

glossaries-extra.sty v1.06: documented code

Nicola L.C. Talbot

Dickimaw Books

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

2016-06-18

Abstract

This is the documented code for the glossaries-extra package. See [glossaries-extra-manual.pdf](#) for the user manual.

This package is experimental and not stable. It's provided for testing purposes only. Future versions may not be compatible with this version. Once it has stabilised I'll add it to CTAN, at which point compatibility with the first stable version will be maintained.

Contents

1 Main Package Code (glossaries-extra.sty)	4
1.1 Package Initialisation and Options	4
1.2 Extra Utilities	12
1.3 Modifications to Commands Provided by glossaries	12
1.3.1 Existence Checks	12
1.3.2 Document Definitions	16
1.3.3 Existing Glossary Style Modifications	20
1.3.4 Entry Formatting, Hyperlinks and Indexing	24
1.3.5 Entry Counting	46
1.3.6 Acronym Modifications	59
1.3.7 Indexing and Displaying Glossaries	61
1.4 Integration with glossaries-accsupp	70
1.5 Categories	81
1.6 Abbreviations	103
1.6.1 Abbreviation Styles Setup	120
1.6.2 Predefined Styles (Default Font)	123
1.6.3 Predefined Styles (Small Capitals)	135
1.6.4 Predefined Styles (Fake Small Capitals)	139
1.6.5 Predefined Styles (Emphasized)	143
1.6.6 Predefined Styles (User Parentheses Hook)	149
1.7 Using Entries in Headings	153
1.8 Multi-Lingual Support	170
2 Style Adjustments (glossaries-extra-stylemods.sty)	171
2.1 Package Initialisation	171
2.2 List-Like Styles	172
2.3 Longtable Styles	172
2.4 Long Ragged Styles	173
2.5 Supertabular Styles	175
2.6 Super Ragged Styles	176
2.7 Inline Style	177
2.8 Tree Styles	177
Glossary	189
Change History	190
Index	198

1 Main Package Code (glossaries-extra.sty)

1.1 Package Initialisation and Options

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries-extra}[2016/06/18 v1.06 (NLCT)]

Requires xkeyval to define package options.
3 \RequirePackage{xkeyval}

Requires etoolbox package.
4 \RequirePackage{etoolbox}

Has glossaries already been loaded?
5 \@ifpackageloaded{glossaries}
6 {%

Already loaded so pass any options to \setupglossaries. This means that the options that
can only be set when glossaries is loaded can't be used.
7   \newcommand{\glstr@dooption}[1]{\setupglossaries{#1}}%
8   \let\@glstr@declareoption\@gls@declareoption
9 }
10 {%

Not already loaded, so pass options to glossaries.
11   \newcommand{\glstr@dooption}[1]{%
12     \PassOptionsToPackage{#1}{glossaries}%
13   }%

Set the defaults.
14   \PassOptionsToPackage{toc}{glossaries}
15   \PassOptionsToPackage{nopostdot}{glossaries}
16   \PassOptionsToPackage{noredefwarn}{glossaries}
17   \@ifpackageloaded{polyglossia}%
18   {%
19   {%
20     \@ifpackageloaded{babel}%
21     {\PassOptionsToPackage{translate=babel}{glossaries}}%
22     {}%
23   }%
24   \newcommand*{\@glstr@declareoption}[2]{%
25     \DeclareOptionX{#1}{#2}%
26     \DeclareOption{#1}{#2}%
27   }
28 }
```

Declare package options.

`\glxtrundefaction` Determines what to do if an entry hasn't been defined. The two arguments are the error or warning message and the help message if an error should be produced.

```
29 \newcommand*{\glxtrundefaction}[2]{%
30   \@glxtrundeftag\PackageError{glossaries-extra}{#1}{#2}%
31 }
```

`\warnonexistsordo` If user wants `undefaction=warn`, then `glossaries v4.19` is required.

```
32 \newcommand*{\glxtr@warnonexistsordo}[1]{}
```

`\glxtrundeftag` Text to display when an entry doesn't exist.

```
33 \newcommand*{\glxtrundeftag}{??}
34 \newcommand*{\@glxtrundeftag}{}
```

This text is switched on at the start of the document to prevent unwanted text inserted into the preamble if any tests are made before the start of the document.

```
35 \define@choicekey{glossaries-extra.sty}{undefaction}[\val\nr]%
36 {warn,error}%
37 {%
38   \ifcase\nr\relax
39     \renewcommand*{\glxtrundefaction}[2]{%
40       \@glxtrundeftag\GlossariesExtraWarning{##1}%
41     }%
42     \renewcommand*{\glxtr@warnonexistsordo}[1]{%
43       \GlossariesExtraWarning{glossaries-extra}{%
44         \string##1\space hasn't been defined, so
45         some errors won't be converted to warnings.
46         (This most likely means your version of
47         glossaries.sty is below version 4.19.)}%
48       }%
49     \or
50       \renewcommand*{\glxtrundefaction}[2]{%
51         \@glxtrundeftag\PackageError{glossaries-extra}{##1}{##2}%
52       }%
53       \renewcommand*{\glxtr@warnonexistsordo}[1]{}%
54     \fi
55 }
```

Version 1.06 changes the `docdef` option to a choice rather than boolean setting. The available values are: `false`, `true` or `restricted`. The `restricted` option permits document definitions as long as they occur before the first glossary is displayed.

`\glxtr@docdefval` The `docdef` value is stored as an integer: 0 (`false`), 1 (`true`) and 2 (`restricted`).

```
56 \newcount\@glxtr@docdefval
```

Need to provide conditional commands that are backward compatible:

`\if@glxtrdocdef`

```
57 \newcommand*{\if@glxtrdocdef}{\ifnum\@glxtr@docdefval>0 }
```

lsxtrdocdeftrue

```
58 \newcommand*{\@glxtrdocdeftrue}{\@glxtr@docdefval=1 }
```

lsxtrdocdeffalse

```
59 \newcommand*{\@glxtrdocdeffalse}{\@glxtr@docdefval=0 }
```

By default don't allow entries to be defined in the document to encourage the user to define them in the preamble, but if the user is really determined to define them in the document allow them to request this.

```
60 \define@choicekey{glossaries-extra.sty}{docdef}[\val\nr]%  
61 {false,true,restricted}[true]%  
62 {%  
63   \@glxtr@docdefval=\nr\relax  
64   \ifnum\@glxtr@docdefval=2\relax  
65     \renewcommand*{\@glsdofexistsorwarn}{\glsdofexists}%  
66   \fi  
67 }
```

ocdefrestricted

```
68 \newcommand*{\if@glxtrdocdefrestricted}{\ifnum\@glxtr@docdefval=2 }
```

oifexistsorwarn

Need an error to notify user if an undefined entry is being referenced in the glossary for the docdef=restricted option. This is used by \glossentryname (but not by \glossentrydesc etc as one error per entry is sufficient).

```
69 \newcommand*{\@glsdofexistsorwarn}{\glsdofexistsorwarn}
```

indexcrossrefs

Automatically index cross references at the end of the document

```
70 \define@boolkey{glossaries-extra.sty}[\@glxtr]{indexcrossrefs}[true]{%  
71   \if@glxtrindexcrossrefs  
72   \else  
73     \renewcommand*{\@glxtr@autoindexcrossrefs}{}%  
74   \fi  
75 }
```

Switch off since this can increase the build time.

```
76 \@glxtrindexcrossrefsfalse
```

But allow see key to switch it on automatically.

oindexcrossrefs

```
77 \newcommand*{\@glxtr@autoindexcrossrefs}{\@glxtrindexcrossrefstrue}
```

iesExtraWarning

Allow users to suppress warnings.

```
78 \newcommand*{\GlossariesExtraWarning}[1]{\PackageWarning{glossaries-extra}{#1}}
```

raWarningNoLine

Allow users to suppress warnings.

```
79 \newcommand*{\GlossariesExtraWarningNoLine}[1]{%  
80   \PackageWarningNoLine{glossaries-extra}{#1}}
```

```

81 \@glsxtr@declareoption{nowarn}{%
82   \let\GlossariesExtraWarning\@gobble
83   \let\GlossariesExtraWarningNoLine\@gobble
84   \glsxtr@doption{nowarn}%
85 }

glsxtrabbrvtype Glossary type for abbreviations.
86 \newcommand*{\glsxtrabbrvtype}{\glsdefaulttype}

abbreviationsdef Set by abbreviations option.
87 \newcommand*{\@glsxtr@abbreviationsdef}{}

abbreviationsdef
88 \newcommand*{\@glsxtr@doabbreviationsdef}{%
89   \ifpackageloaded{babel}%
90     {\providecommand{\abbreviationsname}{\acronymname}}%
91     {\providecommand{\abbreviationsname}{Abbreviations}}%
92   \newglossary[glg-abr]{abbreviations}{gls-abr}{glo-abr}{\abbreviationsname}%
93   \renewcommand*{\glsxtrabbrvtype}{abbreviations}%
94   \newcommand*{\printabbreviations}[1][1]{%
95     \printglossary[type=\glsxtrabbrvtype,##1]%
96   }%
97   \disable@keys{glossaries-extra.sty}{abbreviations}%

If the acronym option hasn't been used, change \acronymtype to \glsxtrabbrvtype.
98   \ifglsacronym
99   \else
100     \renewcommand*{\acronymtype}{\glsxtrabbrvtype}%
101   \fi
102 }%

abbreviations If abbreviations, create a new glossary type for abbreviations.
103 \@glsxtr@declareoption{abbreviations}{%
104   \let\@glsxtr@abbreviationsdef\@glsxtr@doabbreviationsdef
105 }

iationShortcuts Enable shortcut commands for the abbreviations. Unlike the analogous command provided
by glossaries, this uses \newcommand instead of \let as a safety feature.
106 \newcommand*{\GlsXtrDefineAbbreviationShortcuts}{%
107   \newcommand*{\ab}{\cglsl}%
108   \newcommand*{\abp}{\cglspl}%
109   \newcommand*{\as}{\glsxtrshort}%
110   \newcommand*{\asp}{\glsxtrshortpl}%
111   \newcommand*{\al}{\glsxtrlong}%
112   \newcommand*{\alp}{\glsxtrlongpl}%
113   \newcommand*{\af}{\glsxtrfull}%
114   \newcommand*{\afp}{\glsxtrfullpl}%
115   \newcommand*{\Ab}{\cGls}%
116   \newcommand*{\Abp}{\cGlspl}%

```

```

117 \newcommand*{\As}{\Glsxtrshort}%
118 \newcommand*{\Asp}{\Glsxtrshortpl}%
119 \newcommand*{\Al}{\Glsxtrlong}%
120 \newcommand*{\Alp}{\Glsxtrlongpl}%
121 \newcommand*{\Af}{\Glsxtrfull}%
122 \newcommand*{\Afp}{\Glsxtrfullpl}%
123 \newcommand*{\AB}{\cGLS}%
124 \newcommand*{\ABP}{\cGLSp1}%
125 \newcommand*{\AS}{\GLSxtrshort}%
126 \newcommand*{\ASP}{\GLSxtrshortpl}%
127 \newcommand*{\AL}{\GLSxtrlong}%
128 \newcommand*{\ALP}{\GLSxtrlongpl}%
129 \newcommand*{\AF}{\GLSxtrfull}%
130 \newcommand*{\AFP}{\GLSxtrfullpl}%
131 \newcommand*{\newabbr}{\newabbreviation}%

Disable this command after it's been used.
132 \let\GlsXtrDefineAbbreviationShortcuts\relax
133 }

```

@OtherShortcuts Similarly provide shortcut versions for the commands provided by the symbols and numbers options.

```

134 \newcommand*{\GlsXtrDefineOtherShortcuts}{%
135   \newcommand*{\newentry}{\newglossaryentry}%
136   \ifdef\printsymbols
137     {%
138       \newcommand*{\newsym}{\glxtrnewsymbol}%
139     }{}%
140   \ifdef\printnumbers
141     {%
142       \newcommand*{\newnum}{\glxtrnewnumber}%
143     }{}%
144   \let\GlsXtrDefineOtherShortcuts\relax
145 }

```

Always use the long forms, not the shortcuts, where portability is an issue. (For example, when defining entries in a file that may be input by multiple documents.)

@setupshortcuts Command used to set the shortcuts option.

```

146 \newcommand*{\@glxtr@setupshortcuts}{}

```

Provide shortcuts option. Unlike the glossaries version, this is a choice rather than a boolean key but it also provides `shortcuts=true` and `shortcuts=false`, which are equivalent to `shortcuts=all` and `shortcuts=none`. Multiple use of this option in the *same* option list will override each other.

```

147 \define@choicekey{glossaries-extra.sty}{shortcuts}[\val\nr]%
148 {acronyms,acro,abbreviations,abbr,other,all,true,none,false}[true]{%
149   \ifcase\nr\relax % acronyms
150     \renewcommand*{\@glxtr@setupshortcuts}{%

```



```

151     \glsacrshortcutstrue
152     \DefineAcronymSynonyms
153 }%
154 \or % acro
155     \renewcommand*{\@glsxtr@setupshortcuts}{%
156         \glsacrshortcutstrue
157         \DefineAcronymSynonyms
158     }%
159 \or % abbreviations
160     \renewcommand*{\@glsxtr@setupshortcuts}{%
161         \GlsXtrDefineAbbreviationShortcuts
162     }%
163 \or % abbr
164     \renewcommand*{\@glsxtr@setupshortcuts}{%
165         \GlsXtrDefineAbbreviationShortcuts
166     }%
167 \or % other
168     \renewcommand*{\@glsxtr@setupshortcuts}{%
169         \GlsXtrDefineOtherShortcuts
170     }%
171 \or % all
172     \renewcommand*{\@glsxtr@setupshortcuts}{%
173         \glsacrshortcutstrue
174         \DefineAcronymSynonyms
175         \GlsXtrDefineAbbreviationShortcuts
176         \GlsXtrDefineOtherShortcuts
177     }%
178 \or % true
179     \renewcommand*{\@glsxtr@setupshortcuts}{%
180         \glsacrshortcutstrue
181         \DefineAcronymSynonyms
182         \GlsXtrDefineAbbreviationShortcuts
183         \GlsXtrDefineOtherShortcuts
184     }%
185 \else % none, false
186     \renewcommand*{\@glsxtr@setupshortcuts}{}%
187 \fi
188 }

```

lsxtr@doaccsupp

```
189 \newcommand*{\@glsxtr@doaccsupp}{}

```

accsupp If accsupp, load glossaries-accsupp package.

```

190 \@glsxtr@declareoption{accsupp}{%
191     \renewcommand*{\@glsxtr@doaccsupp}{\RequirePackage{glossaries-accsupp}}}

```

GlossaryWarning Warning text displayed in document if the external glossary file given by the argument is missing.

```
192 \newcommand{\glsxtrNoGlossaryWarning}[1]{%

```

```

193 \@glsxtr@defaultnoglossarywarning{#1}%
194 }

```

omissingglstext If true, suppress the text produced if the external glossary file is missing.

```

195 \define@choicekey{glossaries-extra.sty}{nomissingglstext}[\val\nr]%
196 {true,false}[true]{%
197   \ifcase\nr\relax % true
198     \renewcommand{\glsxtrNoGlossaryWarning}[1]{%
199       \null
200     }%
201   \else % false
202     \renewcommand{\glsxtrNoGlossaryWarning}[1]{%
203       \@glsxtr@defaultnoglossarywarning{#1}%
204     }%
205   \fi
206 }

```

Provide option to load glossaries-extra-stylemods (Deferred to the end.)

xtr@redefstyles

```

207 \newcommand*{\@glsxtr@redefstyles}{}

```

stylemods

```

208 \define@key{glossaries-extra.sty}{stylemods}{%
209   \ifblank{#1}%
210   {%
211     \renewcommand*{\@glsxtr@redefstyles}{%
212       \RequirePackage{glossaries-extra-stylemods}}%
213   }%
214   {%
215     \renewcommand*{\@glsxtr@redefstyles}{}%
216     \@for\@glsxtr@tmp:=#1\do{%
217       \IfFileExists{glossary-\@glsxtr@tmp.sty}%
218       {%
219         \eappto\@glsxtr@redefstyles{%
220           \noexpand\RequirePackage{glossary-\@glsxtr@tmp}}%
221       }%
222       {%
223         \PackageError{glossaries-extra}%
224           {Glossaries style package ‘glossary-\@glsxtr@tmp.sty’
225            doesn’t exist (did you mean to use the ‘style’ key?)}%
226           {The list of values (#1) in the ‘stylemods’ key should
227            match the glossary-xxx.sty files provided with
228            glossaries.sty}%
229       }%
230     }%
231     \appto\@glsxtr@redefstyles{\RequirePackage{glossaries-extra-stylemods}}%
232   }%
233 }

```

glsxtr@do@style

```
234 \newcommand*{\@glsxtr@do@style}{}
```

style Since the stylemods option can automatically load extra style packages, deal with the style option after those packages have been loaded.

```
235 \define@key{glossaries-extra.sty}{style}{%
```

```
236 \renewcommand*{\@glsxtr@do@style}{%
```

Set this as the default style:

```
237 \setkeys{glossaries.sty}{style={#1}}%
```

Set this style:

```
238 \setglossarystyle{#1}%
```

```
239 }%
```

```
240 }
```

Pass all other options to glossaries.

```
241 \DeclareOptionX*{%
```

```
242 \expandafter\glsxtr@doooption\expandafter{\CurrentOption}}
```

Process options.

```
243 \ProcessOptionsX
```

Load glossaries if not already loaded.

```
244 \RequirePackage{glossaries}
```

Load the glossaries-accsupp package if required.

```
245 \@glsxtr@doaccsupp
```

Define abbreviations glossaries if required.

```
246 \@glsxtr@abbreviationsdef
```

```
247 \let\@glsxtr@abbreviationsdef\relax
```

Setup shortcuts if required.

```
248 \@glsxtr@setupshortcuts
```

ariesextrasetup Allow user to set options after the package has been loaded. First modify \glsxtr@doooption so that it now uses \setupglossaries:

```
249 \renewcommand{\@glsxtr@doooption}[1]{\setupglossaries{#1}}%
```

Now define the user command:

```
250 \newcommand*{\glossariesextrasetup}[1]{%
```

```
251 \let\@glsxtr@setupshortcuts\relax
```

```
252 \setkeys{glossaries-extra.sty}{#1}%
```

```
253 \@glsxtr@abbreviationsdef
```

```
254 \let\@glsxtr@abbreviationsdef\relax
```

```
255 \@glsxtr@setupshortcuts
```

```
256 }
```

Disable preamble-only options and switch on the undefined tag at the start of the document.

```
257 \AtBeginDocument{%
258   \disable@keys{glossaries-extra.sty}{abbreviations,docdef}%
259   \def\@glsxtrundeftag{\glsxtrundeftag}%
260 }
```

1.2 Extra Utilities

`\glsxtrifemptyglossary`

```
\glsxtrifemptyglossary{<type>}{<true>}{<false>}
```

Provide command to determine if any entries have been added to the glossary (where the glossary label is provided in the first argument). The entries are stored in the comma-separated list `\glolist@<type>`. If this hasn't been defined, the glossary doesn't exist. If it has been defined and is simply a comma, the glossary exists and is empty. (It's initialised to a comma.)

```
261 \newcommand{\glsxtrifemptyglossary}[3]{%
262   \ifglossaryexists{#1}%
263   {%
264     \ifcsstring{glolist@#1}{,}{#2}{#3}%
265   }%
266   {%
267     \glsxtrundefaction{Glossary type '#1' doesn't exist}{}%
268     #2%
269   }%
270 }
```

1.3 Modifications to Commands Provided by glossaries

Some of the commands provided by glossaries are modified to take into account new options or to change default behaviour.

1.3.1 Existence Checks

`\glsdoifexists` Modify `\glsdoifexists` to take account of the undefaction setting.

```
271 \renewcommand{\glsdoifexists}[2]{%
272   \ifglsentryexists{#1}{#2}%
273   {%
274     \glsxtrundefaction{Glossary entry '\glsdetoklabel{#1}'
275       has not been defined}{You need to define a glossary entry before
276       you can reference it.}%
277   }%
278 }
```

`glsdoifnoexists` Modify `\glsdoifnoexists` to take account of the undefaction setting.

```
279 \renewcommand{\glsdoifnoexists}[2]{%
280   \ifglstryexists{#1}{%
281     \glstrundefaction{Glossary entry ‘\glsdetoklabel{#1}’
282       has already been defined}{}}{#2}%
283 }
```

`glsdoifexistsordo` Modify `\glsdoifexistsordo` to take account of the undefaction setting. This command was introduced in glossaries version 4.19, so check if it has been defined first.

```
284 \ifdef\glsdoifexistsordo
285 {%
286   \renewcommand{\glsdoifexistsordo}[3]{%
287     \ifglstryexists{#1}{#2}%
288     {%
289       \glstrundefaction{Glossary entry ‘\glsdetoklabel{#1}’
290         has not been defined}{You need to define a glossary entry
291         before you can use it.}%
292       #3%
293     }%
294   }%
295 }
296 {%
297   \glstr@warnonexistsordo\glsdoifexistsordo
298   \newcommand{\glsdoifexistsordo}[3]{%
299     \ifglstryexists{#1}{#2}%
300     {%
301       \glstrundefaction{Glossary entry ‘\glsdetoklabel{#1}’
302         has not been defined}{You need to define a glossary entry
303         before you can use it.}%
304       #3%
305     }%
306   }%
307 }
```

`glsarynoexistsordo` Similarly for `\doifglossarynoexistsordo`.

```
308 \ifdef\doifglossarynoexistsordo
309 {%
310   \renewcommand{\doifglossarynoexistsordo}[3]{%
311     \ifglossaryexists{#1}%
312     {%
313       \glstrundefaction{Glossary type ‘#1’ already exists}{}%
314       #3%
315     }%
316     {#2}%
317   }%
318 }
319 {%
320   \glstr@warnonexistsordo\doifglossarynoexistsordo
321   \newcommand{\doifglossarynoexistsordo}[3]{%

```

```

322 \ifglossaryexists{#1}%
323 {%
324 \glxtrundefaction{Glossary type ‘#1’ already exists}{}%
325 #3%
326 }%
327 {#2}%
328 }%
329 }
330

```

`ryentryposthook` Hook into end of `\newglossaryentry` to add “see” value as a field.

```

331 \appto\@newglossaryentryposthook{%
332 \ifdefvoid\@glo@see
333 {\csxdef{glo@\@glo@label @see}{}}%
334 {%
335 \csxdef{glo@\@glo@label @see}{\@glo@see}%
336 \@glxtr@autoindexcrossrefs
337 }%
338 }
339 \appto\@gls@keymap{,{see}{see}}

```

`\glxtrusesee` Apply `\glsseeformat` to the see key if not empty.

```

340 \newcommand*{\glxtrusesee}[1]{%
341 \glsdoifexists{#1}%
342 {%
343 \letcs{\@glo@see}{glo@\glsdetoklabel{#1}@see}%
344 \ifdefempty\@glo@see
345 {}%
346 {%
347 \expandafter\glxtr@usesee\@glo@see\@end@glxtr@usesee
348 }%
349 }%
350 }

```

`\glxtr@usesee`

```

351 \newcommand*{\glxtr@usesee}[1][\seename]{%
352 \@glxtr@usesee[#1]%
353 }

```

`\@glxtr@usesee`

```

354 \def\@glxtr@usesee[#1]#2\@end@glxtr@usesee{%
355 \glxtruseseeformat{#1}{#2}%
356 }

```

`xtruseseeformat` The format used by `\glxtrusesee`. The first argument is the tag (such as `\seename`). The second argument is the comma-separated list of cross-referenced labels.

```

357 \newcommand*{\glxtruseseeformat}[2]{%
358 \glsseeformat[#1]{#2}{}%
359 }

```

Add all unused cross-references at the end of the document.

```
360 \AtEndDocument{\ifglxtrindexcrossrefs\glxtraddallcrossrefs\fi}
```

addallcrossrefs Iterate through all used entries and if they have a cross-reference, make sure the cross-reference has been added.

```
361 \newcommand*{\glxtraddallcrossrefs}{%
362   \forallglossaries{\@glo@type}%
363   {%
364     \forlgsentries[\@glo@type]{\@glo@label}%
365     {%
366       \ifglused{\@glo@label}{\@glxtr@addunusedxrefs{\@glo@label}}{%
367       }%
368     }%
369 }
```

@addunusedxrefs If the given entry has a see field add all unused cross-references.

```
370 \newcommand*{\@glxtr@addunusedxrefs}[1]{%
371   \letcs{\@glo@see}{glo@\glsetoklabel{#1}@see}%
372   \ifdefvoid\@glo@see
373   {%
374   {%
375     \expandafter\glxtr@addunused\@glo@see\@end@glxtr@addunused
376   }%
377 }
```

lsxtr@addunused Adds all the entries if they haven't been used.

```
378 \newcommand*{\glxtr@addunused}[1][]{%
379   \@glxtr@addunused
380 }
```

lsxtr@addunused Adds all the entries if they haven't been used.

```
381 \def\@glxtr@addunused#1\@end@glxtr@addunused{%
382   \@for\@glxtr@label:=#1\do
383   {%
384     \ifglused{\@glxtr@label}{}%
385     {%
386       \gladd[format=glxtrunusedformat]{\@glxtr@label}%
387       \glunset{\@glxtr@label}%
388       \@glxtr@addunusedxrefs{\@glxtr@label}%
389     }%
390   }%
391 }
```

xtrunusedformat

```
392 \newcommand*{\glxtrunusedformat}[1]{\unskip}
```

1.3.2 Document Definitions

`noidxglossaries` Modify `\makenoidxglossaries` so that it automatically switches off (unless the restricted setting is on) and disables the `docdef` key.

```
393 \let\glstr@orgmakenoidxglossaries\makenoidxglossaries
394 \renewcommand{\makenoidxglossaries}{%
395   \glstr@orgmakenoidxglossaries
396   \if@glstrdocdefrestricted
```

If restricted document definitions allowed, adjust `\@gls@reference` so that it doesn't test for existence.

```
397   \renewcommand*{\@gls@reference}[3]{%
398     \ifcsundef{@glsref@##1}{\csgdef{@glsref@##1}{}}{}%
399     \ifinlistcs{##2}{@glsref@##1}%
400     {}%
401     {\listcsgadd{@glsref@##1}{##2}}%
402     \ifcsundef{glo@glstetoklabel{##2}@loclist}%
403     {\csgdef{glo@glstetoklabel{##2}@loclist}{}}%
404     {}%
405     \listcsgadd{glo@glstetoklabel{##2}@loclist}{##3}%
406   }%
407 \else
```

Disable document definitions.

```
408   \@glstrdocdeffalse
409 \fi
410 \disable@keys{glossaries-extra.sty}{docdef}%
411 }
```

`ewglossaryentry` Modify `\gls@defdocnewglossaryentry` so that it checks the `docdef` value.

```
412 \renewcommand*{\gls@defdocnewglossaryentry}{%
413   \ifcase\@glstr@docdefval
414   docdef=false:
414     \renewcommand*{\newglossaryentry}[2]{%
415       \PackageError{glossaries-extra}{Glossary entries must
416       be \MessageBreak defined in the preamble with \MessageBreak
417       package option 'docdef=false'\MessageBreak(consider using
418       'docdef=restricted')}{Move your glossary definitions to
419       the preamble. You can also put them in a \MessageBreak separate file
420       and load them with \string\loadglsentries.}%
421     }%
422   \or
```

`docdef=true` Since the `see` value is now saved in a field, it can be used by entries that have been defined in the document.

```
423   \let\gls@checkseeallowed\relax
424   \let\newglossaryentry\new@glossaryentry
425 \or
```


Restricted mode just needs to allow the see value.

```
426 \let\gls@checkseeallowed\relax
427 \fi
428 }%
```

Permit a special form of document definition, but only allow it if the glossaries come at the end of the document. These commands behave a little like a combination of `\newterm` and `\gls`. This must be explicitly enabled with the following.

`\rEnableOnTheFly`

```
429 \newcommand*{\GlsXtrEnableOnTheFly}{%
430 \ifstar\@sGlsXtrEnableOnTheFly\@GlsXtrEnableOnTheFly
431 }
```

`\rEnableOnTheFly`

The starred version attempts to allow UTF8 characters in the label, but this may break! (Formatting commands mustn't be used in the label, but the label may be a command whose replacement text is the actual label. This doesn't take into account a command that's defined in terms of another command that may eventually expand to the label text.)

