glsshorttok}},%
10635       access={\noexpand\@glo@symbolaccess},%
10636       sort={\the\glsshorttok},%
10637       short={\the\glsshorttok},%
10638       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10639       shortaccess={\the\glslongtok},%
10640       long={\the\glslongtok},%
10641       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10642       text={\noexpand\@glo@short},%
10643       textaccess={\noexpand\@glo@shortaccess},%
10644       plural={\noexpand\@glo@shortpl},%
10645       first={\the\glslongtok},%
10646       firstaccess=\relax,
10647       firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
10648       description={\noexpand\@glo@first},%

```

```

10649     descriptionplural={\noexpand\@glo@firstplural},%
10650     symbol={\the\glsshorttok},%
10651     symbolaccess={\the\glslongtok},%
10652     symbolplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
10653     \the\glskeylisttok
10654 }%
10655 }%
10656 \let\@org@gls@assign@firstpl\gls@assign@firstpl
10657 \let\@org@gls@assign@plural\gls@assign@plural
10658 \let\@org@gls@assign@descplural\gls@assign@descplural
10659 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
10660 \def\gls@assign@firstpl##1##2{%
10661     \@@gls@expand@field{##1}{firstpl}{##2}%
10662 }%
10663 \def\gls@assign@plural##1##2{%
10664     \@@gls@expand@field{##1}{plural}{##2}%
10665 }%
10666 \def\gls@assign@descplural##1##2{%
10667     \@@gls@expand@field{##1}{descplural}{##2}%
10668 }%
10669 \def\gls@assign@symbolplural##1##2{%
10670     \@@gls@expand@field{##1}{symbolplural}{##2}%
10671 }%
10672 \do@newglossaryentry
10673 \let\gls@assign@firstpl\@org@gls@assign@firstpl
10674 \let\gls@assign@plural\@org@gls@assign@plural
10675 \let\gls@assign@descplural\@org@gls@assign@descplural
10676 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
10677 }

```

The following are kept for compatibility with versions before 3.0:

```

\glsshortaccesskey
10678 \newcommand*{\glsshortaccesskey}{\glsshortkey access}%

hortpluralaccesskey
10679 \newcommand*{\glsshortpluralaccesskey}{\glsshortpluralkey access}%

\glslongaccesskey
10680 \newcommand*{\glslongaccesskey}{\glslongkey access}%

longpluralaccesskey
10681 \newcommand*{\glslongpluralaccesskey}{\glslongpluralkey access}%

```

7.5 Debugging Commands

```

\showglongnameaccess
10682 \newcommand*{\showglongnameaccess}[1]{%
10683     \expandafter\show\csize glo@\glsdetoklabel{#1}@textaccess\endcsname
10684 }

```

```

\showglotextaccess
10685 \newcommand*{\showglotextaccess}[1]{%
10686   \expandafter\show\csname glo@\glsdetoklabel{#1}@textaccess\endcsname
10687 }

showglopluralaccess
10688 \newcommand*{\showglopluralaccess}[1]{%
10689   \expandafter\show\csname glo@\glsdetoklabel{#1}@pluralaccess\endcsname
10690 }

\showglofirstaccess
10691 \newcommand*{\showglofirstaccess}[1]{%
10692   \expandafter\show\csname glo@\glsdetoklabel{#1}@firstaccess\endcsname
10693 }

lofirstpluralaccess
10694 \newcommand*{\showglofirstpluralaccess}[1]{%
10695   \expandafter\show\csname glo@\glsdetoklabel{#1}@firstpluralaccess\endcsname
10696 }

showglosymbolaccess
10697 \newcommand*{\showglosymbolaccess}[1]{%
10698   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolaccess\endcsname
10699 }

osymbolpluralaccess
10700 \newcommand*{\showglosymbolpluralaccess}[1]{%
10701   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolpluralaccess\endcsname
10702 }

\showglodescaccess
10703 \newcommand*{\showglodescaccess}[1]{%
10704   \expandafter\show\csname glo@\glsdetoklabel{#1}@descaccess\endcsname
10705 }

glodescpluralaccess
10706 \newcommand*{\showglodescpluralaccess}[1]{%
10707   \expandafter\show\csname glo@\glsdetoklabel{#1}@descpluralaccess\endcsname
10708 }

\showgloshortaccess
10709 \newcommand*{\showgloshortaccess}[1]{%
10710   \expandafter\show\csname glo@\glsdetoklabel{#1}@shortaccess\endcsname
10711 }

loshortpluralaccess
10712 \newcommand*{\showgloshortpluralaccess}[1]{%
10713   \expandafter\show\csname glo@\glsdetoklabel{#1}@shortpluralaccess\endcsname
10714 }

```

```

\showglolongaccess
10715 \newcommand*{\showglolongaccess}[1]{%
10716   \expandafter\show\csname glo@glstetoklabel{#1}@longaccess\endcsname
10717 }

glolongpluralaccess
10718 \newcommand*{\showglolongpluralaccess}[1]{%
10719   \expandafter\show\csname glo@glstetoklabel{#1}@longpluralaccess\endcsname
10720 }

```

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.

```

10721 \NeedsTeXFormat{LaTeX2e}
10722 \ProvidesPackage{glossaries-babel}[2014/11/22 v4.12 (NLCT)]

```

Load tracklang to obtain language settings.

```

10723 \RequirePackage{tracklang}
10724 \let\glstettranslator\@secondoftwo

```

Check for tracked languages:

```

10725 \AnyTrackedLanguages
10726 {%
10727   \ForEachTrackedDialect{\this@dialect}{%
10728     \IfTrackedLanguageFileExists{\this@dialect}%
10729     {glossaries-}% prefix
10730     {.ldf}%
10731     {%
10732       \RequireGlossariesLang{\CurrentTrackedTag}%
10733     }%
10734     {%
10735       \PackageWarningNoLine{glossaries}%
10736       {No language module detected for ‘\this@dialect’.\MessageBreak
10737       Language modules need to be installed separately.\MessageBreak
10738       Please check on CTAN for a bundle called\MessageBreak
10739       ‘glossaries-\CurrentTrackedLanguage’ or similar}%
10740     }%
10741   }%
10742 }%
10743 {}%

```

8.1 Polyglossia Captions

Language support has now been split off into independent language modules.

```

10744 \NeedsTeXFormat{LaTeX2e}
10745 \ProvidesPackage{glossaries-polyglossia}[2014/11/22 v4.12 (NLCT)]

```

Load tracklang to obtain language settings.

```
10746 \RequirePackage{tracklang}
10747 \let\gl@ifusetranslator\@secondoftwo
```

Check for tracked languages:

