

Documented Code For glossaries v3.05

Nicola L.C. Talbot

Dickimaw Books

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

2013-04-21

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 v3.05: \TeX 2e Package to Assist Generating Glossaries”.

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

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

INSTALL Installation instructions.

CHANGES Change log.

README Package summary.

Contents

1 Main Package Code	3
1.1 Package Definition	3
1.2 Package Options	5
1.3 Default values	20
1.4 Xindy	28
1.5 Loops and conditionals	38
1.6 Defining new glossaries	40
1.7 Defining new entries	43
1.8 Resetting and unsetting entry flags	55
1.9 Loading files containing glossary entries	56
1.10 Using glossary entries in the text	57
1.10.1 Links to glossary entries	58
1.10.2 Displaying entry details without adding information to the glossary	109
1.11 Adding an entry to the glossary without generating text	116
1.12 Creating associated files	117
1.13 Writing information to associated files	126
1.14 Glossary Entry Cross-References	131
1.15 Displaying the glossary	132
1.16 Acronyms	146
1.17 Predefined acronym styles	150
1.18 Predefined Glossary Styles	165
1.19 Debugging Commands	165
1.20 Compatibility with version 2.07 and below	170
2 Mfirstuc Documented Code	170
3 Glossary Styles	172
3.1 Glossary hyper-navigation definitions (glossary-hypernav package)	172
3.2 In-line Style (glossary-inline.sty)	175
3.3 List Style (glossary-list.sty)	177
3.4 Glossary Styles using longtable (the glossary-long package)	180
3.5 Glossary Styles using longtable (the glossary-longragged package)	185
3.6 Glossary Styles using multicol (glossary-mcols.sty)	191
3.7 Glossary Styles using supertabular environment (glossary-super package)	194
3.8 Glossary Styles using supertabular environment (glossary-superragged package)	201
3.9 Tree Styles (glossary-tree.sty)	206
4 glossaries-compatible-207	214

5 Accessibility Support (glossaries-accsupp Code)	221
5.1 Defining Replacement Text	221
5.2 Accessing Replacement Text	224
5.3 Displaying the Glossary	237
5.4 Acronyms	237
5.5 Debugging Commands	241
6 Multi-Lingual Support	242
6.1 Babel Captions	242
6.2 Polyglossia Captions	248
6.3 Brazilian Dictionary	252
6.4 Danish Dictionary	252
6.5 Dutch Dictionary	252
6.6 English Dictionary	252
6.7 French Dictionary	253
6.8 German Dictionary	253
6.9 Irish Dictionary	253
6.10 Italian Dictionary	254
6.11 Magyar Dictionary	254
6.12 Polish Dictionary	254
6.13 Serbian Dictionary	255
6.14 Spanish Dictionary	255
Glossary	255
Change History	255
Index	265

1 Main Package Code

1.1 Package Definition

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

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries}[2013/04/21 v3.05 (NLCT)]
```

Required packages:

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

7 \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.)

```
8 \RequirePackage{amsgen}
```

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

```
9 \RequirePackage{etoolbox}
```

Check if doc has been loaded.

```
\if@gls@docloaded
```

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

18 \if@gls@docloaded
```

It has been loaded, so some modifications need to be made to ensure both packages can work together.

```
\glsorg@glossary First, save the original behaviour of \glossary
```

```
19 \newcommand{\glsorg@glossary}{%
20   \@bsphack
21   \begingroup
22     \@sanitize \glsorg@wrglossary
23 }
```

```
\glsorg@wrglossary
```

```
24 \newcommand{\glsorg@wrglossary}[1]{%
25   \protected@write\@glossaryfile{}{%
26     \string \glossaryentry{#1}{\thepage}}%
27   \endgroup
28   \@esphack
29 }
```

```
\changes Now we need to redefine \changes so that it uses the original definition of
\glossary.
```

```
30 \let\glsorg@changes\changes
31 \renewcommand{\changes}[3]{%
32   \begingroup
33     \let\glossary\glsorg@glossary
34     \glsorg@changes{#1}{#2}{#3}%
35   \endgroup
36 }
```

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

`\glsorg@theglossary`

```
37 \let\glsorg@theglossary\theglossary
```

`sorg@endtheglossary`

```
38 \let\glsorg@endtheglossary\endtheglossary
```

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

```
39 \let\glsorg@PrintChanges\PrintChanges
40 \renewcommand{\PrintChanges}{%
41   \begingroup
42     \let\theglossary\glsorg@theglossary
43     \let\endtheglossary\glsorg@endtheglossary
44     \glsorg@PrintChanges
45   \endgroup
46 }
```

End of doc stuff.

```
47 \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.

```
48 \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`.

```
49 \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`.

```
50 \ifcsundef{chapter}%
51   {\newcommand*\@@glossarysec{section}}%
52   {\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`.

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

Determine whether or not to use numbered sections.

```

\@glossarysecstar
56 \newcommand*{\@glossarysecstar}{*}

\@glossaryseclabel
57 \newcommand*{\@glossaryseclabel}{}

\glsautoprefix Prefix to add before label if automatically generated:
58 \newcommand*{\glsautoprefix}{}

numberedsection
59 \define@choicekey{glossaries.sty}{numberedsection}[\val\nr]{%
60 false,nolabel,autolabel}[nolabel]{%
61   \ifcase\nr\relax
62     \renewcommand*{\@glossarysecstar}{*}%
63     \renewcommand*{\@glossaryseclabel}{}%
64   \or
65     \renewcommand*{\@glossarysecstar}{}%
66     \renewcommand*{\@glossaryseclabel}{}%
67   \or
68     \renewcommand*{\@glossarysecstar}{}%
69     \renewcommand*{\@glossaryseclabel}{}%
70     \label{\glsautoprefix\glo@type}%
71   \fi
72 }

```

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
73 \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.
74 \define@key{glossaries.sty}{style}{%
75 \renewcommand*{\@glossary@default@style}{#1}}

```

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
76 \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).

```

77 \DeclareOptionX{nonumberlist}{%
78   \renewcommand*{\glossaryentrynumbers}[1]{\gls@save@numberlist{#1}}%
79 }

```

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

```

80 \define@boolkey{glossaries.sty}{gls}{savenumberlist}[true]{}
81 \glssavenumberlistfalse

```

`o@seeautonumberlist`

```

82 \newcommand*{\glo@seeautonumberlist}{}

```

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

```

83 \DeclareOptionX{seeautonumberlist}{%
84   \renewcommand*{\glo@seeautonumberlist}{%
85     \def\glo@prefix{\glsnextpages}%
86   }%
87 }

```

`\@gls@loadlong`

```

88 \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.

```

89 \DeclareOptionX{nolong}{\renewcommand*{\@gls@loadlong}{} }

```

`\@gls@loadsuper` The package isn't loaded if isn't installed.

```

90 \IfFileExists{supertabular.sty}{%
91   \newcommand*{\@gls@loadsuper}{\RequirePackage{glossary-super}}}%
92   \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.

```

93 \DeclareOptionX{nosuper}{\renewcommand*{\@gls@loadsuper}{} }

```

`\@gls@loadlist`

```

94 \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.

```

95 \DeclareOptionX{nolist}{\renewcommand*{\@gls@loadlist}{} }

```

`\@gls@loadtree`

```

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

notree This option prevents from being loaded (to reduce overheads if required). Nat-
       urally, the styles defined in will not be available if this option is used.
97 \DeclareOptionX{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).
98 \DeclareOptionX{nostyles}{%
99   \renewcommand*{\@gls@loadlong}{}%
100  \renewcommand*{\@gls@loadsuper}{}%
101  \renewcommand*{\@gls@loadlist}{}%
102  \renewcommand*{\@gls@loadtree}{}%
103  \let\@glossary@default@style\relax
104 }

\glspostdescription The description terminator is given by \glspostdescription (except for the
                    3 and 4 column styles). This is a full stop by default:
105 \newcommand*{\glspostdescription}{\ifglsnopostdot\else.\fi}

nopostdot Boolean option to suppress post description dot
106 \define@boolkey{glossaries.sty}[gls]{nopostdot}[true]{}
107 \glsnopostdotfalse

nogroupskip Boolean option to suppress vertical space between groups in the pre-defined
            styles.
108 \define@boolkey{glossaries.sty}[gls]{nogroupskip}[true]{}
109 \glsnogroupskipfalse

ucmark Boolean option to determine whether or not to use \MakeUppercase in defini-
        tion of \glossarymark
110 \define@boolkey{glossaries.sty}[gls]{ucmark}[true]{}
111 \glsucmarkfalse

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.
112 \define@boolkey{glossaries.sty}[gls]{entrycounter}[true]{}
113 \glsentrycounterfalse

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

```


`\@gls@counterwithin` The default value is no parent counter:

```
118 \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`.

```
119 \define@boolkey{glossaries.sty}[gls]{subentrycounter}[true]\{}
120 \glssubentrycounterfalse
```

`sort` Define the sort method: `sort=standard` (default), `sort=def` (order of definition) or `sort=use` (order of use).

```
121 \define@choicekey{glossaries.sty}{sort}{standard,def,use}\{}
122 \csname @gls@setupsort@#1\endcsname
123 }
```

`@setupsort@standard` Set up the macros for default sorting.

```
124 \newcommand*\@gls@setupsort@standard\{}
    Store entry information when it's defined.
125 \def\do@glo@storeentry{\@glo@storeentry}%
    No count register required for standard sort.
126 \def\@gls@defsortcount##1\{}%
    Sort according to sort key (\@glo@sort) if provided otherwise sort according
    to the entry's name (\@glo@name).
127 \def\@gls@defsort##1##2\{}
128 \ifx\@glo@sort\@glsdefaultsort
129 \let\@glo@sort\@glo@name
130 \fi
131 \@gls@sanitizesort
132 \expandafter\protected@xdef\csname glo@##2@sort\endcsname{\@glo@sort}%
133 }\}
```

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

```
134 \def\@gls@setsort##1\{}%
135 }
```

Set standard sort as the default:

```
136 \@gls@setupsort@standard
```

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

```
137 \newcommand*\glssortnumberfmt[1]\{}
138 \ifnum#1<100000 0\fi
139 \ifnum#1<10000 0\fi
140 \ifnum#1<1000 0\fi
141 \ifnum#1<100 0\fi
142 \ifnum#1<10 0\fi
143 \number#1%
144 }
```

```

\@gls@setupsort@def  Set up the macros for order of definition sorting.
145 \newcommand*{\@gls@setupsort@def}{%
    Store entry information when it's defined.
146   \def\do@glo@storeentry{\@glo@storeentry}%
    Defined count register associated with the glossary.
147   \def\@gls@defsortcount##1{%
148     \expandafter\global
149     \expandafter\newcount\csname glossary@##1@sortcount\endcsname
150   }%
    Increment count register associated with the glossary and use as the sort key.
151   \def\@gls@defsort##1##2{%
152     \expandafter\global\expandafter
153     \advance\csname glossary@##1@sortcount\endcsname by 1\relax
154     \expandafter\protected\edef\csname glo@##2@sort\endcsname{%
155       \expandafter\glssortnumberfmt
156       {\csname glossary@##1@sortcount\endcsname}}%
157   }%
    Don't need to do anything when the entry is used.
158   \def\@gls@setsort##1{%
159 }

\@gls@setupsort@use  Set up the macros for order of use sorting.
160 \newcommand*{\@gls@setupsort@use}{%
    Don't store entry information when it's defined.
161   \let\do@glo@storeentry\@gobble
    Defined count register associated with the glossary.
162   \def\@gls@defsortcount##1{%
163     \expandafter\global
164     \expandafter\newcount\csname glossary@##1@sortcount\endcsname
165   }%
    Initialise the sort key to empty.
166   \def\@gls@defsort##1##2{%
167     \expandafter\gdef\csname glo@##2@sort\endcsname{%
168   }%
    If the sort key hasn't been set, increment the counter associated with the glossary and set the sort key.
169   \def\@gls@setsort##1{%
    Get the parent, if one exists
170     \edef\@glo@parent{\csname glo@##1@parent\endcsname}%
    Set the information for the parent entry if not already done.
171     \ifx\@glo@parent\@empty
172     \else
173       \expandafter\@gls@setsort\expandafter{\@glo@parent}%
174     \fi

```

Set index information for this entry

```

175 \edef\@glo@type{\csname glo###1@type\endcsname}%
176 \edef\@gls@tmp{\csname glo###1@sort\endcsname}%
177 \ifx\@gls@tmp\@empty
178   \expandafter\global\expandafter
179   \advance\csname glossary@\@glo@type @sortcount\endcsname by 1\relax
180   \expandafter\protected@xdef\csname glo###1@sort\endcsname{%
181     \expandafter\glssortnumberfmt
182     {\csname glossary@\@glo@type @sortcount\endcsname}}%
183   \@glo@storeentry{##1}%
184 \fi
185 }%
186 }

```

`\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.)

```

187 \newcommand*{\glsdefmain}{%
188   \if@gls@docloaded
189     \newglossary[glg2]{main}{gls2}{glo2}{\glossaryname}%
190   \else
191     \newglossary{main}{gls}{glo}{\glossaryname}%
192   \fi
193 }

```

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`

```

194 \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`

```

195 \newcommand*{\acronymtype}{\glsdefaulttype}

```

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

```

196 \DeclareOptionX{nomain}{%
197   \let\glsdefaulttype\relax
198   \renewcommand*{\glsdefmain}{}%
199 }

```

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.

```
200 \define@boolkey{glossaries.sty}[gls]{acronym}[true]{%
201   \DeclareAcronymList{acronym}%
202 }
```

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

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

\@addtoacronymlists

```
204 \newcommand*{\@addtoacronymlists}[1]{%
205   \ifx\@glsacronymlists\@empty
206     \protected@xdef\@glsacronymlists{#1}%
207   \else
208     \protected@xdef\@glsacronymlists{\@glsacronymlists,#1}%
209   \fi
210 }
```

\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.)

```
211 \newcommand*{\DeclareAcronymList}[1]{%
212   \glsIfListOfAcronyms{#1}{\@addtoacronymlists{#1}}%
213 }
```

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

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

```
214 \newcommand{\glsIfListOfAcronyms}[1]{%
215   \edef\@do@glis@islistofacronyms{%
216     \noexpand\@glis@islistofacronyms{#1}{\@glsacronymlists}}%
217   \@do@glis@islistofacronyms
218 }
```

Internal command requires label and list to be expanded:

```
219 \newcommand{\@glis@islistofacronyms}[4]{%
220   \def\@glis@islistofacronyms##1,#1,##2\end@glis@islistofacronyms{%
221     \def\@before{##1}\def\@after{##2}}%
222   \glis@islistofacronyms,#2,#1,\@nil\end@glis@islistofacronyms
223   \ifx\@after\@nnil
```

Not found

```
224   #4%
225   \else
```

```

Found
226     #3%
227     \fi
228 }

if@glsisacronymlist Convenient boolean.
229 \newif\if@glsisacronymlist

@checkisacronymlist Sets the above boolean if argument is a label representing a list of acronyms.
230 \newcommand*{\gls@checkisacronymlist}[1]{%
231     \glsIfListOfAcronyms{#1}%
232     {\@glsisacronymlisttrue}{\@glsisacronymlistfalse}%
233 }

\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.)
234 \newcommand*{\SetAcronymLists}[1]{%
235     \renewcommand*{\@glsacronymlists}{#1}%
236 }

acronymlists
237 \define@key{glossaries.sty}{acronymlists}{%
238     \@addtoacronymlists{#1}%
239 }

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
240 \newcommand{\glscounter}{page}

counter The counter option changes the default counter. (This just redefines \glscounter.)
241 \define@key{glossaries.sty}{counter}{%
242     \renewcommand*{\glscounter}{#1}%
243 }

\@gls@nohyperlist
244 \newcommand*{\@gls@nohyperlist}{}

sDeclareNoHyperList
245 \newcommand*{\GlsDeclareNoHyperList}[1]{%
246     \ifdefempty\@gls@nohyperlist
247     {%
248         \renewcommand*{\@gls@nohyperlist}{#1}%
249     }%
250     {%

```

```

251 \appto\@gls@nohyperlist{, #1}%
252 }%
253 }

```

nohypertypes

```

254 \define@key{glossaries.sty}{nohypertypes}{%
255 \GlsDeclareNoHyperList{#1}%
256 }

```

The glossary keys whose values are written to another file (i.e. sort, name, description and symbol) need to be sanitized, otherwise fragile commands would not be able to be used in `\newglossaryentry`. However, strange results will occur if you then use those fields in the document. As these fields are not normally used in the document, but are by default only used in the glossary, the default is to sanitize them. If however you want to use these values in the document (either by redefining commands like `\glsdisplay` or by using commands like `\glsentrydesc`) you will have to switch off the sanitization using the `sanitize` package option, but you will then have to use `\protect` to protect fragile commands when defining new glossary entries. The `sanitize` option takes a key-value list as its value, which can be used to switch individual values on and off. For example:

```
\usepackage[sanitize={description,name,symbol=false}]{glossaries}
```

will switch off the sanitization for the symbol key, but switch it on for the description and name keys. This would mean that you can use fragile commands in the description and name when defining a new glossary entry, but not for the symbol.

The default values are defined as:

```

\@gls@sanitizedesc
257 \newcommand*{\@gls@sanitizedesc}{\@onelevel@sanitize\@glo@desc}

\@gls@sanitizename
258 \newcommand*{\@gls@sanitizename}{\@onelevel@sanitize\@glo@name}

\@gls@sanitizesymbol
259 \newcommand*{\@gls@sanitizesymbol}{\@onelevel@sanitize\@glo@symbol}

\@gls@sanitizesort
260 \newcommand*{\@gls@sanitizesort}{\@onelevel@sanitize\@glo@sort}

```

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.

Firstly the description. If set, it will redefine `\@gls@sanitizedesc` to use `\@onelevel@sanitize`, otherwise `\@gls@sanitizedesc` will do nothing.

```

261 \define@boolkey[glS]{sanitize}{description}[true]{%
262 \ifglS@sanitize@description
263   \renewcommand*{\@glS@sanitizedesc}{\@onelevel@sanitize\@glo@desc}%
264 \else
265   \renewcommand*{\@glS@sanitizedesc}{}%
266 \fi
267 }

```

Similarly for the name key:

```

268 \define@boolkey[glS]{sanitize}{name}[true]{%
269 \ifglS@sanitize@name
270   \renewcommand*{\@glS@sanitizename}{\@onelevel@sanitize\@glo@name}%
271 \else
272   \renewcommand*{\@glS@sanitizename}{}%
273 \fi}

```

and for the symbol key:

```

274 \define@boolkey[glS]{sanitize}{symbol}[true]{%
275 \ifglS@sanitize@symbol
276   \renewcommand*{\@glS@sanitizesymbol}{%
277 \@onelevel@sanitize\@glo@symbol}%
278 \else
279   \renewcommand*{\@glS@sanitizesymbol}{}%
280 \fi}

```

and for the sort key:

```

281 \define@boolkey[glS]{sanitize}{sort}[true]{%
282 \ifglS@sanitize@sort
283   \renewcommand*{\@glS@sanitizesort}{%
284 \@onelevel@sanitize\@glo@sort}%
285 \else
286   \renewcommand*{\@glS@sanitizesort}{}%
287 \fi}

```

sanitize Now define the sanitize option. It can either take a key-val list as its value, or it can take the keyword none, which is equivalent to description=false, symbol=false, name=false:

```

288 \define@key{glossaries.sty}{sanitize}[description=true,symbol=true,
289 name=true]{%
290 \ifthenelse{\equal{#1}{none}}{%
291 {%
292   \renewcommand*{\@glS@sanitizedesc}{}%
293   \renewcommand*{\@glS@sanitizename}{}%
294   \renewcommand*{\@glS@sanitizesymbol}{}%
295 }%
296 {%
297   \setkeys[glS]{sanitize}{#1}}%
298 }

```

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

```

299 \define@boolkey{glossaries.sty}[glS]{translate}[true]{%

```

Set the default value:

```
300 \glstranslatefalse
301 \ifpackageloaded{translator}%
302   {\glstranslatetrue}%
303   {%
304     \ifpackageloaded{polyglossia}%
305       {\glstranslatetrue}%
306       {%
307         \ifpackageloaded{babel}{\glstranslatetrue}{}}%
308       }%
309 }
```

indexonlyfirst Set whether to only index on first use.

```
310 \define@boolkey{glossaries.sty}[gls]{indexonlyfirst}[true]{}
311 \glindexonlyfirstfalse
```

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

```
312 \define@boolkey{glossaries.sty}[gls]{hyperfirst}[true]{}
313 \glshyperfirsttrue
```

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

```
314 \define@boolkey{glossaries.sty}[glsacr]{footnote}[true]{}%
315 \ifthenelse{\boolean{glsacrdescription}}{}{}%
316 {\renewcommand*{\@gls@sanitizedesc}{}}%
317 }
```

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

```
318 \define@boolkey{glossaries.sty}[glsacr]{description}[true]{}%
319 \renewcommand*{\@gls@sanitizesymbol}{}%
320 }
```

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

```
321 \define@boolkey{glossaries.sty}[glsacr]{smallcaps}[true]{}%
322 \renewcommand*{\@gls@sanitizesymbol}{}%
323 }
```

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

```
324 \define@boolkey{glossaries.sty}[glsacr]{smaller}[true]{}%
325 \renewcommand*{\@gls@sanitizesymbol}{}%
326 }
```

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

```
327 \define@boolkey{glossaries.sty}[glsacr]{dua}[true]{}%
328 \renewcommand*{\@gls@sanitizesymbol}{}%
329 }
```


`shortcuts` Define acronym shortcuts.
330 `\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.
331 `\newcommand*{\glsorder}{word}`

`\@glsorder` The ordering information is written to the auxiliary file for `makeglossaries`, so ignore the auxiliary information.
332 `\newcommand*{\@glsorder}[1]{}`

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

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

The default is `makeindex`:
336 `\glxindyfalse`

Define package option to specify that `makeindex` will be used to sort the glossaries:
337 `\DeclareOptionX{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.
338 `\define@boolkey[glx]{xindy}{glsnumbers}[true]{}`
339 `\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.)
340 `\def\@xdy@main@language{\rootlanguage}%`

Define key to set the language
341 `\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.
342 `\ifcsundef{inputencodingname}{%`
343 `\def\gls@codepage{}}{%`
344 `\def\gls@codepage{\inputencodingname}`
345 `}`

Define a key to set the code page.

```
346 \define@key[glS]{xindy}{codepage}{\def\glS@codepage{#1}}
```

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

```
347 \define@key{glossaries.sty}{xindy}[]{%
348   \glSxindytrue
349   \setkeys[glS]{xindy}{#1}%
350 }
```

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

```
351 \define@boolkey{glossaries.sty}[glS]{savewrites}[true]{}%
```

Set default:

```
352 \glSsavewritesfalse
```

\GlossariesWarning Prints a warning message.

```
353 \newcommand*{\GlossariesWarning}[1]{%
354   \PackageWarning{glossaries}{#1}%
355 }
```

sariesWarningNoLine Prints a warning message without the line number.

```
356 \newcommand*{\GlossariesWarningNoLine}[1]{%
357   \PackageWarningNoLine{glossaries}{#1}%
358 }
```

Define package option to suppress warnings

```
359 \DeclareOptionX{nowarn}{%
360   \renewcommand*{\GlossariesWarning}[1]{}%
361   \renewcommand*{\GlossariesWarningNoLine}[1]{}%
362 }
```

compatible-2.07

```
363 \define@boolkey{glossaries.sty}[glS]{compatible-2.07}[true]{}%
364 \csname glScompatible-2.07false\endcsname
```

Process package options:

```
365 \ProcessOptionsX
```

If package is loaded, check to see if it is installed, but only if translation is required.

```
366 \ifglStranslate
367   \@ifpackageloaded{polyglossia}%
368   {%
```

polyglossia fakes babel so need to check for polyglossia first.

```

369 }%
370 {%
371   \@ifpackageloaded{babel}%
372   {%
373     \IfFileExists{translator.sty}%
374     {%
375       \RequirePackage{translator}%
376     }%
377   }%
378 }%
379 {}
380 }
381 \fi

```

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}).

```

382 \ifthenelse{\equal{\glscounter}{section}}{%
383 {%
384   \ifcsundef{chapter}{}%
385   {%
386     \let\@gls@old@chapter\@chapter
387     \def\@chapter[#1]#2{\@gls@old@chapter[#1]{#2}%
388       \ifcsundef{hyperdef}{\hyperdef{section}{\thesection}}}%
389   }%
390 }%
391 {}

```

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

```

392 \newcommand*{\@gls@onlypremakeg}{}

```

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

```

393 \newcommand*{\@onlypremakeg}[1]{%
394 \ifx\@gls@onlypremakeg\@empty
395   \def\@gls@onlypremakeg{#1}%
396 \else
397   \expandafter\toks@\expandafter{\@gls@onlypremakeg}%
398   \edef\@gls@onlypremakeg{\the\toks@,\noexpand#1}%
399 \fi}

```

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

```
400 \newcommand*{\@disable@onlypremakeg}{%
401 \@for\@thiscs:=\@gls@onlypremakeg\do{%
402   \expandafter\@disable@premakecs\@thiscs%
403 }}
```

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

```
404 \newcommand*{\@disable@premakecs}[1]{%
405   \def#1{\PackageError{glossaries}{\string#1\space may only be
406     used before \string\makeglossaries}{You can't use
407     \string#1\space after \string\makeglossaries}}%
408 }
```

1.3 Default values

This section sets up default values 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`

```
409 \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`

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

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

```
411 \newcommand*{\glssettoctitle}[1]{%
412 \def\glossarytoctitle{\csname @glotype@#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`

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

`\descriptionname`

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

`\symbolname`

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

`\pagelistname`

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

Labels for makeindex's symbol and number groups:

glsymbolsgroupname

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

glsnumbersgroupname

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

\glspluralsuffix The default plural is formed by appending \glspluralsuffix to the singular form.

```
419 \newcommand*{\glspluralsuffix}{s}
```

\seename

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

\andname

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

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

dglossarytocaptions 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.)

```
422 \newcommand*{\addglossarytocaptions}[1]{%
423   \ifcsundef{captions#1}{}%
424   {%
425     \expandafter\let\expandafter\@gls@tmp\csname captions#1\endcsname
426     \expandafter\toks@\expandafter{\@gls@tmp
427       \renewcommand*{\glossaryname}{\translate{Glossary}}}%
428     }%
429     \expandafter\edef\csname captions#1\endcsname{\the\toks@}%
430   }%
431 }
```

```
432 \ifglstranslate
```

If is not install, used standard captions, otherwise load dictionary.

```
433 \@ifpackageloaded{translator}{%
434   \usedictionary{glossaries-dictionary}%
435   \addglossarytocaptions{portuges}%
436   \addglossarytocaptions{portuguese}%
437   \addglossarytocaptions{brazil}%
438   \addglossarytocaptions{brazilian}%
439   \addglossarytocaptions{danish}%
440   \addglossarytocaptions{dutch}%
441   \addglossarytocaptions{afrikaans}%
442   \addglossarytocaptions{english}%
443   \addglossarytocaptions{UKenglish}%

```

```

444 \addglossarytocaptions{USenglish}%
445 \addglossarytocaptions{american}%
446 \addglossarytocaptions{australian}%
447 \addglossarytocaptions{british}%
448 \addglossarytocaptions{canadian}%
449 \addglossarytocaptions{newzealand}%
450 \addglossarytocaptions{french}%
451 \addglossarytocaptions{frenchb}%
452 \addglossarytocaptions{français}%
453 \addglossarytocaptions{acadian}%
454 \addglossarytocaptions{canadien}%
455 \addglossarytocaptions{german}%
456 \addglossarytocaptions{germanb}%
457 \addglossarytocaptions{austrian}%
458 \addglossarytocaptions{naustrian}%
459 \addglossarytocaptions{ngerman}%
460 \addglossarytocaptions{irish}%
461 \addglossarytocaptions{italian}%
462 \addglossarytocaptions{magyar}%
463 \addglossarytocaptions{hungarian}%
464 \addglossarytocaptions{polish}%
465 \addglossarytocaptions{spanish}%
466 \renewcommand*{\glstoc{title}}[1]{%
467 \ifthenelse{\equal{#1}{main}}{%
468 \translatelet{\glossarytoc{title}}{Glossary}}{%
469 \ifthenelse{\equal{#1}{acronym}}{%
470 \translatelet{\glossarytoc{title}}{Acronyms}}{%
471 \def\glossarytoc{title}{\csname @glotype@#1@title\endcsname}}}%
472 \renewcommand*{\glossaryname}{\translate{Glossary}}%
473 \renewcommand*{\acronymname}{\translate{Acronyms}}%
474 \renewcommand*{\entryname}{\translate{Notation (glossaries)}}%
475 \renewcommand*{\descriptionname}{%
476 \translate{Description (glossaries)}}%
477 \renewcommand*{\symbolname}{\translate{Symbol (glossaries)}}%
478 \renewcommand*{\pagelistname}{%
479 \translate{Page List (glossaries)}}%
480 \renewcommand*{\glssymbolsgroupname}{%
481 \translate{Symbols (glossaries)}}%
482 \renewcommand*{\glnumbersgroupname}{%
483 \translate{Numbers (glossaries)}}%
484 }{%

485 \@ifpackageloaded{polyglossia}%
486 {\RequirePackage{glossaries-polyglossia}}%
487 {%
488 \@ifpackageloaded{babel}{%
489 \RequirePackage{glossaries-babel}}}%
490 }}
491 \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.

```
492 \DeclareRobustCommand*\nopostdesc{}\}
```

`\@nopostdesc` Suppress next description terminator.

```
493 \newcommand*\@nopostdesc{%
494   \let\org@glspostdescription\glspostdescription
495   \def\glspostdescription{%
496     \let\glspostdescription\org@glspostdescription}%
497 }
```

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

```
498 \newcommand{\glspar}{\par}
```

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

```
499 \ifglsxindy
500   \newcommand{\setStyleFile}[1]{%
501     \renewcommand{\istfilename}{#1.xdy}}
502 \else
503   \newcommand{\setStyleFile}[1]{%
504     \renewcommand{\istfilename}{#1.ist}}
505 \fi
```

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

```
506 \@onlypremakeg\setStyleFile
```

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

`\istfilename`

```
507 \ifglsxindy
508   \def\istfilename{\jobname.xdy}
509 \else
510   \def\istfilename{\jobname.ist}
511 \fi
```

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`

```
512 \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`

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

`\glsSetCompositor` Sets the compositor.

```
514 \newcommand*{\glsSetCompositor}[1]{%
515   \renewcommand*{\glscompositor}{#1}}
Only use before \makeglossaries
516 \@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.

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

`glsSetAlphaCompositor` Sets the alpha compositor.

```
518 \ifglsxindy
519   \newcommand*\glsSetAlphaCompositor[1]{%
520     \renewcommand*\@glsAlphacompositor{#1}}
521 \else
522   \newcommand*\glsSetAlphaCompositor[1]{%
523     \glsnoxindywarning\glsSetAlphaCompositor}
524 \fi
Can only be used before \makeglossaries
525 \@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.

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

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

```
527 \newcommand*{\glsSetSuffixF}[1]{%
528   \renewcommand*{\gls@suffixF}{#1}}
Only has an effect when used before \makeglossaries
529 \@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.

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


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

```
531 \newcommand*{\glsSetSuffixFF}[1]{%  
532   \renewcommand*{\gls@suffixFF}{#1}%  
533 }
```

`\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".

```
534 \ifcsundef{hyperlink}%  
535 {%  
536   \newcommand*{\glsnumberformat}[1]{#1}%  
537 }%  
538 {%  
539   \newcommand*{\glsnumberformat}[1]{\glshypernumber{#1}}%  
540 }
```

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`

```
541 \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`

```
542 \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`

```
543 \newcommand*{\glossarypreamble}{}%
```

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

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

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

`\glossarypostamble`

```
544 \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`.

```
545 \newcommand*{\glossarysection}[2][\@gls@title]{%
546   \def\@gls@title{#2}%
547   \ifcsundef{phantomsection}%
548     {%
549     \@glossarysection{#1}{#2}%
550   }%
551   {%
552     \p@glossarysection{#1}{#2}%
553   }%
554   \glossarymark{\glossarytoctitle}%
555 }
```

`\glossarymark`

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

```
556 \ifcsundef{glossarymark}%
557 {%
558   \ifglsucmark
559     \newcommand{\glossarymark}[1]{%
560       \@mkboth{\MakeUppercase{#1}}{\MakeUppercase{#1}}%
561     }
562   \else
563     \newcommand{\glossarymark}[1]{\@mkboth{#1}{#1}}
564   \fi
565 }%
566 {%
567   \GlossariesWarning{overriding \string\glossarymark}%
568   \@ifclassloaded{memoir}%
569   {
570     \ifglsucmark
571       \renewcommand{\glossarymark}[1]{%
572         \@mkboth{\MakeUppercase{#1}}{\MakeUppercase{#1}}%
573       }
574     \else
```

```

575     \renewcommand{\glossarymark}[1]{%
576         \markboth{\memUHead{#1}}{\memUHead{#1}}%
577     }
578     \fi
579 }
580 {
581     \ifglsucmark
582         \renewcommand{\glossarymark}[1]{%
583             \@mkboth{\MakeUppercase{#1}}{\MakeUppercase{#1}}%
584         }
585     \else
586         \renewcommand{\glossarymark}[1]{\@mkboth{#1}{#1}}
587     \fi
588 }
589 }

```

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`

```

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

```

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

`\@glossarysection`

```

592 \newcommand*\@glossarysection[2]{%
593 \ifx\@glossarysecstar\@empty
594     \csname\@glossarysec\endcsname{#2}%
595 \else
596     \csname\@glossarysec\endcsname*{#2}%
597     \@gls@toc{#1}{\@glossarysec}%
598 \fi
599 \@glossaryseclabel}

```

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.

`\@pglossarysection`

```

600 \newcommand*\@pglossarysection[2]{%
601 \glsclearpage
602 \phantomsection
603 \ifx\@glossarysecstar\@empty

```

```

604 \csname\@@glossarysec\endcsname{#2}%
605 \else
606 \@gls@toc{#1}{\@@glossarysec}%
607 \csname\@@glossarysec\endcsname*{#2}%
608 \fi
609 \@@glossaryseclabel}

```

`\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.

```

610 \newcommand*\gls@doclearpage{%
611 \ifthenelse{\equal{\@@glossarysec}{chapter}}{
612   {
613     \ifcsundef{cleardoublepage}{\clearpage}{\cleardoublepage}%
614   }%
615   {}%
616 }

```

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

```

617 \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`

```

618 \newcommand*\@gls@toc[2]{%
619 \ifglstoc
620   \ifglsnumberline
621     \addcontentsline{toc}{#2}{\numberline{#1}}%
622   \else
623     \addcontentsline{toc}{#2}{#1}%
624   \fi
625 \fi}

```

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

```

626 \newcommand*\glsnoxindywarning[1]{%
627 \GlossariesWarning{Not in xindy mode --- ignoring \string#1}%
628 }

```

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

```
629 \ifglxindy
630   \edef\@xdyattributes{\string"default\string"}%
631 \fi
```

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

```
632 \ifglxindy
633   \edef\@xdyattributelist{}%
634 \fi
```

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

```
635 \ifglxindy
636   \def\@xdylocref{}
637 \fi
```

`\@gls@ifinlist`

```
638 \newcommand*\@gls@ifinlist[4]{%
639   \def\@do@ifinlist##1,#1,##2\end@do@ifinlist{%
640     \def\@gls@listsuffix{##2}%
641     \ifx\@gls@listsuffix\@empty
642       #4%
643     \else
644       #3%
645     \fi
646   }%
647   \@do@ifinlist,#2,#1,\end@do@ifinlist
648 }
```

`\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.

```
649 \ifglxindy
650   \newcommand*\@xdycounters{\@glscounter}
651   \newcommand*\GlsAddXdyCounters[1]{%
652     \@for\@gls@ctr:=#1\do{%
653       \edef\@do@addcounter{%
654         \noexpand\@gls@ifinlist{\@gls@ctr}{\@xdycounters}{}%
655         {%
656           \noexpand\edef\noexpand\@xdycounters{\@xdycounters,%
657             \noexpand\@gls@ctr}%
658         }%
659       }%
660       \@do@addcounter
661     }
662 }
```

Only has an effect before `\writeist`:

```
663 \onlypremakeg\GlsAddXdyCounters
664 \else
665 \newcommand*\GlsAddXdyCounters[1]{%
666 \glsnxindywarning\GlsAddXdyAttribute
667 }
668 \fi
```

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

```
669 \newcommand*\@disabled@glssaddxdycounters{%
670 \PackageError{glossaries}{\string\GlsAddXdyCounters\space
671 can't be used after \string\GlsAddXdyAttribute}{Move all
672 occurrences of \string\GlsAddXdyCounters\space before the first
673 instance of \string\GlsAddXdyAttribute}%
674 }
```

`\GlsAddXdyAttribute` Adds an attribute.

```
675 \ifglxindy
```

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

```
676 \newcommand*\@glssaddxdyattribute[2]{%
```

Add to xindy attribute list

```
677 \edef\@xdyattributes{\@xdyattributes ^^J \string"#1\string" ^^J
678 \string"#2#1\string"}%
```

Add to xindy markup location.

```
679 \expandafter\toks@\expandafter{\@xdylocref}%
680 \edef\@xdylocref{\the\toks@ ^^J}%
681 (markup-locref
682 :open \string"\string~n%
683 \expandafter\string\csname glsX#2X#1\endcsname
684 \string" ^^J
685 :close \string"\string" ^^J
686 :attr \string"#2#1\string"))%
```

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

```
687 \expandafter\gdef\csname glsX#2X#1\endcsname##1##2{%
688 \setentrycounter{##1}{#2}\csname #1\endcsname{##2}%
689 }%
690 }
```

High-level command:

```
691 \newcommand*\GlsAddXdyAttribute[1]{%
```

Add to comma-separated attribute list

```
692 \ifx\@xdyattributelist\@empty
693 \edef\@xdyattributelist{#1}%
694 \else
695 \edef\@xdyattributelist{\@xdyattributelist,#1}%
696 \fi
```

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

```

697 \for\@this@counter:=\@xdycounters\do{%
698 \protected@edef\gls@do@addxdyattribute{%
699 \noexpand\@glsaddxdyattribute{#1}{\@this@counter}%
700 }
701 \gls@do@addxdyattribute
702 }%

```

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

```

703 \let\GlsAddXdyCounters\@disabled@glsaddxdycounters
704 }

```

Only has an effect before `\writeist`:

```

705 \@onlypremakeg\GlsAddXdyAttribute
706 \else
707 \newcommand*\GlsAddXdyAttribute[1]{%
708 \glsnoxindywarning\GlsAddXdyAttribute}
709 \fi

```

redefinedattributes Add known attributes for all defined counters

```

710 \ifglxsindy
711 \newcommand*{\@gls@addpredefinedattributes}{%
712 \GlsAddXdyAttribute{glsnumberformat}
713 \GlsAddXdyAttribute{textrm}
714 \GlsAddXdyAttribute{textsf}
715 \GlsAddXdyAttribute{texttt}
716 \GlsAddXdyAttribute{textbf}
717 \GlsAddXdyAttribute{textmd}
718 \GlsAddXdyAttribute{textit}
719 \GlsAddXdyAttribute{textup}
720 \GlsAddXdyAttribute{textsl}
721 \GlsAddXdyAttribute{textsc}
722 \GlsAddXdyAttribute{emph}
723 \GlsAddXdyAttribute{glshypernumber}
724 \GlsAddXdyAttribute{hyperrrm}
725 \GlsAddXdyAttribute{hypersf}
726 \GlsAddXdyAttribute{hypertt}
727 \GlsAddXdyAttribute{hyperbf}
728 \GlsAddXdyAttribute{hypermd}
729 \GlsAddXdyAttribute{hyperit}
730 \GlsAddXdyAttribute{hyperup}
731 \GlsAddXdyAttribute{hypersl}
732 \GlsAddXdyAttribute{hypersc}
733 \GlsAddXdyAttribute{hyperemph}
734 }
735 \else
736 \let\@gls@addpredefinedattributes\relax
737 \fi

```

\@xdyuseralphabets List of additional alphabets

```
738 \def\@xdyuseralphabets{}
```

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

```
739 \ifglxindy
740   \newcommand*{\GlsAddXdyAlphabet}[2]{%
741     \edef\@xdyuseralphabets{%
742       \@xdyuseralphabets ^^J
743       (define-alphabet "#1" (#2))}}
744 \else
745   \newcommand*{\GlsAddXdyAlphabet}[2]{%
746     \glsnnoxindywarning\GlsAddXdyAlphabet}
747 \fi
```

This code is only required for xindy:

```
748 \ifglxindy
```

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

```
749   \newcommand*{\@gls@xdy@locationlist}{%
750     roman-page-numbers,%
751     Roman-page-numbers,%
752     arabic-page-numbers,%
753     alpha-page-numbers,%
754     Alpha-page-numbers,%
755     Appendix-page-numbers,%
756     arabic-section-numbers%
757   }
```

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

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

```
758   \protected@edef\@gls@roman{\@roman{0}\string"
759     \string"roman-numbers-lowercase\string" :sep \string"}}%
760   \@onelevel@sanitize\@gls@roman
761   \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
762     :sep \string"}%
763   \@onelevel@sanitize\@tmp
764   \ifx\@tmp\@gls@roman
765     \expandafter
766       \edef\csname @gls@xdy@Lclass@roman-page-numbers\endcsname{%
767         \string"roman-numbers-lowercase\string"%
768       }%
769   \else
770     \expandafter
771     \edef\csname @gls@xdy@Lclass@roman-page-numbers\endcsname{
```



```

772         :sep \string"\@gls@roman\string"%
773     }%
774 \fi

@Roman-page-numbers Upper case Roman numerals (I, II, ...).
775 \expandafter\def\csname @gls@xdy@Lclass@Roman-page-numbers\endcsname{%
776     \string"roman-numbers-uppercase\string"%
777 }%

arabic-page-numbers Arabic numbers (1, 2, ...).
778 \expandafter\def\csname @gls@xdy@Lclass@arabic-page-numbers\endcsname{%
779     \string"arabic-numbers\string"%
780 }%

@alpha-page-numbers Lower case alphabetical (a, b, ...).
781 \expandafter\def\csname @gls@xdy@Lclass@alpha-page-numbers\endcsname{%
782     \string"alpha\string"%
783 }%

@Alpha-page-numbers Upper case alphabetical (A, B, ...).
784 \expandafter\def\csname @gls@xdy@Lclass@Alpha-page-numbers\endcsname{%
785     \string"ALPHA\string"%
786 }%

pendix-page-numbers Appendix style locations (e.g. A-1, A-2, ..., B-1, B-2, ...). The separator is given
by \@glsAlphacompositor.
787 \expandafter\def\csname @gls@xdy@Lclass@Appendix-page-numbers\endcsname{%
788     \string"ALPHA\string"
789     :sep \string"\@glsAlphacompositor\string"
790     \string"arabic-numbers\string"%
791 }

arabic-section-numbers Section number style locations (e.g. 1.1, 1.2, ...). The compositor is given by
\glscompositor.
792 \expandafter\def\csname @gls@xdy@Lclass@arabic-section-numbers\endcsname{%
793     \string"arabic-numbers\string"
794     :sep \string"\glscompositor\string"
795     \string"arabic-numbers\string"%
796 }%

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

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

End of xindy-only block:
799 \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.)

```

800 \ifglxindy
801   \newcommand*{\GlsAddXdyLocation}[3][{}]{%
802     \def\@gls@tmp{#1}%
803     \ifx\@gls@tmp\@empty
804       \edef\@xdyuserlocationdefs{%
805         \@xdyuserlocationdefs ^^J%
806         (define-location-class \string"#2\string"^^J\space\space
807         \space(:sep \string"{}\glsoopenbrace\string" #3
808           :sep \string"\glsclosebrace\string"))
809       }%
810     \else
811       \edef\@xdyuserlocationdefs{%
812         \@xdyuserlocationdefs ^^J%
813         (define-location-class \string"#2\string"^^J\space\space
814         \space(:sep "\glsoopenbrace"
815           #1
816           :sep "\glsclosebrace\glsoopenbrace" #3
817           :sep "\glsclosebrace"))
818       }%
819     \fi
820     \edef\@xdyuserlocationnames{%
821       \@xdyuserlocationnames^^J\space\space\space
822       \string"#1\string"}%
823   }

```

Only has an effect before `\writeist`:

```

824   \@onlypremakeg\GlsAddXdyLocation
825 \else
826   \newcommand*{\GlsAddXdyLocation}[2]{%
827     \glsnnoxindywarning\GlsAddXdyLocation}
828 \fi

```

`\locationclassorder` Define location class order

```

829 \ifglxindy
830   \edef\@xdylocationclassorder{^^J\space\space\space
831     \string"roman-page-numbers\string"^^J\space\space\space
832     \string"arabic-page-numbers\string"^^J\space\space\space
833     \string"arabic-section-numbers\string"^^J\space\space\space
834     \string"alpha-page-numbers\string"^^J\space\space\space
835     \string"Roman-page-numbers\string"^^J\space\space\space
836     \string"Alpha-page-numbers\string"^^J\space\space\space
837     \string"Appendix-numbers\string"
838     \@xdyuserlocationnames^^J\space\space\space
839     \string"see\string"
840   }

```

841 \fi

Change the location order.

yLocationClassOrder

```
842 \ifglxindy
843   \newcommand*\GlsSetXdyLocationClassOrder[1]{%
844     \def\@xdylocationclassorder{#1}}
845 \else
846   \newcommand*\GlsSetXdyLocationClassOrder[1]{%
847     \glsnnoxindywarning\GlsSetXdyLocationClassOrder}
848 \fi
```

\@xdysortrules Define sort rules

```
849 \ifglxindy
850   \def\@xdysortrules{}
851 \fi
```

\GlsAddSortRule Add a sort rule

```
852 \ifglxindy
853   \newcommand*\GlsAddSortRule[2]{%
854     \expandafter\toks@\expandafter{\@xdysortrules}%
855     \protected@edef\@xdysortrules{\the\toks@ ^^J
856       (sort-rule \string"#1\string" \string"#2\string")}%
857   }
858 \else
859   \newcommand*\GlsAddSortRule[2]{%
860     \glsnnoxindywarning\GlsAddSortRule}
861 \fi
```

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

```
862 \ifglxindy
863   \def\@xdyrequiredstyles{tex}
864 \fi
```

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

```
865 \ifglxindy
866   \newcommand*\GlsAddXdyStyle[1]{%
867     \edef\@xdyrequiredstyles{\@xdyrequiredstyles,#1}}%
868 \else
869   \newcommand*\GlsAddXdyStyle[1]{%
870     \glsnnoxindywarning\GlsAddXdyStyle}
871 \fi
```

\GlsSetXdyStyles Reset the list of required styles

```
872 \ifglxindy
873   \newcommand*\GlsSetXdyStyles[1]{%
```

```

874 \edef\@xdyrequiredstyles{#1}}
875 \else
876 \newcommand*\GlsSetXdyStyles[1]{%
877 \glsnoxindywarning\GlsSetXdyStyles}
878 \fi

```

`\findrootlanguage` The root language name is required by xindy. This information is for makeglossaries to pass to xindy. Since `\language` only stores the regional dialect rather than the root language name, some trickery is required to determine the root language.

```

879 \ifglsxindy
880 \ifpackageloaded{babel}{%

Need to parse babel.sty to determine the root language. This code was provided by Enrico Gregorio.

881 \def\findrootlanguage{\begingroup
882 \escapechar=-1\relax

normalize \language to category 12 chars

883 \edef\language{%
884 \expandafter\string\csname\language\endcsname}%

disable babel.sty useless commands

885 \def\NeedsTeXFormat##1[##2]{}%
886 \def\ProvidesPackage##1[##2]{}%
887 \let\LdfInit\relax
888 \def\languageattribute##1##2{%

change the meaning of \DeclareOption
889 \def\DeclareOption##1##2{%

at \DeclareOption* we end

890 \ifx##1*\expandafter\endinput\else

else we build a string with the first argument

891 \edef\testlanguage{\expandafter\string\csname##1\endcsname}%

if \testlanguage and \language are the same we execute the second argument

892 \ifx\testlanguage\language##2\fi
893 \fi}

almost all options of babel are \input{<name>.ldf}

894 \def\input##1{\stripldf##1}%

we put the root language name in \rootlanguage
895 \def\stripldf##1.ldf{\gdef\rootlanguage{##1}}%

now input babel.sty, using the primitive \input

896 \@@input babel.sty
897 \endgroup}%
898 }{%

```

hasn't been loaded, so check if has been loaded

```
899 \ifpackageloaded{ngerman}{%  
900 \def\findrootlanguage{%  
901 \def\rootlanguage{german}}}%  
902 }{%
```

Neither babel nor ngerman have been loaded, so assume the root language is English

```
903 \def\findrootlanguage{%  
904 \def\rootlanguage{english}}}%  
905 }%  
906 }%  
907 \fi
```

`\rootlanguage` Set default root language to English.

```
908 \def\rootlanguage{english}
```

`\@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.

```
909 \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.

```
910 \ifglxindy  
911 \newcommand*\GlsSetXdyLanguage[2][\glsdefaulttype]{%  
912 \ifglossaryexists{#1}{%  
913 \expandafter\def\csname @xdy@#1@language\endcsname{#2}%  
914 }{%  
915 \PackageError{glossaries}{Can't set language type for  
916 glossary type '#1' --- no such glossary}{%  
917 You have specified a glossary type that doesn't exist}}}  
918 \else  
919 \newcommand*\GlsSetXdyLanguage[2][]{%  
920 \glsnoxywarning\GlsSetXdyLanguage}  
921 \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.

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

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

```
923 \ifglxindy  
924 \newcommand*\GlsSetXdyCodePage[1]{%
```

```

925 \renewcommand*{\gls@codepage}{#1}%
926 }
927 \else
928 \newcommand*{\GlsSetXdyCodePage}[1]{%
929 \glsnoxindywarning\GlsSetXdyCodePage}
930 \fi

```

\@xdylettergroups Store letter group definitions.

```

931 \ifglxsindy
932 \ifglxsindy@glnumbers
933 \def\@xdylettergroups{(define-letter-group
934 \string"glnumbers\string"^^J\space\space\space
935 :prefixes (\string"0\string" \string"1\string"
936 \string"2\string" \string"3\string" \string"4\string"
937 \string"5\string" \string"6\string" \string"7\string"
938 \string"8\string" \string"9\string")^^J\space\space\space
939 :before \string"@glfirstletter\string")}
940 \else
941 \def\@xdylettergroups{}
942 \fi
943 \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.

```

944 \newcommand*\GlsAddLetterGroup[2]{%
945 \expandafter\toks@\expandafter{\@xdylettergroups}%
946 \protected@edef\@xdylettergroups{\the\toks@^^J%
947 (define-letter-group \string"#1\string"^^J\space\space\space#2)}}%
948 }%

```

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.

```

949 \newcommand*\forallglossaries[3][\@glo@types]{%
950 \@for#2:=#1\do{\ifx#2\@empty\else#3\fi}%
951 }

```

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

```
\forglentries[<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.

```

952 \newcommand*\forlgsentries}[3][\glsdefaulttype]{%
953   \edef\@glo@list{\csname glolist@#1\endcsname}%
954   \@for#2:=\@glo@list\do{\ifx#2\@empty\else#3\fi}%
955 }

```

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

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

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

```

956 \newcommand*\foralllgsentries}[3][\@glo@types]{%
957 \expandafter\foralllglossaries\expandafter[#1]{\@thisglo@}{%
958 \forlgsentries[\@thisglo@]{#2}{#3}}

```

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

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

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

```

959 \newcommand{\ifglossaryexists}[3]{%
960   \ifcsundef{glo@#1@out}{#3}{#2}%
961 }

```

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

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

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

```

962 \newcommand{\iflgsentryexists}[3]{%
963   \ifcsundef{glo@#1@name}{#3}{#2}%
964 }

```

`\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>*.

```
965 \newcommand*\ifglsused}[3]{\ifthenelse{\boolean{glo@#1@flag}}{#2}{#3}}
```

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

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

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

```
966 \newcommand{\glsdoifexists}[2]{%
```

```

967 \ifglstryexists{#1}{#2}{%
968   \PackageError{glossaries}{Glossary entry ‘#1’ has not been
969   defined}{You need to define a glossary entry before you
970   can use it.}}%
971 }

```

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

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

```

972 \newcommand{\glsdoifnoexists}[2]{%
973   \ifglstryexists{#1}{%
974     \PackageError{glossaries}{Glossary entry ‘#1’ has already
975     been defined}{}}{#2}%
976 }

```

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

```

977 \newcommand{\ifglshaschildren}[3]{%
978   \glsdoifexists{#1}%
979   {%
980     \def\do@glshaschildren{#3}%
981     \expandafter\for@glstry\expandafter[\csname glo@#1@type\endcsname]
982     {\glo@label}%
983     {%
984       \letcs\glo@parent{glo@\glo@label @parent}%
985       \ifthenelse{\equal{#1}{\glo@parent}}{%
986         {%
987           \def\do@glshaschildren{#2}%
988           \@endfortrue
989         }%
990         {}%
991       }%
992       \do@glshaschildren
993     }%
994 }

```

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

```

995 \newcommand{\ifglshasparent}[3]{%
996   \glsdoifexists{#1}%
997   {%
998     \ifcempty{glo@#1@parent}{#3}{#2}%
999   }%
1000 }

```

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`

```
1001 \newcommand*{\@glo@types}{,}
```

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`

```
1002 \newcommand*{\newglossary}[5][glg]{%
1003 \ifglossaryexists{#2}{%
1004   \PackageError{glossaries}{Glossary type ‘#2’ already exists}{%
1005     You can’t define a new glossary called ‘#2’ because it already
1006     exists}%
1007 }{%
```

Check if default has been set

```
1008   \ifx\glsdefaulttype\relax
1009     \gdef\glsdefaulttype{#2}%
1010   \fi
```

Add this to the list of glossary types:

```
1011   \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.

```
1012   \expandafter\gdef\csname glolist@#2\endcsname{,}%
```

Store details of this new glossary type:

```
1013   \expandafter\def\csname @glotype@#2@in\endcsname{#3}%
1014   \expandafter\def\csname @glotype@#2@out\endcsname{#4}%
1015   \expandafter\def\csname @glotype@#2@title\endcsname{#5}%
1016   \protected@write\@auxout{}{\string\@newglossary{#2}{#1}{#3}{#4}}%
```

How to display this entry in the document text (uses `\glsdisplay` and `\glsdisplayfirst` by default). These can be redefined by the user later if required (see `\defglsdisplay` and `\defglsdisplayfirst`). These may already have been defined if this has been specified as a list of acronyms.

```
1017   \ifcsundef{gls@#2@display}%
1018   {%
1019     \expandafter\gdef\csname gls@#2@display\endcsname{\glsdisplay}%
```

```

1020 }%
1021 {}%
1022 \ifcsundef{gls@#2@displayfirst}%
1023 {%
1024   \expandafter\gdef\csname gls@#2@displayfirst\endcsname{%
1025     \glsdisplayfirst
1026   }%
1027 }%
1028 {}%

```

Define sort counter if required:

```

1029 \@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.)

```

1030 \@ifnextchar[{\@gls@setcounter{#2}}%
1031   {\@gls@setcounter{#2}[\glscounter]}}%
1032 }

```

`\altnewglossary`

```

1033 \newcommand*{\altnewglossary}[3]{%
1034   \newglossary[#2-glg]{#1}{#2-gls}{#2-glo}{#3}%
1035 }

```

Only define new glossaries in the preamble:

```

1036 \@onlypreamble{\newglossary}

```

Only define new glossaries before `\makeglossaries`

```

1037 \@onlypremakeg\newglossary

```

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

`\@newglossary`

```

1038 \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`

```

1039 \def\@gls@setcounter#1[#2]{%
1040   \expandafter\def\csname @glotype@#1@counter\endcsname{#2}%

```

Add counter to xindy list, if not already added:

```

1041   \ifglsxindy
1042     \GlsAddXdyCounters{#2}%
1043   \fi
1044 }

```

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

`\@gls@getcounter`

```
1045 \newcommand*{\@gls@getcounter}[1]{%
1046 \csname @glo@type@#1@counter\endcsname}
```

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

```
1047 \glsdefmain
```

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.

```
1048 \define@key{glossentry}{name}{%
1049 \def\@glo@name{#1}%
1050 }
```

description The description key is usually only used in the glossary, but can be made to appear in the text by redefining `\glsdisplay` and `\glsdisplayfirst` (or using `\defglsdisplay` and `\defglsdisplayfirst`), however, you will have to disable the `sanitize` option (using the `sanitize` package option, `sanitize={description=false}`, and protect fragile commands). The description key is required when defining a new glossary entry. (Be careful not to make the description too long, because `makeindex` has a limited buffer. `\@glo@desc` is defined to be a short command to discourage lengthy descriptions for this reason. If you do have a very long description, or if you require paragraph breaks, define a separate command that contains the description, and use it as the value to the description key.)

```
1051 \define@key{glossentry}{description}{%
1052 \def\@glo@desc{#1}%
1053 }
```

descriptionplural

```
1054 \define@key{glossentry}{descriptionplural}{%
1055 \def\@glo@descplural{#1}%
1056 }
```

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>*.