```
432 \newcommand*{\@sGlsXtrEnableOnTheFly}{%
433 \renewcommand*{\glsdetoklabel}[1]{%
434 \expandafter\@glsxtr@ifcsstart\string##1 \@glsxtr@end@
435 {%
436 \expandafter\detokenize\expandafter{##1}%
437 }%
438 {\detokenize{##1}}}%
439 }%
440 \@GlsXtrEnableOnTheFly
441 }
442 \def\@glsxtr@ifcsstart#1#2\@glsxtr@end@#3#4{%
443 \expandafter\if\glsbackslash#1%
444 #3%
445 \else
446 #4%
447 \fi
448 }
```

`\sXtrstarflywarn`

```
449 \newcommand*{\glsxtrstarflywarn}{%
450 \GlossariesExtraWarning{Experimental starred version of
451 \string\GlsXtrEnableOnTheFly\space in use (please ensure you have
452 read the warnings in the glossaries-extra user manual)}}%
453 }
```

`\rEnableOnTheFly`

```
454 \newcommand*{\@GlsXtrEnableOnTheFly}{%
```

Don't redefine `\glsdetoklabel` if LuaTeX or XeTeX is being used, since it's mainly to allow accented characters in the label.

These definitions are all assigned the category given by:

```

\glstrcat
455 \newcommand*\glstrcat{general}

\glstr
456 \newcommand*\glstr[1] [] {%
457 \def\glstr@keylist{##1}%
458 \@glstr
459 }

\@glstr
460 \newcommand*\@glstr[2] [] {%
461 \ifglstryexists{##2}%
462 {%
463 \ifblank{##1}{-}{\GlsXtrWarning{##1}{##2}}%
464 }%
465 {%
466 \gls@defglossaryentry{##2}{name={##2},category=\glstrcat,
467 description={\nopostdesc},##1}%
468 }%
469 \expandafter\gls\expandafter[\glstr@keylist]{##2}%
470 }

\Glsxtr
471 \newcommand*\Glsxtr[1] [] {%
472 \def\glstr@keylist{##1}%
473 \@Glsxtr
474 }

\@Glsxtr
475 \newcommand*\@Glsxtr[2] [] {%
476 \ifglstryexists{##2}%
477 {%
478 \ifblank{##1}{-}{\GlsXtrWarning{##1}{##2}}%
479 }%
480 {%
481 \gls@defglossaryentry{##2}{name={##2},category=\glstrcat,
482 description={\nopostdesc},##1}%
483 }%
484 \expandafter\Gls\expandafter[\glstr@keylist]{##2}%
485 }

\glstrpl
486 \newcommand*\glstrpl[1] [] {%
487 \def\glstr@keylist{##1}%
488 \@glstrpl
489 }

\@glstrpl

```

```

490 \newcommand*{\@glxstrpl}[2][{}]{%
491   \ifglentryexists{##2}%
492   {%
493     \ifblank{##1}{-}{\GlsXtrWarning{##1}{##2}}%
494   }%
495   {%
496     \gls@defglossaryentry{##2}{name={##2},category=\glxtrcat,
497       description={\nopostdesc},##1}%
498   }%
499   \expandafter\glsp\expandafter[\glxtr@keylist]{##2}%
500 }

```

\Glsxtrpl

```

501 \newcommand*{\Glsxtrpl}[1][{}]{%
502   \def\glxtr@keylist{##1}%
503   \@Glsxtrpl
504 }

```

\@Glsxtrpl

```

505 \newcommand*{\@Glsxtrpl}[2][{}]{%
506   \ifglentryexists{##2}
507   {%
508     \ifblank{##1}{-}{\GlsXtrWarning{##1}{##2}}%
509   }%
510   {%
511     \gls@defglossaryentry{##2}{name={##2},category=\glxtrcat,
512       description={\nopostdesc},##1}%
513   }%
514   \expandafter\glsp\expandafter[\glxtr@keylist]{##2}%
515 }

```

\GlsXtrWarning

```

516 \newcommand*{\GlsXtrWarning}[2]{%
517   \def\glxtr@optlist{##1}%
518   \@onelevel@sanitize\glxtr@optlist
519   \GlossariesExtraWarning{The options ‘\glxtr@optlist’ have
520     been ignored for entry ‘##2’ as it has already been defined}%
521 }

```

Disable commands after the glossary:

```

522 \let\@glxtr@orgprintglossary\@printglossary
523 \renewcommand\@printglossary[2]{%
524   \@glxtr@orgprintglossary{##1}{##2}%
525   \def\@glxtr{\@glxtr@disabledflycommand\glxtr}%
526   \def\@glxtrpl{\@glxtr@disabledflycommand\glxtrpl}%
527   \def\@Glsxtr{\@glxtr@disabledflycommand\Glsxtr}%
528   \def\@Glsxtrpl{\@glxtr@disabledflycommand\Glsxtrpl}%
529 }

```

abledflycommand

```

530 \newcommand*{\@glxtr@disabledflycommand}[1]{%
531   \PackageError{glossaries-extra}%
532   {\string##1\space can't be used after any of the \MessageBreak
533    glossaries have been displayed}%
534   {The on-the-fly commands enabled by
535    \string\GlsXtrEnableOnTheFly\space may only be used \MessageBreak
536    before the glossaries. If you want to use any entries \MessageBreak
537    after any of the glossaries, you must use the standard \MessageBreak
538    method of first defining the entry and then using the \MessageBreak
539    entry with commands like \string\gls}%
540   \@glxtr@disabledflycommand
541 }%
542 \newcommand*{\@glxtr@disabledflycommand}[2][\{##2}

```

End of \GlsXtrEnableOnTheFly. Disable since it can only be used once.

```

543 \let\GlsXtrEnableOnTheFly\relax
544 }
545 \@onlypreamble\GlsXtrEnableOnTheFly

```

1.3.3 Existing Glossary Style Modifications

Modify \setglossarystyle to keep track of the current style. This allows the \glossaries-extra-stylemods package to reset the current style after the required modifications have been made.

r@current@style Initialise the current style to the default style.

```

546 \newcommand*{\@glxtr@current@style}{\@glossary@default@style}

```

Modify \setglossarystyle to set the above.

etglossarystyle

```

547 \renewcommand*{\setglossarystyle}[1]{%
548   \ifcsundef{@glsstyle@#1}%
549   {%
550     \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
551   }%
552   {%
553     \csname @glsstyle@#1\endcsname
554     \protected@edef\@glxtr@current@style{#1}%
555   }%
556   \ifx\@glossary@default@style\relax
557     \protected@edef\@glossary@default@style{#1}%
558   \fi
559 }

```

In case we have an old version of glossaries:

```

560 \ifdef\@glossary@default@style
561 {}

```

```

562 {%
563   \let\@glossary@default@style\relax
564 }

```

listdottedwidth If \glslistdottedwidth has been defined and is currently equal to .5\hsize then make the modification suggested in [bug report #92](#)

```

565 \ifdef\glslistdottedwidth
566 {%
567   \ifdim\glslistdottedwidth=.5\hsize
568     \setlength{\glslistdottedwidth}{-\dimexpr\maxdimen-1sp\relax}
569     \AtBeginDocument{%
570       \ifdim\glslistdottedwidth=-\dimexpr\maxdimen-1sp\relax
571         \setlength{\glslistdottedwidth}{.5\columnwidth}%
572       \fi
573     }%
574   \fi
575 }
576 {}%

```

Similarly for \glsdescwidth:

\glsdescwidth

```

577 \ifdef\glsdescwidth
578 {%
579   \ifdim\glsdescwidth=.6\hsize
580     \setlength{\glsdescwidth}{-\dimexpr\maxdimen-1sp\relax}
581     \AtBeginDocument{%
582       \ifdim\glsdescwidth=-\dimexpr\maxdimen-1sp\relax
583         \setlength{\glsdescwidth}{.6\columnwidth}%
584       \fi
585     }%
586   \fi
587 }
588 {}%

```

and for \glspagelistwidth:

lspagelistwidth

```

589 \ifdef\glspagelistwidth
590 {%
591   \ifdim\glspagelistwidth=.1\hsize
592     \setlength{\glspagelistwidth}{-\dimexpr\maxdimen-1sp\relax}
593     \AtBeginDocument{%
594       \ifdim\glspagelistwidth=-\dimexpr\maxdimen-1sp\relax
595         \setlength{\glspagelistwidth}{.1\columnwidth}%
596       \fi
597     }%
598   \fi
599 }
600 {}%

```

aryentrynumbers Has the nonumberlist option been used?

```
601 \def\org@glossaryentrynumbers#1{#1\gls@save@numberlist{#1}}%
602 \ifx\org@glossaryentrynumbers\glossaryentrynumbers
603   \glsnonumberlistfalse
604   \renewcommand*{\glossaryentrynumbers}[1]{%
605     \ifglsentryexists{\glscurrententrylabel}%
606     {%
607       \@glsxtrpreloctag
608       \GlsXtrFormatLocationList{#1}%
609       \@glsxtrpostloctag
610       \gls@save@numberlist{#1}%
611     }{}%
612   }%
613 \else
614   \glsnonumberlisttrue
615   \renewcommand*{\glossaryentrynumbers}[1]{%
616     \ifglsentryexists{\glscurrententrylabel}%
617     {%
618       \gls@save@numberlist{#1}%
619     }{}%
620   }%
621 \fi
```

matLocationList Provide an easy interface to change the format of the location list without removing the save number list stuff.

```
622 \newcommand*{\GlsXtrFormatLocationList}[1]{#1}
```

Sometimes users want to prefix the location list with “page”/“pages”. The simplest way to determine if the location list consists of a single location is to check for instances of `\delimN` or `\delimR`, but this isn’t so easy to do as they might be embedded inside the argument of formatting commands. With a bit of trickery we can find out by adjusting `\delimN` and `\delimR` to set a flag and then save information to the auxiliary file for the next run.

ePreLocationTag

```
623 \newcommand*{\GlsXtrEnablePreLocationTag}[2]{%
624   \let\@glsxtrpreloctag\@glsxtrpreloctag
625   \let\@glsxtrpostloctag\@glsxtrpostloctag
626   \renewcommand*{\@glsxtr@pagetag}{#1}%
627   \renewcommand*{\@glsxtr@pagetag}{#2}%
628   \renewcommand*{\@glsxtr@savepreloctag}[2]{%
629     \csgdef{\@glsxtr@preloctag@##1}{##2}%
630   }%
631   \renewcommand*{\@glsxtr@doloctag}{%
632     \ifcsundef{\@glsxtr@preloctag\@glscurrententrylabel}%
633     {%
634       \GlossariesWarning{Missing pre-location tag for ‘\glscurrententrylabel’.
635       Rerun required}%
636     }%
637     {%
```

```

638      \csuse{@glxstr@preloctag@\glscurrententrylabel}%
639    }%
640  }%
641 }
642 \@onlypreamble\GlsXtrEnablePreLocationTag

glxstrpreloctag
643 \newcommand*{\@@glxstrpreloctag}{%
644   \let\@glxstr@org@delimN\delimN
645   \let\@glxstr@org@delimR\delimR
646   \let\@glxstr@org@glsgignore\glsgignore
647   \gdef is required as the delimiters may occur inside a scope.
648   \gdef\@glxstr@thisloctag{\@glxstr@pagetag}%
649   \renewcommand*{\delimN}{%
650     \gdef\@glxstr@thisloctag{\@glxstr@pagetag}%
651     \@glxstr@org@delimN}%
652   \renewcommand*{\delimR}{%
653     \gdef\@glxstr@thisloctag{\@glxstr@pagetag}%
654     \@glxstr@org@delimR}%
655   \renewcommand*{\glsgignore}[1]{%
656     \gdef\@glxstr@thisloctag{\relax}%
657     \@glxstr@org@glsgignore{##1}}%
658   \@glxstr@doloctag
659 }

glxstrpreloctag
660 \newcommand*{\@glxstrpreloctag}{%

@glxstr@pagetag
661 \newcommand*{\@glxstr@pagetag}{%

glxstr@pagetag
662 \newcommand*{\@glxstr@pagetag}{%

lsxtrpostloctag
663 \newcommand*{\@@glxtrpostloctag}{%
664   \let\delimN\@glxstr@org@delimN
665   \let\delimR\@glxstr@org@delimR
666   \let\glsgignore\@glxstr@org@glsgignore
667   \protected@write\@auxout{%
668     {\string\@glxstr@savepreloctag{\glscurrententrylabel}\@glxstr@thisloctag}}%
669 }

lsxtrpostloctag
670 \newcommand*{\@glxtrpostloctag}{%

```

lsxtr@preloctag

```
670 \newcommand*{\@glxtr@savepreloctag}[2]{%
671 \protected@write\@auxout{}{%
672   \string\providecommand\string\@glxtr@savepreloctag[2]{}}
```

glxtr@doloctag

```
673 \newcommand*{\@glxtr@doloctag}{}
```

ss@nonumberlist Modify the nonumberlist key to use \GlsXtrFormatLocationList (and also save the number list):

```
674 \renewcommand*{\KV@printgloss@nonumberlist}[1]{%
675   \XKV@plfalse
676   \XKV@sttrue
677   \XKV@checkchoice[\XKV@resa]{#1}{true,false}%
678   {%
679     \csname glsnonumberlist\XKV@resa\endcsname
680     \ifglsnonumberlist
681       \def\glossaryentrynumbers##1{\gls@save@numberlist{##1}}%
682     \else
683       \def\glossaryentrynumbers##1{%
684         \@glxtr@preloctag
685         \GlsXtrFormatLocationList{##1}%
686         \@glxtr@postloctag
687         \gls@save@numberlist{##1}}%
688       \fi
689   }%
690 }
```

1.3.4 Entry Formatting, Hyperlinks and Indexing

\glsentryfmt Change default entry format. Use the generic format for regular terms (that is, entries that have a category with the regular attribute set) or non-regular terms without a short value and use the abbreviation format for non-regular terms that have a short value. If further attributes need to be checked, then \glsentryfmt will need redefining as appropriate (or use \defglsentryfmt). The abbreviation format is set here for entries that have a short form, even if they are regular entries to ensure the abbreviation fonts are correct.

```
691 \renewcommand*{\glsentryfmt}{%
692   \ifglshasshort{\glslabel}{\glssetabbrvfmt{\glscategory{\glslabel}}}%
693   \glsifregular{\glslabel}%
694   {\glxtrregularfont{\glsentryfmt}}%
695   {%
696     \ifglshasshort{\glslabel}%
697     {\glxtrgenabbrvfmt}%
698     {\glxtrregularfont{\glsentryfmt}}%
699   }%
700 }
```


sxtrregularfont Font used for regular entries.

```
701 \newcommand*{\glsxtrregularfont}[1]{#1}
```

Commands like `\glsifplural` are only used by the `\gls`-like commands in the `glossaries` package, but it might be useful for the `postlink` hook to know if the user has used, say, `\glsfirst` or `\glsplural`. This can provide better consistency with the formatting of the `\gls`-like commands, even though they don't use `\glstryentryfmt`.

@gls@field@link Redefine `\@gls@field@link` so that commands like `\glsfirst` can setup `\glsxtrifwasfirstuse` etc to allow the `postlink` hook to work better. This now has an optional argument that sets up the defaults.

```
702 \renewcommand{\@gls@field@link}[4] [] {%
703   \glsoifexists{#3}%
704   {%
705     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
706     \def\glscustomtext{#4}%
707     \@glsxtr@field@linkdefs
708     #1%
709     \@gls@link[#2]{#3}{#4}%
710   }%
711   \glspostlinkhook
712 }
```

@field@linkdefs Default settings for `\@gls@field@link`

```
713 \newcommand*{\@glsxtr@field@linkdefs}{%
714   \let\glsxtrifwasfirstuse\@secondoftwo
715   \let\glsifplural\@secondoftwo
716   \let\glscapscase\@firstofthree
717   \let\glsinsert\@empty
718 }
```

Redefine the field link commands that need to modify the above. Also add accessibility support and set the abbreviation styles if required.

assignfieldfont

```
719 \newcommand*{\glsxtrassignfieldfont}[1]{%
720   \ifglshasshort{#1}%
721   {%
722     \glsssetabbrfmt{\glscategory{#1}}%
723     \glsifregular{#1}%
724     {\let\@gls@field@font\glsxtrregularfont}%
725     {\let\@gls@field@font\@firstofone}%
726   }%
727   {%
728     \glsifnotregular{#1}%
729     {\let\@gls@field@font\@firstofone}%
730     {\let\@gls@field@font\glsxtrregularfont}%
731   }%
732 }
```

\@glstext@ The abbreviation format may also need setting.

```
733 \def\@glstext@#1#2[#3]{%
734   \glstrassignfieldfont{#2}%
735   \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccesstext{#2}#3}}%
736 }
```

\@GLStext@ All uppercase version of \@glstext. The abbreviation format may also need setting.

```
737 \def\@GLStext@#1#2[#3]{%
738   \glstrassignfieldfont{#2}%
739   \@gls@field@link[\let\glscapscase\@thirdofthree]{#1}{#2}%
740   {\@gls@field@font{\GLSaccesstext{#2}\mfirstucMakeUppercase{#3}}}%
741 }
```

\@Glstext@ First letter uppercase version. The abbreviation format may also need setting.

```
742 \def\@Glstext@#1#2[#3]{%
743   \glstrassignfieldfont{#2}%
744   \@gls@field@link[\let\glscapscase\@secondofthree]{#1}{#2}%
745   {\@gls@field@font{\GLSaccesstext{#2}#3}}%
746 }
```

\@glsfirst@ No case changing version. The abbreviation format may also need setting.

```
747 \def\@glsfirst@#1#2[#3]{%
748   \glstrassignfieldfont{#2}%
749   \@gls@field@link[\let\glstrifwasfirstuse\@firstoftwo]{#1}{#2}%
750   {\@gls@field@font{\glsaccessfirst{#2}#3}}%
751 }
```

\@Glsfirst@ First letter uppercase version. The abbreviation format may also need setting.

```
752 \def\@Glsfirst@#1#2[#3]{%
753   \glstrassignfieldfont{#2}%
754   \@gls@field@link
755   [\let\glstrifwasfirstuse\@firstoftwo
756   \let\glscapscase\@secondofthree
757   ]%
758   {#1}{#2}{\@gls@field@font{\Glsaccessfirst{#2}#3}}%
759 }
```

\@GLSfirst@ All uppercase version. The abbreviation format may also need setting.

```
760 \def\@GLSfirst@#1#2[#3]{%
761   \glstrassignfieldfont{#2}%
762   \@gls@field@link
763   [\let\glstrifwasfirstuse\@firstoftwo
764   \let\glscapscase\@thirdofthree
765   ]%
766   {#1}{#2}{\@gls@field@font{\GLSaccessfirst{#2}\mfirstucMakeUppercase{#3}}}%
767 }
```

\@glsplural@ No case changing version. The abbreviation format may also need setting.

```

768 \def\@glsplural@#1#2[#3]{%
769   \glstrassignfieldfont{#2}%
770   \@gls@field@link[\let\glsifplural\@firstoftwo]{#1}{#2}%
771   {\@gls@field@font{\glsaccessplural{#2}#3}}%
772 }

```

\@Glsplural@ First letter uppercase version. The abbreviation format may also need setting.

```

773 \def\@Glsplural@#1#2[#3]{%
774   \glstrassignfieldfont{#2}%
775   \@gls@field@link
776   [\let\glsifplural\@firstoftwo
777    \let\glscapscase\@secondofthree
778   ]%
779   {#1}{#2}{\@gls@field@font{\Glsaccessplural{#2}#3}}%
780 }

```

\@GLSplural@ All uppercase version. The abbreviation format may also need setting.

```

781 \def\@GLSplural@#1#2[#3]{%
782   \glstrassignfieldfont{#2}%
783   \@gls@field@link
784   [\let\glsifplural\@firstoftwo
785    \let\glscapscase\@thirdofthree
786   ]%
787   {#1}{#2}{\@gls@field@font{\GLSaccessplural{#2}\mfirstucMakeUppercase{#3}}}%
788 }

```

glsfirstplural@ No case changing version. The abbreviation format may also need setting.

```

789 \def\@glsfirstplural@#1#2[#3]{%
790   \glstrassignfieldfont{#2}%
791   \@gls@field@link
792   [\let\glstrifwasfirstuse\@firstoftwo
793    \let\glsifplural\@firstoftwo
794   ]%
795   {#1}{#2}{\@gls@field@font{\glsaccessfirstplural{#2}#3}}%
796 }

```

Glsfirstplural@ First letter uppercase version. The abbreviation format may also need setting.

```

797 \def\@Glsfirstplural@#1#2[#3]{%
798   \glstrassignfieldfont{#2}%
799   \@gls@field@link
800   [\let\glstrifwasfirstuse\@firstoftwo
801    \let\glsifplural\@firstoftwo
802    \let\glscapscase\@secondofthree
803   ]%
804   {#1}{#2}{\@gls@field@font{\Glsaccessfirstplural{#2}#3}}%
805 }

```

GLSfirstplural@ All uppercase version. The abbreviation format may also need setting.

```

806 \def\@GLSfirstplural@#1#2[#3]{%
807   \glstrassignfieldfont{#2}%
808   \@gls@field@link
809   [\let\glstrifwasfirstuse\@firstoftwo
810    \let\glsifplural\@firstoftwo
811    \let\glscapscase\@thirdofthree
812   ]%
813   {#1}{#2}%
814   {\@gls@field@font{\@GLSaccessfirstplural{#2}\mfirstucMakeUppercase{#3}}}%
815 }

```

\@glsname@ Redefine to use accessibility support. The abbreviation format may also need setting.

```

816 \def\@glsname@#1#2[#3]{%
817   \glstrassignfieldfont{#2}%
818   \@gls@field@link{#1}{#2}{\@gls@field@font{\@glsaccessname{#2}#3}}%
819 }

```

\@Glsname@ First letter uppercase version. The abbreviation format may also need setting.

```

820 \def\@Glsname@#1#2[#3]{%
821   \glstrassignfieldfont{#2}%
822   \@gls@field@link
823   [\let\glscapscase\@secondoftwo]{#1}{#2}%
824   {\@gls@field@font{\@Glsaccessname{#2}#3}}%
825 }

```

\@GLSname@ All uppercase version. The abbreviation format may also need setting.

```

826 \def\@GLSname@#1#2[#3]{%
827   \glstrassignfieldfont{#2}%
828   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
829   {#1}{#2}%
830   {\@gls@field@font{\@GLSaccessname{#2}\mfirstucMakeUppercase{#3}}}%
831 }

```

\@glsdesc@

```

832 \def\@glsdesc@#1#2[#3]{%
833   \glstrassignfieldfont{#2}%
834   \@gls@field@link{#1}{#2}{\@gls@field@font{\@glsaccessdesc{#2}#3}}%
835 }

```

\@Glsdesc@ First letter uppercase version.

```

836 \def\@Glsdesc@#1#2[#3]{%
837   \glstrassignfieldfont{#2}%
838   \@gls@field@link
839   [\let\glscapscase\@secondoftwo]{#1}{#2}%
840   {\@gls@field@font{\@Glsaccessdesc{#2}#3}}%
841 }

```

\@GLSdesc@ All uppercase version.

```

842 \def\@GLSdesc@#1#2[#3]{%
843   \glstrassignfieldfont{#2}%
844   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
845   {#1}{#2}{\@gls@field@font{\GLSaccessdesc{#2}\mfirstucMakeUppercase{#3}}}%
846 }

```

@glsdescplural@ No case-changing version.

```

847 \def\@glsdescplural@#1#2[#3]{%
848   \glstrassignfieldfont{#2}%
849   \@gls@field@link
850   [\let\glscapscase\@secondoftwo
851   \let\glsifplural\@firstoftwo
852   ]{#1}{#2}{\@gls@field@font{\glsaccessdescplural{#2}#3}}%
853 }

```

@Glsdescplural@ First letter uppercase version.

```

854 \def\@Glsdescplural@#1#2[#3]{%
855   \glstrassignfieldfont{#2}%
856   \@gls@field@link
857   [\let\glscapscase\@secondoftwo
858   \let\glsifplural\@firstoftwo
859   ]{#1}{#2}{\@gls@field@font{\Glsaccessdescplural{#2}#3}}%
860 }

```

@GLSdescplural@ All uppercase version.

```

861 \def\@GLSdesc@#1#2[#3]{%
862   \glstrassignfieldfont{#2}%
863   \@gls@field@link
864   [\let\glscapscase\@thirdoftwo
865   \let\glsifplural\@firstoftwo
866   ]%
867   {#1}{#2}%
868   {\@gls@field@font{\GLSaccessdescplural{#2}\mfirstucMakeUppercase{#3}}}%
869 }

```

\@glssymbol@

```

870 \def\@glssymbol@#1#2[#3]{%
871   \glstrassignfieldfont{#2}%
872   \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccesssymbol{#2}#3}}%
873 }

```

\@Glsymbol@ First letter uppercase version.

```

874 \def\@Glsymbol@#1#2[#3]{%
875   \glstrassignfieldfont{#2}%
876   \@gls@field@link
877   [\let\glscapscase\@secondoftwo]%
878   {#1}{#2}{\@gls@field@font{\Glsaccesssymbol{#2}#3}}%
879 }

```

\@GLSsymbol@ All uppercase version.

```
880 \def\@GLSsymbol@#1#2[#3]{%
881   \glstrassignfieldfont{#2}%
882   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
883   {#1}{#2}{\@gls@field@font{\GLSaccesssymbol{#2}\mfirstucMakeUppercase{#3}}}%
884 }
```

lssymbolplural@ No case-changing version.

```
885 \def\@glssymbolplural@#1#2[#3]{%
886   \glstrassignfieldfont{#2}%
887   \@gls@field@link
888   [\let\glscapscase\@secondoftwo
889   \let\glsifplural\@firstoftwo
890   ]{#1}{#2}{\@gls@field@font{\glsaccesssymbolplural{#2}#3}}%
891 }
```

lssymbolplural@ First letter uppercase version.

```
892 \def\@Glssymbolplural@#1#2[#3]{%
893   \glstrassignfieldfont{#2}%
894   \@gls@field@link
895   [\let\glscapscase\@secondoftwo
896   \let\glsifplural\@firstoftwo
897   ]{#1}{#2}{\@gls@field@font{\Glsaccesssymbolplural{#2}#3}}%
898 }
```

LSsymbolplural@ All uppercase version.

```
899 \def\@GLSsymbol@#1#2[#3]{%
900   \glstrassignfieldfont{#2}%
901   \@gls@field@link
902   [\let\glscapscase\@thirdoftwo
903   \let\glsifplural\@firstoftwo
904   ]%
905   {#1}{#2}%
906   {\@gls@field@font{\GLSaccesssymbolplural{#2}\mfirstucMakeUppercase{#3}}}%
907 }
```

\@Glsuseri@ First letter uppercase version.

```
908 \def\@Glsuseri@#1#2[#3]{%
909   \glstrassignfieldfont{#2}%
910   \@gls@field@link
911   [\let\glscapscase\@secondoftwo]{#1}{#2}%
912   {\@gls@field@font{\Glsentryuseri{#2}#3}}%
913 }
```

\@GLSuseri@ All uppercase version.

```
914 \def\@GLSuseri@#1#2[#3]{%
915   \glstrassignfieldfont{#2}%
916   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
```

```

917     {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuseri{#2}#3}}}%
918 }

```

\@Glsuserii@ First letter uppercase version.

```

919 \def\@Glsuserii@#1#2[#3]{%
920   \glstrassignfieldfont{#2}%
921   \@gls@field@link
922   [\let\glscapscase\@secondoftwo]%
923   {#1}{#2}{\@gls@field@font{\Glentryuserii{#2}#3}}}%
924 }

```

\@GLSuserii@ All uppercase version.

```

925 \def\@GLSuserii@#1#2[#3]{%
926   \glstrassignfieldfont{#2}%
927   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
928   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuserii{#2}#3}}}%
929 }

```

\@Glsuseriii@ First letter uppercase version.

```

930 \def\@Glsuseriii@#1#2[#3]{%
931   \glstrassignfieldfont{#2}%
932   \@gls@field@link
933   [\let\glscapscase\@secondoftwo]%
934   {#1}{#2}{\@gls@field@font{\Glentryuseriii{#2}#3}}}%
935 }

```

\@GLSuseriii@ All uppercase version.

```

936 \def\@GLSuseriii@#1#2[#3]{%
937   \glstrassignfieldfont{#2}%
938   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
939   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuseriii{#2}#3}}}%
940 }

```

\@Glsuseriv@ First letter uppercase version.

```

941 \def\@Glsuseriv@#1#2[#3]{%
942   \glstrassignfieldfont{#2}%
943   \@gls@field@link
944   [\let\glscapscase\@secondoftwo]%
945   {#1}{#2}{\@gls@field@font{\Glentryuseriv{#2}#3}}}%
946 }

```

\@GLSuseriv@ All uppercase version.

```

947 \def\@GLSuseriv@#1#2[#3]{%
948   \glstrassignfieldfont{#2}%
949   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
950   {#1}{#2}%
951   {\@gls@field@font{\mfirstucMakeUppercase{\glentryuseriv{#2}#3}}}%
952 }

```

\@Glsuserv@ First letter uppercase version.

```
953 \def\@Glsuserv@#1#2[#3]{%
954   \glstrassignfieldfont{#2}%
955   \@gls@field@link
956   [\let\glscapscase\@secondoftwo]%
957   {#1}{#2}{\@gls@field@font{\Glsentryuserv{#2}#3}}%
958 }
```

\@GLSuserv@ All uppercase version.

```
959 \def\@GLSuserv@#1#2[#3]{%
960   \glstrassignfieldfont{#2}%
961   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
962   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuserv{#2}#3}}}%
963 }
```

\@Glsuservi@ First letter uppercase version.

```
964 \def\@Glsuservi@#1#2[#3]{%
965   \glstrassignfieldfont{#2}%
966   \@gls@field@link
967   [\let\glscapscase\@secondoftwo]%
968   {#1}{#2}{\@gls@field@font{\Glsentryuservi{#2}#3}}%
969 }
```

\@GLSuservi@ All uppercase version.

```
970 \def\@GLSuservi@#1#2[#3]{%
971   \glstrassignfieldfont{#2}%
972   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
973   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuservi{#2}#3}}}%
974 }
```

Commands like \acrshort already set \glsifplural, but they don't set \glstrifwasfirstuse so they need adjusting.

\@acrshort No case change.

```
975 \def\@acrshort#1#2[#3]{%
976   \glsdoifexists{#2}%
977   {%
978     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
979     \let\glstrifwasfirstuse\@secondoftwo
980     \let\glsifplural\@secondoftwo
981     \let\glscapscase\@firstofthree
982     \let\glsinsert\@empty
983     \def\glscustomtext{%
984       \acronymfont{\glsaccesssshort{#2}}#3%
985     }%
986     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
987   }%
988   \glspostlinkhook
989 }
```


\@Acrshort First letter uppercase.

```
990 \def\@Acrshort#1#2[#3]{%
991   \glsdoifexists{#2}%
992   {%
993     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
994     \let\glxtrifwasfirstuse\@secondoftwo
995     \let\gl@sifplural\@secondoftwo
996     \let\glscapscase\@secondofthree
997     \let\gl$insert\@empty
998     \def\glscustomtext{%
999       \acronymfont{\Glsaccessshort{#2}}#3%
1000     }%
1001     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1002   }%
1003   \glspostlinkhook
1004 }
```

\@ACRshort All uppercase.

```
1005 \def\@ACRshort#1#2[#3]{%
1006   \glsdoifexists{#2}%
1007   {%
1008     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1009     \let\glxtrifwasfirstuse\@secondoftwo
1010     \let\gl@sifplural\@secondoftwo
1011     \let\glscapscase\@thirdofthree
1012     \let\gl$insert\@empty
1013     \def\glscustomtext{%
1014       \mfirstucMakeUppercase{\acronymfont{\Glsaccessshort{#2}}#3}%
1015     }%
1016     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1017   }%
1018   \glspostlinkhook
1019 }
```

\@acrshortpl No case change.