```
10748 \AnyTrackedLanguages
10749 {%
10750   \ForEachTrackedDialect{\this@dialect}{%
10751     \IfTrackedLanguageFileExists{\this@dialect}%
10752     {glossaries-}% prefix
10753     {.ldf}%
10754     {%
10755       \RequireGlossariesLang{\CurrentTrackedTag}%
10756     }%
10757   }%
10758   \PackageWarningNoLine{glossaries}%
10759   {No language module detected for ‘\this@dialect’.\MessageBreak
10760   Language modules need to be installed separately.\MessageBreak
10761   Please check on CTAN for a bundle called\MessageBreak
10762   ‘glossaries-\CurrentTrackedLanguage’ or similar}%
10763 }%
10764 }%
10765 }%
10766 {}%
```

Glossary

`makeindex` An indexing application. [10](#), [25](#), [26](#), [160](#)

`xindy` An flexible indexing application with multilingual support written in Perl. [10](#), [25](#), [26](#), [160](#)

Change History

??		tected@edef to 'def	245
	super: fixed typo in \subglossentry	1.04	
	(\glossentrydesc)	272	General: Added \glstextformat
1.01			82
	General: Added range facility in	1.05	
	format key	98	\glossarysection: added
	\writeist: Added spaces after		\@mkboth to \glossarysection
	\delimN and \delimR in ist	
	file	145	37
1.03			\gls@defglossaryentry:
	\makefirsttuc: changed 'pro-		Changed the default value of
			the sort key to just the value of
			the name key
			72

- `\glsmakefirstuc`: new 246
- 1.06
 - General: now requires `etoolbox` . 244
 - `\capitalisewords`: new 246
 - `\xcapitalisewords`: new 247
- 1.07
 - `@gls@link`: fixed bug caused by `\theglsentrycounter` setting the page number too soon 96
 - `\glsadd`: fixed bug caused by `\theglsentrycounter` setting the page number too soon 142
- 1.08
 - General: Added babel support ... 31
 - `\capitalisewords`: made robust 246
 - `listgroup`: changed `listgroup` style to use `\glsgetgrouptitle` 253
 - `altlistgroup`: changed `altlistgroup` style to use `\glsgetgrouptitle` 254
 - `\makefirstuc`: made robust ... 244
- 1.09
 - `@mfu@nocaplist`: new 247
 - `\capitalisewords`: added check for words that shouldn't be capitalised 246
 - `\gMFUnocap`: new 247
 - `\mfu@checkword`: new 246
 - `\MFUclear`: new 247
- 1.1
 - `@glossarysection`: numbered sections and auto label added 39
 - `@gls@tmpb`: changed `\toksdef` to `\newtoks` 100
 - `@gls@toc`: numberline added .. 40
 - `@p@glossarysection`: numbered sections and auto label added 39
 - General: `amsgen` now loaded (`\new@ifnextchar` needed) .. 4
 - `translate`: `translate` option added 22
 - `\setglossarysection`: new ... 38
 - `numberedsection`: numbered-section package option added . 6
- numberline: numberline option added 5
- 1.12
 - `@GLSp1`: now uses `\glsentrydescplural` and `\glsentrysymbolplural` instead of `\glsentrydesc` and `\glsentrysymbol` 112
 - `@Glspl0`: now uses `\glsentrydescplural` and `\glsentrysymbolplural` instead of `\glsentrydesc` and `\glsentrysymbol` 111
 - `@glspl0`: now uses `\glsentrydescplural` and `\glsentrysymbolplural` instead of `\glsentrydesc` and `\glsentrysymbol` 110
 - General: added check for `\hypertarget` separate to `\hyperlink` (memoir defines `\hyperlink` but not `\hypertarget`) 106
 - `descriptionplural`: new 60
 - `\gls@defglossaryentry`: Changed default first plural to be first key with `s` appended (was text key with `s` appended) 72
 - `descriptionplural` support added 72
 - `symbolplural` support added .. 72
 - `\Glsentrydescplural`: New .. 136
 - `\glsentrydescplural`: New .. 136
 - `\Glsentrysymbolplural`: New 137
 - `\glsentrysymbolplural`: New 137
 - `\SetDescriptionFootnoteAcronymStyle`: Added `\protect` before `\footnote` and `\glslink` . 218
 - `\SetFootnoteAcronymStyle`: Added `\protect` before `\footnote` and `\glslink` . 225
 - `symbolplural`: new 61
- 1.13
 - General: fixed bug that ignored 3rd parameter 113–121
 - `\ACRfullpl`: new 200
 - `\Acrfullpl`: new 200
 - `\acrfullpl`: new 199
 - `\acrpluralsuffix`: New 197
 - `\gls@defglossaryentry`: Changed default first value .. 72

Changed default firstplural value	72	\@Glspl@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	111
Removed restriction on only using \newglossaryentry in the preamble	77	\@gls@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	108
\newacronym: Removed restriction on only using \newacronym in the preamble	197	\@glsdisp: Test glossary type is \acronymtype in addition to checking if footnote option has been used	113
1.14		\@glspl@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	110
\@gls@hypergroup: new	249	\@glstarget: raised the hyper-target so the target text doesn't scroll off the top of the page	106
General: added nonumberlist key to \printglossary	183	\gls@defglossaryentry: Changed def to let	72
added numberedsection key to \printglossary	181	1.17	
\firstacronymfont: new	201	\@do@wrglossary: new	163
\glsautoprefix: new	6	\@do@seeglossary: new	165
\glsnavhyperlink: changed 'edef to 'protected@edef ...	248	\@glo@storeentry: new	78
\glsnavhypertarget: added write to aux file	248	\@gls@glossary: changed definition to use \index instead of \@index	160
\glsnavigation: changed to only use labels for groups that are present	249	\@glsdefaultplural: new	63
1.15		\@glsdefaultsort: new	64
\@gls@link: added \glslabel .	96	\@glshypernumber: new	194
\gls@defglossaryentry: check for \@glo@first in description	76	\@glsnoname: new	63
check for \@glo@text in symbol	76	\@glsnonextpages: new	184
\gls@hypergroup: new .	249	General: added xindy support ...	25
\glsnavhypertarget: added check if rerun required	248	parent: new	62
\glssettoctitle: new	30	see: new	61
\printglossary: changed the way the TOC title is set	168	\gls@defglossaryentry: added nonumberlist key	72
1.16		added parent key	72
\@GLS@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	110	added see key	72
\@GLSpl: Test glossary type is \acronymtype in addition to checking if footnote option has been used	112	Stored main part of entry format when entry is defined	77
\@Gls@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	109	\gls@suffixF: new	35
		\gls@suffixFF: new	36
		\gls@wrglossary: modified to allow for xindy support	161
		\glshyperlink: new	142
		\glshypernumber: modified to allow material to be attached	

to location	194	\SetDescriptionFootnoteAcronymStyle:	
\glsnavhyperlink: replaced 'hyperlink to '@glslink	248	changed \acronymfont to use \textsmaller instead of \smaller	218
\glsnavhypertarget: replaced 'hypertarget to '@glstarget ..	248	\SetFootnoteAcronymStyle:	
\glssee: new	166	changed \acronymfont to use \textsmaller instead of \smaller	225
\glsseeformat: new	166	\SetSmallAcronymStyle:	
\glsSetSuffixF: new	35	changed \acronymfont to use \textsmaller instead of \smaller	228
\glsSetSuffixFF: new	36		
\ifglsxindy: new	25		
\istfilename: added xindy support	34		
\newglossarystyle: made		2.01	
\newglossarystyle long ..	193	\@gls@link: moved \@do@wrglossary before term is displayed to prevent unwanted whatsit	97
\nopostdesc: new	33	\forallglossaries: replaced \ifthenelse with \ifx	49
nonumberlist: new	62	\forglsentries: replaced \ifthenelse with \ifx	50
\printglossary: added check to determine if \printglossary is already defined	168	\glsdefmain: new	12
added print language to aux file	168	\glsdescwidth: changed \linewidth to \hsize ..	256, 271
order: order package option added	25	\glslistdottedwidth: changed \linewidth to \hsize	255
\writeist: added xindy support	145	\glspagelistwidth: changed \linewidth to \hsize ..	256, 271
1.18		nomain: added nomain package option	13
\@gls@loadlist: new	8	\writeist: removed item_02 - no such makeindex key	149
\@gls@loadlong: new	8		
\@gls@loadsuper: new	8	2.02	
\@gls@loadtree: new	8	\@printglossary: suppressed warning globally rather than locally	170
\gls@defglossaryentry:		\glossarysection: changed \@mkboth to \glossarymark	37
Changed default value of sort to \@glsdefaultsort	72	\glsglossarymark: New	38
moved sort sanitization to \newglossaryentry	76		
\glstarget: new	187	2.03	
\oldacronym: new	196	\@GLS@: Added check for hyper-first	110
nolist: new	8	\@GLSpl: Added check for hyper-first	112
nolong: new	8	\@Gls@: Added check for hyper-first	109
sort: moved sanitization to \newglossaryentry	60	\@Glspl@: Added check for hyper-first	111
nostyles: new	8	\@gls@: Added check for hyper-first	108
nosuper: new	8		
notree: new	8		
1.19			
\glsclearpage: new	40		
\glsdisp: new	112		
\SetDescriptionAcronymStyle:			
changed \acronymfont to use \textsmaller instead of \smaller	222		

\@gls@link: new	95	\Glsentryuseri: new	138
\@gls@link: added \leavevmode	96	\glsentryuseri: new	138
Moved entry existence check to avoid duplicate code	96	\Glsentryuserii: new	138
\@glsdisp: Added check for hyperfirst	113	\glsentryuserii: new	138
\@glspl@: Added check for hyperfirst	110	\Glsentryuseriii: new	139
\gls glossarymark: Added check to see if it's already defined ..	38	\glsentryuseriii: new	139
hyperfirst: new	23	\Glsentryuseriv: new	139
2.04		\glsentryuseriv: new	139
\@GLS@: Changed test to check if glossary type has been identified as a list of acronyms ...	110	\Glsentryuseriv: new	139
\@GLSpl@: Changed test to check if glossary type has been identified as a list of acronyms ...	112	\Glsentryuserv: new	139
\@GLs@: Changed test to check if glossary type has been identified as a list of acronyms ...	109	\glsentryuserv: new	139
\@GLSpl@: Changed test to check if glossary type has been identified as a list of acronyms ..	111	\Glsentryuservi: new	139
\@glossaryentryfield: new ..	78	\glsentryuservi: new	139
\@glossarysubentryfield: new	78	\ns@newglossary: added check to determine if \gls@<type>@display and \gls@<type>@displayfirst have been defined.	57
\@gls@: Changed test to check if glossary type has been identified as a list of acronyms ...	108	\SetAcronymLists: new	15
\@glsacronymlists: new	14	\SetDefaultAcronymDisplayStyle: new	215
\@glsdisp: Changed test to check if glossary type has been identified as a list of acronyms ..	113	\SetDefaultAcronymStyle: new	215
\@glspl@: Changed test to check if glossary type has been identified as a list of acronyms ..	110	\SetDescriptionAcronymDisplayStyle: new	220
\@newglossaryentryposthook: new	77	\SetDescriptionDUAAcronymDisplayStyle: new	219
\@newglossaryentryprehook: new	77	\SetDescriptionFootnoteAcronymDisplayStyle: new	216
acronymlists: new	15	\SetDUADisplayStyle: new ..	228
\DeclareAcronymList: new ...	14	\SetFootnoteAcronymDisplayStyle: new	223
\DefineAcronymSynonyms: new	213	\SetSmallAcronymDisplayStyle: new	225
\gls@defglossaryentry: added user1-6 keys	73	2.05	
\glsadd: fixed bug that ignored counter	142	\@glsdisp: Added closing brace. Patch provided by Sergiu Dotenco	113
		Removed spurious brace. Patch provided by Sergiu Dotenco	113
		\writeist: Added \string before opening and closing braces. Patch provided by Segiu Dotenco	150
		2.06	
		\altnewglossary: new	57
		\CustomAcronymFields: new ..	230
		\CustomNewAcronymDef: new ..	231
		\SetCustomDisplayStyle: new	230
		\SetCustomStyle: new	231

2.07		\acrfull: added starred version	198
General: glssadd format key stored in \@glsnumberformat (was mistakenly stored in \@glo@format)	142	\ACRfullpl: added starred ver- sion	200
3.0		\Acrfullpl: added starred ver- sion	200
\@do@wrglossary: added check for hyper location prefix ...	163	\acrfullpl: added starred ver- sion	199
modified to use new format ..	163	\acrlinkfootnote: new	216
\@glossarysec: replaced \@ifundefined with \ifcsundef	6	\acrnolinkfootnote: new ...	216
\@do@seeglossary: Sanitize and escape cross-referencing in- formation	165	savewrites: new	26
\@gls@counterwithin: new ...	10	see: added \@glo@seeautonumberlist	61
\@gls@ifinlist: new	41	seeautonumberlist: new	8
\@gls@link: added \@gls@saveentrycounter	97	\glossarysection: replaced \@ifundefined with \ifcsundef	37
added \@gls@setsort	97	\glossarystyle: replaced \@ifundefined with \ifcsundef	192
\@gls@saveentrycounter: new	97	\gls@codepage: replaced \@ifundefined with \ifcsundef	25
\@gls@setupsort@def: new ...	11	\gls@defglossaryentry: added \@gls@defsort	76
\@gls@setupsort@standard: new	10	added short and long keys	73
\@gls@setupsort@use: new ...	11	replaced \@ifundefined with \ifcsundef	73
\@gls@xdy@locationlist: new	44	\gls@doclearpage: replaced \@ifundefined with \ifcsundef	39
\@glslink: replaced \@ifundefined with \ifcsundef	106	\gls@wrglossary: modified to take into account savewrites	161
\@glsnextpages: new	184	\glsadd: added \@gls@saveentrycounter	143
\@makeglossary: Added check for savewrites	151	\GlsAddXdyCounters: new	41
\@print@glossary: replaced \@ifundefined with \ifcsundef	170	\glsentrycounterlabel: new	186
\@printglossary: added \currentglossary	169	\glsentryitem: new	186
added \@glsnextpages	169	\Glsentrylong: new	140
make toctitle default to title ..	169	\glsentrylong: new	140
\@set@glo@numformat: added 4th argument	98	\Glsentrylongpl: new	140
\@xdy@attributelist: new	41	\glsentrylongpl: new	140
General: added prefix to hyperlink	195	\Glsentryshort: new	140
etoolbox now loaded	4	\glsentryshort: new	139
replaced \@ifundefined with \ifcsundef	29, 32, 93, 181	\Glsentryshortpl: new	140
\acrfootnote: new	216	\glsentryshortpl: new	140
\ACRfull: added starred version	199	\glsgetgrouptitle: re- placed \@ifundefined with \ifcsundef	191
\Acrfull: added starred version	198		

<code>\glsglossarymark:</code>	replaced	<code>entrycounterwithin:new</code> 9
<code>\@ifundefined</code>	with	<code>\oldacronym:replaced\@ifundefined</code>	
<code>\ifcsundef</code> 38	<code>with\ifcsundef</code> 196
<code>\glshyperlink:</code>	changed default from <code>\glsentryname</code> to	<code>compatible-2.07: compatible-</code>	
<code>\glsentrytext</code> 142	<code>2.07 option added</code> 27
<code>\glshypernumber:</code>	replaced	<code>long:new</code> 63
<code>\@ifundefined</code>	with	<code>longplural:new</code> 63
<code>\ifcsundef</code> 194	<code>nonumberlist: now boolean</code>	... 62
<code>\glsnumberformat:</code>	replaced	<code>sort:new</code> 10
<code>\@ifundefined</code>	with	<code>counter:replaced\@ifundefined</code>	
<code>\ifcsundef</code> 36	<code>with\ifcsundef</code> 61
<code>\glsrefentry:new</code> 186	<code>\printglossary:</code>	replaced