```
1057 \define@key{glossentry}{sort}{%
1058 \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.

```
1059 \define@key{glossentry}{text}{%
1060 \def\@glo@text{#1}%
1061 }
```

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.

```
1062 \define@key{glossentry}{plural}{%
1063 \def\@glo@plural{#1}%
1064 }
```

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.

```
1065 \define@key{glossentry}{first}{%
1066 \def\@glo@first{#1}%
1067 }
```

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.

```
1068 \define@key{glossentry}{firstplural}{%
1069 \def\@glo@firstplural{#1}%
1070 }
```

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 `\glossaryentryfield` so that it uses its fourth parameter. If you want this value to appear in the text when the term is used by commands like `\gls`, you will need to change `\glsdisplay` and `\glsdisplayfirst` (either explicitly for all glossaries or via `\defglsdisplay` and `\defglsdisplayfirst` for individual glossaries).

```
1071 \define@key{glossentry}{symbol}{%
1072 \def\@glo@symbol{#1}%
1073 }
```

symbolplural

```
1074 \define@key{glossentry}{symbolplural}{%
1075 \def\@glo@symbolplural{#1}%
1076 }
```

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

```
1077 \define@key{glossentry}{type}{%
1078 \def\@glo@type{#1}}
```

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

```
1079 \define@key{glossentry}{counter}{%
1080 \ifcsundef{c@#1}%
1081 {%
1082 \PackageError{glossaries}%
1083 {There is no counter called ‘#1’}%
1084 {%
1085 The counter key should have the name of a valid counter
1086 as its value%
1087 }%
1088 }%
1089 {%
1090 \def\@glo@counter{#1}%
1091 }%
1092 }
```

see The see key specifies a list of cross-references

```
1093 \define@key{glossentry}{see}{%
1094 \def\@glo@see{#1}%
1095 \@glo@seeautonumberlist
1096 }
```

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

```
1097 \define@key{glossentry}{parent}{%
1098 \def\@glo@parent{#1}}
```

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

```
1099 \define@choicekey{glossentry}{nonumberlist}[\val\nr]{true,false}[true]{%
1100 \ifcase\nr\relax
1101 \def\@glo@prefix{\glsnonextpages}%
1102 \else
1103 \def\@glo@prefix{\glsnextpages}%
1104 \fi
1105 }
```

Define some generic user keys. (6 ought to be enough!)

user1

```
1106 \define@key{glossentry}{user1}{%  
1107   \def\@glo@useri{#1}%  
1108 }
```

user2

```
1109 \define@key{glossentry}{user2}{%  
1110   \def\@glo@userii{#1}%  
1111 }
```

user3

```
1112 \define@key{glossentry}{user3}{%  
1113   \def\@glo@useriii{#1}%  
1114 }
```

user4

```
1115 \define@key{glossentry}{user4}{%  
1116   \def\@glo@useriv{#1}%  
1117 }
```

user5

```
1118 \define@key{glossentry}{user5}{%  
1119   \def\@glo@userv{#1}%  
1120 }
```

user6

```
1121 \define@key{glossentry}{user6}{%  
1122   \def\@glo@uservi{#1}%  
1123 }
```

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.

```
1124 \define@key{glossentry}{short}{%  
1125   \def\@glo@short{#1}%  
1126 }
```

shortplural This key is provided for use by \newacronym.

```
1127 \define@key{glossentry}{shortplural}{%  
1128   \def\@glo@shortpl{#1}%  
1129 }
```

long This key is provided for use by \newacronym.

```
1130 \define@key{glossentry}{long}{%  
1131   \def\@glo@long{#1}%  
1132 }
```

longplural This key is provided for use by \newacronym.

```
1133 \define@key{glossentry}{longplural}{%
1134   \def\@glo@longpl{#1}%
1135 }
```

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

```
1136 \newcommand*\@glsnname{%
1137   \PackageError{glossaries}{name key required in
1138   \string\newglossaryentry\space for entry '\@glo@label'}{You
1139   haven't specified the entry name}}
```

\@glsdefaultplural Define command to set default plural.

```
1140 \newcommand*\@glsdefaultplural{\@glo@text\glspluralsuffix}
```

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

```
1141 \newcommand*\@gls@missingnumberlist[1]{%
1142   ??%
1143   \ifglssavenumberlist
1144     \GlossariesWarning{Missing number list for entry '#1'.
1145     Maybe makeglossaries + rerun required.}%
1146   \else
1147     \PackageError{glossaries}%
1148     {Package option 'savenumberlist=true' required.}%
1149     {%
1150     You must use the 'savenumberlist' package option
1151     to reference location lists.%
1152     }%
1153   \fi
1154 }
```

\@glsdefaultsort Define command to set default sort.

```
1155 \newcommand*\@glsdefaultsort{\@glo@name}
```

\gls@level Register to increment entry levels.

```
1156 \newcount\gls@level
```

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

```
1157 \newrobustcmd{\newglossaryentry}[2]{%
```

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

```
1158   \glsdoifnoexists{#1}%
1159   {%
```

Store label

```
1160   \def\@glo@label{#1}%
```

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

```
1161   \let\@glo@name\@glsnname
```

```

1162 \def\@glo@desc{%
1163     \PackageError{glossaries}
1164     {%
1165         description key required in \string\newglossaryentry\space
1166         for entry '\@glo@label'%
1167     }%
1168     {%
1169         You haven't specified the entry description%
1170     }%
1171 }%

1172 \def\@glo@descplural{\@glo@desc}%
1173 \def\@glo@type{\glsdefaulttype}%
1174 \def\@glo@symbol{\relax}%

1175 \def\@glo@symbolplural{\@glo@symbol}%
1176 \def\@glo@text{\@glo@name}%
1177 \let\@glo@plural\glsdefaultplural

    Using \let instead of \def to make later comparison avoid expansion issues.
    (Thanks to Ulrich Diez for suggesting this.)

1178 \let\@glo@first\relax

1179 \let\@glo@firstplural\relax

    Set the default sort:

1180 \let\@glo@sort\glsdefaultsort

    Set the default counter:

1181 \def\@glo@counter{\@gls@getcounter{\@glo@type}}%

1182 \def\@glo@see{}%

1183 \def\@glo@parent{}%

1184 \def\@glo@prefix{}%

1185 \def\@glo@useri{}%
1186 \def\@glo@userii{}%
1187 \def\@glo@useriii{}%
1188 \def\@glo@useriv{}%
1189 \def\@glo@userv{}%
1190 \def\@glo@uservi{}%

1191 \def\@glo@short{}%
1192 \def\@glo@shortpl{}%
1193 \def\@glo@long{}%
1194 \def\@glo@longpl{}%

```


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

```

1195 \newglossaryentryprehook

```

Extract key-val information from third parameter:

```

1196 \setkeys{glossentry}{#2}%

```

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

```

1197 \ifcsundef{glolist@\@glo@type}%
1198 {%
1199   \PackageError{glossaries}%
1200   {Glossary type '\@glo@type' has not been defined}%
1201   {You need to define a new glossary type, before making entries
1202    in it}%
1203 }%
1204 {%
1205   \protected@edef\@glolist@{\csname glolist@\@glo@type\endcsname}%
1206   \expandafter\xdef\csname glolist@\@glo@type\endcsname{\@glolist@{#1},}%
1207 }%

```

Initialise level to 0.

```

1208 \gls@level=0\relax

```

Has this entry been assigned a parent?

```

1209 \ifx\@glo@parent\@empty

```

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

```

1210 \expandafter\gdef\csname glo@#1@parent\endcsname{}%
1211 \else

```

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

```

1212 \ifthenelse{\equal{#1}{\@glo@parent}}%
1213 {%
1214   \PackageError{glossaries}{Entry '#1' can't be its own parent}{}%
1215   \def\@glo@parent{}%
1216   \expandafter\gdef\csname glo@#1@parent\endcsname{}%
1217 }%
1218 {%

```

Check the parent exists:

```

1219 \ifglsentryexists{\@glo@parent}%
1220 {%

```

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

```

1221 \expandafter\xdef\csname glo@#1@parent\endcsname{\@glo@parent}%

```

Determine level.

```

1222 \gls@level=\csname glo@\@glo@parent @level\endcsname\relax
1223 \advance\gls@level by 1\relax

```

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

```

1224 \ifx\@glo@name\@glsnoname
1225   \expandafter\let\expandafter\@glo@name
1226   \csname glo@\@glo@parent @name\endcsname

```

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

```

1227         \ifx\@glo@plural\@glsdefaultplural
1228         \expandafter\let\expandafter\@glo@plural
1229             \csname glo@\@glo@parent @plural\endcsname
1230         \fi
1231     \fi
1232 }%
1233 {%

```

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

```

1234     \PackageError{glossaries}%
1235     {%
1236         Invalid parent '@glo@parent'
1237         for entry '#1' - parent doesn't exist%
1238     }%
1239     {%
1240         Parent entries must be defined before their children%
1241     }%
1242     \def\@glo@parent{%
1243     \expandafter\gdef\csname glo@#1@parent\endcsname{%
1244     }%
1245     }%
1246 \fi

```

Set the level for this entry

```

1247 \expandafter\xdef\csname glo@#1@level\endcsname{\number\gls@level}%

```

Check if first and firstplural have been use. If firstplural hasn't been specified, but first has been specified, then form firstplural by appending \glspluralsuffix to value of first key, otherwise obtain the value from the plural key. This now uses \ifx instead of \if to avoid expansion issues. (Thanks to Ulrich Diez for suggesting this.)

```

1248 \ifx\relax\@glo@firstplural
1249     \ifx\relax\@glo@first
1250         \def\@glo@firstplural{\@glo@plural}%
1251         \def\@glo@first{\@glo@text}%
1252     \else
1253         \def\@glo@firstplural{\@glo@first\glspluralsuffix}%
1254     \fi
1255 \else
1256     \ifx\relax\@glo@first
1257         \def\@glo@first{\@glo@text}%
1258     \fi
1259 \fi

```

Define commands associated with this entry:

```

1260 \expandafter
1261 \protected@xdef\csname glo@#1@text\endcsname{\@glo@text}%
1262 \expandafter

```

```

1263 \protected@xdef\csname glo@#1@plural\endcsname{\@glo@plural}%
1264 \expandafter
1265 \protected@xdef\csname glo@#1@first\endcsname{\@glo@first}%
1266 \expandafter
1267 \protected@xdef\csname glo@#1@firstpl\endcsname{\@glo@firstplural}%
1268 \expandafter
1269 \protected@xdef\csname glo@#1@type\endcsname{\@glo@type}%
1270 \expandafter
1271 \protected@xdef\csname glo@#1@counter\endcsname{\@glo@counter}%
1272 \expandafter
1273 \protected@xdef\csname glo@#1@useri\endcsname{\@glo@useri}%
1274 \expandafter
1275 \protected@xdef\csname glo@#1@userii\endcsname{\@glo@userii}%
1276 \expandafter
1277 \protected@xdef\csname glo@#1@useriii\endcsname{\@glo@useriii}%
1278 \expandafter
1279 \protected@xdef\csname glo@#1@useriv\endcsname{\@glo@useriv}%
1280 \expandafter
1281 \protected@xdef\csname glo@#1@userv\endcsname{\@glo@userv}%
1282 \expandafter
1283 \protected@xdef\csname glo@#1@uservi\endcsname{\@glo@uservi}%
1284 \expandafter
1285 \protected@xdef\csname glo@#1@short\endcsname{\@glo@short}%
1286 \expandafter
1287 \protected@xdef\csname glo@#1@shortpl\endcsname{\@glo@shortpl}%
1288 \expandafter
1289 \protected@xdef\csname glo@#1@long\endcsname{\@glo@long}%
1290 \expandafter
1291 \protected@xdef\csname glo@#1@longpl\endcsname{\@glo@longpl}%
1292 \@gls@sanitizename
1293 \expandafter\protected@xdef\csname glo@#1@name\endcsname{\@glo@name}%

```

Set default numberlist if not defined:

```

1294 \ifcsundef{glo@#1@numberlist}%
1295 {%
1296 \csxdef{glo@#1@numberlist}{\noexpand\@gls@missingnumberlist{\@glo@label}}%
1297 }%
1298 {}%

```

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.

```

1299 \def\@glo@@desc{\@glo@first}%
1300 \ifx\@glo@desc\@glo@@desc
1301 \let\@glo@desc\@glo@first
1302 \fi
1303 \@gls@sanitizedesc
1304 \expandafter\protected@xdef\csname glo@#1@desc\endcsname{\@glo@desc}%
1305 \expandafter\protected@xdef\csname glo@#1@descplural\endcsname{\@glo@descplural}%

```

Set the sort key for this entry:

```

1306 \gls@defsort{\@glo@type}{#1}%
1307 \def\@glo@@symbol{\@glo@text}%
1308 \ifx\@glo@symbol\@glo@@symbol
1309 \let\@glo@symbol\@glo@text
1310 \fi
1311 \gls@sanitizesymbol
1312 \expandafter\protected@xdef\csname glo@#1@symbol\endcsname{\@glo@symbol}%
1313 \expandafter\protected@xdef\csname glo@#1@symbolplural\endcsname{\@glo@symbolplural}%

```

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

```

1314 \expandafter\gdef\csname glo@#1@flagfalse\endcsname{%
1315 \expandafter\global\expandafter
1316 \let\csname ifglo@#1@flag\endcsname\iffalse
1317 }%
1318 \expandafter\gdef\csname glo@#1@flagtrue\endcsname{%
1319 \expandafter\global\expandafter
1320 \let\csname ifglo@#1@flag\endcsname\iftrue
1321 }%
1322 \csname glo@#1@flagfalse\endcsname

```

Sort out any cross-referencing if required.

```

1323 \ifx\@glo@see\@empty
1324 \else
1325 \protected@edef\@do@glsee{%
1326 \noexpand\@gls@fixbraces\noexpand\@glo@list\@glo@see
1327 \noexpand\@nil
1328 \noexpand\expandafter\noexpand\@glsee\noexpand\@glo@list{#1}}%
1329 \do@glsee
1330 \fi
1331 }%

```

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

```

1332 \do@glo@storeentry{#1}%

```

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

```

1333 \@newglossaryentryposthook
1334 }

```

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

```

1335 \newcommand*{\@newglossaryentryprehook}{}

```

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

```

1336 \newcommand*{\@newglossaryentryposthook}{}

```

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

```

1337 \newcommand*{\glsmoveentry}[2]{%

```

```

1338 \edef\glo@type{\csname glo@#1@type\endcsname}%
1339 \def\glo@list{,}%
1340 \forlslentries[\glo@type]{\glo@label}%
1341 {%
1342     \ifthenelse{\equal{\glo@label}{#1}}{\}\{\eappto\glo@list{\glo@label,}}%
1343 }%
1344 \cslet{glolist@\glo@type}{\glo@list}%
1345 \csdef{glo@#1@type}{#2}%
1346 }

```

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

```

1347 \ifglxindy
1348   \newcommand*{\@glossaryentryfield}{\string\@glossaryentryfield}
1349 \else
1350   \newcommand*{\@glossaryentryfield}{\string\glossaryentryfield}
1351 \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.)

```

1352 \ifglxindy
1353   \newcommand*{\@glossarysubentryfield}{%
1354     \string\@glossarysubentryfield}
1355 \else
1356   \newcommand*{\@glossarysubentryfield}{%
1357     \string\glossarysubentryfield}
1358 \fi

```

\@glo@storeentry Determine the format to write the entry in the glossary output (`.glo`) file. The argument is the entry's label. The result is stored in `\glo@<label>@entry`, where `<label>` is the entry's label. (This doesn't include any formatting or location information.)

```

1359 \newcommand{\@glo@storeentry}[1]{%
  Get the sort string and escape any special characters
1360 \protected@edef\@glo@sort{\csname glo@#1@sort\endcsname}%
1361 \@gls@checkmkidxchars\@glo@sort
  Same again for the name string.
1362 \protected@edef\@glo@name{\csname glo@#1@name\endcsname}%
1363 \@gls@checkmkidxchars\@glo@name
  Add the font command. (The backslash needs to be escaped for xindy.)
1364 \ifglxindy
1365   \protected@edef\@glo@name{\string\@glsnamefont{\@glo@name}}%
1366 \else
1367   \protected@edef\@glo@name{\string\glsnamefont{\@glo@name}}%
1368 \fi

```

Get the description string and escape any special characters

```
1369 \protected@edef\@glo@desc{\csname glo@#1@desc\endcsname}%  
1370 \@gls@checkmkidxchars\@glo@desc
```

Same again for the symbol

```
1371 \protected@edef\@glo@symbol{\csname glo@#1@symbol\endcsname}%  
1372 \@gls@checkmkidxchars\@glo@symbol
```

Escape any special characters in the prefix

```
1373 \@gls@checkmkidxchars\@glo@prefix
```

Get the parent, if one exists

```
1374 \edef\@glo@parent{\csname glo@#1@parent\endcsname}%
```

Write the information to the glossary file.

```
1375 \ifglxindy
```

Store using xindy syntax.

```
1376 \ifx\@glo@parent\@empty
```

Entry doesn't have a parent

```
1377 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%  
1378 (\string"\@glo@sort\string" %  
1379 \string"\@glo@prefix\@glossaryentryfield{#1}\@glo@name  
1380 }\@glo@desc}\@glo@symbol}\string") %  
1381 }%  
1382 \else
```

Entry has a parent

```
1383 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%  
1384 \csname glo@\@glo@parent @index\endcsname  
1385 (\string"\@glo@sort\string" %  
1386 \string"\@glo@prefix\@glossarysubentryfield%  
1387 {\csname glo@#1@level\endcsname}{#1}\@glo@name  
1388 }\@glo@desc}\@glo@symbol}\string") %  
1389 }%  
1390 \fi  
1391 \else
```

Store using makeindex syntax.

```
1392 \ifx\@glo@parent\@empty
```

Sanitize \@glo@prefix

```
1393 \@onelevel@sanitize\@glo@prefix
```

Entry doesn't have a parent

```
1394 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%  
1395 \@glo@sort\@gls@actualchar\@glo@prefix  
1396 \@glossaryentryfield{#1}\@glo@name}\@glo@desc  
1397 }\@glo@symbol}%  
1398 }%  
1399 \else
```

Entry has a parent

```
1400 \expandafter\protected\edef\csname glo@#1@index\endcsname{%
1401 \csname glo@\@glo@parent @index\endcsname\@gls@levelchar
1402 \@glo@sort\@gls@actualchar\@glo@prefix
1403 \@glossarysubentryfield
1404 {\csname glo@#1@level\endcsname}{#1}{\@glo@name}{\@glo@desc
1405 }\@glo@symbol}%
1406 }%
1407 \fi
1408 \fi
1409 }
```

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:

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.

`\glsreset`

```
1410 \newcommand*\glsreset}[1]{%
1411 \glsdoifexists{#1}{%
1412 \expandafter\global\csname glo@#1@flagfalse\endcsname}}
```

As above, but with only a local effect:

`\glslocalreset`

```
1413 \newcommand*\glslocalreset}[1]{%
1414 \glsdoifexists{#1}{%
1415 \expandafter\let\csname ifglo@#1@flag\endcsname\iffalse}}
```

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.

`\glsunset`

```
1416 \newcommand*\glsunset}[1]{%
1417 \glsdoifexists{#1}{%
1418 \expandafter\global\csname glo@#1@flagtrue\endcsname}}
```

As above, but with only a local effect:

`\glslocalunset`

```
1419 \newcommand*\glslocalunset}[1]{%
1420 \glsdoifexists{#1}{%
1421 \expandafter\let\csname ifglo@#1@flag\endcsname\iftrue}}
```

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

Syntax: `\glsresetall[<glossary-list>]`

`\glsresetall`

```
1422 \newcommand*{\glsresetall}[1][\@glo@types]{%
1423 \forallglsentries[#1]{\@glsentry}{%
1424 \glsreset{\@glsentry}}}
```

As above, but with only a local effect:

`\glslocalresetall`

```
1425 \newcommand*{\glslocalresetall}[1][\@glo@types]{%
1426 \forallglsentries[#1]{\@glsentry}{%
1427 \glslocalreset{\@glsentry}}}
```

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

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

`\glsunsetall`

```
1428 \newcommand*{\glsunsetall}[1][\@glo@types]{%
1429 \forallglsentries[#1]{\@glsentry}{%
1430 \glsunset{\@glsentry}}}
```

As above, but with only a local effect:

`\glslocalunsetall`

```
1431 \newcommand*{\glslocalunsetall}[1][\@glo@types]{%
1432 \forallglsentries[#1]{\@glsentry}{%
1433 \glslocalunset{\@glsentry}}}
```

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`

```
1434 \newcommand*{\loadglsentries}[2][\@gls@default]{%
1435 \let\@gls@default\glsdefaulttype
1436 \def\glsdefaulttype{#1}\input{#2}%
1437 \let\glsdefaulttype\@gls@default}
```

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

```
1438 \@onlypreamble{\loadglsentries}
```

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

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

`\glstextformat`

```
1439 \newcommand*{\glstextformat}[1]{#1}
```

The first time an entry is used, the way in which it is displayed is governed by `\glsdisplayfirst`. This takes four parameters: #1 will be the value of the entry’s first or firstplural key, #2 will be the value of the entry’s description key, #3 will be the value of the entry’s symbol key and #4 is additional text supplied by the final optional argument to commands like `\gls` and `\glspl`. The default is to display the first parameter followed by the additional text.

`\glsdisplayfirst`

```
1440 \newcommand*{\glsdisplayfirst}[4]{#1#4}
```

After the first use, the entry is displayed according to the format of `\glsdisplay`. Again, it takes four parameters: #1 will be the value of the entry’s text or plural key, #2 will be the value of the entry’s description key, #3 will be the value of the entry’s symbol key and #4 is additional text supplied by the final optional argument to commands like `\gls` and `\glspl`.

`\glsdisplay`

```
1441 \newcommand*{\glsdisplay}[4]{#1#4}
```

When a new glossary is created it uses `\glsdisplayfirst` and `\glsdisplay` as the default way of displaying its entry in the text. This can be changed for the entries belonging to an individual glossary using `\defglsdisplay` and `\defglsdisplayfirst`.

`\defglsdisplay[⟨type⟩]{⟨definition⟩}`

The glossary type is given by `⟨type⟩` (the default glossary if omitted) and `⟨definition⟩` should have at most #1, #2, #3 and #4. These represent the same arguments as those described for `\glsdisplay`.

`\defglsdisplay`

```
1442 \newcommand*{\defglsdisplay}[2][\glsdefaulttype]{%
1443 \expandafter\def\csname gls@#1@display\endcsname##1##2##3##4{#2}}
```

`\defglsdisplayfirst[⟨type⟩]{⟨definition⟩}`

The glossary type is given by `⟨type⟩` (the default glossary if omitted) and `⟨definition⟩` should have at most #1, #2, #3 and #4. These represent the same arguments as those described for `\glsdisplayfirst`.

`\defglsdisplayfirst`

```
1444 \newcommand*{\defglsdisplayfirst}[2][\glsdefaultttype]{%
1445 \expandafter\def\csname gls@#1@displayfirst\endcsname##1##2##3##4{#2}}
```

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`, 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 `\defglsdisplay` and `\defglsdisplayfirst`). It goes against the L^AT_EX 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.

```
1446 \define@key{glslink}{counter}{%
1447   \ifcsundef{c@#1}%
1448   {%
1449     \PackageError{glossaries}%
1450     {There is no counter called ‘#1’}%
1451     {%
1452       The counter key should have the name of a valid counter
1453       as its value%
1454     }%
1455   }%
1456   {%
1457     \def\@gls@counter{#1}%
1458   }%
1459 }
```

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.

```
1460 \define@key{glslink}{format}{%
1461 \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.

```
1462 \define@boolkey{glslink}{hyper}[true]{}
```

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.

```
1463 \define@boolkey{glslink}{local}[true]{}
```

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 whether or not we are using the starred version:

```
\glslink
```

```
1464 \newrobustcmd*{\glslink}{%
```

```
1465 \@ifstar\@sgls@link\@gls@@link}
```

```
\@sgls@link The starred version of \glslink calls the unstarred version with hyperlinks disabled.
```

```
1466 \newcommand*{\@sgls@link}[1][\@gls@@link[hyper=false,#1]]
```

```
\@gls@@link The unstarred version of \glslink checks for the existence of the term. 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.
```

```
1467 \newcommand*{\@gls@@link}[3][\@gls@link]{%
```

```
1468 \ifglsentryexists{#2}%
```

```
1469 {%
```

```
1470 \@gls@link[#1]{#2}{#3}%
```

```
1471 }%
```

```
1472 \PackageError{glossaries}{Glossary entry ‘#2’ has not been
1473 defined}{You need to define a glossary entry before you
1474 can use it.}%
```

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

```
1475 \glstextformat{#3}%
```

```
1476 }%
```

```
1477 }
```

```
\@gls@link
```

```
1478 \def\@gls@link[#1]#2#3{%
```

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

```
1479 \leavevmode
1480 \def\glslabel{#2}%
1481 \def\@glsnumberformat{glsnumberformat}%
1482 \edef\@gls@counter{\csname glo@#2@counter\endcsname}%
```

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

```
1483 \expandafter\xifinlist\expandafter
1484   {\csname glo@#2@type\endcsname}{\@gls@nohyperlist}%
1485 {%
1486   \KV@glslink@hyperfalse
1487 }%
1488 {%
1489   \KV@glslink@hypertrue
1490 }%
1491 \setkeys{glslink}{#1}%
```

Store the entry’s counter in `\theglsentrycounter`

```
1492 \@gls@saveentrycounter
```

Define sort key if necessary:

```
1493 \@gls@setsort{#2}%

1494 \@do@wrglossary{#2}%
1495 \ifKV@glslink@hyper
1496   \@glslink{\glolinkprefix#2}{\glstextformat{#3}}%
1497 \else
1498   \glstextformat{#3}\relax
1499 \fi
1500 }
```

`\glolinkprefix`

```
1501 \newcommand*{\glolinkprefix}{glo:}
```

`\glsentrycounter` Set default value of entry counter

```
1502 \def\glsentrycounter{\glscounter}%
```

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

```
1503 \newcommand*{\@gls@saveentrycounter}{%
1504   \def\@gls@Hcounter{}%

   Are we using equation counter?
1505   \ifthenelse{\equal{\@gls@counter}{equation}}{%
1506     {
```

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

```
1507   \ifcsundef{xatlevel@}%
```

```

1508   {%
1509     \edef\theglentrycounter{\expandafter\noexpand
1510       \csname the\@gls@counter\endcsname}%
1511   }%
1512   {%
1513     \ifx\xatlevel@\@empty
1514       \edef\theglentrycounter{\expandafter\noexpand
1515         \csname the\@gls@counter\endcsname}%
1516     \else
1517       \savecounters@
1518       \advance\c@equation by 1\relax
1519       \edef\theglentrycounter{\csname the\@gls@counter\endcsname}%

```

Check if hyperref version of this counter

```

1520     \ifcsundef{theH\@gls@counter}%
1521     {%
1522       \def\@gls@Hcounter{\theglentrycounter}%
1523     }%
1524     {%
1525       \def\@gls@Hcounter{\csname theH\@gls@counter\endcsname}%
1526     }%
1527     \protected@edef\theHglentrycounter{\@gls@Hcounter}%
1528     \restorecounters@
1529   \fi
1530 }%
1531 }%
1532 {%

```

Not using equation counter so no special measures:

```

1533   \edef\theglentrycounter{\expandafter\noexpand
1534     \csname the\@gls@counter\endcsname}%
1535 }%

```

Check if hyperref version of this counter

```

1536   \ifx\@gls@Hcounter\@empty
1537     \ifcsundef{theH\@gls@counter}%
1538     {%
1539       \def\theHglentrycounter{\theglentrycounter}%
1540     }%
1541     {%
1542       \protected@edef\theHglentrycounter{\expandafter\noexpand
1543         \csname theH\@gls@counter\endcsname}%
1544     }%
1545   \fi
1546 }

```

`\@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.

```

1547 \def\@set@glo@numformat#1#2#3#4{%
1548   \expandafter\@glo@check@mkidxrangechar#3\@nil
1549   \protected@edef#1{%
1550     \@glo@prefix setentrycounter[#4]{#2}%
1551     \expandafter\string\curname\@glo@suffix\endcurname
1552   }%
1553   \@gls@checkmkidxchars#1%
1554 }

```

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.

```

1555 \def\@glo@check@mkidxrangechar#1#2\@nil{%
1556 \if#1(\relax
1557   \def\@glo@prefix{(%
1558   \if\relax#2\relax
1559     \def\@glo@suffix{glsnumberformat}%
1560   \else
1561     \def\@glo@suffix{#2}%
1562   \fi
1563 \else
1564   \if#1)\relax
1565     \def\@glo@prefix{)%
1566     \if\relax#2\relax
1567       \def\@glo@suffix{glsnumberformat}%
1568     \else
1569       \def\@glo@suffix{#2}%
1570     \fi
1571   \else
1572     \def\@glo@prefix{}\def\@glo@suffix{#1#2}%
1573   \fi
1574 \fi}

```

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

```

1575 \newcommand*\@gls@escbsdq[1]{%
1576   \def\@gls@checkedmkidx{%
1577     \let\gls@xdystring=#1\relax
1578     \@onelevel@sanitize\gls@xdystring
1579     \edef\do@gls@xdycheckbackslash{%
1580       \noexpand\@gls@xdycheckbackslash\gls@xdystring\noexpand\@nil
1581       \@backslashchar\@backslashchar\noexpand\null}%
1582     \do@gls@xdycheckbackslash
1583     \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdystring}%
1584     \def\@gls@checkedmkidx{%

```

```

1585 \expandafter\@gls@xdycheckquote\gls@xdystring\@nil""\null
1586 \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdystring}%
    Unsanitize \gls@numberpage, \gls@alphpage, \gls@Alphpage and \gls@romanpage
    (thanks to David Carlisle for the suggestion.)
1587 \@for\@gls@tmp:=\gls@protected@pagefmts\do
1588 {%
1589     \edef\@gls@sanitized@tmp{\expandafter\@gobble\string\\expandonce\@gls@tmp}%
1590     \@onelevel@sanitize\@gls@sanitized@tmp
1591     \edef\gls@dosubst{%
1592         \noexpand\DTLsubstituteall\noexpand\gls@xdystring
1593         {\@gls@sanitized@tmp}{\expandonce\@gls@tmp}%
1594     }%
1595     \gls@dosubst
1596 }%
    Assign to required control sequence
1597 \let#1=\gls@xdystring
1598 }

```

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

`\gls@checkmkidxchars`

```

1599 \newcommand{\@gls@checkmkidxchars}[1]{%
1600 \ifglxsindy
1601   \@gls@escbsdq{#1}%
1602 \else
1603   \def\@gls@checkedmkidx{}%
1604   \expandafter\@gls@checkquote#1\@nil""\null
1605   \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
1606   \def\@gls@checkedmkidx{}%
1607   \expandafter\@gls@checkescquote#1\@nil\""\null
1608   \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
1609   \def\@gls@checkedmkidx{}%
1610   \expandafter\@gls@checkescactual#1\@nil\?\?\null
1611   \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
1612   \def\@gls@checkedmkidx{}%
1613   \expandafter\@gls@checkactual#1\@nil??\null
1614   \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
1615   \def\@gls@checkedmkidx{}%
1616   \expandafter\@gls@checkbar#1\@nil||\null
1617   \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
1618   \def\@gls@checkedmkidx{}%
1619   \expandafter\@gls@checkesbar#1\@nil\\|\null
1620   \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
1621   \def\@gls@checkedmkidx{}%
1622   \expandafter\@gls@checklevel#1\@nil!!\null
1623   \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
1624 \fi
1625 }

```

Update the control sequence and strip trailing \@nil:

\@gls@updatechecked

```
1626 \def\@gls@updatechecked#1\@nil#2{\def#2{#1}}
```

\@gls@tmpb Define temporary token

```
1627 \newtoks\@gls@tmpb
```

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

```
1628 \def\@gls@checkquote#1"#2"#3\null{%
1629 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
1630 \toks@={#1}%
1631 \ifx\null#2\null
1632 \ifx\null#3\null
1633 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
1634 \def\@gls@checkquote{\relax}%
1635 \else
1636 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1637 \@gls@quotechar\@gls@quotechar\@gls@quotechar\@gls@quotechar}%
1638 \def\@gls@checkquote{\@gls@checkquote#3\null}%
1639 \fi
1640 \else
1641 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1642 \@gls@quotechar\@gls@quotechar}%
1643 \ifx\null#3\null
1644 \def\@gls@checkquote{\@gls@checkquote#2""\null}%
1645 \else
1646 \def\@gls@checkquote{\@gls@checkquote#2"#3\null}%
1647 \fi
1648 \fi
1649 \@gls@checkquote}
```

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

```
1650 \def\@gls@checkescquote#1\"#2\"#3\null{%
1651 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
1652 \toks@={#1}%
1653 \ifx\null#2\null
1654 \ifx\null#3\null
1655 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
1656 \def\@gls@checkescquote{\relax}%
1657 \else
1658 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1659 \@gls@quotechar\string\" \@gls@quotechar
1660 \@gls@quotechar\string\" \@gls@quotechar}%
1661 \def\@gls@checkescquote{\@gls@checkescquote#3\null}%
1662 \fi
1663 \else
1664 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1665 \@gls@quotechar\string\" \@gls@quotechar}%

```



```

1666 \ifx\null#3\null
1667   \def\@gls@checkescquote{\@gls@checkescquote#2\""\null}%
1668 \else
1669   \def\@gls@checkescquote{\@gls@checkescquote#2\"#3\null}%
1670 \fi
1671 \fi
1672 \@gls@checkescquote}

```

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

```

1673 \def\@gls@checkescactual#1\?#2\?#3\null{%
1674 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
1675 \toks@={#1}%
1676 \ifx\null#2\null
1677 \ifx\null#3\null
1678   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
1679   \def\@gls@checkescactual{\relax}%
1680 \else
1681   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1682     \@gls@quotechar\string\""\@gls@actualchar
1683     \@gls@quotechar\string\""\@gls@actualchar}%
1684   \def\@gls@checkescactual{\@gls@checkescactual#3\null}%
1685 \fi
1686 \else
1687   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1688     \@gls@quotechar\string\""\@gls@actualchar}%
1689 \ifx\null#3\null
1690   \def\@gls@checkescactual{\@gls@checkescactual#2\?\?\null}%
1691 \else
1692   \def\@gls@checkescactual{\@gls@checkescactual#2\?#3\null}%
1693 \fi
1694 \fi
1695 \@gls@checkescactual}

```

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

```

1696 \def\@gls@checkescbar#1\|#2\|#3\null{%
1697 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
1698 \toks@={#1}%
1699 \ifx\null#2\null
1700 \ifx\null#3\null
1701   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
1702   \def\@gls@checkescbar{\relax}%
1703 \else
1704   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1705     \@gls@quotechar\string\""\@gls@encapchar
1706     \@gls@quotechar\string\""\@gls@encapchar}%
1707   \def\@gls@checkescbar{\@gls@checkescbar#3\null}%
1708 \fi
1709 \else
1710   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@

```

```

1711 \@gls@quotechar\string\" \@gls@encapchar}%
1712 \ifx\@null#3\@null
1713 \def\@gls@checkescbar{\@gls@checkescbar#2\|\|\@null}%
1714 \else
1715 \def\@gls@checkescbar{\@gls@checkescbar#2\|#3\@null}%
1716 \fi
1717 \fi
1718 \@gls@checkescbar}

```

\@gls@checkesclevel Similarly for \!:

```

1719 \def\@gls@checkesclevel#1\!#2\!#3\@null{%
1720 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
1721 \toks@={#1}%
1722 \ifx\@null#2\@null
1723 \ifx\@null#3\@null
1724 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
1725 \def\@gls@checkesclevel{\relax}%
1726 \else
1727 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1728 \@gls@quotechar\string\" \@gls@levelchar
1729 \@gls@quotechar\string\" \@gls@levelchar}%
1730 \def\@gls@checkesclevel{\@gls@checkesclevel#3\@null}%
1731 \fi
1732 \else
1733 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1734 \@gls@quotechar\string\" \@gls@levelchar}%
1735 \ifx\@null#3\@null
1736 \def\@gls@checkesclevel{\@gls@checkesclevel#2\!\!\@null}%
1737 \else
1738 \def\@gls@checkesclevel{\@gls@checkesclevel#2\!#3\@null}%
1739 \fi
1740 \fi
1741 \@gls@checkesclevel}

```

\@gls@checkbar and for |:

```

1742 \def\@gls@checkbar#1|#2|#3\@null{%
1743 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
1744 \toks@={#1}%
1745 \ifx\@null#2\@null
1746 \ifx\@null#3\@null
1747 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
1748 \def\@gls@checkbar{\relax}%
1749 \else
1750 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1751 \@gls@quotechar\@gls@encapchar\@gls@quotechar\@gls@encapchar}%
1752 \def\@gls@checkbar{\@gls@checkbar#3\@null}%
1753 \fi
1754 \else
1755 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@

```

```

1756 \@gls@quotechar\@gls@encapchar}%
1757 \ifx\null#3\null
1758 \def\@gls@checkbar{\@gls@checkbar#2||\null}%
1759 \else
1760 \def\@gls@checkbar{\@gls@checkbar#2|#3\null}%
1761 \fi
1762 \fi
1763 \@gls@checkbar}

```

\@gls@checklevel and for !:

```

1764 \def\@gls@checklevel#1!#2!#3\null{%
1765 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
1766 \toks@={#1}%
1767 \ifx\null#2\null
1768 \ifx\null#3\null
1769 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
1770 \def\@gls@checklevel{\relax}%
1771 \else
1772 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1773 \@gls@quotechar\@gls@levelchar\@gls@quotechar\@gls@levelchar}%
1774 \def\@gls@checklevel{\@gls@checklevel#3\null}%
1775 \fi
1776 \else
1777 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1778 \@gls@quotechar\@gls@levelchar}%
1779 \ifx\null#3\null
1780 \def\@gls@checklevel{\@gls@checklevel#2!!\null}%
1781 \else
1782 \def\@gls@checklevel{\@gls@checklevel#2!#3\null}%
1783 \fi
1784 \fi
1785 \@gls@checklevel}

```

\@gls@checkactual and for ?:

```

1786 \def\@gls@checkactual#1?#2?#3\null{%
1787 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
1788 \toks@={#1}%
1789 \ifx\null#2\null
1790 \ifx\null#3\null
1791 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
1792 \def\@gls@checkactual{\relax}%
1793 \else
1794 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1795 \@gls@quotechar\@gls@actualchar\@gls@quotechar\@gls@actualchar}%
1796 \def\@gls@checkactual{\@gls@checkactual#3\null}%
1797 \fi
1798 \else
1799 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1800 \@gls@quotechar\@gls@actualchar}%

```

```

1801 \ifx\null#3\null
1802   \def\@gls@checkactual{\@gls@checkactual#2??\null}%
1803 \else
1804   \def\@gls@checkactual{\@gls@checkactual#2?#3\null}%
1805 \fi
1806 \fi
1807 \@gls@checkactual}

```

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

```

1808 \def\@gls@xdycheckquote#1"#2"#3\null{%
1809 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
1810 \toks@={#1}%
1811 \ifx\null#2\null
1812 \ifx\null#3\null
1813   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
1814   \def\@gls@xdycheckquote{\relax}%
1815 \else
1816   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1817     \string"\string"}%
1818   \def\@gls@xdycheckquote{\@gls@xdycheckquote#3\null}%
1819 \fi
1820 \else
1821   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1822     \string"}%
1823   \ifx\null#3\null
1824     \def\@gls@xdycheckquote{\@gls@xdycheckquote#2""\null}%
1825   \else
1826     \def\@gls@xdycheckquote{\@gls@xdycheckquote#2"#3\null}%
1827   \fi
1828 \fi
1829 \@gls@xdycheckquote
1830 }

```

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

```

\@gls@xdycheckbackslash
1831 \edef\def@gls@xdycheckbackslash{%
1832 \noexpand\def\noexpand\@gls@xdycheckbackslash##1\@backslashchar
1833   ##2\@backslashchar##3\noexpand\null{%
1834 \noexpand\@gls@tmpb=\noexpand\expandafter
1835   {\noexpand\@gls@checkedmkidx}%
1836 \noexpand\toks@={##1}%
1837 \noexpand\ifx\noexpand\null##2\noexpand\null
1838 \noexpand\ifx\noexpand\null##3\noexpand\null
1839 \noexpand\edef\noexpand\@gls@checkedmkidx{%
1840   \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
1841 \noexpand\def\noexpand\@gls@xdycheckbackslash{\relax}%
1842 \noexpand\else
1843 \noexpand\edef\noexpand\@gls@checkedmkidx{%
1844   \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@

```

```

1845 \backslashchar\backslashchar\backslashchar\backslashchar}%
1846 \noexpand\def\noexpand\@gls@xdycheckbackslash{%
1847 \noexpand\@gls@xdycheckbackslash##3\noexpand\null}%
1848 \noexpand\fi
1849 \noexpand\else
1850 \noexpand\edef\noexpand\@gls@checkedmkidx{%
1851 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
1852 \backslashchar\backslashchar}%
1853 \noexpand\ifx\noexpand\null##3\noexpand\null
1854 \noexpand\def\noexpand\@gls@xdycheckbackslash{%
1855 \noexpand\@gls@xdycheckbackslash##2\backslashchar
1856 \backslashchar\noexpand\null}%
1857 \noexpand\else
1858 \noexpand\def\noexpand\@gls@xdycheckbackslash{%
1859 \noexpand\@gls@xdycheckbackslash##2\backslashchar
1860 ##3\noexpand\null}%
1861 \noexpand\fi
1862 \noexpand\fi
1863 \noexpand\@gls@xdycheckbackslash
1864 }%
1865 }

```

Now go ahead and define \gls@xdycheckbackslash

```

1866 \def@gls@xdycheckbackslash

```

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

```

1867 \ifcsundef{hyperlink}%
1868 {%
1869 \gdef@glslink#1#2{#2}%
1870 }%
1871 {%
1872 \gdef@glslink#1#2{\hyperlink{#1}{#2}}%
1873 }

```

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

```

1874 \newlength@gls@tmplen
1875 \ifcsundef{hypertarget}%
1876 {%
1877 \gdef@glstarget#1#2{#2}%
1878 }%
1879 {%
1880 \gdef@glstarget#1#2{%
1881 \settoheight@gls@tmplen{#2}%
1882 \raisebox@gls@tmplen{\hypertarget{#1}}{#2}%
1883 }%
1884 }

```

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

`\glsdisablehyper`

```
1885 \newcommand{\glsdisablehyper}{%
1886 \renewcommand*\@glslink[2]{##2}%
1887 \renewcommand*\@glsstarget[2]{##2}}
```

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

`\glsenablehyper`

```
1888 \newcommand{\glsenablehyper}{%
1889 \renewcommand*\@glslink[2]{\hyperlink{##1}{##2}}%
1890 \renewcommand*\@glsstarget[2]{%
1891   \settoheight{\gls@tmplen}{##2}%
1892   \raisebox{\gls@tmplen}{\hypertarget{##1}{}}##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 if we are using the starred form:

`\gls`

```
1893 \newrobustcmd*\gls{\@ifstar\@sgls\@gls}
```

Define the starred form:

`\@sgls`

```
1894 \newcommand*\@sgls[1][]{\@gls[hyper=false,#1]}
```

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

`\@gls`

```
1895 \newcommand*\@gls[2][]{%
1896   \new@ifnextchar{\@gls@{#1}{#2}}{\@gls@{#1}{#2}[]}%
1897 }
```

\@gls@ Read in the final optional argument:

```

1898 \def\@gls@#1#2[#3]{%
1899   \glsdoifexists{#2}%
1900   {%
1901     \edef\@glo@type{\glsentrytype{#2}}%
1902     \def\@gls@link@opts{#1}%
1903     \def\@gls@link@label{#2}%
1904     \ifglsused{#2}%
1905     {%
1906       \def\@glo@text{%
1907         \csname gls@\@glo@type @display\endcsname
1908         {\glsentrytext{#2}}{\glsentrydesc{#2}}{\glsentrysymbol{#2}}{#3}}%
1909       }%
1910       {%
1911         \def\@glo@text{%
1912           \csname gls@\@glo@type @displayfirst\endcsname
1913           {\glsentryfirst{#2}}{\glsentrydesc{#2}}{\glsentrysymbol{#2}}{#3}}%
1914         }%

```

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.

```

1915   \ifglsused{#2}%
1916   {%
1917     \@gls@link[#1]{#2}{\@glo@text}%
1918   }%
1919   {%
1920     \gls@checkisacronymlist\@glo@type
1921     \ifthenelse
1922     {(\boolean{@glsisacronymlist}\AND \boolean{glsacrfootnote})\OR \NOT\boolean{glshyperfirst}}
1923     {}%
1924     {%
1925       \@gls@link[#1,hyper=false]{#2}{\@glo@text}%
1926     }%
1927     {%
1928       \@gls@link[#1]{#2}{\@glo@text}%
1929     }%
1930   }%
1931 }%

```

Indicate that this entry has now been used

```

1932   \ifKV@glslink@local
1933   \glslocalunset{#2}%
1934   \else
1935   \glsunset{#2}%
1936   \fi
1937 }%

```

1938 }

`\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`

```
1939 \newrobustcmd*{\Gls}{\@ifstar\@sGls\@Gls}
```

Define the starred form:

```
1940 \newcommand*{\@sGls}[1] [] {\@Gls[hyper=false,#1]}
```

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

```
1941 \newcommand*{\@Gls}[2] [] {%
```

```
1942   \new@ifnextchar[{\@Gls@{#1}{#2}}{\@Gls@{#1}{#2} []}]%
```

```
1943 }
```

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

```
1944 \def\@Gls@#1#2[#3]{%
```

```
1945   \glsdoifexists{#2}%
```

```
1946   {%
```

```
1947     \edef\@glo@type{\glsentrytype{#2}}%
```

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

```
1948   \def\@gls@link@opts{#1}%
```

```
1949   \def\@gls@link@label{#2}%
```

```
1950   \def\glslabel{#2}%
```

Determine what the link text should be (this is stored in `\@glo@text`)

```
1951   \ifglsused{#2}%
```

```
1952   {%
```

```
1953     \protected@edef\@glo@text{%
```

```
1954       \csname gls@\@glo@type @display\endcsname
```

```
1955         {\glsentrytext{#2}}{\glsentrydesc{#2}}%
```

```
1956         {\glsentrysymbol{#2}}{#3}}%
```

```
1957   }%
```

```
1958   {%
```

```
1959     \protected@edef\@glo@text{%
```

```
1960       \csname gls@\@glo@type @displayfirst\endcsname
```

```
1961         {\glsentryfirst{#2}}{\glsentrydesc{#2}}%
```

```
1962         {\glsentrysymbol{#2}}{#3}}%
```

```
1963   }%
```

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.

```
1964   \ifglsused{#2}%
```

```
1965   {%
```



```

1966 \gls@link[#1]{#2}{%
1967 \expandafter\makefirstuc\expandafter{\@glo@text}}%
1968 }%
1969 {%
1970 \gls@checkisacronymlist\@glo@type
1971 \ifthenelse
1972 {%
1973 \(\boolean{@glsisacronymlist}\AND \boolean{glsacrfootnote}\)
1974 \OR \NOT\boolean{glshyperfirst}}%
1975 }%
1976 {%
1977 \gls@link[#1,hyper=false]{#2}{%
1978 \expandafter\makefirstuc\expandafter{\@glo@text}}%
1979 }%
1980 {%
1981 \gls@link[#1]{#2}{%
1982 \expandafter\makefirstuc\expandafter{\@glo@text}}%
1983 }%
1984 }%

```

Indicate that this entry has now been used

```

1985 \ifKV@glslink@local
1986 \glslocalunset{#2}%
1987 \else
1988 \glsunset{#2}%
1989 \fi
1990 }%
1991 }

```

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

\GLS

```

1992 \newrobustcmd*{\GLS}{\@ifstar\@sGLS\@GLS}

```

Define the starred form:

```

1993 \newcommand*{\@sGLS}[1][\@GLS[hyper=false,#1]]

```

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

```

1994 \newcommand*{\@GLS}[2][\@GLS@#1]{%
1995 \new@ifnextchar[\@GLS@{#1}{#2}}{\@GLS@{#1}{#2}[]}%
1996 }

```

\@GLS@ Read in the final optional argument:

```

1997 \def\@GLS@#1#2[#3]{%
1998 \glsdoifexists{#2}%
1999 {%
2000 \edef\@glo@type{\glsentrytype{#2}}%

```

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

```

2001 \def\@gls@link@opts{#1}%
2002 \def\@gls@link@label{#2}%
    Determine what the link text should be (this is stored in \@glo@text).
2003 \ifglsused{#2}%
2004 {%
2005     \def\@glo@text{%
2006         \csname gls@\@glo@type @display\endcsname
2007         {\glsentrytext{#2}}{\glsentrydesc{#2}}{\glsentrysymbol{#2}}{#3}%
2008     }%
2009 }%
2010 {%
2011     \def\@glo@text{%
2012         \csname gls@\@glo@type @displayfirst\endcsname
2013         {\glsentryfirst{#2}}{\glsentrydesc{#2}}{\glsentrysymbol{#2}}{#3}%
2014     }%
2015 }%

```

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.

```

2016 \ifglsused{#2}%
2017 {%
2018     \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%
2019 }%
2020 {%
2021     \gls@checkisacronymlist\@glo@type
2022     \ifthenelse
2023     {%
2024         \(\boolean{@glsisacronymlist}\AND \boolean{glsacrfootnote}\)
2025         \OR \NOT\boolean{glshyperfirst}}{%
2026         \@gls@link[#1,hyper=false]{#2}{\MakeUppercase{\@glo@text}}%
2027     }%
2028     {%
2029         \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%
2030     }%
2031 }%

```

Indicate that this entry has now been used

```

2032 \ifKV@glslink@local
2033     \glslocalunset{#2}%
2034 \else
2035     \glsunset{#2}%
2036 \fi
2037 }%
2038 }

```

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

\glspl

```

2039 \newrobustcmd*{\glspl}{\@ifstar\@sglspl\@glspl}

```

Define the starred form:

```
2040 \newcommand*{\@glspl}[1][\@glspl[hyper=false,#1]]
```

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

```
2041 \newcommand*{\@glspl}[2][\%
2042   \new@ifnextchar[\@glspl@{#1}{#2}]{\@glspl@{#1}{#2}[]}%
2043 }
```

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

```
2044 \def\@glspl@#1#2[#3]{%
2045   \glsdoifexists{#2}%
2046   {%
2047     \edef\@glo@type{\glsentrytype{#2}}%
```

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

```
2048   \def\@gls@link@opts{#1}%
2049   \def\@gls@link@label{#2}%
```

Determine what the link text should be (this is stored in `\@glo@text`)

```
2050   \ifglsused{#2}%
2051   {%
2052     \def\@glo@text{%
2053       \csname gls@\@glo@type @display\endcsname
2054       {\glsentryplural{#2}}{\glsentrydescplural{#2}}%
2055       {\glsentrysymbolplural{#2}}{#3}}%
2056   }%
2057   {%
2058     \def\@glo@text{%
2059       \csname gls@\@glo@type @displayfirst\endcsname
2060       {\glsentryfirstplural{#2}}{\glsentrydescplural{#2}}%
2061       {\glsentrysymbolplural{#2}}{#3}}%
2062   }%
```

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.

```
2063   \ifglsused{#2}%
2064   {%
2065     \@gls@link[#1]{#2}{\@glo@text}%
2066   }%
2067   {%
2068     \gls@checkisacronymlist\@glo@type
2069     \ifthenelse
2070     {%
2071       \(\boolean{@glsisacronymlist}\AND \boolean{glsacrfootnote}\)
2072       \OR \NOT\boolean{glshyperfirst}%
2073     }%
2074     {%
2075       \@gls@link[#1,hyper=false]{#2}{\@glo@text}%
```

```

2076     }%
2077     {%
2078     \@gls@link[#1]{#2}{\@glo@text}%
2079     }%
2080 }%

```

Indicate that this entry has now been used

```

2081     \ifKV@glslink@local
2082     \glslocalunset{#2}%
2083     \else
2084     \glsunset{#2}%
2085     \fi
2086 }%
2087 }

```

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

`\Glspl`

```

2088 \newrobustcmd*{\Glspl}{\@ifstar\@sGlspl\@Glspl}

```

Define the starred form:

```

2089 \newcommand*{\@sGlspl}[1][\@Glspl[hyper=false,#1]]{

```

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

```

2090 \newcommand*{\@Glspl}[2][\@Glspl@{#1}{#2}]{\@Glspl@{#1}{#2}[]}
2091 \new@ifnextchar[\@Glspl@{#1}{#2}]{\@Glspl@{#1}{#2}[]}
2092 }

```

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

```

2093 \def\@Glspl@#1#2[#3]{%
2094   \glsdoifexists{#2}%
2095   {%
2096     \edef\@glo@type{\glsentrytype{#2}}%

```

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

```

2097   \def\@gls@link@opts{#1}%
2098   \def\@gls@link@label{#2}%
2099   \def\@gls@label{#2}%

```

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`.

```

2100   \ifglsused{#2}%
2101   {%
2102     \protected@edef\@glo@text{%
2103       \csname gls@\@glo@type @display\endcsname
2104       {\glsentryplural{#2}}{\glsentrydescplural{#2}}%

```

```

2105      {\glsentrysymbolplural{#2}}{#3}}%
2106    }%
2107    {%
2108      \protected@edef\@glo@text{%
2109        \csname gls@\@glo@type @displayfirst\endcsname
2110        {\glsentryfirstplural{#2}}{\glsentrydescplural{#2}}%
2111        {\glsentrysymbolplural{#2}}{#3}}%
2112    }%

```

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.

```

2113    \ifglsused{#2}%
2114    {%
2115      \@gls@link[#1]{#2}{%
2116        \expandafter\makefirstuc\expandafter{\@glo@text}}%
2117    }%
2118    {%
2119      \gls@checkisacronymlist\@glo@type
2120      \ifthenelse
2121      {%
2122        \(\boolean{glsisacronymlist}\AND \boolean{glsacrfootnote}\)
2123        \OR \NOT\boolean{glshyperfirst}}%
2124      }%
2125      {%
2126        \@gls@link[#1,hyper=false]{#2}{%
2127          \expandafter\makefirstuc\expandafter{\@glo@text}}%
2128      }%
2129      {%
2130        \@gls@link[#1]{#2}{%
2131          \expandafter\makefirstuc\expandafter{\@glo@text}}%
2132      }%
2133    }%

```

Indicate that this entry has now been used

```

2134    \ifKV@glslink@local
2135      \glslocalunset{#2}%
2136    \else
2137      \glsunset{#2}%
2138    \fi
2139  }%
2140}

```

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

`\GLSp1`

```

2141 \newrobustcmd*{\GLSp1}{\@ifstar\@sGLSp1\@GLSp1}

```

Define the starred form:

```

2142 \newcommand*{\@sGLSp1}[1][\@GLSp1[hyper=false,#1]]{

```

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

```
2143 \newcommand*{\@GLSp1}[2] [] {%
2144   \new@ifnextchar[{\@GLSp1@{#1}{#2}}{\@GLSp1@{#1}{#2} []}%
2145 }
```

\@GLSp1 Read in the final optional argument:

```
2146 \def\@GLSp1@#1#2[#3] {%
2147   \glsdoifexists{#2}%
2148   {%
2149     \edef\@glo@type{\glsentrytype{#2}}%
```

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

```
2150   \def\@gls@link@opts{#1}%
2151   \def\@gls@link@label{#2}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2152   \ifglsused{#2}%
2153   {%
2154     \def\@glo@text{%
2155       \csname gls@\@glo@type @display\endcsname
2156       {\glsentryplural{#2}}{\glsentrydescplural{#2}}%
2157       {\glsentrysymbolplural{#2}}{#3}%
2158     }%
2159   }%
2160   {%
2161     \def\@glo@text{%
2162       \csname gls@\@glo@type @displayfirst\endcsname
2163       {\glsentryfirstplural{#2}}{\glsentrydescplural{#2}}%
2164       {\glsentrysymbolplural{#2}}{#3}%
2165     }%
2166   }%
```

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.

```
2167   \ifglsused{#2}%
2168   {%
2169     \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%
2170   }%
2171   {%
2172     \gls@checkisacronymlist\@glo@type
2173     \ifthenelse
2174     {%
2175       \(\boolean{glsisacronymlist}\AND \boolean{glsacrfootnote}\)
2176       \OR \NOT\boolean{glshyperfirst}%
2177     }%
2178     {%
2179       \@gls@link[#1,hyper=false]{#2}{\MakeUppercase{\@glo@text}}%
2180     }%
```

```

2181      {%
2182      \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%
2183      }%
2184  }%

```

Indicate that this entry has now been used

```

2185      \ifKV@glslink@local
2186      \glslocalunset{#2}%
2187      \else
2188      \glsunset{#2}%
2189      \fi
2190  }%
2191 }

```

`\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:

```

2192 \newrobustcmd*{\glsdisp}{\@ifstar\sglsdisp\@glsdisp}

```

Define the starred form:

```

\@sgls
2193 \newcommand*{\sglsdisp}[1][\@glsdisp[hyper=false,#1]]{

```

Defined the un-starred form.

`\@glsdisp`

```

2194 \newcommand*{\@glsdisp}[3][\@glsdisp]{
2195   \glsdoifexists{#2}{%
2196     \edef\@glo@type{\glsentrytype{#2}}%

```

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

```

2197   \def\@gls@link@opts{#1}%
2198   \def\@gls@link@label{#2}%

```

Determine what the link text should be (this is stored in `\@glo@text`)

```

2199   \ifglsused{#2}%
2200   {%
2201     \def\@glo@text{%
2202       \csname gls@\@glo@type @display\endcsname
2203       {#3}{\glsentrydesc{#2}}{\glsentrysymbol{#2}}{}}%
2204   }%
2205   {%
2206     \def\@glo@text{%
2207       \csname gls@\@glo@type @displayfirst\endcsname
2208       {#3}{\glsentrydesc{#2}}{\glsentrysymbol{#2}}{}}%
2209   }%

```

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.