```
1020 \def\@acrshortpl#1#2[#3]{%
1021   \glsdoifexists{#2}%
1022   {%
1023     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1024     \let\glxtrifwasfirstuse\@secondoftwo
1025     \let\gl@sifplural\@firstoftwo
1026     \let\glscapscase\@firstofthree
1027     \let\gl$insert\@empty
1028     \def\glscustomtext{%
1029       \acronymfont{\Glsaccessshortpl{#2}}#3%
1030     }%
1031     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1032   }%
1033   \glspostlinkhook
```

1034 }

\@Acrshortpl First letter uppercase.

```
1035 \def\@Acrshortpl#1#2[#3]{%
1036   \glsdoifexists{#2}%
1037   {%
1038     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1039     \let\glxtrifwasfirstuse\@secondoftwo
1040     \let\glsifplural\@firstoftwo
1041     \let\glscapscase\@secondofthree
1042     \let\glsinsert\@empty
1043     \def\glscustomtext{%
1044       \acronymfont{\Glsaccessshortpl{#2}}#3%
1045     }%
1046     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1047   }%
1048   \glspostlinkhook
1049 }
```

\@ACRshortpl All uppercase.

```
1050 \def\@ACRshortpl#1#2[#3]{%
1051   \glsdoifexists{#2}%
1052   {%
1053     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1054     \let\glxtrifwasfirstuse\@secondoftwo
1055     \let\glsifplural\@firstoftwo
1056     \let\glscapscase\@thirdofthree
1057     \let\glsinsert\@empty
1058     \def\glscustomtext{%
1059       \mfirstucMakeUppercase{\acronymfont{\Glsaccessshortpl{#2}}#3}%
1060     }%
1061     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1062   }%
1063   \glspostlinkhook
1064 }
```

\@acrlong No case change.

```
1065 \def\@acrlong#1#2[#3]{%
1066   \glsdoifexists{#2}%
1067   {%
1068     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1069     \let\glxtrifwasfirstuse\@secondoftwo
1070     \let\glsifplural\@secondoftwo
1071     \let\glscapscase\@firstofthree
1072     \let\glsinsert\@empty
1073     \def\glscustomtext{%
1074       \acronymfont{\Glsaccesslong{#2}}#3%
1075     }%
1076     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1077   }
```

```

1077 }%
1078 \glspostlinkhook
1079 }

```

\@Acrlong First letter uppercase.

```

1080 \def\@Acrlong#1#2[#3]{%
1081   \glsdoidexists{#2}%
1082   {%
1083     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1084     \let\glxtrifwasfirstuse\@secondoftwo
1085     \let\gl@sifplural\@secondoftwo
1086     \let\glscapscase\@secondofthree
1087     \let\gl$insert\@empty
1088     \def\glscustomtext{%
1089       \acronymfont{\Glsaccesslong{#2}}#3%
1090     }%
1091     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1092   }%
1093   \glspostlinkhook
1094 }

```

\@ACRlong All uppercase.

```

1095 \def\@ACRlong#1#2[#3]{%
1096   \glsdoidexists{#2}%
1097   {%
1098     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1099     \let\glxtrifwasfirstuse\@secondoftwo
1100     \let\gl@sifplural\@secondoftwo
1101     \let\glscapscase\@thirdofthree
1102     \let\gl$insert\@empty
1103     \def\glscustomtext{%
1104       \mfirstucMakeUppercase{\acronymfont{\Glsaccesslong{#2}}#3}%
1105     }%
1106     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1107   }%
1108   \glspostlinkhook
1109 }

```

\@acrlongpl No case change.

```

1110 \def\@acrlongpl#1#2[#3]{%
1111   \glsdoidexists{#2}%
1112   {%
1113     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1114     \let\glxtrifwasfirstuse\@secondoftwo
1115     \let\gl@sifplural\@firstoftwo
1116     \let\glscapscase\@firstofthree
1117     \let\gl$insert\@empty
1118     \def\glscustomtext{%
1119       \acronymfont{\Glsaccesslongpl{#2}}#3%

```

```

1120 }%
1121 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1122 }%
1123 \glspostlinkhook
1124 }

```

\@Acrlongpl First letter uppercase.

```

1125 \def\@Acrlongpl#1#2[#3]{%
1126   \glsdoifexists{#2}%
1127   {%
1128     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
1129     \let\glstrifwasfirstuse\@secondoftwo
1130     \let\glsifplural\@firstoftwo
1131     \let\glscapscase\@secondofthree
1132     \let\glsinsert\@empty
1133     \def\glscustomtext{%
1134       \acronymfont{\Glsaccesslongpl{#2}}#3%
1135     }%
1136     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1137   }%
1138   \glspostlinkhook
1139 }

```

\@ACRlongpl All uppercase.

```

1140 \def\@ACRlongpl#1#2[#3]{%
1141   \glsdoifexists{#2}%
1142   {%
1143     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
1144     \let\glstrifwasfirstuse\@secondoftwo
1145     \let\glsifplural\@firstoftwo
1146     \let\glscapscase\@thirdofthree
1147     \let\glsinsert\@empty
1148     \def\glscustomtext{%
1149       \mfirstucMakeUppercase{\acronymfont{\Glsaccesslongpl{#2}}#3}%
1150     }%
1151     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1152   }%
1153   \glspostlinkhook
1154 }

```

Modify \@glsaddkey so additional keys provided by the user can be treated in a similar way.

\@glsaddkey

```

1155 \renewcommand*{\@glsaddkey}[7]{%
1156   \key@ifundefined{glossentry}{#1}%
1157   {%
1158     \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
1159     \appto\@gls@keymap{,{#1}{#1}}%

```

```

1160 \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
1161 \appto\@newglossaryentryposthook{%
1162   \letcs{\@glo@tmp}{@glo@#1}%
1163   \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
1164 }%
1165 \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
1166 \newcommand*{#4}[1]{\@Gls@entry@field{##1}{#1}}%

```

Now for the commands with links. First the version with no case change (same as before):

```

1167 \ifcsdef{@gls@user@#1@}%
1168 {%
1169   \PackageError{glossaries}%
1170   {Can't define '\string#5' as helper command
1171     '\expandafter\string\csname @gls@user@#1@endcsname' already
1172     exists}%
1173 }%
1174 }%
1175 {%
1176   \expandafter\newcommand\expandafter*\expandafter
1177   {\csname @gls@user@#1@endcsname}[2][ ]{%
1178     \new@ifnextchar[%
1179       {\csuse{@gls@user@#1@}{##1}{##2}}%
1180       {\csuse{@gls@user@#1@}{##1}{##2}[ ]}}%
1181   \csdef{@gls@user@#1@}##1##2[##3]{%
1182     \@gls@field@link{##1}{##2}{#3{##2}##3}%
1183   }%
1184   \newrobustcmd*{#5}{%
1185     \expandafter\@gls@hyp@opt\csname @gls@user@#1@endcsname}%
1186   }%

```

Next the version with the first letter converted to upper case (modified):

```

1187 \ifcsdef{@Gls@user@#1@}%
1188 {%
1189   \PackageError{glossaries}%
1190   {Can't define '\string#6' as helper command
1191     '\expandafter\string\csname @Gls@user@#1@endcsname' already
1192     exists}%
1193 }%
1194 }%
1195 {%
1196   \expandafter\newcommand\expandafter*\expandafter
1197   {\csname @Gls@user@#1@endcsname}[2][ ]{%
1198     \new@ifnextchar[%
1199       {\csuse{@Gls@user@#1@}{##1}{##2}}%
1200       {\csuse{@Gls@user@#1@}{##1}{##2}[ ]}}%
1201   \csdef{@Gls@user@#1@}##1##2[##3]{%
1202     \@gls@field@link[\let\glscaps@case\@secondofthree]%
1203     {##1}{##2}{#4{##2}##3}%
1204   }%
1205   \newrobustcmd*{#6}{%

```

```

1206     \expandafter\@gls@hyp@opt\csname @Gls@user@#1\endcsname}%
1207 }%
    Finally the all caps version (modified):
1208     \ifcsdef{@GLS@user@#1@}%
1209     {%
1210         \PackageError{glossaries}%
1211         {Can't define '\string#7' as helper command
1212         '\expandafter\string\csname @GLS@user@#1\endcsname' already
1213         exists}%
1214     }%
1215 }%
1216 {%
1217     \expandafter\newcommand\expandafter*\expandafter
1218     {\csname @GLS@user@#1\endcsname}[2][ ]{%
1219         \new@ifnextchar[%
1220             {\csuse{@GLS@user@#1@}{##1}{##2}}%
1221             {\csuse{@GLS@user@#1@}{##1}{##2}[ ]}}%
1222     \csdef{@GLS@user@#1@}##1##2[##3]{%
1223         \@gls@field@link[\let\glscaps@case\@thirdofthree]%
1224         {##1}{##2}{\mfirstucMakeUppercase{#3{##2}##3}}%
1225     }%
1226     \newrobustcmd*{#7}{%
1227         \expandafter\@gls@hyp@opt\csname @GLS@user@#1\endcsname}%
1228     }%
1229 }%
1230 {%
1231     \PackageError{glossaries-extra}{Key '#1' already exists}{}%
1232 }%
1233 }

```

checkfirsthyper Old versions of glossaries don't define this, so provide it just in case it hasn't been defined.

```

1234 \providecommand*\@gls@link@nocheckfirsthyper{}

```

checkfirsthyper Modify check to determine if the hyperlink should be automatically suppressed, but save the original in case the acronyms are restored.

```

1235 \let\@gls@xtr@org@checkfirsthyper\@gls@link@checkfirsthyper
1236 \renewcommand*\@gls@link@checkfirsthyper{%

```

\ifglsused isn't useful in the post link hook as it's already been unset by then, so define a command that can be used in the post link hook. Since \@gls@link@checkfirsthyper is only used by commands like \gls but not by other commands, this seems the best place to put it.

```

1237 \ifglsused{\glslabel}%
1238 {\let\glsxtrifwasfirstuse\@secondoftwo}
1239 {\let\glsxtrifwasfirstuse\@firstoftwo}%

```

Store the category label for convenience.

```

1240 \edef\gls@categorylabel{\gls@category{\glslabel}}%
1241 \ifglsused{\glslabel}%

```

```

1242 {%
1243   \glsifcategoryattribute{\glscategorylabel}{nohypernext}{true}%
1244   {\KV@glslink@hyperfalse}{}}%
1245 }%
1246 {%
1247   \glsifcategoryattribute{\glscategorylabel}{nohyperfirst}{true}%
1248   {\KV@glslink@hyperfalse}{}}%
1249 }%
1250 \glslinkcheckfirsthyperhook
1251 }

```

`\glsdisablehyperinlist` This command was introduced in glossaries v4.19. If it hasn't been defined, we're using an earlier version, in which case the `nohyper` attribute can't be implemented.

```

1252 \ifdef\do@glsglsdisablehyperinlist
1253 {%
1254   \let\@glslxtr\do@glsglsdisablehyperinlist\do@glsglsdisablehyperinlist
1255   \renewcommand*{\do@glsglsdisablehyperinlist}{%
1256     \@glslxtr\do@glsglsdisablehyperinlist
1257     \glsifattribute{\glslabel}{nohyper}{true}{\KV@glslink@hyperfalse}{}}%
1258   }
1259 }
1260 {}

```

Define a `noindex` key to prevent writing information to the external file.

```

1261 \define@boolkey{glslink}{noindex}[true]{}
1262 \KV@glslink@noindexfalse

```

If `\@gls@setdefault@glslink@opts` has been defined (glossaries v4.20) use it to set the default keys in `\@glslink`.

`\@glslink@opts`

```

1263 \ifdef\@gls@setdefault@glslink@opts
1264 {
1265   \renewcommand*{\@gls@setdefault@glslink@opts}{%
1266     \KV@glslink@noindexfalse
1267   }
1268 }
1269 {

```

Not defined so prepend it to `\do@glsglsdisablehyperinlist` to achieve the same effect.

```

1270   \newcommand*{\@gls@setdefault@glslink@opts}{%
1271     \KV@glslink@noindexfalse
1272   }
1273   \pretol\do@glsglsdisablehyperinlist{\@gls@setdefault@glslink@opts}
1274 }

```

`\setdefaultgls@opts` Set the default options for `\glslink` etc.

```

1275 \newcommand*{\GlsXtrSetDefaultGlsOpts}[1]{%
1276   \renewcommand*{\@gls@setdefault@glslink@opts}{\setkeys{glslink}{#1}}%
1277 }

```

`\lsxtrifindexing` Provide user level command to access it in `\glswriteentry`.

```
1278 \newcommand*{\lsxtrifindexing}[2]{%
1279   \ifKV@glslink@noindex #2\else #1\fi
1280 }
```

`\glswriteentry` Redefine to test for `indexonlyfirst` category attribute.

```
1281 \renewcommand*{\glswriteentry}[2]{%
1282   \lsxtrifindexing
1283   {%
1284     \ifglindexonlyfirst
1285       \ifglused{#1}
1286       {\lsxtrdoautoindexname{#1}{dualindex}}%
1287       {#2}%
1288     \else
1289       \gl@ifattribute{#1}{indexonlyfirst}{true}%
1290       {\ifglused{#1}
1291        {\lsxtrdoautoindexname{#1}{dualindex}}%
1292        {#2}}%
1293       {#2}%
1294     \fi
1295   }%
1296   {}%
1297 }
```

`@do@@wrglossary` Hook into glossary indexing command so that it can also use `\index` at the same time if required and add user hook.

```
1298 \appto@@do@@wrglossary{\@glxtr@do@@wrindex
1299   \lsxtrdowrglossaryhook{\@gl@label}%
1300 }
```

(The label can be obtained from `\@gl@label` at this point.)

Similarly for the “noidx” version:

`s@noidxglossary`

```
1301 \appto\gl@s@noidxglossary{\@glxtr@do@@wrindex
1302   \lsxtrdowrglossaryhook{\@gl@label}%
1303 }
```

`xtr@do@@wrindex`

```
1304 \newcommand*{\@glxtr@do@@wrindex}{%
1305   \lsxtrdoautoindexname{\@gl@label}{dualindex}%
1306 }
```

`owrglossaryhook` Allow user to hook into indexing code. (Always used by `\gl@sadd`. Used by `\gl@s` when indexing, which may or may not occur depending on the indexing settings.)

```
1307 \newcommand*{\glxtrdowrglossaryhook}[1]{}
```


`\gls@alt@hyp@opt` Commands like `\gls` have a star or plus version. Provide a third symbol that the user can adapt for convenience.

```

1308 \newcommand*{\@gls@alt@hyp@opt}[1]{%
1309   \let\glslinkvar\@firstofthree
1310   \let\@gls@hyp@opt@cs#1\relax
1311   \@ifstar{\s@gls@hyp@opt}%
1312   {\@ifnextchar+%
1313     {\@firstoftwo{\p@gls@hyp@opt}}%
1314     {%
1315       \expandafter\@ifnextchar\@gls@alt@hyp@opt@char
1316       {\@firstoftwo{\@alt@gls@hyp@opt}}%
1317       {#1}%
1318     }%
1319   }%
1320 }

```

`\alt@gls@hyp@opt` User version

```

1321 \newcommand*{\@alt@gls@hyp@opt}[1][ ]{%
1322   \let\glslinkvar\@firstofthree
1323   \expandafter\@gls@hyp@opt@cs\expandafter[\@gls@alt@hyp@opt@keys,#1]}

```

`\lt@hyp@opt@char` Contains the character used as the command modifier.

```

1324 \newcommand*{\@gls@alt@hyp@opt@char}{}

```

`\lt@hyp@opt@keys` Contains the option list used as the command modifier.

```

1325 \newcommand*{\@gls@alt@hyp@opt@keys}{}

```

`\rSetAltModifier`

```

1326 \newcommand*{\GlsXtrSetAltModifier}[2]{%
1327   \let\@gls@hyp@opt\@gls@alt@hyp@opt
1328   \def\@gls@alt@hyp@opt@char{#1}%
1329   \def\@gls@alt@hyp@opt@keys{#2}%
1330 }

```

`\glsdohyperlink` Unpleasant complications can occur if the text or first key etc contains `\gls`, particularly if there are hyperlinks. To get around this problem, patch `\glsdohyperlink` so that it temporarily makes `\gls` behave like `\glstext[<hyper=false,noindex>]`. (This will be overridden if the user explicitly cancels either of those options in the optional argument of `\gls` or using the plus version.) This also patches the short form commands like `\acrshort` and `\glsxtrshort` to use `\glsentryshort` and, similarly, the long form commands like `\acrlong` and `\glsxtrlong` to use `\glsentrylong`.

```

1331 \renewcommand*{\glsdohyperlink}[2]{%
1332   \hyperlink{#1}{\@glsxtrprotectlinks#2}}

```

`\glsdisablehyper` Redefine in case we have an old version of glossaries.

```

1333 \ifundef\glsdonohyperlink
1334 {%

```

```

1335 \renewcommand{\glsdisablehyper}{%
1336   \KV@glslink@hyperfalse
1337   \let\@glslink\glsdonohyperlink
1338   \let\@glstarget\@secondoftwo
1339 }
1340 }
1341 {}

```

`\glsdonohyperlink` This command was only introduced in glossaries v4.20, so it may not be defined. For older glossaries versions, this won't be used if `hyperref` hasn't been loaded, which means the indexing will still take place.

```

1342 \def\glsdonohyperlink#1#2{{\glstrprotectlinks #2}}

```

Reset `\@glslink` with patched versions:

```

1343 \ifcsundef{hyperlink}%
1344 {%
1345   \let\@glslink\glsdonohyperlink
1346 }%
1347 {%
1348   \let\@glslink\glsdohyperlink
1349 }

```

`\glstrprotectlinks` Make `\gls` (and variants) behave like the corresponding `\gls{text}` (and variants) with hyperlinking and indexing off.

```

1350 \newcommand*{\glstrprotectlinks}{%
1351   \KV@glslink@hyperfalse
1352   \KV@glslink@noindextrue
1353   \let\@gls@\@glstrp@text@
1354   \let\@Gls@\@Glsxtrp@text@
1355   \let\@GLS@\@GLSxtrp@text@
1356   \let\@glspl@\@glstrp@plural@
1357   \let\@Glspl@\@Glsxtrp@plural@
1358   \let\@GLSpl@\@GLSxtrp@plural@
1359   \let\@glstrshort@\@glstrp@short@
1360   \let\@Glsxtrshort@\@Glsxtrp@short@
1361   \let\@GLSxtrshort@\@GLSxtrp@short@
1362   \let\@glstrlong@\@glstrp@long@
1363   \let\@Glsxtrlong@\@Glsxtrp@long@
1364   \let\@GLSxtrlong@\@GLSxtrp@long@
1365   \let\@glstrshortpl@\@glstrp@shortpl@
1366   \let\@Glsxtrshortpl@\@Glsxtrp@shortpl@
1367   \let\@GLSxtrshortpl@\@GLSxtrp@shortpl@
1368   \let\@glstrlongpl@\@glstrp@longpl@
1369   \let\@Glsxtrlongpl@\@Glsxtrp@longpl@
1370   \let\@GLSxtrlongpl@\@GLSxtrp@longpl@
1371   \let\@acrshort@\@glstrp@acrshort@
1372   \let\@Acrshort@\@Glsxtrp@acrshort@
1373   \let\@ACRshort@\@GLSxtrp@acrshort@
1374   \let\@acrshortpl@\@glstrp@acrshortpl@

```

```

1375 \let\@Acrshortpl\@Glsxtr@p@acrshortpl@
1376 \let\@ACRshortpl\@GLSxtr@p@acrshortpl@
1377 \let\@acrlong\@glxtr@p@acrlong@
1378 \let\@Acrlong\@Glsxtr@p@acrlong@
1379 \let\@ACRlong\@GLSxtr@p@acrlong@
1380 \let\@acrlongpl\@glxtr@p@acrlongpl@
1381 \let\@Acrlongpl\@Glsxtr@p@acrlongpl@
1382 \let\@ACRlongpl\@GLSxtr@p@acrlongpl@
1383 }

```

These protected versions need grouping to prevent the label from getting confused.

@Glsxtr@p@text@

```
1384 \def\@glxtr@p@text@#1#2[#3]{\@glstext@{#1}{#2}[#3]}
```

@Glsxtr@p@text@

```
1385 \def\@Glsxtr@p@text@#1#2[#3]{\@Glstext@{#1}{#2}[#3]}
```

@GLSxtr@p@text@

```
1386 \def\@GLSxtr@p@text@#1#2[#3]{\@GLStext@{#1}{#2}[#3]}
```

lsxtr@p@plural@

```
1387 \def\@glxtr@p@plural@#1#2[#3]{\@glsplural@{#1}{#2}[#3]}
```

lsxtr@p@plural@

```
1388 \def\@Glsxtr@p@plural@#1#2[#3]{\@Glsplural@{#1}{#2}[#3]}
```

LSxtr@p@plural@

```
1389 \def\@GLSxtr@p@plural@#1#2[#3]{\@GLSplural@{#1}{#2}[#3]}
```

glxtr@p@short@

```

1390 \def\@glxtr@p@short@#1#2[#3]{%
1391 {%
1392   \glsssetabbrvfmt{\glscategory{#2}}%
1393   \glsabbrvfont{\gl Sentryshort{#2}}#3%
1394 }%
1395 }

```

Glsxtr@p@short@

```

1396 \def\@Glsxtr@p@short@#1#2[#3]{%
1397 {%
1398   \glsssetabbrvfmt{\glscategory{#2}}%
1399   \glsabbrvfont{\Glsentryshort{#2}}#3%
1400 }%
1401 }

```

GLSxtr@p@short@

```
1402 \def\@GLSxtr@p@short@#1#2[#3]{%
1403   {%
1404     \glsetabbrvfmt{\glscategory{#2}}%
1405     \mfirstucMakeUppercase{\glabbrvfont{\glentryshort{#2}}#3}%
1406   }%
1407 }
```

sxtr@p@shortpl@

```
1408 \def\@glxtr@p@shortpl@#1#2[#3]{%
1409   {%
1410     \glsetabbrvfmt{\glscategory{#2}}%
1411     \glabbrvfont{\glentryshortpl{#2}}#3%
1412   }%
1413 }
```

sxtr@p@shortpl@

```
1414 \def\@Glsxtr@p@shortpl@#1#2[#3]{%
1415   {%
1416     \glsetabbrvfmt{\glscategory{#2}}%
1417     \glabbrvfont{\Glsentryshortpl{#2}}#3%
1418   }%
1419 }
```

Sxtr@p@shortpl@

```
1420 \def\@GLSxtr@p@shortpl@#1#2[#3]{%
1421   {%
1422     \glsetabbrvfmt{\glscategory{#2}}%
1423     \mfirstucMakeUppercase{\glabbrvfont{\glentryshortpl{#2}}#3}%
1424   }%
1425 }
```

@glxtr@p@long@

```
1426 \def\@glxtr@p@long@#1#2[#3]{\{\glentrylong{#2}#3}}
```

@Glsxtr@p@long@

```
1427 \def\@Glsxtr@p@long@#1#2[#3]{\{\Glsentrylong{#2}#3}}
```

@GLSxtr@p@long@

```
1428 \def\@GLSxtr@p@long@#1#2[#3]{%
1429   {\mfirstucMakeUppercase{\glslongfont{\glentrylong{#2}}#3}}}
```

lsxtr@p@longpl@

```
1430 \def\@glxtr@p@longpl@#1#2[#3]{\{\glentrylongpl{#2}#3}}
```

lsxtr@p@longpl@

```
1431 \def\@Glsxtr@p@longpl@#1#2[#3]{\{\glslongfont{\Glsentrylongpl{#2}}#3}}
```

LSxtr@p@longpl@
1432 \def\@GLSxtr@p@longpl@#1#2[#3]{%
1433 {\mfirstucMakeUppercase{\glslongfont{\glsentrylongpl{#2}}#3}}}

xtr@p@acrshort@
1434 \def\@glsxtr@p@acrshort@#1#2[#3]{\acronymfont{\glsentryshort{#2}}#3}}

xtr@p@acrshort@
1435 \def\@Glsxtr@p@acrshort@#1#2[#3]{\acronymfont{\Glsentryshort{#2}}#3}}

xtr@p@acrshort@
1436 \def\@GLSxtr@p@acrshort@#1#2[#3]{%
1437 {\mfirstucMakeUppercase{\acronymfont{\glsentryshort{#2}}#3}}}

r@p@acrshortpl@
1438 \def\@glsxtr@p@acrshortpl@#1#2[#3]{\acronymfont{\glsentryshortpl{#2}}#3}}

r@p@acrshortpl@
1439 \def\@Glsxtr@p@acrshortpl@#1#2[#3]{\acronymfont{\Glsentryshortpl{#2}}#3}}

r@p@acrshortpl@
1440 \def\@GLSxtr@p@acrshortpl@#1#2[#3]{%
1441 {\mfirstucMakeUppercase{\acronymfont{\glsentryshortpl{#2}}#3}}}

sxtr@p@acrlong@
1442 \def\@glsxtr@p@acrlong@#1#2[#3]{\glsentrylong{#2}#3}}

sxtr@p@acrlong@
1443 \def\@Glsxtr@p@acrlong@#1#2[#3]{\Glsentrylong{#2}#3}}

Sxtr@p@acrlong@
1444 \def\@GLSxtr@p@acrlong@#1#2[#3]{%
1445 {\mfirstucMakeUppercase{\glsentrylong{#2}#3}}}

tr@p@acrlongpl@
1446 \def\@glsxtr@p@acrlongpl@#1#2[#3]{\glsentrylongpl{#2}#3}}

tr@p@acrlongpl@
1447 \def\@Glsxtr@p@acrlongpl@#1#2[#3]{\Glsentrylongpl{#2}#3}}

tr@p@acrlongpl@
1448 \def\@GLSxtr@p@acrlongpl@#1#2[#3]{%
1449 {\mfirstucMakeUppercase{\glsentrylongpl{#2}#3}}}

1.3.5 Entry Counting

The entry counting mechanism from glossaries is adjusted here to work with category attributes. Provide a convenient command to enable entry counting, set the entrycount attribute for given categories and redefine \gls etc to use \cgl's instead.

First adjust definitions of the unset and reset commands to provide a hook.

```
\@glsunset Global unset.
1450 \renewcommand*{\@glsunset}[1]{%
1451   \@glsunset{#1}%
1452   \glsxtrpostunset{#1}%
1453 }%

glsxtrpostunset
1454 \newcommand*{\glsxtrpostunset}[1]{%

\@glslocalunset Local unset.
1455 \renewcommand*{\@glslocalunset}[1]{%
1456   \@glslocalunset{#1}%
1457   \glsxtrpostlocalunset{#1}%
1458 }%

rpostlocalunset
1459 \newcommand*{\glsxtrpostlocalunset}[1]{%

\@glsreset Global reset.
1460 \renewcommand*{\@glsreset}[1]{%
1461   \@glsreset{#1}%
1462   \glsxtrpostreset{#1}%
1463 }%

glsxtrpostreset
1464 \newcommand*{\glsxtrpostreset}[1]{%

\@glslocalreset Local reset.
1465 \renewcommand*{\@glslocalreset}[1]{%
1466   \@glslocalreset{#1}%
1467   \glsxtrpostlocalreset{#1}%
1468 }%

rpostlocalreset
1469 \newcommand*{\glsxtrpostlocalreset}[1]{%

leEntryCounting The first argument is the list of categories and the second argument is the value of the en-
entrycount attribute.
1470 \newcommand*{\GlsXtrEnableEntryCounting}[2]{%
```

Enable entry counting:

```
1471 \glsenableentrycount
```

Redefine \gls etc:

```
1472 \renewcommand*{\gls}{\cglsl}%
1473 \renewcommand*{\Gls}{\cGls}%
1474 \renewcommand*{\glspl}{\cglspl}%
1475 \renewcommand*{\Glspl}{\cGlspl}%
1476 \renewcommand*{\GLS}{\cGLS}%
1477 \renewcommand*{\GLSpl}{\cGLSpl}%
```

Set the entrycount attribute:

```
1478 \@glsxtr@setentrycountunsetattr{#1}{#2}%
```

In case this command is used again:

```
1479 \let\GlsXtrEnableEntryCounting\@glsxtr@setentrycountunsetattr
1480 \renewcommand*{\GlsXtrEnableEntryUnitCounting}[3]{%
1481   \PackageError{glossaries-extra}{\string\GlsXtrEnableEntryUnitCounting\space
1482     can't be used with \string\GlsXtrEnableEntryCounting}%
1483   {Use one or other but not both commands}}%
1484 }
```

ycountunsetattr

```
1485 \newcommand*{\@glsxtr@setentrycountunsetattr}[2]{%
1486   \@for\@glsxtr@cat:=#1\do
1487   {%
1488     \ifdefempty{\@glsxtr@cat}{}%
1489     {%
1490       \glssetcategoryattribute{\@glsxtr@cat}{entrycount}{#2}%
1491     }%
1492   }%
1493 }
```

Redefine the entry counting commands to take into account the entrycount attribute.

nableentrycount

```
1494 \renewcommand*{\glsenableentrycount}{%
```

Enable new fields:

```
1495 \appto\@newglossaryentry@defcounters{\@newglossaryentry@defcounters}%
```

Just in case the user has switched on the docdef option.

```
1496 \renewcommand*{\gls@defdocnewglossaryentry}{%
1497   \renewcommand*{\newglossaryentry}[2]{%
1498     \PackageError{glossaries}{\string\newglossaryentry\space
1499       may only be used in the preamble when entry counting has
1500       been activated}{If you use \string\glsenableentrycount\space
1501       you must place all entry definitions in the preamble not in
1502       the document environment}%
1503   }%
1504 }
```

New commands to access new fields:

```

1505 \newcommand*\glsentrycurrcount}[1]{%
1506 \ifcsundef{glo@glsdetoklabel{##1}@currcount}%
1507 {0}{\@gls@entry@field{##1}{currcount}}%
1508 }%
1509 \newcommand*\glsentryprevcount}[1]{%
1510 \ifcsundef{glo@glsdetoklabel{##1}@prevcount}%
1511 {0}{\@gls@entry@field{##1}{prevcount}}%
1512 }%

```

Adjust post unset and reset:

```

1513 \let\@glsxtr@entrycount@org@unset\glsxtrpostunset
1514 \renewcommand*\glsxtrpostunset}[1]{%
1515 \@glsxtr@entrycount@org@unset{##1}%
1516 \@gls@increment@currcount{##1}%
1517 }%
1518 \let\@glsxtr@entrycount@org@localunset\glsxtrpostlocalunset
1519 \renewcommand*\glsxtrpostlocalunset}[1]{%
1520 \@glsxtr@entrycount@org@localunset{##1}%
1521 \@gls@local@increment@currcount{##1}%
1522 }%
1523 \let\@glsxtr@entrycount@org@reset\glsxtrpostreset
1524 \renewcommand*\glsxtrpostreset}[1]{%
1525 \@glsxtr@entrycount@org@reset{##1}%
1526 \csgdef{glo@glsdetoklabel{##1}@currcount}{0}%
1527 }%
1528 \let\@glsxtr@entrycount@org@localreset\glsxtrpostlocalreset
1529 \renewcommand*\glsxtrpostlocalreset}[1]{%
1530 \@glsxtr@entrycount@org@localreset{##1}%
1531 \csdef{glo@glsdetoklabel{##1}@currcount}{0}%
1532 }%

```

Modifications to take into account the attributes that govern whether the entry should be unset.