<code>\glsresetsubentrycounter:</code>		<code>\@ifundefined</code>	with
<code>new</code> 185	<code>\ifcsundef</code> 168
<code>\glsseeitem:</code>	hyperlink uses	<code>\SetDescriptionFootnoteAcronymDisplayStyle:</code>	
<code>\glsseeitemformat</code>	instead	<code>expanded options link op-</code>	
<code>of\glsentryname</code> 167	<code>tions</code> 216
<code>\glsseeitemformat:new</code> 167	<code>\setentrycounter:</code>	added op-
<code>\glssortnumberfmt:new</code> 11	<code>tional argument</code> 192
<code>\glsstepentry:new</code> 185	<code>\showacronymlists:new</code> 236
<code>\glsstepsubentry:new</code> 185	<code>\showglocounter:new</code> 233
<code>\glssubentrycounterlabel:</code>		<code>\showglodesc:new</code> 235
<code>new</code> 186	<code>\showglodescplural:new</code>	... 235
<code>\glssubentryitem:new</code> 186	<code>\showglofirst:new</code> 233
<code>theglossary:replaced\@ifundefined</code>		<code>\showglofirstpl:new</code> 233
<code>with\ifcsundef</code> 186	<code>\showgloflag:new</code> 236
<code>short:new</code> 63	<code>\showgloindex:new</code> 236
<code>shortplural:new</code> 63	<code>\showglolevel:new</code> 232
<code>\ifglossaryexists:</code>	re-	<code>\showglongame:new</code> 235
<code>placed \@ifundefined with</code>		<code>\showgloparent:new</code> 232
<code>\ifcsundef</code> 50	<code>\showgloplural:new</code> 233
<code>\ifglentryexists:</code>	re-	<code>\showglosort:new</code> 235
<code>placed \@ifundefined with</code>		<code>\showglossaries:new</code> 237
<code>\ifcsundef</code> 51	<code>\showglossarycounter:new</code>	. 237
<code>\istfile: deprecated</code> 159	<code>\showglossaryentries:new</code>	. 237
<code>glossaryentry:new</code> 184	<code>\showglossaryin:new</code> 237
<code>glossarysubentry:new</code> 185	<code>\showglossaryout:new</code> 237
<code>\newglossaryentry:</code>	replaced	<code>\showglossarytitle:new</code>	... 237
<code>\DeclareRobustCommand</code>		<code>\showglosymbol:new</code> 235
<code>with\newrobustcmd</code> 66	<code>\showglosymbolplural:new</code>	. 235
<code>\newglossarystyle:</code>	re-	<code>\showglotext:new</code> 233
<code>placed \@ifundefined with</code>		<code>\showglotype:new</code> 233
<code>\ifcsundef</code> 193	<code>\showglouseri:new</code> 234
<code>\ns@newglossary:</code>	added	<code>\showglouserii:new</code> 234
<code>\@gls@defsortcount</code> 57	<code>\showglouseriii:new</code> 234
<code>replaced \@ifundefined with</code>		<code>\showglouseriv:new</code> 234
<code>\ifcsundef</code> 57	<code>\showglouserv:new</code> 234
<code>entrycounter:new</code> 9	<code>\showglouservi:new</code> 234
		<code>subentrycounter:new</code> 10

\writeist: added xindy-only macro definitions to glossary open tag	147	\Glspl: made robust	111
modified to support new for- mat	145	\glspl: made robust	110
3.01		\GLSplural: made robust	116
\@glswritefiles: added check for empty glossaries	159	\GLSsymbol: made robust	120
General: made robust	109	\Glsymbol: made robust	120
\ACRfull: made robust	199	\glssymbol: made robust	120
\Acrfull: made robust	198	\GLSsymbolplural: made robust	121
\acrfull: made robust	198	\Glsymbolplural: made robust	121
\acrfullformat: removed \acronymfont as it should al- ready be set in the second ar- gument.	198	\glssymbolplural: made robust	120
\ACRfullpl: made robust	200	\Glstext: made robust	114
\Acrfullpl: made robust	200	\glstext: made robust	113
\acrfullpl: made robust	199	\GLSuseri: made robust	122
\ACRlong: made robust	131	\Glsuseri: made robust	122
\Acrlong: made robust	131	\glsuseri: made robust	121
\acrlong: made robust	130	\GLSuserii: made robust	123
\ACRlongpl: made robust	133	\Glsuserii: made robust	123
\Acrlongpl: made robust	133	\glsuserii: made robust	122
\acrlongpl: made robust	132	\GLSuseriii: made robust	124
\ACRshort: made robust	128	\Glsuseriii: made robust	123
\Acrshort: made robust	127	\glsuseriii: made robust	123
\acrshort: made robust	127	\GLSuseriv: made robust	125
\ACRshortpl: made robust	130	\Glsuseriv: made robust	124
\Acrshortpl: made robust	129	\glsuseriv: made robust	124
\acrshortpl: made robust	128	\GLSuseriv: made robust	126
\Gls: made robust	108	\Glsuseriv: made robust	125
\glsadd: made robust	142	\GLSuseriv: made robust	126
\glsaddall: made robust	143	\Glsuseriv: made robust	126
\GLSdesc: made robust	118	\GLSuseriv: made robust	126
\Glsdesc: made robust	118	\GLSuseriv: made robust	126
\glsdesc: made robust	118	\GLSuseriv: made robust	126
\GLSdescplural: made robust .	119	\GLSuseriv: made robust	126
\Glsdescplural: made robust .	119	\GLSuseriv: made robust	126
\glsdescplural: made robust .	119	\GLSuseriv: made robust	126
\glsfirst: made robust	114	\GLSuseriv: made robust	126
\GLSfirstplural: made robust	117	\GLSuseriv: made robust	126
\Glsfirstplural: made robust	116	\GLSuseriv: made robust	126
\glsfirstplural: made robust	116	\GLSuseriv: made robust	126
\glslink: made robust	95	\GLSuseriv: made robust	126
\GLSname: made robust	117	\GLSuseriv: made robust	126
\Glsname: made robust	117	\GLSuseriv: made robust	126
\glsname: made robust	117	\GLSuseriv: made robust	126
\GLSpl: made robust	112	\GLSuseriv: made robust	126
		3.02	
		\@do@wrglossary: changed \@glslocref to \theglscopycounter	164
		\@do@wrglossary: changed \@do@wr@glossary to test for indexonlyfirst option; put old \@do@wr@glossary code into \@do@wrglossary	161
		\@gls@missingnumberlist: new	64
		\@glswritefiles: added check for existence of token in case \makeglossaries has been omitted	159
		\@printglossary: add a way to fetch current entry label ...	169

savenumberlist:new	7	\glspostinline: replaced “.”	
ucmark:new	9	with \glspostdescription	252
\gls@defglossaryentry: added		altlongragged4col: added	
numberlist element	76	check for glsnogroupskip ..	266
\gls@save@numberlist:new .	167	altsuperragged4col: added	
\gls@wrglossary: added check		check for glsnogroupskip ..	282
for glossary file defined	161	alttree: added check for	
\glsdisplaynumberlist:new	141	glsnogroupskip	290
\glsentrycounter: set default		index: added check for	
value	97	glsnogroupskip	285
\Glsentryfull: fixed bug (re-		nogroupskip:new	9
placed \glsentryshortpl		long: added check for	
with \glsentryshort)	140	glsnogroupskip	256
\glsentryfullpl: fixed bug (re-		long3col: added check for	
placed \glsentryshort with		glsnogroupskip	258
\glsentryshortpl)	140	long4col: added check for	
\glsentrynumberlist:new ..	141	glsnogroupskip	259
\glsmoveentry:new	77	longragged: added check for	
\glsnumlistlastsep:new ...	142	glsnogroupskip	263
\glsnumlistsep:new	142	longragged3col: added check	
\glsresetsubentrycounter:		for glsnogroupskip	264
new	185	nopostdot:new	9
\ifglshaschildren:new	52	tree: added check for	
\ifglshasparent:new	52	glsnogroupskip	286
\makeglossaries: added list		treenoname: added check for	
parser	154	glsnogroupskip	288
indexonlyfirst:new	23	super: added check for	
\renewglossarystyle:new ..	193	glsnogroupskip	272
\showglossaryentries: fixed		super3col: added check for	
misspelt command	237	glsnogroupskip	274
\SmallNewAcronymDef: fixed		super4col: added check for	
broken short and long plural	226	glsnogroupskip	275
3.03		superragged: added check for	
\@gls@sanitizesort:new	18	glsnogroupskip	279
\@gls@setupsort@standard:		superragged3col: added check	
used \@gls@sanitizesort .	10	for glsnogroupskip	281
\@printglossary: allow title to		3.04	
override default toctitle	169	\@do@wrglossary: changed	
General: allow title to set toctitle	181	\theglsentrycounter back	
\glsinlinedescformat:new .	252	to \@glslocref	164
\glsinlineemptydescformat:		\@do@wrglossary: modified to	
new	252	compensate for possible incor-	
\glsinlinenameformat:new .	252	rect page number	163
\glsinlinepostchild:new ..	252	\@gls@escbsdq: unsani-	
\glsinlinesubdescformat:		tize \gls@numberpage,	
new	252	\gls@alphpage, \gls@Alphpage	
\glsinlinesubnameformat:		and \gls@romanpage	99
new	252	\@print@glossary: Moved aux	
		write to end of document to	

prevent unwanted whatsit occurring here.	170		
General: Added check for doc package	4		
added datatool-base as a required package	4		
added local key	93		
\gls@Alphpage: new	162		
\gls@alphpage: new	162		
\gls@disablepagerefexpansion: new	161		
\gls@numberpage: new	162		
\gls@protected@pagefmts: new	161		
\gls@romanpage: new	162		
\glsdefmain: added check for doc package	12		
\glsorg@endtheglossary: new .	5		
\glsorg@theglossary: new	5		
altlist: replaced \newline with paragraph break	254		
\PrintChanges: new	5		
3.05			
\@do@wrglossary: add Roman case. Fixed bugs in the else statements	163		
\@gls@link: added check for “no-hypertypes”	96		
\@gls@nohyperlist: new	16		
mcolalttree: replaced ‘2’ with \glsmcols	270		
mcolindex: replaced ‘2’ with \glsmcols	268		
mcoltree: replaced ‘2’ with \glsmcols	268		
mcoltreename: replaced ‘2’ with \glsmcols	269		
\gls@protected@pagefmts: added Roman to list	161		
\gls@Romanpage: new	162		
\GlsDeclareNoHyperList: new	16		
\glsgetgrouplabel: fixed bug (typo in \equal)	191		
\nopostdesc: made robust	33		
nohypertypes: new	16		
3.06			
\@xdy@main@language: Changed back to using \language name	25		
\findrootlanguage: Obsoleted	48		
		3.07	
		\@gls@link: fixed bug that failed to find entry in list	96
		\glossarypreamble: modified to work with \setglossarypreamble	37
		\gls@docclearpage: added check for openright	39
		\glspostdescription: Added spacefactor code	9
		\GlsSetXdyCodePage: Added check for fontspec	49
		\SetDescriptionAcronymDisplayStyle: now using \glsdoparenifnotempty	220
		\setglossarypreamble: new ..	37
		3.08a	
		\@glo@storeentry: no longer need to check for special characters in any of the fields other than sort	78
		updated for \glossentry	79
		\@glossaryentryfield: switched to \glossentry	78
		\@glossarysubentryfield: switched to \subglossentry	78
		General: added nogroupskip key to \printglossary	182
		removed definition of \@glossaryentryfield ..	331
		removed definition of \@glossarysubentryfield	331
		\compatibleglossentry: new	187
		\compatiblesubglossentry: new	188
		\glossaryentryfield: deprecated	189
		\Glossentrydesc: new	188
		\glossentrydesc: new	188
		\Glossentryname: new	188
		\glossentryname: new	187
		\Glossentrysymbol: new	188
		\glossentrysymbol: new	188
		\gls@assign@desc@field: new	18
		\gls@assign@descplural@field: new	18
		\gls@assign@field: new	66
		\gls@ifnotmeasuring: new ...	80
		\glsaddallunused: new	143

\glsexpandfields: new	66	3.09a	
\glsnoexpandfields: new	66		\@gls@assign@symbolplural@field:
\glssee: made robust	166		new
\glsseeformat: made robust ..	166		18
\glsseeitem: made robust	167		\@gls@default@value: new ...
\glsseelist: made robust	166		60
\ifglsdescsuppressed: new ..	53		\Glsentrydesc: made robust ..
\ifglshasdesc: new	53		136
\ifglshassymbol: new	53		\Glsentrydescplural: made ro-
list: updated list style to			bust
use \glossentry and			136
\subglossentry	253		\Glsentryfirst: made robust .
listdotted: updated listdotted			137
style to use \glossentry and			\Glsentryfirstplural: made
\subglossentry	255		robust
altlist: updated altlist style			138
to use \glossentry and			\Glsentryfull: made robust ..
\subglossentry	254		140
altlongragged4col: updated			\Glsentryfullpl: made robust
to use \glossentry and			141
\subglossentry	265		\Glsentrylong: made robust ..
alttree: updated to use			140
\glossentry and \subglossentry			\Glsentrylongpl: made robust
.....	289		140
index: added paragraph break at			\Glsentryname: made robust ..
end of environment	284		135
updated to use \glossentry			\Glsentryplural: made robust
and \subglossentry	284		137
inline: updated inline style			\Glsentryshort: made robust .
to use \glossentry and			140
\subglossentry	250		\Glsentryshortpl: made robust
long: updated to use \glossentry		
and \subglossentry	256		140
longragged: updated to			\Glsentrysymbol: made robust
use \glossentry and			137
\subglossentry	262		\Glsentrysymbolplural: made
longragged3col: updated			robust
to use \glossentry and			137
\subglossentry	264		\Glsentrytext: made robust ..
tree: updated to use \glossentry			136
and \subglossentry	286		\Glsentryuseri: made robust .
\setglossarystyle: new	192		138
\setglossentrycompatibility:			\Glsentryuserii: made robust
new	189		138
superragged: updated to			\Glsentryuseriii: made robust
use \glossentry and		
\subglossentry	279		139
			\Glsentryuseriv: made robust
			139
			\Glsentryuserv: made robust .
			139
			\Glsentryuservi: made robust
			139
			\glstextup: new
			197
			\ifglshassymbol: changed test
			to check for \@gls@default@symbol
		
			53
		3.10a	
			\@gls@keymap: new
			68
			\@gls@provide@newglossary:
			new
			55
			\@gls@writedef: new
			67
			\@gls@defaultplural: Obsolete .
			63
			\@gls@nodesc: new
			63
			\@print@glossary: Added
			providecommand code to aux
			file
			170, 171
			\gls@assign@type@field: new
			17
			\gls@defglossaryentry:
			Changed to using \@gls@default@value
		
			72
			new
			72

<code>\glswritedefhook: new</code>	71	removed <code>\makefirstuc</code> (now dealt with in <code>\glentryfmt</code>)	109
<code>\makeglossaries:</code> Added providecommand code to aux file	153	<code>\@Glspl@:</code> add <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glscustomtext</code> and <code>\glsinsert</code>	111
<code>\new@glossaryentry: new</code>	66	change to using <code>\glentryfmt</code> style commands	111
<code>\ns@newglossary:</code> added <code>\@gls@provide@newglossary</code>	57	removed <code>\makefirstuc</code> (now dealt with in <code>\glentryfmt</code>)	111
3.11a		<code>\@acrlong:</code> added <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glsinsert</code> and <code>\glscustomtext</code>	330
<code>\@ACRlong:</code> added <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glsinsert</code> and <code>\glscustomtext</code>	330	<code>\@acrshort:</code> added <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glsinsert</code> and <code>\glscustomtext</code>	329
<code>\@ACRshort:</code> added <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glsinsert</code> and <code>\glscustomtext</code>	329	<code>\@gls@:</code> add <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glscustomtext</code> and <code>\glsinsert</code>	108