```

2210 \ifglsused{#2}%
2211 {%
2212 \@gls@link[#1]{#2}{\@glo@text}%
2213 }%
2214 {%
2215 \gls@checkisacronymlist\@glo@type
2216 \ifthenelse{(\boolean{@glsisacronymlist}\AND
2217 \boolean{glsacrfootnote})\OR \NOT\boolean{glshyperfirst}}%
2218 {%
2219 \@gls@link[#1,hyper=false]{#2}{\@glo@text}%
2220 }%
2221 {%
2222 \@gls@link[#1]{#2}{\@glo@text}%
2223 }%
2224 }%

```

Indicate that this entry has now been used

```

2225 \ifKV@glslink@local
2226 \glslocalunset{#2}%
2227 \else
2228 \glsunset{#2}%
2229 \fi
2230 }%
2231 }

```

`\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`

```

2232 \newrobustcmd*{\glstext}{\@ifstar\@sglstext\@glstext}

```

Define the starred form:

```

2233 \newcommand*{\@sglstext}[1][\@glstext[hyper=false,#1]]{

```

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

```

2234 \newcommand*{\@glstext}[2][\@glstext@{#1}{#2}]{\@glstext@{#1}{#2}[]}
2235 \new@ifnextchar[\@glstext@{#1}{#2}]{\@glstext@{#1}{#2}[]}

```

Read in the final optional argument:

```

2236 \def\@glstext@#1#2[#3]{%
2237 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%

```

Determine what the link text should be (this is stored in `\@glo@text`)

```

2238 \protected@edef\@glo@text{\glsentrytext{#2}}%

```

Call `\@gls@link`

```

2239 \@gls@link[#1]{#2}{\@glo@text#3}%

```


2240 }%

2241 }

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

\GLStext

2242 \newrobustcmd*{\GLStext}{\@ifstar\@sGLStext\@GLStext}

Define the starred form:

2243 \newcommand*{\@sGLStext}[1][]{\@GLStext[hyper=false,#1]}

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

2244 \newcommand*{\@GLStext}[2][]{%

2245 \new@ifnextchar[\@GLStext@{#1}{#2}]{\@GLStext@{#1}{#2}[]}}

Read in the final optional argument:

2246 \def\@GLStext@#1#2[#3]{%

2247 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%

Determine what the link text should be (this is stored in \@glo@text)

2248 \protected@edef\@glo@text{\glsentrytext{#2}}%

Call \@gls@link

2249 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#3}}%

2250 }%

2251 }

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

\Glstext

2252 \newrobustcmd*{\Glstext}{\@ifstar\@sGlstext\@Glstext}

Define the starred form:

2253 \newcommand*{\@sGlstext}[1][]{\@Glstext[hyper=false,#1]}

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

2254 \newcommand*{\@Glstext}[2][]{%

2255 \new@ifnextchar[\@Glstext@{#1}{#2}]{\@Glstext@{#1}{#2}[]}}

Read in the final optional argument:

2256 \def\@Glstext@#1#2[#3]{%

2257 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%

Determine what the link text should be (this is stored in \@glo@text)

2258 \protected@edef\@glo@text{\glsentrytext{#2}}%

Call \@gls@link

2259 \@gls@link[#1]{#2}{%

2260 \expandafter\makefirstuc\expandafter{\@glo@text#3}}%

2261 }%

2262 }

`\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`

```
2263 \newrobustcmd*{\glsfirst}{\@ifstar\sglsfirst\glsfirst}
```

Define the starred form:

```
2264 \newcommand*{\sglsfirst}[1] [] {\glsfirst[hyper=false,#1]}
```

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

```
2265 \newcommand*{\glsfirst}[2] [] {%
```

```
2266 \new@ifnextchar[{\glsfirst@{#1}{#2}}{\glsfirst@{#1}{#2} []}]}
```

Read in the final optional argument:

```
2267 \def\glsfirst@#1#2[#3] {%
```

```
2268 \glsdoifexists{#2}{\edef\glo@type{\glsentrytype{#2}}}%
```

Determine what the link text should be (this is stored in `\glo@text`)

```
2269 \protected@edef\glo@text{\glsentryfirst{#2}}%
```

Call `\@gls@link`

```
2270 \@gls@link[#1]{#2}{\glo@text#3}%
```

```
2271 }%
```

```
2272 }
```

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

`\Glsfirst`

```
2273 \newrobustcmd*{\Glsfirst}{\@ifstar\sglsfirst\Glsfirst}
```

Define the starred form:

```
2274 \newcommand*{\sglsfirst}[1] [] {\Glsfirst[hyper=false,#1]}
```

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

```
2275 \newcommand*{\Glsfirst}[2] [] {%
```

```
2276 \new@ifnextchar[{\Glsfirst@{#1}{#2}}{\Glsfirst@{#1}{#2} []}]}
```

Read in the final optional argument:

```
2277 \def\Glsfirst@#1#2[#3] {%
```

```
2278 \glsdoifexists{#2}{\edef\glo@type{\glsentrytype{#2}}}%
```

Determine what the link text should be (this is stored in `\glo@text`)

```
2279 \protected@edef\glo@text{\glsentryfirst{#2}}%
```

Call `\@gls@link`

```
2280 \@gls@link[#1]{#2}{%
```

```
2281 \expandafter\makefirstuc\expandafter{\glo@text}#3}%
```

```
2282 }%
```

```
2283 }
```

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

`\GLSfirst`

```
2284 \newrobustcmd*{\GLSfirst}{\@ifstar\@sGLSfirst\@GLSfirst}
```

Define the starred form:

```
2285 \newcommand*{\@sGLSfirst}[1] [] {\@GLSfirst[hyper=false,#1]}
```

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

```
2286 \newcommand*{\@GLSfirst}[2] [] {%
```

```
2287 \new@ifnextchar[{\@GLSfirst@{#1}{#2}}{\@GLSfirst@{#1}{#2} []}]
```

Read in the final optional argument:

```
2288 \def\@GLSfirst@#1#2[#3] {%
```

```
2289 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in `\@glo@text`)

```
2290 \protected@edef\@glo@text{\glsentryfirst{#2}}%
```

Call `\@gls@link`

```
2291 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#3}}%
```

```
2292 }%
```

```
2293 }
```

`\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`

```
2294 \newrobustcmd*{\glsplural}{\@ifstar\@sglsplural\@glsplural}
```

Define the starred form:

```
2295 \newcommand*{\@sglsplural}[1] [] {\@glsplural[hyper=false,#1]}
```

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

```
2296 \newcommand*{\@glsplural}[2] [] {%
```

```
2297 \new@ifnextchar[{\@glsplural@{#1}{#2}}{\@glsplural@{#1}{#2} []}]
```

Read in the final optional argument:

```
2298 \def\@glsplural@#1#2[#3] {%
```

```
2299 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in `\@glo@text`)

```
2300 \protected@edef\@glo@text{\glsentryplural{#2}}%
```

Call `\@gls@link`

```
2301 \@gls@link[#1]{#2}{\@glo@text#3}}%
```

```
2302 }%
```

```
2303 }
```

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

`\Glsplural`

```
2304 \newrobustcmd*{\Glsplural}{\@ifstar\@sGlsplural\@Glsplural}
```

Define the starred form:

```
2305 \newcommand*{\@sGlsplural}[1] [] {\@Glsplural[hyper=false,#1]}
```

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

```
2306 \newcommand*{\@Glsplural}[2] [] {%
```

```
2307 \new@ifnextchar [\@Glsplural@{#1}{#2}]{\@Glsplural@{#1}{#2} []}}
```

Read in the final optional argument:

```
2308 \def\@Glsplural@#1#2[#3] {%
```

```
2309 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2310 \protected@edef\@glo@text{\glsentryplural{#2}}%
```

Call \@gls@link

```
2311 \@gls@link[#1]{#2}{%
```

```
2312 \expandafter\makefirstuc\expandafter{\@glo@text}#3}%
```

```
2313 }%
```

```
2314 }
```

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

\Glsplural

```
2315 \newrobustcmd*{\Glsplural}{\@ifstar\@sGlsplural\@Glsplural}
```

Define the starred form:

```
2316 \newcommand*{\@sGlsplural}[1] [] {\@Glsplural[hyper=false,#1]}
```

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

```
2317 \newcommand*{\@Glsplural}[2] [] {%
```

```
2318 \new@ifnextchar [\@Glsplural@{#1}{#2}]{\@Glsplural@{#1}{#2} []}}
```

Read in the final optional argument:

```
2319 \def\@Glsplural@#1#2[#3] {%
```

```
2320 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2321 \protected@edef\@glo@text{\glsentryplural{#2}}%
```

Call \@gls@link

```
2322 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#3}}%
```

```
2323 }%
```

```
2324 }
```

\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

```
2325 \newrobustcmd*{\glsfirstplural}{\@ifstar\@sglsfirstplural\@glsfirstplural}
```

Define the starred form:

```
2326 \newcommand*{\@sglsfirstplural}[1] [] {\@glsfirstplural[hyper=false,#1]}
```

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

```
2327 \newcommand*{\@glsfirstplural}[2] [] {%
```

```
2328 \new@ifnextchar [\@glsfirstplural@{#1}{#2}]{\@glsfirstplural@{#1}{#2} []}}
```

Read in the final optional argument:

```
2329 \def\@glsfirstplural@#1#2[#3] {%
```

```
2330 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2331 \protected@edef\@glo@text{\glsentryfirstplural{#2}}%
```

Call \@gls@link

```
2332 \@gls@link[#1]{#2}{\@glo@text#3}%
```

```
2333 }%
```

```
2334 }
```

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

\Glsfirstplural

```
2335 \newrobustcmd*{\Glsfirstplural}{\@ifstar\@sGlsfirstplural\@Glsfirstplural}
```

Define the starred form:

```
2336 \newcommand*{\@sGlsfirstplural}[1] [] {\@Glsfirstplural[hyper=false,#1]}
```

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

```
2337 \newcommand*{\@Glsfirstplural}[2] [] {%
```

```
2338 \new@ifnextchar [\@Glsfirstplural@{#1}{#2}]{\@Glsfirstplural@{#1}{#2} []}}
```

Read in the final optional argument:

```
2339 \def\@Glsfirstplural@#1#2[#3] {%
```

```
2340 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2341 \protected@edef\@glo@text{\glsentryfirstplural{#2}}%
```

Call \@gls@link

```
2342 \@gls@link[#1]{#2}{%
```

```
2343 \expandafter\makefirstuc\expandafter{\@glo@text}#3}%
```

```
2344 }%
```

```
2345 }
```

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

\GLSfirstplural

```
2346 \newrobustcmd*{\GLSfirstplural}{\@ifstar\@sGLSfirstplural\@GLSfirstplural}
```

Define the starred form:

```
2347 \newcommand*{\@sGLSfirstplural}[1] [] {\@GLSfirstplural[hyper=false,#1]}
```

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

```
2348 \newcommand*{\@GLSfirstplural}[2] [] {%
```

```
2349 \new@ifnextchar [\@GLSfirstplural@{#1}{#2}]{\@GLSfirstplural@{#1}{#2} []}}
```

Read in the final optional argument:

```
2350 \def\@GLSfirstplural@#1#2[#3] {%
```

```
2351 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2352 \protected@edef\@glo@text{\glsentryfirstplural{#2}}%
```

Call \@gls@link

```
2353 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#3}}%
```

```
2354 }%
```

```
2355 }
```

\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

```
2356 \newrobustcmd*{\glsname}{\@ifstar\@sglsname\@glsname}
```

Define the starred form:

```
2357 \newcommand*{\@sglsname}[1] [] {\@glsname[hyper=false,#1]}
```

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

```
2358 \newcommand*{\@glsname}[2] [] {%
```

```
2359 \new@ifnextchar [\@glsname@{#1}{#2}]{\@glsname@{#1}{#2} []}}
```

Read in the final optional argument:

```
2360 \def\@glsname@#1#2[#3] {%
```

```
2361 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2362 \protected@edef\@glo@text{\glsentryname{#2}}%
```

Call \@gls@link

```
2363 \@gls@link[#1]{#2}{\@glo@text#3}}%
```

```
2364 }%
```

```
2365 }
```

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

\Glsname

```
2366 \newrobustcmd*{\Glsname}{\@ifstar\@sGlsname\@Glsname}
```

Define the starred form:

```
2367 \newcommand*{\@sGlsname}[1] [] {\@Glsname[hyper=false,#1]}
```

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

```
2368 \newcommand*{\@Glsname}[2] [] {%
2369 \new@ifnextchar [{\@Glsname@{#1}{#2}}{\@Glsname@{#1}{#2} [] }}
```

Read in the final optional argument:

```
2370 \def\@Glsname@#1#2[#3] {%
2371 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2372 \protected@edef\@glo@text{\glsentryname{#2}}%
```

Call \@gls@link

```
2373 \@gls@link[#1]{#2}{%
2374 \expandafter\makefirstuc\expandafter{\@glo@text}#3}%
2375 }%
2376 }
```

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

\GLSname

```
2377 \newrobustcmd*{\@GLSname}{\@ifstar\@sGLSname\@GLSname}
```

Define the starred form:

```
2378 \newcommand*{\@sGLSname}[1] [] {\@GLSname[hyper=false,#1]}
```

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

```
2379 \newcommand*{\@GLSname}[2] [] {%
2380 \new@ifnextchar [{\@GLSname@{#1}{#2}}{\@GLSname@{#1}{#2} [] }}
```

Read in the final optional argument:

```
2381 \def\@GLSname@#1#2[#3] {%
2382 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2383 \protected@edef\@glo@text{\glsentryname{#2}}%
```

Call \@gls@link

```
2384 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#3}}%
2385 }%
2386 }
```

\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

```
2387 \newrobustcmd*{\glsdesc}{\@ifstar\@sglsdesc\@glsdesc}
```

Define the starred form:

```
2388 \newcommand*{\@sglsdesc}[1] [] {\@glsdesc[hyper=false,#1]}
```

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

```
2389 \newcommand*{\@glsdesc}[2] [] {%
2390 \new@ifnextchar [{\@glsdesc@{#1}{#2}}{\@glsdesc@{#1}{#2} []}}
```

Read in the final optional argument:

```
2391 \def\@glsdesc@#1#2[#3] {%
2392 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2393 \protected@edef\@glo@text{\glsentrydesc{#2}}%
```

Call \@gls@link

```
2394 \@gls@link[#1]{#2}{\@glo@text#3}%
2395 }%
2396 }
```

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

\Glsdesc

```
2397 \newrobustcmd*{\Glsdesc}{\@ifstar\@sGlsdesc\@Glsdesc}
```

Define the starred form:

```
2398 \newcommand*{\@sGlsdesc}[1] [] {\@Glsdesc[hyper=false,#1]}
```

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

```
2399 \newcommand*{\@Glsdesc}[2] [] {%
2400 \new@ifnextchar [{\@Glsdesc@{#1}{#2}}{\@Glsdesc@{#1}{#2} []}}
```

Read in the final optional argument:

```
2401 \def\@Glsdesc@#1#2[#3] {%
2402 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2403 \protected@edef\@glo@text{\glsentrydesc{#2}}%
```

Call \@gls@link

```
2404 \@gls@link[#1]{#2}{%
2405   \expandafter\makefirstuc\expandafter{\@glo@text}#3}%
2406 }%
2407 }
```

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

\GLSdesc

```
2408 \newrobustcmd*{\GLSdesc}{\@ifstar\@sGLSdesc\@GLSdesc}
```

Define the starred form:

```
2409 \newcommand*{\@sGLSdesc}[1] [] {\@GLSdesc[hyper=false,#1]}
```


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

```
2410 \newcommand*{\@GLSdesc}[2] [] {%
2411 \new@ifnextchar [{\@GLSdesc@{#1}{#2}}{\@GLSdesc@{#1}{#2} []}}
```

Read in the final optional argument:

```
2412 \def\@GLSdesc@#1#2[#3] {%
2413 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2414 \protected@edef\@glo@text{\glsentrydesc{#2}}%
```

Call \@gls@link

```
2415 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#3}}%
2416 }%
2417 }
```

\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

```
2418 \newrobustcmd*{\glsdescplural}{\@ifstar\sglsdescplural\@glsdescplural}
```

Define the starred form:

```
2419 \newcommand*{\@sglsdescplural}[1] [] {\@glsdescplural[hyper=false,#1]}
```

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

```
2420 \newcommand*{\@glsdescplural}[2] [] {%
2421 \new@ifnextchar [{\@glsdescplural@{#1}{#2}}{\@glsdescplural@{#1}{#2} []}}
```

Read in the final optional argument:

```
2422 \def\@glsdescplural@#1#2[#3] {%
2423 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2424 \protected@edef\@glo@text{\glsentrydescplural{#2}}%
```

Call \@gls@link

```
2425 \@gls@link[#1]{#2}{\@glo@text#3}}%
2426 }%
2427 }
```

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

\Glsdescplural

```
2428 \newrobustcmd*{\Glsdescplural}{\@ifstar\@sGlsdescplural\@Glsdescplural}
```

Define the starred form:

```
2429 \newcommand*{\@sGlsdescplural}[1] [] {\@Glsdescplural[hyper=false,#1]}
```

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

```
2430 \newcommand*{\@GLSdescplural}[2] [] {%
2431 \new@ifnextchar [{\@GLSdescplural@{#1}{#2}}]{\@GLSdescplural@{#1}{#2} []}}
```

Read in the final optional argument:

```
2432 \def\@GLSdescplural@#1#2[#3] {%
2433 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2434 \protected@edef\@glo@text{\glsentrydescplural{#2}}%
```

Call \@gls@link

```
2435 \@gls@link[#1]{#2}{%
2436 \expandafter\makefirstuc\expandafter{\@glo@text}#3}%
2437 }%
2438 }
```

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

\GLSdescplural

```
2439 \newrobustcmd*{\GLSdescplural}{\@ifstar\@sGLSdescplural\@GLSdescplural}
```

Define the starred form:

```
2440 \newcommand*{\@sGLSdescplural}[1] [] {\@GLSdescplural[hyper=false,#1]}
```

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

```
2441 \newcommand*{\@GLSdescplural}[2] [] {%
2442 \new@ifnextchar [{\@GLSdescplural@{#1}{#2}}]{\@GLSdescplural@{#1}{#2} []}}
```

Read in the final optional argument:

```
2443 \def\@GLSdescplural@#1#2[#3] {%
2444 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2445 \protected@edef\@glo@text{\glsentrydescplural{#2}}%
```

Call \@gls@link

```
2446 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#3}}%
2447 }%
2448 }
```

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

\glsymbol

```
2449 \newrobustcmd*{\glsymbol}{\@ifstar\@sglsymbol\@glsymbol}
```

Define the starred form:

```
2450 \newcommand*{\@sglsymbol}[1] [] {\@glsymbol[hyper=false,#1]}
```

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

```
2451 \newcommand*{\@glssymbol}[2] [] {%
2452 \new@ifnextchar [{\@glssymbol@{#1}{#2}}{\@glssymbol@{#1}{#2} []}}
```

Read in the final optional argument:

```
2453 \def\@glssymbol@#1#2[#3] {%
2454 \glsoifexists{#2}{\edef\@glo@type{\glentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2455 \protected@edef\@glo@text{\glentrysymbol{#2}}%
```

Call \@gls@link

```
2456 \@gls@link[#1]{#2}{\@glo@text#3}%
2457 }%
2458 }
```

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

\Glsymbol

```
2459 \newrobustcmd*{\Glsymbol}{\@ifstar\@sGlsymbol\@Glsymbol}
```

Define the starred form:

```
2460 \newcommand*{\@sGlsymbol}[1] [] {\@Glsymbol[hyper=false,#1]}
```

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

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

Read in the final optional argument:

```
2463 \def\@Glsymbol@#1#2[#3] {%
2464 \glsoifexists{#2}{\edef\@glo@type{\glentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2465 \protected@edef\@glo@text{\glentrysymbol{#2}}%
```

Call \@gls@link

```
2466 \@gls@link[#1]{#2}{%
2467 \expandafter\makefirstuc\expandafter{\@glo@text}#3}%
2468 }%
2469 }
```

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

\GLSsymbol

```
2470 \newrobustcmd*{\GLSsymbol}{\@ifstar\@sGLSsymbol\@GLSsymbol}
```

Define the starred form:

```
2471 \newcommand*{\@sGLSsymbol}[1] [] {\@GLSsymbol[hyper=false,#1]}
```

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

```
2472 \newcommand*{\@GLSsymbol}[2] [] {%
2473 \new@ifnextchar [{\@GLSsymbol@{#1}{#2}}{\@GLSsymbol@{#1}{#2} []}}
```

Read in the final optional argument:

```
2474 \def\@GLSsymbol@#1#2[#3] {%
2475 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2476 \protected@edef\@glo@text{\glsentrysymbol{#2}}%
```

Call \@gls@link

```
2477 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#3}}%
2478 }%
2479 }
```

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

\glsymbolplural

```
2480 \newrobustcmd*{\glsymbolplural}{\@ifstar\sglsymbolplural\@glsymbolplural}
```

Define the starred form:

```
2481 \newcommand*{\@sglsymbolplural}[1] [] {\@glsymbolplural[hyper=false,#1]}
```

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

```
2482 \newcommand*{\@glsymbolplural}[2] [] {%
2483 \new@ifnextchar [{\@glsymbolplural@{#1}{#2}}{\@glsymbolplural@{#1}{#2} []}}
```

Read in the final optional argument:

```
2484 \def\@glsymbolplural@#1#2[#3] {%
2485 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2486 \protected@edef\@glo@text{\glsentrysymbolplural{#2}}%
```

Call \@gls@link

```
2487 \@gls@link[#1]{#2}{\@glo@text#3}}%
2488 }%
2489 }
```

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

\Glsymbolplural

```
2490 \newrobustcmd*{\Glsymbolplural}{\@ifstar\@sGlsymbolplural\@Glsymbolplural}
```

Define the starred form:

```
2491 \newcommand*{\@sGlsymbolplural}[1] [] {\@Glsymbolplural[hyper=false,#1]}
```

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

```
2492 \newcommand*{\@Glssymbolplural}[2] [] {%
2493 \new@ifnextchar [{\@Glssymbolplural@{#1}{#2}}{\@Glssymbolplural@{#1}{#2} [] }}
```

Read in the final optional argument:

```
2494 \def\@Glssymbolplural@#1#2[#3] {%
2495 \glsdoifexists{#2}{\edef\@glo@type{\glstentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2496 \protected@edef\@glo@text{\glstentrysymbolplural{#2}}%
```

Call \@gls@link

```
2497 \@gls@link[#1]{#2}{%
2498 \expandafter\makefirstuc\expandafter{\@glo@text}#3}%
2499 }%
2500 }
```

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

\GLSsymbolplural

```
2501 \newrobustcmd*{\GLSsymbolplural}{\@ifstar\@sGLSsymbolplural\@GLSsymbolplural}
```

Define the starred form:

```
2502 \newcommand*{\@sGLSsymbolplural}[1] [] {\@GLSsymbolplural[hyper=false,#1]}
```

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

```
2503 \newcommand*{\@GLSsymbolplural}[2] [] {%
2504 \new@ifnextchar [{\@GLSsymbolplural@{#1}{#2}}{\@GLSsymbolplural@{#1}{#2} [] }}
```

Read in the final optional argument:

```
2505 \def\@GLSsymbolplural@#1#2[#3] {%
2506 \glsdoifexists{#2}{\edef\@glo@type{\glstentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2507 \protected@edef\@glo@text{\glstentrysymbolplural{#2}}%
```

Call \@gls@link

```
2508 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#3}}%
2509 }%
2510 }
```

\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

```
2511 \newrobustcmd*{\glsuseri}{\@ifstar\@sglsuseri\@glsuseri}
```

Define the starred form:

```
2512 \newcommand*{\@sglsuseri}[1] [] {\@glsuseri[hyper=false,#1]}
```

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

```
2513 \newcommand*{\@glsuseri}[2] [] {%
2514 \new@ifnextchar [\@glsuseri@{#1}{#2}]{\@glsuseri@{#1}{#2} []}}
```

Read in the final optional argument:

```
2515 \def\@glsuseri@#1#2[#3] {%
2516 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2517 \protected@edef\@glo@text{\glsentryuseri{#2}}%
```

Call \@gls@link

```
2518 \@gls@link[#1]{#2}{\@glo@text#3}%
2519 }%
2520 }
```

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

\Glsuseri

```
2521 \newrobustcmd*{\Glsuseri}{\@ifstar\@sGlsuseri\@Glsuseri}
```

Define the starred form:

```
2522 \newcommand*{\@sGlsuseri}[1] [] {\@Glsuseri[hyper=false,#1]}
```

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

```
2523 \newcommand*{\@Glsuseri}[2] [] {%
2524 \new@ifnextchar [\@Glsuseri@{#1}{#2}]{\@Glsuseri@{#1}{#2} []}}
```

Read in the final optional argument:

```
2525 \def\@Glsuseri@#1#2[#3] {%
2526 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2527 \protected@edef\@glo@text{\glsentryuseri{#2}}%
```

Call \@gls@link

```
2528 \@gls@link[#1]{#2}{%
2529 \expandafter\makefirstuc\expandafter{\@glo@text}#3}%
2530 }%
2531 }
```

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

\GLSuseri

```
2532 \newrobustcmd*{\GLSuseri}{\@ifstar\@sGLSuseri\@GLSuseri}
```

Define the starred form:

```
2533 \newcommand*{\@sGLSuseri}[1] [] {\@GLSuseri[hyper=false,#1]}
```

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

```
2534 \newcommand*{\@GLSuseri}[2] [] {%
2535 \new@ifnextchar [{\@GLSuseri@{#1}{#2}}{\@GLSuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
2536 \def\@GLSuseri@#1#2[#3] {%
2537 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2538 \protected@edef\@glo@text{\glsentryuseri{#2}}%
```

Call \@gls@link

```
2539 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#3}}%
2540 }%
2541 }
```

\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

```
2542 \newrobustcmd*{\glsuserii}{\@ifstar\@sglsuserii\@glsuserii}
```

Define the starred form:

```
2543 \newcommand*{\@sglsuserii}[1] [] {\@glsuserii[hyper=false,#1]}
```

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

```
2544 \newcommand*{\@glsuserii}[2] [] {%
2545 \new@ifnextchar [{\@glsuserii@{#1}{#2}}{\@glsuserii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
2546 \def\@glsuserii@#1#2[#3] {%
2547 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2548 \protected@edef\@glo@text{\glsentryuserii{#2}}%
```

Call \@gls@link

```
2549 \@gls@link[#1]{#2}{\@glo@text#3}}%
2550 }%
2551 }
```

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

\Glsuserii

```
2552 \newrobustcmd*{\Glsuserii}{\@ifstar\@sGlsuserii\@Glsuserii}
```

Define the starred form:

```
2553 \newcommand*{\@sGlsuserii}[1] [] {\@Glsuserii[hyper=false,#1]}
```

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

```
2554 \newcommand*{\@Glsuserii}[2] [] {%
2555 \new@ifnextchar [{\@Glsuserii@{#1}{#2}}{\@Glsuserii@{#1}{#2} []}}
```

Read in the final optional argument:

```
2556 \def\@Glsuserii@#1#2[#3] {%
2557 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2558 \protected@edef\@glo@text{\glsentryuserii{#2}}%
```

Call \@gls@link

```
2559 \@gls@link[#1]{#2}{%
2560 \expandafter\makefirstuc\expandafter{\@glo@text}#3}%
2561 }%
2562 }
```

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

\GLSuserii

```
2563 \newrobustcmd*{\GLSuserii}{\@ifstar\@sGLSuserii\@GLSuserii}
```

Define the starred form:

```
2564 \newcommand*{\@sGLSuserii}[1] [] {\@GLSuserii[hyper=false,#1]}
```

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

```
2565 \newcommand*{\@GLSuserii}[2] [] {%
2566 \new@ifnextchar [{\@GLSuserii@{#1}{#2}}{\@GLSuserii@{#1}{#2} []}}
```

Read in the final optional argument:

```
2567 \def\@GLSuserii@#1#2[#3] {%
2568 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2569 \protected@edef\@glo@text{\glsentryuserii{#2}}%
```

Call \@gls@link

```
2570 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#3}}%
2571 }%
2572 }
```

\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

```
2573 \newrobustcmd*{\glsuseriii}{\@ifstar\@sglsuseriii\@glsuseriii}
```

Define the starred form:

```
2574 \newcommand*{\@sglsuseriii}[1] [] {\@glsuseriii[hyper=false,#1]}
```


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

```
2575 \newcommand*{\@glsuseriii}[2] [] {%
2576 \new@ifnextchar [{\@glsuseriii@{#1}{#2}}]{\@glsuseriii@{#1}{#2} []}}
```

Read in the final optional argument:

```
2577 \def\@glsuseriii@#1#2[#3]{%
2578 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%

```

Determine what the link text should be (this is stored in \@glo@text)

```
2579 \protected@edef\@glo@text{\glsentryuseriii{#2}}%
```

Call \@gls@link

```
2580 \@gls@link[#1]{#2}{\@glo@text#3}%
2581 }%
2582 }
```

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

\Glsuseriii

```
2583 \newrobustcmd*{\Glsuseriii}{\@ifstar\@sGlsuseriii\@Glsuseriii}
```

Define the starred form:

```
2584 \newcommand*{\@sGlsuseriii}[1] [] {\@Glsuseriii[hyper=false,#1]}
```

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

```
2585 \newcommand*{\@Glsuseriii}[2] [] {%
2586 \new@ifnextchar [{\@Glsuseriii@{#1}{#2}}]{\@Glsuseriii@{#1}{#2} []}}
```

Read in the final optional argument:

```
2587 \def\@Glsuseriii@#1#2[#3]{%
2588 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%

```

Determine what the link text should be (this is stored in \@glo@text)

```
2589 \protected@edef\@glo@text{\glsentryuseriii{#2}}%
```

Call \@gls@link

```
2590 \@gls@link[#1]{#2}{%
2591 \expandafter\makefirstuc\expandafter{\@glo@text}#3}%
2592 }%
2593 }
```

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

\GLSuseriii

```
2594 \newrobustcmd*{\GLSuseriii}{\@ifstar\@sGLSuseriii\@GLSuseriii}
```

Define the starred form:

```
2595 \newcommand*{\@sGLSuseriii}[1] [] {\@GLSuseriii[hyper=false,#1]}
```

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

```
2596 \newcommand*{\@GLSuseriii}[2] [] {%
2597 \new@ifnextchar [{\@GLSuseriii@{#1}{#2}}{\@GLSuseriii@{#1}{#2} [] }}
```

Read in the final optional argument:

```
2598 \def\@GLSuseriii@#1#2[#3] {%
2599 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2600 \protected@edef\@glo@text{\glsentryuseriii{#2}}%
```

Call \@gls@link

```
2601 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#3}}%
2602 }%
2603 }
```

\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

```
2604 \newrobustcmd*{\glsuseriv}{\@ifstar\@sglsuseriv\@glsuseriv}
```

Define the starred form:

```
2605 \newcommand*{\@sglsuseriv}[1] [] {\@glsuseriv[hyper=false,#1]}
```

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

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

Read in the final optional argument:

```
2608 \def\@glsuseriv@#1#2[#3] {%
2609 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2610 \protected@edef\@glo@text{\glsentryuseriv{#2}}%
```

Call \@gls@link

```
2611 \@gls@link[#1]{#2}{\@glo@text#3}}%
2612 }%
2613 }
```

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

\Glsuseriv

```
2614 \newrobustcmd*{\Glsuseriv}{\@ifstar\@sGlsuseriv\@Glsuseriv}
```

Define the starred form:

```
2615 \newcommand*{\@sGlsuseriv}[1] [] {\@Glsuseriv[hyper=false,#1]}
```

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

```
2616 \newcommand*{\@Glsuseriv}[2] [] {%
2617 \new@ifnextchar [{\@Glsuseriv@{#1}{#2}}{\@Glsuseriv@{#1}{#2} []}}
```

Read in the final optional argument:

```
2618 \def\@Glsuseriv@#1#2[#3] {%
2619 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2620 \protected@edef\@glo@text{\glsentryuseriv{#2}}%
```

Call \@gls@link

```
2621 \@gls@link[#1]{#2}{%
2622 \expandafter\makefirstuc\expandafter{\@glo@text}#3}%
2623 }%
2624 }
```

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

\GLSuseriv

```
2625 \newrobustcmd*{\GLSuseriv}{\@ifstar\@sGLSuseriv\@GLSuseriv}
```

Define the starred form:

```
2626 \newcommand*{\@sGLSuseriv}[1] [] {\@GLSuseriv[hyper=false,#1]}
```

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

```
2627 \newcommand*{\@GLSuseriv}[2] [] {%
2628 \new@ifnextchar [{\@GLSuseriv@{#1}{#2}}{\@GLSuseriv@{#1}{#2} []}}
```

Read in the final optional argument:

```
2629 \def\@GLSuseriv@#1#2[#3] {%
2630 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2631 \protected@edef\@glo@text{\glsentryuseriv{#2}}%
```

Call \@gls@link

```
2632 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#3}}%
2633 }%
2634 }
```

\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

```
2635 \newrobustcmd*{\glsuserv}{\@ifstar\@sglsuserv\@glsuserv}
```

Define the starred form:

```
2636 \newcommand*{\@sglsuserv}[1] [] {\@glsuserv[hyper=false,#1]}
```

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

```
2637 \newcommand*{\@glsuserv}[2] [] {%
2638 \new@ifnextchar [\@glsuserv@{#1}{#2}]{\@glsuserv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
2639 \def\@glsuserv@#1#2[#3] {%
2640 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2641 \protected@edef\@glo@text{\glsentryuser{#2}}%
```

Call \@gls@link

```
2642 \@gls@link[#1]{#2}{\@glo@text#3}%
2643 }%
2644 }
```

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

\Glsuserv

```
2645 \newrobustcmd*{\Glsuserv}{\@ifstar\@sGlsuserv\@Glsuserv}
```

Define the starred form:

```
2646 \newcommand*{\@sGlsuserv}[1] [] {\@Glsuserv[hyper=false,#1]}
```

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

```
2647 \newcommand*{\@Glsuserv}[2] [] {%
2648 \new@ifnextchar [\@Glsuserv@{#1}{#2}]{\@Glsuserv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
2649 \def\@Glsuserv@#1#2[#3] {%
2650 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2651 \protected@edef\@glo@text{\glsentryuser{#2}}%
```

Call \@gls@link

```
2652 \@gls@link[#1]{#2}{%
2653 \expandafter\makefirstuc\expandafter{\@glo@text}#3}%
2654 }%
2655 }
```

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

\GLSuserv

```
2656 \newrobustcmd*{\GLSuserv}{\@ifstar\@sGLSuserv\@GLSuserv}
```

Define the starred form:

```
2657 \newcommand*{\@sGLSuserv}[1] [] {\@GLSuserv[hyper=false,#1]}
```

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

```
2658 \newcommand*{\@GLSuserv}[2] [] {%
2659 \new@ifnextchar [{\@GLSuserv@{#1}{#2}}{\@GLSuserv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
2660 \def\@GLSuserv@#1#2[#3] {%
2661 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2662 \protected@edef\@glo@text{\glsentryuser{#2}}%
```

Call \@gls@link

```
2663 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#3}}%
2664 }%
2665 }
```

\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

```
2666 \newrobustcmd*{\glsuservi}{\@ifstar\sglsuservi\@glsuservi}
```

Define the starred form:

```
2667 \newcommand*{\@sglsuservi}[1] [] {\@glsuservi[hyper=false,#1]}
```

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

```
2668 \newcommand*{\@glsuservi}[2] [] {%
2669 \new@ifnextchar [{\@glsuservi@{#1}{#2}}{\@glsuservi@{#1}{#2} [] }}
```

Read in the final optional argument:

```
2670 \def\@glsuservi@#1#2[#3] {%
2671 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2672 \protected@edef\@glo@text{\glsentryuser{#2}}%
```

Call \@gls@link

```
2673 \@gls@link[#1]{#2}{\@glo@text#3}}%
2674 }%
2675 }
```

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

\Glsuservi

```
2676 \newrobustcmd*{\Glsuservi}{\@ifstar\sglsuservi\@Glsuservi}
```

Define the starred form:

```
2677 \newcommand*{\@sglsuservi}[1] [] {\@Glsuservi[hyper=false,#1]}
```

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

```
2678 \newcommand*{\@Glsuservi}[2] [] {%
2679 \new@ifnextchar [{\@Glsuservi@{#1}{#2}}{\@Glsuservi@{#1}{#2} []}}
```

Read in the final optional argument:

```
2680 \def\@Glsuservi@#1#2[#3] {%
2681 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2682 \protected@edef\@glo@text{\glsentryuservi{#2}}%
```

Call \@gls@link

```
2683 \@gls@link[#1]{#2}{%
2684 \expandafter\makefirstuc\expandafter{\@glo@text}#3}%
2685 }%
2686 }
```

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

\GLSuservi

```
2687 \newrobustcmd*{\GLSuservi}{\@ifstar\@sGLSuservi\@GLSuservi}
```

Define the starred form:

```
2688 \newcommand*{\@sGLSuservi}[1] [] {\@GLSuservi[hyper=false,#1]}
```

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

```
2689 \newcommand*{\@GLSuservi}[2] [] {%
2690 \new@ifnextchar [{\@GLSuservi@{#1}{#2}}{\@GLSuservi@{#1}{#2} []}}
```

Read in the final optional argument:

```
2691 \def\@GLSuservi@#1#2[#3] {%
2692 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2693 \protected@edef\@glo@text{\glsentryuservi{#2}}%
```

Call \@gls@link

```
2694 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#3}}%
2695 }%
2696 }
```

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

\acrshort

```
2697 \newrobustcmd*{\acrshort}{\@ifstar\s@acrshort\ns@acrshort}
```

Define the starred form:

```
2698 \newcommand*{\s@acrshort}[2] [] {%
2699 \new@ifnextchar [{\@acrshort{hyper=false,#1}{#2}}%
2700 {\@acrshort{hyper=false,#1}{#2} []}%
2701 }
```

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

```
2702 \newcommand*{\ns@acrshort}[2] [] {%
2703   \new@ifnextchar[{\@acrshort{#1}{#2}}{\@acrshort{#1}{#2} []}%
2704 }
```

Read in the final optional argument:

```
2705 \def\@acrshort#1#2[#3] {%
2706   \glsdoifexists{#2}%
2707   {%
2708     \edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2709     \protected@edef\@glo@text{\glsentryshort{#2}}%
```

Call \@gls@link

```
2710     \@gls@link[#1]{#2}{\acronymfont{\@glo@text}#3}%
2711   }%
2712 }
```

\Acrshort

```
2713 \newrobustcmd*{\Acrshort}{\@ifstar\s@Acrshort\ns@Acrshort}
```

Define the starred form:

```
2714 \newcommand*{\s@Acrshort}[2] [] {%
2715   \new@ifnextchar[{\@Acrshort{hyper=false,#1}{#2}}%
2716   {\@Acrshort{hyper=false,#1}{#2} []}%
2717 }
```

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

```
2718 \newcommand*{\ns@Acrshort}[2] [] {%
2719   \new@ifnextchar[{\@Acrshort{#1}{#2}}{\@Acrshort{#1}{#2} []}%
2720 }
```

Read in the final optional argument:

```
2721 \def\@Acrshort#1#2[#3] {%
2722   \glsdoifexists{#2}%
2723   {%
2724     \edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2725     \protected@edef\@glo@text{\glsentryshort{#2}}%
```

Call \@gls@link

```
2726     \@gls@link[#1]{#2}%
2727     {%
2728       \acronymfont{\expandafter\makefirstuc\expandafter{\@glo@text}}#3%
2729     }%
2730   }%
2731 }
```

\ACRshort

```
2732 \newrobustcmd*{\ACRshort}{\@ifstar\s@ACRshort\ns@ACRshort}
```

Define the starred form:

```
2733 \newcommand*{\s@ACRshort}[2] [] {%
2734   \new@ifnextchar[{\@ACRshort{hyper=false,#1}{#2}}%
2735   {\@ACRshort{hyper=false,#1}{#2} []}%
2736 }
```

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

```
2737 \newcommand*{\ns@ACRshort}[2] [] {%
2738   \new@ifnextchar[{\@ACRshort{#1}{#2}}{\@ACRshort{#1}{#2} []}%
2739 }
```

Read in the final optional argument:

```
2740 \def\@ACRshort#1#2[#3] {%
2741   \glsdoifexists{#2}%
2742   {%
2743     \edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2744   \protected@edef\@glo@text{\glsentryshort{#2}}%
```

Call \@gls@link

```
2745   \@gls@link[#1]{#2}{\acronymfont{\MakeUppercase{\@glo@text#3}}}%
2746   }%
2747 }
```

Short plural:

\acrshortpl

```
2748 \newrobustcmd*{\acrshortpl}{\@ifstar\s@acrshortpl\ns@acrshortpl}
```

Define the starred form:

```
2749 \newcommand*{\s@acrshortpl}[2] [] {%
2750   \new@ifnextchar[{\@acrshortpl{hyper=false,#1}{#2}}%
2751   {\@acrshortpl{hyper=false,#1}{#2} []}%
2752 }
```

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

```
2753 \newcommand*{\ns@acrshortpl}[2] [] {%
2754   \new@ifnextchar[{\@acrshortpl{#1}{#2}}{\@acrshortpl{#1}{#2} []}%
2755 }
```

Read in the final optional argument:

```
2756 \def\@acrshortpl#1#2[#3] {%
2757   \glsdoifexists{#2}%
2758   {%
2759     \edef\@glo@type{\glsentrytype{#2}}%
```


Determine what the link text should be (this is stored in \@glo@text)

```
2760 \protected@edef\@glo@text{\glentryshortpl{#2}}%
```

Call \@gls@link

```
2761 \@gls@link[#1]{#2}{\acronymfont{\@glo@text}{#3}}%
```

```
2762 }%
```

```
2763 }
```

\Acrshortpl

```
2764 \newrobustcmd*{\Acrshortpl}{\@ifstar\s@Acrshortpl\ns@Acrshortpl}
```

Define the starred form:

```
2765 \newcommand*{\s@Acrshortpl}[2][]{%}
```

```
2766 \new@ifnextchar[{\@Acrshortpl{hyper=false,#1}{#2}}%
```

```
2767 {\@Acrshortpl{hyper=false,#1}{#2}[]}%
```

```
2768 }
```

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

```
2769 \newcommand*{\ns@Acrshortpl}[2][]{%}
```

```
2770 \new@ifnextchar[{\@Acrshortpl{#1}{#2}}{\@Acrshortpl{#1}{#2}[]}%
```

```
2771 }
```

Read in the final optional argument:

```
2772 \def\@Acrshortpl#1#2[#3]{%
```

```
2773 \glsdoifexists{#2}}%
```

```
2774 {%
```

```
2775 \edef\@glo@type{\glentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2776 \protected@edef\@glo@text{\glentryshortpl{#2}}%
```

Call \@gls@link

```
2777 \@gls@link[#1]{#2}%
```

```
2778 {%
```

```
2779 \acronymfont{\expandafter\makefirstuc\expandafter{\@glo@text}}#3%
```

```
2780 }%
```

```
2781 }%
```

```
2782 }
```

\ACRshortpl

```
2783 \newrobustcmd*{\ACRshortpl}{\@ifstar\s@ACRshortpl\ns@ACRshortpl}
```

Define the starred form:

```
2784 \newcommand*{\s@ACRshortpl}[2][]{%}
```

```
2785 \new@ifnextchar[{\@ACRshortpl{hyper=false,#1}{#2}}%
```

```
2786 {\@ACRshortpl{hyper=false,#1}{#2}[]}%
```

```
2787 }
```

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

```
2788 \newcommand*{\ns@ACRshortpl}[2] [] {%
2789   \new@ifnextchar[{\@ACRshortpl{#1}{#2}}{\@ACRshortpl{#1}{#2} []}%
2790 }
```

Read in the final optional argument:

```
2791 \def\@ACRshortpl#1#2[#3] {%
2792   \glsdoifexists{#2}%
2793   {%
2794     \edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2795     \protected@edef\@glo@text{\glsentryshortpl{#2}}%
```

Call \@gls@link

```
2796     \@gls@link[#1]{#2}{\acronymfont{\MakeUppercase{\@glo@text#3}}}%
2797   }%
2798 }
```

\acrlong

```
2799 \newrobustcmd*{\acrlong}{\@ifstar\s@acrlong\ns@acrlong}
```

Define the starred form:

```
2800 \newcommand*{\s@acrlong}[2] [] {%
2801   \new@ifnextchar[{\@acrlong{hyper=false,#1}{#2}}%
2802   {\@acrlong{hyper=false,#1}{#2} []}%
2803 }
```

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

```
2804 \newcommand*{\ns@acrlong}[2] [] {%
2805   \new@ifnextchar[{\@acrlong{#1}{#2}}{\@acrlong{#1}{#2} []}%
2806 }
```

Read in the final optional argument:

```
2807 \def\@acrlong#1#2[#3] {%
2808   \glsdoifexists{#2}%
2809   {%
2810     \edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2811     \protected@edef\@glo@text{\glsentrylong{#2}}%
```

Call \@gls@link

```
2812     \@gls@link[#1]{#2}{\@glo@text#3}%
2813   }%
2814 }
```

\Acrlong

```
2815 \newrobustcmd*{\Acrlong}{\@ifstar\s@Acrlong\ns@Acrlong}
```

Define the starred form:

```
2816 \newcommand*{\s@Acrlong}[2] [] {%
2817   \new@ifnextchar[{\@Acrlong{hyper=false,#1}{#2}}}%
2818   {\@Acrlong{hyper=false,#1}{#2} []}%
2819 }
```

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

```
2820 \newcommand*{\ns@Acrlong}[2] [] {%
2821   \new@ifnextchar[{\@Acrlong{#1}{#2}}{\@Acrlong{#1}{#2} []}%
2822 }
```

Read in the final optional argument:

```
2823 \def\@Acrlong#1#2[#3] {%
2824   \glsdoifexists{#2}%
2825   {%
2826     \edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2827   \protected@edef\@glo@text{\glsentrylong{#2}}%
```

Call \@gls@link

```
2828   \@gls@link[#1]{#2}%
2829   {%
2830     \expandafter\makefirstuc\expandafter{\@glo@text}#3%
2831   }%
2832 }%
2833 }
```

\ACRlong

```
2834 \newrobustcmd*{\ACRlong}{\@ifstar\s@ACRlong\ns@ACRlong}
```

Define the starred form:

```
2835 \newcommand*{\s@ACRlong}[2] [] {%
2836   \new@ifnextchar[{\@ACRlong{hyper=false,#1}{#2}}}%
2837   {\@ACRlong{hyper=false,#1}{#2} []}%
2838 }
```

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

```
2839 \newcommand*{\ns@ACRlong}[2] [] {%
2840   \new@ifnextchar[{\@ACRlong{#1}{#2}}{\@ACRlong{#1}{#2} []}%
2841 }
```

Read in the final optional argument:

```
2842 \def\@ACRlong#1#2[#3] {%
2843   \glsdoifexists{#2}%
2844   {%
2845     \edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2846   \protected@edef\@glo@text{\glsentrylong{#2}}%
```

Call \@gls@link

```
2847 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#3}}%  
2848 }%  
2849 }
```

Short plural:

\acrlongpl

```
2850 \newrobustcmd*{\acrlongpl}{\@ifstar\s@acrlongpl\ns@acrlongpl}
```

Define the starred form:

```
2851 \newcommand*{\s@acrlongpl}[2] [] {%  
2852 \new@ifnextchar[{\@acrlongpl{hyper=false,#1}{#2}}%  
2853 {\@acrlongpl{hyper=false,#1}{#2} []}%  
2854 }
```

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

```
2855 \newcommand*{\ns@acrlongpl}[2] [] {%  
2856 \new@ifnextchar[{\@acrlongpl{#1}{#2}}{\@acrlongpl{#1}{#2} []}%  
2857 }
```

Read in the final optional argument:

```
2858 \def\@acrlongpl#1#2[#3] {%  
2859 \glsdoifexists{#2}%  
2860 {%  
2861 \edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2862 \protected@edef\@glo@text{\glsentrylongpl{#2}}%
```

Call \@gls@link

```
2863 \@gls@link[#1]{#2}{\@glo@text#3}%  
2864 }%  
2865 }
```

\Acrlongpl

```
2866 \newrobustcmd*{\Acrlongpl}{\@ifstar\s@Acrlongpl\ns@Acrlongpl}
```

Define the starred form:

```
2867 \newcommand*{\s@Acrlongpl}[2] [] {%  
2868 \new@ifnextchar[{\@Acrlongpl{hyper=false,#1}{#2}}%  
2869 {\@Acrlongpl{hyper=false,#1}{#2} []}%  
2870 }
```

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

```
2871 \newcommand*{\ns@Acrlongpl}[2] [] {%  
2872 \new@ifnextchar[{\@Acrlongpl{#1}{#2}}{\@Acrlongpl{#1}{#2} []}%  
2873 }
```

Read in the final optional argument:

```

2874 \def\@Acrlongpl#1#2[#3]{%
2875   \glsdoifexists{#2}%
2876   {%
2877     \edef\@glo@type{\glsentrytype{#2}}%
2878     \protected@edef\@glo@text{\glsentrylongpl{#2}}%
2879     \expandafter\makefirstuc\expandafter{\@glo@text}%
2880     \expandafter\@gls@link\@glo@text#3%
2881     }%
2882   }%
2883 }%
2884 }

```

\ACRlongpl

```

2885 \newrobustcmd*{\ACRlongpl}{\@ifstar\s@ACRlongpl\@ns@ACRlongpl}

```

Define the starred form:

```

2886 \newcommand*{\s@ACRlongpl}[2][{}]{%
2887   \new@ifnextchar[{\@ACRlongpl{hyper=false,#1}{#2}}%
2888   {\@ACRlongpl{hyper=false,#1}{#2}[]}%
2889 }

```

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

```

2890 \newcommand*{\ns@ACRlongpl}[2][{}]{%
2891   \new@ifnextchar[{\@ACRlongpl{#1}{#2}}{\@ACRlongpl{#1}{#2}[]}%
2892 }

```

Read in the final optional argument:

```

2893 \def\@ACRlongpl#1#2[#3]{%
2894   \glsdoifexists{#2}%
2895   {%
2896     \edef\@glo@type{\glsentrytype{#2}}%
2897     \protected@edef\@glo@text{\glsentrylongpl{#2}}%
2898     \expandafter\makefirstuc\expandafter{\@glo@text}%
2899     \expandafter\@gls@link\@glo@text#3%
2900   }%

```

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.

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`

```
2901 \newcommand*{\glsentryname}[1]{\csname glo@#1@name\endcsname}
```

`\Glsentryname`

```
2902 \newcommand*{\Glsentryname}[1]{%
2903 \protected@edef\@glo@text{\csname glo@#1@name\endcsname}%
2904 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

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`

```
2905 \newcommand*{\glsentrydesc}[1]{\csname glo@#1@desc\endcsname}
```

`\Glsentrydesc`

```
2906 \newcommand*{\Glsentrydesc}[1]{%
2907 \protected@edef\@glo@text{\csname glo@#1@desc\endcsname}%
2908 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

Plural form:

`\glsentrydescplural`

```
2909 \newcommand*{\glsentrydescplural}[1]{%
2910 \csname glo@#1@descplural\endcsname}
```

`\Glsentrydescplural`

```
2911 \newcommand*{\Glsentrydescplural}[1]{%
2912 \protected@edef\@glo@text{\csname glo@#1@descplural\endcsname}%
2913 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

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`

```
2914 \newcommand*{\glsentrytext}[1]{\csname glo@#1@text\endcsname}
```

`\Glsentrytext`

```
2915 \newcommand*{\Glsentrytext}[1]{%
2916 \protected@edef\@glo@text{\csname glo@#1@text\endcsname}%
2917 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

Get the plural form:

`\glsentryplural`

```
2918 \newcommand*{\glsentryplural}[1]{\csname glo@#1@plural\endcsname}
```

`\Glsentryplural`

```
2919 \newcommand*{\Glsentryplural}[1]{%
2920 \protected@edef\@glo@text{\csname glo@#1@plural\endcsname}%
2921 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

Get the symbol associated with this entry. The argument is the label associated with the entry. Note that unless you used `symbol=false` in the `sanitize` package option you may get unexpected results if the symbol key contained any commands.

`\glsentrysymbol`

```
2922 \newcommand*{\glsentrysymbol}[1]{\csname glo@#1@symbol\endcsname}
```

`\Glsentrysymbol`

```
2923 \newcommand*{\Glsentrysymbol}[1]{%
2924 \protected@edef\@glo@text{\csname glo@#1@symbol\endcsname}%
2925 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

Plural form:

`\glsentrysymbolplural`

```
2926 \newcommand*{\glsentrysymbolplural}[1]{%
2927 \csname glo@#1@symbolplural\endcsname}
```

`\Glsentrysymbolplural`

```
2928 \newcommand*{\Glsentrysymbolplural}[1]{%
2929 \protected@edef\@glo@text{\csname glo@#1@symbolplural\endcsname}%
2930 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

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`

```
2931 \newcommand*{\glsentryfirst}[1]{\csname glo@#1@first\endcsname}
```

`\Glsentryfirst`

```
2932 \newcommand*{\Glsentryfirst}[1]{%
2933 \protected@edef\@glo@text{\csname glo@#1@first\endcsname}%
2934 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

Get the plural form (as specified by the `firstplural` key when the entry was defined).

`\glsentryfirstplural`

```
2935 \newcommand*{\glsentryfirstplural}[1]{%
2936 \csname glo@#1@firstpl\endcsname}
```

Glsentryfirstplural

```
2937 \newcommand*{\Glsentryfirstplural}[1]{%
2938 \protected@edef\@glo@text{\csname glo@#1@firstpl\endcsname}%
2939 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

Display the glossary type with which this entry is associated (as specified by the type key used when the entry was defined)

\glsentrytype

```
2940 \newcommand*{\glsentrytype}[1]{\csname glo@#1@type\endcsname}
```

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

```
2941 \newcommand*{\glsentrysort}[1]{\csname glo@#1@sort\endcsname}
```

\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.

```
2942 \newcommand*{\glsentryuseri}[1]{\csname glo@#1@useri\endcsname}
```

\Glsentryuseri

```
2943 \newcommand*{\Glsentryuseri}[1]{%
2944 \protected@edef\@glo@text{\csname glo@#1@useri\endcsname}%
2945 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

\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.

```
2946 \newcommand*{\glsentryuserii}[1]{\csname glo@#1@userii\endcsname}
```

\Glsentryuserii

```
2947 \newcommand*{\Glsentryuserii}[1]{%
2948 \protected@edef\@glo@text{\csname glo@#1@userii\endcsname}%
2949 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

\glsentryuseriii Get the third user key (as specified by the user3 when the entry was defined).
The argument is the label associated with the entry.

```
2950 \newcommand*{\glsentryuseriii}[1]{\csname glo@#1@useriii\endcsname}
```

\Glsentryuseriii

```
2951 \newcommand*{\Glsentryuseriii}[1]{%
2952 \protected@edef\@glo@text{\csname glo@#1@useriii\endcsname}%
2953 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

\glsentryuseriv Get the fourth user key (as specified by the user4 when the entry was defined).
The argument is the label associated with the entry.

```
2954 \newcommand*{\glsentryuseriv}[1]{\csname glo@#1@useriv\endcsname}
```


\Glsentryuseriv

```
2955 \newcommand*{\Glsentryuseriv}[1]{%
2956 \protected@edef\@glo@text{\csname glo@#1@useriv\endcsname}%
2957 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

\glsentryuseriv Get the fifth user key (as specified by the user5 when the entry was defined).
The argument is the label associated with the entry.

```
2958 \newcommand*{\glsentryuseriv}[1]{\csname glo@#1@useriv\endcsname}
```

\Glsentryuseriv

```
2959 \newcommand*{\Glsentryuseriv}[1]{%
2960 \protected@edef\@glo@text{\csname glo@#1@useriv\endcsname}%
2961 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

\glsentryuservi Get the sixth user key (as specified by the user6 when the entry was defined).
The argument is the label associated with the entry.

```
2962 \newcommand*{\glsentryuservi}[1]{\csname glo@#1@uservi\endcsname}
```

\Glsentryuservi

```
2963 \newcommand*{\Glsentryuservi}[1]{%
2964 \protected@edef\@glo@text{\csname glo@#1@uservi\endcsname}%
2965 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

\glsentryshort Get the short key (as specified by the short the entry was defined). The argu-
ment is the label associated with the entry.

```
2966 \newcommand*{\glsentryshort}[1]{\csname glo@#1@short\endcsname}
```

\Glsentryshort

```
2967 \newcommand*{\Glsentryshort}[1]{%
2968 \protected@edef\@glo@text{\csname glo@#1@short\endcsname}%
2969 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

\glsentryshortpl Get the short plural key (as specified by the shortplural the entry was defined).
The argument is the label associated with the entry.

```
2970 \newcommand*{\glsentryshortpl}[1]{\csname glo@#1@shortpl\endcsname}
```

\Glsentryshortpl

```
2971 \newcommand*{\Glsentryshortpl}[1]{%
2972 \protected@edef\@glo@text{\csname glo@#1@shortpl\endcsname}%
2973 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

\glsentrylong Get the long key (as specified by the long the entry was defined). The argument
is the label associated with the entry.

```
2974 \newcommand*{\glsentrylong}[1]{\csname glo@#1@long\endcsname}
```

```

\Glsentrylong
2975 \newcommand*{\Glsentrylong}[1]{%
2976 \protected@edef\@glo@text{\csname glo@#1@long\endcsname}%
2977 \expandafter\makefirstuc\expandafter{\@glo@text}}

\glsentrylongpl  Get the long plural key (as specified by the longplural the entry was defined).
                  The argument is the label associated with the entry.
2978 \newcommand*{\glsentrylongpl}[1]{\csname glo@#1@longpl\endcsname}

\Glsentrylongpl
2979 \newcommand*{\Glsentrylongpl}[1]{%
2980 \protected@edef\@glo@text{\csname glo@#1@longpl\endcsname}%
2981 \expandafter\makefirstuc\expandafter{\@glo@text}}

Short cut macros to access full form:

\glsentryfull
2982 \newcommand*{\glsentryfull}[1]{%
2983   \glsentrylong{#1}\space(\glsentryshort{#1})%
2984 }

\Glsentryfull
2985 \newcommand*{\Glsentryfull}[1]{%
2986   \Glsentrylong{#1}\space(\glsentryshort{#1})%
2987 }

\glsentryfullpl
2988 \newcommand*{\glsentryfullpl}[1]{%
2989   \glsentrylongpl{#1}\space(\glsentryshortpl{#1})%
2990 }

\Glsentryfullpl
2991 \newcommand*{\Glsentryfullpl}[1]{%
2992   \Glsentrylongpl{#1}\space(\glsentryshortpl{#1})%
2993 }

\glsentrynumberlist  Displays the number list as is.
2994 \newcommand*{\glsentrynumberlist}[1]{%
2995   \glsdoifexists{#1}%
2996   {%
2997     \csname glo@#1@numberlist\endcsname
2998   }%
2999 }

\displaynumberlist  Formats the number list for the given entry label. Doesn't work with hyperref.
3000 \@ifpackageloaded{hyperref}
3001 {%
3002   \newcommand*{\displaynumberlist}[1]{%

```

```

3003 \GlossariesWarning
3004 {%
3005 \string\glsdisplaynumberlist\space
3006 doesn't work with hyperref.^^JUsing
3007 \string\glsentrynumberlist\space instead%
3008 }%
3009 \glsentrynumberlist{#1}%
3010 }%
3011 }%
3012 {%
3013 \newcommand*{\glsdisplaynumberlist}[1]{%
3014 \glsdoifexists{#1}%
3015 {%
3016 \bgroup
3017 \def\@glo@label{#1}%
3018 \let\@org@glsglnumberformat\glsglnumberformat
3019 \def\glsglnumberformat##1{##1}%
3020 \protected@edef\the@numberlist{\csname glo@\@glo@label @numberlist\endcsname}%
3021 \def\@glsgl@numlist@sep{}%
3022 \def\@glsgl@numlist@nextsep{}%
3023 \def\@glsgl@numlist@lastsep{}%
3024 \def\@glsgl@thislist{}%
3025 \def\@glsgl@donext@def{}%
3026 \renewcommand\do[1]{%
3027 \protected@edef\@glsgl@thislist{%
3028 \@glsgl@thislist
3029 \noexpand\@glsgl@numlist@sep
3030 ##1%
3031 }%
3032 \let\@glsgl@numlist@sep\@glsgl@numlist@nextsep
3033 \def\@glsgl@numlist@nextsep{\glsgl@numlist@sep}%
3034 \@glsgl@donext@def
3035 \def\@glsgl@donext@def{%
3036 \def\@glsgl@numlist@lastsep{\glsgl@numlist@lastsep}%
3037 }%
3038 }%
3039 \expandafter \glsgl@numlist@parser \expandafter{\the@numberlist}%
3040 \let\@glsgl@numlist@sep\@glsgl@numlist@lastsep
3041 \@glsgl@thislist
3042 \egroup
3043 }%
3044 }
3045 }

```

\glsgl@numlist@sep

```

3046 \newcommand*{\glsgl@numlist@sep}{, }

```

\glsgl@numlist@lastsep

```

3047 \newcommand*{\glsgl@numlist@lastsep}{ \& }

```

`\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.

```
3048 \newcommand*{\glshyperlink}[2][\glsentrytext{\@glo@label}]{%
3049 \def\@glo@label{#2}%
3050 \@glslink{\glo@linkprefix#2}{#1}}
```

1.11 Adding an entry to the glossary without generating text

The following keys are provided for `\glsadd` and `\glsaddall`:

```
3051 \define@key{glossadd}{counter}{\def\@gls@counter{#1}}
3052 \define@key{glossadd}{format}{\def\@glsnumberformat{#1}}
```

This key is only used by `\glsaddall`:

```
3053 \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`

```
3054 \newrobustcmd*{\glsadd}[2][\def\@gls@counter{#1}]{%
3055 \glsdoifexists{#2}%
3056 {%
3057 \def\@glsnumberformat{#2}%
3058 \edef\@gls@counter{\csname glo@#2@counter\endcsname}%
3059 \setkeys{glossadd}{#1}%
```

Store the entry's counter in `\theglsentrycounter`

```
3060 \@gls@saveentrycounter
3061 \do@wrglossary{#2}%
3062 }%
3063 }
```

`\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`

```
3064 \newrobustcmd*{\glsaddall}[1][\def\@gls@counter{#1}]{%
3065 \edef\@glo@type{\@glo@types}%
3066 \setkeys{glossadd}{#1}%
```

```

3067 \forallglsentries[\@glo@type]{\@glo@entry}{%
3068 \glsadd[#1]{\@glo@entry}}%
3069 }

```

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 `glsymbols` and `glsnumbers`, the group titles can be translated (so that `\glsymbolsgroupname` replaces `glsymbols` and `\glsnumbersgroupname` replaces `glsnumbers`) using the command `\glsgetgrouptitle` which is defined in `.`. This is done to prevent any problem characters in `\glsymbolsgroupname` and `\glsnumbersgroupname` from breaking hyperlinks.

`\glsopenbrace` Define `\glsopenbrace` to make it easier to write an opening brace to a file.

```

3070 \edef\glsopenbrace{\expandafter\@gobble\string\{ }

```

`\glsclosebrace` Define `\glsclosebrace` to make it easier to write an opening brace to a file.

```

3071 \edef\glsclosebrace{\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.

```

3072 \edef\glsquote#1{\string"#1\string"}

```

`\@glsfirstletter` Define the first letter to come after the digits 0,...,9. Only required for `xindy`.

```

3073 \ifglsxindy
3074   \newcommand*{\@glsfirstletter}{A}
3075 \fi

```

`stLetterAfterDigits` Sets the first letter to come after the digits 0,...,9.

```

3076 \ifglsxindy
3077   \newcommand*{\GlsSetXdyFirstLetterAfterDigits}[1]{%
3078     \renewcommand*{\@glsfirstletter}{#1}}
3079 \else
3080   \newcommand*{\GlsSetXdyFirstLetterAfterDigits}[1]{%
3081     \glsnoxindywarning\GlsSetXdyFirstLetterAfterDigits}
3082 \fi

```

`\@glsminrange` Define the minimum number of successive location references to merge into a range.

```

3083 \newcommand*{\@glsminrange}{2}

```

`\GlsSetXdyMinRangeLength` Set the minimum range length. The value must either be none or a positive integer. The glossaries package doesn't check if the argument is valid, that is left to xindy.

```

3084 \ifglsxindy
3085   \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
3086     \renewcommand*{\@glsminrange}{#1}}
3087 \else
3088   \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
3089     \glsnoxywarning\GlsSetXdyMinRangeLength}
3090 \fi

```

`\writeist`

```

3091 \ifglsxindy

```

Code to use if xindy is required.

```

3092 \def\writeist{%

```

Update attributes list

```

3093   \@gls@addpredefinedattributes

```

Open the file.

```

3094   \openout\glswrite=\istfilename

```

Write header comment at the start of the file

```

3095   \write\glswrite{;; xindy style file created by the glossaries
3096     package}%
3097   \write\glswrite{;; for document '\jobname' on
3098     \the\year-\the\month-\the\day}%

```

Specify the required styles

```

3099   \write\glswrite{^^J; required styles^^J}
3100   \@for\@xdystyle:=\@xdyrequiredstyles\do{%
3101     \ifx\@xdystyle\empty
3102     \else
3103       \protected@write\glswrite{{(require
3104         \string"\@xdystyle.xdy\string")}}%
3105     \fi
3106   }%

```

List the allowed attributes (possible values used by the format key)

```

3107   \write\glswrite{^^J%
3108     ; list of allowed attributes (number formats)^^J}%
3109   \write\glswrite{(define-attributes ((\@xdyattributes)))}%

```

Define any additional alphabets

```

3110   \write\glswrite{^^J; user defined alphabets^^J}%
3111   \write\glswrite{\@xdyuseralphabets}%

```

Define location classes.

```
3112 \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.

```
3113 \@for\@gls@classI:=\@gls@xdy@locationlist\do{%
```

Case were $\langle Hprefix \rangle$ is empty:

```
3114 \protected@write\glswrite{}\{(define-location-class
3115 \string"\@gls@classI\string"^^J\space\space\space
3116 (
3117 :sep "{"
3118 \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
3119 :sep "}"
3120 )
3121 ^^J\space\space\space
3122 :min-range-length \@glsminrange^^J%
3123 )
3124 }%
```

Nested iteration over all classes:

```
3125 {%
3126 \@for\@gls@classII:=\@gls@xdy@locationlist\do{%
3127 \protected@write\glswrite{}\{(define-location-class
3128 \string"\@gls@classII-\@gls@classI\string"
3129 ^^J\space\space\space
3130 (
3131 :sep "{"
3132 \csname @gls@xdy@Lclass@\@gls@classII\endcsname\space
3133 :sep "}"
3134 \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
3135 :sep "}"
3136 )
3137 ^^J\space\space\space
3138 :min-range-length \@glsminrange^^J%
3139 )
3140 }%
3141 }%
3142 }%
3143 }%
```

User defined location classes (needs checking for new location format).

```
3144 \write\glswrite{^^J; user defined location classes}%
3145 \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.)

```
3146 \write\glswrite{^^J; define cross-reference class^^J}%
3147 \write\glswrite{(define-crossref-class \string"see\string"
3148 :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.)

```
3149 \write\glswrite{(markup-crossref-list
3150      :class \string"see\string"^^J\space\space\space
3151      :open \string"\string\glsseeformat\string"
3152      :close \string"{}\string")}%
```

List the order to sort the classes.

```
3153 \write\glswrite{^^J; define the order of the location classes}%
3154 \write\glswrite{(define-location-class-order
3155      (\@xdylocationclassorder))}%
```

Specify what to write to the start and end of the glossary file.

```
3156 \write\glswrite{^^J; define the glossary markup^^J}%

3157 \write\glswrite{(markup-index^^J\space\space\space
3158      :open \string"\string
3159      \glossarysection[\string\glossarytoctitle]{\string
3160      \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)

```
3161 \@for\@this@ctr:=\@xdycounters\do{%
3162     {%
3163         \@for\@this@attr:=\@xdyattributelist\do{%
3164             \protected\write\glswrite{{}\string\providecommand*%
3165                 \expandafter\string
3166                 \csname glsX\@this@ctr X\@this@attr\endcsname[2]%
3167                 {%
3168                     \string\setentrycounter
3169                     [\expandafter\@gobble\string\#1]{\@this@ctr}%
3170                     \expandafter\string
3171                     \csname\@this@attr\endcsname
3172                     {\expandafter\@gobble\string\#2}%
3173                 }%
3174             }%
3175         }%
3176     }%
3177 }
```

Add the end part of the open tag and the rest of the markup-index information:

```
3178 \write\glswrite{%
3179     \string\begin
3180     {theglossary}\string\glossaryheader\string~n\string" ^^J\space
3181     \space\space:close \string"\expandafter\@gobble
3182     \string%\string~n\string
3183     \end{theglossary}\string\glossarypostamble
3184     \string~n\string" ^^J\space\space\space
3185     :tree)}}%
```


Specify what to put between letter groups

```
3186 \write\glswrite{(markup-letter-group-list
3187 :sep \string"\string\glsgroupskip\string~n\string")}%
```

Specify what to put between entries

```
3188 \write\glswrite{(markup-indexentry
3189 :open \string"\string\relax \string\glresetentrylist
3190 \string~n\string")}%
```

Specify how to format entries

```
3191 \write\glswrite{(markup-locclass-list :open
3192 \string"\glsoopenbrace\string\glossaryentrynumbers
3193 \glsoopenbrace\string\relax\space \string"^^J\space\space\space
3194 :sep \string", \string"
3195 :close \string"\glsclosebrace\glsclosebrace\string")}%
```

Specify how to separate location numbers

```
3196 \write\glswrite{(markup-locref-list
3197 :sep \string"\string\delimN\space\string")}%
```

Specify how to indicate location ranges

```
3198 \write\glswrite{(markup-range
3199 :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.