```

1533 \let\@cgl@s\@cgl@s
1534 \let\@cgl spl\@cgl spl
1535 \let\@cGLS\@cGLS
1536 \let\@cGL spl\@cGL spl
1537 \let\@cGLS\@cGLS
1538 \let\@cGL Spl\@cGL Spl

```

The rest is as the original definition.

```

1539 \AtEndDocument{\@gls@write@entrycounts}%
1540 \renewcommand*\@gls@entry@count}[2]{%
1541 \csgdef{glo@glsdetoklabel{##1}@prevcount}{##2}%
1542 }%
1543 \let\glsenableentrycount\relax
1544 \renewcommand*\glsenableentryunitcount{%
1545 \PackageError{glossaries-extra}{\string\glsenableentryunitcount\space
1546 can't be used with \string\glsenableentrycount}%

```



```

1547     {Use one or other but not both commands}%
1548 }%
1549 }

```

ite@entrycounts Modify this command so that it only writes the information for entries with the entrycount attribute and issue warning if no entries have this attribute set.

```

1550 \renewcommand*{\@gls@write@entrycounts}{%
1551   \immediate\write\@auxout
1552     {\string\providecommand*{\string\@gls@entry@count}[2]{}}%
1553   \count@=0\relax
1554   \forallglsentries{\@glsentry}{%
1555     \glshasattribute{\@glsentry}{entrycount}%
1556     {%
1557       \ifglsused{\@glsentry}%
1558       {%
1559         \immediate\write\@auxout
1560           {\string\@gls@entry@count{\@glsentry}{\glsentrycurrcount{\@glsentry}}}%
1561       }%
1562     }%
1563     \advance\count@ by \@ne
1564   }%
1565 }%
1566 }%
1567 \ifnum\count@=0
1568   \GlossariesExtraWarningNoLine{Entry counting has been enabled
1569     \MessageBreak with \string\glsenableentrycount\space but the
1570     \MessageBreak attribute 'entrycount' hasn't
1571     \MessageBreak been assigned to any of the defined
1572     \MessageBreak entries}%
1573 \fi
1574 }

```

trifcounttrigger `\glstrifcounttrigger{<label>}{<trigger format>}{<normal>}`

```

1575 \newcommand*{\glstrifcounttrigger}[3]{%
1576   \glshasattribute{#1}{entrycount}%
1577   {%
1578     \ifnum\glsentryprevcount{#1}>\glsgetattribute{#1}{entrycount}\relax
1579     #3%
1580   \else
1581     #2%
1582   \fi
1583 }%
1584 {#3}%
1585 }

```

Actual internal definitions of \cgl's used when entry counting is enabled.

\@@cgl's@

```

1586 \def\@@cgl's@#1#2[#3]{%
1587   \gl'sxtrifcounttrigger{#2}%
1588   {%
1589     \cgl'sformat{#2}{#3}%
1590     \gl'sunset{#2}%
1591   }%
1592   {%
1593     \@gl's@{#1}{#2}[#3]%
1594   }%
1595 }%
```

\@@cgl'spl@

```

1596 \def\@@cgl'spl@#1#2[#3]{%
1597   \gl'sxtrifcounttrigger{#2}%
1598   {%
1599     \cgl'splformat{#2}{#3}%
1600     \gl'sunset{#2}%
1601   }%
1602   {%
1603     \@gl'spl@{#1}{#2}[#3]%
1604   }%
1605 }%
```

\@@cGl's@

```

1606 \def\@@cGl's@#1#2[#3]{%
1607   \gl'sxtrifcounttrigger{#2}%
1608   {%
1609     \cGl'sformat{#2}{#3}%
1610     \gl'sunset{#2}%
1611   }%
1612   {%
1613     \@Gl's@{#1}{#2}[#3]%
1614   }%
1615 }%
```

\@@cGl'spl@

```

1616 \def\@@cGl'spl@#1#2[#3]{%
1617   \gl'sxtrifcounttrigger{#2}%
1618   {%
1619     \cGl'splformat{#2}{#3}%
1620     \gl'sunset{#2}%
1621   }%
1622   {%
1623     \@Gl'spl@{#1}{#2}[#3]%
1624   }%
1625 }%
```

```

\@cGLS@
1626 \def\@cGLS@#1#2[#3]{%
1627   \glxtrifcounttrigger{#2}%
1628   {%
1629     \cGLSformat{#2}{#3}%
1630     \glset{#2}%
1631   }%
1632   {%
1633     \@GLS@{#1}{#2}[#3]%
1634   }%
1635 }%

\@cGLSpl@
1636 \def\@cGLSpl@#1#2[#3]{%
1637   \glxtrifcounttrigger{#2}%
1638   {%
1639     \cGLSplformat{#2}{#3}%
1640     \glset{#2}%
1641   }%
1642   {%
1643     \@GLSpl@{#1}{#2}[#3]%
1644   }%
1645 }%
1646 %
1647 % Remove default warnings from \cs{cgl} etc so that it can be used
1648 % interchangeable with \cs{gl} etc.
1649 %\begin{macro}\@cgl@
1650 %   \begin{macrocode}
1651 \def\@cgl@#1#2[#3]{\@gl@{#1}{#2}[#3]}

\@cGls@
1652 \def\@cGls@#1#2[#3]{\@gl@{#1}{#2}[#3]}

\@cglspl@
1653 \def\@cglspl@#1#2[#3]{\@glspl@{#1}{#2}[#3]}

\@cGlspl@
1654 \def\@cGlspl@#1#2[#3]{\@glspl@{#1}{#2}[#3]}

    Add all upper case versions not provided by glossaries.

\cGLS
1655 \newrobustcmd*{\cGLS}{\@glshyp@opt\@cGLS}

\@cGLS   Defined the un-starred form. Need to determine if there is a final optional argument
1656 \newcommand*{\@cGLS}[2][{}]{%
1657   \new@ifnextchar[{\@cGLS@{#1}{#2}}{\@cGLS@{#1}{#2}[]}]%
1658 }

```

```

\@cGLS@
1659 \def\@cGLS@#1#2[#3]{\@GLS@{#1}{#2}[#3]}

\cGLSformat  Format used by \cGLS if entry only used once on previous run. The first argument is the label,
              the second argument is the insert text.
1660 \newcommand*\cGLSformat[2]{%
1661   \expandafter\mfirstucMakeUppercase\expandafter{\cglformat{#1}{#2}}%
1662 }

\cGLSpl
1663 \newrobustcmd*\cGLSpl{\@gls@hyp@opt\@cGLSpl}

\@cGLSpl  Defined the un-starred form. Need to determine if there is a final optional argument
1664 \newcommand*\@cGLSpl[2][{}]{%
1665   \new@ifnextchar[\@cGLSpl@{#1}{#2}}{\@cGLSpl@{#1}{#2}[{}]}%
1666 }

\@cGLSpl@
1667 \def\@cGLSpl@#1#2[#3]{\@GLSpl@{#1}{#2}[#3]}

\cGLSplformat  Format used by \cGLSpl if entry only used once on previous run. The first argument is the
               label, the second argument is the insert text.
1668 \newcommand*\cGLSplformat[2]{%
1669   \expandafter\mfirstucMakeUppercase\expandafter{\cglsplformat{#1}{#2}}%
1670 }

        Modify the trigger formats to check for the regular attribute.

\cglformat
1671 \renewcommand*\cglformat[2]{%
1672   \glsifregular{#1}
1673   {\glsentryfirst{#1}}%
1674   {\ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}}%#2%
1675 }

\cGlsformat
1676 \renewcommand*\cGlsformat[2]{%
1677   \glsifregular{#1}
1678   {\Glsentryfirst{#1}}%
1679   {\ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}}%#2%
1680 }

\cglsplformat
1681 \renewcommand*\cglsplformat[2]{%
1682   \glsifregular{#1}
1683   {\glsentryfirstplural{#1}}%
1684   {\ifglshaslong{#1}{\glsentrylongpl{#1}}{\glsentryfirstplural{#1}}}%#2%
1685 }

```

\cGlsplformat

```
1686 \renewcommand*{\cGlsplformat}[2]{%
1687   \glsifregular{#1}
1688   {\Glsentryfirstplural{#1}}%
1689   {\ifglshaslong{#1}{\Glsentrylongpl{#1}}{\Glsentryfirstplural{#1}}}\#2%
1690 }
```

New code similar to above for unit counting.

defunitcounters

```
1691 \newcommand*{\@@newglossaryentry@defunitcounters}{%
1692   \edef\@glo@countunit{\csuse{\glxtr@categoryattr@@\@glo@category @unitcount}}%
1693   \ifdefined\@glo@countunit
1694     {}%
1695     {%
1696       \@glxtr@ifunitcounter{\@glo@countunit}%
1697       {}%
1698       {\expandafter\@glxtr@addunitcounter\expandafter{\@glo@countunit}}%
1699     }%
1700 }
```

r@unitcountlist List to keep track of which counters are being used by the entry unit count facility.

```
1701 \newcommand*{\@glxtr@unitcountlist}{}
```

@addunitcounter

```
1702 \newcommand*{\@glxtr@addunitcounter}[1]{%
1703   \listadd{\@glxtr@unitcountlist}{#1}%
1704   \ifcsundef{glxtr@theunit@#1}
1705     {%
1706       \ifcsdef{theH#1}%
1707       {\csdef{glxtr@theunit@#1}{\csuse{theH#1}}}%
1708       {\csdef{glxtr@theunit@#1}{\csuse{the#1}}}%
1709     }%
1710   {}%
1711 }
```

r@ifunitcounter

```
1712 \newcommand*{\@glxtr@ifunitcounter}[3]{%
1713   \xifinlist{#1}{\@glxtr@unitcountlist}{#2}{#3}%
1714 }
```

urrentunitcount

```
1715 \newcommand*{\@glxtr@currentunitcount}[1]{%
1716   glo@\glsetoklabel{#1}@currunit@\glsetattribute{#1}{unitcount}.%
1717   \csuse{glxtr@theunit@\glsetattribute{#1}{unitcount}}%
1718 }
```

previousunitcount

```
1719 \newcommand*\@glxtr@previousunitcount[1]{%
1720   glo@\glstdetoklabel{#1}@prevunit@\glsggetattribute{#1}{unitcount}.%
1721   \csuse{glxtr@theunit@\glsggetattribute{#1}{unitcount}}}%
1722 }
```

t@currunitcount

```
1723 \newcommand*\@glxtr@increment@currunitcount[1]{%
1724   \glshasattribute{#1}{unitcount}%
1725   {%
1726     \edef\@glxtr@csname{\@glxtr@currentunitcount{#1}}%
1727     \ifcsundef{\@glxtr@csname}%
1728     {%
1729       \csgdef{\@glxtr@csname}{1}%
1730       \listcsxadd
1731       {glo@\glstdetoklabel{#1}@unitlist}%
1732       {\glsggetattribute{#1}{unitcount}.%
1733       \csuse{glxtr@theunit@\glsggetattribute{#1}{unitcount}}}%
1734     }%
1735   }%
1736   {%
1737     \csxdef{\@glxtr@csname}%
1738     {\number\numexpr\csname\@glxtr@csname\endcsname+1}%
1739   }%
1740 }%
1741 {}%
1742 }
```

t@currunitcount

```
1743 \newcommand*\@glxtr@local@increment@currunitcount[1]{%
1744   \glshasattribute{#1}{unitcount}%
1745   {%
1746     \edef\@glxtr@csname{\@glxtr@currentunitcount{#1}}%
1747     \ifcsundef{\@glxtr@csname}%
1748     {%
1749       \csdef{\@glxtr@csname}{1}%
1750       \listcseadd
1751       {glo@\glstdetoklabel{#1}@unitlist}%
1752       {\glsggetattribute{#1}{unitcount}.%
1753       \csuse{glxtr@theunit@\glsggetattribute{#1}{unitcount}}}%
1754     }%
1755   }%
1756   {%
1757     \csedef{\@glxtr@csname}%
1758     {\number\numexpr\csname\@glxtr@csname\endcsname+1}%
1759   }%
1760 }%
1761 {}%
1762 }
```

r@currunitcount

```
1763 \newcommand*{\@glxstr@currunitcount}[2]{%
1764   \ifcsundef
1765     {glo@\glstetoklabel{#1}@currunit@#2}%
1766     {0}%
1767     {\csuse{glo@\glstetoklabel{#1}@currunit@#2}}%
1768 }%
```

r@prevunitcount

```
1769 \newcommand*{\@glxstr@prevunitcount}[2]{%
1770   \ifcsundef
1771     {glo@\glstetoklabel{#1}@prevunit@#2}%
1772     {0}%
1773     {\csuse{glo@\glstetoklabel{#1}@prevunit@#2}}%
1774 }%
```

entryunitcount

```
1775 \newcommand*{\glsenableentryunitcount}{%
  Enable new fields:
1776   \appto\@newglossaryentry@defcounters{\@newglossaryentry@defunitcounters}%
  Just in case the user has switched on the docdef option.
1777   \renewcommand*{\gls@defdocnewglossaryentry}{%
1778     \renewcommand*\newglossaryentry[2]{%
1779       \PackageError{glossaries}{\string\newglossaryentry\space
1780         may only be used in the preamble when entry counting has
1781         been activated}{If you use \string\glsenableentryunitcount\space
1782         you must place all entry definitions in the preamble not in
1783         the document environment}%
1784     }%
1785   }%
```

New commands to access new fields:

```
1786 \newcommand*{\glsentrycurrcount}[1]{%
1787   \@glxstr@currunitcount{##1}{\glsgetattribute{##1}{unitcount}}.%
1788   \csuse{glxstr@theunit@\glsgetattribute{##1}{unitcount}}}%
1789 }%
1790 \newcommand*{\glsentryprevcount}[1]{%
1791   \@glxstr@prevunitcount{##1}{\glsgetattribute{##1}{unitcount}}.%
1792   \csuse{glxstr@theunit@\glsgetattribute{##1}{unitcount}}}%
1793 }%
```

Access total count:

```
1794 \newcommand*{\glsentryprevtotalcount}[1]{%
1795   \ifcsundef{glo@\glstetoklabel{##1}@prevunittotal}%
1796     {0}%
1797     {%
1798       \number\csuse{glo@\glstetoklabel{##1}@prevunittotal}
1799     }%
1800 }%
```

Access max value:

```

1801 \newcommand*{\glstentryprevmaxcount}[1]{%
1802   \ifcsundef{glo@\glstetoklabel{##1}@prevunitmax}%
1803     {0}%
1804     {%
1805       \number\csuse{glo@\glstetoklabel{##1}@prevunitmax}
1806     }%
1807   }%

```

Adjust post unset and reset:

```

1808 \let\@glstxtr@entryunitcount@org@unset\glstxtrpostunset
1809 \renewcommand*{\glstxtrpostunset}[1]{%
1810   \@glstxtr@entryunitcount@org@unset{##1}%
1811   \@glst@increment@currunitcount{##1}%
1812 }%
1813 \let\@glstxtr@entryunitcount@org@localunset\glstxtrpostlocalunset
1814 \renewcommand*{\glstxtrpostlocalunset}[1]{%
1815   \@glstxtr@entryunitcount@org@localunset{##1}%
1816   \@glst@local@increment@currunitcount{##1}%
1817 }%
1818 \let\@glstxtr@entryunitcount@org@reset\glstxtrpostreset
1819 \renewcommand*{\glstxtrpostreset}[1]{%
1820   \glshasattribute{##1}{unitcount}%
1821   {%
1822     \edef\@glstxtr@csname{\@glstxtr@currentunitcount{##1}}%
1823     \ifcsundef{\@glstxtr@csname}%
1824       {}%
1825       {\csgdef{\@glstxtr@csname}{0}}%
1826     }%
1827   }%
1828 }%
1829 \let\@glstxtr@entryunitcount@org@localreset\glstxtrpostlocalreset
1830 \renewcommand*{\glstxtrpostlocalreset}[1]{%
1831   \@glstxtr@entryunitcount@org@localreset{##1}%
1832   \glshasattribute{##1}{unitcount}%
1833   {%
1834     \edef\@glstxtr@csname{\@glstxtr@currentunitcount{##1}}%
1835     \ifcsundef{\@glstxtr@csname}%
1836       {}%
1837       {\csdef{\@glstxtr@csname}{0}}%
1838     }%
1839   }%
1840 }%

```

Modifications to take into account the attributes that govern whether the entry should be unset.

```

1841 \let\@cglst@\@cglst@
1842 \let\@cglstpl@\@cglstpl@
1843 \let\@cGLS@\@cGLS@
1844 \let\@cGLstpl@\@cGLstpl@

```



```

1845 \let\@cGLS@ \@cGLS@
1846 \let\@cGLSpl@ \@cGLSpl@

Write information to the aux file.

1847 \AtEndDocument{\@gls@write@entryunitcounts}%
1848 \renewcommand*{\@gls@entry@unitcount}[3]{%
1849   \csgdef{glo@\glsdetoklabel{##1}@prevunit@##3}{##2}%
1850   \ifcsundef{glo@\glsdetoklabel{##1}@prevunittotal}%
1851   {\csgdef{glo@\glsdetoklabel{##1}@prevunittotal}{##2}}%
1852   {%
1853     \csxdef{glo@\glsdetoklabel{##1}@prevunittotal}{
1854       \number\numexpr\csuse{glo@\glsdetoklabel{##1}@prevunittotal}+##2}%
1855     }%
1856     \ifcsundef{glo@\glsdetoklabel{##1}@prevunitmax}%
1857     {\csgdef{glo@\glsdetoklabel{##1}@prevunitmax}{##2}}%
1858     {%
1859       \ifnum\csuse{glo@\glsdetoklabel{##1}@prevunitmax}<##2
1860       \csgdef{glo@\glsdetoklabel{##1}@prevunitmax}{##2}%
1861       \fi
1862     }%
1863   }%
1864 \let\glsenableentryunitcount\relax
1865 \renewcommand*{\glsenableentrycount}{%
1866   \PackageError{glossaries-extra}{\string\glsenableentrycount\space
1867     can't be used with \string\glsenableentryunitcount}%
1868   {Use one or other but not both commands}%
1869 }%
1870 }
1871 \@onlypreamble\glsenableentryunitcount

```

entry@unitcount

```

1872 \newcommand*{\@gls@entry@unitcount}[3]{}

```

ryunitcounts@do

```

1873 \newcommand*{\@gls@write@entryunitcounts@do}[1]{%
1874   \immediate\write\@auxout
1875   {\string\@gls@entry@unitcount
1876     {\@glsentry}%
1877     {\@glsxtr@currunitcount{\@glsentry}{#1}%
1878     }%
1879     {#1}}%
1880 }

```

entryunitcounts

```

1881 \newcommand*{\@gls@write@entryunitcounts}{%
1882   \immediate\write\@auxout
1883   {\string\providecommand*{\string\@gls@entry@unitcount}[3]{}}%
1884   \count@=0\relax
1885   \forallglsentries{\@glsentry}{%

```

```

1886 \glsattribute{\@glsentry}{unitcount}%
1887 {%
1888 \ifglsused{\@glsentry}%
1889 {%
1890 \forlistcsloop
1891 {\@gls@write@entryunitcounts@do}%
1892 {glo@\glsdetoklabel{\@glsentry}@unitlist}%
1893 }%
1894 {}%
1895 \advance\count@ by \@ne
1896 }%
1897 {}%
1898 }%
1899 \ifnum\count@=0
1900 \GlossariesExtraWarningNoLine{Entry counting has been enabled
1901 \MessageBreak with \string\glsenableentryunitcount\space but the
1902 \MessageBreak attribute ‘unitcount’ hasn’t
1903 \MessageBreak been assigned to any of the defined
1904 \MessageBreak entries}%
1905 \fi
1906 }

```

`\glsenableentryunitcount` The first argument is the list of categories, the second argument is the value of the entrycount attribute and the third is the counter name.

```

1907 \newcommand*{\GlsXtrEnableEntryUnitCounting}[3]{%

```

Enable entry counting:

```

1908 \glsenableentryunitcount

```

Redefine `\gls` etc:

```

1909 \renewcommand*{\gls}{\cgl}%
1910 \renewcommand*{\Gls}{\cGls}%
1911 \renewcommand*{\glspl}{\cglspl}%
1912 \renewcommand*{\Glspl}{\cGlspl}%
1913 \renewcommand*{\GLS}{\cGLS}%
1914 \renewcommand*{\GLSpl}{\cGLSpl}%

```

Set the entrycount attribute:

```

1915 \@glsxtr@setentryunitcountunsetattr{#1}{#2}{#3}%

```

In case this command is used again:

```

1916 \let\GlsXtrEnableEntryUnitCounting\@glsxtr@setentryunitcountunsetattr
1917 \renewcommand*{\GlsXtrEnableEntryCounting}[2]{%
1918 \PackageError{glossaries-extra}{\string\GlsXtrEnableEntryCounting\space
1919 can’t be used with \string\GlsXtrEnableEntryUnitCounting}%
1920 {Use one or other but not both commands}}%
1921 }

```

`\glsxtr@setentryunitcountunsetattr`

```

1922 \newcommand*{\@glsxtr@setentryunitcountunsetattr}[3]{%
1923 \@for\@glsxtr@cat:=#1\do

```

```

1924 {%
1925   \ifdefempty{\@glstr@cat}{}%
1926   {%
1927     \glsssetcategoryattribute{\@glstr@cat}{entrycount}{#2}%
1928     \glsssetcategoryattribute{\@glstr@cat}{unitcount}{#3}%
1929   }%
1930 }%
1931 }

```

1.3.6 Acronym Modifications

It's more consistent to use the abbreviation code for acronyms, but make some adjustments to allow for continued use of the glossaries package's custom acronym format. (For example, user may already have defined some acronym styles with `\newacronymstyle` which they would like to continue to use.) The original glossaries acronym code can be restored with `\RestoreAcronyms`, but adjust `\SetGenericNewAcronym` so that `\newacronym` adds the category.

`\SetGenericNewAcronym`

```

1932 \renewcommand*{\SetGenericNewAcronym}{%
1933   \let\@Gls@entryname\@Gls@acrenryname
1934   \renewcommand{\newacronym}[4][]{%
1935     \ifdefempty{\@glssacronymlists}%
1936     {%
1937       \def\@glo@type{\acronymtype}%
1938       \setkeys{glossentry}{##1}%
1939       \DeclareAcronymList{\@glo@type}%
1940     }%
1941   }%
1942   \glsskeylisttok{##1}%
1943   \glsslabeltok{##2}%
1944   \glssshorttok{##3}%
1945   \glsslongtok{##4}%
1946   \newacronymhook
1947   \protected@edef\@do@newglossaryentry{%
1948     \noexpand\newglossaryentry{\the\glsslabeltok}%
1949     {%
1950       type=\acronymtype,%
1951       name={\expandonce{\acronymentry{##2}}},%
1952       sort={\acronymssort{\the\glssshorttok}{\the\glsslongtok}},%
1953       text={\the\glssshorttok},%
1954       short={\the\glssshorttok},%
1955       shortplural={\the\glssshorttok\noexpand\acrpluralsuffix},%
1956       long={\the\glsslongtok},%
1957       longplural={\the\glsslongtok\noexpand\acrpluralsuffix},%
1958       category=acronym,%
1959       \GenericAcronymFields,%
1960       \the\glsskeylisttok

```

```

1961     }%
1962 }%
1963 \do@newglossaryentry
1964 }%
1965 \renewcommand*{\acrfullfmt}[3]{%
1966   \glslink{##1}{##2}{\genacrfullformat{##2}{##3}}}%
1967 \renewcommand*{\Acrfullfmt}[3]{%
1968   \glslink{##1}{##2}{\Genacrfullformat{##2}{##3}}}%
1969 \renewcommand*{\ACRfullfmt}[3]{%
1970   \glslink{##1}{##2}{%
1971     \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}%
1972 \renewcommand*{\acrfullplfmt}[3]{%
1973   \glslink{##1}{##2}{\genplacrfullformat{##2}{##3}}}%
1974 \renewcommand*{\Acrfullplfmt}[3]{%
1975   \glslink{##1}{##2}{\Genplacrfullformat{##2}{##3}}}%
1976 \renewcommand*{\ACRfullplfmt}[3]{%
1977   \glslink{##1}{##2}{%
1978     \mfirstucMakeUppercase{\genplacrfullformat{##2}{##3}}}%
1979 \renewcommand*{\glsentryfull}[1]{\genacrfullformat{##1}{}}}%
1980 \renewcommand*{\Glsentryfull}[1]{\Genacrfullformat{##1}{}}}%
1981 \renewcommand*{\glsentryfullpl}[1]{\genplacrfullformat{##1}{}}}%
1982 \renewcommand*{\Glsentryfullpl}[1]{\Genplacrfullformat{##1}{}}}%
1983 }

```

This will cause a problem for glossaries that contain a mixture of acronyms and abbreviations, so redefine `\newacronym` to use the new abbreviation interface.

First save the original definitions:

```

1984 \let\@glstr@org@setacronymstyle\setacronymstyle
1985 \let\@glstr@org@newacronymstyle\newacronymstyle

```

msAbbreviations Make acronyms use the same interface as abbreviations. Note that `\newacronymstyle` has a different implementation to `\newabbreviationstyle` so disable `\newacronymstyle` and `\setacronymstyle`.

```

1986 \newcommand*{\MakeAcronymsAbbreviations}{%
1987   \renewcommand*{\newacronym}[4][ ]{%
1988     \newabbreviation[type=\acronymtype,category=acronym,##1]{##2}{##3}{##4}%
1989   }%
1990   \renewcommand*{\firstacronymfont}[1]{\glsfirstabbrvfont{##1}}%
1991   \renewcommand*{\acronymfont}[1]{\glsabbrvfont{##1}}%
1992   \renewcommand*{\setacronymstyle}[1]{%
1993     \PackageError{glossaries-extra}{\string\setacronymstyle{##1}}
1994     unavailable.
1995     Use \string\setabbreviationstyle\space instead.
1996     The original acronym interface can be restored with
1997     \string\RestoreAcronyms}{}%
1998 }%
1999 \renewcommand*{\newacronymstyle}[1]{%
2000   \GlossariesExtraWarning{New acronym style ‘##1’ won’t be
2001   available unless you restore the original acronym interface with

```

```

2002      \string\RestoreAcronyms}%
2003      \@glstr@org@newacronymstyle{##1}%
2004    }%
2005  }

```

Switch acronyms to abbreviations:

```
2006 \MakeAcronymsAbbreviations
```

RestoreAcronyms Restore acronyms to glossaries interface.

```

2007 \newcommand*{\RestoreAcronyms}{%
2008   \SetGenericNewAcronym
2009   \renewcommand{\firstacronymfont}[1]{\acronymfont{##1}}%
2010   \renewcommand{\acronymfont}[1]{##1}%
2011   \let\setacronymstyle\@glstr@org@setacronymstyle
2012   \let\newacronymstyle\@glstr@org@newacronymstyle
2013   \let\@gls@link@checkfirsthyper\@glstr@org@checkfirsthyper
2014   \glssetcategoryattribute{acronym}{regular}{false}%
2015   \setacronymstyle{long-short}%
2016 }

```

\glsacspace Allow the user to customise the maximum value.

```

2017 \renewcommand*{\glsacspace}[1]{%
2018   \settowidth{\dimen@}{(\firstacronymfont{\glsentryshort{#1}})}%
2019   \ifdim\dimen@<\glsacspacemax~\else\space\fi
2020 }

```

\glsacspacemax Value used in the above.

```
2021 \newcommand*{\glsacspacemax}{3em}
```

1.3.7 Indexing and Displaying Glossaries

From time-to-time users ask if they can have one glossary sorted normally and another sorted by definition or usage. With the base glossaries package this can only be achieved with the “noidx” commands (Option 1). This is an attempt to mix and match.

First we need a list of the glossaries that require **makeindex/xindy**.

r@reg@glosslist

```
2022 \newcommand*{\@glstr@reg@glosslist}{}
```

Save the original definition of **\makeglossaries**:

```
2023 \let\@glstr@org@makeglossaries\makeglossaries
```

Redefine **\makeglossaries** to take an optional argument. This should be empty for the usual behaviour (all glossaries need processing with an indexing application) or a comma-separated list of glossary labels indicating those glossaries that should be processed with an indexing application.

\makeglossaries

```

2024 \renewcommand*{\makeglossaries}[1][]{%
2025   \ifblank{#1}%
2026   {\@glstr@org@makeglossaries}%
2027   {%
2028     \edef\@glstr@reg@glosslist{#1}%
2029     \ifundef{\glswrite}{\newwrite\glswrite}{}%
2030     \protected@write\@auxout{}{\string\providecommand
2031       \string\@glstr@order[1]}{}
2032     \protected@write\@auxout{}{\string\providecommand
2033       \string\@istfilename[1]}{}
2034     \protected@write\@auxout{}{\string\@istfilename{\istfilename}}%
2035     \protected@write\@auxout{}{\string\@glstr@order{\glstr@order}}
2036     \write\@auxout{\string\providecommand\string\@glstr@reference[3]}{}%

```

Iterate through each supplied glossary type and activate it.