<code>\@Acrlong:</code> added <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glsinsert</code> and <code>\glscustomtext</code>	330	change to using <code>\glentryfmt</code> style commands	108
<code>\@Acrshort:</code> added <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glsinsert</code> and <code>\glscustomtext</code>	329	<code>\@gls@noexpand@fields:</code> Fixed bug expand replaced with noexpand	64
<code>\@GLS@:</code> add <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glscustomtext</code> and <code>\glsinsert</code>	109	<code>\@glsdisp:</code> add <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glscustomtext</code> and <code>\glsinsert</code>	113
change to using <code>\glentryfmt</code> style commands	109	change to using <code>\glentryfmt</code> style commands	113
removed <code>\MakeUppercase</code> (now moved to <code>\glentryfmt</code>)	110	<code>\@Glspl@:</code> add <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glscustomtext</code> and <code>\glsinsert</code>	110
<code>\@GLSpl:</code> add <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glscustomtext</code> and <code>\glsinsert</code>	112	change to using <code>\glentryfmt</code> style commands	110
change to using <code>\glentryfmt</code> style commands	112	General: added <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glsinsert</code> and <code>\glscustomtext</code>	127–134
removed <code>\MakeUppercase</code> as now dealt with in <code>\glentryfmt</code>	112	changed to just use <code>\Glsentrydescplural</code>	119
<code>\@Gls@:</code> add <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glscustomtext</code> and <code>\glsinsert</code>	109	changed to just use <code>\glentrydescplural</code>	119
change to using <code>\glentryfmt</code> style commands	109	changed to just use <code>\Glsentrydesc</code>	118

changed to just use <code>\glentrydesc</code> 118, 119	changed to just use <code>\glentryuservi</code> 126, 127
changed to just use <code>\Glsentryfirstplural</code> 116	changed to just use <code>\Glsentryuserv</code> 125
changed to just use <code>\glentryfirstplural</code> 116, 117	changed to just use <code>\glentryuserv</code> 125, 126
changed to just use <code>\Glsentryfirst</code> 115	Now requires textcase 4
changed to just use <code>\glentryfirst</code> 114, 115	acronymlists: replaced <code>\@addtoacronymlists</code> with <code>\DeclareAcronymList</code> 15
changed to just use <code>\Glsentryname</code> 117	<code>\defglssdisplay: obsolete</code> 92
changed to just use <code>\glentryname</code> 117, 118	<code>\defglssdisplayfirst: obso-</code> leted 92
changed to just use <code>\Glsentryplural</code> 116	<code>\defglssentryfmt: new</code> 55
changed to just use <code>\glentryplural</code> 115, 116	<code>\forglssentries: replaced \ifx</code> with <code>\ifdefempty</code> 50
changed to just use <code>\Glsentrysymbolplural</code> 121	<code>\gls@assign@desc: new</code> 71
changed to just use <code>\glentrysymbolplural</code> 121	<code>\gls@defglossaryentry: Fixed</code> default counter if none sup- plied 75
changed to just use <code>\Glsentrysymbol</code> 120	<code>\gls@doentryfmt: new</code> 55
changed to just use <code>\glentrysymbol</code> 120	<code>\glsdisplay: obsolete</code> 92
changed to just use <code>\glentrysymbol</code> 120	<code>\glsdisplayfirst: obsolete</code> .. 91
Changed to just use <code>\Glsentrytext</code> 114	<code>\gls@genentryfmt: new</code> 86
changed to just use <code>\glentrytext</code> 113	<code>\glsgetgrouptitle: Added</code> check in case non-Latin alpha- bet in use 191
changed to just use <code>\Glsentryuseriii</code> 124	<code>\gls@navigation: switched to us-</code> ing <code>\@gls@getgrouptitle</code> 249
changed to just use <code>\glentryuseriii</code> 123, 124	<code>\ifglshasdesc: replaced</code> <code>\ifdefempty</code> with <code>\ifcsemt</code> 53
changed to just use <code>\Glsentryuserii</code> 123	<code>\ifglshaslong: new</code> 53
changed to just use <code>\glentryuserii</code> 122, 123	<code>\ifglshasshort: new</code> 53
changed to just use <code>\Glsentryuseriv</code> 125	<code>\ifglshassymbol: replaced</code> <code>\ifdefempty</code> with <code>\ifcsemt</code> 53
changed to just use <code>\glentryuseriv</code> 124, 125	<code>\ifglssused: replaced \ifthenelse</code> with <code>\ifbool</code> 51
changed to just use <code>\Glsentryuseri</code> 122	<code>\longnewglossaryentry: new</code> . 71
changed to just use <code>\glentryuseri</code> 122	<code>\ns@newglossary: replaced</code> <code>\glsdisplay</code> and <code>\glsdisplayfirst</code> with <code>\glssentryfmt</code> 57
changed to just use <code>\Glsentryuservi</code> 126	compatible-3.07: cnew 27
	<code>\SetCustomDisplayStyle: up-</code> dated to use <code>\defglssentryfmt</code>

.....	230	\glossarystyle: fixed bug
\SetDefaultAcronymDisplayStyle:		caused by using \ifdef in-
changed to use \defglentryfmt		stead of \ifcsdef
.....	215	192
\SetDescriptionAcronymDisplayStyle:		\gls@assign@desc@field:
updated to use \defglentryfmt		changed to use \glsetnoexpandfield
.....	220 18
\SetDescriptionDUAAcronymDisplayStyle:		\gls@assign@descplural@field:
updated to use \defglentryfmt		changed to use \glsetnoexpandfield
.....	219 18
\SetDescriptionFootnoteAcronymDisplayStyle:		\gls@assign@name@field:
updated to use \defglentryfmt		changed to use \glsetnoexpandfield
.....	216 18
\SetDUADisplayStyle: updated		\gls@assign@type@field:
to use \defglentryfmt ..	228	changed to use \glsetexpandfield
\SetFootnoteAcronymDisplayStyle:	 17
updated to use \defglentryfmt		\gls@checkseeallowed: new .. 61
.....	223	\glsaddallunused: set default to
\SetSmallAcronymDisplayStyle:		\@glo@types
updated to use \defglentryfmt		143
.....	225	\Glsentryfull: changed to use
\setupglossaries: new	28	\acrfullformat
\showglolong: new	236	140
\showgloshort: new	236	\Glsentryfullpl: changed to
numbers: new	27	use \acrfullformat
symbols: new	27	141
3.12a		\glsentryfullpl: changed to
\gls@defglossaryentry: added		use \acrfullformat
\glslabel	72	140
\glsaddkey: new	69	\gls@checkseeallowed: renamed
3.13a		\glossarymark to \glsglossarymark
\@gls@assign@symbol@field:		to avoid conflict with memoir .. 38
changed to use \glsetnoexpandfield		\glspreststandardsort: new ... 10
.....	18	\glsetexpandfield: new 17
\@gls@assign@symbolplural@field:		\glsetnoexpandfield: new .. 17
changed to use \glsetnoexpandfield		altsuper4colheader: switched
.....	18	to \tabularnewline
\@gls@link: removed \relax ..	97	277
\@gls@notranslatorhook: new	21	altsuper4colheaderborder:
\@gls@setupsort@standard:		switched to \tabularnewline
moved \@gls@santizesort	 277
to \glspreststandardsort ..	10	long: switched to \tabularnewline
ucmark: added check for memoir ..	9 256
see: added \gls@checkseeallowed		long3col: switched to \tabularnewline
.....	61 258
\glossarysection: changed		long3colheader: switched to
\glossarymark to \glsglossarymark		\tabularnewline
.....	37	258
		long3colheaderborder: switched
		to \tabularnewline
		259
		long4col: switched to \tabularnewline
	 259
		long4colheader: switched to
		\tabularnewline
		260

longheader: switched to \tablearnewline 257	document class 7
longheaderborder: switched to \tablearnewline 257	\CustomAcronymFields: in- serted missing comma 231
\SetFootnoteAcronymDisplayStyle: fixed missing argument bug 223	4.02
super: switched to \tablearnewline 272	\@acrfull: now using \acrfullfmt 198
super3col: switched to \tablearnewline 273	\@gls@indexdef: new 28
super3colheader: switched to \tablearnewline 274	\@gls@numbersdef: new 27
super4col: switched to \tablearnewline 275	\@gls@symbolsdef: new 27
super4colheader: switched to \tablearnewline 275	General: Removed \acronymfont 131–134
super4colheaderborder: switched to \tablearnewline 276	\ACRfullfmt: new 199
superheader: switched to \tablearnewline 272	\Acrfullfmt: new 199
superheaderborder: switched to \tablearnewline 273	\acrfullfmt: new 198
3.14a	\ACRfullplfmt: new 200
\@glswritefiles: renamed \glswritefiles to \@glswritefiles and used “savewrites” option to set \glswritefiles 159	\Acrfullplfmt: new 200
General: new 238	\acrfullplfmt: new 200
acronyms: new 14	\acronymentry: new 202
\gls@defglossaryentry: added check for existence of default glossary 73	sanitize: fixed bug that caused an error here 21
set the default for firstplural to be the value of plural 75	sc-short-long: new 206
xindygloss: new 26	sc-short-long-desc: new ... 207
\longprovideglossaryentry: new 72	\Genacrfullformat: new 91
compatible-2.07: added check for 2.07 before setting 3.07 compatibility 27	\genacrfullformat: new 91
notranslate: new 22	\GenericAcronymFields: new 202
\provideglossaryentry: new . 66	\Genplacrfullformat: new ... 91
4.0	\genplacrfullformat: new ... 91
\gls@defglossaryentry: added check for first key 75	\Glsentryfull: bug fix: added missing \acronymfont 140
4.01	\Glsentryfull: bug fix: added missing \acronymfont 140
General: fixed non-value options so that they can be passed to	\Glsentryfullpl: bug fix: added missing \acronymfont 141
	\Glsentryfullpl: bug fix: added missing \acronymfont 140
	\glsngenacfmt: new 89
	\GlsUseAcrEntryDisplayStyle: new 204
	\GlsUseAcrStyleDefs: new .. 204
	short-long: new 205
	short-long-desc: new 207
	xindynoglsnumbers: new 26
	sm-short-long: new 206
	sm-short-long-desc: new ... 208
	\makeglossaries: made pream- ble only 154
	index: new 28
	\newacronymstyle: new 203
	long-sc-short: new 205

long-sc-short-desc: new ...	206	\@Gls@: removed \glslabel (defined in \@gls@link)	109
long-short: new	204	\@Gls@entry@field: new	134
long-short-desc: new	206	\@Glspl@: removed \glslabel (defined in \@gls@link) ...	111
long-sm-short: new	205	\@acrlong: removed \glslabel (defined in \@gls@link) ...	330
long-sm-short-desc: new ...	207	\@acrshort: removed \glslabel (defined in \@gls@link) ...	329
footnote: new	210	\@gls@: removed \glslabel (defined in \@gls@link)	108
footnote-desc: new	212	\@gls@access@display: new .	317
footnote-sc: new	211	\@gls@entry@field: new	134
footnote-sc-desc: new	212	\@gls@fetchfield: new	68
footnote-sm: new	212	\@gls@field@link: new	113
footnote-sm-desc: new	212	\@gls@link: added \glsdetoklabel	
\setacronymstyle: new	203	moved \@gls@link@opts and \@gls@link@label to \@gls@link	96
\SetDescriptionAcronymDisplayStyle: Moved check for empty custom text to prevent unwanted parenthetical material	220	\@gls@writedef: added \glsdetoklabel	67
\SetDescriptionFootnoteAcronymDisplayStyle: Moved check for empty custom text to prevent unwanted parenthetical material	217	\@glsdisp: removed \glslabel (defined in \@gls@link) ...	113
\SetFootnoteAcronymDisplayStyle: Moved check for empty custom text to prevent unwanted parenthetical material	223	\@glspl@: removed \glslabel (defined in \@gls@link) ...	110
\SetGenericNewAcronym: new	201	\@printglossary: added \glsdetoklabel	169
\SetSmallAcronymDisplayStyle: Moved check for empty custom text to prevent unwanted parenthetical material	225	General: changed default to \@empty instead of \relax ..	26
dua: new	208	removed \glslabel (defined in \@gls@link)	127
dua-desc: new	210	sc-short-long-desc: redefined to use accessibility information	335
numberedsection: added		\compatibleglossentry: added \glsdetoklabel	311
nameref option	6	\compatiblesubglossentry: added \glsdetoklabel ...	312
4.03		\Genacrfullformat: redefined to use accessibility information	328
\@@do@wrglossary: added		\genacrfullformat: redefined to use accessibility information	328
\glsdetoklabel	164	\Genplacrfullformat: redefined to use accessibility information	328
\@ACRlong: removed \glslabel (defined in \@gls@link) ...	330		
\@ACRshort: removed \glslabel (defined in \@gls@link) ...	329		
\@Acrlong: removed \glslabel (defined in \@gls@link) ...	330		
\@Acrshort: removed \glslabel (defined in \@gls@link) ...	329		
\@GLS@: removed \glslabel (defined in \@gls@link)	109		
\@GLSpl: removed \glslabel (defined in \@gls@link) ...	112		

<code>\genplacrfullformat:</code> redefined to use accessibility information 328	<code>\glstentrytextaccess:</code> switched to using <code>\@gls@entry@field</code> 316
<code>\glossentryname:</code> added	<code>\glsgenacfmt:</code> redefined to use accessibility information ... 326
<code>\glsdetoklabel</code> 187	<code>\glsgenentryfmt:</code> redefined to use accessibility information 323
<code>\gls@defglossaryentry:</code> added	<code>\glshyperlink:</code> added <code>\glsdetoklabel</code> 142
<code>\glsdetoklabel</code> 72	<code>\glslocalreset:</code> added
replaced #1 with <code>\@gls@label</code> 73	<code>\glsdetoklabel</code> 80
replaced <code>\ifthenelse</code> with <code>\ifdefequal</code> 74	<code>\glslocalunset:</code> added
<code>\glsadd:</code> added <code>\glsdetoklabel</code> 142	<code>\glsdetoklabel</code> 81
<code>\glsaddkey:</code> switched to using <code>\@gls@field@link</code> 69	<code>\glsmoveentry:</code> added <code>\glsdetoklabel</code> 77
<code>\glsdetoklabel:</code> new 51	replaced <code>\ifthenelse</code> with <code>\ifdefequal</code> 78
<code>\glsdisplaynumberlist:</code> added	<code>\glsrefentry:</code> added <code>\glsdetoklabel</code> 186
<code>\glsdetoklabel</code> 141	<code>\glsreset:</code> added <code>\glsdetoklabel</code> 80
<code>\glsdoifexistsorwarn:</code> new .. 52	<code>\glsseelist:</code> added <code>\expandafter</code> commands 166
<code>\glstentryaccess:</code> switched to using <code>\@gls@entry@field</code> . 316	<code>\glsstepentry:</code> added <code>\glsdetoklabel</code> 185
<code>\glstentrydescaccess:</code> switched to using <code>\@gls@entry@field</code> 316	<code>\glsstepsubentry:</code> added
<code>\glstentrydescpluralaccess:</code> switched to using <code>\@gls@entry@field</code> 316	<code>\glsdetoklabel</code> 185
<code>\glstentryfirstaccess:</code> switched to using <code>\@gls@entry@field</code> 316	<code>\glsunset:</code> added <code>\glsdetoklabel</code> 80
<code>\glstentryfirstplural:</code> added	short-long: commented spurious EOL 205
<code>\glsdetoklabel</code> 138	redefined to use accessibility information 333