```
3200 \@onelevel@sanitize\gls@suffixF
3201 \@onelevel@sanitize\gls@suffixFF
3202 \ifx\gls@suffixF\@empty
3203 \else
3204 \write\glswrite{(markup-range
3205 :close "\gls@suffixF" :length 1 :ignore-end)}%
3206 \fi
3207 \ifx\gls@suffixFF\@empty
3208 \else
3209 \write\glswrite{(markup-range
3210 :close "\gls@suffixFF" :length 2 :ignore-end)}%
3211 \fi
```

Specify how to format locations.

```
3212 \write\glswrite{^^J; define format to use for locations^^J}%
3213 \write\glswrite{\@xdylocref}%
```

Specify how to separate letter groups.

```
3214 \write\glswrite{^^J; define letter group list format^^J}%
3215 \write\glswrite{(markup-letter-group-list
3216 :sep \string"\string\glsgroupskip\string~n\string")}%
```

Define letter group headings.

```
3217 \write\glswrite{^^J; letter group headings^^J}%
3218 \write\glswrite{(markup-letter-group
```

```

3219      :open-head \string"\string\glsgroupheading
3220      \glsopenbrace\string"^^J\space\space\space
3221      :close-head \string"\glsclosebrace\string"))}%

```

Define additional letter groups.

```

3222      \write\glswrite{^^J; additional letter groups^^J}%
3223      \write\glswrite{\@xdylettergroups}%

```

Define additional sort rules

```

3224      \write\glswrite{^^J; additional sort rules^^J}
3225      \write\glswrite{\@xdysortrules}%

```

Close the style file

```

3226      \closeout\glswrite

```

Suppress any further calls.

```

3227      \let\writeist\relax
3228  }
3229 \else

```

Code to use if makeindex is required.

```

3230  \edef\@gls@actualchar{\string?}
3231  \edef\@gls@encapchar{\string|}
3232  \edef\@gls@levelchar{\string!}
3233  \edef\@gls@quotechar{\string"}
3234  \def\writeist{\relax
3235  \openout\glswrite=\istfilename
3236  \write\glswrite{\expandafter\@gobble\string\% makeindex style file
3237  created by the glossaries package}
3238  \write\glswrite{\expandafter\@gobble\string\% for document
3239  '\jobname' on \the\year-\the\month-\the\day}
3240  \write\glswrite{actual '\@gls@actualchar'}
3241  \write\glswrite{encap '\@gls@encapchar'}
3242  \write\glswrite{level '\@gls@levelchar'}
3243  \write\glswrite{quote '\@gls@quotechar'}
3244  \write\glswrite{keyword \string"\string\glossaryentry\string"}
3245  \write\glswrite{preamble \string"\string\glossarysection[\string
3246  \glossarytoctitle]{\string\glossarytitle}\string
3247  \glossarypreamble\string\n\string\begin{theglossary}\string
3248  \glossaryheader\string\n\string"}
3249  \write\glswrite{postamble \string"\string\%\string\n\string
3250  \end{theglossary}\string\glossarypostamble\string\n
3251  \string"}
3252  \write\glswrite{group_skip \string"\string\glsgroupskip\string\n
3253  \string"}
3254  \write\glswrite{item_0 \string"\string\%\string\n\string"}
3255  \write\glswrite{item_1 \string"\string\%\string\n\string"}
3256  \write\glswrite{item_2 \string"\string\%\string\n\string"}
3257  \write\glswrite{item_01 \string"\string\%\string\n\string"}
3258  \write\glswrite{item_x1
3259  \string"\string\relax \string\glsresetentrylist\string\n

```

```

3260     \string"}
3261     \write\glswrite{item_12 \string"\string%\string\n\string"}
3262     \write\glswrite{item_x2
3263         \string"\string\\relax \string\\glsresetentrylist\string\n
3264         \string"}

3265     \write\glswrite{delim_0 \string"\string\{\string
3266         \\glossaryentrynumbers\string\{\string\\relax \string"}
3267     \write\glswrite{delim_1 \string"\string\{\string
3268         \\glossaryentrynumbers\string\{\string\\relax \string"}
3269     \write\glswrite{delim_2 \string"\string\{\string
3270         \\glossaryentrynumbers\string\{\string\\relax \string"}
3271     \write\glswrite{delim_t \string"\string\}\string\}\string"}
3272     \write\glswrite{delim_n \string"\string\\delimN \string"}
3273     \write\glswrite{delim_r \string"\string\\delimR \string"}
3274     \write\glswrite{headings_flag 1}
3275     \write\glswrite{heading_prefix
3276         \string"\string\\glsgroupheading\string\{\string"}
3277     \write\glswrite{heading_suffix
3278         \string"\string\}\string\\relax
3279         \string\\glsresetentrylist \string"}
3280     \write\glswrite{symhead_positive \string"glssymbols\string"}
3281     \write\glswrite{numhead_positive \string"glslnumbers\string"}
3282     \write\glswrite{page_compositor \string"glsc compositor\string"}
3283     \@gls@escbsdq\gls@suffixF
3284     \@gls@escbsdq\gls@suffixFF
3285     \ifx\gls@suffixF\@empty
3286     \else
3287         \write\glswrite{suffix_2p \string"\gls@suffixF\string"}
3288     \fi
3289     \ifx\gls@suffixFF\@empty
3290     \else
3291         \write\glswrite{suffix_3p \string"\gls@suffixFF\string"}
3292     \fi
3293     \closeout\glswrite
3294     \let\writeist\relax
3295 }
3296 \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`

```

3297 \newcommand{\noist}{%
    Update attributes list
3298     \@gls@addpredefinedattributes
3299     \let\writeist\relax
3300 }

```

`\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`

```
3301 \newcommand*{\makeglossary}[1]{%
3302   \ifglossaryexists{#1}%
3303   {%
```

Only create a new write if `savewrites=false` otherwise create a token to collect the information.

```
3304   \ifglssavewrites
3305     \expandafter\newtoks\csname glo@#1@filetok\endcsname
3306   \else
3307     \expandafter\newwrite\csname glo@#1@file\endcsname
3308     \expandafter\@glsopenfile\csname glo@#1@file\endcsname{#1}%
3309   \fi
3310   \@gls@renewglossary
3311   \writeist
3312 }%
3313 {%
3314   \PackageError{glossaries}%
3315   {Glossary type ‘#1’ not defined}%
3316   {New glossaries must be defined before using \string\makeglossary}%
3317 }%
3318 }
```

`\@glsopenfile` Open write file associated with the given glossary.

```
3319 \newcommand*{\@glsopenfile}[2]{%
3320   \immediate\openout#1=\jobname.\csname @glo@#2@out\endcsname
3321   \PackageInfo{glossaries}{Writing glossary file
3322     \jobname.\csname @glo@#2@out\endcsname}%
3323 }
```

`\@nomakeglossaries` Issue warning that `\makeglossaries` hasn't been used.

```
3324 \newcommand*{\@nomakeglossaries}{%
3325   \GlossariesWarningNoLine{\string\makeglossaries\space
3326   hasn't been used,^^Jthe glossaries will not be updated}%
3327 }
```

`\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`

```
3328 \newcommand*{\makeglossaries}{%
```

Write the name of the style file to the aux file (needed by `makeglossaries`)

```
3329 \protected@write\@auxout{}\string\@istfilename{\istfilename}}%
```

```
3330 \protected@write\@auxout{}\string\@glsorder{\glsorder}}%
```

Iterate through each glossary type and activate it.

```
3331 \@for\@glo@type:=\@glo@types\do{%
```

```
3332 \ifthenelse{\equal{\@glo@type}{}}{\}%
```

```
3333 \makeglossary{\@glo@type}}%
```

```
3334 }%
```

New glossaries must be created before `\makeglossaries` so disable `\newglossary`.

```
3335 \renewcommand*\newglossary[4][]{%
```

```
3336 \PackageError{glossaries}{New glossaries
```

```
3337 must be created before \string\makeglossaries}{You need
```

```
3338 to move \string\makeglossaries\space after all your
```

```
3339 \string\newglossary\space commands}}%
```

Any subsequent instances of this command should have no effect

```
3340 \let\@makeglossary\relax
```

```
3341 \let\makeglossary\relax
```

```
3342 \let\makeglossaries\relax
```

Disable all commands that have no effect after `\makeglossaries`

```
3343 \@disable@onlypremakeg
```

Suppress warning about no `\makeglossaries`

```
3344 \let\warn@nomakeglossaries\relax
```

Declare list parser for `\glsdisplaynumberlist`

```
3345 \ifglssavenumberlist
```

```
3346 \edef\@gls@doddeflistparser{\noexpand\DeclareListParser
```

```
3347 {\noexpand\glsnumlistparser}{\delimN}}%
```

```
3348 \@gls@doddeflistparser
```

```
3349 \fi
```

```
3350 }
```

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`

```
3351 \let\makeglossary\makeglossaries
```

If `\makeglossaries` hasn't been used, issue a warning. Also issue a warning if neither `\printglossaries` nor `\printglossary` have been used.

```
3352 \AtEndDocument{%
```

```

3353 \warn@nomakeglossaries
3354 \warn@noprintglossary
3355 }

```

1.13 Writing information to associated files

`\glswrite` The write used for style file also used for all other output files if `savewrites=true`.

```
3356 \newwrite\glswrite
```

`\istfile` Deprecated.

```
3357 \def\istfile{\glswrite}
```

At the end of the document, the files should be created if `savewrites=true`.

```

3358 \AtEndDocument{%
3359 \glswritefiles
3360 }

```

`\glswritefiles` Only write the files if `savewrites=true`

```

3361 \ifglssavewrites
3362 \newcommand*{\glswritefiles}{%

```

Iterate through all the glossaries

```
3363 \forallglossaries{\@glo@type}{%
```

Check for empty glossaries (patch provided by Patrick Häcker)

```

3364 \ifcsundef{glo@\@glo@type @filetok}%
3365 {%
3366 \def\gls@tmp{}%
3367 }%
3368 {%
3369 \edef\gls@tmp{\expandafter\the
3370 \csname glo@\@glo@type @filetok\endcsname}%
3371 }%
3372 \ifx\gls@tmp\@empty
3373 \ifx\@glo@type\glsdefaulttype
3374 \GlossariesWarningNoLine{Glossary '\@glo@type' has no
3375 entries.^^JRemember to use package option 'nomain' if
3376 you
3377 don't want to^^Juse the main glossary}%
3378 \else
3379 \GlossariesWarningNoLine{Glossary '\@glo@type' has no
3380 entries}%
3381 \fi
3382 \else
3383 \@glsopenfile{\glswrite}{\@glo@type}%
3384 \immediate\write\glswrite{%
3385 \expandafter\the
3386 \csname glo@\@glo@type @filetok\endcsname}%
3387 \immediate\closeout\glswrite

```

```

3388     \fi
3389   }%
3390 }
3391 \else
3392   \let\glswritefiles\relax
3393 \fi

```

The `\glossary` command is redefined so that it takes an optional argument *<type>* to specify the glossary type (use `\glsdefaulttype` glossary by default). This shouldn't be used at user level as `\glslink` sets the correct format. The associated number should be stored in `\theglsentrycounter` before using `\glossary`.

`\glossary`

```

3394 \renewcommand*{\glossary}[1][\glsdefaulttype]{%
3395   \@glossary[#1]%
3396 }

```

Define internal `\@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.)

`\@glossary`

```

3397 \def\@glossary[#1]{\index}

```

This is a convenience command to set `\@glossary`. It is used by `\@makeglossary` and then redefined to do nothing, as it only needs to be done once.

`\@gls@renewglossary`

```

3398 \newcommand{\@gls@renewglossary}{%
3399   \gdef\@glossary[##1]{\@bsphack\begingroup\@wrglossary{##1}}%
3400   \let\@gls@renewglossary\empty
3401 }

```

The `\@wrglossary` command is redefined 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`).

`\@wrglossary`

```

3402 \renewcommand*{\@wrglossary}[2]{%
3403   \ifglssavewrites
3404     \protected@edef\@gls@tmp{\the\csname glo@#1@filetok\endcsname#2}%
3405     \expandafter\global\expandafter\csname glo@#1@filetok\endcsname
3406       \expandafter{\@gls@tmp^^J}%
3407   \else
3408     \ifcsdef{glo@#1@file}%
3409       {%
3410         \expandafter\protected@write\csname glo@#1@file\endcsname{%
3411           \gls@disablepagerefexpansion}{#2}%

```

```

3412 }%
3413 {%
3414 \GlossariesWarning{No file defined for glossary ‘#1’}%
3415 }%
3416 \fi
3417 \endgroup\@esphack
3418 }

```

\@do@wrglossary

```

3419 \newcommand*{\@do@wrglossary}[1]{%
3420 \ifglindexonlyfirst
3421 \ifglused{#1}{\@do@wrglossary{#1}}%
3422 \else
3423 \@do@wrglossary{#1}%
3424 \fi
3425 }

```

@protected@pagefmts List of page formats to be protected against expansion.

```

3426 \newcommand{\gls@protected@pagefmts}{%
3427 \gls@numberpage,\gls@alphpage,\gls@Alphpage,\gls@romanpage,\gls@Romanpage%
3428 }

```

blepagerefexpansion

```

3429 \newcommand*{\gls@disablepagerefexpansion}{%
3430 \@for\@gls@this:=\gls@protected@pagefmts\do
3431 {%
3432 \expandafter\let\@gls@this\relax
3433 }%
3434 }

```

\gls@alphpage

```

3435 \newcommand*{\gls@alphpage}{\@alph\c@page}

```

\gls@Alphpage

```

3436 \newcommand*{\gls@Alphpage}{\@Alph\c@page}

```

\gls@numberpage

```

3437 \newcommand*{\gls@numberpage}{\number\c@page}

```

\gls@romanpage

```

3438 \newcommand*{\gls@romanpage}{\romannumeral\c@page}

```

\gls@Romanpage

```

3439 \newcommand*{\gls@Romanpage}{\@Roman\c@page}

```

\@do@wrglossary Write the glossary entry in the appropriate format. (Need to set \@gls@numberformat and \@gls@counter prior to use.) The argument is the entry’s label.

```

3440 \newcommand*{\@do@wrglossary}[1]{%
3441 \begingroup

```


First a bit of hackery to prevent premature expansion of \c@page. Store original definitions:

```

3442 \let\orgthe\the
3443 \let\orgnumber\number
3444 \let\orgromannumeral\romannumeral
3445 \let\orgalph\@alph
3446 \let\orgAlph\@Alph
3447 \let\orgRoman\@Roman

```

Redefine:

```

3448 \def\the##1{%
3449   \ifx##1\c@page \gls@numberpage\else\orgthe##1\fi}%
3450 \def\number##1{%
3451   \ifx##1\c@page \gls@numberpage\else\orgnumber##1\fi}%
3452 \def\romannumeral##1{%
3453   \ifx##1\c@page \gls@romanpage\else\orgromannumeral##1\fi}%
3454 \def\@Roman##1{%
3455   \ifx##1\c@page \gls@Romanpage\else\orgRoman##1\fi}%
3456 \def\@alph##1{%
3457   \ifx##1\c@page \gls@alphpage\else\orgalph##1\fi}%
3458 \def\@Alph##1{%
3459   \ifx##1\c@page \gls@Alphpage\else\orgAlph##1\fi}%

```

Prevent expansion:

```

3460 \gls@disablepagerefexpansion

```

Now store location in \@glslocref:

```

3461 \protected@xdef\@glslocref{\theHglentrycounter}%
3462 \endgroup

```

Escape any special characters

```

3463 \@gls@checkmkidxchars\@glslocref

```

Check if the hyper-location is the same as the location and set the hyper prefix.

```

3464 \expandafter\ifx\theHglentrycounter\theHglentrycounter
3465   \def\@gls@counterprefix{%
3466   \else
3467     \protected@edef\@glsHlocref{\theHglentrycounter}%
3468     \@gls@checkmkidxchars\@glsHlocref
3469     \edef\@do@gls@getcounterprefix{\noexpand\@gls@getcounterprefix
3470       {\@glslocref}{\@glsHlocref}%
3471     }%
3472     \@do@gls@getcounterprefix
3473   \fi

```

Determine whether to use xindy or makeindex syntax

```

3474 \ifglxindy

```

Need to determine if the formatting information starts with a (or) indicating a range.

```

3475 \expandafter\@glo@check@mkidxrangechar\@glsnumberformat\@nil
3476 \def\@glo@range{}%
3477 \expandafter\if\@glo@prefix(\relax
3478 \def\@glo@range{:open-range}%
3479 \else
3480 \expandafter\if\@glo@prefix)\relax
3481 \def\@glo@range{:close-range}%
3482 \fi
3483 \fi

```

Write to the glossary file using xindy syntax.

```

3484 \glossary[\csname glo@#1@type\endcsname]{%
3485 (indexentry :tkey (\csname glo@#1@index\endcsname)

3486 :locref \string"\@glo@counterprefix}{\@glslocref}\string" %
3487 :attr \string"\@gls@counter\@glo@suffix\string"
3488 \@glo@range
3489 )
3490 }%
3491 \else

```

Convert the format information into the format required for makeindex

```

3492 \@set@glo@numformat{\@glo@numfmt}{\@gls@counter}{\@glsnumberformat}%
3493 {\@glo@counterprefix}%

```

Write to the glossary file using makeindex syntax.

```

3494 \glossary[\csname glo@#1@type\endcsname]{%
3495 \string\glossaryentry{\csname glo@#1@index\endcsname
3496 \@gls@encapchar\@glo@numfmt}{\@glslocref}}%
3497 \fi
3498 }

```

`\ls@getcounterprefix` Get the prefix that needs to be prepended to counter in order to get the hyper counter. (For example, with the standard article class and hyperref, `\theequation` needs to be prefixed with `\section num` to get the equivalent `\theHequation`.) NB this assumes that the prefix ends with a dot, which is the standard. (Otherwise it makes the xindy location classes more complicated.)

```

3499 \newcommand*\@gls@getcounterprefix[2]{%
3500 \edef\@gls@thisloc{#1}\edef\@gls@thisHloc{#2}%
3501 \ifx\@gls@thisloc\@gls@thisHloc
3502 \def\@glo@counterprefix{}%
3503 \else
3504 \def\@gls@get@counterprefix##1.#1##2\end@getprefix{%
3505 \def\@glo@tmp{##2}%
3506 \ifx\@glo@tmp\@empty
3507 \def\@glo@counterprefix{}%
3508 \else
3509 \def\@glo@counterprefix{##1}%
3510 \fi
3511 }%

```

```

3512 \@gls@get@counterprefix#2.#1\end@getprefix
3513 \fi
3514 }

```

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.

```

3515 \newcommand{\@do@seeglossary}[2]{%
3516 \def\@gls@xref{#2}%
3517 \@onelevel@sanitize\@gls@xref
3518 \@gls@checkmkidxchars\@gls@xref
3519 \ifglxsindy
3520 \glossary[\csname glo@#1@type\endcsname]{%
3521 (indexentry
3522 :tkey (\csname glo@#1@index\endcsname)
3523 :xref (\string"\@gls@xref\string")
3524 :attr \string"see\string"
3525 )
3526 }%
3527 \else
3528 \glossary[\csname glo@#1@type\endcsname]{%
3529 \string\glossaryentry{\csname glo@#1@index\endcsname
3530 \@gls@encapchar glsseeformat\@gls@xref}{Z}}%
3531 \fi
3532 }

```

`\@gls@fixbraces` If no optional argument is specified, list needs to be enclosed in a set of braces.

```

3533 \def\@gls@fixbraces#1#2#3\@nil{%
3534 \ifx#2[\relax
3535 \def#1{#2#3}%
3536 \else
3537 \def#1{{#2#3}}%
3538 \fi
3539 }

```

`\glssee` `\glssee{\langle label \rangle}{\langle cross-ref list \rangle}`

```

3540 \newcommand*\glssee[3][\seename]{%
3541 \@do@seeglossary{#2}{#1}{#3}}
3542 \newcommand*\@glssee[3][\seename]{%
3543 \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.

```

3544 \newcommand*\glsseeformat[3][\seename]{\emph{#1} \glsseelist{#2}}

```

`\glsseelist` `\glsseelist{\langle list \rangle}` formats list of entry labels.

```

3545 \newcommand*\glsseelist[1]{%

```

If there is only one item in the list, set the last separator to do nothing.

```
3546 \let\@gls@dolast\relax
```

Don't display separator on the first iteration of the loop

```
3547 \let\@gls@donext\relax
```

Iterate through the labels

```
3548 \@for\@gls@thislabel:=#1\do{%
```

Check if on last iteration of loop

```
3549 \ifx\@xfor@nextelement\@nnil
```

```
3550 \@gls@dolast
```

```
3551 \else
```

```
3552 \@gls@donext
```

```
3553 \fi
```

display the entry for this label

```
3554 \glsseeitem{\@gls@thislabel}%
```

Update separators

```
3555 \let\@gls@dolast\glsseelastsep
```

```
3556 \let\@gls@donext\glsseesep
```

```
3557 }%
```

```
3558 }
```

`\glsseelastsep` Separator to use between penultimate and ultimate entries in a cross-referencing list.

```
3559 \newcommand*{\glsseelastsep}{\space\andname\space}
```

`\glsseesep` Separator to use between entires in a cross-referencing list.

```
3560 \newcommand*{\glsseesep}{, }
```

`\glsseeitem` `\glsseeitem{<label>}` formats individual entry in a cross-referencing list.

```
3561 \newcommand*{\glsseeitem}[1]{\glshyperlink[\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.)

```
3562 \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.

```
3563 \newcommand*{\gls@save@numberlist}[1]{%
3564   \ifglssavenumberlist
3565     \toks@{#1}%
3566   \edef\@do@writeaux@info{%
3567     \noexpand\csgdef{glo@\glscurrententrylabel @numberlist}{\the\toks@}%
3568   }%
3569   \@onelevel@sanitize\@do@writeaux@info
3570   \protected@write\@auxout{}\{\@do@writeaux@info}%
3571 \fi
3572 }
```

`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.)

```
3573 \def\warn@noprintglossary{%
3574   \GlossariesWarningNoLine{No \string\printglossary\space
3575     or \string\printglossaries\space
3576     found.^^JThis document will not have a glossary}%
3577 }
```

`\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.

```
3578 \ifcsundef{printglossary}{}%
3579 {%
```

If `\printglossary` is already defined, issue a warning and undefine it.

```
3580 \GlossariesWarning{Overriding \string\printglossary}%
3581 \undef\printglossary
3582 }
```

`\printglossary` has an optional argument. The default value is to set the glossary
type to the main glossary.

```
3583 \newcommand*{\printglossary}[1][type=\glsdefaulttype]{%
```

If `xindy` is being used, need to find the root language for `makeglossaries` to
pass to `xindy`.

```
3584 \ifglsxindy\findrootlanguage\fi
```

Set up defaults.

```
3585 \def\@glo@type{\glsdefaulttype}%
3586 \def\glossarytitle{\csname @glo@type@\@glo@type @title\endcsname}%

3587 \def\glossarytoctitle{\glossarytitle}%
3588 \let\org@glossarytitle\glossarytitle
3589 \def\@glossarystyle{}%
3590 \def\gls@dotoc@title{\glssettoctitle{\@glo@type}}%
```

Store current value of `\glossaryentrynumbers`. (This may be changed via the optional argument)

```
3591 \let\org@glossaryentrynumbers\glossaryentrynumbers
```

Localise the effects of the optional argument

```
3592 \bgroup
```

Determine settings specified in the optional argument.

```
3593 \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)

```
3594 \ifx\glossarytitle\org@glossarytitle
3595 \else
3596 \expandafter\let\csname @glo@type @\glo@type @title\endcsname
3597 \glossarytitle
3598 \fi
```

Allow a high-level user command to indicate the current glossary

```
3599 \let\currentglossary\@glo@type
```

Enable individual number lists to be suppressed.

```
3600 \let\org@glossaryentrynumbers\glossaryentrynumbers
3601 \let\glsnonextpages\@glsnonextpages
```

Enable individual number list to be activated:

```
3602 \let\glsnextpages\@glsnextpages
```

Enable suppression of description terminators.

```
3603 \let\nopostdesc\@nopostdesc
```

Set up the entry for the TOC

```
3604 \gls@dotoc@title
```

Set the glossary style

```
3605 \@glossarystyle
```

added a way to fetch the current entry label:

```
3606 \let\gls@org@glossaryentryfield\glossaryentryfield
3607 \let\gls@org@glossarysubentryfield\glossarysubentryfield
3608 \renewcommand{\glossaryentryfield}[1]{%
3609 \gdef\glscurrententrylabel{##1}%
3610 \gls@org@glossaryentryfield{##1}%
3611 }%
3612 \renewcommand{\glossarysubentryfield}[2]{%
3613 \gdef\glscurrententrylabel{##2}%
3614 \gls@org@glossarysubentryfield{##1}{##2}%
3615 }%
```

Some macros may end up being expanded into internals in the glossary, so need to make `@` a letter.

```
3616 \makeatletter
```

Input the glossary file, if it exists.

```
3617 \input@{\jobname.\csname @glo@type@\glo@type @in\endcsname}%
```

If the glossary file doesn't exist, do \null. (This ensures that the page is shipped out and all write commands are done.) This might produce an empty page, but at this point the document isn't complete, so it shouldn't matter.

```
3618 \IfFileExists{\jobname.\csname @glo@type@\glo@type @in\endcsname}%
```

```
3619 {}%
```

```
3620 {\null}%
```

If xindy is being used, need to write the language dependent information to the .aux file for makeglossaries.

```
3621 \ifglxindy
```

```
3622 \ifcsundef{@xdy@\glo@type @language}%
```

```
3623 {%
```

```
3624 \edef\@do@auxoutstuff{%
```

```
3625 \noexpand\AtEndDocument{%
```

```
3626 \noexpand\immediate\noexpand\write\@auxout{%
```

```
3627 \string\@xdylanguage{\glo@type}{\@xdy@main@language}}%
```

```
3628 }%
```

```
3629 }%
```

```
3630 }%
```

```
3631 {%
```

```
3632 \edef\@do@auxoutstuff{%
```

```
3633 \noexpand\AtEndDocument{%
```

```
3634 \noexpand\immediate\noexpand\write\@auxout{%
```

```
3635 \string\@xdylanguage{\glo@type}{\csname @xdy@\glo@type
```

```
3636 @language\endcsname}}%
```

```
3637 }%
```

```
3638 }%
```

```
3639 }%
```

```
3640 \@do@auxoutstuff
```

```
3641 \edef\@do@auxoutstuff{%
```

```
3642 \noexpand\AtEndDocument{%
```

```
3643 \noexpand\immediate\noexpand\write\@auxout{%
```

```
3644 \string\@gls@codepage{\glo@type}{\gls@codepage}}%
```

```
3645 }%
```

```
3646 }%
```

```
3647 \@do@auxoutstuff
```

```
3648 \fi
```

```
3649 \egroup
```

Reset \glossaryentrynumbers

```
3650 \global\let\glossaryentrynumbers\org@glossaryentrynumbers
```

Suppress warning about no \printglossary

```
3651 \global\let\warn@noprintglossary\relax
```

```
3652 }
```

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`

```
3653 \newcommand*{\printglossaries}{%
3654   \forallglossaries{\@glo@type}{\printglossary[type=\@glo@type]}%
3655 }
```

The keys that can be used in the optional argument to `\printglossary` are as follows: The `type` key sets the glossary type.

```
3656 \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`.

```
3657 \define@key{printgloss}{title}{%
3658   \def\glossarytitle{#1}%
3659   \let\gls@dotoc@title\relax
3660 }
```

The `toctitle` sets the text used for the relevant entry in the table of contents.

```
3661 \define@key{printgloss}{toctitle}{%
3662   \def\glossarytoctitle{#1}%
3663   \let\gls@dotoc@title\relax
3664 }
```

The `style` key sets the glossary style (but only for the given glossary).

```
3665 \define@key{printgloss}{style}{%
3666   \ifcsundef{@glsstyle@#1}%
3667   {%
3668     \PackageError{glossaries}%
3669     {Glossary style ‘#1’ undefined}{}%
3670   }%
3671   {%
3672     \def\@glossarystyle{\csname @glsstyle@#1\endcsname}%
3673   }%
3674 }
```

The `numberedsection` key determines if this glossary should be in a numbered section.

```
3675 \define@choicekey{printgloss}{numberedsection}[\val\nr]{%
3676 false,nolabel,autolabel}[nolabel]{%
3677 \ifcase\nr\relax
3678   \renewcommand*{\@glossarysecstar}{*}%
3679   \renewcommand*{\@glossaryseclabel}{}%
3680 \or
3681   \renewcommand*{\@glossarysecstar}{}%
3682 }
```



```

3682 \renewcommand*{\@@glossaryseclabel}{}%
3683 \or
3684 \renewcommand*{\@@glossarysecstar}{}%
3685 \renewcommand*{\@@glossaryseclabel}{\label{\glsautoprefix\@glo@type}}%
3686 \fi}

```

The nonnumberlist key determines if this glossary should have a number list.

```

3687 \define@boolkey{printgloss}[gls]{nonnumberlist}[true]{%
3688 \ifglsnonnumberlist
3689 \def\glossaryentrynumbers##1{%
3690 \else
3691 \def\glossaryentrynumbers##1{##1}%
3692 \fi}

```

`\@glsnonextpages` Suppresses the next number list only. Global assignments required as it may not occur in the same level of grouping as the next numberlist. (For example, if `\glsnonextpages` is place in the entry's description and 3 column tabular style glossary is used.) `\org@glossaryentrynumbers` needs to be set at the start of each glossary, in the event that `\glossaryentrynumber` is redefined.

```

3693 \newcommand*{\@glsnonextpages}{%
3694 \gdef\glossaryentrynumbers##1{%
3695 \glsresetentrylist
3696 }%
3697 }

```

`\@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.

```

3698 \newcommand*{\@glsnextpages}{%
3699 \gdef\glossaryentrynumbers##1{%
3700 ##1\glsresetentrylist}}

```

`\glsresetentrylist` Resets `\glossaryentrynumbers`

```

3701 \newcommand*{\glsresetentrylist}{%
3702 \global\let\glossaryentrynumbers\org@glossaryentrynumbers}

```

`\glsnonextpages` Outside of `\printglossary` this does nothing.

```

3703 \newcommand*{\glsnonextpages}{}

```

`\glsnextpages` Outside of `\printglossary` this does nothing.

```

3704 \newcommand*{\glsnextpages}{}

```

`glossaryentry` If the `entrycounter` package option has been used, define a counter to number each level 0 entry.

```

3705 \ifglsentrycounter

```

```

3706 \ifx\@gls@counterwithin\@empty
3707   \newcounter{glossaryentry}
3708 \else
3709   \newcounter{glossaryentry}[\@gls@counterwithin]
3710 \fi
3711 \def\theHglossaryentry{\currentglossary.\theglossaryentry}
3712 \fi

```

glossarysubentry If the subentrycounter package option has been used, define a counter to number each level 1 entry.

```

3713 \ifglssubentrycounter
3714 \ifglentrycounter
3715   \newcounter{glossarysubentry}[glossaryentry]
3716 \else
3717   \newcounter{glossarysubentry}
3718 \fi
3719 \def\theHglossarysubentry{\currentglssubentry.\theglossarysubentry}
3720 \fi

```

resetsubentrycounter Resets the glossarysubentry counter.

```

3721 \ifglssubentrycounter
3722   \newcommand*{\glsresetsubentrycounter}{%
3723     \setcounter{glossarysubentry}{0}%
3724   }
3725 \else
3726   \newcommand*{\glsresetsubentrycounter}{}
3727 \fi

```

resetentrycounter Resets the glossareentry counter.

```

3728 \ifglentrycounter
3729   \newcommand*{\glsresetentrycounter}{%
3730     \setcounter{glossaryentry}{0}%
3731   }
3732 \else
3733   \newcommand*{\glsresetentrycounter}{}
3734 \fi

```

\glsstepentry Advance the glossaryentry counter if in use. The argument is the label associated with the entry.

```

3735 \ifglentrycounter
3736   \newcommand*{\glsstepentry}[1]{%
3737     \refstepcounter{glossaryentry}%
3738     \label{glentry-#1}%
3739   }
3740 \else
3741   \newcommand*{\glsstepentry}[1]{}
3742 \fi

```

`\glsstepsubentry` Advance the glossarysubentry counter if in use. The argument is the label associated with the subentry.

```
3743 \ifglssubentrycounter
3744   \newcommand*{\glsstepsubentry}[1]{%
3745     \def\currentglssubentry{#1}%
3746     \refstepcounter{glossarysubentry}%
3747     \label{glentry-#1}%
3748   }
3749 \else
3750   \newcommand*{\glsstepsubentry}[1]{%
3751 \fi
```

`\glsrefentry` Reference the entry or sub-entry counter if in use, otherwise just do `\gls`.

```
3752 \ifglentrycounter
3753   \newcommand*{\glsrefentry}[1]{\ref{glentry-#1}}
3754 \else
3755   \ifglssubentrycounter
3756     \newcommand*{\glsrefentry}[1]{\ref{glentry-#1}}
3757   \else
3758     \newcommand*{\glsrefentry}[1]{\gls{#1}}
3759   \fi
3760 \fi
```

`glentrycounterlabel` Defines how to display the glossaryentry counter.

```
3761 \ifglentrycounter
3762   \newcommand*{\glentrycounterlabel}{\theglossaryentry.\space}
3763 \else
3764   \newcommand*{\glentrycounterlabel}{}
3765 \fi
```

`glsubentrycounterlabel` Defines how to display the glossarysubentry counter.

```
3766 \ifglssubentrycounter
3767   \newcommand*{\glsubentrycounterlabel}{\theglossarysubentry)\space}
3768 \else
3769   \newcommand*{\glsubentrycounterlabel}{}
3770 \fi
```

`\glentryitem` Step and display glossaryentry counter, if appropriate.

```
3771 \ifglentrycounter
3772   \newcommand*{\glentryitem}[1]{%
3773     \glsstepentry{#1}\glentrycounterlabel
3774   }
3775 \else
3776   \newcommand*{\glentryitem}[1]{\glsresetsubentrycounter}
3777 \fi
```

`\glssubentryitem` Step and display glossarysubentry counter, if appropriate.

```
3778 \ifglssubentrycounter
```

```

3779 \newcommand*{\glssubentryitem}[1]{%
3780   \glssubentry{#1}\glssubentrycounterlabel
3781 }
3782 \else
3783 \newcommand*{\glssubentryitem}[1]{%
3784 \fi

```

theglossary If the theglossary environment has already been defined, a warning will be issued. This environment should be redefined by glossary styles.

```

3785 \ifcsundef{theglossary}%
3786 {%
3787   \newenvironment{theglossary}{}{}%
3788 }%
3789 {%
3790   \GlossariesWarning{overriding ‘theglossary’ environment}%
3791   \renewenvironment{theglossary}{}{}%
3792 }

```

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

```

3793 \newcommand*{\glossaryheader}{}

```

\glstarget `\glstarget{<label>}{<name>}`

Provide user interface to `\@glstarget` to make it easier to modify the glossary style in the document.

```

3794 \newcommand*{\glstarget}[2]{\@glstarget{\glolinkprefix#1}{#2}}

```

\glossaryentryfield `\glossaryentryfield{<label>}{<name>}{<description>}{<symbol>}{<page-list>}`

This command governs how each entry row should be formatted in the glossary. Glossary styles need to redefine this command. Most of the predefined styles ignore `<symbol>`.

```

3795 \newcommand*{\glossaryentryfield}[5]{%
3796 \noindent\textbf{\glstarget{#1}{#2}} #4 #3. #5\par}

```

\glossaryentryfield `\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.)

```

3797 \newcommand*{\glossarysubentryfield}[6]{%
3798 \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`

```

3799 \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`

```

3800 \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.

`\glsgrouptitle`

```

3801 \newcommand*{\glsgrouptitle}[1]{%
3802   \ifcsundef{#1groupname}{#1}{\csname #1groupname\endcsname}%
3803 }

```

`\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`

```
3804 \newcommand*{\glsgetgrouplabel}[1]{%
3805 \ifthenelse{\equal{#1}{\glssymbolsgroupname}}{\glssymbols}{%
3806 \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`

```
3807 \newcommand*{\setentrycounter}[2][ ]{%
3808   \def\@glo@counterprefix{#1}%
3809   \ifx\@glo@counterprefix\@empty
3810     \def\@glo@counterprefix{.}%
3811   \else
3812     \def\@glo@counterprefix{.#1.}%
3813   \fi
3814   \def\glsentrycounter{#2}%
3815 }
```

The current glossary style can be set using `\glossarystyle{<style>}`.

`\glossarystyle`

```
3816 \newcommand*{\glossarystyle}[1]{%
3817   \ifcsundef{@glsstyle@#1}%
3818   {%
3819     \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
3820   }%
3821   {%
3822     \csname @glsstyle@#1\endcsname
3823   }%
3824 }
```

`\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.

```
3825 \newcommand{\newglossarystyle}[2]{%
```

```

3826 \ifcsundef{@glsstyle@#1}%
3827 {%
3828   \expandafter\def\csname @glsstyle@#1\endcsname{#2}%
3829 }%
3830 {%
3831   \PackageError{glossaries}{Glossary style ‘#1’ is already defined}{}%
3832 }%
3833 }

```

`\renewglossarystyle` Code for this macro supplied by Marco Daniel.

```

3834 \newcommand{\renewglossarystyle}[2]{%
3835   \ifcsundef{@glsstyle@#1}%
3836   {%
3837     \PackageError{glossaries}{Glossary style ‘#1’ isn’t already defined}{}%
3838   }%
3839   {%
3840     \csdef{@glsstyle@#1}{#2}%
3841   }%
3842 }

```

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`

```

3843 \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`

```

3844 \ifcsundef{hyperlink}%
3845 {%

```

```

3846 \def\glshypernumber#1{#1}%
3847 }%
3848 {%
3849 \def\glshypernumber#1{\@glshypernumber#1\nohyperpage{}}\@nil}
3850 }

```

`\@glshypernumber` This code was provided by Heiko Oberdiek to allow material to be attached to the location.

```

3851 \def\@glshypernumber#1\nohyperpage#2#3\@nil{%
3852 \ifx\#1\%
3853 \else
3854 \@delimR#1\delimR\delimR\%
3855 \fi
3856 \ifx\#2\%
3857 \else
3858 #2%
3859 \fi
3860 \ifx\#3\%
3861 \else
3862 \@glshypernumber#3\@nil
3863 \fi
3864 }

```

`\@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`

```

3865 \def\@delimR#1\delimR #2\delimR #3\%
3866 \ifx\#2\%
3867 \@delimN{#1}%
3868 \else
3869 \@gls@numberlink{#1}\delimR\@gls@numberlink{#2}%
3870 \fi}

```

`\@delimN` displays a list of individual numbers, instead of a range:

`\@delimN`

```

3871 \def\@delimN#1{\@delimN#1\delimN \delimN\}
3872 \def\@delimN#1\delimN #2\delimN#3\%
3873 \ifx\#3\%
3874 \@gls@numberlink{#1}%
3875 \else
3876 \@gls@numberlink{#1}\delimN\@gls@numberlink{#2}%
3877 \fi
3878 }

```

The following code is modified from hyperref's `\HyInd@pagelink` where the name of the counter being used is given by `\@gls@counter`.

```

3879 \def\@gls@numberlink#1{%

```



```

3880 \begingroup
3881 \toks@={}%
3882 \@gls@removespaces#1 \@nil
3883 \endgroup}

3884 \def\@gls@removespaces#1 #2\@nil{%
3885 \toks@=\expandafter{\the\toks@#1}%
3886 \ifx\#2\%
3887 \edef\x{\the\toks@}%
3888 \ifx\x\empty
3889 \else

3890 \hyperlink{\glsentrycounter\@glo@counterprefix\the\toks@}%
3891 {\the\toks@}%
3892 \fi
3893 \else
3894 \@gls@ReturnAfterFi{%
3895 \@gls@removespaces#2\@nil
3896 }%
3897 \fi
3898 }
3899 \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
3900 \newcommand*\hyperrm[1]{\textrm{\glshypernumber{#1}}}

\hypersf
3901 \newcommand*\hypersf[1]{\textsf{\glshypernumber{#1}}}

\hypertt
3902 \newcommand*\hypertt[1]{\texttt{\glshypernumber{#1}}}

\hyperbf
3903 \newcommand*\hyperbf[1]{\textbf{\glshypernumber{#1}}}

\hypermd
3904 \newcommand*\hypermd[1]{\textmd{\glshypernumber{#1}}}

\hyperit
3905 \newcommand*\hyperit[1]{\textit{\glshypernumber{#1}}}

\hypersl
3906 \newcommand*\hypersl[1]{\textsl{\glshypernumber{#1}}}

\hyperup
3907 \newcommand*\hyperup[1]{\textup{\glshypernumber{#1}}}

```

`\hypersc`

```
3908 \newcommand*{\hypersc}[1]{\textsc{\glshypernumber{#1}}}
```

`\hyperemph`

```
3909 \newcommand*{\hyperemph}[1]{\emph{\glshypernumber{#1}}}
```

1.16 Acronyms

If the acronym package option is used, a new glossary called acronym is created

```
3910 \ifglsacronym
```

```
3911 \newglossary[alg]{acronym}{acr}{acn}{\acronymname}
```

and `\acronymtype` is set to the name of this new glossary.

```
3912 \renewcommand*{\acronymtype}{acronym}
```

```
3913 \fi
```

```
\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.

```
3914 \newcommand{\oldacronym}[4][\gls@label]{%
```

```
3915 \def\gls@label{#2}%
```

```
3916 \newacronym[#4]{#1}{#2}{#3}%
```

```
3917 \ifcsundef{xspace}%
```

```
3918 {%
```

```
3919 \expandafter\edef\csname#1\endcsname{%
```

```
3920 \noexpand\@ifstar{\noexpand\Gls{#1}}{\noexpand\gls{#1}}%
```

```
3921 }%
```

```
3922 }%
```

```
3923 {%
```

```
3924 \expandafter\edef\csname#1\endcsname{%
```

```
3925 \noexpand\@ifstar{\noexpand\Gls{#1}\noexpand\xspace}{%
```

```
3926 \noexpand\gls{#1}\noexpand\xspace}%
```

```
3927 }%
```

```
3928 }%
```

```
3929 }
```

`\newacronym[⟨key-val list⟩]{⟨label⟩}{⟨abbrev⟩}{⟨long⟩}`

This is a quick way of defining acronyms, all it does is call `\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`

```
3930 \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.

```
3931 \newcommand*{\acrpluralsuffix}{\glspluralsuffix}
```

The following are defined for compatibility with version 2.07 and earlier.

`\glsshortkey`

```
3932 \newcommand*{\glsshortkey}{short}
```

`\glsshortpluralkey`

```
3933 \newcommand*{\glsshortpluralkey}{shortplural}
```

`\glslongkey`

```
3934 \newcommand*{\glslongkey}{long}
```

`\glslongpluralkey`

```
3935 \newcommand*{\glslongpluralkey}{longplural}
```

`\acrfull` Full form of the acronym.

```
3936 \newrobustcmd*{\acrfull}{%
```

```
3937 \ifstar\s@acrfull\ns@acrfull
```

```
3938 }
```

```
3939 \newcommand*\s@acrfull[2] [] {%
```

```
3940 \new@ifnextchar[{\@acrfull{hyper=false,#1}{#2}}%
```

```
3941 {\@acrfull{hyper=false,#1}{#2} [] }%
```

```

3942 }
3943 \newcommand*\ns@acrfull[2][{}]{%
3944   \new@ifnextchar[{\@acrfull{#1}{#2}}{%
3945     {\@acrfull{#1}{#2}[]}%
3946 }

```

Low-level macro:

```

3947 \def\@acrfull#1#2[#3]{%
3948   \acrlinkfullformat{\@acrlong}{\@acrshort}{#1}{#2}{#3}%
3949 }

```

`\acrlinkfullformat` Format for full links like `\acrfull`. Syntax: `\acrlinkfullformat{<long cs>}{<short cs>}{<options>}{<label>}{<insert>}`

```

3950 \newcommand{\acrlinkfullformat}[5]{%
3951   \acrfullformat{#1{#3}{#4}[#5]}{#2{#3}{#4}[]}%
3952 }

```

`\acrfullformat` Default full form is *<long>* (*<short>*).

```

3953 \newcommand{\acrfullformat}[2]{#1\space(#2)}

```

Default format for full acronym

`\Acrfull`

```

3954 \newrobustcmd*\Acrfull}{%
3955   \@ifstar\s@Acrfull\ns@Acrfull
3956 }

3957 \newcommand*\s@Acrfull[2][{}]{%
3958   \new@ifnextchar[{\@Acrfull{hyper=false,#1}{#2}}{%
3959     {\@Acrfull{hyper=false,#1}{#2}[]}%
3960 }
3961 \newcommand*\ns@Acrfull[2][{}]{%
3962   \new@ifnextchar[{\@Acrfull{#1}{#2}}{%
3963     {\@Acrfull{#1}{#2}[]}%
3964 }

```

Low-level macro:

```

3965 \def\@Acrfull#1#2[#3]{%
3966   \acrlinkfullformat{\@Acrlong}{\@acrshort}{#1}{#2}{#3}%
3967 }

```

`\ACRfull`

```

3968 \newrobustcmd*\ACRfull}{%
3969   \@ifstar\s@ACRfull\ns@ACRfull
3970 }

3971 \newcommand*\s@ACRfull[2][{}]{%
3972   \new@ifnextchar[{\@ACRfull{hyper=false,#1}{#2}}{%
3973     {\@ACRfull{hyper=false,#1}{#2}[]}%
3974 }

```

```

3975 \newcommand*\ns@ACRfull[2][]{%
3976   \new@ifnextchar[{\@ACRfull{#1}{#2}}{%
3977     {\@ACRfull{#1}{#2}[]}%
3978 }

```

Low-level macro:

```

3979 \def\@ACRfull#1#2[#3]{%
3980   \acrlinkfullformat{\@ACRlong}{\@ACRshort}{#1}{#2}{#3}%
3981 }

```

Plural:

\acrfullpl

```

3982 \newrobustcmd*\acrfullpl{%
3983   \@ifstar\s@acrfullpl\ns@acrfullpl
3984 }

3985 \newcommand*\s@acrfullpl[2][]{%
3986   \new@ifnextchar[{\@acrfullpl{hyper=false,#1}{#2}}{%
3987     {\@acrfullpl{hyper=false,#1}{#2}[]}%
3988 }
3989 \newcommand*\ns@acrfullpl[2][]{%
3990   \new@ifnextchar[{\@acrfullpl{#1}{#2}}{%
3991     {\@acrfullpl{#1}{#2}[]}%
3992 }

```

Low-level macro:

```

3993 \def\@acrfullpl#1#2[#3]{%
3994   \acrlinkfullformat{\@acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%
3995 }

```

\Acrfullpl

```

3996 \newrobustcmd*\Acrfullpl{%
3997   \@ifstar\s@Acrfullpl\ns@Acrfullpl
3998 }

3999 \newcommand*\s@Acrfullpl[2][]{%
4000   \new@ifnextchar[{\@Acrfullpl{hyper=false,#1}{#2}}{%
4001     {\@Acrfullpl{hyper=false,#1}{#2}[]}%
4002 }
4003 \newcommand*\ns@Acrfullpl[2][]{%
4004   \new@ifnextchar[{\@Acrfullpl{#1}{#2}}{%
4005     {\@Acrfullpl{#1}{#2}[]}%
4006 }

```

Low-level macro:

```

4007 \def\@Acrfullpl#1#2[#3]{%
4008   \acrlinkfullformat{\@Acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%
4009 }

```

`\ACRfullpl`

```
4010 \newrobustcmd*{\ACRfullpl}{%
4011   \@ifstar\s@ACRfullpl\ns@ACRfullpl
4012 }

4013 \newcommand*\s@ACRfullpl[2][]{%
4014   \new@ifnextchar[{\@ACRfullpl{hyper=false,#1}{#2}}%
4015     {\@ACRfullpl{hyper=false,#1}{#2}[]}%
4016 }
4017 \newcommand*\ns@ACRfullpl[2][]{%
4018   \new@ifnextchar[{\@ACRfullpl{#1}{#2}}%
4019     {\@ACRfullpl{#1}{#2}[]}%
4020 }
```

Low-level macro:

```
4021 \def\@ACRfullpl#1#2[#3]{%
4022   \acrlinkfullformat{\@ACRlongpl}{\@ACRshortpl}{#1}{#2}{#3}%
4023 }
```

1.17 Predefined acronym styles

`\acronymfont` This is only used with the additional acronym styles:

```
4024 \newcommand{\acronymfont}[1]{#1}
```

`\firstacronymfont` This is only used with the additional acronym styles:

```
4025 \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.

```
4026 \newcommand*\acrnameformat[2]{\acronymfont{#1}}
```

Define some tokens used by `\newacronym`:

`\glskeylisttok`

```
4027 \newtoks\glskeylisttok
```

`\glslabeltok`

```
4028 \newtoks\glslabeltok
```

`\glsshorttok`

```
4029 \newtoks\glsshorttok
```

`\glslongtok`

```
4030 \newtoks\glslongtok
```

`\newacronymhook` Provide a hook for `\newacronym`:

```
4031 \newcommand*\newacronymhook{}
```

AcronymDisplayStyle Sets the default acronym display style for given glossary.

```
4032 \newcommand*{\SetDefaultAcronymDisplayStyle}[1]{%
4033   \defglsdisplay[#1]{##1##4}%
4034   \defglsdisplayfirst[#1]{##1##4}%
4035 }
```

DefaultNewAcronymDef Sets up the acronym definition for the default style. The information is provided by the tokens `\glslabeltok`, `\glsshorttok`, `\glslongtok` and `\glskeylisttok`.

```
4036 \newcommand*{\DefaultNewAcronymDef}{%
4037   \edef\@do@newglossaryentry{%
4038     \noexpand\newglossaryentry{\the\glslabeltok}%
4039     {%
4040       type=\acronymtype,%
4041       name={\the\glsshorttok},%
4042       sort={\the\glsshorttok},%
4043       text={\the\glsshorttok},%
4044       first={\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
4045       plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
4046       firstplural={\acrfullformat{\noexpand\@glo@longpl}%
4047                     {\noexpand\@glo@shortpl}},%
4048       short={\the\glsshorttok},%
4049       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
4050       long={\the\glslongtok},%
4051       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
4052       description={\the\glslongtok},%
4053       descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
```

Remaining options specified by the user:

```
4054     \the\glskeylisttok
4055   }%
4056 }%
4057 \@do@newglossaryentry
4058 }
```

DefaultAcronymStyle Set up the default acronym style:

```
4059 \newcommand*{\SetDefaultAcronymStyle}{%
```

Set the display style:

```
4060   \@for\@gls@type:=\@glsacronymlists\do{%
4061     \SetDefaultAcronymDisplayStyle{\@gls@type}%
4062   }%
```

Set up the definition of `\newacronym`:

```
4063   \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).

```
4064     \ifx\@glsacronymlists\@empty
```

```

4065 \def\@glo@type{\acronymtype}%
4066 \setkeys{glossentry}{##1}%
4067 \DeclareAcronymList{\@glo@type}%
4068 \SetDefaultAcronymDisplayStyle{\@glo@type}%
4069 \fi
4070 \glskeylisttok{##1}%
4071 \glslabeltok{##2}%
4072 \glsshorttok{##3}%
4073 \gslongtok{##4}%
4074 \newacronymhook
4075 \DefaultNewAcronymDef
4076 }%
4077 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
4078 }

```

`\acrfootnote` Used by the footnote acronym styles.

```

4079 \newcommand*{\acrfootnote}[3]{\acrlinkfootnote{#1}{#2}{#3}}

```

`\acrlinkfootnote`

```

4080 \newcommand*{\acrlinkfootnote}[3]{%
4081 \footnote{\glslink{#1}{#2}{#3}}%
4082 }

```

`\acrnoflinkfootnote`

```

4083 \newcommand*{\acrnoflinkfootnote}[3]{%
4084 \footnote{#3}%
4085 }

```

`AcronymDisplayStyle` Sets the acronym display style for given glossary for the description and footnote combination.

```

4086 \newcommand*{\SetDescriptionFootnoteAcronymDisplayStyle}[1]{%
4087 \def\glsdisplayfirst[#1]{%
4088 \firstacronymfont{##1}##4%
4089 \expandafter\protect\expandafter\acrfootnote\expandafter
4090 {\@gls@link@opts}{\@gls@link@label}{##3}%
4091 }%
4092 \def\glsdisplay[#1]{\acronymfont{##1}##4}%
4093 }

```

`otnoteNewAcronymDef`

```

4094 \newcommand*{\DescriptionFootnoteNewAcronymDef}{%
4095 \edef\@do@newglossaryentry{%
4096 \noexpand\newglossaryentry{\the\glslabeltok}%
4097 {%
4098 type=\acronymtype,%
4099 name={\noexpand\acronymfont{\the\glsshorttok}},%
4100 sort={\the\glsshorttok},%
4101 text={\the\glsshorttok},%

```



```

4102     plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
4103     short={\the\glsshorttok},%
4104     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
4105     long={\the\glslongtok},%
4106     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
4107     symbol={\the\glslongtok},%
4108     symbolplural={\the\glslongtok\noexpand\acrpluralsuffix},%
4109     \the\glskeylisttok
4110   }%
4111 }%
4112 \do@newglossaryentry
4113 }

```

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.

```

4114 \newcommand*\SetDescriptionFootnoteAcronymStyle{%
4115   \renewcommand{\newacronym}[4][\]{%
4116     \ifx\@glsacronymlists\@empty
4117       \def\@glo@type{\acronymtype}%
4118       \setkeys{glossentry}{##1}%
4119       \DeclareAcronymList{\@glo@type}%
4120       \SetDescriptionFootnoteAcronymDisplayStyle{\@glo@type}%
4121     \fi
4122     \glskeylisttok{##1}%
4123     \glslabeltok{##2}%
4124     \glsshorttok{##3}%
4125     \glslongtok{##4}%
4126     \newacronymhook
4127     \DescriptionFootnoteNewAcronymDef
4128   }%

```

If footnote package option is specified, set the first use to append the long form (stored in symbol) as a footnote.

```

4129   \@for\@gls@type:=\@glsacronymlists\do{%
4130     \SetDescriptionFootnoteAcronymDisplayStyle{\@gls@type}%
4131   }%

```

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.

```

4132   \ifglsacrsmallcaps
4133     \renewcommand*\acronymfont{[1]{\textsc{##1}}}%
4134     \renewcommand*\acrpluralsuffix{%
4135       \textup{\glspluralsuffix}}%
4136   \else
4137     \ifglsacrsmaller
4138       \renewcommand*\acronymfont{[1]{\textsmaller{##1}}}%

```

```

4139   \fi
4140   \fi

  Check for package option clash
4141   \ifglssacrdua
4142     \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’
4143       can’t both be set}{}%
4144   \fi
4145 }%

```

AcronymDisplayStyle Sets the acronym display style for given glossary with description and dua combination.

```

4146 \newcommand*{\SetDescriptionDUAAcronymDisplayStyle}[1]{%
4147   \defglssdisplay[#1]{##1##4}%
4148   \defglssdisplayfirst[#1]{##1##4}%
4149 }

```

ionDUANewAcronymDef

```

4150 \newcommand*{\DescriptionDUANewAcronymDef}{%
4151   \edef\@do@newglossaryentry{%
4152     \noexpand\newglossaryentry{\the\glslabeltok}%
4153     {%
4154       type=\acronymtype,%
4155       name={\the\glslongtok},%
4156       sort={\the\glslongtok},%
4157       text={\the\glslongtok},%
4158       plural={\the\glslongtok\noexpand\acrpluralsuffix},%
4159       short={\the\glsshorttok},%
4160       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
4161       long={\the\glslongtok},%
4162       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
4163       symbol={\the\glsshorttok},%
4164       symbolplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
4165       \the\glskeylisttok
4166     }%
4167   }%
4168   \@do@newglossaryentry
4169 }

```

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.

```

4170 \newcommand*{\SetDescriptionDUAAcronymStyle}{%
4171   \ifglssacrsmallcaps
4172     \PackageError{glossaries}{Option clash: ‘smallcaps’ and ‘dua’
4173       can’t both be set}{}%
4174   \else
4175     \ifglssacrsmaller
4176       \PackageError{glossaries}{Option clash: ‘smaller’ and ‘dua’

```

```

4177     can't both be set}{}%
4178   \fi
4179 \fi
4180 \renewcommand{\newacronym}[4][]{%
4181   \ifx\@glsacronymlists\@empty
4182     \def\@glo@type{\acronymtype}%
4183     \setkeys{glossentry}{##1}%
4184     \DeclareAcronymList{\@glo@type}%
4185     \SetDescriptionDUAAcronymDisplayStyle{\@glo@type}%
4186   \fi
4187   \glskeylisttok{##1}%
4188   \glslabeltok{##2}%
4189   \glsshorttok{##3}%
4190   \glslongtok{##4}%
4191   \newacronymhook
4192   \DescriptionDUANewAcronymDef
4193 }%

Set display.
4194 \@for\@gls@type:=\@glsacronymlists\do{%
4195   \SetDescriptionDUAAcronymDisplayStyle{\@gls@type}%
4196 }%
4197 }%

```

AcronymDisplayStyle Sets the acronym display style for given glossary using the description setting (but not footnote or dua).

```

4198 \newcommand*\SetDescriptionAcronymDisplayStyle}[1]{%
4199   \def\glsdisplayfirst[#1]{%
4200     ##1##4 (\firstacronymfont{##3})}%
4201   \def\glsdisplay[#1]{\acronymfont{##1}##4}%
4202 }

```

DescriptionNewAcronymDef

```

4203 \newcommand*\DescriptionNewAcronymDef{%
4204   \edef\@do@newglossaryentry{%
4205     \noexpand\newglossaryentry{\the\glslabeltok}%
4206     {%
4207       type=\acronymtype,%
4208       name={\noexpand
4209         \acrnameformat{\the\glsshorttok}{\the\glslongtok}},%
4210       sort={\the\glsshorttok},%
4211       first={\the\glslongtok},%
4212       firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
4213       text={\the\glsshorttok},%
4214       plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
4215       short={\the\glsshorttok},%
4216       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
4217       long={\the\glslongtok},%
4218       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%

```

```

4219     symbol={\noexpand\@glo@text},%
4220     symbolplural={\noexpand\@glo@plural},%
4221     \the\glskeylisttok}%
4222 }%
4223 \@do@newglossaryentry
4224 }

```

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.

```

4225 \newcommand*\SetDescriptionAcronymStyle{%
4226   \renewcommand{\newacronym}[4][]{%
4227     \ifx\@glsacronymlists\@empty
4228       \def\@glo@type{\acronymtype}%
4229       \setkeys{glossentry}{##1}%
4230       \DeclareAcronymList{\@glo@type}%
4231       \SetDescriptionAcronymDisplayStyle{\@glo@type}%
4232     \fi
4233     \glskeylisttok{##1}%
4234     \glslabeltok{##2}%
4235     \glsshorttok{##3}%
4236     \gslongtok{##4}%
4237     \newacronymhook
4238     \DescriptionNewAcronymDef
4239   }%

```

Set display.

```

4240   \@for\@gls@type:=\@glsacronymlists\do{%
4241     \SetDescriptionAcronymDisplayStyle{\@gls@type}%
4242   }%

```

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.

```

4243   \ifglsacrsmallcaps
4244     \renewcommand{\acronymfont}[1]{\textsc{##1}}
4245     \renewcommand*\{acrpluralsuffix}{%
4246       \textup{\glspluralsuffix}}%
4247   \else
4248     \ifglsacrsmaller
4249       \renewcommand*\{acronymfont}[1]{\textsmaller{##1}}%
4250     \fi
4251   \fi
4252 }%

```

AcronymDisplayStyle Sets the acronym display style for given glossary with footnote setting (but not description or dua).

```

4253 \newcommand*\SetFootnoteAcronymDisplayStyle[1]{%

```

```

4254 \defglsdisplayfirst[#1]{%
4255   \firstacronymfont{##1}##4%
4256   \expandafter\protect\expandafter\acrfootnote\expandafter
4257   {\@gls@link@opts}{\@gls@link@label}{##2}%
4258 }%
4259 \defglsdisplay[#1]{\acronymfont{##1}##4}%
4260 }

```

otnoteNewAcronymDef

```

4261 \newcommand*{\FootnoteNewAcronymDef}{%
4262   \edef\@do@newglossaryentry{%
4263     \noexpand\newglossaryentry{\the\glslabeltok}%
4264     {%
4265       type=\acronymtype,%
4266       name={\noexpand\acronymfont{\the\glsshorttok}},%
4267       sort={\the\glsshorttok},%
4268       text={\the\glsshorttok},%
4269       plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
4270       short={\the\glsshorttok},%
4271       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
4272       long={\the\glslongtok},%
4273       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
4274       description={\the\glslongtok},%
4275       descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
4276       \the\glskeylisttok
4277     }%
4278   }%
4279   \@do@newglossaryentry
4280 }

```

otnoteAcronymStyle 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.

```

4281 \newcommand*{\SetFootnoteAcronymStyle}{%
4282   \renewcommand{\newacronym}[4][\]{%
4283     \ifx\@glsacronymlists\@empty
4284       \def\@glo@type{\acronymtype}%
4285       \setkeys{glossentry}{##1}%
4286       \DeclareAcronymList{\@glo@type}%
4287       \SetFootnoteAcronymDisplayStyle{\@glo@type}%
4288     \fi
4289     \glskeylisttok{##1}%
4290     \glslabeltok{##2}%
4291     \glsshorttok{##3}%
4292     \glslongtok{##4}%
4293     \newacronymhook
4294     \FootnoteNewAcronymDef
4295   }%

```

Set display

```
4296 \for\@gls@type:=\@glsacronymlists\do{%
4297   \SetFootnoteAcronymDisplayStyle{\@gls@type}%
4298 }%
```

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.

```
4299 \ifglsacrsmallcaps
4300   \renewcommand*\acronymfont}[1]{\textsc{##1}}%
4301   \renewcommand*\acrpluralsuffix}{%
4302     \textup{\glspluralsuffix}}%
4303 \else
4304   \ifglsacrsmaller
4305     \renewcommand*\acronymfont}[1]{\textsmaller{##1}}%
4306   \fi
4307 \fi
```

Check for option clash

```
4308 \ifglsacrdua
4309   \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’
4310     can’t both be set}{}%
4311 \fi
4312 }%
```

AcronymDisplayStyle Sets the acronym display style for given glossary where neither footnote nor description is required, but smallcaps or smaller specified.

```
4313 \newcommand*\SetSmallAcronymDisplayStyle}[1]{%
4314   \defglsdisplayfirst[#1]{##1##4 (\firstacronymfont{##3})}
4315   \defglsdisplay[#1]{\acronymfont{##1}##4}%
4316 }
```

\SmallNewAcronymDef

```
4317 \newcommand*\SmallNewAcronymDef}{%
4318   \edef\@do@newglossaryentry{%
4319     \noexpand\newglossaryentry{\the\glslabeltok}%
4320     {%
4321       type=\acronymtype,%
4322       name={\noexpand\acronymfont{\the\glsshorttok}},%
4323       sort={\the\glsshorttok},%
4324       text={\noexpand\@glo@symbol},%
```

Default to the short plural.

```
4325     plural={\noexpand\@glo@shortpl},%
4326     first={\the\glslongtok},%
```

Default to the long plural.

```
4327     firstplural={\noexpand\@glo@longpl},%
4328     short={\the\glsshorttok},%
4329     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
```

```

4330     long={\the\glslongtok},%
4331     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
4332     description={\noexpand\@glo@first},%
4333     descriptionplural={\noexpand\@glo@firstplural},%
4334     symbol={\the\glsshorttok},%

```

Default to the short plural.

```

4335     symbolplural={\noexpand\@glo@shortpl},%
4336     \the\glskeylisttok
4337   }%
4338 }%
4339 \@do@newglossaryentry
4340 }

```

`\setSmallAcronymStyle` Neither footnote nor description required, but smallcaps or smaller specified.
Use the symbol key to store the short form and first to store the long form.

```

4341 \newcommand*\setSmallAcronymStyle{%
4342   \renewcommand{\newacronym}[4][\]{%
4343     \ifx\@glsacronymlists\@empty
4344       \def\@glo@type{\acronymtype}%
4345       \setkeys{glossentry}{##1}%
4346       \DeclareAcronymList{\@glo@type}%
4347       \setSmallAcronymDisplayStyle{\@glo@type}%
4348     \fi
4349     \glskeylisttok{##1}%
4350     \glslabeltok{##2}%
4351     \glsshorttok{##3}%
4352     \glslongtok{##4}%
4353     \newacronymhook
4354     \SmallNewAcronymDef
4355   }%

```

Change the display since first only contains long form.

```

4356   \@for\@gls@type:=\@glsacronymlists\do{%
4357     \setSmallAcronymDisplayStyle{\@gls@type}%
4358   }%

```

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.

```

4359   \ifglsacrsmallcaps
4360     \renewcommand*\acronymfont[1]{\textsc{##1}}
4361     \renewcommand*\acrpluralsuffix{%
4362       \textup{\glspluralsuffix}}%
4363   \else
4364     \renewcommand*\acronymfont[1]{\textsmaller{##1}}
4365   \fi

```

check for option clash

```

4366   \ifglsacrdua

```

```

4367 \ifglsmallcaps
4368 \PackageError{glossaries}{Option clash: ‘smallcaps’ and ‘dua’
4369 can’t both be set}{}%
4370 \else
4371 \PackageError{glossaries}{Option clash: ‘smaller’ and ‘dua’
4372 can’t both be set}{}%
4373 \fi
4374 \fi
4375 }%

```

\SetDUADisplayStyle Sets the acronym display style for given glossary with dua setting.

```

4376 \newcommand*\SetDUADisplayStyle[1]{%
4377 \defglstdisplay[#1]{##1##4}%
4378 \defglstdisplayfirst[#1]{##1##4}%
4379 }

```

\DUANewAcronymDef

```

4380 \newcommand*\DUANewAcronymDef{%
4381 \edef\@do@newglossaryentry{%
4382 \noexpand\newglossaryentry{\the\glslabeltok}%
4383 {%
4384 type=\acronymtype,%
4385 name={\the\glsshorttok},%
4386 text={\the\glslongtok},%
4387 plural={\the\glslongtok\noexpand\acrpluralsuffix},%
4388 short={\the\glsshorttok},%
4389 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
4390 long={\the\glslongtok},%
4391 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
4392 description={\the\glslongtok},%
4393 symbol={\the\glsshorttok},%
4394 symbolplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
4395 \the\glskeylisttok
4396 }%
4397 }%
4398 \@do@newglossaryentry
4399 }

```

\SetDUASStyle Always expand acronyms.

```

4400 \newcommand*\SetDUASStyle{%
4401 \renewcommand{\newacronym}[4][[]]{%
4402 \ifx\@glsacronymlists\@empty
4403 \def\@glo@type{\acronymtype}%
4404 \setkeys{glossentry}{##1}%
4405 \DeclareAcronymList{\@glo@type}%
4406 \SetDUADisplayStyle{\@glo@type}%
4407 \fi
4408 \glskeylisttok{##1}%
4409 \glslabeltok{##2}%

```



```

4410 \glsshorttok{##3}%
4411 \glslongtok{##4}%
4412 \newacronymhook
4413 \DUANewAcronymDef
4414 }%

Set the display
4415 \@for\@gls@type:=\@glsacronymlists\do{%
4416 \SetDUADisplayStyle{\@gls@type}%
4417 }%
4418 }

```

\SetAcronymStyle

```

4419 \newcommand*\SetAcronymStyle{%
4420 \SetDefaultAcronymStyle
4421 \ifglsacrdescription
4422 \ifglsacrfootnote
4423 \SetDescriptionFootnoteAcronymStyle
4424 \else
4425 \ifglsacrdua
4426 \SetDescriptionDUAAcronymStyle
4427 \else
4428 \SetDescriptionAcronymStyle
4429 \fi
4430 \fi
4431 \else
4432 \ifglsacrfootnote
4433 \SetFootnoteAcronymStyle
4434 \else
4435 \ifthenelse{\boolean{glsacrsmallsac}}{\OR
4436 \boolean{glsacrsmallsac}}{%
4437 }%
4438 \SetSmallAcronymStyle
4439 }%
4440 }%
4441 \ifglsacrdua
4442 \SetDUASyle
4443 \fi
4444 }%
4445 \fi
4446 \fi
4447 }

```

Set the acronym style according to the package options

```
4448 \SetAcronymStyle
```

Allow user to define their own custom acronyms. (For compatibility with versions before v3.0, the short form is stored in the user1 key, the plural short form is stored in the user2 key, the long form is stored in the user3 key and

the plural long form is stored in the user4 key.) Defaults to displaying only the acronym with the long form as the description.