```

2037   \@for\@glo@type:=#1\do{%
2038     \ifdefempty{\@glo@type}{\@makeglossary{\@glo@type}}%
2039   }%

```

New glossaries must be created before \makeglossaries:

```

2040 \renewcommand*\newglossary[4][]{%
2041 \PackageError{glossaries}{New glossaries
2042 must be created before \string\makeglossaries}{You need
2043 to move \string\makeglossaries\space after all your
2044 \string\newglossary\space commands}}%

```

Any subsequent instances of this command should have no effect

```

2045 \let\@makeglossary\relax
2046 \let\makeglossary\relax
2047 \let\makeglossaries\relax

```

Disable all commands that have no effect after \makeglossaries

```

2048 \@disable@onlypremakeg

```

Allow see key:

```

2049 \let\glstr@checkseeallowed\relax

```

Suppress warning about no \makeglossaries

```

2050 \let\warn@nomakeglossaries\relax
2051 \def\warn@noprintglossary{%
2052   \GlossariesWarningNoLine{No \string\printglossary\space
2053     or \string\printglossaries\space
2054     found.^^J(Remove \string\makeglossaries\space if you don't
2055 want
2056   any glossaries.)^^JThis document will not have a glossary}%
2057 }%

```

Adjust display number list to check for type:

```

2058 \renewcommand*\glstrdisplaynumberlist[1]{%
2059 \expandafter\DTLifinlist\expandafter{##1}{\@glstr@reg@glosslist}%
2060 {\@glstr@idx@displaynumberlist{##1}}%

```

```

2061   {\@glxtr@noidx@displaynumberlist{##1}}}%
2062 }%

```

Adjust entry list:

```

2063 \renewcommand*{\glstrynumberlist}[1]{%
2064   \expandafter\DTLifinlist\expandafter{##1}{\@glxtr@reg@glosslist}%
2065   {\@glxtr@idx@entrynumberlist{##1}}}%
2066   {\@glxtr@noidx@entrynumberlist{##1}}}%
2067 }%

```

Adjust number list loop

```

2068 \renewcommand*{\glslnumberlistloop}[2]{%
2069   \expandafter\DTLifinlist\expandafter{##1}{\@glxtr@reg@glosslist}%
2070   {%
2071     \PackageError{glossaries-extra}{\string\glslnumberlistloop\space
2072       not available for glossary ‘##1’}{}%
2073   }%
2074   {\@glxtr@noidx@numberlistloop{##1}{##2}}}%
2075 }%

```

Only sanitize sort for normal indexing glossaries.

```

2076 \renewcommand*{\glsprestandardsort}[3]{%
2077   \expandafter\DTLifinlist\expandafter{##2}{\@glxtr@reg@glosslist}%
2078   {%
2079     \glsdosanitizesort
2080   }%
2081   {%
2082     \ifglssanitizesort
2083       \@gl@noidx@sanitizesort
2084     \else
2085       \@gl@noidx@nosanitizesort
2086     \fi
2087   }%
2088 }%

```

Unlike \makenoidxglossaries we can't automatically set sanitizesort=false. All entries must be defined in the preamble.

```

2089 \renewcommand*{\new@glossaryentry}[2]{%
2090   \PackageError{glossaries-extra}{Glossary entries must be defined
2091     in the preamble\MessageBreak when you use the optional argument
2092     of \string\makeglossaries}{Either move your definitions to the
2093     preamble or don't use the optional argument of
2094     \string\makeglossaries}%
2095 }%

```

Only activate sort key for glossaries that aren't listed in #1 (glossary label is stored in \@glo@type but this defaults to \gldefaulttype so some expansion is required).

```

2096 \renewcommand*{\@printgloss@setsort}{%
2097   \renewcommand*{\@glo@assign@sortkey}{%
2098     \edef\@glo@type{\@glo@type}%
2099     \expandafter\DTLifinlist\expandafter{\@glo@type}{\@glxtr@reg@glosslist}%

```

```

2100      {%
2101      \@@glo@no@assign@sortkey
2102      }%
2103      {%
2104      \@@glo@assign@sortkey
2105      }%
2106      }%
2107      \def\@glo@sorttype{\@glo@default@sorttype}%
2108  }%

```

Check automake setting:

```

2109      \ifglsautomake
2110      \renewcommand*{\@gls@doautomake}{%
2111      \@for\@gls@type:=\@glsxtr@reg@glosslist\do{%
2112      \ifdefempty{\@gls@type}{\@gls@automake{\@gls@type}}%
2113      }%
2114      }%
2115      \fi
2116  }%
2117 }

```

Display number list for the regular version:

splaynumberlist

```

2118 \let\@glsxtr@idx@displaynumberlist\glsdisplaynumberlist

```

Display number list for the “noidx” version:

splaynumberlist

```

2119 \newcommand*{\@glsxtr@noidx@displaynumberlist}[1]{%
2120 \letcs{\@gls@loclist}{glo\@glsdetoklabel{#1}@loclist}%
2121 \ifdef\@gls@loclist
2122 {%
2123 \def\@gls@noidxloclist@sep{%
2124 \def\@gls@noidxloclist@sep{%
2125 \def\@gls@noidxloclist@sep{%
2126 \glsnumlistsep
2127 }%
2128 \def\@gls@noidxloclist@finalsep{\glsnumlistlastsep}%
2129 }%
2130 }%
2131 \def\@gls@noidxloclist@finalsep{}%
2132 \def\@gls@noidxloclist@prev{}%
2133 \forlistloop{\glsnoidxdisplayloclisthandler}{\@gls@loclist}%
2134 \@gls@noidxloclist@finalsep
2135 \@gls@noidxloclist@prev
2136 }%
2137 {%
2138 ??\glsdoifexists{#1}%
2139 {%
2140 \GlossariesWarning{Missing location list for ‘#1’. Either

```



```

2141         a rerun is required or you haven't referenced the entry.}%
2142     }%
2143 }%
2144 }%
2145

```

And for the number list loop:

@numberlistloop

```

2146 \newcommand*{\@glsxtr@noidx@numberlistloop}[3]{%
2147   \letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
2148   \let\@gls@org@glsnoidxdisplayloc\glsnoidxdisplayloc
2149   \let\@gls@org@glssееformat\glssееformat
2150   \let\glsnoidxdisplayloc#2\relax
2151   \let\glssееformat#3\relax
2152   \ifdef\@gls@loclist
2153   {%
2154     \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
2155   }%
2156   {%
2157     ??\glsdoifexists{#1}%
2158     {%
2159       \GlossariesWarning{Missing location list for ‘##1’. Either
2160         a rerun is required or you haven't referenced the entry.}%
2161     }%
2162   }%
2163   \let\glsnoidxdisplayloc\@gls@org@glsnoidxdisplayloc
2164   \let\glssееformat\@gls@org@glssееformat
2165 }%

```

Same for entry number list.

entrynumberlist

```

2166 \newcommand*{\@glsxtr@noidx@entrynumberlist}[1]{%
2167   \letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
2168   \ifdef\@gls@loclist
2169   {%
2170     \glsnoidxloclist{\@gls@loclist}%
2171   }%
2172   {%
2173     ??\glsdoifexists{#1}%
2174     {%
2175       \GlossariesWarning{Missing location list for ‘#1’. Either
2176         a rerun is required or you haven't referenced the entry.}%
2177     }%
2178   }%
2179 }%

```

entrynumberlist

```

2180 \newcommand*{\@glsxtr@idx@entrynumberlist}[1]{\glsentrynumberlist{#1}}

```

Give a bit of assistance to new users who are confused and don't know how to read transcript messages.

@print@glossary

```

2181 \renewcommand{\@print@glossary}{%
2182   \makeatletter
2183   \@input@{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
2184   \IfFileExists{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
2185   {}%
2186   {\glstrNoGlossaryWarning{\@glo@type}}%
2187   \ifglxindy
2188     \ifcsundef{@xdy@\@glo@type @language}%
2189     {%
2190       \edef\@do@auxoutstuff{%
2191         \noexpand\AtEndDocument{%
2192           \noexpand\immediate\noexpand\write\@auxout{%
2193             \string\providecommand\string\@xdylanguage[2]{}}%
2194           \noexpand\immediate\noexpand\write\@auxout{%
2195             \string\@xdylanguage{\@glo@type}{\@xdy@main@language}}%
2196         }%
2197       }%
2198     }%
2199     {%
2200       \edef\@do@auxoutstuff{%
2201         \noexpand\AtEndDocument{%
2202           \noexpand\immediate\noexpand\write\@auxout{%
2203             \string\providecommand\string\@xdylanguage[2]{}}%
2204           \noexpand\immediate\noexpand\write\@auxout{%
2205             \string\@xdylanguage{\@glo@type}{\csname @xdy@\@glo@type
2206               @language\endcsname}}%
2207         }%
2208       }%
2209     }%
2210     \@do@auxoutstuff
2211     \edef\@do@auxoutstuff{%
2212       \noexpand\AtEndDocument{%
2213         \noexpand\immediate\noexpand\write\@auxout{%
2214           \string\providecommand\string\@gls@codepage[2]{}}%
2215         \noexpand\immediate\noexpand\write\@auxout{%
2216           \string\@gls@codepage{\@glo@type}{\@gls@codepage}}%
2217       }%
2218     }%
2219     \@do@auxoutstuff
2220   \fi
2221   \renewcommand*{\@warn@nomakeglossaries}{%
2222     \GlossariesWarningNoLine{\string\makeglossaries\space
2223       hasn't been used,^^Jthe glossaries will not be updated}%
2224   }%
2225 }
```

Setup the warning text to display if the external file for the given glossary is missing.

`oGlsWarningHead` Header message.

```
2226 \newcommand{\GlsXtrNoGlsWarningHead}[2]{%
2227   This document is incomplete. The external file associated with
2228   the glossary ‘#1’ (which should be called \texttt{#2})
2229   hasn’t been created.%
2230 }
```

`arningEmptyStart` No entries have been added to the glossary.

```
2231 \newcommand{\GlsXtrNoGlsWarningEmptyStart}{%
2232   This has probably happened because there are no entries defined
2233   in this glossary.%
2234 }
```

`arningEmptyMain` The default “main” glossary is empty.

```
2235 \newcommand{\GlsXtrNoGlsWarningEmptyMain}{%
2236   If you don’t want this glossary,
2237   add \texttt{nomain} to your package option list when you load
2238   \texttt{glossaries-extra.sty}. For example:%
2239 }
```

`ingEmptyNotMain` A glossary that isn’t the default “main” glossary is empty.

```
2240 \newcommand{\GlsXtrNoGlsWarningEmptyNotMain}[1]{%
2241   Did you forget to use \texttt{type=#1} when you defined your
2242   entries? If you tried to load entries into this glossary with
2243   \texttt{\string\loadglsentries} did you remember to use
2244   \texttt{[#1]} as the optional argument? If you did, check that
2245   the definitions in the file you loaded all had the type set
2246   to \texttt{\string\glsdefaulttype}.%
2247 }
```

`arningCheckFile` Advisory message to check the file contents.

```
2248 \newcommand{\GlsXtrNoGlsWarningCheckFile}[1]{%
2249   Check the contents of the file \texttt{#1}. If
2250   it’s empty, that means you haven’t indexed any of your entries in this
2251   glossary (using commands like \texttt{\string\gls} or
2252   \texttt{\string\glsadd}) so this list can’t be generated.
2253   If the file isn’t empty, the document build process hasn’t been
2254   completed.%
2255 }
```

`WarningAutoMake` Message when automake option has been used.

```
2256 \newcommand{\GlsXtrNoGlsWarningAutoMake}[1]{%
2257   You may need to rerun \LaTeX. If you already have, it may be that
2258   \TeX’s shell escape doesn’t allow you to run
2259   \ifglxindy xindy\else makeindex\fi. Check the
2260   transcript file \texttt{\jobname.log}. If the shell escape is
```

```

2261 disabled, try one of the following:
2262
2263 \begin{itemize}
2264   \item Run the external (Lua) application:
2265
2266     \texttt{makeglossaries-lite.lua \string"\jobname\string"}
2267
2268   \item Run the external (Perl) application:
2269
2270     \texttt{makeglossaries \string"\jobname\string"}
2271 \end{itemize}
2272
2273 Then rerun \LaTeX\ on this document.
2274 \GlossariesExtraWarning{Rerun required to build the
2275 glossary ‘#1’ or check TeX’s shell escape allows
2276 you to run \ifglxindy xindy\else makeindex\fi}%
2277 }

```

WarningMismatch Mismatching \makenoidxglossaries.

```

2278 \newcommand{\GlsXtrNoGlsWarningMismatch}{%
2279   You need to either replace \texttt{\string\makenoidxglossaries}
2280   with \texttt{\string\makeglossaries} or replace
2281   \texttt{\string\printglossary} (or \texttt{\string\printglossaries}) with
2282   \texttt{\string\printnoidxglossary}
2283   (or \texttt{\string\printnoidxglossaries}) and then rebuild
2284   this document.%
2285 }

```

WarningBuildInfo Build advice.

```

2286 \newcommand{\GlsXtrNoGlsWarningBuildInfo}{%
2287   Try one of the following:
2288   \begin{itemize}
2289     \item Add \texttt{automake} to your package option list when you load
2290       \texttt{glossaries-extra.sty}. For example:
2291
2292       \texttt{\string\usepackage[automake]%
2293         \glsgopenbrace glossaries-extra\glsgclosebrace}
2294
2295     \item Run the external (Lua) application:
2296
2297       \texttt{makeglossaries-lite.lua \string"\jobname\string"}
2298
2299     \item Run the external (Perl) application:
2300
2301       \texttt{makeglossaries \string"\jobname\string"}
2302   \end{itemize}
2303
2304   Then rerun \LaTeX\ on this document.%
2305 }

```

oGlsWarningTail Final paragraph.

```
2306 \newcommand{\GlsXtrNoGlsWarningTail}{%
2307 This message will be removed once the problem has been fixed.%
2308 }
```

GlsWarningNoOut No out file created. Build advice.

```
2309 \newcommand{\GlsXtrNoGlsWarningNoOut}[1]{%
2310 The file \texttt{#1} doesn't exist. This most likely means you haven't used
2311 \texttt{\string\makeglossaries} or you have used
2312 \texttt{\string\nofiles}. If this is just a draft version of the
2313 document, you can suppress this message using the
2314 \texttt{nomissinggls} package option.%
2315 }
```

glossarywarning

```
2316 \newcommand*{@@glstr@defaultnoglossarywarning}[1]{%
2317 \glossarysection[\glossarytoctitle]{\glossarytitle}
2318 \GlsXtrNoGlsWarningHead{#1}{\jobname.\csname @glotype@ \@glo@type @in\endcsname}
2319 \par
2320 \glstrifemptyglossary{#1}%
2321 {%
2322 \GlsXtrNoGlsWarningEmptyStart\space
2323 \ifthenelse{\equal{#1}{main}}{\GlsXtrNoGlsWarningEmptyMain\par
2324 \medskip
2325 \noindent\texttt{\string\usepackage[nomain\ifglstracronym ,acronym\fi]}%
2326 \glsoopenbrace glossaries-extra\glsclosebrace}
2327 \medskip
2328 }%
2329 {\GlsXtrNoGlsWarningEmptyNotMain{#1}}%
2330 }%
2331 {%
2332 \IfFileExists{\jobname.\csname @glotype@ \@glo@type @out\endcsname}
2333 {%
2334 \GlsXtrNoGlsWarningCheckFile
2335 {\jobname.\csname @glotype@ \@glo@type @out\endcsname}
2336
2337 \ifglstrautomake
2338
2339 \GlsXtrNoGlsWarningAutoMake{#1}
2340
2341 \else
2342
2343 \ifthenelse{\equal{#1}{main}}%
2344 {%
2345 \GlsXtrNoGlsWarningEmptyMain\par
2346 \medskip
2347 \noindent\texttt{\string\usepackage[nomain]}%
2348 \glsoopenbrace glossaries-extra\glsclosebrace}
2349 \medskip
```

```

2350     }%
2351     {}%
2352
2353     \ifdefequal\makeglossaries\@no@makeglossaries
2354     {%
2355         \GlsXtrNoGlsWarningMisMatch
2356     }%
2357     {%
2358         \GlsXtrNoGlsWarningBuildInfo
2359     }%
2360     \fi
2361 }%
2362 {%
2363     \GlsXtrNoGlsWarningNoOut
2364     {\jobname.\csname @glotype@\@glo@type @out\endcsname}%
2365 }%
2366 }%
2367 \par
2368 \GlsXtrNoGlsWarningTail
2369 }

```

1.4 Integration with glossaries-accsupp

Provide better integration with the glossaries-accsupp package. (Must be loaded before the main code of glossaries-extra either explicitly or through the accsupp package option.)

These commands have their definitions set according to whether or not glossaries-extra has been loaded.

```

2370 \ifpackageloaded{glossaries-accsupp}
2371 {

```

Define (or redefine) commands to use the accessibility information.

\glsaccessname Display the name value (no link and no check for existence).

```

2372 \newcommand*\glsaccessname}[1]{%
2373     \glsnameaccessdisplay
2374     {%
2375         \glsentryname{#1}%
2376     }%
2377     {#1}%
2378 }

```

\Glsaccessname Display the name value (no link and no check for existence) with the first letter converted to upper case.

```

2379 \newcommand*\Glsaccessname}[1]{%
2380     \glsnameaccessdisplay
2381     {%
2382         \Glsentryname{#1}%
2383     }%

```

```

2384     {#1}%
2385 }

```

`\GLSaccessname` Display the name value (no link and no check for existence) converted to upper case.

```

2386 \newcommand*{\GLSaccessname}[1]{%
2387   \glsnameaccessdisplay
2388   {%
2389     \mfirstucMakeUppercase{\glstryname{#1}}%
2390   }%
2391   {#1}%
2392 }

```

`\glsaccessstext` Display the text value (no link and no check for existence).

```

2393 \newcommand*{\glsaccessstext}[1]{%
2394   \glstextaccessdisplay
2395   {%
2396     \glstrytext{#1}%
2397   }%
2398   {#1}%
2399 }

```

`\Glsaccessstext` Display the text value (no link and no check for existence) with the first letter converted to upper case.

```

2400 \newcommand*{\Glsaccessstext}[1]{%
2401   \glstextaccessdisplay
2402   {%
2403     \Glsstrytext{#1}%
2404   }%
2405   {#1}%
2406 }

```

`\GLSAccessstext` Display the text value (no link and no check for existence) converted to upper case.

```

2407 \newcommand*{\GLSAccessstext}[1]{%
2408   \glstextaccessdisplay
2409   {%
2410     \mfirstucMakeUppercase{\glstrytext{#1}}%
2411   }%
2412   {#1}%
2413 }

```

`glsaccessplural` Display the plural value (no link and no check for existence).

```

2414 \newcommand*{\glsaccessplural}[1]{%
2415   \glspluralaccessdisplay
2416   {%
2417     \glstryplural{#1}%
2418   }%
2419   {#1}%
2420 }

```

Glsaccessplural Display the plural value (no link and no check for existence) with the first letter converted to upper case.

```

2421 \newcommand*{\Glsaccessplural}[1]{%
2422   \glspluralaccessdisplay
2423   {%
2424     \Glsentryplural{#1}%
2425   }%
2426   {#1}%
2427 }

```

GLSaccessplural Display the plural value (no link and no check for existence) converted to upper case.

```

2428 \newcommand*{\GLSaccessplural}[1]{%
2429   \glspluralaccessdisplay
2430   {%
2431     \mfirstucMakeUppercase{\glsentryplural{#1}}%
2432   }%
2433   {#1}%
2434 }

```

\glsaccessfirst Display the first value (no link and no check for existence).

```

2435 \newcommand*{\glsaccessfirst}[1]{%
2436   \glsfirstaccessdisplay
2437   {%
2438     \glsentryfirst{#1}%
2439   }%
2440   {#1}%
2441 }

```

\GLSaccessfirst Display the first value (no link and no check for existence) with the first letter converted to upper case.

```

2442 \newcommand*{\GLSaccessfirst}[1]{%
2443   \glsfirstaccessdisplay
2444   {%
2445     \Glsentryfirst{#1}%
2446   }%
2447   {#1}%
2448 }

```

\GLSaccessfirst Display the first value (no link and no check for existence) converted to upper case.

```

2449 \newcommand*{\GLSaccessfirst}[1]{%
2450   \glsfirstaccessdisplay
2451   {%
2452     \mfirstucMakeUppercase{\glsentryfirst{#1}}%
2453   }%
2454   {#1}%
2455 }

```

cessfirstplural Display the firstplural value (no link and no check for existence).


```

2456 \newcommand*{\glsaccessfirstplural}[1]{%
2457   \glsfirstpluralaccessdisplay
2458   {%
2459     \glsentryfirstplural{#1}%
2460   }%
2461   {#1}%
2462 }

```

glsaccessfirstplural Display the firstplural value (no link and no check for existence) with the first letter converted to upper case.

```

2463 \newcommand*{\Glsaccessfirstplural}[1]{%
2464   \glsfirstpluralaccessdisplay
2465   {%
2466     \Glsentryfirstplural{#1}%
2467   }%
2468   {#1}%
2469 }

```

glsfirstplural Display the firstplural value (no link and no check for existence) converted to upper case.

```

2470 \newcommand*{\GLSaccessfirstplural}[1]{%
2471   \glsfirstpluralaccessdisplay
2472   {%
2473     \mfirstucMakeUppercase{\glsentryfirstplural{#1}}%
2474   }%
2475   {#1}%
2476 }

```

glsaccesssymbol Display the symbol value (no link and no check for existence).

```

2477 \newcommand*{\glsaccesssymbol}[1]{%
2478   \glssymbolaccessdisplay
2479   {%
2480     \glsentrysymbol{#1}%
2481   }%
2482   {#1}%
2483 }

```

Glsaccesssymbol Display the symbol value (no link and no check for existence) with the first letter converted to upper case.

```

2484 \newcommand*{\Glsaccesssymbol}[1]{%
2485   \glssymbolaccessdisplay
2486   {%
2487     \Glsentrysymbol{#1}%
2488   }%
2489   {#1}%
2490 }

```

GLSaccesssymbol Display the symbol value (no link and no check for existence) converted to upper case.

```

2491 \newcommand*{\GLSaccesssymbol}[1]{%

```

```

2492 \glssymbolaccessdisplay
2493 {%
2494 \mfirstucMakeUppercase{\glentrysymbol{#1}}%
2495 }%
2496 {#1}%
2497 }

```

esssymbolplural Display the symbolplural value (no link and no check for existence).

```

2498 \newcommand*{\glsaccesssymbolplural}[1]{%
2499 \glssymbolpluralaccessdisplay
2500 {%
2501 \glentrysymbolplural{#1}%
2502 }%
2503 {#1}%
2504 }

```

esssymbolplural Display the symbolplural value (no link and no check for existence) with the first letter converted to upper case.

```

2505 \newcommand*{\Glsaccesssymbolplural}[1]{%
2506 \glssymbolpluralaccessdisplay
2507 {%
2508 \glentrysymbolplural{#1}%
2509 }%
2510 {#1}%
2511 }

```

esssymbolplural Display the symbolplural value (no link and no check for existence) converted to upper case.

```

2512 \newcommand*{\GLSaccesssymbolplural}[1]{%
2513 \glssymbolpluralaccessdisplay
2514 {%
2515 \mfirstucMakeUppercase{\glentrysymbolplural{#1}}%
2516 }%
2517 {#1}%
2518 }

```

\glsaccessdesc Display the desc value (no link and no check for existence).

```

2519 \newcommand*{\glsaccessdesc}[1]{%
2520 \glsdescriptionaccessdisplay
2521 {%
2522 \glentrydesc{#1}%
2523 }%
2524 {#1}%
2525 }

```

\Glsaccessdesc Display the desc value (no link and no check for existence) with the first letter converted to upper case.

```

2526 \newcommand*{\GLSaccessdesc}[1]{%
2527 \glsdescriptionaccessdisplay

```

```

2528     {%
2529         \Glsentrydesc{#1}%
2530     }%
2531     {#1}%
2532 }

```

`\GLSaccessdesc` Display the desc value (no link and no check for existence) converted to upper case.

```

2533 \newcommand*{\GLSaccessdesc}[1]{%
2534     \glsdescriptionaccessdisplay
2535     {%
2536         \mfirstucMakeUppercase{\glsentrydesc{#1}}%
2537     }%
2538     {#1}%
2539 }

```

`ccessdescplural` Display the descplural value (no link and no check for existence).

```

2540 \newcommand*{\glsaccessdescplural}[1]{%
2541     \glsdescriptionpluralaccessdisplay
2542     {%
2543         \glsentrydescplural{#1}%
2544     }%
2545     {#1}%
2546 }

```

`ccessdescplural` Display the descplural value (no link and no check for existence) with the first letter converted to upper case.

```

2547 \newcommand*{\Glsaccessdescplural}[1]{%
2548     \glsdescriptionpluralaccessdisplay
2549     {%
2550         \Glsentrydescplural{#1}%
2551     }%
2552     {#1}%
2553 }

```

`ccessdescplural` Display the descplural value (no link and no check for existence) converted to upper case.

```

2554 \newcommand*{\GLSaccessdescplural}[1]{%
2555     \glsdescriptionpluralaccessdisplay
2556     {%
2557         \mfirstucMakeUppercase{\glsentrydescplural{#1}}%
2558     }%
2559     {#1}%
2560 }

```

`\glsaccessshort` Display the short form (no link and no check for existence).

```

2561 \newcommand*{\glsaccessshort}[1]{%
2562     \glsshortaccessdisplay
2563     {%
2564         \glsentryshort{#1}%

```

```

2565     }%
2566     {#1}%
2567 }

```

`\Glsaccessshort` Display the short form with first letter converted to uppercase (no link and no check for existence).

```

2568 \newcommand*\Glsaccessshort}[1]{%
2569   \glsshortaccessdisplay
2570   {%
2571     \Glsentryshort{#1}%
2572   }%
2573   {#1}%
2574 }

```

`\GLSaccessshort` Display the short value (no link and no check for existence) converted to upper case.

```

2575 \newcommand*\GLSaccessshort}[1]{%
2576   \glsshortaccessdisplay
2577   {%
2578     \mfirstucMakeUppercase{\Glsentryshort{#1}}%
2579   }%
2580   {#1}%
2581 }

```

`\lsaccessshortpl` Display the short plural form (no link and no check for existence).

```

2582 \newcommand*\lsaccessshortpl}[1]{%
2583   \glsshortpluralaccessdisplay
2584   {%
2585     \Glsentryshortpl{#1}%
2586   }%
2587   {#1}%
2588 }

```

`\lsaccessshortpl` Display the short plural form with first letter converted to uppercase (no link and no check for existence).

```

2589 \newcommand*\lsaccessshortpl}[1]{%
2590   \glsshortpluralaccessdisplay
2591   {%
2592     \Glsentryshortpl{#1}%
2593   }%
2594   {#1}%
2595 }

```

`\LSaccessshortpl` Display the shortplural value (no link and no check for existence) converted to upper case.

```

2596 \newcommand*\LSaccessshortpl}[1]{%
2597   \glsshortpluralaccessdisplay
2598   {%
2599     \mfirstucMakeUppercase{\Glsentryshortpl{#1}}%
2600   }%

```

```

2601     {#1}%
2602 }

```

`\glsaccesslong` Display the long form (no link and no check for existence).

```

2603 \newcommand*{\glsaccesslong}[1]{%
2604     \glslongaccessdisplay{\glsentrylong{#1}}{#1}%
2605 }

```

`\Glsaccesslong` Display the long form (no link and no check for existence).

```

2606
2607 \newcommand*{\Glsaccesslong}[1]{%
2608     \glslongaccessdisplay{\Glsentrylong{#1}}{#1}%
2609 }

```

`\GLSaccesslong` Display the long value (no link and no check for existence) converted to upper case.

```

2610 \newcommand*{\GLSaccesslong}[1]{%
2611     \glslongaccessdisplay
2612     {%
2613         \mfirstucMakeUppercase{\glsentrylong{#1}}%
2614     }%
2615     {#1}%
2616 }

```

`\glsaccesslongpl` Display the long plural form (no link and no check for existence).

```

2617 \newcommand*{\glsaccesslongpl}[1]{%
2618     \glslongpluralaccessdisplay{\glsentrylongpl{#1}}{#1}%
2619 }

```

`\Glsaccesslongpl` Display the long plural form (no link and no check for existence).

```

2620
2621 \newcommand*{\Glsaccesslongpl}[1]{%
2622     \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}%
2623 }

```

`\GLSaccesslongpl` Display the longplural value (no link and no check for existence) converted to upper case.