<code>\glstentrylongaccess:</code> switched to using <code>\@gls@entry@field</code> 317	short-long-desc: redefined to use accessibility information 335
<code>\glstentrylongpluralaccess:</code> switched to using <code>\@gls@entry@field</code> 317	<code>\ifglsdscsuppressed:</code> added
<code>\glstentrypluralaccess:</code> switched to using <code>\@gls@entry@field</code> 316	<code>\glsdetoklabel</code> 53
<code>\glstentryshortaccess:</code> switched to using <code>\@gls@entry@field</code> 317	fixed typo 53
<code>\glstentryshortpluralaccess:</code> switched to using <code>\@gls@entry@field</code> 317	<code>\ifglstentryexists:</code> added
<code>\glstentrysymbolaccess:</code> switched to using <code>\@gls@entry@field</code> 316	<code>\glsdetoklabel</code> 51
<code>\glstentrysymbolpluralaccess:</code> switched to using <code>\@gls@entry@field</code> 316	<code>\ifglshaschildren:</code> added
	<code>\glsdetoklabel</code> 52
	<code>\ifglshasdesc:</code> added <code>\glsdetoklabel</code> 53
	<code>\ifglshasfield:</code> new 54
	<code>\ifglshaslong:</code> added <code>\glsdetoklabel</code> 53
	<code>\ifglshasparent:</code> added
	<code>\glsdetoklabel</code> 52

<code>\ifglshasshort:</code>	added	<code>\showglofirstpl:</code>	added
<code>\glsdetoklabel</code>	53	<code>\glsdetoklabel</code>	233
<code>\ifglshassymbol:</code>	added	<code>\showglofirstpluralaccess:</code>	
<code>\glsdetoklabel</code>	53	added <code>\glsdetoklabel</code> ...	347
replaced <code>\ifcempty</code> with		<code>\showgloflag:added\glsdetoklabel</code>	
<code>\ifdefempty</code> and replaced		236
<code>\ifx</code> with <code>\ifdefequal</code>	53	<code>\showgloindex:added\glsdetoklabel</code>	
<code>\ifglused:added\glsdetoklabel</code>		236
.....	51	<code>\showglolevel:added\glsdetoklabel</code>	
<code>sm-short-long-desc:</code> redefined		232
to use accessibility informa-		<code>\showglolong:added\glsdetoklabel</code>	
tion	335	236
<code>long-sc-short-desc:</code> redefined		<code>\showglolongaccess:</code> added	
to use accessibility informa-		<code>\glsdetoklabel</code>	348
tion	334	<code>\showglolongpluralaccess:</code>	
<code>long-short:</code> redefined to use ac-		added <code>\glsdetoklabel</code> ...	348
cessibility information	332	<code>\showglongname:added\glsdetoklabel</code>	
<code>long-short-desc:</code> redefined to		235
use accessibility information	334	<code>\showglongnameaccess:</code> added	
<code>long-sm-short-desc:</code> redefined		<code>\glsdetoklabel</code>	346
to use accessibility informa-		<code>\showgloparent:</code> added	
tion	334	<code>\glsdetoklabel</code>	232
<code>footnote:</code> redefined to use acces-		<code>\showgloplural:</code> added	
sibility information	338	<code>\glsdetoklabel</code>	233
<code>footnote-desc:</code> redefined to use		<code>\showglopluralaccess:</code> added	
accessibility information ...	340	<code>\glsdetoklabel</code>	347
<code>footnote-sc:</code> redefined to use ac-		<code>\showgloshort:added\glsdetoklabel</code>	
cessibility information	340	236
<code>footnote-sc-desc:</code> redefined to		<code>\showgloshortaccess:</code> added	
use accessibility information	341	<code>\glsdetoklabel</code>	347
<code>footnote-sm:</code> redefined to use ac-		<code>\showgloshortpluralaccess:</code>	
cessibility information	340	added <code>\glsdetoklabel</code> ...	347
<code>footnote-sm-desc:</code> redefined to		<code>\showglosort:added\glsdetoklabel</code>	
use accessibility information	341	235
<code>\renewacronymstyle:</code> new ...	203	<code>\showglosymbol:</code> added	
<code>\showglocounter:</code> added		<code>\glsdetoklabel</code>	235
<code>\glsdetoklabel</code>	233	<code>\showglosymbolaccess:</code> added	
<code>\showglodesc:added\glsdetoklabel</code>		<code>\glsdetoklabel</code>	347
.....	235	<code>\showglosymbolplural:</code> added	
<code>\showglodescaccess:</code> added		<code>\glsdetoklabel</code>	235
<code>\glsdetoklabel</code>	347	<code>\showglosymbolpluralaccess:</code>	
<code>\showglodescplural:</code> added		added <code>\glsdetoklabel</code> ...	347
<code>\glsdetoklabel</code>	235	<code>\showglotext:added\glsdetoklabel</code>	
<code>\showglodescpluralaccess:</code>		233
added <code>\glsdetoklabel</code> ...	347	<code>\showglotextaccess:</code> added	
<code>\showglofirst:added\glsdetoklabel</code>		<code>\glsdetoklabel</code>	347
.....	233	<code>\showglotype:added\glsdetoklabel</code>	
<code>\showglofirstaccess:</code> added		233
<code>\glsdetoklabel</code>	347		

<code>\showglouserii:added\glsdetoklabel</code> 234	<code>\@gls@getcounterprefix:</code> added warning if no prefix can be formed 165
<code>\showglouserii:</code> added <code>\glsdetoklabel</code> 234	<code>\@gls@getothergrouptitle:</code> new 191
<code>\showglouseriii:</code> added <code>\glsdetoklabel</code> 234	<code>\@gls@noidx@do:new</code> 178
<code>\showglouseriv:</code> added <code>\glsdetoklabel</code> 234	<code>\@gls@noref@warn:new</code> 158
<code>\showglouserv:added\glsdetoklabel</code> 234	<code>\@gls@reference:new</code> 180
<code>\showglouservi:</code> added <code>\glsdetoklabel</code> 234	<code>\@gls@warnonglossdefined:</code> new 17
<code>dua: fixed bug in \acrfullfmt</code> . 209	<code>\@gls@warnontheGLOSSdefined:</code> new 17
fixed bug in <code>\Acrfullplfmt</code> . 209	<code>\@no@makeGLOSSaries:new</code> .. 158
fixed bug in <code>\acrfullplfmt</code> . 209	<code>\@print@GLOSSary:new</code> 170
redefined to use accessibility in- formation 335	<code>\@print@noidx@GLOSSary:new</code> 177
<code>dua-desc: commented spurious</code> EOL 210	<code>\@printGLOSS@setsort:new</code> . 168
redefined to use accessibility in- formation 338	<code>\@printGLOSSary:new</code> 168
4.04	General: added sort key to print- GLOSS group 183
<code>\@@gls@noidx@nosanitizesort:</code> new 19	<code>\compatIBLEGLOSSentry:</code> changed <code>\newcommand</code> to <code>\def</code> as is may or may not be defined 311
<code>\@@gls@noidx@sanitizesort:</code> new 18	<code>\compatIBLESUBGLOSSentry:</code> changed <code>\newcommand</code> to <code>\def</code> as is may or may not be defined 312
<code>\@@gls@nosanitizesort:new</code> . 18	<code>\defGLSdisplayfirst: fixed un-</code> wanted space 92
<code>\@@gls@sanitizesort:new</code> ... 18	<code>\glo@grabfirst:new</code> 177
<code>\@glo@addchildren:new</code> 172	<code>\gls@defGLOSSaryentry:</code> re-
<code>\@glo@do@sortentries:new</code> . 173	placed <code>\ifx</code> with <code>\ifdefvoid</code> 77
<code>\@glo@grabfirst:new</code> 178	<code>\glsnoidxdisplayloc:new</code> .. 180
<code>\@glo@sortedinsert:new</code> ... 173	<code>\glsnoidxdisplaylocLISThandler:</code> new 180
<code>\@glo@sortentries:new</code> 171	<code>\glsnoidxlocLIST:new</code> 179
<code>\@glo@sorthandler@case:new</code> 174	<code>\glsnoidxlocLISThandler:</code> new 179
<code>\@glo@sorthandler@letter:</code> new 174	<code>\glsnoidxstripaccents:new</code> . 19
<code>\@glo@sorthandler@nocase:</code> new 174	<code>alttree: moved hangindent and</code> <code>parindent</code> assignments out-
<code>\@glo@sorthandler@word:new</code> 173	side level test 289
<code>\@glo@sortmacro@case:new</code> . 175	<code>\makeGLOSSaries: Moved def-</code> inition of <code>\glswrite</code> to
<code>\@glo@sortmacro@def:new</code> .. 176	<code>\makeGLOSSaries</code> 153
<code>\@glo@sortmacro@def@do:new</code> 176	<code>\makenoidxGLOSSaries:new</code> . 155
<code>\@glo@sortmacro@letter:new</code> 175	<code>\printGLOSSary: changed to use</code> new <code>\@printGLOSSary</code> ... 168
<code>\@glo@sortmacro@nocase:new</code> 176	<code>\printnoidxGLOSSaries:new</code> 168
<code>\@glo@sortmacro@standard:</code> new 175	
<code>\@glo@sortmacro@use:new</code> .. 176	
<code>\@glo@sortmacro@word:new</code> . 174	

\printnoidxglossary:new .. 168	\@GLSpl:moved\glsifhyper .. 112
\showgloclist:new 236	moved check for first use to
\warn@noprintglossary: Activate warning in \makeglossaries .. 167	\@gls@link 112
\writeist: checked for definition of \glswrite 145, 149	\@Gls@:moved\glsifhyper .. 109
4.06	moved check for first use to
\@GLS@:added\glsifhyper .. 109	\@gls@link 109
\@GLSpl:added\glsifhyper .. 112	\@GLspl@:moved\glsifhyper 111
\@Gls@:added\glsifhyper .. 109	moved check for first use to
\@GLspl@:added\glsifhyper 111	\@gls@link 111
\@gls@:added\glsifhyper .. 108	\@acrlong:added\do@gls@link@checkfirsthyper .. 330
\@gls@numbersdef: added hook to set toc title 28	\@acrshort:added\do@gls@link@checkfirsthyper .. 329
\@gls@symbolsdef: added hook to set toc title 27	\@closegls:new 152
\@glsdisp:added\glsifhyper 113	\@gls@:moved\glsifhyper .. 108
\@GLspl@:added\glsifhyper 110	moved check for first use to
General:added\glsifhyper .. 127–134	\@gls@link 108
acronym:added hook to set toc title 13	\@gls@doautomake:new 26
acronyms:added hook to set toc title 14	\@gls@field@link: added assignment of \do@gls@link@checkfirsthyper .. 113
\glsdefmain:added hook to set toc title 13	\@gls@forbidtexext:new 55
4.07	\@gls@hyp@opt:new 94
\@glossarysection:added optional argument when using unstarred version 39	\@gls@link: removed redundancy 96
\@gls@noidx@do:added\global in case it's used in a tabular-like style 178	renamed \gls@type to \glstype 96
\Acrfullplfmt: fixed no case change bug 200	\@gls@link@checkfirsthyper: new 95
\glsletentryfield:new 134	\@glsdisp:moved\glsifhyper 113
4.08	moved check for first use to
\@ACRlong:added\do@gls@link@checkfirsthyper .. 330	\@gls@link 113
\@ACRshort:added\do@gls@link@checkfirsthyper .. 329	\@GLspl@:moved\glsifhyper 110
\@Acrlong:added\do@gls@link@checkfirsthyper .. 330	moved check for first use to
\@Acrshort:added\do@gls@link@checkfirsthyper .. 329	\@gls@link 110
\@GLS@:moved\glsifhyper .. 109	\@ignored@glossaries:new .. 59
moved check for first use to	General:added entrycounter option to printgloss family .. 182
\@gls@link 109	added \nopostdot option to printgloss family 182
	added \subentrycounter option to printgloss family 182
	explicitly initialise hyper key .. 93
	moved\glsifhyper ... 127–134
	removed\@sACRlongpl 133
	removed\@sAcrlongpl 132
	removed\@sACRlong 131
	removed\@sAcrlong 131

removed \@sacrlong	130	removed \@sGlsuseriv	124
removed \@sACRshortpl ...	130	removed \@sglsuseriv	124
removed \@sAcrshortpl ...	129	removed \@sGLSuseri	122
removed \@sacrshortpl ...	129	removed \@sGlsuseri	122
removed \@sACRshort	128	removed \@sglsuseri	121
removed \@sAcrshort	127	removed \@sGLSuservi	127
removed \@sacrshort	127	removed \@sGlsuservi	126
removed \@sgls@link	95	removed \@sglsuservi	126
removed \@sGLSdescplural	119	removed \@sGLSuserv	126
removed \@sGlsdescplural	119	removed \@sGlsuserv	125
removed \@sglsdescplural	119	removed \@sglsuserv	125
removed \@sGLSdesc	118	removed \@sGLS	109
removed \@sGlsdesc	118	removed \@sGls	108
removed \@sglsdesc	118	removed \@sgls	107
removed \@sglsdisp	112	removed \@thirdofthree (de-	
removed \@sGLSfirstplural	117	fined in kernel)	107
removed \@sGlsfirstplural	116	removed sPGLS	243
removed \@sglsfirstplural	116	removed sPgls	242
removed \@sGLSfirst	115	removed spgls	240
removed \@sGlsfirst	115	removed sPGLSpl	244
removed \@sglsfirst	114	removed sPglspl	242
removed \@sGLSname	118	removed spglspl	241
removed \@sGlsname	117	\ACRfull: removed \s@ACRfull	199
removed \@sglsname	117	switched to using \@gls@hyp@opt	
removed \@sGLSplural	116	199
removed \@sGlsplural	115	\Acrfull: removed \s@Acrfull	198
removed \@sglsplural	115	switched to using \@gls@hyp@opt	
removed \@sGLSpl	112	198
removed \@sGlspl	111	\acrfull: removed \s@acrfull	198
removed \@sglspl	110	switched to using \@gls@hyp@opt	
removed \@sGLSsymbolplural		198
.....	121	\ACRfullpl: removed \s@ACRfullpl	
removed \@sGLssymbolplural		200
.....	121	switched to using \@gls@hyp@opt	
removed \@sglssymbolplural		200
.....	121	\Acrfullpl: removed \s@Acrfullpl	
removed \@sGLSsymbol	120	200
removed \@sGLssymbol	120	switched to using \@gls@hyp@opt	
removed \@sglssymbol	120	200
removed \@sGLStext	114	\acrfullpl: removed \s@acrfullpl	
removed \@sGlstext	114	199
removed \@sglstext	113	switched to using \@gls@hyp@opt	
removed \@sGLSuseriii ...	124	199
removed \@sGlsuseriii ...	124	\ACRlong: switched to using	
removed \@sglsuseriii ...	123	\@gls@hyp@opt	131
removed \@sGLSuserii	123	\Acrlong: switched to using	
removed \@sGlsuserii	123	\@gls@hyp@opt	131
removed \@sglsuserii	122	\acrlong: switched to using	
removed \@sGLSuseriv	125	\@gls@hyp@opt	130

\ACRlongpl: switched to using \@gls@hyp@opt 133	\glsdisp: switched to using \@gls@hyp@opt 112
\Acrlongpl: switched to using \@gls@hyp@opt 133	\glsdohyperlink:new 106
\acrlongpl: switched to using \@gls@hyp@opt 132	\glsdohypertarget:new 106
\ACRshort: switched to using \@gls@hyp@opt 128	\glsenablehyper: added \KV@glslink@hypertrue to definition 107
\Acrshort: switched to using \@gls@hyp@opt 127	\GLSfirst: switched to using \@gls@hyp@opt 115
\acrshort: switched to using \@gls@hyp@opt 127	\Glsfirst: switched to using \@gls@hyp@opt 114
\ACRshortpl: switched to using \@gls@hyp@opt 130	\glsfirst: switched to using \@gls@hyp@opt 114
\Acrshortpl: switched to using \@gls@hyp@opt 129	\GLSfirstplural: switched to using \@gls@hyp@opt 117
\acrshortpl: switched to using \@gls@hyp@opt 128	\Glsfirstplural: switched to using \@gls@hyp@opt 116
\forallacronyms: new 50	\glsfirstplural: switched to using \@gls@hyp@opt 116
\GLS: switched to using \@gls@hyp@opt 109	\glsifhyper: deprecated 94
\Gls: switched to using \@gls@hyp@opt 108	\glslink: switched to using \@gls@hyp@opt 95
\gls: switched to using \@gls@hyp@opt 107	\glslinkcheckfirsthyperhook: new 96
\gls@defglossaryentry: added check for ignored glossary ... 73	\glslinkvar:new 94
\gls@istfilebase: new 34	\GLSname: switched to using \@gls@hyp@opt 117
\glsaddkey: removed \@sGLS@user@<key> 71	\Glsname: switched to using \@gls@hyp@opt 117
removed \@sGls@user@<key> . 70	\glsname: switched to using \@gls@hyp@opt 117
removed \@sgls@user@<key> . 70	\GLSpl: switched to using \@gls@hyp@opt 112
switched to using \@gls@hyp@opt 70, 71	\Glspl: switched to using \@gls@hyp@opt 111
\GLSdesc: switched to using \@gls@hyp@opt 118	\glspl: switched to using \@gls@hyp@opt 110