`\SetCustomDisplayStyle` Sets the acronym display style.

```
4449 \newcommand*\SetCustomDisplayStyle[1]{%
4450   \def\glsdisplay[#1]{##1##4}%
4451   \def\glsdisplayfirst[#1]{##1##4}%
4452 }
```

`\CustomAcronymFields`

```
4453 \newcommand*\CustomAcronymFields{%
4454   name={\the\glsshorttok},%
4455   description={\the\glslongtok},%
4456   first={\noexpand\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
4457   firstplural={\noexpand\acrfullformat
4458     {\the\glslongtok\noexpand\acrpluralsuffix}{\the\glsshorttok}}%
4459   text={\the\glsshorttok},%
4460   plural={\the\glsshorttok\noexpand\acrpluralsuffix}%
4461 }
```

`\CustomNewAcronymDef`

```
4462 \newcommand*\CustomNewAcronymDef{%
4463   \protected@edef\@do@newglossaryentry{%
4464     \noexpand\newglossaryentry{\the\glslabeltok}%
4465     {%
4466       type=\acronymtype,%
4467       short={\the\glsshorttok},%
4468       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
4469       long={\the\glslongtok},%
4470       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
4471       user1={\the\glsshorttok},%
4472       user2={\the\glsshorttok\noexpand\acrpluralsuffix},%
4473       user3={\the\glslongtok},%
4474       user4={\the\glslongtok\noexpand\acrpluralsuffix},%
4475       \CustomAcronymFields,%
4476       \the\glskeylisttok
4477     }%
4478   }%
4479   \@do@newglossaryentry
4480 }
```

`\SetCustomStyle`

```
4481 \newcommand*\SetCustomStyle{%
4482   \renewcommand{\newacronym}[4][\]{%
4483     \ifx\@glsacronymlists\@empty
4484       \def\@glo@type{\acronymtype}%
4485       \setkeys{glossentry}{##1}%
4486       \DeclareAcronymList{\@glo@type}%
4487       \SetCustomDisplayStyle{\@glo@type}%
4488     \fi
4489   }
```

```

4488 \fi
4489 \glskeylisttok{##1}%
4490 \glslabeltok{##2}%
4491 \glsshorttok{##3}%
4492 \glslongtok{##4}%
4493 \newacronymhook
4494 \CustomNewAcronymDef
4495 }%

Set the display
4496 \@for\@gls@type:=\@glsacronymlists\do{%
4497 \SetCustomDisplayStyle{\@gls@type}%
4498 }%
4499 }

```

`\defineAcronymSynonyms`

```

4500 \newcommand*{\DefineAcronymSynonyms}{%

```

Short form

`\acs`

```

4501 \let\acs\acrshort

```

First letter uppercase short form

`\Acs`

```

4502 \let\Acs\Acrshort

```

Plural short form

`\acsp`

```

4503 \let\acsp\acrshortpl

```

First letter uppercase plural short form

`\Acsp`

```

4504 \let\Acsp\Acrshortpl

```

Long form

`\acl`

```

4505 \let\acl\aclong

```

Plural long form

`\aclp`

```

4506 \let\aclp\aclongpl

```

First letter upper case long form

`\Acl`

```

4507 \let\Acl\Aclong

```

First letter upper case plural long form

\Ac1p

4508 \let\Ac1p\Acrlongpl

Full form

\acf

4509 \let\acf\acrfull

Plural full form

\acfp

4510 \let\acfp\acrfullpl

First letter upper case full form

\Acf

4511 \let\Acf\Acrfull

First letter upper case plural full form

\Acfp

4512 \let\Acfp\Acrfullpl

Standard form

\ac

4513 \let\ac\gls

First upper case standard form

\Ac

4514 \let\Ac\Gls

Standard plural form

\acp

4515 \let\acp\glspl

Standard first letter upper case plural form

\Acp

4516 \let\Acp\Glspl

4517 }

Define synonyms if required

4518 \ifglsacrshortcuts

4519 \DefineAcronymSynonyms

4520 \fi

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.

```
4521 \RequirePackage{glossary-hypernav}
```

The styles that use list-like environments. These are not loaded if the `nolist` option is used:

```
4522 \@gls@loadlist
```

The styles that use the `longtable` environment. These are not loaded if the `no-long package` option is used.

```
4523 \@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.

```
4524 \@gls@loadsuper
```

The tree-like styles. These are not loaded if the `notree package` option is used.

```
4525 \@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.

```
4526 \ifx\@glossary@default@style\relax
4527 \else
4528   \glossarystyle{\@glossary@default@style}
4529 \fi
```

1.19 Debugging Commands

`\showgloparent` `\showgloparent{<label>}`

```
4530 \newcommand*{\showgloparent}[1]{%
4531   \expandafter\show\csname glo@#1@parent\endcsname
4532 }
```

`\showglolevel` `\showglolevel{<label>}`

```
4533 \newcommand*{\showglolevel}[1]{%
4534   \expandafter\show\csname glo@#1@level\endcsname
4535 }
```

`\showglotext` `\showglotext{<label>}`

```

4536 \newcommand*{\showglotext}[1]{%
4537   \expandafter\show\csname glo@#1@text\endcsname
4538 }

```

\showgloplural \showgloplural{\label}

```

4539 \newcommand*{\showgloplural}[1]{%
4540   \expandafter\show\csname glo@#1@plural\endcsname
4541 }

```

\showglofirst \showglofirst{\label}

```

4542 \newcommand*{\showglofirst}[1]{%
4543   \expandafter\show\csname glo@#1@first\endcsname
4544 }

```

\showglofirstpl \showglofirstpl{\label}

```

4545 \newcommand*{\showglofirstpl}[1]{%
4546   \expandafter\show\csname glo@#1@firstpl\endcsname
4547 }

```

\showglotype \showglotype{\label}

```

4548 \newcommand*{\showglotype}[1]{%
4549   \expandafter\show\csname glo@#1@type\endcsname
4550 }

```

\showglocounter \showglocounter{\label}

```

4551 \newcommand*{\showglocounter}[1]{%
4552   \expandafter\show\csname glo@#1@counter\endcsname
4553 }

```

\showglouser \showglouser{\label}

```

4554 \newcommand*{\showglouser}[1]{%
4555   \expandafter\show\csname glo@#1@user\endcsname
4556 }

```

\showglouserii \showglouserii{\label{}}

```
4557 \newcommand*{\showglouserii}[1]{%  
4558   \expandafter\show\csname glo@#1@userii\endcsname  
4559 }
```

\showglouseriii \showglouseriii{\label{}}

```
4560 \newcommand*{\showglouseriii}[1]{%  
4561   \expandafter\show\csname glo@#1@useriii\endcsname  
4562 }
```

\showglouseriv \showglouseriv{\label{}}

```
4563 \newcommand*{\showglouseriv}[1]{%  
4564   \expandafter\show\csname glo@#1@useriv\endcsname  
4565 }
```

\showglouserv \showglouserv{\label{}}

```
4566 \newcommand*{\showglouserv}[1]{%  
4567   \expandafter\show\csname glo@#1@userv\endcsname  
4568 }
```

\showglouservi \showglouservi{\label{}}

```
4569 \newcommand*{\showglouservi}[1]{%  
4570   \expandafter\show\csname glo@#1@uservi\endcsname  
4571 }
```

\showgloname \showgloname{\label{}}

```
4572 \newcommand*{\showgloname}[1]{%  
4573   \expandafter\show\csname glo@#1@name\endcsname  
4574 }
```

\showglodesc \showglodesc{\label{}}

```
4575 \newcommand*{\showglodesc}[1]{%  
4576   \expandafter\show\csname glo@#1@desc\endcsname  
4577 }
```

`\showglodescplural` `\showglodescplural{<label>}`

```
4578 \newcommand*{\showglodescplural}[1]{%
4579   \expandafter\show\csname glo@#1@descplural\endcsname
4580 }
```

`\showglosort` `\showglosort{<label>}`

```
4581 \newcommand*{\showglosort}[1]{%
4582   \expandafter\show\csname glo@#1@sort\endcsname
4583 }
```

`\showglosymbol` `\showglosymbol{<label>}`

```
4584 \newcommand*{\showglosymbol}[1]{%
4585   \expandafter\show\csname glo@#1@symbol\endcsname
4586 }
```

`\showglosymbolplural` `\showglosymbolplural{<label>}`

```
4587 \newcommand*{\showglosymbolplural}[1]{%
4588   \expandafter\show\csname glo@#1@symbolplural\endcsname
4589 }
```

`\showgloindex` `\showgloindex{<label>}`

```
4590 \newcommand*{\showgloindex}[1]{%
4591   \expandafter\show\csname glo@#1@index\endcsname
4592 }
```

`\showgloflag` `\showgloflag{<label>}`

```
4593 \newcommand*{\showgloflag}[1]{%
4594   \expandafter\show\csname ifglo@#1@flag\endcsname
4595 }
```

`\showacronymlists` `\showacronymlists`

Show list of glossaries that have been flagged as a list of acronyms.


```

4596 \newcommand*{\showacronymlists}{%
4597   \show\@glsacronymlists
4598 }

```

\showglossaries \showglossaries

Show list of defined glossaries.

```

4599 \newcommand*{\showglossaries}{%
4600   \show\@glo@types
4601 }

```

\showglossaryin \showglossaryin{<glossary-label>}

Show the ‘in’ extension for the given glossary.

```

4602 \newcommand*{\showglossaryin}[1]{%
4603   \expandafter\show\csname @glo@type@#1@in\endcsname
4604 }

```

\showglossaryout \showglossaryout{<glossary-label>}

Show the ‘out’ extension for the given glossary.

```

4605 \newcommand*{\showglossaryout}[1]{%
4606   \expandafter\show\csname @glo@type@#1@out\endcsname
4607 }

```

\showglossarytitle \showglossarytitle{<glossary-label>}

Show the title for the given glossary.

```

4608 \newcommand*{\showglossarytitle}[1]{%
4609   \expandafter\show\csname @glo@type@#1@title\endcsname
4610 }

```

\showglossarycounter \showglossarycounter{<glossary-label>}

Show the counter for the given glossary.

```

4611 \newcommand*{\showglossarycounter}[1]{%
4612   \expandafter\show\csname @glo@type@#1@counter\endcsname
4613 }

```

\showglossaryentries \showglossaryentries{<glossary-label>}

Show the list of entry labels for the given glossary.

```
4614 \newcommand*{\showglossaryentries}[1]{%
4615   \expandafter\show\csname glolist@#1\endcsname
4616 }
```

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.

```
4617 \csname ifglscpatible-2.07\endcsname
4618   \RequirePackage{glossaries-compatible-207}
4619 \fi
```

2 Mfirstuc Documented Code

```
4620 \NeedsTeXFormat{LaTeX2e}
4621 \ProvidesPackage{mfirstuc}[2012/05/21 v1.06 (NLCT)]
```

Requires etoolbox:

```
4622 \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`.

```
4623 \newif\if@glscs
4624 \newtoks\@glsmfirst
4625 \newtoks\@glsmrest
4626 \def\makefirstuc#1{%
4627   \def\gls@argi{#1}%
4628   \ifx\gls@argi\@empty
```

If the argument is empty, do nothing.

```
4629   \else
```

```

4630 \def\@gls@tmp{\ #1}%
4631 \@onelevel@sanitize\@gls@tmp
4632 \expandafter\@gls@checkcs\@gls@tmp\relax\relax
4633 \if@glscs
4634 \@gls@getbody #1{}\@nil
4635 \ifx\@gls@rest\@empty
4636 \glsmakefirstuc{#1}%
4637 \else
4638 \expandafter\@gls@split\@gls@rest\@nil
4639 \ifx\@gls@first\@empty
4640 \glsmakefirstuc{#1}%
4641 \else
4642 \expandafter\@glsmfirst\expandafter{\@gls@first}%
4643 \expandafter\@glsmrest\expandafter{\@gls@rest}%
4644 \edef\@gls@domfirstuc{\noexpand\@gls@body
4645 {\noexpand\glsmakefirstuc\the\@glsmfirst}%
4646 \the\@glsmrest}%
4647 \@gls@domfirstuc
4648 \fi
4649 \fi
4650 \else
4651 \glsmakefirstuc{#1}%
4652 \fi
4653 \fi
4654 }

```

Put first argument in \@gls@first and second argument in \@gls@rest:

```

4655 \def\@gls@split#1#2\@nil{%
4656 \def\@gls@first{#1}\def\@gls@rest{#2}%
4657 }

4658 \def\@gls@checkcs#1 #2#3\relax{%
4659 \def\@gls@argi{#1}\def\@gls@argii{#2}%
4660 \ifx\@gls@argi\@gls@argii
4661 \@glscstrue
4662 \else
4663 \@glscsfalse
4664 \fi
4665 }

```

Make first thing upper case:

```

4666 \def\@gls@makefirstuc#1{\MakeUppercase #1}

```

\glsmakefirstuc Provide a user command to make it easier to customise.

```

4667 \newcommand*{\glsmakefirstuc}[1]{\@gls@makefirstuc{#1}}

```

Get the first grouped argument and stores in \@gls@body.

```

4668 \def\@gls@getbody#1#{\def\@gls@body{#1}\@gls@gobbletonil}

```

Scoup up everything to \@nil and store in \@gls@rest:

```

4669 \def\@gls@gobbletonil#1\@nil{\def\@gls@rest{#1}}

```

`\xmakefirstuc` Expand argument once before applying `\makefirstuc` (added v1.01).

```
4670 \newcommand*\xmakefirstuc}[1]{%
4671 \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).

```
4672 \newcommand*\capitalisewords}[1]{%
4673   \def\gls@add@space{%
4674     \mfu@capitalisewords#1 \@nil\mfu@endcap
4675     %\gls@add@space\makefirstuc{##1}\def\gls@add@space{ }%
4676 }

4677 \def\mfu@capitalisewords#1 #2\mfu@endcap{%
4678   \def\mfu@cap@first{#1}%
4679   \def\mfu@cap@second{#2}%
4680   \gls@add@space
4681   \makefirstuc{#1}%
4682   \def\gls@add@space{ }%
4683   \ifx\mfu@cap@second\@nnil
4684     \let\next\mfu@cap\mfu@noop
4685   \else
4686     \let\next\mfu@cap\mfu@capitalisewords
4687   \fi
4688   \next\mfu@cap#2\mfu@endcap
4689 }
4690 \def\mfu@noop#1\mfu@endcap{}
```

`\xcapitalisewords` Short-cut command:

```
4691 \newcommand*\xcapitalisewords}[1]{%
4692   \expandafter\capitalisewords\expandafter{#1}%
4693 }
```

3 Glossary Styles

3.1 Glossary hyper-navigation definitions (glossary-hypernav package)

Package Definition:

```
4694 \ProvidesPackage{glossary-hypernav}[2007/07/04 v1.01 (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`

```
4695 \newcommand*{\glsnavhyperlink}[3][\@glo@type]{%
4696   \edef\gls@grplabel{#2}\protected@edef\gls@grptitle{#3}%
4697   \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`

```
4698 \newcommand*{\glsnavhypertarget}[3][\@glo@type]{%
  Add this group to the aux file for re-run check.
4699   \protected@write\@auxout{}{\string\@gls@hypergroup{#1}{#2}}%
  Add the target.
4700   \@glstarget{glsn:#1@#2}{#3}%
  Check list of know groups to determine if a re-run is required.
4701   \expandafter\let
4702     \expandafter\@gls@list\csname @gls@hypergroup@list@#1\endcsname
  Iterate through list and terminate loop if this group is found.
4703   \@for\@gls@elem:=\@gls@list\do{%
4704     \ifthenelse{\equal{\@gls@elem}{#2}}{\@endfortrue}{}%
  Check if list terminated prematurely.
4705   \if@endfor
4706   \else
    This group was not included in the list, so issue a warning.
4707     \GlossariesWarningNoLine{Navigation panel
4708       for glossary type ‘#1’^^Jmissing group ‘#2’}%
4709     \gdef\gls@hypergroup@rerun{%
4710       \GlossariesWarningNoLine{Navigation panel
4711         has changed. Rerun LaTeX}}%
4712     \fi
4713 }
```

`\gls@hypergroup@rerun` Give a warning at the end if re-run required

```
4714 \let\gls@hypergroup@rerun\relax
4715 \AtEndDocument{\gls@hypergroup@rerun}
```

`\@gls@hypergroup` This adds to (or creates) the command `\@gls@hypergroup@list@<glossary type>` which lists all groups for a given glossary, so that the navigation bar only contains those groups that are present. However it requires at least 2 runs to ensure the information is up-to-date.

```
4716 \newcommand*{\@gls@hypergroup}[2]{%
```

```

4717 \ifundefined{@gls@hypergroup@#1}{%
4718   \expandafter\xdef\csname @gls@hypergroup@#1\endcsname{#2}%
4719 }{%
4720   \expandafter\let\expandafter@gls@tmp
4721     \csname @gls@hypergroup@#1\endcsname
4722   \expandafter\xdef\csname @gls@hypergroup@#1\endcsname{%
4723     @gls@tmp,#2}%
4724 }%
4725 }

```

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`

```

4726 \newcommand*{\glsnavigation}{%
4727 \def@gls@between{}%
4728 \ifundefined{@gls@hypergroup@}\@glo@type}{%
4729   \def@gls@list{}%
4730 }{%
4731   \expandafter\let\expandafter@gls@list
4732     \csname @gls@hypergroup@\@glo@type\endcsname
4733 }%
4734 \@for@gls@tmp:=@gls@list\do{%
4735   @gls@between
4736   \glsnavhyperlink{@gls@tmp}{\glsgetgrouptitle{@gls@tmp}}%
4737   \let@gls@between\glshypernavsep%
4738 }%
4739 }

```

`\glshypernavsep` Separator for the hyper navigation bar.

```

4740 \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`

```

4741 \newcommand*{\glssymbolnav}{%
4742 \glsnavhyperlink{glssymbols}{\glsgetgrouptitle{glssymbols}}%
4743 \glshypernavsep
4744 \glsnavhyperlink{glsnumbers}{\glsgetgrouptitle{glsnumbers}}%
4745 \glshypernavsep
4746 }

```

3.2 In-line Style (glossary-inline.sty)

This defines an in-line style where the entries are comma-separated with just the name and description displayed.

```
4747 \ProvidesPackage{glossary-inline}[2012/09/21 v3.03 (NLCT)]
```

`inline` Define the inline style.

```
4748 \newglossarystyle{inline}{%
```

Start of glossary sets up first empty separator between entries. (This is then changed by `\glossaryentryfield`)

```
4749   \renewenvironment{theglossary}{%
4750     {%
4751       \def\gls@inlinesep{}%
4752       \def\gls@inlinesubsep{}%
4753       \def\gls@inlinepostchild{}%
4754     }%
4755     {\glspostinline}%
```

No header:

```
4756 \renewcommand*{\glossaryheader}{}%
```

No group headings (if heading is required, add `\glsinlinedopostchild` to start definition in case heading follows a child entry):

```
4757 \renewcommand*{\glsgroupheading}[1]{}%
```

Just display separator followed by name and description:

```
4758 \renewcommand{\glossaryentryfield}[5]{%
4759   \glsinlinedopostchild
4760   \gls@inlinesep
4761   \def\glo@desc{##3}%
4762   \def\@no@post@desc{\nopostdesc}%
4763   \glsentryitem{##1}\glsinlinenameformat{##1}{##2}%
4764   \ifx\glo@desc\@no@post@desc
4765     \glsinlineemptydescformat{##4}{##5}%
4766   \else
4767     \ifstrempy{##3}%
4768     {\glsinlineemptydescformat{##4}{##5}}%
4769     {\glsinlinedescformat{##3}{##4}{##5}}%
4770   \fi
4771   \ifglshaschildren{##1}%
4772   {%
4773     \glsresetsubentrycounter
4774     \glsinlineparentchildseparator
4775     \def\gls@inlinesubsep{}%
4776     \def\gls@inlinepostchild{\glsinlinepostchild}%
4777   }%
4778   {%
4779     \def\gls@inlinesep{\glsinlineseparator}%
4780   }%
```

Sub-entries display description:

```

4781 \renewcommand{\glossarysubentryfield}[6]{%
4782   \gls@inlinesubsep%
4783   \glsinlinesubnameformat{##2}{##3}%
4784   \glssubentryitem{##2}\glsinlinesubdescformat{##4}{##5}{##6}%
4785   \def\gls@inlinesubsep{\glsinlinesubseparator}%
4786 }%
```

Nothing special between groups:

```

4787 \renewcommand*{\glsgroupskip}{}%
4788 }
```

`\glsinlinedopostchild`

```

4789 \newcommand*{\glsinlinedopostchild}{%
4790   \gls@inlinepostchild
4791   \def\gls@inlinepostchild{}%
4792 }
```

`\glsinlineseparator` Separator to use between entries.

```

4793 \newcommand*{\glsinlineseparator}{;\space}
```

`\glsinlinesubseparator` Separator to use between sub-entries.

```

4794 \newcommand*{\glsinlinesubseparator}{,\space}
```

`\glsinlinparentchildseparator` Separator to use between parent and children.

```

4795 \newcommand*{\glsinlinparentchildseparator}{:\space}
```

`\glsinlinepostchild` Hook to use between child and next entry

```

4796 \newcommand*{\glsinlinepostchild}{}%
```

`\glspostinline` Terminator for inline glossary.

```

4797 \newcommand*{\glspostinline}{\glspostdescription\space}
```

`\glsinlinenameformat` Formats the name of the entry (first argument label, second argument name):

```

4798 \newcommand*{\glsinlinenameformat}[2]{\glstarget{#1}{#2}}
```

`\glsinlinedescformat` Formats the entry's description, symbol and location list:

```

4799 \newcommand*{\glsinlinedescformat}[3]{\space#1}
```

`\glsinlineemptydescformat` Formats the entry's symbol and location list when the description is empty:

```

4800 \newcommand*{\glsinlineemptydescformat}[2]{}
```

`\glsinlinesubnameformat` Formats the name of the subentry (first argument label, second argument name):

```

4801 \newcommand*{\glsinlinesubnameformat}[2]{\glstarget{#1}{}}
```

`\glsinlinesubdescformat` Formats the subentry's description, symbol and location list:

```

4802 \newcommand*{\glsinlinesubdescformat}[3]{#1}
```


3.3 List Style (glossary-list.sty)

The style file defines glossary styles that use the description environment. Note that since the entry name is placed in the optional argument to the `\item` command, it will appear in a bold font by default.

```
4803 \ProvidesPackage{glossary-list}[2012/11/11 v3.04 (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.

```
4804 \newglossarystyle{list}{%
```

Use description environment:

```
4805 \renewenvironment{theglossary}{%
```

```
4806 {\begin{description}}{\end{description}}%
```

No header at the start of the environment:

```
4807 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
4808 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries start a new item in the list:

```
4809 \renewcommand*{\glossaryentryfield}[5]{%
```

```
4810 \item[\glsubentryitem{##1}\glstarget{##1}{##2}]
```

```
4811 ##3\glspostdescription\space ##5}%
```

Sub-entries continue on the same line:

```
4812 \renewcommand*{\glossarysubentryfield}[6]{%
```

```
4813 \glssubentryitem{##2}%
```

```
4814 \glstarget{##2}{\strut}##4\glspostdescription\space ##6.}%
```

```
4815% \end{macrocode}
```

```
4816% Add vertical space between groups:
```

```
4817%\changes{3.03}{2012/09/21}{added check for glsnogroupskip}
```

```
4818% \begin{macrocode}
```

```
4819 \renewcommand*{\glsgroupskip}{\ifglsgroupskip\else\indexspace\fi}%
```

```
4820 }
```

`listgroup` The listgroup style is like the list style, but the glossary groups have headings.

```
4821 \newglossarystyle{listgroup}{%
```

Base it on the list style:

```
4822 \glossarystyle{list}%
```

Each group has a heading:

```
4823 \renewcommand*{\glsgroupheading}[1]{\item[\glsgrouptitle{##1}]}
```

`listhypergroup` The `listhypergroup` style is like the `listgroup` style, but has a set of links to the groups at the start of the glossary.

```
4824 \newglossarystyle{listhypergroup}{%
```

Base it on the list style:

```
4825 \glossarystyle{list}%
```

Add navigation links at the start of the environment:

```
4826 \renewcommand*{\glossaryheader}{%
```

```
4827 \item[\glsnavigation]}%
```

Each group has a heading with a hypertarget:

```
4828 \renewcommand*{\glsgroupheading}[1]{%
```

```
4829 \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.

```
4830 \newglossarystyle{altlist}{%
```

Base it on the list style:

```
4831 \glossarystyle{list}%
```

Main (level 0) entries start a new item in the list with a line break after the entry name:

```
4832 \renewcommand*{\glossaryentryfield}[5]{%
```

```
4833 \item[\glsentryitem{##1}\glstarget{##1}{##2}]%
```

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.

```
4834 \mbox{}\par\nobreak\@afterheading
```

```
4835 ##3\glspostdescription\space ##5}%
```

Sub-entries start a new paragraph:

```
4836 \renewcommand{\glossarysubentryfield}[6]{%
```

```
4837 \par
```

```
4838 \glssubentryitem{##2}%
```

```
4839 \glstarget{##2}{\strut}##4\glspostdescription\space ##6}%
```

```
4840 }
```

`altlistgroup` The `altlistgroup` glossary style is like the `altlist` style, but the glossary groups have headings.

```
4841 \newglossarystyle{altlistgroup}{%
```

Base it on the `altlist` style:

```
4842 \glossarystyle{altlist}%
```

Each group has a heading:

```
4843 \renewcommand*{\glsgroupheading}[1]{\item[\glsgetgrouptitle{##1}]}}
```

`altlisthypergroup` The `altlisthypergroup` glossary style is like the `altlistgroup` style, but has a set of links to the groups at the start of the glossary.

```
4844 \newglossarystyle{altlisthypergroup}{%
```

Base it on the `altlist` style:

```
4845 \glossarystyle{altlist}%
```

Add navigation links at the start of the environment:

```
4846 \renewcommand*{\glossaryheader}{%
```

```
4847 \item[\glsnavigation]}%
```

Each group has a heading with a `hypertarget`:

```
4848 \renewcommand*{\glsgroupheading}[1]{%
```

```
4849 \item[\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}]}}
```

`listdotted` The `listdotted` glossary style was supplied by Axel Menzel. I've modified it slightly so that the distance from the start of the name to the end of the dotted line is specified by `\glslistdottedwidth`. Note that this style ignores the page numbers as well as the symbol. Sub-entries are displayed in the same way as top-level entries.

```
4850 \newglossarystyle{listdotted}{%
```

Base it on the `list` style:

```
4851 \glossarystyle{list}%
```

Each main (level 0) entry starts a new item:

```
4852 \renewcommand*{\glossaryentryfield}[5]{%
```

```
4853 \item[]\makebox[\glslistdottedwidth][l]{%
```

```
4854 \glstryitem{##1}\glstarget{##1}{##2}%
```

```
4855 \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##3}%
```

Sub entries have the same format as main entries:

```
4856 \renewcommand*{\glossarysubentryfield}[6]{%
```

```
4857 \item[]\makebox[\glslistdottedwidth][l]{%
```

```
4858 \glssubentryitem{##2}%
```

```
4859 \glstarget{##2}{##3}%
```

```
4860 \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##4}%
```

```
4861 }
```

`\glslistdottedwidth`

```
4862 \newlength\glslistdottedwidth
```

```
4863 \setlength{\glslistdottedwidth}{.5\hsize}
```

`sublistdotted` This style is similar to the `glostylelistdotted` style, except that the main entries just have the name displayed.

```
4864 \newglossarystyle{sublistdotted}{%
```

Base it on the `listdotted` style:

```
4865 \glossarystyle{listdotted}%
```

Main (level 0) entries just display the name:

```
4866 \renewcommand*{\glossaryentryfield}[5]{%
4867 \item[\glsentryitem{##1}\glstarget{##1}{##2}]}%
4868 }
```

3.4 Glossary Styles using longtable (the glossary-long package)

The glossary styles defined in the package used the longtable environment in the glossary.

```
4869 \ProvidesPackage{glossary-long}[2012/09/21 v3.03 (NLCT)]
```

Requires the package:

```
4870 \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`.)

```
4871 \@ifundefined{glsdescwidth}{%
4872 \newlength\glsdescwidth
4873 \setlength{\glsdescwidth}{0.6\hsize}
4874 }{}
```

`\glspagelistwidth` This is a length that governs the width of the page list column.

```
4875 \@ifundefined{glspagelistwidth}{%
4876 \newlength\glspagelistwidth
4877 \setlength{\glspagelistwidth}{0.1\hsize}
4878 }{}
```

`long` The long glossary style command which uses the longtable environment:

```
4879 \newglossarystyle{long}{%
```

Use longtable with two columns:

```
4880 \renewenvironment{theglossary}%
4881 {\begin{longtable}\lp{\glsdescwidth}}%
4882 {\end{longtable}}%
```

Do nothing at the start of the environment:

```
4883 \renewcommand*{\glossaryheader}{}%
```

No heading between groups:

```
4884 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries displayed in a row:

```
4885 \renewcommand*{\glossaryentryfield}[5]{%
4886 \glsentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
```

Sub entries displayed on the following row without the name:

```
4887 \renewcommand*{\glossarysubentryfield}[6]{%
4888 &
```

```

4889     \glssubentryitem{##2}%
4890     \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%

```

Blank row between groups:

```

4891     \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else & \\fi}%
4892 }

```

longborder The longborder style is like the above, but with horizontal and vertical lines:

```

4893 \newglossarystyle{longborder}{%
    Base it on the glostylelong style:
4894     \glossarystyle{long}%
    Use longtable with two columns with vertical lines between each column:
4895     \renewenvironment{theglossary}{%
4896         \begin{longtable}{|l|p{\glstdescwidth}|}{\end{longtable}}%
    Place horizontal lines at the head and foot of the table:
4897     \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
4898 }

```

longheader The longheader style is like the long style but with a header:

```

4899 \newglossarystyle{longheader}{%
    Base it on the glostylelong style:
4900     \glossarystyle{long}%
    Set the table's header:
4901     \renewcommand*{\glossaryheader}{%
4902         \bfseries \entryname & \bfseries \descriptionname\\endhead}%
4903 }

```

longheaderborder The longheaderborder style is like the long style but with a header and border:

```

4904 \newglossarystyle{longheaderborder}{%
    Base it on the glostylelongborder style:
4905     \glossarystyle{longborder}%
    Set the table's header and add horizontal line to table's foot:
4906     \renewcommand*{\glossaryheader}{%
4907         \hline\bfseries \entryname & \bfseries \descriptionname\\hline
4908         \endhead
4909         \hline\endfoot}%
4910 }

```

long3col The long3col style is like long but with 3 columns

```

4911 \newglossarystyle{long3col}{%
    Use a longtable with 3 columns:
4912     \renewenvironment{theglossary}{%
4913         {\begin{longtable}{lp{\glstdescwidth}p{\glspagelistwidth}}}%
4914         {\end{longtable}}%

```

No table header:

```
4915 \renewcommand*{\glossaryheader}{}%
```

No headings between groups:

```
4916 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
4917 \renewcommand*{\glossaryentryfield}[5]{%
```

```
4918 \glstryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\\}%
```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```
4919 \renewcommand*{\glossarysubentryfield}[6]{%
```

```
4920 &
```

```
4921 \glssubentryitem{##2}%
```

```
4922 \glstarget{##2}{\strut}##4 & ##6\\}%
```

Blank row between groups:

```
4923 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else & &\\fi}%
```

```
4924 }
```

long3colborder The long3colborder style is like the long3col style but with a border:

```
4925 \newglossarystyle{long3colborder}{%
```

Base it on the glostylelong3col style:

```
4926 \glossarystyle{long3col}%
```

Use a longtable with 3 columns with vertical lines around them:

```
4927 \renewenvironment{theglossary}{%
```

```
4928 {\begin{longtable}{|l|p{\glsdescwidth}|p{\glspagelistwidth|}}%
```

```
4929 {\end{longtable}}}%
```

Place horizontal lines at the head and foot of the table:

```
4930 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
```

```
4931 }
```

long3colheader The long3colheader style is like long3col but with a header row:

```
4932 \newglossarystyle{long3colheader}{%
```

Base it on the glostylelong3col style:

```
4933 \glossarystyle{long3col}%
```

Set the table's header:

```
4934 \renewcommand*{\glossaryheader}{%
```

```
4935 \bfseries\entryname&\bfseries\descriptionname&
```

```
4936 \bfseries\pagelistname\\endhead}%
```

```
4937 }
```

ong3colheaderborder The long3colheaderborder style is like the above but with a border

```
4938 \newglossarystyle{long3colheaderborder}{%
```

Base it on the `glostylelong3colborder` style:

```
4939 \glossarystyle{long3colborder}%
```

Set the table's header and add horizontal line at table's foot:

```
4940 \renewcommand*{\glossaryheader}{%
4941   \hline
4942   \bfseries\entryname&\bfseries\descriptionname&
4943   \bfseries\pagelistname\\ \hline\endhead
4944   \hline\endfoot}%
4945 }
```

`long4col` The `long4col` style has four columns where the third column contains the value of the associated symbol key.

```
4946 \newglossarystyle{long4col}{%
```

Use a `longtable` with 4 columns:

```
4947 \renewenvironment{theglossary}%
4948   {\begin{longtable}{llll}}%
4949   {\end{longtable}}%
```

No table header:

```
4950 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
4951 \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):

```
4952 \renewcommand*{\glossaryentryfield}[5]{%
4953   \glstentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
4954 \renewcommand*{\glossarysubentryfield}[6]{%
4955   &
4956   \glssubentryitem{##2}%
4957   \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%
```

Blank row between groups:

```
4958 \renewcommand*{\glsgroupskip}{\ifglsgnোগroupskip\else & & \\ \fi}%
4959 }
```

`long4colheader` The `long4colheader` style is like `long4col` but with a header row.

```
4960 \newglossarystyle{long4colheader}{%
```

Base it on the `glostylelong4col` style:

```
4961 \glossarystyle{long4col}%
```

Table has a header:

```
4962 \renewcommand*{\glossaryheader}{%
4963   \bfseries\entryname&\bfseries\descriptionname&
4964   \bfseries \symbolname&
```

```

4965 \bfseries\pagelistname\\\endhead}%
4966 }

```

long4colborder The long4colborder style is like long4col but with a border.

```

4967 \newglossarystyle{long4colborder}{%

```

Base it on the glostylelong4col style:

```

4968 \glossarystyle{long4col}%

```

Use a longtable with 4 columns surrounded by vertical lines:

```

4969 \renewenvironment{theglossary}%
4970 {\begin{longtable}{|l|l|l|l|}}%
4971 {\end{longtable}}%

```

Add horizontal lines to the head and foot of the table:

```

4972 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
4973 }

```

long4colheaderborder The long4colheaderborder style is like the above but with a border.

```

4974 \newglossarystyle{long4colheaderborder}{%

```

Base it on the glostylelong4col style:

```

4975 \glossarystyle{long4col}%

```

Use a longtable with 4 columns surrounded by vertical lines:

```

4976 \renewenvironment{theglossary}%
4977 {\begin{longtable}{|l|l|l|l|}}%
4978 {\end{longtable}}%

```

Add table header and horizontal line at the table's foot:

```

4979 \renewcommand*{\glossaryheader}{%
4980 \hline\bfseries\entryname&\bfseries\descriptionname&
4981 \bfseries \symbolname&
4982 \bfseries\pagelistname\\\hline\endhead\hline\endfoot}%
4983 }

```

altlong4col The altlong4col style is like the long4col style but can have multiline descriptions and page lists.

```

4984 \newglossarystyle{altlong4col}{%

```

Base it on the glostylelong4col style:

```

4985 \glossarystyle{long4col}%

```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```

4986 \renewenvironment{theglossary}%
4987 {\begin{longtable}{lp{\glstdescwidth}lp{\glspagelistwidth}}}%
4988 {\end{longtable}}%
4989 }

```

altlong4colheader The altlong4colheader style is like altlong4col but with a header row.

```

4990 \newglossarystyle{altlong4colheader}{%

```


Base it on the `glostylelong4colheader` style:

```
4991 \glossarystyle{long4colheader}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
4992 \renewenvironment{theglossary}%  
4993   {\begin{longtable}{lp{\glsdescwidth}lp{\glspagelistwidth}}}%  
4994   {\end{longtable}}%  
4995 }
```

`altlong4colborder` The `altlong4colborder` style is like `altlong4col` but with a border.

```
4996 \newglossarystyle{altlong4colborder}{%
```

Base it on the `glostylelong4colborder` style:

```
4997 \glossarystyle{long4colborder}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
4998 \renewenvironment{theglossary}%  
4999   {\begin{longtable}{|lp{\glsdescwidth}|lp{\glspagelistwidth}|}%  
5000   {\end{longtable}}%  
5001 }
```

`altlong4colheaderborder` The `altlong4colheaderborder` style is like the above but with a header as well as a border.

```
5002 \newglossarystyle{altlong4colheaderborder}{%
```

Base it on the `glostylelong4colheaderborder` style:

```
5003 \glossarystyle{long4colheaderborder}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
5004 \renewenvironment{theglossary}%  
5005   {\begin{longtable}{|lp{\glsdescwidth}|lp{\glspagelistwidth}|}%  
5006   {\end{longtable}}%  
5007 }
```

3.5 Glossary Styles using `longtable` (the `glossary-longragged` package)

The glossary styles defined in the package used the `longtable` environment in the glossary and use ragged right formatting for the multiline columns.

```
5008 \ProvidesPackage{glossary-longragged}[2012/09/21 v3.03 (NLCT)]
```

Requires the package:

```
5009 \RequirePackage{array}
```

Requires the package:

```
5010 \RequirePackage{longtable}
```

`\glsdescwidth` This is a length that governs the width of the description column. This may have already been defined.

```
5011 \@ifundefined{glsdescwidth}{%
5012   \newlength\glsdescwidth
5013   \setlength{\glsdescwidth}{0.6\hsize}
5014 }{}
```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```
5015 \@ifundefined{glspagelistwidth}{%
5016   \newlength\glspagelistwidth
5017   \setlength{\glspagelistwidth}{0.1\hsize}
5018 }{}
```

`longragged` The longragged glossary style is like the long but uses ragged right formatting for the description column.

```
5019 \newglossarystyle{longragged}{%
```

Use longtable with two columns:

```
5020   \renewenvironment{theglossary}%
5021     {\begin{longtable}[l>{\raggedright}p{\glsdescwidth}]}%
5022     {\end{longtable}}%
```

Do nothing at the start of the environment:

```
5023   \renewcommand*{\glossaryheader}{}%
```

No heading between groups:

```
5024   \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries displayed in a row:

```
5025   \renewcommand*{\glossaryentryfield}[5]{%
5026     \glstarget{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
5027     \tabularnewline}%
```

Sub entries displayed on the following row without the name:

```
5028   \renewcommand*{\glossarysubentryfield}[6]{%
5029     &
5030     \glssubentryitem{##2}%
5031     \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
5032     \tabularnewline}%
```

Blank row between groups:

```
5033   \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else & \tabularnewline\fi}%
5034 }
```

`longraggedborder` The longraggedborder style is like the above, but with horizontal and vertical lines:

```
5035 \newglossarystyle{longraggedborder}{%
```

Base it on the `glostylelongragged` style:

```
5036   \glossarystyle{longragged}%
```

Use longtable with two columns with vertical lines between each column:

```
5037 \renewenvironment{theglossary}{%
5038 \begin{longtable}{|l|>{\raggedright}p{\glsdescwidth}|}%
5039 {\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
5040 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
5041 }
```

longraggedheader The longraggedheader style is like the longragged style but with a header:

```
5042 \newglossarystyle{longraggedheader}{%
```

Base it on the glostylelongragged style:

```
5043 \glossarystyle{longragged}%
```

Set the table's header:

```
5044 \renewcommand*{\glossaryheader}{%
5045 \bfseries \entryname & \bfseries \descriptionname
5046 \tabularnewline\endhead}%
5047 }
```

longraggedheaderborder The longraggedheaderborder style is like the longragged style but with a header and border:

```
5048 \newglossarystyle{longraggedheaderborder}{%
```

Base it on the glostylelongraggedborder style:

```
5049 \glossarystyle{longraggedborder}%
```

Set the table's header and add horizontal line to table's foot:

```
5050 \renewcommand*{\glossaryheader}{%
5051 \hline\bfseries \entryname & \bfseries \descriptionname
5052 \tabularnewline\hline
5053 \endhead
5054 \hline\endfoot}%
5055 }
```

longragged3col The longragged3col style is like longragged but with 3 columns

```
5056 \newglossarystyle{longragged3col}{%
```

Use a longtable with 3 columns:

```
5057 \renewenvironment{theglossary}{%
5058 {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}%
5059 >{\raggedright}p{\gls pagelistwidth}}}%
5060 {\end{longtable}}%
```

No table header:

```
5061 \renewcommand*{\glossaryheader}{}%
```

No headings between groups:

```
5062 \renewcommand*{\gls groupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
5063 \renewcommand*{\glossaryentryfield}[5]{%
5064   \glstentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%
```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```
5065 \renewcommand*{\glossarysubentryfield}[6]{%
5066   &
5067   \glssubentryitem{##2}%
5068   \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%
```

Blank row between groups:

```
5069 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else & &\tabularnewline\fi}%
5070 }
```

`longragged3colborder` The `longragged3colborder` style is like the `longragged3col` style but with a border:

```
5071 \newglossarystyle{longragged3colborder}{%
```

Base it on the `glostylelongragged3col` style:

```
5072 \glossarystyle{longragged3col}%
```

Use a `longtable` with 3 columns with vertical lines around them:

```
5073 \renewenvironment{theglossary}%
5074   {\begin{longtable}{|l|>{\raggedright}p{\glsdescwidth}|%
5075    >{\raggedright}p{\glspagelistwidth}|}%
5076   {\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
5077 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
5078 }
```

`longragged3colheader` The `longragged3colheader` style is like `longragged3col` but with a header row:

```
5079 \newglossarystyle{longragged3colheader}{%
```

Base it on the `glostylelongragged3col` style:

```
5080 \glossarystyle{longragged3col}%
```

Set the table's header:

```
5081 \renewcommand*{\glossaryheader}{%
5082   \bfseries\entryname&\bfseries\descriptionname&
5083   \bfseries\pagelistname\tabularnewline\endhead}%
5084 }
```

`longragged3colheaderborder` The `longragged3colheaderborder` style is like the above but with a border

```
5085 \newglossarystyle{longragged3colheaderborder}{%
```

Base it on the `glostylelongragged3colborder` style:

```
5086 \glossarystyle{longragged3colborder}%
```

Set the table's header and add horizontal line at table's foot:

```

5087 \renewcommand*{\glossaryheader}{%
5088     \hline
5089     \bfseries\entryname&\bfseries\descriptionname&
5090     \bfseries\pagelistname\tabularnewline\hline\endhead
5091     \hline\endfoot}%
5092 }
```

`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.

```

5093 \newglossarystyle{altlongragged4col}{%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```

5094 \renewenvironment{theglossary}{%
5095     {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}l%
5096       >{\raggedright}p{\glspagelistwidth}}}%
5097     {\end{longtable}}}%

```

No table header:

```

5098 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```

5099 \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):

```

5100 \renewcommand*{\glossaryentryfield}[5]{%
5101     \glstentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%

```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```

5102 \renewcommand*{\glossarysubentryfield}[6]{%
5103     &
5104     \glssubentryitem{##2}%
5105     \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%

```

Blank row between groups:

```

5106 \renewcommand*{\glsgroupskip}{\ifglsgnোগroupskip\else & & \tabularnewline\fi}%
5107 }
```

`ongragged4colheader` The `altlongragged4colheader` style is like `altlongragged4col` but with a header row.

```

5108 \newglossarystyle{altlongragged4colheader}{%
```

Base it on the `glostylealtlongragged4col` style:

```

5109 \glossarystyle{altlongragged4col}%

```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
5110 \renewenvironment{theglossary}%
5111     {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}l%
5112      >{\raggedright}p{\glspagelistwidth}}}%
5113     {\end{longtable}}%
```

Table has a header:

```
5114 \renewcommand*{\glossaryheader}{%
5115     \bfseries\entryname&\bfseries\descriptionname&
5116     \bfseries \symbolname&
5117     \bfseries\pagelistname\tabularnewline\endhead}%
5118 }
```

`ongragged4colborder` The `altlongragged4colborder` style is like `altlongragged4col` but with a border.

```
5119 \newglossarystyle{altlongragged4colborder}{%
```

Base it on the `glostylealtlongragged4col` style:

```
5120 \glossarystyle{altlongragged4col}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
5121 \renewenvironment{theglossary}%
5122     {\begin{longtable}{|l|>{\raggedright}p{\glsgdescwidth}|l|%
5123      >{\raggedright}p{\glspagelistwidth}|}%
5124     {\end{longtable}}%
```

Add horizontal lines to the head and foot of the table:

```
5125 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
5126 }
```

`ged4colheaderborder` The `altlongragged4colheaderborder` style is like the above but with a header as well as a border.

```
5127 \newglossarystyle{altlongragged4colheaderborder}{%
```

Base it on the `glostylealtlongragged4col` style:

```
5128 \glossarystyle{altlongragged4col}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
5129 \renewenvironment{theglossary}%
5130     {\begin{longtable}{|l|>{\raggedright}p{\glsgdescwidth}|l|%
5131      >{\raggedright}p{\glspagelistwidth}|}%
5132     {\end{longtable}}%
```

Add table header and horizontal line at the table's foot:

```
5133 \renewcommand*{\glossaryheader}{%
5134     \hline\bfseries\entryname&\bfseries\descriptionname&
5135     \bfseries \symbolname&
5136     \bfseries\pagelistname\tabularnewline\hline\endhead
5137     \hline\endfoot}%
5138 }
```

3.6 Glossary Styles using multicol (glossary-mcols.sty)

The style file defines glossary styles that use the multicol package. These use the tree-like glossary styles in a multicol environment.

```
5139 \ProvidesPackage{glossary-mcols}[2013/04/21 v3.05 (NLCT)]
```

Required packages:

```
5140 \RequirePackage{multicol}
```

```
5141 \RequirePackage{glossary-tree}
```

`\glsmcols` Define macro in which to store the number of columns. (Defaults to 2.)

```
5142 \newcommand*\glsmcols{2}
```

`mcolindex` Multi-column index style. Same as the index, but puts the glossary in multiple columns. (Ideally the glossary title should go in the optional argument of multicol, but the title isn't part of the glossary style.)

```
5143 \newglossarystyle{mcolindex}{%
```

```
5144   \glossarystyle{index}%
```

```
5145   \renewenvironment{theglossary}%
```

```
5146     {%
```

```
5147       \begin{multicols}{\glsmcols}
```

```
5148       \setlength{\parindent}{0pt}%
```

```
5149       \setlength{\parskip}{0pt plus 0.3pt}%
```

```
5150       \let\item\@idxitem}%
```

```
5151     }\end{multicols}}%
```

```
5152 }
```

`mcolindexgroup` As `mcolindex` but has headings:

```
5153 \newglossarystyle{mcolindexgroup}{%
```

```
5154   \glossarystyle{mcolindex}%
```

```
5155   \renewcommand*\glsgroupheading[1]{%
```

```
5156     \item\textbf{\glsgroupheading{##1}}\indexspace}%
```

```
5157 }
```

`mcolindexhypergroup` The `mcolindexhypergroup` style is like the `mcolindexgroup` style but has hyper navigation.

```
5158 \newglossarystyle{mcolindexhypergroup}{%
```

Base it on the `glostylemcolindex` style:

```
5159   \glossarystyle{mcolindex}%
```

Put navigation links to the groups at the start of the glossary:

```
5160   \renewcommand*\glossaryheader{%
```

```
5161     \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.

```
5162   \renewcommand*\glsgroupheading[1]{%
```

```
5163     \item\textbf{\glshypernavigation{\glshypernavigation{##1}}}\indexspace}%
```

```

5164 \indexspace}%
5165 }

```

mcoltree Multi-column index style. Same as the tree, but puts the glossary in multiple columns.

```

5166 \newglossarystyle{mcoltree}{%
5167 \glossarystyle{tree}%
5168 \renewenvironment{theglossary}%
5169 {%
5170 \begin{multicols}{\glsmcols}
5171 \setlength{\parindent}{0pt}%
5172 \setlength{\parskip}{0pt plus 0.3pt}%
5173 }%
5174 {\end{multicols}}}%
5175 }

```

mcoltreegroup Like the mcoltree style but the glossary groups have headings.

```

5176 \newglossarystyle{mcoltreegroup}{%
    Base it on the glostylemcoltree style:
5177 \glossarystyle{mcoltree}%
    Each group has a heading (in bold) followed by a vertical gap):
5178 \renewcommand{\glsgroupheading}[1]{\par
5179 \noindent\textbf{\glsgrouptitle{##1}}\par\indexspace}%
5180 }

```

mcoltreehypergroup The mcoltreehypergroup style is like the treegroup style, but has a set of links to the groups at the start of the glossary.

```

5181 \newglossarystyle{mcoltreehypergroup}{%
    Base it on the glostylemcoltree style:
5182 \glossarystyle{mcoltree}%
    Put navigation links to the groups at the start of the theglossary environment:
5183 \renewcommand*{\glossaryheader}{%
5184 \par\noindent\textbf{\glsnavigation}\par\indexspace}%
    Each group has a heading (in bold with a target) followed by a vertical gap):
5185 \renewcommand*{\glsgroupheading}[1]{%
5186 \par\noindent
5187 \textbf{\glsnavigationhypertarget{##1}{\glsgrouptitle{##1}}}\par
5188 \indexspace}%
5189 }

```

mcoltreenoname Multi-column index style. Same as the treenoname, but puts the glossary in multiple columns.

```

5190 \newglossarystyle{mcoltreenoname}{%
5191 \glossarystyle{treenoname}%
5192 \renewenvironment{theglossary}%
5193 {%

```



```

5194     \begin{multicols}{\glsmcols}
5195     \setlength{\parindent}{0pt}%
5196     \setlength{\parskip}{0pt plus 0.3pt}%
5197 }%
5198 {\end{multicols}}}%
5199 }

```

mcoltreenonamegroup Like the mcoltreenoname style but the glossary groups have headings.

```

5200 \newglossarystyle{mcoltreenonamegroup}{%
    Base it on the glostylemcoltreenoname style:
5201   \glossarystyle{mcoltreenoname}%
    Give each group a heading:
5202   \renewcommand{\glsgroupheading}[1]{\par
5203     \noindent\textbf{\glsgrouptitle{##1}}\par\indexspace}%
5204 }

```

treenonamehypergroup The mcoltreenonamehypergroup style is like the mcoltreenonamegroup style, but has a set of links to the groups at the start of the glossary.

```

5205 \newglossarystyle{mcoltreenonamehypergroup}{%
    Base it on the glostylemcoltreenoname style:
5206   \glossarystyle{mcoltreenoname}%
    Put navigation links to the groups at the start of the theglossary environment:
5207   \renewcommand*{\glossaryheader}{%
5208     \par\noindent\textbf{\glsnavigation}\par\indexspace}%
    Each group has a heading (in bold with a target) followed by a vertical gap):
5209   \renewcommand*{\glsgroupheading}[1]{%
5210     \par\noindent
5211     \textbf{\glsnavigationhypertarget{##1}{\glsgrouptitle{##1}}}\par
5212     \indexspace}%
5213 }

```

mcolalmtree Multi-column index style. Same as the almtree, but puts the glossary in multiple columns.

```

5214 \newglossarystyle{mcolalmtree}{%
5215   \glossarystyle{almtree}%
5216   \renewenvironment{theglossary}%
5217   {%
5218     \begin{multicols}{\glsmcols}
5219     \def\@gls@prevlevel{-1}%
5220     \mbox{}\par
5221   }%
5222   {\par\end{multicols}}}%
5223 }

```

`mcolalattoreegroup` Like the `mcolalattoree` style but the glossary groups have headings.

```
5224 \newglossarystyle{mcolalattoreegroup}{%  
    Base it on the glostylemcolalattoree style:  
5225 \glossarystyle{mcolalattoree}%  
    Give each group a heading.  
5226 \renewcommand{\glsgroupheading}[1]{\par  
5227 \def\@gls@prevlevel{-1}%  
5228 \hangindent0pt\relax  
5229 \parindent0pt\relax  
5230 \textbf{\glsgrouptitle{##1}}\par\indexspace}%  
5231 }
```

`colalattoreehypergroup` The `mcolalattoreehypergroup` style is like the `mcolalattoreegroup` style, but has a set of links to the groups at the start of the glossary.

```
5232 \newglossarystyle{mcolalattoreehypergroup}{%  
    Base it on the glostylemcolalattoree style:  
5233 \glossarystyle{mcolalattoree}%  
    Put the navigation links in the header  
5234 \renewcommand*{\glossaryheader}{%  
5235 \par  
5236 \def\@gls@prevlevel{-1}%  
5237 \hangindent0pt\relax  
5238 \parindent0pt\relax  
5239 \textbf{\glsnavigation}\par\indexspace}%  
    Put a hypertarget at the start of each group  
5240 \renewcommand*{\glsgroupheading}[1]{%  
5241 \par  
5242 \def\@gls@prevlevel{-1}%  
5243 \hangindent0pt\relax  
5244 \parindent0pt\relax  
5245 \textbf{\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}\par  
5246 \indexspace}}
```

3.7 Glossary Styles using supertabular environment (`glossary-super` package)

The glossary styles defined in the package use the `supertabular` environment.

```
5247 \ProvidesPackage{glossary-super}[2012/09/21 v3.03 (NLCT)]
```

Requires the package:

```
5248 \RequirePackage{supertabular}
```

`\glsdescwidth` This is a length that governs the width of the description column. This may already have been defined if `has` has been loaded.

```
5249 \@ifundefined{glsdescwidth}{%
```

```

5250 \newlength\glsdescwidth
5251 \setlength{\glsdescwidth}{0.6\hsize}
5252 }{}

```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined if has been loaded.

```

5253 \@ifundefined{glspagelistwidth}{%
5254 \newlength\glspagelistwidth
5255 \setlength{\glspagelistwidth}{0.1\hsize}
5256 }{}

```

`super` The super glossary style uses the supertabular environment (it uses lengths defined in the package.)

```

5257 \newglossarystyle{super}{%

```

Put the glossary in a supertabular environment with two columns and no head or tail:

```

5258 \renewenvironment{theglossary}%
5259 {\tablehead{}\tabletail{}}%
5260 \begin{supertabular}{lp{\glsdescwidth}}%
5261 {\end{supertabular}}%

```

Do nothing at the start of the table:

```

5262 \renewcommand*{\glossaryheader}{}%

```

No group headings:

```

5263 \renewcommand*{\glsgroupheading}[1]{}%

```

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```

5264 \renewcommand*{\glossaryentryfield}[5]{%
5265 \glstryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%

```

Sub entries put in a row (no name, description and page list in second column):

```

5266 \renewcommand*{\glossarysubentryfield}[6]{%
5267 &
5268 \glssubentryitem{##2}%
5269 \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%

```

Blank row between groups:

```

5270 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else & \\fi}%
5271 }

```

`superborder` The superborder style is like the above, but with horizontal and vertical lines:

```

5272 \newglossarystyle{superborder}{%

```

Base it on the `glostylessuper` style:

```

5273 \glossarystyle{super}%

```

Put the glossary in a supertabular environment with two columns and a horizontal line in the head and tail:

```
5274 \renewenvironment{theglossary}%
5275   {\tablehead{\hline}\tabletail{\hline}%
5276   \begin{supertabular}{|l|p{\glsdescwidth}|}}%
5277   {\end{supertabular}}%
5278 }
```

superheader The superheader style is like the super style, but with a header:

```
5279 \newglossarystyle{superheader}{%
```

Base it on the glostylesuper style:

```
5280 \glossarystyle{super}%
```

Put the glossary in a supertabular environment with two columns, a header and no tail:

```
5281 \renewenvironment{theglossary}%
5282   {\tablehead{\bfseries \entryname & \bfseries \descriptionname\\}%
5283   \tabletail{}}%
5284   \begin{supertabular}{lp{\glsdescwidth}}%
5285   {\end{supertabular}}%
5286 }
```

superheaderborder The superheaderborder style is like the super style but with a header and border:

```
5287 \newglossarystyle{superheaderborder}{%
```

Base it on the glostylesuper style:

```
5288 \glossarystyle{super}%
```

Put the glossary in a supertabular environment with two columns, a header and horizontal lines above and below the table:

```
5289 \renewenvironment{theglossary}%
5290   {\tablehead{\hline\bfseries \entryname &
5291   \bfseries \descriptionname\\ \hline}%
5292   \tabletail{\hline}
5293   \begin{supertabular}{|l|p{\glsdescwidth}|}}%
5294   {\end{supertabular}}%
5295 }
```

super3col The super3col style is like the super style, but with 3 columns:

```
5296 \newglossarystyle{super3col}{%
```

Put the glossary in a supertabular environment with three columns and no head or tail:

```
5297 \renewenvironment{theglossary}%
5298   {\tablehead{} \tabletail{}}%
5299   \begin{supertabular}{lp{\glsdescwidth}p{\glspagelistwidth}}%
5300   {\end{supertabular}}%
```

Do nothing at the start of the table:

```
5301 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
5302 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
5303 \renewcommand*{\glossaryentryfield}[5]{%
```

```
5304 \glsentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\\}%
```

Sub entries on a row (no name, description in second column, page list in last column):

```
5305 \renewcommand*{\glossarysubentryfield}[6]{%
```

```
5306 &
```

```
5307 \glssubentryitem{##2}%
```

```
5308 \glstarget{##2}{\strut}##4 & ##6\\}%
```

Blank row between groups:

```
5309 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else & &\\fi}%
```

```
5310 }
```

super3colborder The `super3colborder` style is like the `super3col` style, but with a border:

```
5311 \newglossarystyle{super3colborder}{%
```

Base it on the `glostylesuper3col` style:

```
5312 \glossarystyle{super3col}%
```

Put the glossary in a `supertabular` environment with three columns and a horizontal line in the head and tail:

```
5313 \renewenvironment{theglossary}%
```

```
5314 {\tablehead{\hline}\tabletail{\hline}%
```

```
5315 \begin{supertabular}{|l|p{\glsdescwidth}|p{\glspagelistwidth}|}%
```

```
5316 {\end{supertabular}}%
```

```
5317 }
```

super3colheader The `super3colheader` style is like the `super3col` style but with a header row:

```
5318 \newglossarystyle{super3colheader}{%
```

Base it on the `glostylesuper3col` style:

```
5319 \glossarystyle{super3col}%
```

Put the glossary in a `supertabular` environment with three columns, a header and no tail:

```
5320 \renewenvironment{theglossary}%
```

```
5321 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
```

```
5322 \bfseries\pagelistname\\}\tabletail{}%
```

```
5323 \begin{supertabular}{lp{\glsdescwidth}p{\glspagelistwidth}}%
```

```
5324 {\end{supertabular}}%
```

```
5325 }
```

super3colheaderborder The `super3colheaderborder` style is like the `super3col` style but with a header and border:

```
5326 \newglossarystyle{super3colheaderborder}{%
```

Base it on the `glostylesuper3colborder` style:

```
5327 \glossarystyle{super3colborder}%
    Put the glossary in a supertabular environment with three columns, a header
    with horizontal lines and a horizontal line in the tail:
5328 \renewenvironment{theglossary}%
5329     {\tablehead{\hline
5330         \bfseries\entryname&\bfseries\descriptionname&
5331         \bfseries\pagelistname\\ \hline}%
5332     \tabletail{\hline}%
5333     \begin{supertabular}{|l|p{\glstdescwidth}|p{\glspagelistwidth}|}%
5334     {\end{supertabular}}%
5335 }
```

`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.