```

2624 \newcommand*{\GLSaccesslongpl}[1]{%
2625     \glslongpluralaccessdisplay
2626     {%
2627         \mfirstucMakeUppercase{\glsentrylongpl{#1}}%
2628     }%
2629     {#1}%
2630 }

```

End of if part

```

2631 }
2632 {

```

No accessibility support. Just define these commands to do `\glsentry<xxx>`

`\glsaccessname` Display the name value (no link and no check for existence).
2633 `\newcommand*{\glsaccessname}[1]{\glsentryname{#1}}`

`\Glsaccessname` Display the name value (no link and no check for existence) with the first letter converted to upper case.
2634 `\newcommand*{\Glsaccessname}[1]{\Glsentryname{#1}}`

`\GLSaccessname` Display the name value (no link and no check for existence). converted to upper case.
2635 `\newcommand*{\GLSaccessname}[1]{%`
2636 `\protect\mfirstucMakeUppercase{\glsentryname{#1}}}`

`\glsaccesstext` Display the text value (no link and no check for existence).
2637 `\newcommand*{\glsaccesstext}[1]{\glsentrytext{#1}}`

`\Glsaccesstext` Display the text value (no link and no check for existence) with the first letter converted to upper case.
2638 `\newcommand*{\Glsaccesstext}[1]{\Glsentrytext{#1}}`

`\GLSaccesstext` Display the text value (no link and no check for existence). converted to upper case.
2639 `\newcommand*{\GLSaccesstext}[1]{%`
2640 `\protect\mfirstucMakeUppercase{\glsentrytext{#1}}}`

`glsaccessplural` Display the plural value (no link and no check for existence).
2641 `\newcommand*{\glsaccessplural}[1]{\glsentryplural{#1}}`

`Glsaccessplural` Display the plural value (no link and no check for existence) with the first letter converted to upper case.
2642 `\newcommand*{\Glsaccessplural}[1]{\Glsentryplural{#1}}`

`GLSaccessplural` Display the plural value (no link and no check for existence). converted to upper case.
2643 `\newcommand*{\GLSaccessplural}[1]{%`
2644 `\protect\mfirstucMakeUppercase{\glsentryplural{#1}}}`

`\glsaccessfirst` Display the first value (no link and no check for existence).
2645 `\newcommand*{\glsaccessfirst}[1]{\glsentryfirst{#1}}`

`\Glsaccessfirst` Display the first value (no link and no check for existence) with the first letter converted to upper case.
2646 `\newcommand*{\Glsaccessfirst}[1]{\Glsentryfirst{#1}}`

`\GLSaccessfirst` Display the first value (no link and no check for existence). converted to upper case.
2647 `\newcommand*{\GLSaccessfirst}[1]{%`
2648 `\protect\mfirstucMakeUppercase{\glsentryfirst{#1}}}`

`cessfirstplural` Display the firstplural value (no link and no check for existence).
2649 `\newcommand*{\glsaccessfirstplural}[1]{\glsentryfirstplural{#1}}`

cessfirstplural Display the firstplural value (no link and no check for existence) with the first letter converted to upper case.

2650 \newcommand*{\GLsaccessfirstplural}[1]{\Glsentryfirstplural{#1}}

cessfirstplural Display the firstplural value (no link and no check for existence). converted to upper case.

2651 \newcommand*{\GLSaccessfirstplural}[1]{%

2652 \protect\mfirstucMakeUppercase{\glsentryfirstplural{#1}}}

glsaccesssymbol Display the symbol value (no link and no check for existence).

2653 \newcommand*{\glsaccesssymbol}[1]{\glsentrysymbol{#1}}

GLsaccesssymbol Display the symbol value (no link and no check for existence) with the first letter converted to upper case.

2654 \newcommand*{\GLsaccesssymbol}[1]{\Glsentrysymbol{#1}}

GLSaccesssymbol Display the symbol value (no link and no check for existence). converted to upper case.

2655 \newcommand*{\GLSaccesssymbol}[1]{%

2656 \protect\mfirstucMakeUppercase{\glsentrysymbol{#1}}}

esssymbolplural Display the symbolplural value (no link and no check for existence).

2657 \newcommand*{\glsaccesssymbolplural}[1]{\glsentrysymbolplural{#1}}

esssymbolplural Display the symbolplural value (no link and no check for existence) with the first letter converted to upper case.

2658 \newcommand*{\GLsaccesssymbolplural}[1]{\Glsentrysymbolplural{#1}}

esssymbolplural Display the symbolplural value (no link and no check for existence). converted to upper case.

2659 \newcommand*{\GLSaccesssymbolplural}[1]{%

2660 \protect\mfirstucMakeUppercase{\glsentrysymbolplural{#1}}}

\glsaccessdesc Display the desc value (no link and no check for existence).

2661 \newcommand*{\glsaccessdesc}[1]{\glsentrydesc{#1}}

\GLsaccessdesc Display the desc value (no link and no check for existence) with the first letter converted to upper case.

2662 \newcommand*{\GLsaccessdesc}[1]{\Glsentrydesc{#1}}

\GLSaccessdesc Display the desc value (no link and no check for existence). converted to upper case.

2663 \newcommand*{\GLSaccessdesc}[1]{%

2664 \protect\mfirstucMakeUppercase{\glsentrydesc{#1}}}

ccessdescplural Display the descplural value (no link and no check for existence).

2665 \newcommand*{\glsaccessdescplural}[1]{\glsentrydescplural{#1}}

ccessdesclplural Display the desclplural value (no link and no check for existence) with the first letter converted to upper case.
2666 \newcommand*{\GLsaccessdesclplural}[1]{\Glsentrydesclplural{#1}}

ccessdesclplural Display the desclplural value (no link and no check for existence). converted to upper case.
2667 \newcommand*{\GLSaccessdesclplural}[1]{%
2668 \protect\mfirstucMakeUppercase{\glsentrydesclplural{#1}}}

\glsaccessshort Display the short form (no link and no check for existence).
2669 \newcommand*{\glsaccessshort}[1]{\glsentryshort{#1}}

\Glsaccessshort Display the short form with first letter converted to uppercase (no link and no check for existence).
2670 \newcommand*{\Glsaccessshort}[1]{\Glsentryshort{#1}}

\GLSaccessshort Display the short value (no link and no check for existence). converted to upper case.
2671 \newcommand*{\GLSaccessshort}[1]{%
2672 \protect\mfirstucMakeUppercase{\glsentryshort{#1}}}

lsaccessshortpl Display the short plural form (no link and no check for existence).
2673 \newcommand*{\glsaccessshortpl}[1]{\glsentryshortpl{#1}}

lsaccessshortpl Display the short plural form with first letter converted to uppercase (no link and no check for existence).
2674 \newcommand*{\Glsaccessshortpl}[1]{\Glsentryshortpl{#1}}

LSaccessshortpl Display the shortplural value (no link and no check for existence). converted to upper case.
2675 \newcommand*{\GLSaccessshortpl}[1]{%
2676 \protect\mfirstucMakeUppercase{\glsentryshortpl{#1}}}

\glsaccesslong Display the long form (no link and no check for existence).
2677 \newcommand*{\glsaccesslong}[1]{\glsentrylong{#1}}

\Glsaccesslong Display the long form (no link and no check for existence).
2678 \newcommand*{\Glsaccesslong}[1]{\Glsentrylong{#1}}

\GLSaccesslong Display the long value (no link and no check for existence). converted to upper case.
2679 \newcommand*{\GLSaccesslong}[1]{%
2680 \protect\mfirstucMakeUppercase{\glsentrylong{#1}}}

glsaccesslongpl Display the long plural form (no link and no check for existence).
2681 \newcommand*{\glsaccesslongpl}[1]{\glsentrylongpl{#1}}

Glsaccesslongpl Display the long plural form (no link and no check for existence).
2682 \newcommand*{\Glsaccesslongpl}[1]{\Glsentrylongpl{#1}}

GLSaccesslongpl Display the longplural value (no link and no check for existence). converted to upper case.

```
2683 \newcommand*{\GLSaccesslongpl}[1]{%
2684 \protect\mfirstucMakeUppercase{\gl Sentrylongpl{#1}}}
```

End of else part

```
2685 }
```

1.5 Categories

`\glscategory` Add a new storage key that can be used to indicate a category. The default category is general.

```
2686 \glsaddstoragekey{category}{general}{\glscategory}
```

`\glsifcategory` Convenient shortcut to determine if an entry has the given category.

```
2687 \newcommand{\glsifcategory}[4]{%
2688 \ifgl sfieldeq{#1}{category}{#2}{#3}{#4}%
2689 }
```

Categories can have attributes.

`categoryattribute` `\gl ssetcategoryattribute{<category>}{<attribute-label>}{<value>}`

Set (or override if already set) an attribute for the given category.

```
2690 \newcommand*{\gl ssetcategoryattribute}[3]{%
2691 \csdef{@gl sxtr@categoryattr@@#1@#2}{#3}%
2692 }
```

`categoryattribute` `\gl sgetcategoryattribute{<category>}{<attribute-label>}`

Get the value of the given attribute for the given category. Does nothing if the attribute isn't defined.

```
2693 \newcommand*{\gl sgetcategoryattribute}[2]{%
2694 \csuse{@gl sxtr@categoryattr@@#1@#2}%
2695 }
```

`categoryattribute` `\gl shascategoryattribute{<category>}{<attribute-label>}{<true>}{<false>}`

Tests if the category has the given attribute set.

```
2696 \newcommand*{\gl shascategoryattribute}[4]{%
2697 \ifcsvoid{@gl sxtr@categoryattr@@#1@#2}{#4}{#3}%
2698 }
```

`\glsssetattribute` `\glsssetattribute{<entry label>}{<attribute-label>}{<value>}`

Short cut where the category label is obtained from the entry information.

```
2699 \newcommand*{\glsssetattribute}[3]{%
2700   \glsssetcategoryattribute{\glscategory{#1}}{#2}{#3}%
2701 }
```

`\glssgetattribute` `\glssgetattribute{<entry label>}{<attribute-label>}`

Short cut where the category label is obtained from the entry information.

```
2702 \newcommand*{\glssgetattribute}[2]{%
2703   \glssgetcategoryattribute{\glscategory{#1}}{#2}%
2704 }
```

`\glsshasattribute` `\glsshasattribute{<entry label>}{<attribute-label>}{<true>}{<false>}`

Short cut to test if the given attribute has been set where the category label is obtained from the entry information.

```
2705 \newcommand*{\glsshasattribute}[4]{%
2706   \ifglssentryexists{#1}%
2707   {\glsshascategoryattribute{\glscategory{#1}}{#2}{#3}{#4}}%
2708   {#4}%
2709 }
```

`categoryattribute` `\glssifcategoryattribute{<category>}{<attribute-label>}{<value>}{<true part>}{<false part>}`

True if category has the attribute with the given value.

```
2710 \newcommand{\glssifcategoryattribute}[5]{%
2711   \ifcsundef{@glssxtr@categoryattr@#1@#2}%
2712   {#5}%
2713   {\ifcsstring{@glssxtr@categoryattr@#1@#2}{#3}{#4}{#5}}%
2714 }
```

`\glssifattribute` `\glssifattribute{<entry label>}{<attribute-label>}{<value>}{<true part>}{<false part>}`

Short cut to determine if the given entry has a category with the given attribute set.

```
2715 \newcommand{\glsifattribute}[5]{%
2716   \ifglstryexists{#1}%
2717   {\glsifcategoryattribute{\glscategory{#1}}{#2}{#3}{#4}{#5}}%
2718   {#5}%
2719 }
```

Set attributes for the default general category:

```
2720 \glssetcategoryattribute{general}{regular}{true}
```

Acronyms are regular by default, since they're typically just treated like normal words.

```
2721 \glssetcategoryattribute{acronym}{regular}{true}
```

`regularcategory` Convenient shortcut to create add the regular attribute.

```
2722 \newcommand*{\glssetregularcategory}[1]{%
2723   \glssetcategoryattribute{#1}{regular}{true}%
2724 }
```

`ifregularcategory` `\glsifregularcategory{<category>}{<true part>}{<false part>}`

Short cut to determine if a category has the regular attribute explicitly set to true.

```
2725 \newcommand{\glsifregularcategory}[3]{%
2726   \glsifcategoryattribute{#1}{regular}{true}{#2}{#3}%
2727 }
```

`notregularcategory` `\glsifnotregularcategory{<category>}{<true part>}{<false part>}`

Short cut to determine if a category has the regular attribute explicitly set to false.

```
2728 \newcommand{\glsifnotregularcategory}[3]{%
2729   \glsifcategoryattribute{#1}{regular}{false}{#2}{#3}%
2730 }
```

`\glsifregular` `\glsifregular{<entry label>}{<true part>}{<false part>}`

Short cut to determine if an entry has a regular attribute set to true.

```
2731 \newcommand{\glsifregular}[3]{%
2732   \glsifregularcategory{\glscategory{#1}}{#2}{#3}%
2733 }
```

`\glsifnotregular`

`\glsifnotregular{<entry label>}{<true part>}{<>false part>}`

Short cut to determine if an entry has a regular attribute set to false.

```
2734 \newcommand{\glsifnotregular}[3]{%
2735   \glsifnotregularcategory{\glscategory{#1}}{#2}{#3}%
2736 }
```

`\glsforeachincategory`

`\glsforeachincategory[<glossary labels>]{<category-label>}{<glossary-cs>}{<label-cs>}{<body>}`

Iterates through all entries in all the glossaries (or just those listed in *<glossary labels>*) and does *<body>* if the category matches *<category-label>*. The control sequences *<glossary-cs>* and *<label-cs>* may be used in *<body>* to access the glossary label and entry label for the current iteration.

```
2737 \newcommand{\glsforeachincategory}[5][\@glo@types]{%
2738   \forallglossaries[#1]{#3}%
2739   {%
2740     \forlgsentries[#3]{#4}%
2741     {%
2742       \glsifcategory{#4}{#2}{#5}{}%
2743     }%
2744   }%
2745 }
```

`\glsforeachwithattribute`

`\glsforeachwithattribute[<glossary labels>]{<attribute-label>}{<attribute-value>}{<glossary-cs>}{<label-cs>}{<body>}`

Iterates through all entries in all the glossaries (or just those listed in *<glossary labels>*) and does *<body>* if the category attribute *<attribute-label>* matches *<attribute-value>*. The control sequences *<glossary-cs>* and *<label-cs>* may be used in *<body>* to access the glossary label and entry label for the current iteration.

```
2746 \newcommand{\glsforeachwithattribute}[6][\@glo@types]{%
2747   \forallglossaries[#1]{#4}%
2748   {%
2749     \forlgsentries[#4]{#5}%
2750     {%
2751       \glsifattribute{#5}{#2}{#3}{#6}{}%
2752     }%
2753   }%
2754 }
```

If `\newterm` has been defined, redefine it so that it automatically sets the category label to index and add `\glstrpostdescription`.

```
2755 \ifdef\newterm
2756 {%
```

`\newterm`

```
2757 \renewcommand*\newterm}[2] [] {%
2758 \newglossaryentry{#2}%
2759 {type={index},category=index,name={#2},%
2760 description={\glstrpostdescription\nopostdesc},#1}%
2761 }
```

Indexed terms are regular by default.

```
2762 \glsssetcategoryattribute{index}{regular}{true}
```

`trpostdescindex`

```
2763 \newcommand*\glstrpostdescindex{}
2764 }
2765 {}
```

If the `symbols` package option was used, define a similar command for symbols, but set the default sort to the label rather than the name as the symbols will typically contain commands that will confuse `makeindex` and `xindy`.

```
2766 \ifdef\printsymbols
2767 {%
```

`glstrnewsymbol` Unlike `\newterm`, this has a separate argument for the label (since the symbol will likely contain commands).

```
2768 \newcommand*\glstrnewsymbol}[3] [] {%
2769 \newglossaryentry{#2}{name={#3},sort={#2},type=symbols,category=symbol,#1}%
2770 }
```

Symbols are regular by default.

```
2771 \glsssetcategoryattribute{symbol}{regular}{true}
```

`rpostdescsymbol`

```
2772 \newcommand*\glstrpostdescsymbol{}
2773 }
2774 {}
```

Similar for the `numbers` option.

```
2775 \ifdef\printnumbers
2776 {%
```

glsxtrnewnumber

```
2777 \ifdef\printnumbers
2778   \newcommand*{\glsxtrnewnumber}[3] [] {%
2779     \newglossaryentry{#2}{name={#3},sort={#2},type=numbers,category=number,#1}%
2780   }
```

Numbers are regular by default.

```
2781 \glssetcategoryattribute{number}{regular}{true}
```

rpostdescnumber

```
2782 \newcommand*{\glsxtrpostdescnumber}{}

2783 }
2784 {}
```

glsxtrsetcategory Set the category for all listed labels. The first argument is the list of entry labels and the second argument is the category label.

```
2785 \newcommand*{\glsxtrsetcategory}[2] {%
2786   \@for\@glsxtr@label:=#1\do
2787   {%
2788     \glsfieldxddef{\@glsxtr@label}{category}{#2}%
2789   }%
2790 }
```

glsxtrsetcategoryforall Set the category for all entries in the listed glossaries. The first argument is the list of glossary labels and the second argument is the category label.

```
2791 \newcommand*{\glsxtrsetcategoryforall}[2] {%
2792   \forallglossaries[#1]{\@glsxtr@type}{%
2793     \forglentries[\@glsxtr@type]{\@glsxtr@label}%
2794     {%
2795       \glsfieldxddef{\@glsxtr@label}{category}{#2}%
2796     }%
2797   }%
2798 }
```

glsxtrfieldtitlecase

`\glsxtrfieldtitlecase{<label>}{<field>}`

Apply title casing to the contents of the given field.

```
2799 \newcommand*{\glsxtrfieldtitlecase}[2] {%
2800   \expandafter\xcapitalisewords\expandafter
2801     {\csname glo@glsdetoklabel{#1}@#2\endcsname}%
2802 }
```

Provide a convenient way to modify glossary styles without having to define a new style just to convert the first letter of fields to upper case.

`\glossentrydesc` If the `glossdesc` attribute is “firstuc” convert first letter to upper case. If the attribute is “title” use title case.

```

2803 \@ifpackageloaded{glossaries-accsupp}
2804 {
2805   \renewcommand*{\glossentrydesc}[1]{%
2806     \glsdoifexistsorwarn{#1}%
2807     {%
2808       \glssetabbrvfmt{\glscategory{#1}}%
2809       \glshasattribute{#1}{glossdescfont}%
2810       {%
2811         \edef\@glxtr@attrval{\glsggetattribute{#1}{glossdescfont}}%
2812         \ifcsdef{\@glxtr@attrval}%
2813         {%
2814           \letcs{\@glxtr@glossdescfont}{\@glxtr@attrval}%
2815         }%
2816         {%
2817           \GlossariesExtraWarning{Unknown control sequence name
2818             ‘\@glxtr@attrval’ supplied in glossdescfont attribute
2819             for entry ‘#1’. Ignoring}%
2820           \let\@glxtr@glossdescfont\@firstofone
2821         }%
2822       }%
2823       {\let\@glxtr@glossdescfont\@firstofone}%
2824       \glusifattribute{#1}{glossdesc}{firstuc}%
2825       {%
2826         \@glxtr@glossdescfont{\Glsaccessdesc{#1}}%
2827       }%
2828       {%
2829         \glusifattribute{#1}{glossdesc}{title}%
2830         {%
2831           \@glxtr@do@titlecaps@warn
2832           \glsdescriptionaccessdisplay
2833           {%
2834             \@glxtr@glossdescfont{\glxtrfieldtitlecase{#1}{desc}}%
2835           }%
2836           {#1}%
2837         }%
2838         {%
2839           \@glxtr@glossdescfont{\Glsaccessdesc{#1}}%
2840         }%
2841       }%
2842     }%
2843   }
2844 }
2845 {
2846   \renewcommand*{\glossentrydesc}[1]{%
2847     \glsdoifexistsorwarn{#1}%

```

```

2848 {%
2849   \glsetabbrvfmt{\glscategory{#1}}%
2850   \glshasattribute{#1}{glossdescfont}%
2851   {%
2852     \edef\@glxtr@attrval{\glsggetattribute{#1}{glossdescfont}}%
2853     \ifcsdef{\@glxtr@attrval}%
2854     {%
2855       \letcs{\@glxtr@glossdescfont}{\@glxtr@attrval}%
2856     }%
2857     {%
2858       \GlossariesExtraWarning{Unknown control sequence name
2859         '\@glxtr@attrval' supplied in glossdescfont attribute
2860         for entry '#1'. Ignoring}%
2861       \let\@glxtr@glossdescfont\@firstofone
2862     }%
2863   }%
2864   {\let\@glxtr@glossdescfont\@firstofone}%
2865   \gl@ifattribute{#1}{glossdesc}{firstuc}%
2866   {%
2867     \@glxtr@glossdescfont{\Glsentrydesc{#1}}%
2868   }%
2869   {%
2870     \gl@ifattribute{#1}{glossdesc}{title}%
2871     {%
2872       \@glxtr@do@titlecaps@warn
2873       \@glxtr@glossdescfont{\glxtrfieldtitlecase{#1}{desc}}%
2874     }%
2875     {%
2876       \@glxtr@glossdescfont{\Glsentrydesc{#1}}%
2877     }%
2878   }%
2879 }%
2880 }
2881 }

```

`\glossentryname` If the glossname attribute is “firstuc” convert first letter to upper case. If the attribute is “title” use title case.

```

2882 \ifpackageloaded{glossaries-accsupp}
2883 {
2884   \renewcommand*{\glossentryname}[1]{%
2885     \@glsd@ifexistsorwarn{#1}%
2886     {%
2887       \glsetabbrvfmt{\glscategory{#1}}%

```

As from version 1.04, allow the glossnamefont attribute to determine the font applied.

```

2888   \glshasattribute{#1}{glossnamefont}%
2889   {%
2890     \edef\@glxtr@attrval{\glsggetattribute{#1}{glossnamefont}}%
2891     \ifcsdef{\@glxtr@attrval}%
2892     {%

```



```

2893     \letcs{\@glxtr@glossnamefont}{\@glxtr@attrval}%
2894 }%
2895 {%
2896     \GlossariesExtraWarning{Unknown control sequence name
2897     '\@glxtr@attrval' supplied in glossnamefont attribute
2898     for entry '#1'. Reverting to default \string\glsnamefont}%
2899     \let\@glxtr@glossnamefont\glsnamefont
2900 }%
2901 }%
2902 {\let\@glxtr@glossnamefont\glsnamefont}%
2903 \glsifattribute{#1}{glossname}{firstuc}%
2904 {%
2905     \glsnameaccessdisplay
2906     {%
2907         \@glxtr@glossnamefont{\Glsentryname{#1}}%
2908     }%
2909     {#1}%
2910 }%
2911 {%
2912     \glsifattribute{#1}{glossname}{title}%
2913     {%
2914         \@glxtr@do@titlecaps@warn
2915         \glsnameaccessdisplay
2916         {%
2917             \@glxtr@glossnamefont{\glxtrfieldtitlecase{#1}{name}}%
2918         }%
2919         {#1}%
2920     }%
2921     {%
2922         \glsifattribute{#1}{glossname}{uc}%
2923         {%
2924             \glsnameaccessdisplay
2925             {%

```

Hide the label from the upper-casing command.

```

2926         \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
2927         \@glxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}%
2928     }%
2929     {#1}%
2930 }%
2931 {%
2932     \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
2933     \glsnameaccessdisplay
2934     {%
2935         \expandafter\@glxtr@glossnamefont\expandafter{\glo@name}%
2936     }%
2937     {#1}%
2938 }%
2939 }%
2940 }%

```

Do post-name hook:

```

2941     \glxtrpostnamehook{#1}%
2942   }%
2943 }
2944 }
2945 {
2946   \renewcommand*{\glossentryname}[1]{%
2947     \@glxtrdoifexistsorwarn{#1}%
2948     {%
2949       \glsetabbrvfmt{\glscategory{#1}}%
2950       \glshasattribute{#1}{glossnamefont}%
2951       {%
2952         \edef\@glxtr@attrval{\glsetattribute{#1}{glossnamefont}}%
2953         \ifcsdef{\@glxtr@attrval}%
2954         {%
2955           \letcs{\@glxtr@glossnamefont}{\@glxtr@attrval}%
2956         }%
2957         {%
2958           \GlossariesExtraWarning{Unknown control sequence name
2959             ‘\@glxtr@attrval’ supplied in glossnamefont attribute
2960             for entry ‘#1’. Reverting to default \string\glnamefont}%
2961           \let\@glxtr@glossnamefont\glnamefont
2962         }%
2963       }%
2964       {\let\@glxtr@glossnamefont\glnamefont}%
2965       \gl@ifattribute{#1}{glossname}{firstuc}%
2966       {%
2967         \@glxtr@glossnamefont{\Glsentryname{#1}}%
2968       }%
2969       {%
2970         \gl@ifattribute{#1}{glossname}{title}%
2971         {%
2972           \@glxtr@do@titlecaps@warn
2973           \@glxtr@glossnamefont{\glxtrfieldtitlecase{#1}{name}}%
2974         }%
2975         {%
2976           \gl@ifattribute{#1}{glossname}{uc}%
2977           {%

```

Hide the label from the upper-casing command.

```

2978       \letcs{\glo@name}{glo@\glxtr@do@titlecaps@warn}%
2979       \@glxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}%
2980     }%
2981   }%

```

This little trick is used by glossaries to allow the user to redefine \glnamefont to use \makefirstuc. Support it even though they can now use the firstuc attribute.

```

2982       \letcs{\glo@name}{glo@\glxtr@do@titlecaps@warn}%
2983       \expandafter\@glxtr@glossnamefont\expandafter{\glo@name}%
2984     }%

```

```

2985     }%
2986 }%
    Do post-name hook.
2987     \glsxtrpostnamehook{#1}%
2988 }%
2989 }
2990 }

```

\Glossentryname Redefine to set the abbreviation format and accessibility support.

```

2991 \@ifpackageloaded{glossaries-accsupp}
2992 {
2993     \renewcommand*{\Glossentryname}[1]{%
2994         \@glsdoifexistsorwarn{#1}%
2995         {%
2996             \glssetabbrvfmt{\glscategory{#1}}%

```

As from version 1.04, allow the glossnamefont attribute to determine the font applied.

```

2997     \glshasattribute{#1}{glossnamefont}%
2998     {%
2999         \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossnamefont}}%
3000         \ifcsdef{\@glsxtr@attrval}%
3001         {%
3002             \letcs{\@glsxtr@glossnamefont}{\@glsxtr@attrval}%
3003         }%
3004         {%
3005             \GlossariesExtraWarning{Unknown control sequence name
3006                 '\@glsxtr@attrval' supplied in glossnamefont attribute
3007                 for entry '#1'. Reverting to default \string\glsnamefont}%
3008             \let\@glsxtr@glossnamefont\glsnamefont
3009         }%
3010     }%
3011     {\let\@glsxtr@glossnamefont\glsnamefont}%
3012     \glsnameaccessdisplay
3013     {%
3014         \@glsxtr@glossnamefont{\Glsentryname{#1}}%
3015     }%
3016     {#1}%

```

Do post-name hook:

```

3017     \glsxtrpostnamehook{#1}%
3018 }%
3019 }
3020 }
3021 {
3022     \renewcommand*{\Glossentryname}[1]{%
3023         \@glsdoifexistsorwarn{#1}%
3024         {%
3025             \glssetabbrvfmt{\glscategory{#1}}%
3026             \glshasattribute{#1}{glossnamefont}%

```

```

3027    {%
3028        \edef\@glxstr@attrval{\glsggetattribute{#1}{glossnamefont}}%
3029        \ifcsdef{\@glxstr@attrval}%
3030        {%
3031            \letcs{\@glxstr@glossnamefont}{\@glxstr@attrval}%
3032        }%
3033        {%
3034            \GlossariesExtraWarning{Unknown control sequence name
3035                '\@glxstr@attrval' supplied in glossnamefont attribute
3036                for entry '#1'. Reverting to default \string\glsnamefont}%
3037            \let\@glxstr@glossnamefont\glsnamefont
3038        }%
3039    }%
3040    {\let\@glxstr@glossnamefont\glsnamefont}%
3041    \@glxstr@glossnamefont{\Glsentryname{#1}}}%

```

Do post-name hook:

```

3042        \glxstrpostnamehook{#1}%
3043    }%
3044 }
3045 }

```

Provide a convenient way to also index the entries using the standard `\index` mechanism. This may use different actual, encap and escape characters to those used for the glossaries.

`xtrpostnamehook` Hook to append stuff after the name is displayed in the glossary. The argument is the entry's label.

```

3046 \newcommand*{\glxstrpostnamehook}[1]{%
3047     \def\@glsnumberformat{glsnumberformat}%
3048     \glxstrdoautoindexname{#1}{indexname}%

```

Allow categories to hook in here.

```

3049     \csuse{glxstrpostname\glscategory{\glscurrententrylabel}}%
3050 }

```

`xformat@override` Determines if the format key should override the indexing attribute value.

```

3051 \newif\if@glxstr@format@override
3052 \@glxstr@format@overridefalse

```

If overriding is enabled, the `\glshypernumber` command will have to be redefined in the index to use `\hyperpage` instead.

`xFormatOverride`

```

3053 \@ifpackageloaded{hyperref}
3054 {

```

If `hyperref`'s `hyperindex` option is on, then `hyperref` will automatically add `\hyperpage`, so don't add it.

```

3055     \ifHy@hyperindex
3056     \newcommand*{\GlsXtrEnableIndexFormatOverride}{%

```

```

3057      \@glxtr@format@overridetrue
3058      \appto\theindex{\let\glshypernumber\@firstofone}%
3059    }
3060  \else
3061    \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
3062      \@glxtr@format@overridetrue
3063      \appto\theindex{\let\glshypernumber\hyperpage}%
3064    }
3065  \fi
3066 }
3067 {
3068   \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
3069     \@glxtr@format@overridetrue
3070   }
3071 }
3072 \@onlypreamble\GlsXtrEnableIndexFormatOverride

```

doautoindexname

```

3073 \newcommand*{\glxtrdoautoindexname}[2]{%
3074   \glshasattribute{#1}{#2}%
3075   {%

```

Escape any makeindex/xindy characters in the value of the name field. Take care with babel as this won't work if the category code has changed for those characters.

```

3076   \@glxtr@autoindex@setname{#1}%

```

If the attribute value is simply “true” don't add an encap, otherwise use the value as the encap.

```

3077   \protected@edef\@glxtr@attrval{\glsggetattribute{#1}{#2}}%
3078   \if@glxtr@format@override
3079     \ifdefstring{\@glsnumberformat}{\glsgnumberformat}{}%
3080     {\let\@glxtr@attrval\@glsnumberformat}%
3081   \fi
3082   \ifdefstring{\@glxtr@attrval}{true}%
3083   {%
3084     {\eappto\@glo@name{\@glxtr@autoindex@encap\@glxtr@attrval}}%
3085     \expandafter\index\expandafter{\@glo@name}%
3086   }%
3087   {}%
3088 }

```

toindex@setname Assign \@glo@name for use with indexname attribute.

```

3089 \newcommand*{\@glxtr@autoindex@setname}[1]{%
3090   \def\@glo@name{\string\glssentryname{#1}}%
3091   \glsetentryfield{\@glo@sort}{#1}{sort}%
3092   \@gls@checkmkidxchars\@glo@sort
3093   \@glxtr@autoindex@doextra@esc\@glo@sort
3094   \epreto\@glo@name{\@glo@sort\@glxtr@autoindex@at}%
3095 }

```

dex@doextra@esc

3096 \newcommand*{\@glsxtr@autoindex@doextra@esc}[1]{%

Escape the escape character unless it has already been escaped.

```
3097 \ifx\@glsxtr@autoindex@esc\@gls@quotechar
3098 \else
3099 \def\@gls@checkedmkidx{}%
3100 \edef\@glsxtr@checkspch{%
3101 \noexpand\@glsxtr@autoindex@escquote\expandonce{#1}%
3102 \noexpand\@empty\@glsxtr@autoindex@esc\noexpand\@nnil
3103 \@glsxtr@autoindex@esc\noexpand\@empty\noexpand\@glsxtr@endescspch}%
3104 \@glsxtr@checkspch
3105 \let#1\@gls@checkedmkidx\relax
3106 \fi
```

Escape actual character unless it has already been escaped.