\Glsdesc: switched to using \@gls@hyp@opt 118	\GLSplural: switched to using \@gls@hyp@opt 116
\glsdesc: switched to using \@gls@hyp@opt 118	\Glsplural: switched to using \@gls@hyp@opt 115
\GLSdescplural: switched to us- ing \@gls@hyp@opt 119	\glsplural: switched to using \@gls@hyp@opt 115
\Glsdescplural: switched to us- ing \@gls@hyp@opt 119	\glsspace: new 198
\glsdescplural: switched to us- ing \@gls@hyp@opt 119	\GLSsymbol: switched to using \@gls@hyp@opt 120
\glsdisablehyper: added \KV@glslink@hyperfalse to definition 107	\Glsymbol: switched to using \@gls@hyp@opt 120

\glssymbol: switched to using \@gls@hyp@opt 120	\glsuservi: switched to using \@gls@hyp@opt 126
\GLSsymbolplural: switched to using \@gls@hyp@opt 121	\ifignoredglossary:new 59
\Glssymbolplural: switched to using \@gls@hyp@opt 121	altlongragged4col: fixed bug that displayed description in- stead of symbol 265
\glssymbolplural: switched to using \@gls@hyp@opt 120	\newglossary: added starred ver- sion 56
\GLStext: switched to using \@gls@hyp@opt 114	\newignoredglossary:new ... 58
\Glstext: switched to using \@gls@hyp@opt 114	\ns@newglossary: added \@glotype@<name>@log ... 57
\glstext: switched to using \@gls@hyp@opt 113	new 56
\glstreenamfmt: new 284	\p@gls@hyp@opt:new 94
\GLSuseri: switched to using \@gls@hyp@opt 122	\PGLS: changed to use \@gls@hyp@opt 243
\Glsuseri: switched to using \@gls@hyp@opt 122	\Pgls: changed to use \@gls@hyp@opt 242
\glsuseri: switched to using \@gls@hyp@opt 121	\pgls: changed to use \@gls@hyp@opt 240
\GLSuserii: switched to using \@gls@hyp@opt 123	\PGLSpl: changed to use \@gls@hyp@opt 244
\Glsuserii: switched to using \@gls@hyp@opt 123	\Pglspl: changed to use \@gls@hyp@opt 242
\glsuserii: switched to using \@gls@hyp@opt 122	\pglspl: changed to use \@gls@hyp@opt 241
\GLSuseriii: switched to using \@gls@hyp@opt 124	\s@gls@hyp@opt:new 94
\Glsuseriii: switched to using \@gls@hyp@opt 123	\s@newglossary:new 56
\glsuseriii: switched to using \@gls@hyp@opt 123	automake:new 26
\GLSuseriv: switched to using \@gls@hyp@opt 125	4.09 \glsaddkey: fixed bug in user commands 70
\Glsuseriv: switched to using \@gls@hyp@opt 124	4.10
\glsuseriv: switched to using \@gls@hyp@opt 124	\@Gls@acentryname:new ... 135
\GLSuserv: switched to using \@gls@hyp@opt 126	\@Gls@entryname:new 135
\Glsuserv: switched to using \@gls@hyp@opt 125	\@gls@glossary: Renamed \@glossary to \@gls@glossary 160
\GLSuservi: switched to using \@gls@hyp@opt 126	\glpercentchar:new 144
\Glsuservi: switched to using \@gls@hyp@opt 126	\glstildechar:new 144
	alttree: moved space after sym- bol 289, 290
	4.11
	\@do@wrglossary: added hook 163
	sanitize: none option 21
	\gls@wrglossary: renamed from \@wrglossary to \gls@wrglossary 161
	\glsaddprotectedpagefmt: new 162

\glsbackslash:new	144	\glsadd: added check for vertical	
4.12		mode	142
\@gls@addpredefinedattributes:		\glsaddallunused: replaced	
Added glsignore attribute ...	43	@gobble with glsignore	143
\@gls@adjustmode:new	143	\glsifusedtranslatordict:	
\@gls@notranslatorhook: re-		new	22
moved	21	\glsignore:new	143
\@gls@toc: added \protect to		\glsupacrpluralsuffix:new .	31
\numberline	40	\ProvidesGlossariesLang:	
\@gls@usetranslator:new ...	22	new	31
\glsacrpluralsuffix:new ...	31	\RequireGlossariesLang:new	31

Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Symbols	
\@do@wrglossary	164
\@do@wrglossary	163
\@glo@assign@sortkey	184
\@glossarysec	6
\@glossaryseclabel	6
\@glossarysecstar	6
\@gls@default@entryfmt	319
\@gls@expand@field	65
\@gls@fixbraces	166
\@gls@noidx@nosanitizesort .	19
\@gls@noidx@sanitizesort ...	18
\@gls@nosanitizesort	18
\@gls@sanitizesort	18
\@ACRlong	330
\@ACRshort	329
\@Acrlong	330
\@Acrshort	329
\@GLS@	109
\@GLSpl	112
\@Gls@	108
\@Gls@acrentryname	135
\@Gls@entry@field	134
\@Gls@entryname	135
\@Glspl@	111
\@PGLS	243
\@PGLS@	243
\@PGLSpl	244
\@PGLSpl@	244
\@PglS	242
\@PglS@	242
\@PglSpl	242
\@PglSpl@	243
\@acrfull	198
\@acrlong	330
\@acrshort	329
\@addtoacronymlists	14
\@closegls	152
\@delimN	194
\@delimR	194
\@disable@onlypremakeg	30
\@disable@premakecs	30
\@disabled@glsaddxdycounters	42
\@do@seeglossary	165
\@do@wrglossary	161, 293
\@glo@addchildren	172
\@glo@default@sorttype	10
\@glo@do@sortentries	173
\@glo@grabfirst	178
\@glo@no@assign@sortkey	183
\@glo@seeautonumberlist	8
\@glo@sortedinsert	173
\@glo@sortentries	171
\@glo@sorthandler@case	174
\@glo@sorthandler@letter ...	174
\@glo@sorthandler@nocase ...	174
\@glo@sorthandler@word	173
\@glo@sortmacro@case	175
\@glo@sortmacro@def	176
\@glo@sortmacro@def@do	176
\@glo@sortmacro@letter	175
\@glo@sortmacro@nocase	176
\@glo@sortmacro@standard ...	175
\@glo@sortmacro@use	176
\@glo@sortmacro@word	174
\@glo@storeentry	78
\@glo@types	55
\@glossary@default@style	7
\@glossaryentryfield	78
\@glossarysection	39
\@glossarysubentryfield	78
\@gls	107
\@gls@	108
\@gls@@link	95
\@gls@access@display	317
\@gls@addpredefinedattributes	43
\@gls@adjustmode	143
\@gls@assign@symbol@field ...	18
\@gls@assign@symbolplural@field	18
\@gls@checkactual	104
\@gls@checkbar	103
\@gls@checkescactual	101
\@gls@checkescbar	102
\@gls@checkesclevel	103

\@gls@checkescquote	101	\@gls@sanitizename	18
\@gls@checklevel	104	\@gls@sanitizesort	18
\@gls@checkmkidxchars	100	\@gls@sanitizesymbol	18
\@gls@checkquote	100	\@gls@saveentrycounter	97
\@gls@codepage	48	\@gls@setacrstyle	24
\@gls@counterwithin	10	\@gls@setcounter	58
\@gls@declareoption	7	\@gls@setupsort@def	11
\@gls@default@value	60	\@gls@setupsort@standard	10
\@gls@do@acronymsdef	14	\@gls@setupsort@use	11
\@gls@doautomake	26	\@gls@startswithexpandonce ..	65
\@gls@entry@field	134	\@gls@symbolsdef	27
\@gls@escbsdq	99	\@gls@tmpb	100
\@gls@expand@fields	65	\@gls@toc	40
\@gls@fetchfield	68	\@gls@updatechecked	100
\@gls@field@link	113	\@gls@usetranslator	22
\@gls@fixbraces	166	\@gls@warnonglossdefined	17
\@gls@forbidtexext	55	\@gls@warnonthehglossdefined .	17
\@gls@getcounter	58	\@gls@writedef	67
\@gls@getcounterprefix	164	\@gls@xdy@Lclass@Alpha-page-numbers	
\@gls@getgrouptitle	191	45
\@gls@getothergrouptitle ...	191	\@gls@xdy@Lclass@Appendix-page-numbers	
\@gls@glossary	160	45
\@gls@hyp@opt	94	\@gls@xdy@Lclass@Roman-page-numbers	
\@gls@hypergroup	249	45
\@gls@ifinlist	41	\@gls@xdy@Lclass@alpha-page-numbers	
\@gls@indexdef	28	45
\@gls@keymap	68, 238	\@gls@xdy@Lclass@arabic-page-numbers	
\@gls@link	96	45
\@gls@link@checkfirsthyper ..	95	\@gls@xdy@Lclass@arabic-section-numbers	
\@gls@loadlist	8	45
\@gls@loadlong	8	\@gls@xdy@Lclass@roman-page-numbers	
\@gls@loadsuper	8	44
\@gls@loadtree	8	\@gls@xdy@locationlist	44
\@gls@makefirsttuc	245	\@gls@xdy@checkbackslash	105
\@gls@missingnumberlist	64	\@gls@xdy@checkquote	105
\@gls@noaccess	314	\@gls@Alphacompositor	35, 45
\@gls@noexpand@field	64	\@gls@acronymlists	14
\@gls@noexpand@fields	64	\@gls@defaultplural	63
\@gls@nohyperlist	16	\@gls@defaultsort	64
\@gls@noidx@do	178	\@gls@disp	112
\@gls@noidx@setsanitizesort .	21	\@gls@firstletter	144
\@gls@noref@warn	158	\@gls@hypernumber	194
\@gls@notranslatorhook	21	\@gls@link	106
\@gls@numbersdef	27	\@gls@minrange	145
\@gls@onlypremakeg	29	\@gls@nextpages	184
\@gls@provide@newglossary ...	55	\@gls@nodesc	63
\@gls@reference	180	\@gls@noname	63
\@gls@renewglossary	161	\@gls@nonextpages	184
\@gls@sanitizedesc	17	\@gls@openfile	151

\@glsorder	25
\@glspl@	110
\@glstarget	106
\@glswidestname	288
\@glswritefiles	159
\@ignored@glossaries	59
\@istfilename	34
\@makeglossary	151
\@mfu@nocaplist	247
\@newglossary	58
\@newglossaryentryposthook ..	77
\@newglossaryentryprehook ...	77
\@no@makeglossaries	158
\@no@post@desc	34
\@nopostdesc	34
\@onlypremakeg	30
\@p@glossarysection	39
\@pgls	241
\@pgls@	241
\@pglspl	241
\@pglspl@	241
\@print@glossary	170
\@print@noidx@glossary	177
\@printgloss@setsort	168
\@printglossary	168
\@set@glo@numformat	98, 293
\@wrglossary@pageformat	162
\@wrglossarynumberhook	162
\@xdy@main@language	25
\@xdy@attributelist	41
\@xdy@attributes	40
\@xdy@language	48
\@xdylettergroups	49
\@xdylocationclassorder	46
\@xdylocref	41
\@xdyrequiredstyles	47
\@xdysortrules	47
\@xdyuseralphabets	44
\@xdyuserlocationdefs	45
\@xdyuserlocationnames	45

A

\Ac	214
\ac	214
access (key)	312
accsupp package	311
\accsuppglossaryentryfield .	332
\accsuppglossarysubentryfield	
.....	332

\Acf	214
\acf	214
\Acfp	214
\acfp	214
\Acl	213
\acl	213
\Aclp	214
\aclp	213
\Acp	214
\acp	214
\acrfootnote	216
\ACRfull	199
\Acrfull	198
\acrfull	198, 202, 211, 339
\ACRfullfmt	199
\Acrfullfmt	199
\acrfullfmt	198
\acrfullformat	89, 198
\ACRfullpl	200
\Acrfullpl	200
\acrfullpl	199
\ACRfullplfmt	200
\Acrfullplfmt	200
\acrfullplfmt	200
\acrlinkfootnote	216
\acrlinkfullformat	198
\ACRlong	131
\Acrlong	131
\acrlong	130
\ACRlongpl	133
\Acrlongpl	133
\acrlongpl	132
\acrnameformat	201, 222
\acrlinkfootnote	216
acronym (option)	13
acronym styles:	
dua	208, 335
dua-desc	210, 338
footnote	210, 338
footnote-desc	212, 340
footnote-sc	211, 340
footnote-sc-desc	212, 341
footnote-sm	212, 340
footnote-sm-desc	212, 341
long-sc-short	205
long-sc-short-desc ..	206, 334
long-short	204, 332
long-short-desc	206, 334
long-sm-short	205

long-sm-short-desc ..	207, 334	altsuperragged4colborder	
sc-short-long	206	(style)	283
sc-short-long-desc ..	207, 335	altsuperragged4colheader	
short-long	205, 333	(style)	282
short-long-desc	207, 335	altsuperragged4colheaderborder	
sm-short-long	206	(style)	283
sm-short-long-desc ..	208, 335	alttree (style)	288
\acronymentry	202	alttreegroup (style)	291
\acronymfont		alttreehypergroup (style)	291
.....	89, 201, 218, 222, 225, 228	amsgen package	4, 93
acronymlists (option)	15	amsmath package	80
\acronymname	30	\andname	31
acronyms (option)	14	array package	262, 278
\acronymsort	203	article class	164
\acronymtype	13, 197	automake (option)	26
\acrpluralsuffix	197		
\ACRshort	128	B	
\Acrshort	127	babel package	22, 30, 32, 48
\acrshort	127		
\ACRshortpl	130	C	
\Acrshortpl	129	\capitalisewords	246
\acrshortpl	128	\compatglossarystyle	298
\Acs	213	compatible-2.07 (option)	27
\acs	213	compatible-3.07 (option)	27
\Acsp	213	\compatibleglossentry ..	187, 311
\acsp	213	\compatiblesubglossentry	188, 312
\addglossarytocaptions	32	counter (key)	61
\addto	32	counter (option)	16
align (environment)	80, 97	\CustomAcronymFields	230
altlist (style)	254	\CustomNewAcronymDef	231
altlistgroup (style)	254		
altlisthypergroup (style)	254	D	
altlong4col (style)	260	datatool package	173
altlong4colborder (style)	261	\DeclareAcronymList	14
altlong4colheader (style)	261	\DefaultNewAcronymDef ..	215, 341
altlong4colheaderborder (style)	261	\defentryfmt	93
altlongragged4col (style)	265	\defglstdisplay	92
altlongragged4colborder (style)	266	\defglstdisplayfirst	92
altlongragged4colheader (style)	266	\defglssentry	57
altlongragged4colheaderborder		\defglssentryfmt	55, 59, 61, 83
(style)	267	\DefineAcronymSynonyms	213
\altnewglossary	57	\delimN	36, 193
altsuper4col (style)	276	\delimR	36, 193
altsuper4colborder (style)	277	description (environment)	252, 253
altsuper4colheader (style)	277	description (key)	59
altsuper4colheaderborder		description (option)	24
(style)	277	descriptionaccess (key)	313
altsuperragged4col (style)	282	\DescriptionDUANewAcronymDef	219
		\DescriptionFootnoteNewAcronymDef	
		217, 343

`\descriptionname` 31
`\DescriptionNewAcronymDef` ..
 221, 343
`descriptionplural (key)` 60
`descriptionpluralaccess (key)` 313
`\detokenize` 51
`doc package` 4, 5, 12
`dua (acrstyle)` 208, 335
`dua (option)` 24
`dua-desc (acrstyle)` 210, 338
`\DUANewAcronymDef` 228

E

`entrycounter (option)` 9
`entrycounterwithin (option)` 9
`\entryname` 31
 environments:
 `align` 80, 97
 `description` 252, 253
 `longtable` 8, 232, 255–267
 `multicols` 267
 `supertabular` ... 8, 232, 271–283
 `theglossary` ... 5, 17, 36, 37,
 186, 187, 193, 269, 270, 286–289
 `theindex` 284
`equation (counter)` 97, 98
`etoolbox package` 4, 244

F

file types
 `.aux` 170
 `.glo` 78
 `.ist` 144, 150, 151
 `.toc` 40
 `.xdy` 34
 `glo` 238
`\findrootlanguage` 48
`first (key)` 60
`firstaccess (key)` 313
`\firstacronymfont` 89, 201
`firstplural (key)` 60
`firstpluralaccess (key)` 313
`footnote (acrstyle)` 210, 338
`footnote (option)` 24
`footnote-desc (acrstyle)` .. 212, 340
`footnote-sc (acrstyle)` 211, 340
`footnote-sc-desc (acrstyle)` 212, 341
`footnote-sm (acrstyle)` 212, 340
`footnote-sm-desc (acrstyle)` 212, 341
`\FootnoteNewAcronymDef` . 223, 344

`\forallacronyms` 50
`\forallglossaries` 49
`\forallglentries` 50
`\forallglentries` 50

G

`garamondx package` 197
`\Genacrfullformat` 91, 328
`\genacrfullformat` 91, 328
`\GenericAcronymFields` 202
`\Genplacrfullformat` 91, 328
`\genplacrfullformat` 91, 328
`\glo@grabfirst` 177
`\glolinkprefix` 97
`glossareentry (counter)` 185
`glossaries package` 28,
 48, 145, 232, 238, 252, 291, 311
`glossaries-accsupp package` ... 78, 311
`\GlossariesWarning` 16
`\GlossariesWarningNoLine` 16
`\glossary` 56, 151, 160, 192
 glossary counters:
 `glossaryentry` 184
 `glossarysubentry` 185
 glossary keys:
 `access` 312
 `counter` 61
 `description` 59
 `descriptionaccess` 313
 `descriptionplural` 60
 `descriptionpluralaccess` . 313
 `first` 60
 `firstaccess` 313
 `firstplural` 60
 `firstpluralaccess` 313
 `long` 63
 `longaccess` 313
 `longplural` 63
 `longpluralaccess` 314
 `name` 59
 `nonumberlist` 62
 `parent` 62
 `plural` 60
 `pluralaccess` 313
 `see` 61
 `short` 63
 `shortaccess` 313
 `shortplural` 63
 `shortpluralaccess` 313

sort	60	altsuper4colheaderborder	
symbol	61	277, 311
symbolaccess	313	altsuper4colheaderborder	277
symbolplural	61	altsuperragged4col	282, 283, 309
symbolpluralaccess	313	altsuperragged4col 282
text	60	altsuperragged4colborder	
textaccess	312	283, 309
type	61	altsuperragged4colborder	283
user1	62	altsuperragged4colheader	
user2	62	282, 309
user3	62	altsuperragged4colheader	282
user4	62	altsuperragged4colheaderborder	
user5	62	283, 309
user6	63	altsuperragged4colheaderborder	
glossary package	1, 196	283
glossary styles:		alttree	270, 288, 291, 305
altlist	254, 299	alttree	288
altlist	254	alttreegroup	291, 307
altlistgroup	254, 300	alttreegroup	291
altlistgroup	254	alttreehypergroup ...	291, 307
altlisthypergroup ...	254, 300	alttreehypergroup	291
altlisthypergroup	254	index	267, 284, 285, 303