```
5336 \newglossarystyle{super4col}{%
    Put the glossary in a supertabular environment with four columns and no head
    or tail:
5337 \renewenvironment{theglossary}%
5338     {\tablehead{}\tabletail}%
5339     \begin{supertabular}{|l|l|l|l|}%
5340     \end{supertabular}}%
```

Do nothing at the start of the table:

```
5341 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
5342 \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:

```
5343 \renewcommand*{\glossaryentryfield}[5]{%
5344     \glstentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%
```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```
5345 \renewcommand*{\glossarysubentryfield}[6]{%
5346     &
5347     \glssubentryitem{##2}%
5348     \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%
```

Blank row between groups:

```
5349 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else & & \\ \fi}%
5350 }
```

`super4colheader` The `super4colheader` style is like the `super4col` but with a header row.

```
5351 \newglossarystyle{super4colheader}{%
```

Base it on the `glostylesuper4col` style:

```
5352 \glossarystyle{super4col}%
```

Put the glossary in a supertabular environment with four columns, a header and no tail:

```

5353 \renewenvironment{theglossary}%
5354   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
5355               \bfseries\symbolname &
5356               \bfseries\pagelistname\\}%
5357   \tabletail{}}%
5358   \begin{supertabular}{|l|l|l|l|}%
5359   {\end{supertabular}}%
5360 }

```

super4colborder The super4colborder style is like the super4col but with a border.

```

5361 \newglossarystyle{super4colborder}{%

```

Base it on the glostylesuper4col style:

```

5362 \glossarystyle{super4col}%

```

Put the glossary in a supertabular environment with four columns and a horizontal line in the head and tail:

```

5363 \renewenvironment{theglossary}%
5364   {\tablehead{\hline}\tabletail{\hline}%
5365   \begin{supertabular}{|l|l|l|l|}%
5366   {\end{supertabular}}%
5367 }

```

super4colheaderborder The super4colheaderborder style is like the super4col but with a header and border.

```

5368 \newglossarystyle{super4colheaderborder}{%

```

Base it on the glostylesuper4col style:

```

5369 \glossarystyle{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:

```

5370 \renewenvironment{theglossary}%
5371   {\tablehead{\hline\bfseries\entryname&\bfseries\descriptionname&
5372               \bfseries\symbolname &
5373               \bfseries\pagelistname\\hline}\tabletail{\hline}%
5374   \begin{supertabular}{|l|l|l|l|}%
5375   {\end{supertabular}}%
5376 }

```

altsuper4col The altsuper4col glossary style is like super4col but has provision for multiline descriptions.

```

5377 \newglossarystyle{altsuper4col}{%

```

Base it on the glostylesuper4col style:

```

5378 \glossarystyle{super4col}%

```

Put the glossary in a supertabular environment with four columns and no head or tail:

```
5379 \renewenvironment{theglossary}%
5380   {\tablehead{}\tabletail{}}%
5381   \begin{supertabular}{lp{\glsgdescwidth}lp{\glspagelistwidth}}}%
5382   {\end{supertabular}}}%
5383 }
```

altsuper4colheader The altsuper4colheader style is like the altsuper4col but with a header row.

```
5384 \newglossarystyle{altsuper4colheader}{%
```

Base it on the glostylesuper4colheader style:

```
5385 \glossarystyle{super4colheader}%
```

Put the glossary in a supertabular environment with four columns, a header and no tail:

```
5386 \renewenvironment{theglossary}%
5387   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
5388     \bfseries\symbolname &
5389     \bfseries\pagelistname\\}\tabletail{}}%
5390   \begin{supertabular}{lp{\glsgdescwidth}lp{\glspagelistwidth}}}%
5391   {\end{supertabular}}}%
5392 }
```

altsuper4colborder The altsuper4colborder style is like the altsuper4col but with a border.

```
5393 \newglossarystyle{altsuper4colborder}{%
```

Base it on the glostylesuper4colborder style:

```
5394 \glossarystyle{super4colborder}%
```

Put the glossary in a supertabular environment with four columns and a horizontal line in the head and tail:

```
5395 \renewenvironment{theglossary}%
5396   {\tablehead{\hline}\tabletail{\hline}%
5397   \begin{supertabular}%
5398     {\lllp{\glsgdescwidth}lllp{\glspagelistwidth}}}%
5399   {\end{supertabular}}}%
5400 }
```

per4colheaderborder The altsuper4colheaderborder style is like the altsuper4col but with a header and border.

```
5401 \newglossarystyle{altsuper4colheaderborder}{%
```

Base it on the glostylesuper4colheaderborder style:

```
5402 \glossarystyle{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:

```
5403 \renewenvironment{theglossary}%
5404   {\tablehead{\hline
```



```

5405      \bfseries\entryname &
5406      \bfseries\descriptionname &
5407      \bfseries\symbolname &
5408      \bfseries\pagelistname\\hline}%
5409      \tabletail{\hline}%
5410      \begin{supertabular}%
5411        {\l|p{\glsdescwidth}|l|p{\glspagelistwidth}|}%
5412      {\end{supertabular}}%
5413 }

```

3.8 Glossary Styles using supertabular environment (glossary-superragged package)

The glossary styles defined in the package use the supertabular environment. These styles are like those provided by the package, except that the multiline columns have ragged right justification.

```

5414 \ProvidesPackage{glossary-superragged}[2012/09/21 v3.03 (NLCT)]

```

Requires the package:

```

5415 \RequirePackage{array}

```

Requires the package:

```

5416 \RequirePackage{supertabular}

```

`\glsdescwidth` This is a length that governs the width of the description column. This may already have been defined.

```

5417 \@ifundefined{glsdescwidth}{%
5418   \newlength\glsdescwidth
5419   \setlength{\glsdescwidth}{0.6\hsize}
5420 }{}

```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```

5421 \@ifundefined{glspagelistwidth}{%
5422   \newlength\glspagelistwidth
5423   \setlength{\glspagelistwidth}{0.1\hsize}
5424 }{}

```

`superragged` The superragged glossary style uses the supertabular environment.

```

5425 \newglossarystyle{superragged}{%

```

Put the glossary in a supertabular environment with two columns and no head or tail:

```

5426   \renewenvironment{theglossary}%
5427     {\tablehead{}\tabletail{}}%
5428     \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}}%
5429     {\end{supertabular}}%

```

Do nothing at the start of the table:

```

5430   \renewcommand*{\glossaryheader}{}%

```

No group headings:

```
5431 \renewcommand*{\glsgroupheading}[1]{%
```

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```
5432 \renewcommand*{\glossaryentryfield}[5]{%
5433 \glsentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
5434 \tabularnewline}%
```

Sub entries put in a row (no name, description and page list in second column):

```
5435 \renewcommand*{\glossarysubentryfield}[6]{%
5436 &
5437 \glssubentryitem{##2}%
5438 \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
5439 \tabularnewline}%
```

Blank row between groups:

```
5440 \renewcommand*{\glsgroupskip}{\ifglsgnogroupskip\else & \tabularnewline\fi}%
5441 }
```

superraggedborder The superraggedborder style is like the above, but with horizontal and vertical lines:

```
5442 \newglossarystyle{superraggedborder}{%
```

Base it on the glostylesuperragged style:

```
5443 \glossarystyle{superragged}%
```

Put the glossary in a supertabular environment with two columns and a horizontal line in the head and tail:

```
5444 \renewenvironment{theglossary}%
5445 {\tablehead{\hline}\tabletail{\hline}%
5446 \begin{supertabular}{|l|>\raggedright}p{\glsdescwidth}|}%
5447 {\end{supertabular}}%
5448 }
```

superraggedheader The superraggedheader style is like the super style, but with a header:

```
5449 \newglossarystyle{superraggedheader}{%
```

Base it on the glostylesuperragged style:

```
5450 \glossarystyle{superragged}%
```

Put the glossary in a supertabular environment with two columns, a header and no tail:

```
5451 \renewenvironment{theglossary}%
5452 {\tablehead{\bfseries \entryname & \bfseries \descriptionname
5453 \tabularnewline}%
5454 \tabletail{}%
5455 \begin{supertabular}{|l|>\raggedright}p{\glsdescwidth}}}%
5456 {\end{supertabular}}%
5457 }
```

`rraggedheaderborder` The `superraggedheaderborder` style is like the `superragged` style but with a header and border:

```
5458 \newglossarystyle{superraggedheaderborder}{%
```

Base it on the `glostylesuper` style:

```
5459 \glossarystyle{superragged}%
```

Put the glossary in a `supertabular` environment with two columns, a header and horizontal lines above and below the table:

```
5460 \renewenvironment{theglossary}%
5461 {\tablehead{\hline\bfseries \entryname &
5462 \bfseries \descriptionname\tabularnewline\hline}%
5463 \tabletail{\hline}
5464 \begin{supertabular}{|l|>{\raggedright}p{\glstdescwidth}|}%
5465 {\end{supertabular}}%
5466 }
```

`superragged3col` The `superragged3col` style is like the `superragged` style, but with 3 columns:

```
5467 \newglossarystyle{superragged3col}{%
```

Put the glossary in a `supertabular` environment with three columns and no head or tail:

```
5468 \renewenvironment{theglossary}%
5469 {\tablehead{}\tabletail{}}%
5470 \begin{supertabular}{|l>{\raggedright}p{\glstdescwidth}%
5471 >{\raggedright}p{\glspagelistwidth}}}%
5472 {\end{supertabular}}%
```

Do nothing at the start of the table:

```
5473 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
5474 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
5475 \renewcommand*{\glossaryentryfield}[5]{%
5476 \glstryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%
```

Sub entries on a row (no name, description in second column, page list in last column):

```
5477 \renewcommand*{\glossarysubentryfield}[6]{%
5478 &
5479 \glssubentryitem{##2}%
5480 \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%
```

Blank row between groups:

```
5481 \renewcommand*{\glsgroupskip}{\ifglsgnোগroupskip\else & \tabularnewline\fi}%
5482 }
```

`superragged3colborder` The `superragged3colborder` style is like the `superragged3col` style, but with a border:

```
5483 \newglossarystyle{superragged3colborder}{%
```

Base it on the `glostylesuperragged3col` style:

```
5484 \glossarystyle{superragged3col}%
```

Put the glossary in a `supertabular` environment with three columns and a horizontal line in the head and tail:

```
5485 \renewenvironment{theglossary}%
5486 {\tablehead{\hline}\tabletail{\hline}%
5487 \begin{supertabular}{|l|>{\raggedright}p{\glsgdescwidth}|%
5488 >{\raggedright}p{\glspagelistwidth}|}%
5489 {\end{supertabular}}%
5490 }
```

`superragged3colheader` The `superragged3colheader` style is like the `superragged3col` style but with a header row:

```
5491 \newglossarystyle{superragged3colheader}{%
```

Base it on the `glostylesuperragged3col` style:

```
5492 \glossarystyle{superragged3col}%
```

Put the glossary in a `supertabular` environment with three columns, a header and no tail:

```
5493 \renewenvironment{theglossary}%
5494 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
5495 \bfseries\pagelistname\tabularnewline}\tabletail{}}%
5496 \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}%
5497 >{\raggedright}p{\glspagelistwidth}}}%
5498 {\end{supertabular}}%
5499 }
```

`superragged3colheaderborder` The `superragged3colheaderborder` style is like the `superragged3col` style but with a header and border:

```
5500 \newglossarystyle{superragged3colheaderborder}{%
```

Base it on the `glostylesuperragged3colborder` style:

```
5501 \glossarystyle{superragged3colborder}%
```

Put the glossary in a `supertabular` environment with three columns, a header with horizontal lines and a horizontal line in the tail:

```
5502 \renewenvironment{theglossary}%
5503 {\tablehead{\hline
5504 \bfseries\entryname&\bfseries\descriptionname&
5505 \bfseries\pagelistname\tabularnewline\hline}%
5506 \tabletail{\hline}%
5507 \begin{supertabular}{|l|>{\raggedright}p{\glsgdescwidth}|%
5508 >{\raggedright}p{\glspagelistwidth}|}%
5509 {\end{supertabular}}%
5510 }
```

`altsuperragged4col` The `altsuperragged4col` glossary style is like `altsuper4col` style in the package but uses ragged right formatting in the description and page list columns.

```

5511 \newglossarystyle{altsuperragged4col}{%
    Put the glossary in a supertabular environment with four columns and no head
    or tail:
5512   \renewenvironment{theglossary}%
5513     {\tablehead{}\tabletail}%
5514     \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}l%
5515       >{\raggedright}p{\glspagelistwidth}}}%
5516     {\end{supertabular}}%

    Do nothing at the start of the table:
5517   \renewcommand*{\glossaryheader}{}%

    No group headings:
5518   \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:
5519   \renewcommand*{\glossaryentryfield}[5]{%
5520     \glstryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%

    Sub entries on a row with no name, the description in the second column, sym-
    bol in third column and page list in last column:
5521   \renewcommand*{\glossarysubentryfield}[6]{%
5522     &
5523     \glssubentryitem{##2}%
5524     \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%

    Blank row between groups:
5525   \renewcommand*{\glsgroupskip}{\ifglsgnigroupskip\else & & \tabularnewline\fi}%
5526 }

```

`altsuperragged4colheader` The `altsuperragged4colheader` style is like the `altsuperragged4col` style but with a header row.

```

5527 \newglossarystyle{altsuperragged4colheader}{%
    Base it on the altsuperragged4col style:
5528   \glossarystyle{altsuperragged4col}%

    Put the glossary in a supertabular environment with four columns, a header and
    no tail:
5529   \renewenvironment{theglossary}%
5530     {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
5531       \bfseries\symbolname &
5532       \bfseries\pagelistname\tabularnewline}\tabletail}%
5533     \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}l%
5534       >{\raggedright}p{\glspagelistwidth}}}%
5535     {\end{supertabular}}%
5536 }

```

`altperragged4colborder` The `altperragged4colborder` style is like the `altperragged4col` style but with a border.

```
5537 \newglossarystyle{altperragged4colborder}{%
```

Base it on the `glostylealtperragged4col` style:

```
5538 \glossarystyle{altperragged4col}{%
```

Put the glossary in a `supertabular` environment with four columns and a horizontal line in the head and tail:

```
5539 \renewenvironment{theglossary}{%
5540   {\tablehead{\hline}\tabletail{\hline}}%
5541   \begin{supertabular}%
5542     {\l|>{\raggedright}p{\glsgdescwidth}|l|}%
5543     >{\raggedright}p{\glspagelistwidth}|}}%
5544   {\end{supertabular}}}%
5545 }
```

`altperragged4colheaderborder` The `altperragged4colheaderborder` style is like the `altperragged4col` style but with a header and border.

```
5546 \newglossarystyle{altperragged4colheaderborder}{%
```

Base it on the `glostylealtperragged4col` style:

```
5547 \glossarystyle{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:

```
5548 \renewenvironment{theglossary}{%
5549   {\tablehead{\hline
5550     \bfseries\entryname &
5551     \bfseries\descriptionname &
5552     \bfseries\symbolname &
5553     \bfseries\pagelistname\tablearnewline\hline}%
5554   \tabletail{\hline}%
5555   \begin{supertabular}%
5556     {\l|>{\raggedright}p{\glsgdescwidth}|l|}%
5557     >{\raggedright}p{\glspagelistwidth}|}}%
5558   {\end{supertabular}}}%
5559 }
```

3.9 Tree Styles (`glossary-tree.sty`)

The style file defines glossary styles that have a tree-like structure. These are designed for hierarchical glossaries.

```
5560 \ProvidesPackage{glossary-tree}[2012/09/21 v3.03 (NLCT)]
```

`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.

```
5561 \newglossarystyle{index}{%
```

Set the paragraph indentation and skip and define `\item` to be the same as that used by theindex:

```
5562 \renewenvironment{theglossary}%
5563   {\setlength{\parindent}{0pt}%
5564    \setlength{\parskip}{0pt plus 0.3pt}%
5565    \let\item\@idxitem}%
5566   {}%
```

Do nothing at the start of the environment:

```
5567 \renewcommand*{\glossaryheader}{}%
```

No group headers:

```
5568 \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.

```
5569 \renewcommand*{\glossaryentryfield}[5]{%
5570 \item\glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
5571 \ifx\relax##4\relax
5572   \else
5573     \space{##4}%
5574   \fi
5575 \space ##3\glspostdescription \space ##5}%
```

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 `\glossaryentryfield`, 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.

```
5576 \renewcommand*{\glossarysubentryfield}[6]{%
5577   \ifcase##1\relax
5578     % level 0
5579     \item
5580   \or
5581     % level 1
5582     \subitem
5583     \glssubentryitem{##2}%
5584   \else
5585     % all other levels
5586     \subsubitem
5587   \fi
5588   \textbf{\glstarget{##2}{##3}}%
5589   \ifx\relax##5\relax
5590     \else
5591       \space{##5}%
5592     \fi
5593   \space##4\glspostdescription\space ##6}%
```

Vertical gap between groups is the same as that used by indices:

```
5594 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}}
```

indexgroup The indexgroup style is like the index style but has headings.

```
5595 \newglossarystyle{indexgroup}{%
```

Base it on the glostyleindex style:

```
5596 \glossarystyle{index}{%
```

Add a heading for each group. This puts the group's title in bold followed by a vertical gap.

```
5597 \renewcommand*{\glsgroupheading}[1]{%
```

```
5598 \item\textbf{\glsgroupgetgrouptitle{##1}}\indexspace}%
```

```
5599 }
```

indexhypergroup The indexhypergroup style is like the indexgroup style but has hyper navigation.

```
5600 \newglossarystyle{indexhypergroup}{%
```

Base it on the glostyleindex style:

```
5601 \glossarystyle{index}{%
```

Put navigation links to the groups at the start of the glossary:

```
5602 \renewcommand*{\glossaryheader}{%
```

```
5603 \item\textbf{\glshnavigation}\indexspace}%
```

Add a heading for each group (with a target). The group's title is in bold followed by a vertical gap.

```
5604 \renewcommand*{\glsgroupheading}[1]{%
```

```
5605 \item\textbf{\glshnavhypertarget{##1}}{\glsgroupgetgrouptitle{##1}}}%
```

```
5606 \indexspace}%
```

```
5607 }
```

tree The tree glossary style is similar in style to the index style, but can have arbitrary levels.

```
5608 \newglossarystyle{tree}{%
```

Set the paragraph indentation and skip:

```
5609 \renewenvironment{theglossary}{%
```

```
5610 {\setlength{\parindent}{0pt}%
```

```
5611 \setlength{\parskip}{0pt plus 0.3pt}}%
```

```
5612 {}%
```

Do nothing at the start of the theglossary environment:

```
5613 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
5614 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries: name in bold, followed by symbol in brackets (if it exists), the description and the page list:

```
5615 \renewcommand{\glossaryentryfield}[5]{%
```

```
5616 \hangindent0pt\relax
```

```
5617 \parindent0pt\relax
```

```
5618 \glstentryitem{##1}\textbf{\glstarget{##1}}{##2}}%
```

```
5619 \ifx\relax##4\relax
```



```

5620 \else
5621 \space(##4)%
5622 \fi
5623 \space ##3\glspostdescription \space ##5\par}%

```

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.

```

5624 \renewcommand{\glossarysubentryfield}[6]{%
5625 \hangindent##1\glstreeindent\relax
5626 \parindent##1\glstreeindent\relax
5627 \ifnum##1=1\relax
5628 \glssubentryitem{##2}%
5629 \fi
5630 \textbf{\glstarget{##2}{##3}}%
5631 \ifx\relax##5\relax
5632 \else
5633 \space(##5)%
5634 \fi
5635 \space##4\glspostdescription\space ##6\par}%

```

Vertical gap between groups is the same as that used by indices:

```

5636 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}}

```

treegroup Like the tree style but the glossary groups have headings.

```

5637 \newglossarystyle{treegroup}{%

```

Base it on the `glostyletree` style:

```

5638 \glossarystyle{tree}%

```

Each group has a heading (in bold) followed by a vertical gap):

```

5639 \renewcommand{\glsgroupheading}[1]{\par
5640 \noindent\textbf{\glsgrouptitle{##1}}\par\indexspace}%
5641 }

```

treehypergroup The `treehypergroup` style is like the `treegroup` style, but has a set of links to the groups at the start of the glossary.

```

5642 \newglossarystyle{treehypergroup}{%

```

Base it on the `glostyletree` style:

```

5643 \glossarystyle{tree}%

```

Put navigation links to the groups at the start of the `theglossary` environment:

```

5644 \renewcommand*{\glossaryheader}{%
5645 \par\noindent\textbf{\glsnavigation}\par\indexspace}%

```

Each group has a heading (in bold with a target) followed by a vertical gap):

```

5646 \renewcommand*{\glsgroupheading}[1]{%
5647 \par\noindent
5648 \textbf{\glshypertarget{##1}{\glsgrouptitle{##1}}}\par
5649 \indexspace}%
5650 }

```

`\glstreeindent` Length governing left indent for each level of the tree style.

```
5651 \newlength\glstreeindent
5652 \setlength{\glstreeindent}{10pt}
```

`treenoname` The `treenoname` glossary style is like the tree style, but doesn't print the name or symbol for sub-levels.

```
5653 \newglossarystyle{treenoname}{%
```

Set the paragraph indentation and skip:

```
5654 \renewenvironment{theglossary}{%
5655     {\setlength{\parindent}{0pt}}%
5656     \setlength{\parskip}{0pt plus 0.3pt}}%
5657     {}%
```

No header:

```
5658 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
5659 \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.

```
5660 \renewcommand{\glossaryentryfield}[5]{%
5661     \hangindent0pt\relax
5662     \parindent0pt\relax
5663     \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
5664     \ifx\relax##4\relax
5665     \else
5666         \space{##4}%
5667     \fi
5668     \space ##3\glspostdescription \space ##5\par}%
```

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.

```
5669 \renewcommand{\glossarysubentryfield}[6]{%
5670     \hangindent##1\glstreeindent\relax
5671     \parindent##1\glstreeindent\relax
5672     \ifnum##1=1\relax
5673         \glssubentryitem{##2}%
5674     \fi
5675     \glstarget{##2}{\strut}%
5676     ##4\glspostdescription\space ##6\par}%
```

Vertical gap between groups is the same as that used by indices:

```
5677 \renewcommand*{\glsgroupskip}{\ifglsgnogroupskip\else\indexspace\fi}%
5678 }
```

`treenonamegroup` Like the `treenoname` style but the glossary groups have headings.

```
5679 \newglossarystyle{treenonamegroup}{%
```

Base it on the `glostyletreenoname` style:

```
5680 \glossarystyle{treenoname}%
```

Give each group a heading:

```
5681 \renewcommand{\glsgroupheading}[1]{\par
5682 \noindent\textbf{\glsgrouptitle{##1}}\par\indexspace}%
5683 }
```

`treenonamehypergroup` The `treenonamehypergroup` style is like the `treenonamegroup` style, but has a set of links to the groups at the start of the glossary.

```
5684 \newglossarystyle{treenonamehypergroup}{%
```

Base it on the `glostytreetreenoname` style:

```
5685 \glossarystyle{treenoname}%
```

Put navigation links to the groups at the start of the `theglossary` environment:

```
5686 \renewcommand*{\glossaryheader}{%
5687 \par\noindent\textbf{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
5688 \renewcommand*{\glsgroupheading}[1]{%
5689 \par\noindent
5690 \textbf{\glsnavigationtarget{##1}{\glsgrouptitle{##1}}}\par
5691 \indexspace}%
5692 }
```

`\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.

```
5693 \newcommand*{\glissetwidest}[2][0]{%
5694 \expandafter\def\csname @glswidestname\romannumeral#1\endcsname{%
5695 #2}%
5696 }
```

`\@glswidestname` Initialise `\@glswidestname`.

```
5697 \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`.

```
5698 \newglossarystyle{alttree}{%
```

Redefine the `theglossary` environment.

```
5699 \renewenvironment{theglossary}%
5700 {\def\@gls@prevlevel{-1}%
5701 \mbox{}\par}%
5702 {\par}%
```

Set the header and group headers to nothing.

```
5703 \renewcommand*{\glossaryheader}{}%
5704 \renewcommand*{\glsgroupheading}[1]{}%
```

Redefine the way that the level 0 entries are displayed.

```
5705 \renewcommand{\glossaryentryfield}[5]{%
```

If the level hasn't changed, keep the same settings, otherwise change `\glstreeindent` accordingly.

```
5706 \ifnum\@gls@prevlevel=0\relax
5707 \else
```

Find out how big the indentation should be by measuring the widest entry.

```
5708 \settowidth{\glstreeindent}{\textbf{\@glswidestname\space}}%
```

Set the hangindent and paragraph indent.

```
5709 \hangindent\glstreeindent
5710 \parindent\glstreeindent
5711 \fi
```

Put the name to the left of the paragraph block.

```
5712 \makebox[0pt][r]{\makebox[\glstreeindent][l]{%
5713 \glstryitem{##1}\textbf{\glstarget{##1}{##2}}}}%
```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
5714 \ifx\relax##4\relax
5715 \else
5716 (##4)\space
5717 \fi
```

Do the description followed by the description terminator and location list.

```
5718 ##3\glspostdescription \space ##5\par
```

Set the previous level to 0.

```
5719 \def\@gls@prevlevel{0}%
5720 }%
```

Redefine the way sub-entries are displayed.

```
5721 \renewcommand{\glossarysubentryfield}[6]{%
```

Increment and display the sub-entry counter if this is a level 1 entry and the sub-entry counter is in use.

```
5722 \ifnum##1=1\relax
5723 \glssubentryitem{##2}%
5724 \fi
```

If the level hasn't changed, keep the same settings, otherwise adjust `\glstreeindent` accordingly.

```
5725 \ifnum\@gls@prevlevel=##1\relax
5726 \else
```

Compute the widest entry for this level, or for level 0 if not defined for this level.

Store in `\gls@tmplen`

```
5727 \@ifundefined{\@glswidestname\romannumeral##1}{%
5728 \settowidth{\gls@tmplen}{\textbf{\@glswidestname\space}}}%
5729 \settowidth{\gls@tmplen}{\textbf{%
5730 \csname \@glswidestname\romannumeral##1\endcsname\space}}}%
```

Determine if going up or down a level

```
5731 \ifnum\@gls@prevlevel<##1\relax
```

Depth has increased, so add the width of the widest entry to `\glstreeindent`.

```
5732      \setlength\glstreeindent\gls@tmplen
5733      \addtolength\glstreeindent\parindent
5734      \parindent\glstreeindent
5735      \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`.

```
5736      \@ifundefined{@glswidestname\romannumeral\@gls@prevlevel}{%
5737      \settowidth{\glstreeindent}{\textbf{%
5738      \@glswidestname\space}}}{%
5739      \settowidth{\glstreeindent}{\textbf{%
5740      \csname @glswidestname\romannumeral\@gls@prevlevel
5741      \endcsname\space}}}{%}
```

Subtract this length from the previous level's paragraph indent and set to `\glstreeindent`.

```
5742      \addtolength\parindent{-\glstreeindent}%
5743      \setlength\glstreeindent\parindent
5744      \fi
5745      \fi
```

Set the hanging indentation.

```
5746      \hangindent\glstreeindent
```

Put the name to the left of the paragraph block

```
5747      \makebox[0pt][r]{\makebox[\gls@tmplen][l]{%
5748      \textbf{\glstarget{##2}{##3}}}}%
```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
5749      \ifx##5\relax\relax
5750      \else
5751      (##5)\space
5752      \fi
```

Do the description followed by the description terminator and location list.

```
5753      ##4\glspostdescription\space ##6\par
```

Set the previous level macro to the current level.

```
5754      \def\@gls@prevlevel{##1}%
5755      }%
```

Vertical gap between groups is the same as that used by indices:

```
5756      \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
5757      }
```

almtreegroup Like the `almtree` style but the glossary groups have headings.

```
5758 \newglossarystyle{almtreegroup}{%
```

Base it on the `glostylealmtree` style:

```
5759 \glossarystyle{almtree}%
```

Give each group a heading.

```
5760 \renewcommand{\glsgroupheading}[1]{\par
5761 \def\@gls@prevlevel{-1}%
5762 \hangindent0pt\relax
5763 \parindent0pt\relax
5764 \textbf{\glsgrouptitle{##1}}\par\indexspace}%
5765 }
```

`alttreehypergroup` The `alttreehypergroup` style is like the `alttreegroup` style, but has a set of links to the groups at the start of the glossary.

```
5766 \newglossarystyle{alttreehypergroup}{%
```

Base it on the `glostylealttree` style:

```
5767 \glossarystyle{alttree}%
```

Put the navigation links in the header

```
5768 \renewcommand*\glossaryheader{%
5769 \par
5770 \def\@gls@prevlevel{-1}%
5771 \hangindent0pt\relax
5772 \parindent0pt\relax
5773 \textbf{\glsnavigation}\par\indexspace}%
```

Put a `hypertarget` at the start of each group

```
5774 \renewcommand*\glsgroupheading[1]{%
5775 \par
5776 \def\@gls@prevlevel{-1}%
5777 \hangindent0pt\relax
5778 \parindent0pt\relax
5779 \textbf{\glsnavhypertarget{##1}}{\glsgrouptitle{##1}}\par
5780 \indexspace}}
```

4 glossaries-compatible-207

Provides compatibility with version 2.07 and below. This uses original glossaries `xindy` and `makeindex` formatting, so can be used with old documents that had customized style files, but hyperlinks may not work properly.

```
5781 \NeedsTeXFormat{LaTeX2e}
5782 \ProvidesPackage{glossaries-compatible-207}[2011/04/02 v1.0 (NLCT)]
```

`\GlsAddXdyAttribute` Adds an attribute in old format.

```
5783 \ifglsxindy
5784 \renewcommand*\GlsAddXdyAttribute[1]{%
5785 \edef\@xdyattributes{\@xdyattributes ^~J \string"#1\string"}%
5786 \expandafter\toks@\expandafter{\@xdylocref}%
5787 \edef\@xdylocref{\the\toks@ ^~J%
5788 (markup-locref
5789 :open \string"\string~\string\setentrycounter
```

```

5790    {\noexpand\glscounter}%
5791    \expandafter\string\csname#1\endcsname
5792    \expandafter\@gobble\string\{\string" ^^J
5793    :close \string"\expandafter\@gobble\string\}\string" ^^J
5794    :attr \string"#1\string")}}

```

Only has an effect before \writeist:

```

5795 \fi

```

\GlsAddXdyCounters

```

5796 \renewcommand*\GlsAddXdyCounters[1]{%
5797   \GlossariesWarning{\string\GlsAddXdyCounters\space not available
5798     in compatibility mode.}%
5799 }

```

Add predefined attributes

```

5800 \GlsAddXdyAttribute{glscounterformat}
5801 \GlsAddXdyAttribute{textrm}
5802 \GlsAddXdyAttribute{textsf}
5803 \GlsAddXdyAttribute{texttt}
5804 \GlsAddXdyAttribute{textbf}
5805 \GlsAddXdyAttribute{textmd}
5806 \GlsAddXdyAttribute{textit}
5807 \GlsAddXdyAttribute{textup}
5808 \GlsAddXdyAttribute{textsl}
5809 \GlsAddXdyAttribute{textsc}
5810 \GlsAddXdyAttribute{emph}
5811 \GlsAddXdyAttribute{glshypernumber}
5812 \GlsAddXdyAttribute{hyperrrm}
5813 \GlsAddXdyAttribute{hypersf}
5814 \GlsAddXdyAttribute{hypertt}
5815 \GlsAddXdyAttribute{hyperbf}
5816 \GlsAddXdyAttribute{hypermd}
5817 \GlsAddXdyAttribute{hyperit}
5818 \GlsAddXdyAttribute{hyperup}
5819 \GlsAddXdyAttribute{hypersl}
5820 \GlsAddXdyAttribute{hypersc}
5821 \GlsAddXdyAttribute{hyperemph}

```

\GlsAddXdyLocation Restore v2.07 definition:

```

5822 \ifglxindy
5823   \renewcommand*{\GlsAddXdyLocation}[2]{%
5824     \edef\@xdyuserlocationdefs{%
5825       \@xdyuserlocationdefs ^^J%
5826       (define-location-class \string"#1\string"^^J\space\space
5827         \space(#2))
5828     }%
5829     \edef\@xdyuserlocationnames{%
5830       \@xdyuserlocationnames^^J\space\space\space

```

```

5831         \string"#1\string"}%
5832     }
5833 \fi

```

\do@wrglossary

```

5834 \renewcommand{\do@wrglossary}[1]{%

```

Determine whether to use xindy or makeindex syntax

```

5835 \ifglxindy

```

Need to determine if the formatting information starts with a (or) indicating a range.

```

5836 \expandafter\@glo@check@mkidxrangechar\@glsnumberformat\@nil
5837 \def\@glo@range{%
5838 \expandafter\if\@glo@prefix(\relax
5839 \def\@glo@range{:open-range}%
5840 \else
5841 \expandafter\if\@glo@prefix)\relax
5842 \def\@glo@range{:close-range}%
5843 \fi
5844 \fi

```

Get the location and escape any special characters

```

5845 \protected@edef\@glslocref{\theglentrycounter}%
5846 \@gls@checkmkidxchars\@glslocref

```

Write to the glossary file using xindy syntax.

```

5847 \glossary[\csname glo@#1@type\endcsname]{%
5848 (indexentry :tkey (\csname glo@#1@index\endcsname)
5849 :locref \string"\@glslocref\string" %
5850 :attr \string"\@glo@suffix\string" \@glo@range
5851 )
5852 }%
5853 \else

```

Convert the format information into the format required for makeindex

```

5854 \@set@glo@numformat\@glo@numfmt\@gls@counter\@glsnumberformat

```

Write to the glossary file using makeindex syntax.

```

5855 \glossary[\csname glo@#1@type\endcsname]{%
5856 \string\glossaryentry{\csname glo@#1@index\endcsname
5857 \@gls@encapchar\@glo@numfmt}\theglentrycounter}}%
5858 \fi
5859 }

```

\@set@glo@numformat Only had 3 arguments in v2.07

```

5860 \def\@set@glo@numformat#1#2#3{%
5861 \expandafter\@glo@check@mkidxrangechar#3\@nil
5862 \protected@edef#1{%
5863 \@glo@prefix setentrycounter[] {#2}%
5864 \expandafter\string\csname\@glo@suffix\endcsname

```



```

5865 }%
5866 \@gls@checkmkidxchars#1%
5867 }

```

\writeist Redefine \writeist back to the way it was in v2.07, but change \istfile to
\glswrite.

```

5868 \ifglxindy
5869   \def\writeist{%
5870     \openout\glswrite=\istfilename
5871     \write\glswrite{;; xindy style file created by the glossaries
5872       package in compatible-2.07 mode}%
5873     \write\glswrite{;; for document '\jobname' on
5874       \the\year-\the\month-\the\day}%
5875     \write\glswrite{^^J; required styles^^J}
5876     \@for\@xdystyle:=\@xdyrequiredstyles\do{%
5877       \ifx\@xdystyle\@empty
5878         \else
5879           \protected@write\glswrite{{(require
5880             \string"\@xdystyle.xdy\string")}}%
5881         \fi
5882     }%
5883     \write\glswrite{^^J%
5884       ; list of allowed attributes (number formats)^^J}%
5885     \write\glswrite{(define-attributes ((\@xdyattributes)))}%
5886     \write\glswrite{^^J; user defined alphabets^^J}%
5887     \write\glswrite{\@xdyuseralphabets}%
5888     \write\glswrite{^^J; location class definitions^^J}%
5889     \protected@edef\@gls@roman{\@roman{0}\string"
5890       \string"roman-numbers-lowercase\string" :sep \string"}%
5891     \@onelevel@sanitize\@gls@roman
5892     \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
5893       :sep \string"}%
5894     \@onelevel@sanitize\@tmp
5895     \ifx\@tmp\@gls@roman
5896       \write\glswrite{(define-location-class
5897         \string"roman-page-numbers\string"^^J\space\space\space
5898         (\string"roman-numbers-lowercase\string")
5899         :min-range-length \@glsminrange)}}%
5900     \else
5901       \write\glswrite{(define-location-class
5902         \string"roman-page-numbers\string"^^J\space\space\space
5903         (:sep "\@gls@roman")
5904         :min-range-length \@glsminrange)}}%
5905     \fi
5906     \write\glswrite{(define-location-class
5907       \string"Roman-page-numbers\string"^^J\space\space\space
5908       (\string"roman-numbers-uppercase\string")
5909       :min-range-length \@glsminrange)}}%
5910     \write\glswrite{(define-location-class

```

```

5911 \string"arabic-page-numbers\string"^^J\space\space\space
5912 (\string"arabic-numbers\string")
5913 :min-range-length \@glsmminrange))}%
5914 \write\glswrite{(define-location-class
5915 \string"alpha-page-numbers\string"^^J\space\space\space
5916 (\string"alpha\string")
5917 :min-range-length \@glsmminrange))}%
5918 \write\glswrite{(define-location-class
5919 \string"Alpha-page-numbers\string"^^J\space\space\space
5920 (\string"ALPHA\string")
5921 :min-range-length \@glsmminrange))}%
5922 \write\glswrite{(define-location-class
5923 \string"Appendix-page-numbers\string"^^J\space\space\space
5924 (\string"ALPHA\string"
5925 :sep \string"\@glAlphacompositor\string"
5926 \string"arabic-numbers\string")
5927 :min-range-length \@glsmminrange))}%
5928 \write\glswrite{(define-location-class
5929 \string"arabic-section-numbers\string"^^J\space\space\space
5930 (\string"arabic-numbers\string"
5931 :sep \string"\glsc compositor\string"
5932 \string"arabic-numbers\string")
5933 :min-range-length \@glsmminrange))}%
5934 \write\glswrite{^^J; user defined location classes}%
5935 \write\glswrite{@x dyuserlocationdefs}%
5936 \write\glswrite{^^J; define cross-reference class^^J}%
5937 \write\glswrite{(define-crossref-class \string"see\string"
5938 :unverified )}%
5939 \write\glswrite{(markup-crossref-list
5940 :class \string"see\string"^^J\space\space\space
5941 :open \string"\string\glseeformat\string"
5942 :close \string"{}\string"))}%
5943 \write\glswrite{^^J; define the order of the location classes}%
5944 \write\glswrite{(define-location-class-order
5945 (\@x dylocationclassorder))}%
5946 \write\glswrite{^^J; define the glossary markup^^J}%
5947 \write\glswrite{(markup-index^^J\space\space\space
5948 :open \string"\string
5949 \glossarysection[\string\glossarytoctitle]{\string
5950 \glossarytitle}\string\glossarypreamble\string~n\string\begin
5951 {theglossary}\string\glossaryheader\string~n\string" ^^J\space
5952 \space\space:close \string"\expandafter\@gobble
5953 \string~%\string~n\string
5954 \end{theglossary}\string\glossarypostamble
5955 \string~n\string" ^^J\space\space\space
5956 :tree))}%
5957 \write\glswrite{(markup-letter-group-list
5958 :sep \string"\string\glsgroupskip\string~n\string"))}%
5959 \write\glswrite{(markup-indexentry

```

```

5960      :open \string"\string\relax \string\glsresetentrylist
5961      \string~n\string"))}%
5962 \write\glswrite{(markup-locclass-list :open
5963 \string"\glsopenbrace\string\glossaryentrynumbers
5964 \glsopenbrace\string\relax\space \string"^^J\space\space\space
5965 :sep \string", \string"
5966 :close \string"\glsclosebrace\glsclosebrace\string"))}%
5967 \write\glswrite{(markup-locref-list
5968 :sep \string"\string\delimN\space\string"))}%
5969 \write\glswrite{(markup-range
5970 :sep \string"\string\delimR\space\string"))}%
5971 \@onelevel@sanitize\gls@suffixF
5972 \@onelevel@sanitize\gls@suffixFF
5973 \ifx\gls@suffixF\@empty
5974 \else
5975 \write\glswrite{(markup-range
5976 :close "\gls@suffixF" :length 1 :ignore-end)}}%
5977 \fi
5978 \ifx\gls@suffixFF\@empty
5979 \else
5980 \write\glswrite{(markup-range
5981 :close "\gls@suffixFF" :length 2 :ignore-end)}}%
5982 \fi
5983 \write\glswrite{^^J; define format to use for locations^^J}%
5984 \write\glswrite{\@xdylocref}%
5985 \write\glswrite{^^J; define letter group list format^^J}%
5986 \write\glswrite{(markup-letter-group-list
5987 :sep \string"\string\glsgroupskip\string~n\string"))}%
5988 \write\glswrite{^^J; letter group headings^^J}%
5989 \write\glswrite{(markup-letter-group
5990 :open-head \string"\string\glsgroupheading
5991 \glsopenbrace\string"^^J\space\space\space
5992 :close-head \string"\glsclosebrace\string"))}%
5993 \write\glswrite{^^J; additional letter groups^^J}%
5994 \write\glswrite{\@xdylettergroups}%
5995 \write\glswrite{^^J; additional sort rules^^J}%
5996 \write\glswrite{\@xdysortrules}%
5997 \noist}
5998 \else
5999 \edef\@gls@actualchar{\string?}
6000 \edef\@gls@encapchar{\string|}
6001 \edef\@gls@levelchar{\string!}
6002 \edef\@gls@quotechar{\string"}
6003 \def\writeist{\relax
6004 \openout\glswrite=\istfilename
6005 \write\glswrite{\expandafter\@gobble\string\% makeindex style file
6006 created by the glossaries package}
6007 \write\glswrite{\expandafter\@gobble\string\% for document
6008 '\jobname' on \the\year-\the\month-\the\day}

```

```

6009 \write\glswrite{actual '@gls@actualchar'}
6010 \write\glswrite{encap '@gls@encapchar'}
6011 \write\glswrite{level '@gls@levelchar'}
6012 \write\glswrite{quote '@gls@quotechar'}
6013 \write\glswrite{keyword \string"\string\glossaryentry\string"}
6014 \write\glswrite{preamble \string"\string\glossarysection[\string
6015 \glossarytoctitle]{\string\glossarytitle}\string
6016 \glossarypreamble\string\n\string\begin{theglossary}\string
6017 \glossaryheader\string\n\string"}
6018 \write\glswrite{postamble \string"\string%\string\n\string
6019 \end{theglossary}\string\glossarypostamble\string\n
6020 \string"}
6021 \write\glswrite{group_skip \string"\string\glsgroupskip\string\n
6022 \string"}
6023 \write\glswrite{item_0 \string"\string%\string\n\string"}
6024 \write\glswrite{item_1 \string"\string%\string\n\string"}
6025 \write\glswrite{item_2 \string"\string%\string\n\string"}
6026 \write\glswrite{item_01 \string"\string%\string\n\string"}
6027 \write\glswrite{item_x1
6028 \string"\string\relax \string\glsresetentrylist\string\n
6029 \string"}
6030 \write\glswrite{item_12 \string"\string%\string\n\string"}
6031 \write\glswrite{item_x2
6032 \string"\string\relax \string\glsresetentrylist\string\n
6033 \string"}
6034 \write\glswrite{delim_0 \string"\string\{\string
6035 \glossaryentrynumbers\string\{\string\relax \string"}
6036 \write\glswrite{delim_1 \string"\string\{\string
6037 \glossaryentrynumbers\string\{\string\relax \string"}
6038 \write\glswrite{delim_2 \string"\string\{\string
6039 \glossaryentrynumbers\string\{\string\relax \string"}
6040 \write\glswrite{delim_t \string"\string\}\string\}\string"}
6041 \write\glswrite{delim_n \string"\string\delimN \string"}
6042 \write\glswrite{delim_r \string"\string\delimR \string"}
6043 \write\glswrite{headings_flag 1}
6044 \write\glswrite{heading_prefix
6045 \string"\string\glsgroupheading\string\{\string"}
6046 \write\glswrite{heading_suffix
6047 \string"\string\}\string\relax
6048 \string\glsresetentrylist \string"}
6049 \write\glswrite{symhead_positive \string"glssymbols\string"}
6050 \write\glswrite{numhead_positive \string"glslnumbers\string"}
6051 \write\glswrite{page_compositor \string"glscpositor\string"}
6052 \@gls@escbsdq\gls@suffixF
6053 \@gls@escbsdq\gls@suffixFF
6054 \ifx\gls@suffixF\@empty
6055 \else
6056 \write\glswrite{suffix_2p \string"\gls@suffixF\string"}
6057 \fi

```

```

6058 \ifx\gls@suffixFF\@empty
6059 \else
6060 \write\glswrite{suffix_3p \string"\gls@suffixFF\string"}
6061 \fi
6062 \noist
6063 }
6064 \fi

```

\noist

```

6065 \renewcommand*{\noist}{\let\writeist\relax}

```

5 Accessibility Support (glossaries-accsupp Code)

The package is experimental. It is intended to provide a means of using the PDF accessibility support in glossary entries. See the documentation for further details about accessibility support.

```

6066 \NeedsTeXFormat{LaTeX2e}

```

Package version number now in line with main glossaries package number but will only be updated when glossaries-accsupp.sty is modified.

```

6067 \ProvidesPackage{glossaries-accsupp}[2011/04/02 v3.0 (NLCT)]
6068 Experimental glossaries accessibility

```

Pass all options to glossaries:

```

6069 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}

```

Process options:

```

6070 \ProcessOptions

```

Required packages:

```

6071 \RequirePackage{glossaries}
6072 \RequirePackage{accsupp}

```

5.1 Defining Replacement Text

The version 0.1 stored the replacement text in the symbol key. This has been changed to use the new keys defined here. Example of use:

```

\newglossaryentry{dr}{name=Dr,description={},access={Doctor}}

```

access The replacement text corresponding to the name key:

```

6073 \define@key{glossentry}{access}{%
6074 \def\@glo@access{#1}%
6075 }

```

textaccess The replacement text corresponding to the text key:

```

6076 \define@key{glossentry}{textaccess}{%
6077 \def\@glo@textaccess{#1}%
6078 }

```

firstaccess The replacement text corresponding to the first key:

```
6079 \define@key{glossentry}{firstaccess}{%
6080   \def\@glo@firstaccess{#1}%
6081 }
```

pluralaccess The replacement text corresponding to the plural key:

```
6082 \define@key{glossentry}{pluralaccess}{%
6083   \def\@glo@pluralaccess{#1}%
6084 }
```

firstpluralaccess The replacement text corresponding to the firstplural key:

```
6085 \define@key{glossentry}{firstpluralaccess}{%
6086   \def\@glo@firstpluralaccess{#1}%
6087 }
```

symbolaccess The replacement text corresponding to the symbol key:

```
6088 \define@key{glossentry}{symbolaccess}{%
6089   \def\@glo@symbolaccess{#1}%
6090 }
```

symbolpluralaccess The replacement text corresponding to the symbolplural key:

```
6091 \define@key{glossentry}{symbolpluralaccess}{%
6092   \def\@glo@symbolpluralaccess{#1}%
6093 }
```

descriptionaccess The replacement text corresponding to the description key:

```
6094 \define@key{glossentry}{descriptionaccess}{%
6095   \def\@glo@descaccess{#1}%
6096 }
```

descriptionpluralaccess The replacement text corresponding to the descriptionplural key:

```
6097 \define@key{glossentry}{descriptionpluralaccess}{%
6098   \def\@glo@descpluralaccess{#1}%
6099 }
```

shortaccess The replacement text corresponding to the short key:

```
6100 \define@key{glossentry}{shortaccess}{%
6101   \def\@glo@shortaccess{#1}%
6102 }
```

shortpluralaccess The replacement text corresponding to the shortplural key:

```
6103 \define@key{glossentry}{shortpluralaccess}{%
6104   \def\@glo@shortpluralaccess{#1}%
6105 }
```

longaccess The replacement text corresponding to the long key:

```
6106 \define@key{glossentry}{longaccess}{%
6107   \def\@glo@longaccess{#1}%
6108 }
```

longpluralaccess The replacement text corresponding to the longplural key:

```
6109 \define@key{glossentry}{longpluralaccess}{%
6110   \def\@glo@longpluralaccess{#1}%
6111 }
```

There are no equivalent keys for the user1...user6 keys. The replacement text would have to be explicitly put in the value, e.g., user1={\glsaccsupp{inches}{in}}.

\@gls@noaccess Indicates that no replacement text has been provided.