```
3107 \ifx\@glsxtr@autoindex@at\@gls@actualchar
3108 \else
3109 \def\@gls@checkedmkidx{}%
3110 \edef\@glsxtr@checkspch{%
3111 \noexpand\@glsxtr@autoindex@escat\expandonce{#1}%
3112 \noexpand\@empty\@glsxtr@autoindex@at\noexpand\@nnil
3113 \@glsxtr@autoindex@at\noexpand\@empty\noexpand\@glsxtr@endescspch}%
3114 \@glsxtr@checkspch
3115 \let#1\@gls@checkedmkidx\relax
3116 \fi
```

Escape level character unless it has already been escaped.

```
3117 \ifx\@glsxtr@autoindex@level\@gls@levelchar
3118 \else
3119 \def\@gls@checkedmkidx{}%
3120 \edef\@glsxtr@checkspch{%
3121 \noexpand\@glsxtr@autoindex@esclevel\expandonce{#1}%
3122 \noexpand\@empty\@glsxtr@autoindex@level\noexpand\@nnil
3123 \@glsxtr@autoindex@level\noexpand\@empty\noexpand\@glsxtr@endescspch}%
3124 \@glsxtr@checkspch
3125 \let#1\@gls@checkedmkidx\relax
3126 \fi
```

Escape encap character unless it has already been escaped.

```
3127 \ifx\@glsxtr@autoindex@encap\@gls@encapchar
3128 \else
3129 \def\@gls@checkedmkidx{}%
3130 \edef\@glsxtr@checkspch{%
3131 \noexpand\@glsxtr@autoindex@escencap\expandonce{#1}%
3132 \noexpand\@empty\@glsxtr@autoindex@encap\noexpand\@nnil
3133 \@glsxtr@autoindex@encap\noexpand\@empty\noexpand\@glsxtr@endescspch}%
3134 \@glsxtr@checkspch
3135 \let#1\@gls@checkedmkidx\relax
3136 \fi
3137 }
```

The user commands here have a preamble-only restriction to ensure they are set before required and also to reduce the chances of complications caused by babel's shorthands.

`\tr@autoindex@at` Actual character for use with `\index`.

```
3138 \newcommand*{\@glxtr@autoindex@at}{}

```

`\trSetActualChar` Set the actual character.

```
3139 \newcommand*{\GlsXtrSetActualChar}[1]{%
3140   \gdef\@glxtr@autoindex@at{#1}%
3141   \def\@glxtr@autoindex@escat##1##2##3\@glxtr@endescspch{%
3142     \@glxtr@autoindex@escspch{#1}{\@glxtr@autoindex@escat}{##1}{##2}{##3}%
3143   }%
3144 }
3145 \@onlypreamble\GlsXtrSetActualChar
3146 \makeatother
3147 \GlsXtrSetActualChar{@}
3148 \makeatletter

```

`\autoindex@encap` Encap character for use with `\index`.

```
3149 \newcommand*{\@glxtr@autoindex@encap}{}

```

`\XtrSetEncapChar` Set the encap character.

```
3150 \newcommand*{\GlsXtrSetEncapChar}[1]{%
3151   \gdef\@glxtr@autoindex@encap{#1}%
3152   \def\@glxtr@autoindex@escencap##1##2##3\@glxtr@endescspch{%
3153     \@glxtr@autoindex@escspch{#1}{\@glxtr@autoindex@escencap}{##1}{##2}{##3}%
3154   }%
3155 }
3156 \GlsXtrSetEncapChar{|}
3157 \@onlypreamble\GlsXtrSetEncapChar

```

`\autoindex@level` Level character for use with `\index`.

```
3158 \newcommand*{\@glxtr@autoindex@level}{}

```

`\XtrSetLevelChar` Set the encap character.

```
3159 \newcommand*{\GlsXtrSetLevelChar}[1]{%
3160   \gdef\@glxtr@autoindex@level{#1}%
3161   \def\@glxtr@autoindex@esclevel##1##2##3\@glxtr@endescspch{%
3162     \@glxtr@autoindex@escspch{#1}{\@glxtr@autoindex@esclevel}{##1}{##2}{##3}%
3163   }%
3164 }
3165 \GlsXtrSetLevelChar{!}
3166 \@onlypreamble\GlsXtrSetLevelChar

```

`\r@autoindex@esc` Escape character for use with `\index`.

```
3167 \newcommand*{\@glxtr@autoindex@esc}{"}

```

`\lsXtrSetEscChar` Set the escape character.

```
3168 \newcommand*{\GlsXtrSetEscChar}[1]{%
3169   \gdef\@glxstr@autoindex@esc{#1}%
3170   \def\@glxstr@autoindex@escquote##1##2##3\@glxstr@endescspch{%
3171     \@glxstr@autoindex@escspch{#1}{\@glxstr@autoindex@escquote}{##1}{##2}{##3}%
3172   }%
3173 }
3174 \GlsXtrSetEscChar{"}
3175 \@onlypreamble\GlsXtrSetEscChar
```

Set if defined. (For example, if doc package has been loaded.) Actual character `\actualchar`:

```
3176 \ifdef\actualchar
3177 {\expandafter\GlsXtrSetActualChar\expandafter{\actualchar}}
3178 {}
```

Quote character `\quotechar`:

```
3179 \ifdef\quotechar
3180 {\expandafter\GlsXtrSetEscChar\expandafter{\quotechar}}
3181 {}
```

Level character `\levelchar`:

```
3182 \ifdef\levelchar
3183 {\expandafter\GlsXtrSetLevelChar\expandafter{\levelchar}}
3184 {}
```

Encap character `\encapchar`:

```
3185 \ifdef\encapchar
3186 {\expandafter\GlsXtrSetEncapChar\expandafter{\encapchar}}
3187 {}
```

`\leto@endescspch`

```
3188 \def\@glxstr@gobbleto@endescspch#1\@glxstr@endescspch{}
```

`\toindex@esc@spch`

<code>\@glxstr@autoindex@escspch{<char>}{<cs>}{<pre>}{<mid>}{<post>}</code>

```
3189 \newcommand*{\@glxstr@autoindex@escspch}[5]{%
3190   \@glstmpb=\expandafter{\@glstmpb\the\toks@}%
3191   \toks@={#3}%
3192   \ifx\@nnil#3\relax
3193     \def\@glxstr@checkspch{\@glxstr@gobbleto@endescspch#5\@glxstr@endescspch}%
3194   \else
3195     \ifx\@nnil#4\relax
3196       \edef\@glstmpb{\the\@glstmpb\the\toks@}%
3197       \def\@glxstr@checkspch{\@glxstr@gobbleto@endescspch
3198         #4#5\@glxstr@endescspch}%
3199     \else
3200       \edef\@glstmpb{\the\@glstmpb\the\toks@
```



```

3201      \@glsxtr@autoindex@esc#1}%
3202      \def\@glsxtr@checkspch{#2#5#1\@nnil#1\@glsxtr@endescspch}%
3203      \fi
3204      \fi
3205      \@glsxtr@checkspch
3206 }

```

`\Glossentrydesc` Redefine to set the abbreviation format and accessibility support.

```

3207 \renewcommand*{\Glossentrydesc}[1]{%
3208   \glsdoifexistsorwarn{#1}%
3209   {%
3210     \glssetabbrvfmt{\glscategory{#1}}%
3211     \Glsaccessdesc{#1}%
3212   }%
3213 }

```

`\Glossentrysymbol` Redefine to set the abbreviation format and accessibility support.

```

3214 \renewcommand*{\Glossentrysymbol}[1]{%
3215   \glsdoifexistsorwarn{#1}%
3216   {%
3217     \glssetabbrvfmt{\glscategory{#1}}%
3218     \Glsaccesssymbol{#1}%
3219   }%
3220 }

```

`\Glossentrysymbol` Redefine to set the abbreviation format and accessibility support.

```

3221 \renewcommand*{\Glossentrysymbol}[1]{%
3222   \glsdoifexistsorwarn{#1}%
3223   {%
3224     \glssetabbrvfmt{\glscategory{#1}}%
3225     \Glsaccesssymbol{#1}%
3226   }%
3227 }

```

Allow initials to be marked but only use the formatting for the tag in the glossary.

`\GlsXtrEnableInitialTagging` Allow initial tagging. The first argument is a list of categories to apply this to. The second argument is the name of the command to use to tag the initials. This can't already be defined for safety unless the starred version is used.

```

3228 \newcommand*{\GlsXtrEnableInitialTagging}{%
3229   \@ifstar\s@glsxtr@enabletagging\@glsxtr@enabletagging
3230 }
3231 \@onlypreamble\GlsXtrEnableInitialTagging

```

`\@glsxtr@enabletagging` Starred version undefines command.

```

3232 \newcommand*{\s@glsxtr@enabletagging}[2]{%
3233   \undef#2%
3234   \@glsxtr@enabletagging{#1}{#2}%
3235 }

```

r@enabletagging Internal command.

```
3236 \newcommand*{\@glsxtr@enabletagging}[2]{%
```

Set attributes for categories given in the first argument.

```
3237 \@for\@glsxtr@cat:=#1\do
3238 {%
3239 \ifdefempty\@glsxtr@cat
3240 {}%
3241 {\glssetcategoryattribute{\@glsxtr@cat}{tagging}{true}}%
3242 }%
3243 \newrobustcmd*#2[1]{##1}%
3244 \def\@glsxtr@taggingcs{#2}%
3245 \renewcommand*\@glsxtr@activate@initialtagging{%
3246 \let#2\@glsxtr@tag
3247 }%
3248 \ifundef\@gls@preglossaryhook
3249 {\GlossariesExtraWarning{Initial tagging requires at least
3250 glossaries.sty v4.19 to work correctly}}%
3251 {}%
3252 }
```

Are we using an old version of mfirstuc that has a bug in \capitalisewords? If so, patch it so we don't have a problem with a combination of tagging and title case.

fu@checkword@do If this command hasn't been defined, then we have pre v2.02 of mfirstuc

```
3253 \ifundef\mfu@checkword@do
3254 {
3255 \newcommand*{\mfu@checkword@do}[1]{%
3256 \ifdefstring{\mfu@checkword@arg}{#1}%
3257 {%
3258 \let\@mfu@domakefirstuc\@firstofone
3259 \listbreak
3260 }%
3261 {}%
3262 }
```

\mfu@checkword \capitalisewords was introduced in mfirstuc v1.06. If \mfu@checkword hasn't been defined mfirstuc is too old to support the title case attribute.

```
3263 \ifundef\mfu@checkword
3264 {
3265 \newcommand{\@glsxtr@do@titlecaps@warn}{%
3266 \GlossariesExtraWarning{mfirstuc.sty too old. Title Caps
3267 support not available}}%
```

One warning should suffice.

```
3268 \let\@glsxtr@do@titlecaps@warn\relax
3269 }
3270 }
3271 {
3272 \renewcommand*{\mfu@checkword}[1]{%
```

```

3273      \def\mfu@checkword@arg{#1}%
3274      \let\@mfu@domakefirstuc\makefirstuc
3275      \forlistloop\mfu@checkword@do\@mfu@nocaplist
3276    }
3277  }
3278 }
3279 {}% no patch required

```

`@titlecaps@warn` Do warning if title case not supported.

```

3280 \newcommand*{\@glxtr@do@titlecaps@warn}{}

```

`@initialtagging` Used in `\printglossary` but at least v4.19 of glossaries required.

```

3281 \newcommand*{\@glxtr@activate@initialtagging}{}

```

`\@glxtr@tag` Definition of tagging command when used in glossary.

```

3282 \newrobustcmd*{\@glxtr@tag}[1]{%
3283   \gl@ifattribute{\glscurrententrylabel}{tagging}{true}%
3284   {\glxtrtagfont{#1}}{#1}%
3285 }

```

`\glxtrtagfont` Used in the glossary.

```

3286 \newcommand*{\glxtrtagfont}[1]{\underline{#1}}

```

`preglossaryhook` This macro was introduced in glossaries version 4.19, so it may not be defined. If it hasn't been defined this feature is unavailable. A check is added for the entry's existence to prevent errors from occurring if the user removes an entry or changes the label, which can interrupt the build process.

```

3287 \ifdef\@gl@preglossaryhook
3288 {
3289   \renewcommand*{\@gl@preglossaryhook}{%
3290     \@glxtr@activate@initialtagging
3291     \let\@glxtr@org@postdescription\glspostdescription
3292     \renewcommand*{\glspostdescription}{%
3293       \ifglentryexists{\glscurrententrylabel}%
3294       {%
3295         \glxtrpostdescription
3296         \@glxtr@org@postdescription
3297       }{}%
3298     }%
3299   }%
3300 }
3301 {}

```

`postdescription` This command will only be used if `\@gl@preglossaryhook` is available *and* the glossary style uses `\glspostdescription` without modifying it. (`\nopostdesc` will suppress this.) The glossaries-extra-stylemods package will add the post description hook to all the predefined styles that don't include it.

```

3302 \newcommand*{\glxtrpostdescription}{%
3303   \csuse{glxtrpostdesc\glscategory{\glscurrententrylabel}}%
3304 }

postdescgeneral
3305 \newcommand*{\glxtrpostdescgeneral}{}

xtrpostdescterm
3306 \newcommand*{\glxtrpostdescterm}{}

postdescacronym
3307 \newcommand*{\glxtrpostdescacronym}{}

escabbreviation
3308 \newcommand*{\glxtrpostdescabbreviation}{}

glspostlinkhook  Redefine the post link hook used by commands like \gls to make it easier for categories
                  or attributes to modify this action. Since this hook occurs outside the existence check of
                  commands like \gls, this needs to be checked again here. Do nothing if the entry hasn't
                  been defined.
3309 \renewcommand*{\glspostlinkhook}{%
3310   \ifglseentryexists{\glslabel}{\glxtrpostlinkhook}{}%
3311 }

xtrpostlinkhook  The entry label should already be stored in \glslabel by \@gls@link.
3312 \newcommand*{\glxtrpostlinkhook}{%
3313   \glxtrdiscardperiod{\glslabel}%
3314   {\glxtrpostlinkendsentence}%
3315   {\glxtrpostlink}%
3316 }

\glxtrpostlink
3317 \newcommand*{\glxtrpostlink}{%
3318   \csuse{glxtrpostlink\glscategory{\glslabel}}%
3319 }

linkendsentence  Done by \glxtrpostlinkhook if a full stop is discarded.
3320 \newcommand*{\glxtrpostlinkendsentence}{%
3321   \ifcsdef{glxtrpostlink\glscategory{\glslabel}}
3322   {%
3323     \csuse{glxtrpostlink\glscategory{\glslabel}}%
    Put the full stop back.
3324     .\spacefactor\sfcode'\. \relax
3325   }%
3326   {%

```

Assume the full stop was discarded because the entry ends with a period, so adjust the space-factor.

```
3327 \spacefactor\sfcode'\. \relax
3328 }%
3329 }
```

addDescOnFirstUse Provide a command for appending the description in parentheses on first use, for the convenience of users wanting to add this to the post link hook.

```
3330 \newcommand*{\glxtrpostlinkAddDescOnFirstUse}{%
3331 \glxtrifwasfirstuse{\space(\glssaccessdesc{\glslabel})}{}}%
3332 }
```

addSymbolOnFirstUse Provide a command for appending the symbol (if defined) in parentheses on first use, for the convenience of users wanting to add this to the post link hook.

```
3333 \newcommand*{\glxtrpostlinkAddSymbolOnFirstUse}{%
3334 \glxtrifwasfirstuse
3335 {%
3336 \ifglshassymbol{\glslabel}{\space(\glssaccesssymbol{\glslabel})}{}}%
3337 }%
3338 {}%
3339 }
```

discardperiod Discard following period (if present) if the discardperiod attribute is true. If a period is discarded, do the second argument otherwise do the third argument. The entry label is in the first argument. Since this is designed for abbreviations that end with a period, check if the plural form was used (which typically won't end with a period).

```
3340 \newcommand*{\glxtrdiscardperiod}[3]{%
3341 \glxtrifwasfirstuse
3342 {%
3343 \glssifattribute{#1}{retainfirstuseperiod}{true}%
3344 {#3}%
3345 {%
3346 \glssifattribute{#1}{discardperiod}{true}%
3347 {%
3348 \glssifplural
3349 {%
3350 \glssifattribute{#1}{pluraldiscardperiod}{true}%
3351 {\glxtrifperiod{#2}{#3}}%
3352 {#3}%
3353 }%
3354 }%
3355 \glxtrifperiod{#2}{#3}%
3356 }%
3357 }%
3358 {#3}%
3359 }%
3360 }%
3361 {%
```

```

3362 \glsifattribute{#1}{discardperiod}{true}%
3363 {%
3364   \glsifplural
3365   {%
3366     \glsifattribute{#1}{pluraldiscardperiod}{true}%
3367     {\glsxtrifperiod{#2}{#3}}%
3368     {#3}%
3369   }%
3370   {%
3371     \glsxtrifperiod{#2}{#3}%
3372   }%
3373   }%
3374   {#3}%
3375 }%
3376 }

```

`\glsxtrifperiod` Make a convenient user command to check if the next character is a full stop (period). Works like `\@ifstar` but uses `\new@ifnextchar` rather than `\@ifnextchar`

```

3377 \newcommand*{\glsxtrifperiod}[1]{\new@ifnextchar.\@firstoftwo{#1}}

```

Sometimes it's useful to test if there's a punctuation character following the glossary entry.

`\glsxtr@punclist` List of characters identified as punctuation marks. (Be careful of babel shorthands!) This doesn't allow for punctuation marks made up from multiple characters (such as ' ').

```

3378 \newcommand*{\glsxtr@punclist}{.,:;?!}

```

`\punctuationmark` Add character to punctuation list.

```

3379 \newcommand*{\glsxtraddpunctuationmark}[1]{\appto\glsxtr@punclist{#1}}

```

`\punctuationmarks` Reset the punctuation list.

```

3380 \newcommand*{\glsxtrsetpunctuationmarks}[1]{\def\glsxtr@punclist{#1}}

```

`\glsxtrifpunc` `\glsxtrifnextpunc{<true part>}{<>false part>}`

Test if this is followed by a punctuation mark. (Adapted from `\new@ifnextchar`.)

```

3381 \newcommand*{\glsxtrifnextpunc}[2]{%
3382   \def\reserved@a{#1}%
3383   \def\reserved@b{#2}%
3384   \futurelet\@glspunc@token\glsxtr@ifnextpunc
3385 }

```

`\glsxtr@ifnextpunc`

```

3386 \newcommand*{\glsxtr@ifnextpunc}{%
3387   \glsxtr@ifpunctoken{\@glspunc@token}{\let\reserved@b\reserved@a}{}%
3388   \reserved@b
3389 }

```

```

xtr@ifpunctoken  Test if the token given in the first argument is in the punctuation list.
3390 \newcommand*{\glxtr@ifpunctoken}[1]{%
3391   \expandafter\@glxtr@ifpunctoken\expandafter#1\glxtr@punclist\@nnil
3392 }

```

```

xtr@ifpunctoken
3393 \def\@glxtr@ifpunctoken#1#2{%
3394   \let\reserved@d=#2%
3395   \ifx\reserved@d\@nnil
3396     \let\glxtr@next\@glxtr@notfoundinlist
3397   \else
3398     \ifx#1\reserved@d
3399       \let\glxtr@next\@glxtr@foundinlist
3400     \else
3401       \let\glxtr@next\@glxtr@ifpunctoken
3402     \fi
3403   \fi
3404   \glxtr@next#1%
3405 }

```

```

xtr@foundinlist
3406 \def\@glxtr@foundinlist#1\@nnil{\@firstoftwo}

```

```

@notfoundinlist
3407 \def\@glxtr@notfoundinlist#1{\@secondoftwo}

```

```

glxtrdopostpunc \glxtrdopostpunc{<code>}

```

If this is followed by a punctuation character, do *<code>* after the character otherwise do *<code>* before whatever comes next.

```

3408 \newcommand{\glxtrdopostpunc}[1]{%
3409   \glxtrifnextpunc{\@glxtr@swaptwo{#1}}{#1}%
3410 }

```

```

@glxtr@swaptwo
3411 \newcommand{\@glxtr@swaptwo}[2]{#2#1}

```

1.6 Abbreviations

The “acronym” code from glossaries is misnamed as it’s more often used for other forms of abbreviations. This code corrects this inconsistency, but rather than just having synonyms, provide commands for abbreviations that have a similar, but not identical, underlying mechanism to acronyms.

If there’s a style for the given category, apply it.

```

3412 \define@key{glsxtrabbrv}{category}{%
3413   \edef\glscategorylabel{#1}%
3414   \ifcsdef{@glsabbrv@current@#1}%
3415   {%

```

Warning should already have been issued.

```

3416   \let\@glsxtr@orgwarndep\GlsXtrWarnDeprecatedAbbrStyle
3417   \let\GlsXtrWarnDeprecatedAbbrStyle\@gobbletwo
3418   \glsxtr@applyabbrvstyle{\csname @glsabbrv@current@#1\endcsname}%
3419   \let\GlsXtrWarnDeprecatedAbbrStyle\@glsxtr@orgwarndep
3420 }%
3421 }%
3422 }

```

Save the short plural form. This may be needed before the entry is defined.

```

3423 \define@key{glsxtrabbrv}{shortplural}{%
3424   \def\@gls@shortpl{#1}%
3425 }

```

Similarly for the long plural form.

```

3426 \define@key{glsxtrabbrv}{longplural}{%
3427   \def\@gls@longpl{#1}%
3428 }

```

Token registers for the short plural and long plural, provided for use in the abbreviation style definitions.

\glsshortpltok

```

3429 \newtoks\glsshortpltok

```

\glslongpltok

```

3430 \newtoks\glslongpltok

```

glsxtr@insertdots Provided in case user wants to automatically insert dots between each letter of the abbreviation. This should be applied before defining the abbreviation to optimise the document build. (Otherwise, it would have to be done each time the short form is required, which is an unnecessary waste of time.) For this to work the short form must be expanded when passed to \newabbreviation. Note that explicitly using the short or shortplural keys will override this.

```

3431 \newcommand*{\@glsxtr@insertdots}[2]{%
3432   \def#1{%
3433     \@glsxtr@insert@dots#1#2\@nnil
3434 }

```

glsxtr@insert@dots

```

3435 \newcommand*{\@glsxtr@insert@dots}[2]{%
3436   \ifx\@nnil#2\relax
3437   \let\@glsxtr@insert@dots@next\@gobble
3438   \else
3439   \ifx\relax#2\relax

```



```

3440 \else
3441 \appto#1{#2.}%
3442 \fi
3443 \let\@glstr@insert@dots@next\@glstr@insert@dots
3444 \fi
3445 \@glstr@insert@dots@next#1%
3446 }

```

newabbreviation Define a new generic abbreviation.

```

3447 \newcommand*\newabbreviation[4][{}]{%
3448 \glskeylisttok{#1}%
3449 \glslabeltok{#2}%
3450 \glsshorttok{#3}%
3451 \glslongtok{#4}%

```

Get the category.

```

3452 \def\glscategorylabel{abbreviation}%
3453 \glstr@applyabbrvstyle{\@glstr@current@abbreviation}%
3454 \setkeys*{glstrabbrv}[shortplural,longplural]{#1}%

```

Set the default long plural

```

3455 \def\@gls@longpl{#4\glspluralsuffix}%

```

Has the insertdots attribute been set?

```

3456 \glsifcategoryattribute{\glscategorylabel}{insertdots}{true}%
3457 {%
3458 \@glstr@insertdots\@gls@short{#3}%
3459 \expandafter\glsshorttok\expandafter{\@gls@short\spacefactor1000 \relax}%
3460 \glsifcategoryattribute{\glscategorylabel}{aposplural}{true}%
3461 {%
3462 \expandafter\def\expandafter\@gls@shortpl\expandafter{\@gls@short
3463 '\abbrvpluralsuffix}%
3464 }%
3465 {%
3466 \glsifcategoryattribute{\glscategorylabel}{noshortplural}{true}%
3467 {%
3468 \let\@gls@shortpl\@gls@short
3469 }%
3470 {%
3471 \expandafter\def\expandafter\@gls@shortpl\expandafter{\@gls@short
3472 \abbrvpluralsuffix}%
3473 }%
3474 }%
3475 }%
3476 {%

```

insertdots not true.

```

3477 \glsifcategoryattribute{\glscategorylabel}{aposplural}{true}%
3478 {%
3479 \def\@gls@shortpl{#3'\abbrvpluralsuffix}%
3480 }%

```

```

3481   {%
3482     \glscategoryattribute{\glscategorylabel}{noshortplural}{true}%
3483   {%
3484     \def\@gls@shortpl{#3}%
3485   }%
3486   {%
3487     \def\@gls@shortpl{#3\abbrvpluralsuffix}%
3488   }%
3489   }%
3490 }%

```

Hook for further customisation if required:

```

3491 \glsxtrnewabbrvpresetkeyhook{#1}{#2}{#3}%

```

Get the short and long plurals provided by user in optional argument to override defaults, if necessary.

```

3492 \setkeys*{glxtrabbrv}[category]{#1}%

```

Set the plural token registers so the values can be accessed by the abbreviation styles.

```

3493 \expandafter\glsshortpltok\expandafter{\@gls@shortpl}%
3494 \expandafter\glslongpltok\expandafter{\@gls@longpl}%

```

Do any extra setup provided by hook:

```

3495 \newabbreviationhook

```

Define this entry:

```

3496 \protected@edef\@do@newglossaryentry{%
3497   \noexpand\newglossaryentry{\the\glslabeltok}%
3498   {%
3499     type=\glxtrabbrvtype,%
3500     category=abbreviation,%
3501     short={\the\glsshorttok},%
3502     shortplural={\the\glsshortpltok},%
3503     long={\the\glslongtok},%
3504     longplural={\the\glslongpltok},%
3505     name={\the\glsshorttok},%
3506     \CustomAbbreviationFields,%
3507     \the\glskeylisttok
3508   }%
3509 }%
3510 \@do@newglossaryentry
3511 \GlsXtrPostNewAbbreviation
3512 }

```

`evpresetkeyhook` Hook for extra stuff in `\newabbreviation`

```

3513 \newcommand*{\glsxtrnewabbrvpresetkeyhook}[3]{}

```

`NewAbbreviation` Hook used by abbreviation styles.

```

3514 \newcommand*{\GlsXtrPostNewAbbreviation}{}

```

`bbreviationhook` Hook for use with `\newabbreviation`.

```

3515 \newcommand*{\newabbreviationhook}{}

```

reviousFields

```
3516 \newcommand*{\CustomAbbreviationFields}{}
```

lsxtrfullformat Full format without case change.

```
3517 \newcommand*{\glxtrfullformat}[2]{%
3518   \glsfirstlongfont{\glsaccesslong{#1}}#2\glxtrfullsep{#1}%
3519   (\protect\glsfirstabbrvfont{\glsaccessshort{#1}})%
3520 }
```

lsxtrfullformat Full format with case change.

```
3521 \newcommand*{\Glsxtrfullformat}[2]{%
3522   \glsfirstlongfont{\Glsaccesslong{#1}}#2\glxtrfullsep{#1}%
3523   (\protect\glsfirstabbrvfont{\Glsaccessshort{#1}})%
3524 }
```

xtxtrfullplformat Plural full format without case change.

```
3525 \newcommand*{\glxtrfullplformat}[2]{%
3526   \glsfirstlongfont{\glsaccesslongpl{#1}}#2\glxtrfullsep{#1}%
3527   (\protect\glsfirstabbrvfont{\glsaccessshortpl{#1}})%
3528 }
```

xtxtrfullplformat Plural full format with case change.

```
3529 \newcommand*{\Glsxtrfullplformat}[2]{%
3530   \glsfirstlongfont{\Glsaccesslongpl{#1}}#2\glxtrfullsep{#1}%
3531   (\protect\glsfirstabbrvfont{\Glsaccessshortpl{#1}})%
3532 }
```

\glxtrfullsep Separator used by full format is a space by default. The argument is the entry's label.

```
3533 \newcommand*{\glxtrfullsep}[1]{\space}
```

In-line formats in case first use isn't compatible with \glentryfull (for example, first use suppresses the long form or uses a footnote).

inlinefullformat Full format without case change.

```
3534 \newcommand*{\glxtrininlinefullformat}{\glxtrfullformat}
```

inlinefullformat Full format with case change.

```
3535 \newcommand*{\Glsxtrininlinefullformat}{\Glsxtrfullformat}
```

xtxtrfullplformat Plural full format without case change.

```
3536 \newcommand*{\glxtrininlinefullplformat}{\glxtrfullplformat}
```

inefullplformat Plural full format with case change.

```
3537 \newcommand*{\Glsxtrininlinefullplformat}{\Glsxtrfullplformat}
```

Redefine \glentryfull etc to use the inline format. Since these commands as supposed to be expandable, they can only use the currently applied style. If there are mixed styles, you'll need to use the \glxtrfull set of commands instead.

`\glsentryfull`
3538 `\renewcommand*{\glsentryfull}[1]{\glsxtrinlinefullformat{#1}{}}`

`\Glsentryfull`
3539 `\renewcommand*{\Glsentryfull}[1]{\Glsxtrinlinefullformat{#1}{}}`

`\glsentryfullpl`
3540 `\renewcommand*{\glsentryfullpl}[1]{\glsxtrinlinefullplformat{#1}{}}`

`\Glsentryfullpl`
3541 `\renewcommand*{\Glsentryfullpl}[1]{\Glsxtrinlinefullplformat{#1}{}}`

`\glsfirstabbrvfont` Font changing command used for the abbreviation on first use or in the full format.
3542 `\newcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{#1}}`

`\glsabbrvdefaultfont` Font changing command used for the abbreviation on first use or in the full format.
3543 `\newcommand*{\glsfirstabbrvdefaultfont}[1]{\glsabbrvfont{#1}}`

`\glsabbrvfont` Font changing command used for the abbreviation on subsequent use.
3544 `\newcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{#1}}`

`\glsabbrvdefaultfont`
3545 `\newcommand*{\glsabbrvdefaultfont}[1]{#1}`

`\glslongfont` Font changing command used for the long form in commands like `\glsxtrlong`.
3546 `\newcommand*{\glslongfont}[1]{\glslongdefaultfont{#1}}`

`\glslongdefaultfont` Default font changing command used for the long form in commands like `\glsxtrlong`.
3547 `\newcommand*{\glslongdefaultfont}[1]{#1}`

`\glsfirstlongfont` Font changing command used for the long form on first use or in the full format.
3548 `\newcommand*{\glsfirstlongfont}[1]{\glslongfont{#1}}`

`\glsfirstlongdefaultfont`
3549 `\newcommand*{\glsfirstlongdefaultfont}[1]{\glslongdefaultfont{#1}}`

`\glsbrvpluralsuffix` Default plural suffix.
3550 `\newcommand*{\abbrvpluralsuffix}{\glspluralsuffix}`

`\glsxtrfull` Full form (no case-change).
3551 `\newrobustcmd*{\glsxtrfull}{\@gls@hyp@opt\ns@glsxtrfull}`
3552 `\newcommand*{\ns@glsxtrfull}[2][{}]{%`
3553 `\new@ifnextchar[{\@glsxtr@full{#1}{#2}}{%`
3554 `{\@glsxtr@full{#1}{#2}}[{}]{%`
3555 `}`

`\@glxtr@full` Low-level macro:

```
3556 \def\@glxtr@full#1#2[#3]{%
3557   \glstoifexists{#2}%
3558   {%
3559     \glsetabbrvfmt{\glscategory{#2}}%
3560     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
3561     \let\gl@ifplural\@secondoftwo
3562     \let\glscapscase\@firstofthree
3563     \let\glinsert\@empty
3564     \def\glscustomtext{\glxtrinlinefullformat{#2}{#3}}%
```

What should `\glxtrifwasfirstuse` be set to here? Where the inline and display full forms are the same, this is essentially emulating first use, to it make sense for the postlink hook to pretend it was a first use instance. It makes less sense if the inline and display forms are different. Provide a hook to make it easier to reconfigure.