altlong4col ..	260, 261, 265, 302	index	284
altlong4col	260	indexgroup	285, 304
altlong4colborder ...	261, 302	indexgroup	285
altlong4colborder	261	indexhypergroup	285, 304
altlong4colheader ...	261, 302	indexhypergroup	285
altlong4colheader	261	inline	298
altlong4colheaderborder .		inline	250
.....	261, 302	list	7, 252–255, 299
altlong4colheaderborder .	261	list	252
altlongragged4col	265, 266, 303	listdotted	255, 300
altlongragged4col	265	listdotted	255
altlongragged4colborder .		listgroup	253, 299
.....	266, 303	listgroup	253
altlongragged4colborder .	266	listhypergroup	253, 299
altlongragged4colheader .		listhypergroup	253
.....	266, 303	long	256, 257, 262, 300, 302
altlongragged4colheader .	266	long	256
altlongragged4colheaderborder		long3col	257, 258, 301
.....	267, 303	long3col	257
altlongragged4colheaderborder		long3colborder	258, 301
.....	267	long3colborder	258
altsuper4col .	276, 277, 282, 311	long3colheader	258, 301
altsuper4col	276	long3colheader	258
altsuper4colborder ..	277, 311	long3colheaderborder	258, 301
altsuper4colborder	277	long3colheaderborder ...	258
altsuper4colheader ..	277, 311	long4col	259, 260, 301
altsuper4colheader	277	long4col	259

long4colborder	260, 301	mcoltreenoname	269
long4colborder	260	mcoltreenonamegroup .	269, 307
long4colheader	259, 301	mcoltreenonamegroup	269
long4colheader	259	mcoltreenonamehypergroup	
long4colheaderborder	260, 301	269, 307
long4colheaderborder	260	mcoltreenonamehypergroup	269
longborder	256, 300	sublistdotted	300
longborder	256	sublistdotted	255
longheader	257, 300	super	271–273, 279, 309
longheader	257	super	271
longheaderborder	257, 300	super3col	273, 274, 310
longheaderborder	257	super3col	273
longragged	262–264	super3colborder	274, 310
longragged	262	super3colborder	274
longragged3col ...	264, 265, 302	super3colheader	274, 310
longragged3col	264	super3colheader	274
longragged3colborder	264, 303	super3colheaderborder	274, 310
longragged3colborder	264	super3colheaderborder ...	274
longragged3colheader	265, 303	super4col	275, 276, 310
longragged3colheader	265	super4col	275
longragged3colheaderborder		super4colborder	276, 311
.....	265, 303	super4colborder	276
longragged3colheaderborder		super4colheader	275, 310
.....	265	super4colheader	275
longraggedborder	263, 302	super4colheaderborder	276, 311
longraggedborder	263	super4colheaderborder ...	276
longraggedheader	263, 302	superborder	272, 309
longraggedheader	263	superborder	272
longraggedheaderborder	263, 302	superheader	272, 310
longraggedheaderborder ..	263	superheader	272
mcotalttree	270, 308	superheaderborder ...	273, 310
mcotalttree	270	superheaderborder	273
mcotalttreegroup	270, 308	superragged	278, 280, 308
mcotalttreegroup	270	superragged	278
mcotalttreehypergroup	270, 308	superragged3col ..	280, 281, 308
mcotalttreehypergroup ...	270	superragged3col	280
mcolindex	268, 307	superragged3colborder	281, 309
mcolindex	267	superragged3colborder ...	281
mcolindexgroup	268, 307	superragged3colheader	281, 309
mcolindexgroup	268	superragged3colheader ...	281
mcolindexhypergroup .	268, 307	superragged3colheaderborder	
mcolindexhypergroup	268	281, 309
mcoltree	268, 307	superraggedborder ...	279, 308
mcoltree	268	superraggedborder	279
mcoltreegroup	307	superraggedheader ...	279, 308
mcoltreegroup	268	superraggedheader	279
mcoltreehypergroup ..	269, 307	superraggedheaderborder .	
mcoltreehypergroup	269	280, 308
mcoltreenoname	269, 307	superraggedheaderborder .	280

superraggedright3colheaderborder	\Gls	108, 111, 244
..... 281	\gls	4, 61, 82, 93, 107, 109, 110, 113–126, 186, 240
tree	\gls@Alphpage	162
..... 268, 285–288, 304	\gls@alphpage	162
tree	\gls@assign@desc	71
..... 285	\gls@assign@desc@field	18
treegroup	\gls@assign@descplural@field	18
..... 269, 286, 305	\gls@assign@field	66
treegroup	\gls@assign@name@field	18
..... 286	\gls@assign@type@field	17
treehypergroup	\gls@checkisacronymlist	15
..... 286, 305	\gls@checkseeallowed	61
treehypergroup	\gls@codepage	25
..... 286	\gls@defglossaryentry	72
treenoname	\gls@disablepagerefexpansion	161
... 269, 287, 288, 305	\gls@doclearpage	39
treenoname	\gls@doentryfmt	55
..... 287	\gls@glossary	160
treenonamegroup	\gls@hypergroup prerun	249
..... 288, 305	\gls@ifnotmeasuring	80
treenonamegroup	\gls@istfilebase	34
..... 288	\gls@level	64
treenonamehypergroup	\gls@noidxglossary	159
... 288	\gls@numberpage	162
glossary-hypernav package	\gls@protected@pagefmts	161
..... 144	\gls@Romanpage	162
glossary-list package	\gls@romanpage	162
..... 7, 8, 252	\gls@save@numberlist	167
glossary-long package	\gls@suffixF	35
..... 8, 255, 256, 265, 271	\gls@suffixFF	36
glossary-longragged package	\gls@wrglossary	161
..... 262	\glsaccessdisplay	319
glossary-mcols package	\glsaccsupp	317
..... 267	\glsacrpluralsuffix	31
glossary-super package	\glsadd	82, 142, 192
..... 8, 256, 271, 278, 282	\glsadd options	
glossary-superragged package	counter	142
..... 278	format	142, 193
glossary-tree package	\glsaddall	51, 82, 143
..... 8, 284	\glsaddall options	
glossaryentry (counter)	types	142, 143
..... 9, 185, 186	\glsaddallunused	143
glossaryentry (counter)	\glsaddkey	69
..... 184	\GlsAddLetterGroup	49
\glossaryentryfield	\glsaddprotectedpagefmt	162
..... 189, 193	\GlsAddSortRule	47
\glossaryentrynumber	\GlsAddXdyAlphabet	44
..... 184	\GlsAddXdyAttribute	42, 291
\glossaryentrynumbers		
..... 7, 36, 169, 170		
\glossaryheader		
..... 187, 193		
\glossarymark		
..... 38		
\glossaryname		
..... 30, 32		
\glossarypostamble		
..... 37, 193		
\glossarypreamble		
..... 37, 193		
\glossarysection		
..... 6, 37, 56		
\glossarystyle		
..... 192, 232		
glossarysubentry (counter)		
..... 10, 185, 186		
glossarysubentry (counter)		
..... 185		
\glossarysubentryfield		
..... 189		
\glossentry		
..... 61, 187		
\Glossentrydesc		
..... 188		
\glossentrydesc		
..... 188, 331		
\Glossentryname		
..... 188		
\glossentryname		
..... 187, 331		
\Glossentrysymbol		
..... 188		
\glossentrysymbol		
..... 188, 331		
\GLS		
..... 109		

<code>\GlsAddXdyCounters</code>	41, 292	<code>\glstentryfirstpluralaccess</code> .	316
<code>\GlsAddXdyLocation</code>	46, 292	<code>\glstentryfmt</code>	59, 61, 83
<code>\GlsAddXdyStyle</code>	47	<code>\Glstentryfull</code>	140
<code>\glsclearprefix</code>	6	<code>\glstentryfull</code> ...	140, 202, 211, 340
<code>\glslacksbackslash</code>	144	<code>\Glstentryfullpl</code>	141
<code>\glsclearpage</code>	40	<code>\glstentryfullpl</code>	140
<code>\glsclosebrace</code>	144	<code>\glstentryitem</code>	186
<code>\glscpositor</code>	35, 45	<code>\Glstentrylong</code>	140
<code>\glscounter</code>	16, 57	<code>\glstentrylong</code>	140
<code>\GlsDeclareNoHyperList</code>	16	<code>\glstentrylongaccess</code>	317
<code>\glstdefaulttype</code>	13	<code>\Glstentrylongpl</code>	140
<code>\glstdefmain</code>	12	<code>\glstentrylongpl</code>	140
<code>\GLSdesc</code>	118	<code>\glstentrylongpluralaccess</code> ..	317
<code>\Glsdesc</code>	118	<code>\Glstentryname</code>	135
<code>\glstdesc</code>	118	<code>\glstentryname</code>	135, 167
<code>\GLSdescplural</code>	119	<code>\glstentrynumberlist</code>	141, 156
<code>\Glsdescplural</code>	119	<code>\Glstentryplural</code>	137
<code>\glstdescplural</code>	119	<code>\glstentryplural</code>	137
<code>\glstdescriptionaccessdisplay</code>	318	<code>\glstentrypluralaccess</code>	316
<code>\glstdescriptionpluralaccessdisplay</code>	318	<code>\Glstentryprefix</code>	240
<code>\glstdescwidth</code> ...	256, 262, 271, 278	<code>\glstentryprefix</code>	239
<code>\glstdetoklabel</code>	51	<code>\Glstentryprefixfirst</code>	239
<code>\glstdisablehyper</code>	107	<code>\glstentryprefixfirst</code>	239
<code>\glstdisp</code>	112	<code>\Glstentryprefixfirstplural</code> .	239
<code>\glstdisplay</code>	82, 92, 107	<code>\glstentryprefixfirstplural</code> .	239
<code>\glstdisplayfirst</code>	82, 91, 107	<code>\Glstentryprefixplural</code>	240
<code>\glstdisplaynumberlist</code>	141, 157, 180	<code>\glstentryprefixplural</code>	239
<code>\glstdohyperlink</code>	106	<code>\Glstentryshort</code>	140
<code>\glstdohypertarget</code>	106	<code>\glstentryshort</code>	139
<code>\glstdoifexists</code>	51	<code>\glstentryshortaccess</code>	317
<code>\glstdoifexistsorwarn</code>	52	<code>\Glstentryshortpl</code>	140
<code>\glstdoifnoexists</code>	51	<code>\glstentryshortpl</code>	140
<code>\glstdoparenifnotempty</code>	225	<code>\glstentryshortpluralaccess</code> .	317
<code>\glstenablehyper</code>	107	<code>\glstentrysort</code>	138
<code>\glstentryaccess</code>	316	<code>\Glstentrysymbol</code>	137
<code>\glstentrycounter</code>	97	<code>\glstentrysymbol</code>	137
<code>\glstentrycounterlabel</code>	186	<code>\glstentrysymbolaccess</code>	316
<code>\Glstentrydesc</code>	136	<code>\Glstentrysymbolplural</code>	137
<code>\glstentrydesc</code>	136	<code>\glstentrysymbolplural</code>	137
<code>\glstentrydescaccess</code>	316	<code>\glstentrysymbolpluralaccess</code>	316
<code>\Glstentrydescplural</code>	136	<code>\Glstentrytext</code>	136
<code>\glstentrydescplural</code>	136	<code>\glstentrytext</code>	50, 136, 167
<code>\glstentrydescpluralaccess</code> ..	316	<code>\glstentrytextaccess</code>	316
<code>\Glstentryfirst</code>	137	<code>\glstentrytype</code>	138
<code>\glstentryfirst</code>	137	<code>\Glstentryuseri</code>	138
<code>\glstentryfirstaccess</code>	316	<code>\glstentryuseri</code>	138
<code>\Glstentryfirstplural</code>	138	<code>\Glstentryuserii</code>	138
<code>\glstentryfirstplural</code>	138	<code>\glstentryuserii</code>	138
		<code>\Glstentryuseriii</code>	139

<code>\glsentryuseriii</code>	139	<code>hyper</code>	93, 95, 107
<code>\Glsentryuseriv</code>	139	<code>local</code>	93
<code>\glsentryuseriv</code>	139	<code>\glslinkcheckfirsthyperhook</code> .	96
<code>\Glsentryuserv</code>	139	<code>\glslinkvar</code>	94
<code>\glsentryuserv</code>	139	<code>\glslistdottedwidth</code>	255
<code>\Glsentryuservi</code>	139	<code>\glslocalreset</code>	80
<code>\glsentryuservi</code>	139	<code>\glslocalresetall</code>	81
<code>\glsexpandfields</code>	66	<code>\glslocalunset</code>	81
<code>\GLSfirst</code>	115	<code>\glslocalunsetall</code>	82
<code>\Glsfirst</code>	114, 115	<code>\glslongaccessdisplay</code>	319
<code>\glsfirst</code>	114	<code>\glslongaccesskey</code>	346
<code>\glsfirstaccessdisplay</code>	318	<code>\glslongkey</code>	197
<code>\GLSfirstplural</code>	117	<code>\glslongpluralaccessdisplay</code>	319
<code>\Glsfirstplural</code>	116	<code>\glslongpluralaccesskey</code>	346
<code>\glsfirstplural</code>	116, 117	<code>\glslongpluralkey</code>	198
<code>\glsfirstpluralaccessdisplay</code>	318	<code>\glslongtok</code>	201
<code>\glsgenacfmt</code>	89, 326	<code>\glsmakefirstuc</code>	246
<code>\glsgenentryfmt</code>	86, 323	<code>\glsmcols</code>	267
<code>\glsgetgrouplabel</code>	191	<code>\glsmoveentry</code>	77
<code>\glsgetgrouptitle</code>	144, 191	<code>\GLSname</code>	117
<code>\glsglossarymark</code>	9, 38	<code>\Glsname</code>	117
<code>\glsgroupheading</code>	190, 193	<code>\glsname</code>	117
<code>\glsgroupskip</code>	190, 193, 252	<code>\glsnameaccessdisplay</code>	317
<code>\glshyperlink</code>	142	<code>\glsnamefont</code>	193, 284
<code>\glshypernavsep</code>	250	<code>\glsnavhyperlink</code>	248
<code>\glshypernumber</code>	36, 194	<code>\glsnavhypertarget</code>	248
<code>\glsifhyper</code>	94	<code>\glsnavigation</code>	249
<code>\glsifhyperon</code>	94	<code>\glsnextpages</code>	184
<code>\glsIfListOfAcronyms</code>	15	<code>\glsnoexpandfields</code>	66
<code>\glsifusedtranslatordict</code>	22	<code>\glsnoidxdisplayloc</code>	180
<code>\glsignore</code>	143	<code>\glsnoidxdisplayloclisthandler</code>	
<code>\glsinlinedescformat</code>	252	180
<code>\glsinlinedopostchild</code> ..	250, 251	<code>\glsnoidxloclist</code>	179
<code>\glsinlineemptydescformat</code> ..	252	<code>\glsnoidxloclisthandler</code>	179
<code>\glsinlinenameformat</code>	252	<code>\glsnoidxnumberlistloophandler</code>	
<code>\glsinlineparentchildseparator</code>		158
.....	252	<code>\glsnoidxstripaccents</code>	19
<code>\glsinlinepostchild</code>	252	<code>\glsnonextpages</code>	184
<code>\glsinlineseparator</code>	252	<code>\glsnoxindywarning</code>	40
<code>\glsinlinesubdescformat</code>	252	<code>\glsnumberformat</code>	36
<code>\glsinlinesubnameformat</code>	252	<code>\glsnumberlistloop</code>	158
<code>\glsinlinesubseparator</code>	252	<code>\glsnumbersgroupname</code> .	31, 144, 190
<code>\glskeylisttok</code>	201	<code>\glsnumlistlastsep</code>	142
<code>\glslabeltok</code>	201	<code>\glsnumlistsep</code>	142
<code>\glsletentryfield</code>	134	<code>\glsopenbrace</code>	144
<code>\glslink</code> ...	82, 95, 107, 142, 192, 193	<code>\glsorder</code>	24
<code>\glslink options</code>		<code>\glsorg@endtheglossary</code>	5
<code>counter</code>	93, 107, 238	<code>\glsorg@theglossary</code>	5
<code>format</code>	93, 107, 193	<code>\glspagelistwidth</code>	256, 262, 271, 278

<code>\glspar</code>	34	<code>\glsspace</code>	198
<code>\glsperscentchar</code>	144	<code>\glssstepentry</code>	185
<code>\GLSpl</code>	112	<code>\glssstepsubentry</code>	185
<code>\Glspl</code>	111, 244	<code>\glssubentrycounterlabel</code> ...	186
<code>\glspl</code>	82, 110, 111	<code>\glssubentryitem</code>	186
<code>\GLSplural</code>	116	<code>\GLSsymbol</code>	120
<code>\Glsplural</code>	115	<code>\Glssymbol</code>	120
<code>\glsplural</code>	115, 116	<code>\glssymbol</code>	120
<code>\glspluralaccessdisplay</code>	318	<code>\glssymbolaccessdisplay</code>	318
<code>\glspluralsuffix</code>	31, 60	<code>\glssymbolnav</code>	250
<code>\glspostdescription</code>	9	<code>\GLSsymbolplural</code>	121
<code>\glspostinline</code>	252	<code>\Glssymbolplural</code>	121
<code>\glsprestandardsort</code>	10	<code>\glssymbolplural</code>	120, 121
<code>\glsquote</code>	144	<code>\glssymbolpluralaccessdisplay</code>	
<code>\glsrefentry</code>	186	318
<code>\glsreset</code>	80	<code>\glssymbolsgroupname</code> .	31, 144, 190
<code>\glsresetall</code>	81	<code>\glstarget</code>	187
<code>\glsresetentrylist</code>	184	<code>\GLStext</code>	114
<code>\glsresetsubentrycounter</code> ...	185	<code>\Glstext</code>	114