```
6112 \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):

```
6113 \let\@gls@oldnewglossaryentryprehook\@newglossaryentryprehook
6114 \renewcommand*{\@newglossaryentryprehook}{%
6115   \@gls@oldnewglossaryentryprehook
6116   \def\@glo@access{\@glo@symbol}%

```

Initialise the other keys:

```
6117   \def\@glo@textaccess{\@glo@access}%
6118   \def\@glo@firstaccess{\@glo@access}%
6119   \def\@glo@pluralaccess{\@glo@textaccess}%
6120   \def\@glo@firstpluralaccess{\@glo@pluralaccess}%
6121   \def\@glo@symbolaccess{\relax}%
6122   \def\@glo@symbolpluralaccess{\@glo@symbolaccess}%
6123   \def\@glo@descaccess{\relax}%
6124   \def\@glo@descpluralaccess{\@glo@descaccess}%
6125   \def\@glo@shortaccess{\relax}%
6126   \def\@glo@shortpluralaccess{\@glo@shortaccess}%
6127   \def\@glo@longaccess{\relax}%
6128   \def\@glo@longpluralaccess{\@glo@longaccess}%
6129 }
```

Add to the end hook:

```
6130 \let\@gls@oldnewglossaryentryposthook\@newglossaryentryposthook
6131 \renewcommand*{\@newglossaryentryposthook}{%
6132   \@gls@oldnewglossaryentryposthook

```

Store the access information:

```
6133   \expandafter
6134     \protected@xdef\csname glo@\@glo@label @access\endcsname{%
6135       \@glo@access}%
6136   \expandafter
6137     \protected@xdef\csname glo@\@glo@label @textaccess\endcsname{%
6138       \@glo@textaccess}%
6139   \expandafter
6140     \protected@xdef\csname glo@\@glo@label @firstaccess\endcsname{%
6141       \@glo@firstaccess}%
6142   \expandafter
6143     \protected@xdef\csname glo@\@glo@label @pluralaccess\endcsname{%

```

```

6144     \@glo@pluralaccess}%
6145 \expandafter
6146     \protected@xdef\csname glo@\@glo@label @firstpluralaccess\endcsname{%
6147     \@glo@firstpluralaccess}%
6148 \expandafter
6149     \protected@xdef\csname glo@\@glo@label @symbolaccess\endcsname{%
6150     \@glo@symbolaccess}%
6151 \expandafter
6152     \protected@xdef\csname glo@\@glo@label @symbolpluralaccess\endcsname{%
6153     \@glo@symbolpluralaccess}%
6154 \expandafter
6155     \protected@xdef\csname glo@\@glo@label @descaccess\endcsname{%
6156     \@glo@descaccess}%
6157 \expandafter
6158     \protected@xdef\csname glo@\@glo@label @descpluralaccess\endcsname{%
6159     \@glo@descpluralaccess}%
6160 \expandafter
6161     \protected@xdef\csname glo@\@glo@label @shortaccess\endcsname{%
6162     \@glo@shortaccess}%
6163 \expandafter
6164     \protected@xdef\csname glo@\@glo@label @shortpluralaccess\endcsname{%
6165     \@glo@shortpluralaccess}%
6166 \expandafter
6167     \protected@xdef\csname glo@\@glo@label @longaccess\endcsname{%
6168     \@glo@longaccess}%
6169 \expandafter
6170     \protected@xdef\csname glo@\@glo@label @longpluralaccess\endcsname{%
6171     \@glo@longpluralaccess}%
6172 }

```

5.2 Accessing Replacement Text

`\glsentryaccess` Get the value of the access key for the entry with the given label:

```

6173 \newcommand*{\glsentryaccess}[1]{%
6174   \csname glo@#1@access\endcsname
6175 }

```

`\glsentrytextaccess` Get the value of the textaccess key for the entry with the given label:

```

6176 \newcommand*{\glsentrytextaccess}[1]{%
6177   \csname glo@#1@textaccess\endcsname
6178 }

```

`\glsentryfirstaccess` Get the value of the firstaccess key for the entry with the given label:

```

6179 \newcommand*{\glsentryfirstaccess}[1]{%
6180   \csname glo@#1@firstaccess\endcsname
6181 }

```

`\glsentrypluralaccess` Get the value of the pluralaccess key for the entry with the given label:


```

6182 \newcommand*{\glsentrypluralaccess}[1]{%
6183   \csname glo@#1@pluralaccess\endcsname
6184 }

ryfirstpluralaccess  Get the value of the firstpluralaccess key for the entry with the given label:
6185 \newcommand*{\glsentryfirstpluralaccess}[1]{%
6186   \csname glo@#1@firstpluralaccess\endcsname
6187 }

lsentrysymbolaccess  Get the value of the symbolaccess key for the entry with the given label:
6188 \newcommand*{\glsentrysymbolaccess}[1]{%
6189   \csname glo@#1@symbolaccess\endcsname
6190 }

ysymbolpluralaccess  Get the value of the symbolpluralaccess key for the entry with the given label:
6191 \newcommand*{\glsentrysymbolpluralaccess}[1]{%
6192   \csname glo@#1@symbolpluralaccess\endcsname
6193 }

\glsentrydescaccess  Get the value of the descriptionaccess key for the entry with the given label:
6194 \newcommand*{\glsentrydescaccess}[1]{%
6195   \csname glo@#1@descaccess\endcsname
6196 }

trydescpluralaccess  Get the value of the descriptionpluralaccess key for the entry with the given la-
                        bel:
6197 \newcommand*{\glsentrydescpluralaccess}[1]{%
6198   \csname glo@#1@descaccess\endcsname
6199 }

glsentryshortaccess  Get the value of the shortaccess key for the entry with the given label:
6200 \newcommand*{\glsentryshortaccess}[1]{%
6201   \csname glo@#1@shortaccess\endcsname
6202 }

ryshortpluralaccess  Get the value of the shortpluralaccess key for the entry with the given label:
6203 \newcommand*{\glsentryshortpluralaccess}[1]{%
6204   \csname glo@#1@shortpluralaccess\endcsname
6205 }

\glsentrylongaccess  Get the value of the longaccess key for the entry with the given label:
6206 \newcommand*{\glsentrylongaccess}[1]{%
6207   \csname glo@#1@longaccess\endcsname
6208 }

trylongpluralaccess  Get the value of the longpluralaccess key for the entry with the given label:
6209 \newcommand*{\glsentrylongpluralaccess}[1]{%
6210   \csname glo@#1@longpluralaccess\endcsname
6211 }

```

`\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.)

```
6212 \newcommand*{\glsaccsupp}[2]{%
6213   \BeginAccSupp{ActualText=#1}#2\EndAccSupp{}%
6214 }
```

`\xglsaccsupp` Fully expands replacement text before calling `\glsaccsupp`

```
6215 \newcommand*{\xglsaccsupp}[2]{%
6216   \protected@edef\@gls@replacementtext{#1}%
6217   \expandafter\glsaccsupp\expandafter{\@gls@replacementtext}{#2}%
6218 }
```

`\glsnameaccessdisplay` Displays the first argument with the accessibility text for the entry with the label given by the second argument (if set).

```
6219 \DeclareRobustCommand*{\glsnameaccessdisplay}[2]{%
6220   \protected@edef\@glo@access{\glsentryaccess{#2}}%
6221   \ifx\@glo@access\@gls@noaccess
6222     #1%
6223   \else
6224     \xglsaccsupp{\@glo@access}{#1}%
6225   \fi
6226 }
```

`\glsentrytextaccessdisplay` As above but for the textaccess replacement text.

```
6227 \DeclareRobustCommand*{\glsentrytextaccessdisplay}[2]{%
6228   \protected@edef\@glo@access{\glsentrytextaccess{#2}}%
6229   \ifx\@glo@access\@gls@noaccess
6230     #1%
6231   \else
6232     \xglsaccsupp{\@glo@access}{#1}%
6233   \fi
6234 }
```

`\glsentrypluralaccessdisplay` As above but for the pluralaccess replacement text.

```
6235 \DeclareRobustCommand*{\glsentrypluralaccessdisplay}[2]{%
6236   \protected@edef\@glo@access{\glsentrypluralaccess{#2}}%
6237   \ifx\@glo@access\@gls@noaccess
6238     #1%
6239   \else
6240     \xglsaccsupp{\@glo@access}{#1}%
6241   \fi
6242 }
```

`\glsentryfirstaccessdisplay` As above but for the firstaccess replacement text.

```
6243 \DeclareRobustCommand*{\glsentryfirstaccessdisplay}[2]{%
6244   \protected@edef\@glo@access{\glsentryfirstaccess{#2}}%
```

```

6245 \ifx\@glo@access\@gls@noaccess
6246   #1%
6247 \else
6248   \xglsaccsupp{\@glo@access}{#1}%
6249 \fi
6250 }

```

pluralaccessdisplay As above but for the firstpluralaccess replacement text.

```

6251 \DeclareRobustCommand*\glsfirstpluralaccessdisplay}[2]{%
6252   \protected@edef\@glo@access{\glsentryfirstpluralaccess{#2}}%
6253   \ifx\@glo@access\@gls@noaccess
6254     #1%
6255   \else
6256     \xglsaccsupp{\@glo@access}{#1}%
6257   \fi
6258 }

```

symbolaccessdisplay As above but for the symbolaccess replacement text.

```

6259 \DeclareRobustCommand*\glsymbolaccessdisplay}[2]{%
6260   \protected@edef\@glo@access{\glsentrysymbolaccess{#2}}%
6261   \ifx\@glo@access\@gls@noaccess
6262     #1%
6263   \else
6264     \xglsaccsupp{\@glo@access}{#1}%
6265   \fi
6266 }

```

pluralaccessdisplay As above but for the symbolpluralaccess replacement text.

```

6267 \DeclareRobustCommand*\glsymbolpluralaccessdisplay}[2]{%
6268   \protected@edef\@glo@access{\glsentrysymbolpluralaccess{#2}}%
6269   \ifx\@glo@access\@gls@noaccess
6270     #1%
6271   \else
6272     \xglsaccsupp{\@glo@access}{#1}%
6273   \fi
6274 }

```

captionaccessdisplay As above but for the descriptionaccess replacement text.

```

6275 \DeclareRobustCommand*\glsdescriptionaccessdisplay}[2]{%
6276   \protected@edef\@glo@access{\glsentrydescaccess{#2}}%
6277   \ifx\@glo@access\@gls@noaccess
6278     #1%
6279   \else
6280     \xglsaccsupp{\@glo@access}{#1}%
6281   \fi
6282 }

```

pluralaccessdisplay As above but for the descriptionpluralaccess replacement text.

```

6283 \DeclareRobustCommand*{\glsdescriptionpluralaccessdisplay}[2]{%
6284   \protected@edef\@glo@access{\glsentrydescpluralaccess{#2}}%
6285   \ifx\@glo@access\@gls@noaccess
6286     #1%
6287   \else
6288     \xglsaccsupp{\@glo@access}{#1}%
6289   \fi
6290 }

```

sshortaccessdisplay As above but for the shortaccess replacement text.

```

6291 \DeclareRobustCommand*{\glsshortaccessdisplay}[2]{%
6292   \protected@edef\@glo@access{\glsentryshortaccess{#2}}%
6293   \ifx\@glo@access\@gls@noaccess
6294     #1%
6295   \else
6296     \xglsaccsupp{\@glo@access}{#1}%
6297   \fi
6298 }

```

pluralaccessdisplay As above but for the shortpluralaccess replacement text.

```

6299 \DeclareRobustCommand*{\glsshortpluralaccessdisplay}[2]{%
6300   \protected@edef\@glo@access{\glsentryshortpluralaccess{#2}}%
6301   \ifx\@glo@access\@gls@noaccess
6302     #1%
6303   \else
6304     \xglsaccsupp{\@glo@access}{#1}%
6305   \fi
6306 }

```

lslongaccessdisplay As above but for the longaccess replacement text.

```

6307 \DeclareRobustCommand*{\glslongaccessdisplay}[2]{%
6308   \protected@edef\@glo@access{\glsentrylongaccess{#2}}%
6309   \ifx\@glo@access\@gls@noaccess
6310     #1%
6311   \else
6312     \xglsaccsupp{\@glo@access}{#1}%
6313   \fi
6314 }

```

pluralaccessdisplay As above but for the longpluralaccess replacement text.

```

6315 \DeclareRobustCommand*{\glslongpluralaccessdisplay}[2]{%
6316   \protected@edef\@glo@access{\glsentrylongpluralaccess{#2}}%
6317   \ifx\@glo@access\@gls@noaccess
6318     #1%
6319   \else
6320     \xglsaccsupp{\@glo@access}{#1}%
6321   \fi
6322 }

```

`\glsaccessdisplay` Gets the replacement text corresponding to the named key given by the first argument and calls the appropriate command defined above.

```

6323 \DeclareRobustCommand*\glsaccessdisplay}[3]{%
6324   \ifundefined{gls#1accessdisplay}%
6325   {%
6326     \PackageError{glossaries-accsupp}{No accessibility support
6327       for key ‘#1’}{}%
6328   }%
6329   {%
6330     \csname gls#1accessdisplay\endcsname{#2}{#3}%
6331   }%
6332 }

```

`\@gls@` Redefine `\@gls@` to change the way the link text is defined

```

6333 \def\@gls@#1#2[#3]{%
6334   \glsdoifexists{#2}%
6335   {%
6336     \edef\@glo@type{\glsentrytype{#2}}%

```

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

```

6337   \def\@gls@link@opts{#1}%
6338   \def\@gls@link@label{#2}%

```

Determine what the link text should be (this is stored in `\@glo@text`). This is no longer expanded.

```

6339   \ifglsused{#2}%
6340   {%
6341     \def\@glo@text{\csname gls@\@glo@type @display\endcsname
6342       {\glsfirstaccessdisplay{\glsentryfirst{#2}}{#2}}%
6343       {\glsdescriptionaccessdisplay{\glsentrydesc{#2}}{#2}}%
6344       {\glsymbolaccessdisplay{\glsentrysymbol{#2}}{#2}}%
6345       {#3}}%
6346   }%
6347   {%
6348     \def\@glo@text{\csname gls@\@glo@type @displayfirst\endcsname
6349       {\glsfirstaccessdisplay{\glsentryfirst{#2}}{#2}}%
6350       {\glsdescriptionaccessdisplay{\glsentrydesc{#2}}{#2}}%
6351       {\glsymbolaccessdisplay{\glsentrysymbol{#2}}{#2}}%
6352       {#3}}%
6353   }%

```

Call `\@gls@link`. If footnote package option has been used, suppress hyperlink for first use.

```

6354   \ifglsused{#2}%
6355   {%
6356     \@gls@link[#1]{#2}{\@glo@text}%
6357   }%
6358   {%
6359     \gls@checkisacronymlist\@glo@type
6360     \ifthenelse{(\boolean{@glsisacronymlist})\AND

```

```

6361      \boolean{glsacrfootnote}\) \OR\NOT\boolean{glshyperfirst}}}%
6362      {%
6363      \@gls@link[#1,hyper=false]{#2}{\@glo@text}%
6364      }%
6365      {%
6366      \@gls@link[#1]{#2}{\@glo@text}%
6367      }%
6368      }%

```

Indicate that this entry has now been used

```

6369      \glsunset{#2}%
6370      }%
6371      }

```

\@Gls@

```

6372 \def\@Gls@#1#2[#3]{%
6373   \glsdoifexists{#2}%
6374   {%
6375     \edef\@glo@type{\glentrytype{#2}}%

```

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

```

6376     \def\@gls@link@opts{#1}%
6377     \def\@gls@link@label{#2}%

```

Determine what the link text should be (this is stored in \@glo@text). The first character of the entry text is converted to uppercase before passing to \@gls@<type>@display or \@gls@<type>@displayfirst

```

6378     \ifglsused{#2}%
6379     {%
6380       \def\@glo@text{\csname gls@\@glo@type @display\endcsname
6381         {\glstextaccessdisplay{\Glsentrytext{#2}}{#2}}}%
6382       {\glsdescriptionaccessdisplay{\glentrydesc{#2}}{#2}}}%
6383       {\glssymbolaccessdisplay{\glentrysymbol{#2}}{#2}}}%
6384       {#3}}%
6385     }%
6386     {%
6387       \def\@glo@text{\csname gls@\@glo@type @displayfirst\endcsname
6388         {\glsfirstaccessdisplay{\Glsentryfirst{#2}}{#2}}}%
6389       {\glsdescriptionaccessdisplay{\glentrydesc{#2}}{#2}}}%
6390       {\glssymbolaccessdisplay{\glentrysymbol{#2}}{#2}}}%
6391       {#3}}%
6392     }%

```

Call \@gls@link. If footnote package option has been used, suppress hyperlink for first use.

```

6393     \ifglsused{#2}%
6394     {%
6395       \@gls@link[#1]{#2}{\@glo@text}%
6396     }%
6397     {%

```

```

6398 \gls@checkisacronymlist\@glo@type
6399 \ifthenelse{\( \boolean{glsisacronymlist} \) AND
6400 \boolean{glsacrfootnote} \) \OR \NOT \boolean{glshyperfirst} }%
6401 {%
6402 \@gls@link[#1,hyper=false]{#2}{\@glo@text}%
6403 }%
6404 {%
6405 \@gls@link[#1]{#2}{\@glo@text}%
6406 }%
6407 }%

```

Indicate that this entry has now been used

```

6408 \glsunset{#2}%
6409 }%
6410 }

```

\@GLS@

```

6411 \def\@GLS@#1#2[#3]{%
6412 \glsdoifexists{#2}{%
6413 \edef\@glo@type{\glsentrytype{#2}}%

```

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

```

6414 \def\@gls@link@opts{#1}%
6415 \def\@gls@link@label{#2}%

```

Determine what the link text should be (this is stored in \@glo@text).

```

6416 \ifglsused{#2}%
6417 {%
6418 \def\@glo@text{\csname gls@\@glo@type @display\endcsname
6419 {\gls@textaccessdisplay{\glsentrytext{#2}}{#2}}%
6420 {\gls@descriptionaccessdisplay{\glsentrydesc{#2}}{#2}}%
6421 {\gls@symbolaccessdisplay{\glsentrysymbol{#2}}{#2}}%
6422 {#3}}%
6423 }%
6424 {%
6425 \edef\@glo@text{\csname gls@\@glo@type @displayfirst\endcsname
6426 {\gls@firstaccessdisplay{\glsentryfirst{#2}}{#2}}%
6427 {\gls@descriptionaccessdisplay{\glsentrydesc{#2}}{#2}}%
6428 {\gls@symbolaccessdisplay{\glsentrysymbol{#2}}{#2}}%
6429 {#3}}%
6430 }%

```

Call \@gls@link If footnote package option has been used, suppress hyperlink for first use.

```

6431 \ifglsused{#2}%
6432 {%
6433 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%
6434 }%
6435 {%
6436 \gls@checkisacronymlist\@glo@type
6437 \ifthenelse{\( \boolean{glsisacronymlist} \) AND

```

```

6438      \boolean{glsacrfootnote}\) \OR\NOT\boolean{glshyperfirst}}{%
6439      \@gls@link[#1,hyper=false]{#2}{\MakeUppercase{\@glo@text}}%
6440      }%
6441      {%
6442      \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%
6443      }%
6444      }%

```

Indicate that this entry has now been used

```

6445      \glsunset{#2}%
6446      }%
6447      }

```

\@gls@pl@

```

6448 \def\@glspl@#1#2[#3]{%
6449   \glsdoifexists{#2}%
6450   {%
6451     \edef\@glo@type{\glsentrytype{#2}}%

```

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

```

6452   \def\@gls@link@opts{#1}%
6453   \def\@gls@link@label{#2}%

```

Determine what the link text should be (this is stored in \@glo@text)

```

6454   \ifglsused{#2}%
6455   {%
6456     \def\@glo@text{\csname gls@\@glo@type @display\endcsname
6457       {\glspluralaccessdisplay{\glsentryplural{#2}}{#2}}%
6458       {\glsdescriptionpluralaccessdisplay{\glsentrydescplural{#2}}{#2}}%
6459       {\glsymbolpluralaccessdisplay{\glsentrysymbolplural{#2}}{#2}}%
6460       {#3}}%
6461   }%
6462   {%
6463     \def\@glo@text{\csname gls@\@glo@type @displayfirst\endcsname
6464       {\glsfirstpluralaccessdisplay{\glsentryfirstplural{#2}}{#2}}%
6465       {\glsdescriptionpluralaccessdisplay{\glsentrydescplural{#2}}{#2}}%
6466       {\glsymbolpluralaccessdisplay{\glsentrysymbolplural{#2}}{#2}}%
6467       {#3}}%
6468   }%

```

Call \@gls@link If footnote package option has been used, suppress hyperlink for first use.

```

6469   \ifglsused{#2}%
6470   {%
6471     \@gls@link[#1]{#2}{\@glo@text}%
6472   }%
6473   {%
6474     \gls@checkisacronymlist\@glo@type
6475     \ifthenelse{(\boolean{glsisacronymlist})\AND
6476       \boolean{glsacrfootnote}\) \OR\NOT\boolean{glshyperfirst}}%
6477     {%

```



```

6478      \@gls@link[#1,hyper=false]{#2}{\@glo@text}%
6479      }%
6480      {%
6481      \@gls@link[#1]{#2}{\@glo@text}%
6482      }%
6483      }%

```

Indicate that this entry has now been used

```

6484      \glsunset{#2}%
6485      }%
6486      }

```

\@Glspl@

```

6487 \def\@Glspl@#1#2[#3]{%
6488   \glsdoifexists{#2}%
6489   {%
6490     \edef\@glo@type{\glsentrytype{#2}}%

```

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

```

6491   \def\@gls@link@opts{#1}%
6492   \def\@gls@link@label{#2}%

```

Determine what the link text should be (this is stored in \@glo@text).

```

6493   \ifglsused{#2}%
6494   {%
6495     \def\@glo@text{\csname gls@\@glo@type @display\endcsname
6496       {\glspluralaccessdisplay{\Glsentryplural{#2}}{#2}}%
6497       {\glsdescriptionpluralaccessdisplay{\glsentrydescplural{#2}}{#2}}%
6498       {\glssymbolpluralaccessdisplay{\glsentrysymbolplural{#2}}{#2}}%
6499       {#3}}%
6500   }%
6501   {%
6502     \def\@glo@text{\csname gls@\@glo@type @displayfirst\endcsname
6503       {\glsfirstpluralaccessdisplay{\Glsentryfirstplural{#2}}{#2}}%
6504       {\glsdescriptionpluralaccessdisplay{\glsentrydescplural{#2}}{#2}}%
6505       {\glssymbolpluralaccessdisplay{\glsentrysymbolplural{#2}}{#2}}%
6506       {#3}}%
6507   }%

```

Call \@gls@link If footnote package option has been used, suppress hyperlink for first use.

```

6508   \ifglsused{#2}%
6509   {%
6510     \@gls@link[#1]{#2}{\@glo@text}%
6511   }%
6512   {%
6513     \ifthenelse{\equal{\@glo@type}{\acronymtype}\and
6514       \boolean{glsacrfootnote}}%
6515     {%
6516       \@gls@link[#1,hyper=false]{#2}{\@glo@text}%
6517     }%

```

```

6518      {%
6519      \@gls@link[#1]{#2}{\@glo@text}%
6520      }%
6521      }%

```

Indicate that this entry has now been used

```

6522      \glsunset{#2}%
6523      }%
6524      }

```

\@GLSp1@

```

6525 \def\@GLSp1@#1#2[#3]{%
6526   \glsdoifexists{#2}%
6527   {%
6528     \edef\@glo@type{\glsentrytype{#2}}%

```

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

```

6529   \def\@gls@link@opts{#1}%
6530   \def\@gls@link@label{#2}%

```

Determine what the link text should be (this is stored in \@glo@text)

```

6531   \ifglsused{#2}%
6532   {%
6533     \def\@glo@text{\csname gls@\@glo@type @display\endcsname
6534       {\glspluralaccessdisplay{\glsentryplural{#2}}{#2}}%
6535       {\glsdescriptionpluralaccessdisplay{\glsentrydescplural{#2}}{#2}}%
6536       {\glsymbolpluralaccessdisplay{\glsentrysymbolplural{#2}}{#2}}%
6537       {#3}}%
6538   }%
6539   {%
6540     \def\@glo@text{\csname gls@\@glo@type @displayfirst\endcsname
6541       {\glsfirstpluralaccessdisplay{\glsentryfirstplural{#2}}{#2}}%
6542       {\glsdescriptionpluralaccessdisplay{\glsentrydescplural{#2}}{#2}}%
6543       {\glsymbolpluralaccessdisplay{\glsentrysymbolplural{#2}}{#2}}%
6544       {#3}}%
6545   }%

```

Call \@gls@link If footnote package option has been used, suppress hyperlink for first use.

```

6546   \ifglsused{#2}%
6547   {%
6548     \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%
6549   }%
6550   {%
6551     \gls@checkisacronymlist\@glo@type
6552     \ifthenelse{(\boolean{glsisacronymlist}\AND
6553       \boolean{glsacrfootnote})\OR\not\boolean{glshyperfirst}}%
6554     {%
6555       \@gls@link[#1,hyper=false]{#2}{\MakeUppercase{\@glo@text}}%
6556     }%
6557     {%

```

```

6558      \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%
6559    }%
6560  }%

```

Indicate that this entry has now been used

```

6561    \glsunset{#2}%
6562  }%
6563 }

```

\@acrshort

```

6564 \def\@acrshort#1#2[#3]{%
6565   \glsdoifexists{#2}%
6566   {%
6567     \edef\@glo@type{\glsentrytype{#2}}%

```

Determine what the link text should be (this is stored in \@glo@text)

```

6568   \def\@glo@text{%
6569     \glsshortaccessdisplay{\glsentryshort{#2}}{#2}%
6570   }%

```

Call \@gls@link

```

6571     \@gls@link[#1]{#2}{\acronymfont{\@glo@text}#3}%
6572   }%
6573 }

```

\@Acrshort

```

6574 \def\@Acrshort#1#2[#3]{%
6575   \glsdoifexists{#2}%
6576   {%
6577     \edef\@glo@type{\glsentrytype{#2}}%

```

Determine what the link text should be (this is stored in \@glo@text)

```

6578   \def\@glo@text{%
6579     \glsshortaccessdisplay{\Glsentryshort{#2}}{#2}%
6580   }%

```

Call \@gls@link

```

6581     \@gls@link[#1]{#2}{\acronymfont{\@glo@text}#3}%
6582   }%
6583 }

```

\@ACRshort

```

6584 \def\@ACRshort#1#2[#3]{%
6585   \glsdoifexists{#2}%
6586   {%
6587     \edef\@glo@type{\glsentrytype{#2}}%

```

Determine what the link text should be (this is stored in \@glo@text)

```

6588   \def\@glo@text{%
6589     \glsshortaccessdisplay{\MakeUppercase{\glsentryshort{#2}}}{#2}%
6590   }%

```

```

Call \@gls@link
6591   \@gls@link[#1]{#2}{\acronymfont{\@glo@text#3}}%
6592   }%
6593 }

```

\@acrlong

```

6594 \def\@acrlong#1#2[#3]{%
6595   \glsdoifexists{#2}%
6596   {%
6597     \edef\@glo@type{\glsentrytype{#2}}%
        Determine what the link text should be (this is stored in \@glo@text)
6598     \def\@glo@text{%
6599       \glslongaccessdisplay{\glsentrylong{#2}}{#2}%
6600     }%
        Call \@gls@link
6601     \@gls@link[#1]{#2}{\@glo@text#3}%
6602     }%
6603 }

```

\@Acrlong

```

6604 \def\@Acrlong#1#2[#3]{%
6605   \glsdoifexists{#2}%
6606   {%
6607     \edef\@glo@type{\glsentrytype{#2}}%
        Determine what the link text should be (this is stored in \@glo@text)
6608     \def\@glo@text{%
6609       \glslongaccessdisplay{\Glsentrylong{#2}}{#2}%
6610     }%
        Call \@gls@link
6611     \@gls@link[#1]{#2}{\@glo@text#3}%
6612     }%
6613 }

```

\@ACRlong

```

6614 \def\@ACRlong#1#2[#3]{%
6615   \glsdoifexists{#2}%
6616   {%
6617     \edef\@glo@type{\glsentrytype{#2}}%
        Determine what the link text should be (this is stored in \@glo@text)
6618     \def\@glo@text{%
6619       \glslongaccessdisplay{\MakeUppercase{\glsentrylong{#2}}}{#2}%
6620     }%
        Call \@gls@link
6621     \@gls@link[#1]{#2}{\@glo@text#3}%
6622     }%
6623 }

```

5.3 Displaying the Glossary

Entries within the glossary or list of acronyms are now formatted via `\accsuppglossaryentryfield` and `\accsuppglossarysubentryfield`.

@glossaryentryfield

```
6624 \ifglxindy
6625   \renewcommand*{\@glossaryentryfield}{%
6626     \string\accsuppglossaryentryfield}
6627 \else
6628   \renewcommand*{\@glossaryentryfield}{%
6629     \string\accsuppglossaryentryfield}
6630 \fi
```

ossarysubentryfield

```
6631 \ifglxindy
6632   \renewcommand*{\@glossarysubentryfield}{%
6633     \string\accsuppglossarysubentryfield}
6634 \else
6635   \renewcommand*{\@glossarysubentryfield}{%
6636     \string\accsuppglossarysubentryfield}
6637 \fi
```

pglossaryentryfield

```
6638 \newcommand*{\accsuppglossaryentryfield}[5]{%
6639   \glossaryentryfield{#1}%
6640   {\glxnameaccessdisplay{#2}{#1}}%
6641   {\glxdescriptionaccessdisplay{#3}{#1}}%
6642   {\glxsymbolaccessdisplay{#4}{#1}}{#5}%
6643 }
```

ossarysubentryfield

```
6644 \newcommand*{\accsuppglossarysubentryfield}[6]{%
6645   \glossaryentryfield{#1}{#2}%
6646   {\glxnameaccessdisplay{#3}{#2}}%
6647   {\glxdescriptionaccessdisplay{#4}{#2}}%
6648   {\glxsymbolaccessdisplay{#5}{#2}}{#6}%
6649 }
```

5.4 Acronyms

Use `\newacronymhook` to modify the key list to set the access text to the long version by default.

```
6650 \renewcommand*{\newacronymhook}{%
6651   \edef\@gls@keylist{shortaccess=\the\glxlongtok,%
6652     \the\glxkeylisttok}%
6653   \expandafter\glxkeylisttok\expandafter{\@gls@keylist}%
6654 }
```

efaultNewAcronymDef Modify default style to use access text:

```

6655 \renewcommand*{\DefaultNewAcronymDef}{%
6656   \edef\@do@newglossaryentry{%
6657     \noexpand\newglossaryentry{\the\glslabeltok}%
6658     {%
6659       type=\acronymtype,%
6660       name={\the\glsshorttok},%
6661       description={\the\glslongtok},%
6662       descriptionaccess=\relax,
6663       text={\the\glsshorttok},%
6664       access={\noexpand\@glo@textaccess},%
6665       sort={\the\glsshorttok},%
6666       short={\the\glsshorttok},%
6667       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6668       shortaccess={\the\glslongtok},%
6669       long={\the\glslongtok},%
6670       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6671       descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6672       first={\noexpand\glslongaccessdisplay
6673         {\the\glslongtok}{\the\glslabeltok}\space
6674         (\noexpand\glsshortaccessdisplay
6675         {\the\glsshorttok}{\the\glslabeltok})},%
6676       plural={\the\glsshorttok\acrpluralsuffix},%
6677       firstplural={\noexpand\glslongpluralaccessdisplay
6678         {\noexpand\@glo@longpl}{\the\glslabeltok}\space
6679         (\noexpand\glsshortpluralaccessdisplay
6680         {\noexpand\@glo@shortpl}{\the\glslabeltok})},%
6681       firstaccess=\relax,
6682       firstpluralaccess=\relax,
6683       textaccess={\noexpand\@glo@shortaccess},%
6684       \the\glskeylisttok
6685     }%
6686   }%
6687   \@do@newglossaryentry
6688 }

```

otnoteNewAcronymDef

```

6689 \renewcommand*{\DescriptionFootnoteNewAcronymDef}{%
6690   \edef\@do@newglossaryentry{%
6691     \noexpand\newglossaryentry{\the\glslabeltok}%
6692     {%
6693       type=\acronymtype,%
6694       name={\noexpand\acronymfont{\the\glsshorttok}},%
6695       sort={\the\glsshorttok},%
6696       text={\the\glsshorttok},%
6697       short={\the\glsshorttok},%
6698       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6699       shortaccess={\the\glslongtok},%
6700       long={\the\glslongtok},%

```

```

6701     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6702     access={\noexpand\@glo@textaccess},%
6703     plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6704     symbol={\the\glslongtok},%
6705     symbolplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6706     firstpluralaccess=\relax,
6707     textaccess={\noexpand\@glo@shortaccess},%
6708     \the\glskeylisttok
6709   }%
6710 }%
6711 \@do@newglossaryentry
6712 }

```

ptionNewAcronymDef

```

6713 \renewcommand*{\DescriptionNewAcronymDef}{%
6714   \edef\@do@newglossaryentry{%
6715     \noexpand\newglossaryentry{\the\glslabeltok}%
6716     {%
6717       type=\acronymtype,%
6718       name={\noexpand
6719         \acrnameformat{\the\glsshorttok}{\the\glslongtok}},%
6720       access={\noexpand\@glo@textaccess},%
6721       sort={\the\glsshorttok},%
6722       short={\the\glsshorttok},%
6723       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6724       shortaccess={\the\glslongtok},%
6725       long={\the\glslongtok},%
6726       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6727       first={\the\glslongtok},%
6728       firstaccess=\relax,
6729       firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6730       text={\the\glsshorttok},%
6731       textaccess={\the\glslongtok},%
6732       plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6733       symbol={\noexpand\@glo@text},%
6734       symbolaccess={\noexpand\@glo@textaccess},%
6735       symbolplural={\noexpand\@glo@plural},%
6736       firstpluralaccess=\relax,
6737       textaccess={\noexpand\@glo@shortaccess},%
6738       \the\glskeylisttok}%
6739   }%
6740   \@do@newglossaryentry
6741 }

```

otnoteNewAcronymDef

```

6742 \renewcommand*{\FootnoteNewAcronymDef}{%
6743   \edef\@do@newglossaryentry{%
6744     \noexpand\newglossaryentry{\the\glslabeltok}%
6745     {%

```

```

6746     type=\acronymtype,%
6747     name={\noexpand\acronymfont{\the\glsshorttok}},%
6748     sort={\the\glsshorttok},%
6749     text={\the\glsshorttok},%
6750     textaccess={\the\glslongtok},%
6751     access={\noexpand\@glo@textaccess},%
6752     plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6753     short={\the\glsshorttok},%
6754     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6755     long={\the\glslongtok},%
6756     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6757     description={\the\glslongtok},%
6758     descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6759     \the\glskeylisttok
6760 }%
6761 }%
6762 \@do@newglossaryentry
6763 }

```

\SmallNewAcronymDef

```

6764 \renewcommand*{\SmallNewAcronymDef}{%
6765   \edef\@do@newglossaryentry{%
6766     \noexpand\newglossaryentry{\the\glslabeltok}%
6767     {%
6768       type=\acronymtype,%
6769       name={\noexpand\acronymfont{\the\glsshorttok}},%
6770       access={\noexpand\@glo@symbolaccess},%
6771       sort={\the\glsshorttok},%
6772       short={\the\glsshorttok},%
6773       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6774       shortaccess={\the\glslongtok},%
6775       long={\the\glslongtok},%
6776       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6777       text={\noexpand\@glo@short},%
6778       textaccess={\noexpand\@glo@shortaccess},%
6779       plural={\noexpand\@glo@shortpl},%
6780       first={\the\glslongtok},%
6781       firstaccess=\relax,
6782       firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6783       description={\noexpand\@glo@first},%
6784       descriptionplural={\noexpand\@glo@firstplural},%
6785       symbol={\the\glsshorttok},%
6786       symbolaccess={\the\glslongtok},%
6787       symbolplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6788       \the\glskeylisttok
6789     }%
6790   }%
6791   \@do@newglossaryentry
6792 }

```


The following are kept for compatibility with versions before 3.0:

```
\glsshortaccesskey
6793 \newcommand*{\glsshortaccesskey}{\glsshortkey access}%

shortpluralaccesskey
6794 \newcommand*{\glsshortpluralaccesskey}{\glsshortpluralkey access}%

\glslongaccesskey
6795 \newcommand*{\glslongaccesskey}{\glslongkey access}%

longpluralaccesskey
6796 \newcommand*{\glslongpluralaccesskey}{\glslongpluralkey access}%
```

5.5 Debugging Commands

```
\showglonameaccess
6797 \newcommand*{\showglonameaccess}[1]{%
6798 \expandafter\show\csname glo@#1@textaccess\endcsname
6799 }

\showglotextaccess
6800 \newcommand*{\showglotextaccess}[1]{%
6801 \expandafter\show\csname glo@#1@textaccess\endcsname
6802 }

showglopluralaccess
6803 \newcommand*{\showglopluralaccess}[1]{%
6804 \expandafter\show\csname glo@#1@pluralaccess\endcsname
6805 }

\showglofirstaccess
6806 \newcommand*{\showglofirstaccess}[1]{%
6807 \expandafter\show\csname glo@#1@firstaccess\endcsname
6808 }

lofirstpluralaccess
6809 \newcommand*{\showglofirstpluralaccess}[1]{%
6810 \expandafter\show\csname glo@#1@firstpluralaccess\endcsname
6811 }

showglosymbolaccess
6812 \newcommand*{\showglosymbolaccess}[1]{%
6813 \expandafter\show\csname glo@#1@symbolaccess\endcsname
6814 }
```

osymbolpluralaccess

```
6815 \newcommand*{\showglosymbolpluralaccess}[1]{%
6816   \expandafter\show\csname glo@#1@symbolpluralaccess\endcsname
6817 }
```

\showglodescaccess

```
6818 \newcommand*{\showglodescaccess}[1]{%
6819   \expandafter\show\csname glo@#1@descaccess\endcsname
6820 }
```

glodescpluralaccess

```
6821 \newcommand*{\showglodescpluralaccess}[1]{%
6822   \expandafter\show\csname glo@#1@descpluralaccess\endcsname
6823 }
```

\showgloshortaccess

```
6824 \newcommand*{\showgloshortaccess}[1]{%
6825   \expandafter\show\csname glo@#1@shortaccess\endcsname
6826 }
```

loshortpluralaccess

```
6827 \newcommand*{\showgloshortpluralaccess}[1]{%
6828   \expandafter\show\csname glo@#1@shortpluralaccess\endcsname
6829 }
```

\showglolongaccess

```
6830 \newcommand*{\showglolongaccess}[1]{%
6831   \expandafter\show\csname glo@#1@longaccess\endcsname
6832 }
```

glolongpluralaccess

```
6833 \newcommand*{\showglolongpluralaccess}[1]{%
6834   \expandafter\show\csname glo@#1@longpluralaccess\endcsname
6835 }
```

6 Multi-Lingual Support

Many thanks to everyone who contributed to the translations both via email and on comp.text.tex.

6.1 Babel Captions

Define captions if multi-lingual support is required, but the package is not loaded.

```
6836 \NeedsTeXFormat{LaTeX2e}
6837 \ProvidesPackage{glossaries-babel}[2009/04/16 v1.2 (NLCT)]
```

English:

```
6838 \@ifundefined{captionsenglish}{%}{%
6839   \addto\captionsenglish{%
6840     \renewcommand*{\glossaryname}{Glossary}%
6841     \renewcommand*{\acronymname}{Acronyms}%
6842     \renewcommand*{\entryname}{Notation}%
6843     \renewcommand*{\descriptionname}{Description}%
6844     \renewcommand*{\symbolname}{Symbol}%
6845     \renewcommand*{\pagelistname}{Page List}%
6846     \renewcommand*{\glssymbolsgroupname}{Symbols}%
6847     \renewcommand*{\glsnumbersgroupname}{Numbers}%
6848 }%
6849 }
6850 \@ifundefined{captionsamerican}{%}{%
6851   \addto\captionsamerican{%
6852     \renewcommand*{\glossaryname}{Glossary}%
6853     \renewcommand*{\acronymname}{Acronyms}%
6854     \renewcommand*{\entryname}{Notation}%
6855     \renewcommand*{\descriptionname}{Description}%
6856     \renewcommand*{\symbolname}{Symbol}%
6857     \renewcommand*{\pagelistname}{Page List}%
6858     \renewcommand*{\glssymbolsgroupname}{Symbols}%
6859     \renewcommand*{\glsnumbersgroupname}{Numbers}%
6860 }%
6861 }
6862 \@ifundefined{captionsaustralian}{%}{%
6863   \addto\captionsaustralian{%
6864     \renewcommand*{\glossaryname}{Glossary}%
6865     \renewcommand*{\acronymname}{Acronyms}%
6866     \renewcommand*{\entryname}{Notation}%
6867     \renewcommand*{\descriptionname}{Description}%
6868     \renewcommand*{\symbolname}{Symbol}%
6869     \renewcommand*{\pagelistname}{Page List}%
6870     \renewcommand*{\glssymbolsgroupname}{Symbols}%
6871     \renewcommand*{\glsnumbersgroupname}{Numbers}%
6872 }%
6873 }
6874 \@ifundefined{captionsbritish}{%}{%
6875   \addto\captionsbritish{%
6876     \renewcommand*{\glossaryname}{Glossary}%
6877     \renewcommand*{\acronymname}{Acronyms}%
6878     \renewcommand*{\entryname}{Notation}%
6879     \renewcommand*{\descriptionname}{Description}%
6880     \renewcommand*{\symbolname}{Symbol}%
6881     \renewcommand*{\pagelistname}{Page List}%
6882     \renewcommand*{\glssymbolsgroupname}{Symbols}%
6883     \renewcommand*{\glsnumbersgroupname}{Numbers}%
6884 }}%
6885 \@ifundefined{captionscanadian}{%}{%
```

```

6886 \addto\captionscanadian{%
6887 \renewcommand*{\glossaryname}{Glossary}%
6888 \renewcommand*{\acronymname}{Acronyms}%
6889 \renewcommand*{\entryname}{Notation}%
6890 \renewcommand*{\descriptionname}{Description}%
6891 \renewcommand*{\symbolname}{Symbol}%
6892 \renewcommand*{\pagelistname}{Page List}%
6893 \renewcommand*{\glssymbolsgroupname}{Symbols}%
6894 \renewcommand*{\glsnumbersgroupname}{Numbers}%
6895 }%
6896 }
6897 \@ifundefined{captionsnewzealand}{}{%
6898 \addto\captionscanadian{%
6899 \renewcommand*{\glossaryname}{Glossary}%
6900 \renewcommand*{\acronymname}{Acronyms}%
6901 \renewcommand*{\entryname}{Notation}%
6902 \renewcommand*{\descriptionname}{Description}%
6903 \renewcommand*{\symbolname}{Symbol}%
6904 \renewcommand*{\pagelistname}{Page List}%
6905 \renewcommand*{\glssymbolsgroupname}{Symbols}%
6906 \renewcommand*{\glsnumbersgroupname}{Numbers}%
6907 }%
6908 }
6909 \@ifundefined{captionsUKenglish}{}{%
6910 \addto\captionscanadian{%
6911 \renewcommand*{\glossaryname}{Glossary}%
6912 \renewcommand*{\acronymname}{Acronyms}%
6913 \renewcommand*{\entryname}{Notation}%
6914 \renewcommand*{\descriptionname}{Description}%
6915 \renewcommand*{\symbolname}{Symbol}%
6916 \renewcommand*{\pagelistname}{Page List}%
6917 \renewcommand*{\glssymbolsgroupname}{Symbols}%
6918 \renewcommand*{\glsnumbersgroupname}{Numbers}%
6919 }%
6920 }
6921 \@ifundefined{captionsUSenglish}{}{%
6922 \addto\captionscanadian{%
6923 \renewcommand*{\glossaryname}{Glossary}%
6924 \renewcommand*{\acronymname}{Acronyms}%
6925 \renewcommand*{\entryname}{Notation}%
6926 \renewcommand*{\descriptionname}{Description}%
6927 \renewcommand*{\symbolname}{Symbol}%
6928 \renewcommand*{\pagelistname}{Page List}%
6929 \renewcommand*{\glssymbolsgroupname}{Symbols}%
6930 \renewcommand*{\glsnumbersgroupname}{Numbers}%
6931 }%
6932 }

```

German (quite a few variations were suggested for German; I settled on the following):

```

6933 \@ifundefined{captionsgerman}{%
6934   \addto\captionsgerman{%
6935     \renewcommand*{\glossaryname}{Glossar}%
6936     \renewcommand*{\acronymname}{Akronyme}%
6937     \renewcommand*{\entryname}{Bezeichnung}%
6938     \renewcommand*{\descriptionname}{Beschreibung}%
6939     \renewcommand*{\symbolname}{Symbol}%
6940     \renewcommand*{\pagelistname}{Seiten}%
6941     \renewcommand*{\glssymbolsgroupname}{Symbole}%
6942     \renewcommand*{\glsnumbersgroupname}{Zahlen}}
6943 }

```

ngerman is identical to German:

```

6944 \@ifundefined{captionsgerman}{%
6945   \addto\captionsgerman{%
6946     \renewcommand*{\glossaryname}{Glossar}%
6947     \renewcommand*{\acronymname}{Akronyme}%
6948     \renewcommand*{\entryname}{Bezeichnung}%
6949     \renewcommand*{\descriptionname}{Beschreibung}%
6950     \renewcommand*{\symbolname}{Symbol}%
6951     \renewcommand*{\pagelistname}{Seiten}%
6952     \renewcommand*{\glssymbolsgroupname}{Symbole}%
6953     \renewcommand*{\glsnumbersgroupname}{Zahlen}}
6954 }

```

Italian:

```

6955 \@ifundefined{captionssitalian}{%
6956   \addto\captionssitalian{%
6957     \renewcommand*{\glossaryname}{Glossario}%
6958     \renewcommand*{\acronymname}{Acronimi}%
6959     \renewcommand*{\entryname}{Nomenclatura}%
6960     \renewcommand*{\descriptionname}{Descrizione}%
6961     \renewcommand*{\symbolname}{Simbolo}%
6962     \renewcommand*{\pagelistname}{Elenco delle pagine}%
6963     \renewcommand*{\glssymbolsgroupname}{Simboli}%
6964     \renewcommand*{\glsnumbersgroupname}{Numeri}}
6965 }

```

Dutch:

```

6966 \@ifundefined{captionsdutch}{%
6967   \addto\captionsdutch{%
6968     \renewcommand*{\glossaryname}{Woordenlijst}%
6969     \renewcommand*{\acronymname}{Acroniemen}%
6970     \renewcommand*{\entryname}{Benaming}%
6971     \renewcommand*{\descriptionname}{Beschrijving}%
6972     \renewcommand*{\symbolname}{Symbool}%
6973     \renewcommand*{\pagelistname}{Pagina's}%
6974     \renewcommand*{\glssymbolsgroupname}{Symbolen}%

```

```

6975 \renewcommand*{\glsnumbersgroupname}{Cijfers}}
6976 }

Spanish:

6977 \@ifundefined{captionsspanish}{}{%
6978 \addto\captionsspanish{%
6979 \renewcommand*{\glossaryname}{Glosario}%
6980 \renewcommand*{\acronymname}{Siglas}%
6981 \renewcommand*{\entryname}{Entrada}%
6982 \renewcommand*{\descriptionname}{Descripci\`on}%
6983 \renewcommand*{\symbolname}{S\`{i}mbolo}%
6984 \renewcommand*{\pagelistname}{Lista de p\`aginas}%
6985 \renewcommand*{\glssymbolsgroupname}{S\`{i}mbolos}%
6986 \renewcommand*{\glsnumbersgroupname}{N\`umeros}}
6987 }

French:

6988 \@ifundefined{captionsfrench}{}{%
6989 \addto\captionsfrench{%
6990 \renewcommand*{\glossaryname}{Glossaire}%
6991 \renewcommand*{\acronymname}{Acronymes}%
6992 \renewcommand*{\entryname}{Terme}%
6993 \renewcommand*{\descriptionname}{Description}%
6994 \renewcommand*{\symbolname}{Symbole}%
6995 \renewcommand*{\pagelistname}{Pages}%
6996 \renewcommand*{\glssymbolsgroupname}{Symboles}%
6997 \renewcommand*{\glsnumbersgroupname}{Nombres}}
6998 }

6999 \@ifundefined{captionsfrenchb}{}{%
7000 \addto\captionsfrenchb{%
7001 \renewcommand*{\glossaryname}{Glossaire}%
7002 \renewcommand*{\acronymname}{Acronymes}%
7003 \renewcommand*{\entryname}{Terme}%
7004 \renewcommand*{\descriptionname}{Description}%
7005 \renewcommand*{\symbolname}{Symbole}%
7006 \renewcommand*{\pagelistname}{Pages}%
7007 \renewcommand*{\glssymbolsgroupname}{Symboles}%
7008 \renewcommand*{\glsnumbersgroupname}{Nombres}}
7009 }

7010 \@ifundefined{captionsfrancais}{}{%
7011 \addto\captionsfrancais{%
7012 \renewcommand*{\glossaryname}{Glossaire}%
7013 \renewcommand*{\acronymname}{Acronymes}%
7014 \renewcommand*{\entryname}{Terme}%
7015 \renewcommand*{\descriptionname}{Description}%
7016 \renewcommand*{\symbolname}{Symbole}%
7017 \renewcommand*{\pagelistname}{Pages}%
7018 \renewcommand*{\glssymbolsgroupname}{Symboles}%
7019 \renewcommand*{\glsnumbersgroupname}{Nombres}}
7020 }

```

Danish:

```
7021 \@ifundefined{captionsdanish}{%{}%
7022   \addto\captionsdanish{%
7023     \renewcommand*{\glossaryname}{Ordliste}%
7024     \renewcommand*{\acronymname}{Akronymer}%
7025     \renewcommand*{\entryname}{Symbolforklaring}%
7026     \renewcommand*{\descriptionname}{Beskrivelse}%
7027     \renewcommand*{\symbolname}{Symbol}%
7028     \renewcommand*{\pagelistname}{Side}%
7029     \renewcommand*{\glssymbolsgroupname}{Symboler}%
7030     \renewcommand*{\glsnumbersgroupname}{Tal}}
7031 }
```

Irish:

```
7032 \@ifundefined{captionsirish}{%{}%
7033   \addto\captionsirish{%
7034     \renewcommand*{\glossaryname}{Gluais}%
7035     \renewcommand*{\acronymname}{Acrainmneacha}%
```

wasn't sure whether to go for Nóta (Note), Ciall ('Meaning', 'sense') or Brí ('Meaning'). In the end I chose Ciall.

```
7036     \renewcommand*{\entryname}{Ciall}%
7037     \renewcommand*{\descriptionname}{Tuairisc}%
```

Again, not sure whether to use Comhartha/Comharthaí or Siombail/Siombaile, so have chosen the former.

```
7038     \renewcommand*{\symbolname}{Comhartha}%
7039     \renewcommand*{\glssymbolsgroupname}{Comhartha\'}{\i}}%
7040     \renewcommand*{\pagelistname}{Leathanaigh}%
7041     \renewcommand*{\glsnumbersgroupname}{Uimhreacha}}
7042 }
```

Hungarian:

```
7043 \@ifundefined{captionsmagyar}{%{}%
7044   \addto\captionsmagyar{%
7045     \renewcommand*{\glossaryname}{Sz\'ojegyz\'ek}%
7046     \renewcommand*{\acronymname}{Bet\H uszavak}%
7047     \renewcommand*{\entryname}{Kifejez\'es}%
7048     \renewcommand*{\descriptionname}{Magyar\'azat}%
7049     \renewcommand*{\symbolname}{Jel\'ol\'es}%
7050     \renewcommand*{\pagelistname}{Oldalsz\'am}%
7051     \renewcommand*{\glssymbolsgroupname}{Jelek}%
7052     \renewcommand*{\glsnumbersgroupname}{Sz\'amjegyek}%
7053   }
7054 }
7055 \@ifundefined{captionshungarian}{%{}%
7056   \addto\captionshungarian{%
7057     \renewcommand*{\glossaryname}{Sz\'ojegyz\'ek}%
7058     \renewcommand*{\acronymname}{Bet\H uszavak}%
7059     \renewcommand*{\entryname}{Kifejez\'es}%
7060     \renewcommand*{\descriptionname}{Magyar\'azat}%
```

```

7061 \renewcommand*{\symbolname}{Jel\`ol\`es}%
7062 \renewcommand*{\pagelistname}{Oldalsz\`am}%
7063 \renewcommand*{\glssymbolsgroupname}{Jelek}%
7064 \renewcommand*{\glsnumbersgroupname}{Sz\`amjegyek}%
7065 }
7066 }

```

Polish

```

7067 \@ifundefined{captionspolish}{}{%
7068 \addto\captionspolish{%
7069 \renewcommand*{\glossaryname}{S\lownik termin\`ow}%
7070 \renewcommand*{\acronymname}{Skr\`ot}%
7071 \renewcommand*{\entryname}{Termin}%
7072 \renewcommand*{\descriptionname}{Opis}%
7073 \renewcommand*{\symbolname}{Symbol}%
7074 \renewcommand*{\pagelistname}{Strony}%
7075 \renewcommand*{\glssymbolsgroupname}{Symbole}%
7076 \renewcommand*{\glsnumbersgroupname}{Liczby}}
7077 }

```

Brazilian

```

7078 \@ifundefined{captionsbrazil}{}{%
7079 \addto\captionsbrazil{%
7080 \renewcommand*{\glossaryname}{Gloss\`ario}%
7081 \renewcommand*{\acronymname}{Siglas}%
7082 \renewcommand*{\entryname}{Nota\c c\~ao}%
7083 \renewcommand*{\descriptionname}{Descri\c c\~ao}%
7084 \renewcommand*{\symbolname}{S\`imbolo}%
7085 \renewcommand*{\pagelistname}{Lista de P\`aginas}%
7086 \renewcommand*{\glssymbolsgroupname}{S\`imbolos}%
7087 \renewcommand*{\glsnumbersgroupname}{N\`umeros}%
7088 }%
7089 }

```

6.2 Polyglossia Captions

```

7090 \NeedsTeXFormat{LaTeX2e}
7091 \ProvidesPackage{glossaries-polyglossia}[2009/11/09 v1.0 (NLCT)]

```

English:

```

7092 \@ifundefined{captionseenglish}{}{%
7093 \expandafter\toks@\expandafter{\captionseenglish
7094 \renewcommand*{\glossaryname}{\textenglish{Glossary}}%
7095 \renewcommand*{\acronymname}{\textenglish{Acronyms}}%
7096 \renewcommand*{\entryname}{\textenglish{Notation}}%
7097 \renewcommand*{\descriptionname}{\textenglish{Description}}%
7098 \renewcommand*{\symbolname}{\textenglish{Symbol}}%
7099 \renewcommand*{\pagelistname}{\textenglish{Page List}}%
7100 \renewcommand*{\glssymbolsgroupname}{\textenglish{Symbols}}%
7101 \renewcommand*{\glsnumbersgroupname}{\textenglish{Numbers}}%
7102 }%

```



```

7103 \edef\captionseenglish{\the\toks@}%
7104 }

```

German:

```

7105 \@ifundefined{captionsgerman}{}{%
7106   \expandafter\toks@\expandafter{\captionsgerman
7107     \renewcommand*{\glossaryname}{\textgerman{Glossar}}%
7108     \renewcommand*{\acronymname}{\textgerman{Akronyme}}%
7109     \renewcommand*{\entryname}{\textgerman{Bezeichnung}}%
7110     \renewcommand*{\descriptionname}{\textgerman{Beschreibung}}%
7111     \renewcommand*{\symbolname}{\textgerman{Symbol}}%
7112     \renewcommand*{\pagelistname}{\textgerman{Seiten}}%
7113     \renewcommand*{\glssymbolsgroupname}{\textgerman{Symbole}}%
7114     \renewcommand*{\glsnumbersgroupname}{\textgerman{Zahlen}}%
7115   }%
7116   \edef\captionsgerman{\the\toks@}%
7117 }

```

Italian:

```

7118 \@ifundefined{captionssitalian}{}{%
7119   \expandafter\toks@\expandafter{\captionssitalian
7120     \renewcommand*{\glossaryname}{\textitalian{Glossario}}%
7121     \renewcommand*{\acronymname}{\textitalian{Acronimi}}%
7122     \renewcommand*{\entryname}{\textitalian{Nomenclatura}}%
7123     \renewcommand*{\descriptionname}{\textitalian{Descrizione}}%
7124     \renewcommand*{\symbolname}{\textitalian{Simbolo}}%
7125     \renewcommand*{\pagelistname}{\textitalian{Elenco delle pagine}}%
7126     \renewcommand*{\glssymbolsgroupname}{\textitalian{Simboli}}%
7127     \renewcommand*{\glsnumbersgroupname}{\textitalian{Numeri}}%
7128   }%
7129   \edef\captionssitalian{\the\toks@}%
7130 }

```

Dutch:

```

7131 \@ifundefined{captionsdutch}{}{%
7132   \expandafter\toks@\expandafter{\captionsdutch
7133     \renewcommand*{\glossaryname}{\textdutch{Woordenlijst}}%
7134     \renewcommand*{\acronymname}{\textdutch{Acroniemen}}%
7135     \renewcommand*{\entryname}{\textdutch{Benaming}}%
7136     \renewcommand*{\descriptionname}{\textdutch{Beschrijving}}%
7137     \renewcommand*{\symbolname}{\textdutch{Symbool}}%
7138     \renewcommand*{\pagelistname}{\textdutch{Pagina's}}%
7139     \renewcommand*{\glssymbolsgroupname}{\textdutch{Symbolen}}%
7140     \renewcommand*{\glsnumbersgroupname}{\textdutch{Cijfers}}%
7141   }%
7142   \edef\captionsdutch{\the\toks@}%
7143 }

```

Spanish:

```

7144 \@ifundefined{captionssspanish}{}{%
7145   \expandafter\toks@\expandafter{\captionssspanish
7146     \renewcommand*{\glossaryname}{\textspanish{Glosario}}%

```

```

7147 \renewcommand*{\acronymname}{\textspanish{Siglas}}%
7148 \renewcommand*{\entryname}{\textspanish{Entrada}}%
7149 \renewcommand*{\descriptionname}{\textspanish{Descripci' on}}%
7150 \renewcommand*{\symbolname}{\textspanish{S\'\i mbolo}}%
7151 \renewcommand*{\pagelistname}{\textspanish{Lista de p\ ' aginas}}%
7152 \renewcommand*{\glssymbolsgroupname}{\textspanish{S\'\i mbolos}}%
7153 \renewcommand*{\glslnumbersgroupname}{\textspanish{N\ ' umeros}}%
7154 }%
7155 \edef\captionsspanish{\the\toks@}%
7156 }

```

French:

```

7157 \@ifundefined{captionsfrench}{}{%
7158 \expandafter\toks@\expandafter{\captionsfrench
7159 \renewcommand*{\glossaryname}{\textfrench{Glossaire}}%
7160 \renewcommand*{\acronymname}{\textfrench{Acronymes}}%
7161 \renewcommand*{\entryname}{\textfrench{Terme}}%
7162 \renewcommand*{\descriptionname}{\textfrench{Description}}%
7163 \renewcommand*{\symbolname}{\textfrench{Symbole}}%
7164 \renewcommand*{\pagelistname}{\textfrench{Pages}}%
7165 \renewcommand*{\glssymbolsgroupname}{\textfrench{Symboles}}%
7166 \renewcommand*{\glslnumbersgroupname}{\textfrench{Nombres}}%
7167 }%
7168 \edef\captionsfrench{\the\toks@}%
7169 }

```

Danish:

```

7170 \@ifundefined{captionsdanish}{}{%
7171 \expandafter\toks@\expandafter{\captionsdanish
7172 \renewcommand*{\glossaryname}{\textdanish{Ordliste}}%
7173 \renewcommand*{\acronymname}{\textdanish{Akronymer}}%
7174 \renewcommand*{\entryname}{\textdanish{Symbolforklaring}}%
7175 \renewcommand*{\descriptionname}{\textdanish{Beskrivelse}}%
7176 \renewcommand*{\symbolname}{\textdanish{Symbol}}%
7177 \renewcommand*{\pagelistname}{\textdanish{Side}}%
7178 \renewcommand*{\glssymbolsgroupname}{\textdanish{Symboler}}%
7179 \renewcommand*{\glslnumbersgroupname}{\textdanish{Tal}}%
7180 }%
7181 \edef\captionsdanish{\the\toks@}%
7182 }

```

Irish:

```

7183 \@ifundefined{captionsirish}{}{%
7184 \expandafter\toks@\expandafter{\captionsirish
7185 \renewcommand*{\glossaryname}{\textirish{Gluais}}%
7186 \renewcommand*{\acronymname}{\textirish{Acrainmneacha}}%
7187 \renewcommand*{\entryname}{\textirish{Ciall}}%
7188 \renewcommand*{\descriptionname}{\textirish{Tuaireasc}}%
7189 \renewcommand*{\symbolname}{\textirish{Comhartha}}%
7190 \renewcommand*{\glssymbolsgroupname}{\textirish{Comhartha\ ' \i}}%
7191 \renewcommand*{\pagelistname}{\textirish{Leathanaigh}}%

```

```

7192 \renewcommand*{\glnumbersgroupname}{\textirish{Uimhreacha}}}%
7193 }%
7194 \edef\captionisirish{\the\toks@}%
7195 }

```

Hungarian:

```

7196 \@ifundefined{captionsmagyar}{\}%
7197 \expandafter\toks@\expandafter{\captionsmagyar
7198 \renewcommand*{\glossaryname}{\textmagyar{Sz\'ojegyz\'ek}}}%
7199 \renewcommand*{\acronymname}{\textmagyar{Bet\H uszavak}}}%
7200 \renewcommand*{\entryname}{\textmagyar{Kifejez\'es}}}%
7201 \renewcommand*{\descriptionname}{\textmagyar{Magyar\'azat}}}%
7202 \renewcommand*{\symbolname}{\textmagyar{Jel\'ol\'es}}}%
7203 \renewcommand*{\pagelistname}{\textmagyar{Oldalsz\'am}}}%
7204 \renewcommand*{\glssymbolsgroupname}{\textmagyar{Jelek}}}%
7205 \renewcommand*{\glnumbersgroupname}{\textmagyar{Sz\'amjegyek}}}%
7206 }%
7207 \edef\captionsmagyar{\the\toks@}%
7208 }

```

Polish

```

7209 \@ifundefined{captionspolish}{\}%
7210 \expandafter\toks@\expandafter{\captionspolish
7211 \renewcommand*{\glossaryname}{\textpolish{S{\l}ownik termin\'ow}}}%
7212 \renewcommand*{\acronymname}{\textpolish{Skr\'ot}}}%
7213 \renewcommand*{\entryname}{\textpolish{Termin}}}%
7214 \renewcommand*{\descriptionname}{\textpolish{Opis}}}%
7215 \renewcommand*{\symbolname}{\textpolish{Symbol}}}%
7216 \renewcommand*{\pagelistname}{\textpolish{Strony}}}%
7217 \renewcommand*{\glssymbolsgroupname}{\textpolish{Symbole}}}%
7218 \renewcommand*{\glnumbersgroupname}{\textpolish{Liczby}}}%
7219 }%
7220 \edef\captionspolish{\the\toks@}%
7221 }

```

Portugues

```

7222 \@ifundefined{captionsportuges}{\}%
7223 \expandafter\toks@\expandafter{\captionsportuges
7224 \renewcommand*{\glossaryname}{\textportuges{Gloss\'ario}}}%
7225 \renewcommand*{\acronymname}{\textportuges{Siglas}}}%
7226 \renewcommand*{\entryname}{\textportuges{Nota\c c\~ao}}}%
7227 \renewcommand*{\descriptionname}{\textportuges{Descri\c c\~ao}}}%
7228 \renewcommand*{\symbolname}{\textportuges{S\'imbolo}}}%
7229 \renewcommand*{\pagelistname}{\textportuges{Lista de P\'aginas}}}%
7230 \renewcommand*{\glssymbolsgroupname}{\textportuges{S\'imbolos}}}%
7231 \renewcommand*{\glnumbersgroupname}{\textportuges{N\'umeros}}}%
7232 }%
7233 \edef\captionsportuges{\the\toks@}%
7234 }

```

6.3 Brazilian Dictionary

This is a dictionary file provided by Thiago de Melo for use with the package.

```
7235 \ProvidesDictionary{glossaries-dictionary}{Brazilian}
```

Provide Brazilian translations:

```
7236 \providetranslation{Glossary}{Gloss\`ario}
7237 \providetranslation{Acronyms}{Siglas}
7238 \providetranslation{Notation (glossaries)}{Nota\c c\~ao}
7239 \providetranslation{Description (glossaries)}{Descri\c c\~ao}
7240 \providetranslation{Symbol (glossaries)}{S\`imbolo}
7241 \providetranslation{Page List (glossaries)}{Lista de P\`aginas}
7242 \providetranslation{Symbols (glossaries)}{S\`imbolos}
7243 \providetranslation{Numbers (glossaries)}{N\`umeros}
```

6.4 Danish Dictionary

This is a dictionary file provided for use with the package.

```
7244 \ProvidesDictionary{glossaries-dictionary}{Danish}
```

Provide Danish translations:

```
7245 \providetranslation{Glossary}{Ordliste}
7246 \providetranslation{Acronyms}{Akronymer}
7247 \providetranslation{Notation (glossaries)}{Symbolforklaring}
7248 \providetranslation{Description (glossaries)}{Beskrivelse}
7249 \providetranslation{Symbol (glossaries)}{Symbol}
7250 \providetranslation{Page List (glossaries)}{Side}
7251 \providetranslation{Symbols (glossaries)}{Symboler}
7252 \providetranslation{Numbers (glossaries)}{Tal}
```

6.5 Dutch Dictionary

This is a dictionary file provided for use with the package.

```
7253 \ProvidesDictionary{glossaries-dictionary}{Dutch}
```

Provide Dutch translations:

```
7254 \providetranslation{Glossary}{Woordenlijst}
7255 \providetranslation{Acronyms}{Acroniemen}
7256 \providetranslation{Notation (glossaries)}{Benaming}
7257 \providetranslation{Description (glossaries)}{Beschrijving}
7258 \providetranslation{Symbol (glossaries)}{Symbool}
7259 \providetranslation{Page List (glossaries)}{Pagina's}
7260 \providetranslation{Symbols (glossaries)}{Symbolen}
7261 \providetranslation{Numbers (glossaries)}{Cijfers}
```

6.6 English Dictionary

This is a dictionary file provided for use with the package.

```
7262 \ProvidesDictionary{glossaries-dictionary}{English}
```

Provide English translations:

```
7263 \providetranslation{Glossary}{Glossary}
7264 \providetranslation{Acronyms}{Acronyms}
7265 \providetranslation{Notation (glossaries)}{Notation}
7266 \providetranslation{Description (glossaries)}{Description}
7267 \providetranslation{Symbol (glossaries)}{Symbol}
7268 \providetranslation{Page List (glossaries)}{Page List}
7269 \providetranslation{Symbols (glossaries)}{Symbols}
7270 \providetranslation{Numbers (glossaries)}{Numbers}
```

6.7 French Dictionary

This is a dictionary file provided for use with the package.

```
7271 \ProvidesDictionary{glossaries-dictionary}{French}
```

Provide French translations:

```
7272 \providetranslation{Glossary}{Glossaire}
7273 \providetranslation{Acronyms}{Acronymes}
7274 \providetranslation{Notation (glossaries)}{Terme}
7275 \providetranslation{Description (glossaries)}{Description}
7276 \providetranslation{Symbol (glossaries)}{Symbole}
7277 \providetranslation{Page List (glossaries)}{Pages}
7278 \providetranslation{Symbols (glossaries)}{Symboles}
7279 \providetranslation{Numbers (glossaries)}{Nombres}
```

6.8 German Dictionary

This is a dictionary file provided for use with the package.

```
7280 \ProvidesDictionary{glossaries-dictionary}{German}
```

Provide German translations (quite a few variations were suggested for German; I settled on the following):

```
7281 \providetranslation{Glossary}{Glossar}
7282 \providetranslation{Acronyms}{Akronyme}
7283 \providetranslation{Notation (glossaries)}{Bezeichnung}
7284 \providetranslation{Description (glossaries)}{Beschreibung}
7285 \providetranslation{Symbol (glossaries)}{Symbol}
7286 \providetranslation{Page List (glossaries)}{Seiten}
7287 \providetranslation{Symbols (glossaries)}{Symbole}
7288 \providetranslation{Numbers (glossaries)}{Zahlen}
```

6.9 Irish Dictionary

This is a dictionary file provided for use with the package.

```
7289 \ProvidesDictionary{glossaries-dictionary}{Irish}
```

Provide Irish translations:

```
7290 \providetranslation{Glossary}{Gluais}
7291 \providetranslation{Acronyms}{Acrainmneacha}
```

```

7292 \providetranslation{Notation (glossaries)}{Ciall}
7293 \providetranslation{Description (glossaries)}{Tuairisc}
7294 \providetranslation{Symbol (glossaries)}{Comhartha}
7295 \providetranslation{Page List (glossaries)}{Leathanaigh}
7296 \providetranslation{Symbols (glossaries)}{Comhartha'\{i}}
7297 \providetranslation{Numbers (glossaries)}{Uimhreacha}

```

6.10 Italian Dictionary

This is a dictionary file provided for use with the package.

```
7298 \ProvidesDictionary{glossaries-dictionary}{Italian}
```

Provide Italian translations:

```

7299 \providetranslation{Glossary}{Glossario}
7300 \providetranslation{Acronyms}{Acronimi}
7301 \providetranslation{Notation (glossaries)}{Nomenclatura}
7302 \providetranslation{Description (glossaries)}{Descrizione}
7303 \providetranslation{Symbol (glossaries)}{Simbolo}
7304 \providetranslation{Page List (glossaries)}{Elenco delle pagine}
7305 \providetranslation{Symbols (glossaries)}{Simboli}
7306 \providetranslation{Numbers (glossaries)}{Numeri}

```

6.11 Magyar Dictionary

This is a dictionary file provided for use with the package.

```
7307 \ProvidesDictionary{glossaries-dictionary}{Magyar}
```

Provide translations:

```

7308 \providetranslation{Glossary}{Sz\'ojegyz\'ek}
7309 \providetranslation{Acronyms}{Bet\'H uszavak}
7310 \providetranslation{Notation (glossaries)}{Kifejez\'es}
7311 \providetranslation{Description (glossaries)}{Magyar\'azat}
7312 \providetranslation{Symbol (glossaries)}{Jel\'ol\'es}
7313 \providetranslation{Page List (glossaries)}{Oldalsz\'am}
7314 \providetranslation{Symbols (glossaries)}{Jelek}
7315 \providetranslation{Numbers (glossaries)}{Sz\'amjegyek}

```

6.12 Polish Dictionary

This is a dictionary file provided for use with the package.

```
7316 \ProvidesDictionary{glossaries-dictionary}{Polish}
```

Provide Polish translations:

```

7317 \providetranslation{Glossary}{S\lownik termin\'ow}
7318 \providetranslation{Acronyms}{Skr\'ot}
7319 \providetranslation{Notation (glossaries)}{Termin}
7320 \providetranslation{Description (glossaries)}{Opis}
7321 \providetranslation{Symbol (glossaries)}{Symbol}
7322 \providetranslation{Page List (glossaries)}{Strony}

```

```
7323 \providetranslation{Symbols (glossaries)}{Symbole}
7324 \providetranslation{Numbers (glossaries)}{Liczby}
```

6.13 Serbian Dictionary

This dictionary was provided by Zoran Filipovic.

```
7325 \ProvidesDictionary{glossaries-dictionary}{Serbian}
7326 \providetranslation{Glossary}{Mali re\vnik}
7327 \providetranslation{Acronyms}{Skra\'cenice}
7328 \providetranslation{Notation (glossaries)}{Oznaka}
7329 \providetranslation{Description (glossaries)}{Opis}
7330 \providetranslation{Symbol (glossaries)}{Simbol}
7331 \providetranslation{Page List (glossaries)}{Stranica}
7332 \providetranslation{Symbols (glossaries)}{Simboli}
7333 \providetranslation{Numbers (glossaries)}{Brojevi}
```

6.14 Spanish Dictionary

This is a dictionary file provided for use with the package.

```
7334 \ProvidesDictionary{glossaries-dictionary}{Spanish}
```

Provide Spanish translations:

```
7335 \providetranslation{Glossary}{Glosario}
7336 \providetranslation{Acronyms}{Siglas}
7337 \providetranslation{Notation (glossaries)}{Entrada}
7338 \providetranslation{Description (glossaries)}{Descripci\'on}
7339 \providetranslation{Symbol (glossaries)}{S\'mbolo}
7340 \providetranslation{Page List (glossaries)}{Lista de p\'aginas}
7341 \providetranslation{Symbols (glossaries)}{S\'mbolos}
7342 \providetranslation{Numbers (glossaries)}{N\'umeros}
```

Glossary

`makeindex` An indexing application. 17

`xindy` An flexible indexing application with multilingual support written in Perl. 17

Change History

1.01		'delimN and 'delimR in ist file	118
General: Added range facility in	1.03		
format key	61	\makefirstuc: changed 'pro-	
\writeist: Added spaces after		tected@edef to 'def	171

1.04	General: Added 'glstextformat ... 57	numberedsection: numbered-section package option added . 6
1.05	\glossarysection: added '@mk-both to 'glossarysection 26	numberline: numberline option added 5
	\glsmakefirsttuc: new 171	1.12
	\newglossaryentry: Changed the default value of the sort key to just the value of the name key 48	\@GLSpl: now uses 'glsentrydescplural and 'glsentrysymbolplural instead of 'glsentrydesc and 'glsentrysymbol 78
1.06	General: now requires etoolbox . 170	\@GLspl@: now uses 'glsentrydescplural and 'glsentrysymbolplural instead of 'glsentrydesc and 'glsentrysymbol 76
	\capitalisewords: new 172	\@glspl@: now uses 'glsentrydescplural and 'glsentrysymbolplural instead of 'glsentrydesc and 'glsentrysymbol 75
	\xcapitalisewords: new 172	General: added check for 'hypertarget separate to 'hyperlink (memoir defines 'hyperlink but not 'hypertarget) ... 69
1.07	\@gls@link: fixed bug caused by \theglsentrycounter setting the page number too soon 59	fixed bug ('GLSdesc shouldn't use 'gls@<type>@display) 89
	\glsadd: fixed bug caused by \theglsentrycounter setting the page number too soon 116	fixed bug ('GLSdesc shouldn't use 'gls@<type>@display) 88
1.08	General: Added babel support ... 21	fixed bug ('GLSdesc shouldn't use 'gls@<type>@display) 88
	listgroup: changed listgroup style to use \glsgetgrouptitle 177	fixed bug ('GLSfirst shouldn't use 'gls@<type>@display) 83
	altlistgroup: changed altlistgroup style to use \glsgetgrouptitle 178	fixed bug ('GLSfirst shouldn't use 'gls@<type>@display) 82
	\newglossaryentry: Fixed error message to say "description key" rather than "desc key" .. 48	fixed bug ('GLSfirst shouldn't use 'gls@<type>@display) 82
1.1	\@glossarysection: numbered sections and auto label added 27	fixed bug ('GLSfirstplural shouldn't use 'gls@<type>@display) 86
	\@gls@tmpb: changed \toksdef to \newtoks 64	fixed bug ('GLSfirstplural shouldn't use 'gls@<type>@display) 85
	\@gls@toc: numberline added .. 28	fixed bug ('GLSfirstplural shouldn't use 'gls@<type>@display) 85
	\@p@glossarysection: numbered sections and auto label added 27	fixed bug ('GLSname shouldn't use 'gls@<type>@display) 87
	General: Added support for translator package 21	fixed bug ('GLSname shouldn't use 'gls@<type>@display) .. 86, 87
	amsgen now loaded (\new@ifnextchar needed) 3	fixed bug ('GLSplural shouldn't use 'gls@<type>@display) 84
	translate: translate option added 15	
	\setglossarysection: new ... 27	

fixed bug ('Glsplural shouldn't use 'gls@<type>@display)	84	'newacronym in the preamble	147
fixed bug ('glsplural shouldn't use 'gls@<type>@display)	83	\newglossaryentry: Changed default first value	48
fixed bug ('GLSymbol shouldn't use 'gls@<type>@display)	92	Changed default firstplural value	48
fixed bug ('Glssymbol shouldn't use 'gls@<type>@display)	91	Removed restriction on only using 'newglossaryentry in the preamble	52
fixed bug ('glssymbol shouldn't use 'gls@<type>@display)	91	1.14	
fixed bug ('glssymbolplural shouldn't use 'gls@<type>@display)	92	\@gls@hypergroup: new	173
fixed bug ('GLStext shouldn't use 'gls@<type>@display)	81	General: added nonnumberlist key to 'printglossary	137
fixed bug ('Glstext shouldn't use 'gls@<type>@display)	81	added numberedsection key to 'printglossary	136
fixed bug ('glstext shouldn't use 'gls@<type>@display)	80	\firstacronymfont: new	150
descriptionplural: new	43	\glsautoprefix: new	6
\Glsentrydescplural: New ..	110	\glsnavhyperlink: changed 'edef to 'protected@edef ...	173
\Glsentrydescplural: New ..	110	\glsnavhypertarget: added write to aux file	173
\Glsentrysymbolplural: New	111	\glsnavigation: changed to only use labels for groups that are present	174
\Glsentrysymbolplural: New	111	1.15	
\newglossaryentry: Changed default first plural to be first key with s appended (was text key with s appended)	48	\@gls@link: added 'glslabel	59
descriptionplural support added	48	General: Added 'glssettoctitle ...	21
symbolplural support added ..	48	\gls@hypergroup: new ..	173
\SetDescriptionFootnoteAcronymStyle: Added 'protect before 'footnote and 'glslink	153	\glsnavhypertarget: added check if rerun required	173
\SetFootnoteAcronymStyle: Added 'protect before 'footnote and 'glslink	158	\glssettoctitle: new	20
symbolplural: new	45	\newglossaryentry: check for	
1.13		\@glo@first in description ...	51
General: Add Polish support	248, 251	check for '@glo@text in symbol	52
fixed bug that ignores 3rd parameter	80–93	\printglossary: changed the way the TOC title is set	133
\ACRfullpl: new	150	1.16	
\Acrfullpl: new	149	\@GLS@: Test glossary type is 'acronymtype in addition to checking if footnote option has been used	74, 231
\acrfullpl: new	149	\@GLSpl: Test glossary type is 'acronymtype in addition to checking if footnote option has been used	78
\acrpluralsuffix: New	147	\@Gls@: Test glossary type is 'acronymtype in addition to checking if footnote option has been used	72
\newacronym: Removed restriction on only using			

\@Glspl@: Test glossary type is 'acronym' type in addition to checking if footnote option has been used	77	\glshypernumber: modified to allow material to be attached to location	143
\@gls@: Test glossary type is 'acronym' type in addition to checking if footnote option has been used	71	\glsnavhyperlink: replaced 'hyperlink' to '@glslink'	173
\@gls@pl@: Test glossary type is 'acronym' type in addition to checking if footnote option has been used	232	\glsnavhypertarget: replaced 'hypertarget' to '@glstarget' ..	173
\@glsdisp: Test glossary type is 'acronym' type in addition to checking if footnote option has been used	80	\glssee: new	131
\@glspl@: Test glossary type is 'acronym' type in addition to checking if footnote option has been used	75	\glsseeformat: new	131
\@glstarget: raised the hypertarget so the target text doesn't scroll off the top of the page ..	69	\glsSetSuffixF: new	24
\newglossaryentry: Changed def to let	48	\glsSetSuffixFF: new	25
Changed if to ifx	50	\ifglxindy: new	17
1.17		\istfilename: added xindy support	23
\@do@wrglossary: new	128	\newglossaryentry: added non-numberlist key	48
\@do@seeglossary: new	131	added parent key	48
\@glo@storeentry: new	53	added see key	48
\@glossary: changed definition to use \index instead of \@index	127	Stored main part of entry format when entry is defined	52
\@glsdefaultplural: new	47	\newglossarystyle: made 'newglossarystyle long'	142
\@glsdefaultsort: new	47	\nopostdesc: new	23
\@glshypernumber: new	144	nonumberlist: new	45
\@glsnoname: new	47	\printglossary: added check to determine if 'printglossary' is already defined	133
\@glsnonextpages: new	137	added print language to aux file	133
\@wrglossary: modified to allow for xindy support	127	order: order package option added	17
General: added Brazilian dictionary	252	\writeist: added xindy support	118
Added Brazilian support	248	1.18	
added xindy support	17	\@gls@loadlist: new	7
parent: new	45	\@gls@loadlong: new	7
see: new	45	\@gls@loadsuper: new	7
\gls@suffixF: new	24	\@gls@loadtree: new	8
\gls@suffixFF: new	24	\glstarget: new	140
\glshyperlink: new	116	\newglossaryentry: Changed default value of sort to '@gls-defaultsort'	48
		moved sort sanitization to 'newglossaryentry'	51
		\oldacronym: new	146
		nolist: new	7
		nolong: new	7
		sort: moved sanitization to 'newglossaryentry'	44
		nostyles: new	8
		nosuper: new	7

notree: new	8	2.03	
1.19			\@GLS@: Added check for hyper-
\glsclearpage: new	28		first
\glsdisp: new	79		\@GLSpl: Added check for hyper-
\SetDescriptionAcronymStyle:			first
changed 'acronymfont to use			\@Gls@: Added check for hyper-
'textsmaller instead of 'smaller	156		first
\SetDescriptionFootnoteAcronymStyle\@Glspl@:			Added check for hyper-
changed 'acronymfont to use			first
'textsmaller instead of 'smaller	153		\@Gls@: Added check for hyper-
\SetFootnoteAcronymStyle:			first
changed 'acronymfont to use			\@Gls@@link: new
'textsmaller instead of 'smaller	158		\@Gls@link: added \leavevmode
\SetSmallAcronymStyle:		
changed 'acronymfont to use			Moved entry existence check to
'textsmaller instead of 'smaller	159		avoid duplicate code
1.2			\@glsdisp: Added check for hy-
General: fixed bug in ngerman			perfirst
captions	245		\@glspl@: Added check for hyper-
2.01			first
\@Gls@link: moved \@do@wrglossary			\@glossarymark: Added check to
before term is displayed to pre-			see if it's already defined
vent unwanted whatsit	60		hyperfirst: new
General: added nomain package		2.04	
option	11		\@GLS@: Changed test to check if
\forall glossaries: replaced			glossary type has been identi-
\ifthenelse with \ifx	38		fied as a list of acronyms
\forall glossentries: replaced			\@GLSpl: Changed test to check if
\ifthenelse with \ifx	38		glossary type has been identi-
\glstdefmain: new	11		fied as a list of acronyms
\glstdescwidth: changed			\@Gls@: Changed test to check if
\linewidth to \hsize . 180, 194			glossary type has been identi-
\glstlistdottedwidth: changed			fied as a list of acronyms
\linewidth to \hsize	179		\@GLspl@: Changed test to check
\glstpagelistwidth: changed			if glossary type has been iden-
\linewidth to \hsize . 180, 195			tified as a list of acronyms ...
\writeist: removed item_02 - no			\@glossaryentryfield: new ..
such makeindex key	122		\@glossarysubentryfield:
2.02			new
General: Changed Brazil to Brazil-			\@Gls@: Changed test to check if
ian	252		glossary type has been identi-
false will prevent automatic			fied as a list of acronyms
loading of translator package	18		\@Glsacronymlists: new
\@glossarymark: New	26		\@glstisp: Changed test to check
\@glossarysection: changed			if glossary type has been iden-
'@mkboth to '@glossarymark ..	26		tified as a list of acronyms ...
\@printglossary: suppressed			\@glstisp@: Changed test to check
warning globally rather than			if glossary type has been iden-
locally	135		tified as a list of acronyms ...

\@newglossaryentryposthook: new 52	\writeist: Added \string be- fore opening and closing braces. Patch provided by Sergiu Dotenco 123
\@newglossaryentryprehook: new 52	
acronymlists:new 13	2.06
\DeclareAcronymList:new ... 12	\altnewglossary:new 42
\DefineAcronymSynonyms:new 163	\CustomAcronymFields:new . 162
\glsadd: fixed bug that ignored counter 116	\CustomNewAcronymDef:new . 162
\Glsentryuseri:new 112	\SetCustomDisplayStyle:new 162
\Glsentryuseri:new 112	\SetCustomStyle:new 162
\Glsentryuserii:new 112	2.07
\Glsentryuserii:new 112	General: glssadd format key stored in \@glsnumberformat (was mistakenly stored in \@glo@format) 116
\Glsentryuseriii:new 112	3.0
\Glsentryuseriii:new 112	\@@do@wrglossary: added check for hyper location prefix ... 129
\Glsentryuseriv:new 113	modified to use new format .. 128
\Glsentryuseriv:new 113	\@glossarysec: replaced \@ifundefined with \ifcsundef 5
\Glsentryuseriv:new 113	\@do@seeglossary: Sanitize and escape cross-referencing in- formation 131
\newglossary: added check to determine if \gls@<type>@display and \gls@<type>@displayfirst have been defined. 41	\@gls@counterwithin:new 9
\newglossaryentry: added user1-6 keys 48	\@gls@ifinlist:new 29
\SetAcronymLists:new 13	\@gls@link: added \@gls@saveentrycounter 60
\SetDefaultAcronymDisplayStyle: new 151	added \@gls@setsort 60
\SetDefaultAcronymStyle: new 151	\@gls@saveentrycounter:new 60
\SetDescriptionAcronymDisplayStyle: new 155	\@gls@setupsort@def:new ... 10
\SetDescriptionDUAAcronymDisplayStyle: new 154	\@gls@setupsort@standard: new 9
\SetDescriptionFootnoteAcronymDisplayStyle: new 152	\@gls@setupsort@use:new ... 10
\SetDUADisplayStyle:new .. 160	\@gls@xdy@locationlist:new 32
\SetFootnoteAcronymDisplayStyle: new 156	\@gls@link: replaced \@ifundefined with \ifcsundef 69
\SetSmallAcronymDisplayStyle: new 158	\@gls@nextpages:new 137
2.05	\@makeglossary: Added check for savewrites 124
\@glsdisp: Added closing brace. Patch provided by Sergiu Dotenco 79	\@set@glo@numformat: added 4th argument 62
Removed spurious brace. Patch provided by Sergiu Dotenco . 80	\@wrglossary: modified to take into account savewrites 127
	\@xdy@attributelist:new 29
	General: added prefix to hyperlink 145
	etoolbox now loaded 4

replaced \@ifundefined with \ifcsundef 19, 58, 136	\glsgetgrouptitle: re- placed \@ifundefined with \ifcsundef 141
\acrfootnote: new 152	\glshyperlink: changed de- fault from \glsentryname to \glsentrytext 116
\ACRfull: added starred version 148	\glshypernumber: replaced \@ifundefined with \ifcsundef 143
\Acrfull: added starred version 148	\glsnumberformat: replaced \@ifundefined with \ifcsundef 25
\acrfull: added starred version 147	\glsrefentry: new 139
\ACRfullpl: added starred ver- sion 150	\glsresetsubentrycounter: new 138
\Acrfullpl: added starred ver- sion 149	\glsseeitem: hyperlink uses \glsseeitemformat instead of \glsentryname 132
\acrlinkfootnote: new 152	\glsseeitemformat: new 132
\acrnolinkfootnote: new ... 152	\glssortnumberfmt: new 9
\addglossarytocaptions: re- placed \@ifundefined with \ifcsundef 21	\glsstepentry: new 138
savewrites: new 18	\glsstepsubentry: new 139
see: added \@gls@seeautonumberlist 45	\glsesubentrycounterlabel: new 139
seeautonumberlist: new 7	\glssubentryitem: new 139
\glossarymark: replaced \@ifundefined with \ifcsundef 26	theglossary: replaced \@ifundefined with \ifcsundef 140
\glossarysection: replaced \@ifundefined with \ifcsundef 26	short: new 46
\glossarystyle: replaced \@ifundefined with \ifcsundef 142	shortplural: new 46
\gls@codepage: replaced \@ifundefined with \ifcsundef 17	\ifglossaryexists: re- placed \@ifundefined with \ifcsundef 39
\gls@doclearpage: replaced \@ifundefined with \ifcsundef 28	\ifglentryexists: re- placed \@ifundefined with \ifcsundef 39
\glsadd: added \@gls@saveentrycounterglossaryentry: new 137	\istfile: deprecated 126
..... 116	glossarysubentry: new 138
\GlsAddXdyCounters: new 29	\newglossary: added \@gls@defsortcount 42
\glsentrycounterlabel: new 139	replaced \@ifundefined with \ifcsundef 41
\glsentryitem: new 139	\newglossaryentry: added \@gls@defsort 51
\Glsentrylong: new 114	added short and long keys 48
\glsentrylong: new 113	replaced \@ifundefined with \ifcsundef 49
\Glsentrylongpl: new 114	replaced \DeclareRobustCommand with \newrobustcmd 47
\glsentrylongpl: new 114	
\Glsentryshort: new 113	
\glsentryshort: new 113	
\Glsentryshortpl: new 113	
\glsentryshortpl: new 113	

<code>\newglossarystyle:</code>	re-	<code>\showglouserii:</code>	new	166
placed <code>\@ifundefined</code> with		<code>\showglouseriii:</code>	new	167
<code>\ifcsundef</code>	142	<code>\showglouseriv:</code>	new	167
<code>entrycounter:</code>	new	<code>\showglouserv:</code>	new	167
<code>entrycounterwithin:</code>	new	<code>\showglouservi:</code>	new	167
<code>\oldacronym:</code>	replaced <code>\@ifundefined</code>	<code>subentrycounter:</code>	new	9
with <code>\ifcsundef</code>	146	<code>\writeist:</code>	added xindy-only	
<code>compatible-2.07:</code>	compatible-	macro definitions to glossary		
<code>2.07 option added</code>	18	open tag	120	
<code>long:</code>	new	modified to support new for-		
<code>longplural:</code>	new	mat	118	
<code>nonumberlist:</code>	now boolean			
<code>sort:</code>	new			
<code>counter:</code>	replaced <code>\@ifundefined</code>	General: made robust	73	
with <code>\ifcsundef</code>	45	<code>\ACRfull:</code>	made robust	148
<code>\printglossary:</code>	added	<code>\Acrfull:</code>	made robust	148
<code>\currentglossary</code>	134	<code>\acrfull:</code>	made robust	147
added <code>\glsnextpages</code>	134	<code>\acrfullformat:</code>	removed	
make to title default to title	134	<code>\acronymfont:</code>	as it should al-	
replaced <code>\@ifundefined</code> with		ready be set in the second ar-		
<code>\ifcsundef</code>	133, 135	gument.	148	
<code>\SetDescriptionFootnoteAcronymDisplayStyle</code>	3.01	<code>\ACRfullpl:</code>	made robust	150
expanded options link op-		<code>\Acrfullpl:</code>	made robust	149
tions	152	<code>\acrfullpl:</code>	made robust	149
<code>\setentrycounter:</code>	added op-	<code>\ACRlong:</code>	made robust	107
tional argument	142	<code>\Acrlong:</code>	made robust	106
<code>\showacronymlists:</code>	new	<code>\acrlong:</code>	made robust	106
<code>\showglocounter:</code>	new	<code>\ACRlongpl:</code>	made robust	109
<code>\showglodesc:</code>	new	<code>\Acrlongpl:</code>	made robust	108
<code>\showglodescplural:</code>	new	<code>\acrlongpl:</code>	made robust	108
<code>\showglofirst:</code>	new	<code>\ACRshort:</code>	made robust	104
<code>\showglofirstpl:</code>	new	<code>\Acrshort:</code>	made robust	103
<code>\showgloflag:</code>	new	<code>\acrshort:</code>	made robust	102
<code>\showgloindex:</code>	new	<code>\ACRshortpl:</code>	made robust	105
<code>\showglolevel:</code>	new	<code>\Acrshortpl:</code>	made robust	105
<code>\showgloname:</code>	new	<code>\acrshortpl:</code>	made robust	104
<code>\showgloparent:</code>	new	<code>\Gls:</code>	made robust	72
<code>\showgloplural:</code>	new	<code>\glsadd:</code>	made robust	116
<code>\showglosort:</code>	new	<code>\glsaddall:</code>	made robust	116
<code>\showglossaries:</code>	new	<code>\GLSdesc:</code>	made robust	88
<code>\showglossarycounter:</code>	new	<code>\Glsdesc:</code>	made robust	88
<code>\showglossaryentries:</code>	new	<code>\glsdesc:</code>	made robust	87
<code>\showglossaryin:</code>	new	<code>\GLSdescplural:</code>	made robust	90
<code>\showglossaryout:</code>	new	<code>\Glsdescplural:</code>	made robust	89
<code>\showglossarytitle:</code>	new	<code>\glsdescplural:</code>	made robust	89
<code>\showglosymbol:</code>	new	<code>\glsfirst:</code>	made robust	82
<code>\showglosymbolplural:</code>	new	<code>\GLSfirstplural:</code>	made robust	85
<code>\showglotext:</code>	new	<code>\Glsfirstplural:</code>	made robust	85
<code>\showglotype:</code>	new	<code>\glsfirstplural:</code>	made robust	84

\glslink: made robust	59	General: added check for polyglos-	
\GLSname: made robust	87	sia	19
\Glsname: made robust	86	reversed order of package check	22
\glsname: made robust	86	savenumberlist: new	7
\GLSpl: made robust	77	ucmark: new	8
\Glspl: made robust	76	\gls@save@numberlist: new ..	133
\glspl: made robust	74	\glsdisplaynumberlist: new	114
\GLSplural: made robust	84	\glsentrycounter: set default	
\GLSsymbol: made robust	91	value	60
\Glsymbol: made robust	91	\Glsentryfull: fixed bug (re-	
\glsymbol: made robust	90	placed \glsentryshortpl	
\GLSsymbolplural: made robust	93	with \glsentryshort)	114
\Glsymbolplural: made robust	92	\glsentryfullpl: fixed bug (re-	
\glsymbolplural: made robust	92	placed \glsentryshort with	
\Glstext: made robust	81	\glsentryshortpl)	114
\glstext: made robust	80	\glsentrynumberlist: new ..	114
\GLSuseri: made robust	94	\glsmoveentry: new	52
\Glsuseri: made robust	94	\glsnumlistlastsep: new ...	115
\glsuseri: made robust	93	\glsnumlistsep: new	115
\GLSuserii: made robust	96	\glsresetsubentrycounter:	
\Glsuserii: made robust	95	new	138
\glsuserii: made robust	95	\glswritefiles: added check	
\GLSuseriii: made robust	97	for existence of token in case	
\Glsuseriii: made robust	97	\makeglossaries has been	
\glsuseriii: made robust	96	omitted	126
\GLSuseriv: made robust	99	\ifglshaschildren: new	40
\Glsuseriv: made robust	98	\ifglshasparent: new	40
\glsuseriv: made robust	98	\makeglossaries: added list	
\GLSuserv: made robust	100	parser	125
\Glsuserv: made robust	100	indexonlyfirst: new	16
\glsuserv: made robust	99	\newglossaryentry: added	
\GLSuservi: made robust	102	numberlist element	51
\Glsuservi: made robust	101	\printglossary: add a way to	
\glsuservi: made robust	101	fetch current entry label ...	134
\glswritefiles: added check		\renewglossarystyle: new ..	143
for empty glossaries	126	\showglossaryentries: fixed	
3.02		misspelt command	169
\@do@wrglossary: changed		\SmallNewAcronymDef: fixed	
\@glslocref to \theglsentrycounter		broken short and long plural	158
.....	130	3.03	
\@do@wrglossary: changed		\@gls@sanitizesort: new	14
\@do@wr@glossary to test for		\@gls@setupsort@standard:	
indexonlyfirst option; put old		used \@gls@sanitizesort ..	9
\@do@wr@glossary code into		General: allow title to set toctitle	136
\@do@wrglossary	128	altlongragged4col: added	
\@gls@missingnumberlist:		check for glsnogroupskip ..	189
new	47	altsuperragged4col: added	
\@wrglossary: added check for		check for glsnogroupskip ..	205
glossary file defined	127		

alttree: added check for	added local key	59
glsnogroupskip	\changes: new	4
index: added check for	\gls@Alphpage: new	128
glsnogroupskip	\gls@alphpage: new	128
nogroupskip: new	\gls@disablepagerefexpansion:	
long: added check for	new	128
glsnogroupskip	\gls@numberpage: new	128
long3col: added check for	\gls@protected@pagefmts:	
glsnogroupskip	new	128
long4col: added check for	\gls@romanpage: new	128
glsnogroupskip	\glsdefmain: added check for	
longragged: added check for	doc package	11
glsnogroupskip	\glsorg@endtheglossary: new .	5
longragged3col: added check	\glsorg@glossary: new	4
for glsnogroupskip	\glsorg@theglossary: new	5
nopostdot: new	\glsorg@wrglossary: new	4
\printglossary: allow title to	altlist: replaced \newline with	
override default toctitle	paragraph break	178
tree: added check for	\PrintChanges: new	5
glsnogroupskip	\printglossary: Moved aux	
treenoname: added check for	write to end of document to	
glsnogroupskip	prevent unwanted whatsit oc-	
super: added check for	curring here.	135
glsnogroupskip		
super3col: added check for	3.05	
glsnogroupskip	\@@do@wrglossary: add Roman	
super4col: added check for	case. Fixed bugs in the else	
glsnogroupskip	statements	129
superragged: added check for	\@gls@link: added check for “no-	
glsnogroupskip	hypertypes”	60
superragged3col: added check	\@gls@nohyperlist: new	13
for glsnogroupskip	mcolalttree: replaced ‘2’ with	
3.04	\glsmcols	193
\@@do@wrglossary: changed	mcolindex: replaced ‘2’ with	
\theglsentrycounter back	\glsmcols	191
to \@glslocref	mcoltree: replaced ‘2’ with	
modified to compensate for	\glsmcols	192
possible incorrect page num-	mcoltreenoname: replaced ‘2’	
ber	with \glsmcols	193
129	\gls@protected@pagefmts:	
\@gls@escbsdq: unsani-	added Roman to list	128
tize \gls@numberpage,	\gls@Romanpage: new	128
\gls@alphpage, \gls@Alphpage	\GlsDeclareNoHyperList: new	13
and \gls@romanpage	\glsgetgrouplabel: fixed bug	
63	(typo in \equal)	142
General: Added check for doc	\nopostdesc: made robust	23
package	nohypertypes: new	14
4		
added datatool-base as a re-		
quired package		
3		

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	128
\@glossarysec	5
\@glossaryseclabel	6
\@glossarysecstar	6
\@ACRlong	236
\@ACRshort	235
\@Acrlong	236
\@Acrshort	235
\@GLS@	73, 231
\@GLSpl	78
\@GLSpl@	234
\@Gls@	72, 230
\@Glspl@	76, 233
\@acrlong	236
\@acrshort	235
\@addtoacronymlists	12
\@delimN	144
\@delimR	144
\@disable@onlypremakeg	20
\@disable@premakecs	20
\@disabled@glsgladdxdycounters ..	30
\@do@seeglossary	131
\@do@wrglossary	128, 216
\@glo@seeautonumberlist	7
\@glo@storeentry	53
\@glo@types	41
\@glossary	127
\@glossary@default@style	6
\@glossaryentryfield	53, 237
\@glossarysection	27
\@glossarysubentryfield ..	53, 237
\@gls	70
\@gls@	71, 229
\@gls@@link	59
\@gls@addpredefinedattributes ..	31
\@gls@checkactual	67
\@gls@checkbar	66
\@gls@checkescactual	65
\@gls@checkescbar	65
\@gls@checkesclevel	66
\@gls@checkescquote	64
\@gls@checklevel	67
\@gls@checkmkidxchars	63
\@gls@checkquote	64
\@gls@codepage	37
\@gls@counterwithin	9
\@gls@escbsdq	62
\@gls@fixbraces	131
\@gls@getcounter	43
\@gls@getcounterprefix	130
\@gls@hypergroup	173
\@gls@ifinlist	29
\@gls@link	59
\@gls@loadlist	7
\@gls@loadlong	7
\@gls@loadsuper	7
\@gls@loadtree	8
\@gls@missingnumberlist	47
\@gls@noaccess	223
\@gls@nohyperlist	13
\@gls@onlypremakeg	19
\@gls@pl@	232
\@gls@renewglossary	127
\@gls@sanitizedesc	14
\@gls@sanitizename	14
\@gls@sanitizesort	14
\@gls@sanitizesymbol	14
\@gls@saveentrycounter	60
\@gls@setcounter	42
\@gls@setupsort@def	10
\@gls@setupsort@standard	9
\@gls@setupsort@use	10
\@gls@tmpb	64
\@gls@toc	28
\@gls@updatechecked	64
\@gls@xdy@Lclass@Alpha-page-numbers	33
\@gls@xdy@Lclass@Appendix-page-numbers	33
\@gls@xdy@Lclass@Roman-page-numbers	33
\@gls@xdy@Lclass@alpha-page-numbers	33
\@gls@xdy@Lclass@arabic-page-numbers	33

\@gls@xdy@Lclass@arabic-section-numbers	A	
.....	33	\Ac 164
\@gls@xdy@Lclass@roman-page-numbers		\ac 164
.....	32	access (key) 221
\@gls@xdy@locationlist	32	accsupp package 221
\@gls@xdy@checkbackslash	68	\accsuppglossaryentryfield .. 237
\@gls@xdy@checkquote	68	\accsuppglossarysubentryfield 237
\@gls@Alphacompositor	24, 33	\Acf 164
\@gls@acronymlists	12	\acf 164
\@gls@defaultplural	47	\Acfp 164
\@gls@defaultsort	47	\acfp 164
\@gls@disp	79	\Acl 163
\@gls@firstletter	117	\acl 163
\@gls@hypernumber	144	\Aclp 164
\@gls@link	69	\aclp 163
\@gls@minrange	118	\Acp 164
\@gls@nextpages	137	\acp 164
\@gls@noname	47	\acrfootnote 152
\@gls@nonextpages	137	\ACRfull 148
\@gls@openfile	124	\Acrfull 148
\@gls@order	17	\acrfull 147, 148
\@gls@pl@	75	\acrfullformat 148
\@gls@target	69	\ACRfullpl 150
\@gls@widestname	211	\Acrfullpl 149
\@ist@filename	23	\acrfullpl 149
\@make@glossary	124	\acrlinkfootnote 152
\@new@glossary	42	\acrlinkfullformat 148
\@new@glossaryentryposthook	52	\ACRlong 107
\@new@glossaryentryprehook	52	\Acrlong 106
\@no@postdesc	23	\acrlong 106
\@only@premakeg	19	\ACRlongpl 109
\@p@@glossarysection	27	\Acrlongpl 108
\@set@@glo@numformat	61, 216	\acrlongpl 108
\@sgls	70, 79	\acrnameformat 150, 156
\@sgls@link	59	\acrno@linkfootnote 152
\@wrg@glossary	127	acronym (option) 12
\@xdy@main@language	17	\acronymfont . 150, 153, 156, 158, 159
\@xdy@attributelist	29	acronymlists (option) 13
\@xdy@attributes	29	\acronymname 20
\@xdy@language	37	\acronymtype 11, 146, 147
\@xdy@lettergroups	38	\acrpluralsuffix 147
\@xdy@locationclassorder	34	\ACRshort 104
\@xdy@locref	29	\Acrshort 103
\@xdy@requiredstyles	35	\acrshort 102
\@xdy@sortrules	35	\ACRshortpl 105
\@xdy@useralphabets	31	\Acrshortpl 105
\@xdy@userlocationdefs	33	\acrshortpl 104
\@xdy@userlocationnames	33	\Acs 163
		\acs 163

<code>\Acsp</code>	163	<code>\defglssdisplay</code>	41, 43, 44, 57, 58
<code>\acsp</code>	163	<code>\defglssdisplayfirst</code>	41, 43, 44, 57, 58
<code>\addglossarytocaptions</code>	21	<code>\DefineAcronymSynonyms</code>	163
<code>\addto</code>	21	<code>\delimN</code>	25, 143
<code>align</code> (environment)	60	<code>\delimR</code>	25, 143
<code>altlist</code> (style)	178	<code>description</code> (environment)	177
<code>altlistgroup</code> (style)	178	<code>description</code> (key)	43
<code>altlisthypergroup</code> (style)	179	<code>description</code> (option)	16
<code>atlong4col</code> (style)	184	<code>descriptionaccess</code> (key)	222
<code>atlong4colborder</code> (style)	185	<code>\DescriptionDUANewAcronymDef</code>	154
<code>atlong4colheader</code> (style)	184	<code>\DescriptionFootnoteNewAcronymDef</code>	152, 238
<code>atlong4colheaderborder</code> (style)	185	
<code>atlongragged4col</code> (style)	189	<code>\descriptionname</code>	20
<code>atlongragged4colborder</code> (style)	190	<code>\DescriptionNewAcronymDef</code>	155, 239
<code>atlongragged4colheader</code> (style)	189	<code>descriptionplural</code> (key)	43
<code>atlongragged4colheaderborder</code>		<code>descriptionpluralaccess</code> (key) .	222
(style)	190	<code>doc package</code>	4, 11
<code>\altnewglossary</code>	42	<code>dua</code> (option)	16
<code>altsuper4col</code> (style)	199	<code>\DUANewAcronymDef</code>	160
<code>altsuper4colborder</code> (style)	200		
<code>altsuper4colheader</code> (style)	200	E	
<code>altsuper4colheaderborder</code> (style)	200	<code>entrycounter</code> (option)	8
<code>altsuperragged4col</code> (style)	205	<code>entrycounterwithin</code> (option)	8
<code>altsuperragged4colborder</code> (style)	206	<code>\entryname</code>	20
<code>altsuperragged4colheader</code> (style)	205	<code>environments:</code>	
<code>altsuperragged4colheaderborder</code>		<code>align</code>	60
(style)	206	<code>description</code>	177
<code>alttree</code> (style)	211	<code>longtable</code>	7, 165, 180–190
<code>alttreegroup</code> (style)	213	<code>multicols</code>	191
<code>alttreehypergroup</code> (style)	214	<code>supertabular</code>	7, 165, 194–206
<code>amsgen package</code>	4, 58	<code>theglossary</code>	4, 5,
<code>\andname</code>	21	25, 140, 142, 192, 193, 208, 209, 211	
<code>array package</code>	185, 201	<code>theindex</code>	207
<code>article class</code>	130	<code>equation counter</code>	60, 61
		<code>etoolbox package</code>	4, 170
B			
<code>babel package</code>	18–21, 37, 242	F	
C		<code>file types</code>	
<code>\capitalisewords</code>	172	<code>.aux</code>	135
<code>\changes</code>	4	<code>.glo</code>	53
<code>compatible-2.07</code> (option)	18	<code>.ist</code>	117, 123, 124
<code>counter</code> (key)	45	<code>.toc</code>	28
<code>counter</code> (option)	13	<code>.xdy</code>	23
<code>\CustomAcronymFields</code>	162	<code>glo</code>	170
<code>\CustomNewAcronymDef</code>	162	<code>\findrootlanguage</code>	36
D		<code>first</code> (key)	44
<code>\DeclareAcronymList</code>	12	<code>firstaccess</code> (key)	222
<code>\DefaultNewAcronymDef</code> ...	151, 238	<code>\firstacronymfont</code>	150
		<code>firstplural</code> (key)	44
		<code>firstpluralaccess</code> (key)	222

footnote (option)	16	textaccess	221
\FootnoteNewAcronymDef ..	157, 239	type	45
\forallforallglossaries	38	user1	46
\forallforallglsentries	39	user2	46
\forallforlglsentries	38	user3	46
		user4	46
G		user5	46
\glolinkprefix	60	user6	46
glossareentry counter	138	glossary package	1, 146
glossaries package		glossary styles:	
.....	37, 118, 165, 177, 214, 221	altlist	178, 179
glossaries-accsupp package	53, 221	altlist	178
\GlossariesWarning	18	altlistgroup	178, 179
\GlossariesWarningNoLine	18	altlistgroup	178
\glossary	41, 124, 127, 142	altlisthypergroup	179
glossary counters:		altlisthypergroup	179
glossaryentry	137	altlong4col	184, 185, 189
glossarysubentry	138	altlong4col	184
glossary keys:		altlong4colborder	185
access	221	altlong4colborder	185
counter	45	altlong4colheader	184
description	43	altlong4colheader	184
descriptionaccess	222	altlong4colheaderborder ..	185
descriptionplural	43	altlong4colheaderborder ..	185
descriptionpluralaccess ..	222	altlongragged4col	189, 190
first	44	altlongragged4col	189
firstaccess	222	altlongragged4colborder ..	190
firstplural	44	altlongragged4colborder ..	190
firstpluralaccess	222	altlongragged4colheader ..	189
long	46	altlongragged4colheader ..	189
longaccess	222	altlongragged4colheaderborder	
longplural	47	190
longpluralaccess	223	altlongragged4colheaderborder	
name	43	190
nonumberlist	45	altsuper4col	199, 200, 205
parent	45	altsuper4col	199
plural	44	altsuper4colborder	200
pluralaccess	222	altsuper4colborder	200
see	45	altsuper4colheader	200
short	46	altsuper4colheader	200
shortaccess	222	altsuper4colheaderborder ..	200
shortplural	46	altsuper4colheaderborder ..	200
shortpluralaccess	222	altsuperragged4col ...	205, 206
sort	44	altsuperragged4col	205
symbol	44	altsuperragged4colborder ..	206
symbolaccess	222	altsuperragged4colborder ..	206
symbolplural	45	altsuperragged4colheader ..	205
symbolpluralaccess	222	altsuperragged4colheader ..	205
text	44		

altsuperragged4colheaderborder	longragged	186, 187
altsuperragged4colheaderborder	longragged	186
altsuperragged4colheaderborder	longragged3col	187, 188
altsuperragged4colheaderborder	longragged3col	187
almtree	longragged3colborder	188
almtree	longragged3colborder	188
almtreegroup	longragged3colheader	188
almtreegroup	longragged3colheader	188
almtreehypergroup	longragged3colheaderborder	188
almtreehypergroup	longragged3colheaderborder	188
index	longraggedborder	186
index	longraggedborder	186
indexgroup	longraggedheader	187
indexgroup	longraggedheader	187
indexhypergroup	longraggedheaderborder	187
indexhypergroup	longraggedheaderborder	187
inline	mcollalmtree	194
list	mcollalmtree	193
list	mcollalmtreegroup	194
listdotted	mcollalmtreegroup	194
listdotted	mcollalmtreehypergroup	194
listgroup	mcollalmtreehypergroup	194
listgroup	mcollindex	191
listhypergroup	mcollindex	191
listhypergroup	mcollindexgroup	191
long	mcollindexgroup	191
long	mcollindexhypergroup	191
long3col	mcollindexhypergroup	191
long3col	mcolltree	192
long3colborder	mcolltree	192
long3colborder	mcolltreegroup	192
long3colborder	mcolltreehypergroup	192
long3colheader	mcolltreehypergroup	192
long3colheader	mcolltreename	193
long3colheaderborder	mcolltreename	192
long3colheaderborder	mcolltreenamegroup	193
long3colheaderborder	mcolltreenamegroup	193
long4col	mcolltreenamehypergroup	193
long4col	mcolltreenamehypergroup	193
long4colborder	sublistdotted	179
long4colborder	super	195, 196, 202
long4colheader	super	195
long4colheader	super3col	196, 197
long4colheaderborder	super3col	196
long4colheaderborder	super3colborder	197
longborder	super3colborder	197
longborder	super3colborder	197
longheader	super3colheader	197
longheader	super3colheader	197
longheaderborder	super3colheader	197
longheaderborder	super3colheaderborder	197
longheaderborder	super3colheaderborder	197

super3colheaderborder	197	glossary-mcols package	191
super4col	198, 199	glossary-super package	
super4col	198	7, 180, 194, 201, 205	
super4colborder	199	glossary-superragged package	201
super4colborder	199	glossary-tree package	8, 206
super4colheader	198	glossaryentry (counter)	137
super4colheader	198	glossaryentry counter	8, 138, 139
super4colheaderborder	199	\glossaryentryfield 44, 140, 142, 143	
super4colheaderborder	199	\glossaryentrynumber	137
superborder	195	\glossaryentrynumbers 6, 25, 134, 135	
superborder	195	\glossaryheader	140, 142
superheader	196	\glossarymark	8, 26
superheader	196	\glossaryname	20, 21
superheaderborder	196	\glossarypostamble	26, 142
superheaderborder	196	\glossarypreamble	25, 142
superragged	201, 203	\glossarysection	5, 26, 41
superragged	201	\glossarystyle	142, 165
superragged3col	203, 204	glossarysubentry (counter)	138
superragged3col	203	glossarysubentry counter	9, 138, 139
superragged3colborder	204	\GLS	73
superragged3colborder	204	\Gls	72, 76, 170
superragged3colheader	204	\gls	4, 44, 56, 57, 59,
superragged3colheader	204	70, 73, 74, 80, 82–84, 86, 87, 89,	
superragged3colheaderborder		90, 92, 93, 95, 96, 98, 99, 101, 139	
204		\gls@Alphpage	128
superraggedborder	202	\gls@alphpage	128
superraggedborder	202	\gls@checkisacronymlist	13
superraggedheader	202	\gls@codepage	17
superraggedheader	202	\gls@disablepagerefexpansion	128
superraggedheaderborder . .	203	\gls@doclearpage	28
superraggedheaderborder . .	203	\gls@hypergroup prerun	173
superraggedright3colheaderborder		\gls@level	47
204		\gls@numberpage	128
tree	192, 208–211	\gls@protected@pagefmts	128
tree	208	\gls@Romanpage	128
treegroup	192, 209	\gls@romanpage	128
treegroup	209	\gls@save@numberlist	133
treehypergroup	209	\gls@suffixF	24
treehypergroup	209	\gls@suffixFF	24
treenoname	192, 210	\gls@accessdisplay	229
treenoname	210	\gls@sacccsupp	226
treenonamegroup	211	\gls@sadd	56, 116, 142
treenonamegroup	210	\gls@sadd options	
treenonamehypergroup	211	counter	116
treenonamehypergroup	211	format	116, 143
glossary-hypernav package	117	\gls@saddall	56, 116
glossary-list package	6, 7, 177	\gls@saddall options	
glossary-long package 7, 180, 189, 194, 195		types	116
glossary-longragged package	185	\GlsAddLetterGroup	38

<code>\GlsAddSortRule</code>	35	<code>\Glsentryfullpl</code>	114
<code>\GlsAddXdyAlphabet</code>	32	<code>\glentryfullpl</code>	114
<code>\GlsAddXdyAttribute</code>	30, 214	<code>\glentryitem</code>	139
<code>\GlsAddXdyCounters</code>	29, 215	<code>\Glsentrylong</code>	114
<code>\GlsAddXdyLocation</code>	34, 215	<code>\glentrylong</code>	113
<code>\GlsAddXdyStyle</code>	35	<code>\glentrylongaccess</code>	225
<code>\glsclearpage</code>	6	<code>\Glsentrylongpl</code>	114
<code>\glsclearpage</code>	28	<code>\glentrylongpl</code>	114
<code>\glsclosebrace</code>	117	<code>\glentrylongpluralaccess</code> ..	225
<code>\glscpositor</code>	24, 33	<code>\Glsentryname</code>	110
<code>\glscounter</code>	13, 42	<code>\glentryname</code>	110, 132
<code>\GlsDeclareNoHyperList</code>	13	<code>\glentrynumberlist</code>	114
<code>\glscdefaulttype</code>	11	<code>\Glsentryplural</code>	111
<code>\glscdefmain</code>	11	<code>\glentryplural</code>	111
<code>\GLSdesc</code>	88	<code>\glentrypluralaccess</code>	224
<code>\Glsdesc</code>	88	<code>\Glsentryshort</code>	113
<code>\glscdesc</code>	87, 88	<code>\glentryshort</code>	113
<code>\GLSdescplural</code>	90	<code>\glentryshortaccess</code>	225
<code>\Glsdescplural</code>	89	<code>\Glsentryshortpl</code>	113
<code>\glscdescplural</code>	89, 90	<code>\glentryshortpl</code>	113
<code>\glscdescriptionaccessdisplay</code>	227	<code>\glentryshortpluralaccess</code> ..	225
<code>\glscdescriptionpluralaccessdisplay</code>	227	<code>\glentrysort</code>	112
<code>\glscdescwidth</code>	180, 186, 194, 201	<code>\Glsentrysymbol</code>	111
<code>\glscdisablehyper</code>	70	<code>\glentrysymbol</code>	111
<code>\glscdisp</code>	79	<code>\glentrysymbolaccess</code>	225
<code>\glscdisplay</code>	14, 43, 44, 57, 70	<code>\Glsentrysymbolplural</code>	111
<code>\glscdisplayfirst</code> ..	43, 44, 57, 58, 70	<code>\glentrysymbolplural</code>	111
<code>\glscdisplaynumberlist</code>	114	<code>\glentrysymbolpluralaccess</code> ..	225
<code>\glscdoifexists</code>	39	<code>\Glsentrytext</code>	110
<code>\glscdoifnoexists</code>	40	<code>\glentrytext</code>	110, 132
<code>\glscenablehyper</code>	70	<code>\glentrytextaccess</code>	224
<code>\glscentryaccess</code>	224	<code>\glscentrytype</code>	112
<code>\glscentrycounter</code>	60	<code>\Glsentryuseri</code>	112
<code>\glscentrycounterlabel</code>	139	<code>\glscentryuseri</code>	112
<code>\Glsentrydesc</code>	110	<code>\Glsentryuserii</code>	112
<code>\glscentrydesc</code>	14, 110	<code>\glscentryuserii</code>	112
<code>\glscentrydescaccess</code>	225	<code>\Glsentryuseriii</code>	112
<code>\Glsentrydescplural</code>	110	<code>\glscentryuseriii</code>	112
<code>\glscentrydescplural</code>	110	<code>\Glsentryuseriv</code>	113
<code>\glscentrydescpluralaccess</code> ..	225	<code>\glscentryuseriv</code>	112
<code>\Glsentryfirst</code>	111	<code>\Glsentryuserv</code>	113
<code>\glscentryfirst</code>	111	<code>\glscentryuserv</code>	113
<code>\glscentryfirstaccess</code>	224	<code>\Glsentryuservi</code>	113
<code>\Glsentryfirstplural</code>	112	<code>\glscentryuservi</code>	113
<code>\glscentryfirstplural</code>	111	<code>\GLSfirst</code>	83
<code>\glscentryfirstpluralaccess</code> ..	225	<code>\Glsfirst</code>	82
<code>\Glsentryfull</code>	114	<code>\glscfirst</code>	82
<code>\glscentryfull</code>	114	<code>\glscfirstaccessdisplay</code>	226
		<code>\GLSfirstplural</code>	85

<code>\Glsfirstplural</code>	85	<code>\glsnamefont</code>	143
<code>\glsfirstplural</code>	84, 85	<code>\glsnavhyperlink</code>	173
<code>\glsfirstpluralaccessdisplay</code>	227	<code>\glsnavhypertarget</code>	173
<code>\glsgetgrouplabel</code>	142	<code>\glsnavigation</code>	174
<code>\glsgetgrouptitle</code>	117, 141	<code>\glsnextpages</code>	137
<code>\glsgroupheading</code>	141, 142	<code>\glsnonextpages</code>	137
<code>\glsgroupskip</code>	141, 142, 177	<code>\glsnoxindywarning</code>	28
<code>\glshyperlink</code>	116	<code>\glsnumberformat</code>	25
<code>\glshypernavsep</code>	174	<code>\glsnumbersgroupname</code> ..	21, 117, 141
<code>\glshypernumber</code>	25, 143	<code>\glsnumlistlastsep</code>	115
<code>\glsIfListOfAcronyms</code>	12	<code>\glsnumlistsep</code>	115
<code>\glsinlinedescformat</code>	176	<code>\glsopenbrace</code>	117
<code>\glsinlinedopostchild</code> ...	175, 176	<code>\glsorder</code>	17
<code>\glsinlineemptydescformat</code> ...	176	<code>\glsorg@endtheglossary</code>	5
<code>\glsinlinenameformat</code>	176	<code>\glsorg@glossary</code>	4
<code>\glsinlineparentchildseparator</code>		<code>\glsorg@theglossary</code>	5
.....	176	<code>\glsorg@wrglossary</code>	4
<code>\glsinlinepostchild</code>	176	<code>\glspagelistwidth</code>	180, 186, 195, 201
<code>\glsinlineseparator</code>	176	<code>\glspar</code>	23
<code>\glsinlinesubdescformat</code>	176	<code>\GLSpl</code>	77
<code>\glsinlinesubnameformat</code>	176	<code>\Glspl</code>	76, 170
<code>\glsinlinesubseparator</code>	176	<code>\glspl</code>	56, 57, 74, 76, 77
<code>\glskeylisttok</code>	150	<code>\GLSplural</code>	84
<code>\glslabeltok</code>	150	<code>\Glsplural</code>	83
<code>\glslink</code> ..	56, 57, 59, 70, 116, 142, 143	<code>\glsplural</code>	83, 84
<code>\glslink options</code>		<code>\glspluralaccessdisplay</code>	226
counter	58, 70, 170	<code>\glspluralsuffix</code>	21, 44
format	58, 70, 143	<code>\glspostdescription</code>	8
hyper	58, 70	<code>\glspostinline</code>	176
local	59	<code>\glsquote</code>	117
<code>\glslistdottedwidth</code>	179	<code>\glsrefentry</code>	139
<code>\glslocalreset</code>	55	<code>\glsreset</code>	55
<code>\glslocalresetall</code>	56	<code>\glsresetall</code>	56
<code>\glslocalunset</code>	55	<code>\glsresetentrylist</code>	137
<code>\glslocalunsetall</code>	56	<code>\glsresetsubentrycounter</code>	138
<code>\glslongaccessdisplay</code>	228	<code>\glssee</code>	131
<code>\glslongaccesskey</code>	241	<code>\glsseeformat</code>	119, 131
<code>\glslongkey</code>	147	<code>\glsseeitem</code>	132
<code>\glslongpluralaccessdisplay</code> .	228	<code>\glsseeitemformat</code>	132
<code>\glslongpluralaccesskey</code>	241	<code>\glsseelastsep</code>	132
<code>\glslongpluralkey</code>	147	<code>\glsseelist</code>	131
<code>\glslongtok</code>	150	<code>\glsseesep</code>	132
<code>\glsmakefirsttuc</code>	171	<code>\glsSetAlphaCompositor</code>	24
<code>\glsmcols</code>	191	<code>\glsSetCompositor</code>	23, 24
<code>\glsmoveentry</code>	52	<code>\glsSetSuffixF</code>	24
<code>\GLSname</code>	87	<code>\glsSetSuffixFF</code>	25
<code>\Glsname</code>	86	<code>\glssettoctitle</code>	20
<code>\glsname</code>	86, 87	<code>\glssetwidest</code>	211
<code>\glsnameaccessdisplay</code>	226	<code>\GlsSetXdyCodePage</code>	37

shortpluralaccess	225, 228	entrycounter	137
sort	14, 15, 44, 112, 141	true	8
symbol		entrycounter	8
14, 15, 43, 44, 57, 90, 111, 153,		entrycounterwithin	8
154, 156, 159, 183, 198, 221–223		footnote	71, 72, 74, 75,
symbolaccess	225, 227	77, 78, 80, 153, 155–157, 229–234	
symbolplural	92, 222	footnote	16
symbolpluralaccess	225, 227	hyperfirst	
text 44, 57, 70, 80, 110, 153, 156, 221		false	71, 72, 74, 75, 77, 78, 80
textaccess	224, 226	hyperfirst	16
type	11, 45, 56, 112	indexonlyfirst	263
user1	93, 112, 223	indexonlyfirst	16
user2	95, 112	makeindex	120, 170
user3	96, 112	nogroupskip	8
user4	98, 112	nohypertypes	14
user5	99, 113	nolist	165
user6	101, 113, 223	nolist	7
\newglossarystyle	142	nolong	165, 180
ngerman package	37	nolong	7
nogroupskip (option)	8	nomain	11
nohypertypes (option)	14	nonumberlist	7
\noist	123, 170, 221	nonumberlist	7
nolist (option)	7	noposted	8
nolong (option)	7	nostyles	8
nonumberlist (key)	45	nosuper	165
nonumberlist (option)	7	nosuper	7
\nopostdesc	23	notree	165
noposted (option)	8	notree	8
nostyles (option)	8	numberedsection	6
nosuper (option)	7	numberline	5
notree (option)	8	numberline	5
numberedsection (option)	6	order	17
numberline (option)	5	sanitize	14, 15, 43, 110, 111
		sanitize	15
O		savenuumberlist	7
\oldacronym	146	savewrites	18, 261
order (option)	17	false	124
		true	126
P		savewrites	18
package options:		section	5, 27
acronym 11, 12, 20, 136, 146, 147		section	5
acronym	12	seeautonumberlist	7
acronymlists	13	shotcuts	17
compatible-2.07	18	smallcaps	16
counter	13	smaller	16
counter	13	sort	
description	156	def	9
description	16	standard	9
dua	155, 156	use	9
dua	16		

sort	9	\SetDefaultAcronymDisplayStyle	151
style	6, 165	\SetDefaultAcronymStyle	151
style	6	\SetDescriptionAcronymDisplayStyle	155
subentrycounter	138	\SetDescriptionAcronymStyle	156
subentrycounter	9	\SetDescriptionDUAAcronymDisplayStyle	154
toc	5	\SetDescriptionDUAAcronymStyle	154
true	5	\SetDescriptionFootnoteAcronymDisplayStyle	152
toc	5	\SetDescriptionFootnoteAcronymStyle	153
translate	15	\SetDUADisplayStyle	160
translate	15	\SetDUASStyle	160
ucmark	8	\setentrycounter	142
xindy	17, 120, 170	\SetFootnoteAcronymDisplayStyle	156
\pagelistname	20	\SetFootnoteAcronymStyle	157
parent (key)	45	\setglossarysection	27
\phantomsection	26, 27	\SetSmallAcronymDisplayStyle	158
plural (key)	44	\SetSmallAcronymStyle	159
pluralaccess (key)	222	\setstylefile	23
polyglossia package	19, 21	short (key)	46
\PrintChanges	5	shortaccess (key)	222
\printglossaries 11, 25, 40, 43, 125, 133, 136, 172	shortplural (key)	46
\printglossary	25, 26, 41, 125, 133, 135, 136, 172, 173	shortpluralaccess (key)	222
\printglossary options		shotcuts (option)	17
nonumberlist	137	\showacronymlists	168
numberedsection	136	\showglocounter	166
style	136	\showglodesc	167
title	136	\showglodescaccess	242
toctitle	136	\showglodescplural	168
type	11, 132, 136	\showglodescpluralaccess	242
\protect	14	\showglofirst	166
		\showglofirstaccess	241
		\showglofirststpl	166
		\showglofirststplpluralaccess	241
		\showgloflag	168
		\showgloindex	168
		\showglolevel	165
		\showglolongaccess	242
		\showglolongpluralaccess	242
		\showgloname	167
		\showglonameaccess	241
		\showgloparent	165
		\showgloplural	166
		\showglopluralaccess	241
R			
\renewglossarystyle	143		
\roman	32		
\rootlanguage	36, 37		
S			
sanitize (option)	15		
savenumberlist (option)	7		
savewrites (option)	18		
section (option)	5		
see (key)	45		
seeautonumberlist (option)	7		
\seename	21		
\SetAcronymLists	13		
\SetAcronymStyle	12, 161		
\SetCustomDisplayStyle	162		
\SetCustomStyle	162		

<code>\showgloshortaccess</code>	242	<code>superraggedheader (style)</code>	202
<code>\showgloshortpluralaccess</code> ...	242	<code>superraggedheaderborder (style)</code>	203
<code>\showglosort</code>	168	<code>superraggedright3colheaderborder</code>	
<code>\showglossaries</code>	169	(style)	204
<code>\showglossarycounter</code>	169	<code>supertabular (environment)</code>	
<code>\showglossaryentries</code>	169	7, 165, 194–206
<code>\showglossaryin</code>	169	<code>supertabular package</code> ...	7, 165, 194, 201
<code>\showglossaryout</code>	169	<code>symbol (key)</code>	44
<code>\showglossarytitle</code>	169	<code>symbolaccess (key)</code>	222
<code>\showglosymbol</code>	168	<code>\symbolname</code>	20
<code>\showglosymbolaccess</code>	241	<code>symbolplural (key)</code>	45
<code>\showglosymbolplural</code>	168	<code>symbolpluralaccess (key)</code>	222
<code>\showglosymbolpluralaccess</code> ..	242		
<code>\showglotext</code>	165	T	
<code>\showglotextaccess</code>	241	<code>text (key)</code>	44
<code>\showglotype</code>	166	<code>textaccess (key)</code>	221
<code>\showglouseri</code>	166	<code>\theequation</code>	130
<code>\showglouseriii</code>	167	<code>theglossary (environment)</code> ..	4, 5,
<code>\showglouseriii</code>	167	25, 140, 142, 192, 193, 208, 209, 211	
<code>\showglouseriv</code>	167	<code>\theHequation</code>	130
<code>\showglouserv</code>	167	<code>theindex (environment)</code>	207
<code>\showglouservi</code>	167	<code>toc (option)</code>	5
<code>smallcaps (option)</code>	16	<code>\translate</code>	21
<code>smaller (option)</code>	16	<code>translate (option)</code>	15
<code>\SmallNewAcronymDef</code>	158, 240	<code>translator package</code>	
<code>sort (key)</code>	44	18, 21, 133, 242, 252–255
<code>sort (option)</code>	9	<code>tree (style)</code>	208
<code>style (option)</code>	6	<code>treegroup (style)</code>	209
<code>subentrycounter (option)</code>	9	<code>treehypergroup (style)</code>	209
<code>\subitem</code>	207	<code>treenoname (style)</code>	210
<code>sublistdotted (style)</code>	179	<code>treenonamegroup (style)</code>	210
<code>\subsubitem</code>	207	<code>treenonamehypergroup (style)</code> ...	211
<code>super (style)</code>	195	<code>type (key)</code>	45
<code>super3col (style)</code>	196		
<code>super3colborder (style)</code>	197	U	
<code>super3colheader (style)</code>	197	<code>ucmark (option)</code>	8
<code>super3colheaderborder (style)</code> ..	197	<code>user1 (key)</code>	46
<code>super4col (style)</code>	198	<code>user2 (key)</code>	46
<code>super4colborder (style)</code>	199	<code>user3 (key)</code>	46
<code>super4colheader (style)</code>	198	<code>user4 (key)</code>	46
<code>super4colheaderborder (style)</code> ..	199	<code>user5 (key)</code>	46
<code>superborder (style)</code>	195	<code>user6 (key)</code>	46
<code>superheader (style)</code>	196		
<code>superheaderborder (style)</code>	196	W	
<code>superragged (style)</code>	201	<code>\warn@nomakeglossaries</code>	124
<code>superragged3col (style)</code>	203	<code>\warn@noprintglossary</code>	133
<code>superragged3colborder (style)</code> ..	204	<code>\writeist</code> .	23, 30, 31, 34, 118, 215, 217
<code>superragged3colheader (style)</code> ..	204		
<code>superraggedborder (style)</code>	202	X	
		<code>\xcapitalisewords</code>	172

\xglsacccsupp	226	129, 130, 133, 135, 141, 170, 216
xindy	255	
xindy	17, 18, 23, 24, 28,	\xmakefirstuc 172
	32, 34–38, 53, 54, 68, 117–119,	xspace package 4, 146