<code>\glssee</code>	166	<code>\glstext</code>	113
<code>\glsseeformat</code>	147, 166	<code>\glstextaccessdisplay</code>	318
<code>\glsseeitem</code>	167	<code>\glstextformat</code>	82
<code>\glsseeitemformat</code>	167	<code>\glstextup</code>	197
<code>\glsseelastsep</code>	167	<code>\glstildechar</code>	144
<code>\glsseelist</code>	166	<code>\glstreeindent</code>	286, 287
<code>\glsseesep</code>	167	<code>\glstreenamefmt</code>	284
<code>\glsSetAlphaCompositor</code>	35	<code>\glsunset</code>	80
<code>\glsSetCompositor</code>	35	<code>\glsunsetall</code>	81
<code>\glssetexpandfield</code>	17	<code>\glsupacrpluralsuffix</code>	31
<code>\glssetnoexpandfield</code>	17	<code>\GlsUseAcrEntryDispStyle</code> ...	204
<code>\glsSetSuffixF</code>	35	<code>\GlsUseAcrStyleDefs</code>	204
<code>\glsSetSuffixFF</code>	36	<code>\GLSuseri</code>	122
<code>\glssettoctitle</code>	30	<code>\Glsuseri</code>	122
<code>\glssetwidest</code>	288	<code>\glsuseri</code>	121, 122
<code>\GlsSetXdyCodePage</code>	48	<code>\GLSuserii</code>	123
<code>\GlsSetXdyFirstLetterAfterDigits</code>		<code>\Glsuserii</code>	123
.....	144	<code>\glsuserii</code>	122, 123
<code>\GlsSetXdyLanguage</code>	48	<code>\GLSuseriii</code>	124
<code>\GlsSetXdyLocationClassOrder</code>	47	<code>\Glsuseriii</code>	123
<code>\GlsSetXdyMinRangeLength</code> ...	145	<code>\glsuseriii</code>	123, 124
<code>\GlsSetXdyStyles</code>	48	<code>\GLSuseriv</code>	125
<code>\glsshortaccessdisplay</code>	318	<code>\Glsuseriv</code>	124
<code>\glsshortaccesskey</code>	346	<code>\glsuseriv</code>	124, 125
<code>\glsshortkey</code>	197	<code>\GLSuserv</code>	126
<code>\glsshortpluralaccessdisplay</code>	318	<code>\Glsuserv</code>	125
<code>\glsshortpluralaccesskey</code> ...	346	<code>\glsuserv</code>	125, 126
<code>\glsshortpluralkey</code>	197	<code>\GLSuservi</code>	126
<code>\glsshorttok</code>	201	<code>\Glsuservi</code>	126
<code>\glssortnumberfmt</code>	11	<code>\glsuservi</code>	126

\glswrite 154
\glswritedefhook 71
\gMFUnocap 247

H

\hyperbf 195
\hyperemph 196
hyperfirst (option) 23
\hyperit 196
\hyperlink 106
\hypermd 195
\hyperpage 194
hyperref package ... 164, 168, 194, 238
\hyperrm 195
\hypersc 196
\hypersf 195
\hypersl 196
\hypertarget 106
\hypertt 195
\hyperup 196

I

\if@gl@docloaded 5
\if@gl@isacronymlist 15
\ifglossaryexists 50
\ifgl@descsuppressed 53
\ifgl@sentryexists 51
\ifgl@haschildren 52
\ifgl@hasdesc 53
\ifgl@hasfield 54
\ifgl@haslong 53
\ifgl@hasparent 52
\ifgl@hasprefix 240
\ifgl@hasprefixfirst 240
\ifgl@hasprefixfirstplural 240
\ifgl@hasprefixplural 240
\ifgl@hasshort 53
\ifgl@hassymbol 53
\ifgl@stranslate 21
\ifgl@sused 51, 80
\ifgl@xindy 25
\ifignoredglossary 59
index (option) 28
index (style) 284
indexgroup (style) 285
indexhypergroup (style) 285
indexonlyfirst (option) 23
inline (style) 250
\inputencodingname 25
\istfile 159

\istfilename 34
\item 193, 252, 284

L

link text 82
list (style) 252
listdotted (style) 255
listgroup (style) 253
listhypergroup (style) 253
\loadgl@sentries 13, 82
long (key) 63
long (style) 256
long-sc-short (acrstyle) 205
long-sc-short-desc (acrstyle) ..
..... 206, 334
long-short (acrstyle) 204, 332
long-short-desc (acrstyle) . 206, 334
long-sm-short (acrstyle) 205
long-sm-short-desc (acrstyle) ..
..... 207, 334
long3col (style) 257
long3colborder (style) 258
long3colheader (style) 258
long3colheaderborder (style) .. 258
long4col (style) 259
long4colborder (style) 260
long4colheader (style) 259
long4colheaderborder (style) .. 260
longaccess (key) 313
longborder (style) 256
longheader (style) 257
longheaderborder (style) 257
\longnewglossaryentry 60, 71
longplural (key) 63
longpluralaccess (key) 314
\longprovideglossaryentry ... 72
longragged (style) 262
longragged3col (style) 264
longragged3colborder (style) .. 264
longragged3colheader (style) .. 265
longragged3colheaderborder
(style) 265
longraggedborder (style) 263
longraggedheader (style) 263
longraggedheaderborder (style) 263
longtable (environment)
..... 8, 232, 255–267
longtable package 255, 262

M

`\makefirsttuc` 244
`makeglossaries`
 24, 25, 34, 48, 56, 153, 170
`\makeglossaries` 26,
 29, 30, 34, 35, 55, 57, 153, 155
`\makeglossary` 155
`makeindex` 349
`makeindex` 10, 25, 26, 31, 34–
 36, 56, 58, 60, 79, 98, 101, 144,
 147, 149, 151, 160, 164, 190, 293
 `delim_n` 36
 `delim_r` 36
 `page_compositor` 35
 special characters 100, 144
`makeindex (option)` 25
`\makenoidxglossaries` 21, 155
`mcolalttree (style)` 270
`mcolalttreegroup (style)` 270
`mcolalttreehypergroup (style)` . 270
`mcolindex (style)` 267
`mcolindexgroup (style)` 268
`mcolindexhypergroup (style)` ... 268
`mcoltree (style)` 268
`mcoltreegroup (style)` 268
`mcoltreehypergroup (style)` 269
`mcoltreenoname (style)` 269
`mcoltreenonamegroup (style)` ... 269
`mcoltreenonamehypergroup`
 (style) 269
`memoir class` 160
`mfirsttuc package` 1, 247
`\mfirsttucMakeUppercase` 246
`\mfu@checkword` 246
`\MFUclear` 247
`\MFUnocap` 247
`multicol package` 267
`multicols (environment)` 267

N

`name (key)` 59
`\new@glossaryentry` 66
`\newacronym` 24, 63, 82, 196, 197
`\newacronymhook` 201
`\newacronymstyle` 203
`\newglossary` 16, 56, 58, 151, 153, 181
`\newglossaryentry` .. 60, 66, 82, 197
`\newglossaryentry options`
 `access` 314, 316

`counter` 61
`description` 24, 59, 60,
 63, 66, 72, 118, 136, 197, 224, 313
`descriptionaccess` 316, 318
`descriptionplural` 119, 313
`descriptionpluralaccess` .. 316, 318
`first` 60, 75,
 107, 114, 137, 222, 227, 228, 313
`firstaccess` 316, 318
`firstplural` 60, 116, 138, 313
`firstpluralaccess` 316, 318
`format` 146
`long` 89, 140, 313
`longaccess` 317, 319
`longplural` 140, 314
`longpluralaccess` 317, 319
`name` 59,
 60, 63, 66, 72, 117, 135, 167, 312
`nonumberlist` 62
`parent` 62, 66
`plural` 60, 75, 115, 313
`pluralaccess` 316, 318
`prefix` 239
`prefixfirst` 239
`prefixfirstplural` 239
`prefixplural` 239
`see` 8, 61, 154, 155
`short` 89, 139, 313
`shortaccess` 317, 318
`shortplural` 140, 313
`shortpluralaccess` 317, 318
`sort` 60, 138, 190
`symbol` 59, 61, 120, 218,
 220, 222, 227, 259, 275, 312–314
`symbolaccess` 316, 318
`symbolplural` 120, 313
`symbolpluralaccess` 316, 318
`text` 60, 107, 113, 136, 218, 222, 312
`textaccess` 316, 318
`type` 13, 61, 82, 138
`user1` 121, 138, 314
`user2` 122, 138
`user3` 123, 139
`user4` 124, 139
`user5` 125, 139
`user6` 126, 139, 314
`\newglossarystyle` 192
`\newignoredglossary` 58
`nogroupskip (option)` 9

nohypertypes (option)	16
\noist	151, 238, 298
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)	5
numbers (option)	27

O

\oldacronym	196
order (option)	25

P

\p@gl@s@hyp@opt	94
package options:	
acronym	13, 14, 30, 168, 197
true	14
acronym	13
acronymlists	15
acronyms	14
automake	26
compatible-2.07	27
compatible-3.07	27
counter	16
counter	16
description	222, 223
description	24
dua	220, 222, 223
dua	24
entrycounter	182, 184
true	9
entrycounter	9
entrycounterwithin	9
footnote 108–113, 218, 220, 222, 224	
footnote	24
hyperfirst	
false	108–113
hyperfirst	23

index	28
indexonlyfirst	356
indexonlyfirst	23
makeindex	147, 238
makeindex	25
nogroupskip	9
nohypertypes	16
nolist	232
nolist	8
nolong	232, 256
nolong	8
nomain	12, 13
nomain	13
nonumberlist	7
nonumberlist	7
nopostdot	9
noredefwarn	17
nostyles	8
nosuper	232
nosuper	8
notranslate	22
notree	232
notree	8
nowarn	16
numberedsection	6
numberline	5
numberline	5
numbers	27
order	25
sanitize	20, 59, 135, 136
sanitize	21
sanitizesort	20
savenumberlist	7
savewrites	26, 354
false	151
true	153, 159
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, 232

style	7	\provideglossaryentry	66
subentrycounter	182, 185	\ProvidesGlossariesLang	31
subentrycounter	10		
symbols	27	R	
toc	5	\renewacronymstyle	203
true	5	\renewglossarystyle	193
toc	5	\RequireGlossariesLang	31
translate	22	\roman	44
false	22		
translate	22	S	
translator	21	\s@glshyp@opt	94
ucmark	9	\s@newglossary	56
xindy	25, 26, 147, 238	sanitize (option)	21
xindy	25	sanitizesort (option)	20
xindygloss	26	savenuumberlist (option)	7
xindynoglsnumbers	26	savewrites (option)	26
\pagelistname	31	sc-short-long (acrstyle)	206
parent (key)	62	sc-short-long-desc (acrstyle) ..	
\PGLS	243	207, 335
\Pgls	242	\scantokens	50
\pgls	240	\section	50
\PGLSpl	244	section (option)	6
\Pglspl	242	see (key)	61
\pglspl	241	seeautonumberlist (option)	8
\phantomsection	37–39	\seename	31
plural (key)	60	\SetAcronymLists	15
pluralaccess (key)	313	\SetAcronymStyle	14, 230
polyglossia package	22, 32	\setacronymstyle	203
\printacronyms	14	\SetCustomDisplayStyle	230
\PrintChanges	5	\SetCustomStyle	231
\printglossaries		\SetDefaultAcronymDisplayStyle	
. 12, 36, 55, 58, 155, 167, 168, 248		215
\printglossary	36, 37,	\SetDefaultAcronymStyle	215
56, 58, 155, 167, 168, 170, 181, 248		\SetDescriptionAcronymDisplayStyle	
\printglossary options		220
entrycounter	182	\SetDescriptionAcronymStyle	222
nogroupskip	182	\SetDescriptionDUAAcronymDisplayStyle	
nonumberlist	183	219
nopostdot	182	\SetDescriptionDUAAcronymStyle	
numberedsection	181	220
style	181	\SetDescriptionFootnoteAcronymDisplayStyle	
subentrycounter	182	216
title	181	\SetDescriptionFootnoteAcronymStyle	
toctitle	181	218
type	13, 167, 181	\SetDUADisplayStyle	228
\printnoidxglossaries	168	\SetDUASyle	229
\printnoidxglossary	168, 181	\setentrycounter	192
\printnoidxglossary options		\SetFootnoteAcronymDisplayStyle	
sort	183	223
		\SetFootnoteAcronymStyle ...	224

<code>\SetGenericNewAcronym</code>	201	<code>\showglosymbolaccess</code>	347
<code>\setglossarypreamble</code>	37	<code>\showglosymbolplural</code>	235
<code>\setglossarysection</code>	38	<code>\showglosymbolpluralaccess</code> .	347
<code>\setglossarystyle</code>	192	<code>\showglotext</code>	233
<code>\setglossentrycompatibility</code>	189	<code>\showglotextaccess</code>	347
<code>\SetSmallAcronymDisplayStyle</code>	225	<code>\showglotype</code>	233
<code>\SetSmallAcronymStyle</code>	227	<code>\showglouserii</code>	234
<code>\setStyleFile</code>	34	<code>\showglouseriii</code>	234
<code>\setupglossaries</code>	28	<code>\showglouseriv</code>	234
<code>short (key)</code>	63	<code>\showglouservi</code>	234
<code>short-long (acrstyle)</code>	205, 333	<code>sm-short-long (acrstyle)</code>	206
<code>short-long-desc (acrstyle)</code> .	207, 335	<code>sm-short-long-desc (acrstyle)</code> ..	208, 335
<code>shortaccess (key)</code>	313	<code>smallcaps (option)</code>	24
<code>shortplural (key)</code>	63	<code>smaller (option)</code>	24
<code>shortpluralaccess (key)</code>	313	<code>\SmallNewAcronymDef</code>	226, 345
<code>shotcuts (option)</code>	24	<code>sort (key)</code>	60
<code>\showacronymlists</code>	236	<code>sort (option)</code>	10
<code>\showglocounter</code>	233	<code>style (option)</code>	7
<code>\showglodesc</code>	235	<code>subentrycounter (option)</code>	10
<code>\showglodescaccess</code>	347	<code>\subglossentry</code>	187
<code>\showglodescplural</code>	235	<code>\subitem</code>	284
<code>\showglodescpluralaccess</code> ..	347	<code>sublistdotted (style)</code>	255
<code>\showglofirst</code>	233	<code>\subsubitem</code>	284
<code>\showglofirstaccess</code>	347	<code>super (style)</code>	271
<code>\showglofirsttpl</code>	233	<code>super3col (style)</code>	273
<code>\showglofirsttplpluralaccess</code> ..	347	<code>super3colborder (style)</code>	274
<code>\showgloflag</code>	236	<code>super3colheader (style)</code>	274
<code>\showgloindex</code>	236	<code>super3colheaderborder (style)</code> .	274
<code>\showglolevel</code>	232	<code>super4col (style)</code>	275
<code>\showgloloclist</code>	236	<code>super4colborder (style)</code>	276
<code>\showglolong</code>	236	<code>super4colheader (style)</code>	275
<code>\showglolongaccess</code>	348	<code>super4colheaderborder (style)</code> .	276
<code>\showglolongpluralaccess</code> ..	348	<code>superborder (style)</code>	272
<code>\showglongname</code>	235	<code>superheader (style)</code>	272
<code>\showglongnameaccess</code>	346	<code>superheaderborder (style)</code>	273
<code>\showgloparent</code>	232	<code>superragged (style)</code>	278
<code>\showgloplural</code>	233	<code>superragged3col (style)</code>	280
<code>\showglopluralaccess</code>	347	<code>superragged3colborder (style)</code> .	281
<code>\showgloshort</code>	236	<code>superragged3colheader (style)</code> .	281
<code>\showgloshortaccess</code>	347	<code>superraggedborder (style)</code>	279
<code>\showgloshortpluralaccess</code> ..	347	<code>superraggedheader (style)</code>	279
<code>\showglosort</code>	235	<code>superraggedheaderborder (style)</code>	280
<code>\showglossaries</code>	237	<code>superraggedright3colheaderborder</code>	
<code>\showglossarycounter</code>	237	<code>(style)</code>	281
<code>\showglossaryentries</code>	237	<code>supertabular (environment)</code> ...	
<code>\showglossaryin</code>	237		8, 232, 271–283
<code>\showglossaryout</code>	237		
<code>\showglossarytitle</code>	237		
<code>\showglosymbol</code>	235		

supertabular package . . 8, 232, 271, 278
 symbol (key) 61
 symbolaccess (key) 313
 \symbolname 31
 symbolplural (key) 61
 symbolpluralaccess (key) 313
 symbols (option) 27

T

text (key) 60
 textaccess (key) 312
 textcase package 4
 \theequation 165
 theglossary (environment)
 5, 17, 36, 37,
 186, 187, 193, 269, 270, 286–289
 \theHequation 165
 theindex (environment) 284
 toc (option) 5
 tracklang package 32, 348, 349
 \translate 32
 translate (option) 22
 translator package
 . . . 13, 14, 22, 27, 28, 32, 33, 168
 tree (style) 285
 treegroup (style) 286
 treehypergroup (style) 286
 treenoname (style) 287
 treenonamegroup (style) 288

treenonamehypergroup (style) . . 288
 type (key) 61

U

ucmark (option) 9
 user1 (key) 62
 user2 (key) 62
 user3 (key) 62
 user4 (key) 62
 user5 (key) 62
 user6 (key) 63

W

\warn@nomakeglossaries 153
 \warn@noprintglossary 167
 \writeist 34, 41, 43, 46, 145, 292, 294

X

\xcapitalisewords 247
 \xglaccsupp 317
 xindy 349
 xindy 10, 25, 26, 34, 35, 40,
 44, 46–49, 79, 105, 144, 145,
 147, 160, 164, 170, 190, 238, 293
 xindy (option) 25
 xindygloss (option) 26
 xindynoglsnumbers (option) 26
 \xmakefirstuc 246
 xspace package 4, 196