```
3565   \glxtrsetupfulldefs
3566   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3567   }%
3568   \glspostlinkhook
3569 }
```

`trsetupfulldefs`

```
3570 \newcommand*{\glxtrsetupfulldefs}{%
3571   \let\glxtrifwasfirstuse\@firstoftwo
3572 }
```

`\Glsxtrfull` Full form (first letter uppercase).

```
3573 \newrobustcmd*{\Glsxtrfull}{\@gl@hyp@opt\@ns@Glsxtrfull}
3574 \newcommand*\ns@Glsxtrfull[2][{}]{%
3575   \new@ifnextchar[\@Glsxtr@full{#1}{#2}}%
3576   {\@Glsxtr@full{#1}{#2}[]}%
3577 }
```

`\@Glsxtr@full` Low-level macro:

```
3578 \def\@Glsxtr@full#1#2[#3]{%
3579   \glstoifexists{#2}%
3580   {%
3581     \glsetabbrvfmt{\glscategory{#2}}%
3582     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
3583     \let\gl@ifplural\@secondoftwo
3584     \let\glscapscase\@secondofthree
3585     \let\glinsert\@empty
3586     \def\glscustomtext{\Glsxtrinlinefullformat{#2}{#3}}%
3587     \glxtrsetupfulldefs
3588     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3589   }%
3590   \glspostlinkhook
3591 }
```

`\GLSxtrfull` Full form (all uppercase).

```
3592 \newrobustcmd*{\GLSxtrfull}{\@gls@hyp@opt\ns@GLSxtrfull}
3593 \newcommand*\ns@GLSxtrfull[2][\%
3594   \new@ifnextchar[\@GLSxtr@full{#1}{#2}]{%
3595     {\@GLSxtr@full{#1}{#2}[]}%
3596 }
```

`\@GLSxtr@full` Low-level macro:

```
3597 \def\@GLSxtr@full#1#2[#3]{%
3598   \glsdoifexists{#2}%
3599   {%
3600     \glssetabbrvfmt{\glscategory{#2}}%
3601     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
3602     \let\glsifplural\@secondoftwo
3603     \let\glsapscase\@thirdofthree
3604     \let\glsinsert\@empty
3605     \def\glscustomtext{\mfirstucMakeUppercase{\glsxtrinlinefullformat{#2}{#3}}}%
3606     \glsxtrsetupfulldefs
3607     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3608   }%
3609   \glspostlinkhook
3610 }
```

`\glsxtrfullpl` Plural full form (no case-change).

```
3611 \newrobustcmd*{\glsxtrfullpl}{\@gls@hyp@opt\ns@glsxtrfullpl}
3612 \newcommand*\ns@glsxtrfullpl[2][\%
3613   \new@ifnextchar[\@glsxtr@fullpl{#1}{#2}]{%
3614     {\@glsxtr@fullpl{#1}{#2}[]}%
3615 }
```

`\@glsxtr@fullpl` Low-level macro:

```
3616 \def\@glsxtr@fullpl#1#2[#3]{%
3617   \glsdoifexists{#2}%
3618   {%
3619     \glssetabbrvfmt{\glscategory{#2}}%
3620     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
3621     \let\glsifplural\@firstoftwo
3622     \let\glsapscase\@firstofthree
3623     \let\glsinsert\@empty
3624     \def\glscustomtext{\glsxtrinlinefullplformat{#2}{#3}}%
3625     \glsxtrsetupfulldefs
3626     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3627   }%
3628   \glspostlinkhook
3629 }
```

`\Glsxtrfullpl` Plural full form (first letter uppercase).

```
3630 \newrobustcmd*{\Glsxtrfullpl}{\@gls@hyp@opt\ns@Glsxtrfullpl}
3631 \newcommand*\ns@Glsxtrfullpl[2][\%
```

```

3632 \new@ifnextchar[{\@Glsxtr@fullpl{#1}{#2}}%
3633           {\@Glsxtr@fullpl{#1}{#2}[]}%
3634 }

```

\@Glsxtr@fullpl Low-level macro:

```

3635 \def\@Glsxtr@fullpl#1#2[#3]{%
3636   \glsdoifexists{#2}%
3637   {%
3638     \glssetabbrvfmt{\glscategory{#2}}%
3639     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
3640     \let\glsifplural\@firstoftwo
3641     \let\glscapscase\@secondofthree
3642     \let\glsinsert\@empty
3643     \def\glscustomtext{\Glsxtrinlinefullplformat{#2}{#3}}%
3644     \glsxtrsetupfulldefs
3645     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3646   }%
3647   \glspostlinkhook
3648 }

```

\Glsxtrfullpl Plural full form (all upper case).

```

3649 \newrobustcmd*{\Glsxtrfullpl}{\@gl@hyp@opt\@ns@Glsxtrfullpl}
3650 \newcommand*\ns@Glsxtrfullpl[2][]{%
3651   \new@ifnextchar[{\@Glsxtr@fullpl{#1}{#2}}%
3652           {\@Glsxtr@fullpl{#1}{#2}[]}%
3653 }

```

\@Glsxtr@fullpl Low-level macro:

```

3654 \def\@Glsxtr@fullpl#1#2[#3]{%
3655   \glsdoifexists{#2}%
3656   {%
3657     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
3658     \let\glsifplural\@firstoftwo
3659     \let\glscapscase\@thirdofthree
3660     \let\glsinsert\@empty
3661     \def\glscustomtext{%
3662       \mfirstucMakeUppercase{\glsxtrinlinefullplformat{#2}{#3}}}%
3663     \glsxtrsetupfulldefs
3664     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3665   }%
3666   \glspostlinkhook
3667 }

```

The short and long forms work in a similar way to acronyms.

\glsxtrshort

```

3668 \newrobustcmd*{\glsxtrshort}{\@gl@hyp@opt\@ns@glsxtrshort}

```

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

```

3669 \newcommand*{\ns@glstrshort}[2] [] {%
3670   \new@ifnextchar[{\@glstrshort{#1}{#2}}{\@glstrshort{#1}{#2} []}%
3671 }

```

Read in the final optional argument:

```

3672 \def\@glstrshort#1#2[#3] {%
3673   \glstoifexists{#2}%
3674   {%

```

Need to make sure \glabbrvfont is set correctly.

```

3675     \glsetabbrvfmt{\glscategory{#2}}%
3676     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
3677     \let\glstrifwasfirstuse\@secondoftwo
3678     \let\gl@ifplural\@secondoftwo
3679     \let\glscapscase\@firstofthree
3680     \let\glinsert\@empty
3681     \def\glscustomtext{%
3682       \glabbrvfont{\glaccessshort{#2}\ifglstrinsertinside#3\fi}%
3683       \ifglstrinsertinside\else#3\fi
3684     }%
3685     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3686   }%
3687   \glspostlinkhook
3688 }

```

\Glsxtrshort

```

3689 \newrobustcmd*{\Glsxtrshort}{\@gl@hyp@opt\ns@Glsxtrshort}

```

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

```

3690 \newcommand*{\ns@Glsxtrshort}[2] [] {%
3691   \new@ifnextchar[{\@Glsxtrshort{#1}{#2}}{\@Glsxtrshort{#1}{#2} []}%
3692 }

```

Read in the final optional argument:

```

3693 \def\@Glsxtrshort#1#2[#3] {%
3694   \glstoifexists{#2}%
3695   {%
3696     \glsetabbrvfmt{\glscategory{#2}}%
3697     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
3698     \let\glstrifwasfirstuse\@secondoftwo
3699     \let\gl@ifplural\@secondoftwo
3700     \let\glscapscase\@secondofthree
3701     \let\glinsert\@empty
3702     \def\glscustomtext{%
3703       \Glsabbrvfont{\Glsaccessshort{#2}\ifglstrinsertinside#3\fi}%
3704       \ifglstrinsertinside\else#3\fi
3705     }%
3706     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3707   }%
3708   \glspostlinkhook
3709 }

```


\GLSxtrshort

```
3710 \newrobustcmd*{\GLSxtrshort}{\@gls@hyp@opt\@ns@GLSxtrshort}
```

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

```
3711 \newcommand*{\ns@GLSxtrshort}[2] [] {%
3712   \new@ifnextchar[{\@GLSxtrshort{#1}{#2}}{\@GLSxtrshort{#1}{#2} []}]%
3713 }
```

Read in the final optional argument:

```
3714 \def\@GLSxtrshort#1#2[#3] {%
3715   \glsdoifexists{#2}%
3716   {%
3717     \glssetabbrvfmt{\glscategory{#2}}%
3718     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
3719     \let\glsxtrifwasfirstuse\@secondoftwo
3720     \let\glsifplural\@secondoftwo
3721     \let\glsapscase\@thirdofthree
3722     \let\glsinsert\@empty
3723     \def\glscustomtext{%
3724       \mfirstucMakeUppercase
3725       {\glsabbrvfont{\glsaccessshort{#2}}\ifglsxtrininsertinside#3\fi}%
3726       \ifglsxtrininsertinside\else#3\fi
3727     }%
3728   }%
3729   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3730   }%
3731   \glspostlinkhook
3732 }
```

\glsxtrlong

```
3733 \newrobustcmd*{\glsxtrlong}{\@gls@hyp@opt\@ns@glsxtrlong}
```

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

```
3734 \newcommand*{\ns@glsxtrlong}[2] [] {%
3735   \new@ifnextchar[{\@glsxtrlong{#1}{#2}}{\@glsxtrlong{#1}{#2} []}]%
3736 }
```

Read in the final optional argument:

```
3737 \def\@glsxtrlong#1#2[#3] {%
3738   \glsdoifexists{#2}%
3739   {%
3740     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
3741     \let\glsxtrifwasfirstuse\@secondoftwo
3742     \let\glsifplural\@secondoftwo
3743     \let\glsapscase\@firstofthree
3744     \let\glsinsert\@empty
3745     \def\glscustomtext{%
3746       \glsfont{\glsaccesslong{#2}}\ifglsxtrininsertinside#3\fi}%
3747       \ifglsxtrininsertinside\else#3\fi
3748     }%
3749     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%

```

```

3750 }%
3751 \glspostlinkhook
3752 }

```

\Glsxtrlong

```

3753 \newrobustcmd*{\Glsxtrlong}{\@gls@hyp@opt\ns@Glsxtrlong}
    Define the un-starred form. Need to determine if there is a final optional argument
3754 \newcommand*{\ns@Glsxtrlong}[2] [] {%
3755   \new@ifnextchar[{\@Glsxtrlong{#1}{#2}}{\@Glsxtrlong{#1}{#2} []}%
3756 }

    Read in the final optional argument:
3757 \def\@Glsxtrlong#1#2[#3]{%
3758   \glsdoifexists{#2}%
3759   {%
3760     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
3761     \let\glstrifwasfirstuse\@secondoftwo
3762     \let\glsifplural\@secondoftwo
3763     \let\glscapscase\@secondofthree
3764     \let\glsinsert\@empty
3765     \def\glscustomtext{%
3766       \glslongfont{\Glsaccesslong{#2}\ifglstrinsertinside#3\fi}%
3767       \ifglstrinsertinside\else#3\fi
3768     }%
3769     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3770   }%
3771   \glspostlinkhook
3772 }

```

\GLSxtrlong

```

3773 \newrobustcmd*{\GLSxtrlong}{\@gls@hyp@opt\ns@GLSxtrlong}
    Define the un-starred form. Need to determine if there is a final optional argument
3774 \newcommand*{\ns@GLSxtrlong}[2] [] {%
3775   \new@ifnextchar[{\@GLSxtrlong{#1}{#2}}{\@GLSxtrlong{#1}{#2} []}%
3776 }

    Read in the final optional argument:
3777 \def\@GLSxtrlong#1#2[#3]{%
3778   \glsdoifexists{#2}%
3779   {%
3780     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
3781     \let\glstrifwasfirstuse\@secondoftwo
3782     \let\glsifplural\@secondoftwo
3783     \let\glscapscase\@thirdofthree
3784     \let\glsinsert\@empty
3785     \def\glscustomtext{%
3786       \mfirstucMakeUppercase
3787       {\glslongfont{\Glsaccesslong{#2}\ifglstrinsertinside#3\fi}%
3788       \ifglstrinsertinside\else#3\fi

```

```

3789     }%
3790     }%
3791     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3792     }%
3793     \glspostlinkhook
3794 }

```

Plural short forms:

\glsxtrshortpl

```

3795 \newrobustcmd*{\glsxtrshortpl}{\@gls@hyp@opt\@ns@glsxtrshortpl}

```

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

```

3796 \newcommand*{\ns@glsxtrshortpl}[2] [] {%
3797   \new@ifnextchar[{\@glsxtrshortpl{#1}{#2}}{\@glsxtrshortpl{#1}{#2} []}%
3798 }

```

Read in the final optional argument:

```

3799 \def\@glsxtrshortpl#1#2[#3] {%
3800   \glsdoifexists{#2}%
3801   {%
3802     \glssetabbrvfmt{\glscategory{#2}}%
3803     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
3804     \let\glsxtrifwasfirstuse\@secondoftwo
3805     \let\glsifplural\@firstoftwo
3806     \let\glscapscase\@firstofthree
3807     \let\glsinsert\@empty
3808     \def\glscustomtext{%
3809       \glsabbrvfont{\glsaccessshortpl{#2}\ifglsxtrininsertinside#3\fi}%
3810       \ifglsxtrininsertinside\else#3\fi
3811     }%
3812     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3813   }%
3814   \glspostlinkhook
3815 }

```

\Glsxtrshortpl

```

3816 \newrobustcmd*{\Glsxtrshortpl}{\@gls@hyp@opt\@ns@Glsxtrshortpl}

```

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

```

3817 \newcommand*{\ns@Glsxtrshortpl}[2] [] {%
3818   \new@ifnextchar[{\@Glsxtrshortpl{#1}{#2}}{\@Glsxtrshortpl{#1}{#2} []}%
3819 }

```

Read in the final optional argument:

```

3820 \def\@Glsxtrshortpl#1#2[#3] {%
3821   \glsdoifexists{#2}%
3822   {%
3823     \glssetabbrvfmt{\glscategory{#2}}%
3824     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
3825     \let\glsxtrifwasfirstuse\@secondoftwo

```

```

3826 \let\glsifplural\@firstoftwo
3827 \let\glschapscase\@secondofthree
3828 \let\glsinsert\@empty
3829 \def\glscustomtext{%
3830 \glsabbrvfont{\Glsaccessshortpl{#2}\ifglstrinsertinside#3\fi}%
3831 \ifglstrinsertinside\else#3\fi
3832 }%
3833 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3834 }%
3835 \glspostlinkhook
3836 }

```

\GLSxtrshortpl

```

3837 \newrobustcmd*{\GLSxtrshortpl}{\@gls@hyp@opt\ns@GLSxtrshortpl}
    Define the un-starred form. Need to determine if there is a final optional argument
3838 \newcommand*{\ns@GLSxtrshortpl}[2] [] {%
3839 \new@ifnextchar[{\@GLSxtrshortpl{#1}{#2}}{\@GLSxtrshortpl{#1}{#2} []}%
3840 }

    Read in the final optional argument:
3841 \def\@GLSxtrshortpl#1#2[#3]{%
3842 \glsdoifexists{#2}%
3843 {%
3844 \glssetabbrvfmt{\glscategory{#2}}%
3845 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
3846 \let\glstrifwasfirstuse\@secondoftwo
3847 \let\glsifplural\@firstoftwo
3848 \let\glschapscase\@thirdofthree
3849 \let\glsinsert\@empty
3850 \def\glscustomtext{%
3851 \mfirstucMakeUppercase
3852 {\glsabbrvfont{\Glsaccessshortpl{#2}\ifglstrinsertinside#3\fi}%
3853 \ifglstrinsertinside\else#3\fi
3854 }%
3855 }%
3856 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3857 }%
3858 \glspostlinkhook
3859 }

```

Plural long forms:

\glsxtrlongpl

```

3860 \newrobustcmd*{\glsxtrlongpl}{\@gls@hyp@opt\ns@glsxtrlongpl}
    Define the un-starred form. Need to determine if there is a final optional argument
3861 \newcommand*{\ns@glsxtrlongpl}[2] [] {%
3862 \new@ifnextchar[{\@glsxtrlongpl{#1}{#2}}{\@glsxtrlongpl{#1}{#2} []}%
3863 }

```

Read in the final optional argument:

```

3864 \def\@glxstrlongpl#1#2[#3]{%
3865   \glstoifexists{#2}%
3866   {%
3867     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
3868     \let\glxstrifwasfirstuse\@secondoftwo
3869     \let\gl@ifplural\@firstoftwo
3870     \let\glscapscase\@firstofthree
3871     \let\glinsert\@empty
3872     \def\glscustomtext{%
3873       \glslongfont{\glaccesslongpl{#2}\ifglxtrininsertinside#3\fi}%
3874       \ifglxtrininsertinside\else#3\fi
3875     }%
3876     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3877   }%
3878   \glspostlinkhook
3879 }

```

\Glsxtrlongpl

```

3880 \newrobustcmd*{\Glsxtrlongpl}{\@gl@hyp@opt\ns@Glsxtrlongpl}
    Define the un-starred form. Need to determine if there is a final optional argument
3881 \newcommand*{\ns@Glsxtrlongpl}[2] [] {%
3882   \new@ifnextchar[{\@Glsxtrlongpl{#1}{#2}}{\@Glsxtrlongpl{#1}{#2} []}%
3883 }

```

Read in the final optional argument:

```

3884 \def\@Glsxtrlongpl#1#2[#3]{%
3885   \glstoifexists{#2}%
3886   {%
3887     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
3888     \let\glxstrifwasfirstuse\@secondoftwo
3889     \let\gl@ifplural\@firstoftwo
3890     \let\glscapscase\@secondofthree
3891     \let\glinsert\@empty
3892     \def\glscustomtext{%
3893       \glslongfont{\Glsaccesslongpl{#2}\ifglxtrininsertinside#3\fi}%
3894       \ifglxtrininsertinside\else#3\fi
3895     }%
3896     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3897   }%
3898   \glspostlinkhook
3899 }

```

\GLSxtrlongpl

```

3900 \newrobustcmd*{\GLSxtrlongpl}{\@gl@hyp@opt\ns@GLSxtrlongpl}
    Define the un-starred form. Need to determine if there is a final optional argument
3901 \newcommand*{\ns@GLSxtrlongpl}[2] [] {%
3902   \new@ifnextchar[{\@GLSxtrlongpl{#1}{#2}}{\@GLSxtrlongpl{#1}{#2} []}%
3903 }

```

Read in the final optional argument:

```
3904 \def\@GLSxtrlongpl#1#2[#3]{%
3905   \glsdoifexists{#2}%
3906   {%
3907     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
3908     \let\glxtrifwasfirstuse\@secondoftwo
3909     \let\glsifplural\@firstoftwo
3910     \let\glscapscase\@thirdofthree
3911     \let\glsinsert\@empty
3912     \def\glscustomtext{%
3913       \mfirstucMakeUppercase
3914       {\glslongfont{\glsaccesslongpl{#2}\ifglxtrininsertinside#3\fi}%
3915       \ifglxtrininsertinside\else#3\fi
3916     }%
3917   }%
3918   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3919 }%
3920 \glspostlinkhook
3921 }
```

`\glssetabbrvfmt` Set the current format for the given category (or the abbreviation category if unset).

```
3922 \newcommand*{\glssetabbrvfmt}[1]{%
3923   \ifcsdef{@glsabbrv@current@#1}%
3924   {\glxtr@applyabbrvfmt{\csname @glsabbrv@current@#1\endcsname}}%
3925   {\glxtr@applyabbrvfmt{\@glsabbrv@current@abbreviation}}%
3926 }
```

`sxtrgenabbrvfmt` Similar to `\glsgenacfmt`, but for abbreviations.

```
3927 \newcommand*{\glsxtrgenabbrvfmt}{%
3928   \ifdefempty\glscustomtext
3929   {%
3930     \ifgl@sused\glslabel
3931     {%
```

Subsequent use:

```
3932     \glsifplural
3933     {%
```

Subsequent plural form:

```
3934     \glscapscase
3935     {%
```

Subsequent plural form, don't adjust case:

```
3936     \glsabbrvfont{\glsaccessshortpl{\glslabel}}\glsinsert
3937     }%
3938     {%
```

Subsequent plural form, make first letter upper case:

```
3939     \glsabbrvfont{\Glsaccessshortpl{\glslabel}}\glsinsert
3940     }%
3941     {%
```

Subsequent plural form, all caps:

```
3942      \mfirstucMakeUppercase
3943      {\glsabbrvfont{\glsaccessshortpl{\glslabel}}\glsinsert}%
3944      }%
3945      }%
3946      {%
```

Subsequent singular form

```
3947      \glscapscase
3948      {%
```

Subsequent singular form, don't adjust case:

```
3949      \glsabbrvfont{\glsaccessshort{\glslabel}}\glsinsert
3950      }%
3951      {%
```

Subsequent singular form, make first letter upper case:

```
3952      \glsabbrvfont{\Glsaccessshort{\glslabel}}\glsinsert
3953      }%
3954      {%
```

Subsequent singular form, all caps:

```
3955      \mfirstucMakeUppercase
3956      {\glsabbrvfont{\glsaccessshort{\glslabel}}\glsinsert}%
3957      }%
3958      }%
3959      }%
3960      {%
```

First use:

```
3961      \glsifplural
3962      {%
```

First use plural form:

```
3963      \glscapscase
3964      {%
```

First use plural form, don't adjust case:

```
3965      \glstrfullplformat{\glslabel}{\glsinsert}%
3966      }%
3967      {%
```

First use plural form, make first letter upper case:

```
3968      \Glsxtrfullplformat{\glslabel}{\glsinsert}%
3969      }%
3970      {%
```

First use plural form, all caps:

```
3971      \mfirstucMakeUppercase
3972      {\glstrfullplformat{\glslabel}{\glsinsert}}%
3973      }%
3974      }%
3975      {%
```

First use singular form

```
3976      \glscapscase
3977      {%
```

First use singular form, don't adjust case:

```
3978      \glstrfullformat{\glslabel}{\glsinsert}%
3979      }%
3980      {%
```

First use singular form, make first letter upper case:

```
3981      \Glsxtrfullformat{\glslabel}{\glsinsert}%
3982      }%
3983      {%
```

First use singular form, all caps:

```
3984      \mfirstucMakeUppercase
3985      {\glstrfullformat{\glslabel}{\glsinsert}}%
3986      }%
3987      }%
3988      }%
3989      }%
3990      {%
```

User supplied text.

```
3991      \glscustomtext
3992      }%
3993 }
```

1.6.1 Abbreviation Styles Setup

abbreviationstyle

```
3994 \newcommand*{\setabbreviationstyle}[2][abbreviation]{%
3995   \ifcsundef{@glsabbrv@dispstyle@setup@#2}
3996   {%
3997     \PackageError{glossaries-extra}{Undefined abbreviation style ‘#2’}{}%
3998   }%
3999   {%
```

Have abbreviations already been defined for this category?

```
4000     \ifcsstring{@glsabbrv@current@#1}{#2}%
4001     {%
```

Style already set.

```
4002     }%
4003     {%
4004       \def\@glsxtr@dostylewarn{%
4005         \glsforeachincategory{#1}{\@gls@type}{\@gls@label}%
4006         {%
4007           \def\@glsxtr@dostylewarn{\GlossariesWarning{Abbreviation
4008             style has been switched \MessageBreak
4009             for category ‘#1’, \MessageBreak
```



```

4010         but there have already been entries \MessageBreak
4011         defined for this category. Unwanted \MessageBreak
4012         side-effects may result}}}%
4013     \@endfortrue
4014 }%
4015 \@glsxtr@dostylewarn

Set up the style for the given category.
4016     \csdef{@glsabbrv@current@#1}{#2}%
4017     \glsxtr@applyabbrvstyle{#2}%
4018 }%
4019 }%
4020 }

```

`\applyabbrvstyle` Apply the abbreviation style without existence check.

```

4021 \newcommand*{\glsxtr@applyabbrvstyle}[1]{%
4022   \csuse{@glsabbrv@dispstyle@setup@#1}%
4023   \csuse{@glsabbrv@dispstyle@fmts@#1}%
4024 }

```

`\r@applyabbrvfmt` Only apply the style formats.

```

4025 \newcommand*{\glsxtr@applyabbrvfmt}[1]{%
4026   \csuse{@glsabbrv@dispstyle@fmts@#1}%
4027 }

```

`\abbreviationstyle` This is different from `\newacronymstyle`. The first argument is the label, the second argument sets the information required when defining the new abbreviation and the third argument sets the commands used to display the full format.

```

4028 \newcommand*{\newabbreviationstyle}[3]{%
4029   \ifcsdef{@glsabbrv@dispstyle@setup@#1}
4030   {%
4031     \PackageError{glossaries-extra}{Abbreviation style ‘#1’ already
4032     defined}}}%
4033   }%
4034   {%
4035     \csdef{@glsabbrv@dispstyle@setup@#1}{%

```

Initialise hook to do nothing. The style may change this.

```

4036     \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
4037     #2}%
4038     \csdef{@glsabbrv@dispstyle@fmts@#1}{%

```

Assume in-line form is the same as first use. The style may change this.

```

4039     \renewcommand*{\glsxtrinlinefullformat}{\glsxtrfullformat}%
4040     \renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}%
4041     \renewcommand*{\glsxtrinlinefullplformat}{\glsxtrfullplformat}%
4042     \renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}%
4043     #3}%
4044   }%
4045 }

```

abbreviationstyle

```

4046 \newcommand*{\renewabbreviationstyle}[3]{%
4047   \ifcsundef{@glsabbrv@dispstyle@setup@#1}%
4048   {%
4049     \PackageError{glossaries-extra}{Abbreviation style ‘#1’ not defined}{}%
4050   }%
4051   {%
4052     \csdef{@glsabbrv@dispstyle@setup@#1}{%
      Initialise hook to do nothing. The style may change this.
4053       \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
4054       #2}%
4055     \csdef{@glsabbrv@dispstyle@fmts@#1}{%
      Assume in-line form is the same as first use. The style may change this.
4056       \renewcommand*{\glsxtrinlinefullformat}{\glsxtrfullformat}%
4057       \renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}%
4058       \renewcommand*{\glsxtrinlinefullplformat}{\glsxtrfullplformat}%
4059       \renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}%
4060       #3}%
4061     }%
4062 }

```

abbreviationstyle Define a synonym for an abbreviation style. The first argument is the new name. The second argument is the original style’s name.

```

4063 \newcommand*{\letabbreviationstyle}[2]{%
4064   \csletcs{@glsabbrv@dispstyle@setup@#1}{@glsabbrv@dispstyle@setup@#2}%
4065   \csletcs{@glsabbrv@dispstyle@fmts@#1}{@glsabbrv@dispstyle@fmts@#2}%
4066 }

```

deprecated@abbrstyle

```
\glsxtr@deprecated@abbrstyle{<old-name>}{<new-name>}
```

Define a synonym for a deprecated abbreviation style.

```

4067 \newcommand*{\@glsxtr@deprecated@abbrstyle}[2]{%
4068   \csdef{@glsabbrv@dispstyle@setup@#1}{%
4069     \GlsXtrWarnDeprecatedAbbrStyle{#1}{#2}%
4070   \csuse{@glsabbrv@dispstyle@setup@#2}%
4071   }%
4072   \csletcs{@glsabbrv@dispstyle@fmts@#1}{@glsabbrv@dispstyle@fmts@#2}%
4073 }

```

deprecatedAbbrStyle Generate warning for deprecated style use.

```

4074 \newcommand*{\GlsXtrWarnDeprecatedAbbrStyle}[2]{%
4075   \GlossariesExtraWarning{Deprecated abbreviation style name ‘#1’,
4076   use ‘#2’ instead}%
4077 }

```

eAbbrStyleSetup

```
4078 \newcommand*{\GlsXtrUseAbbrStyleSetup}[1]{%
4079   \ifcsundef{@glsabbrv@dispstyle@setup@#1}%
4080   {%
4081     \PackageError{glossaries-extra}%
4082     {Unknown abbreviation style definitions ‘#1’}{}%
4083   }%
4084   {%
4085     \csname @glsabbrv@dispstyle@setup@#1\endcsname
4086   }%
4087 }
```

seAbbrStyleFmts

```
4088 \newcommand*{\GlsXtrUseAbbrStyleFmts}[1]{%
4089   \ifcsundef{@glsabbrv@dispstyle@fmts@#1}%
4090   {%
4091     \PackageError{glossaries-extra}%
4092     {Unknown abbreviation style formats ‘#1’}{}%
4093   }%
4094   {%
4095     \csname @glsabbrv@dispstyle@fmts@#1\endcsname
4096   }%
4097 }
```

1.6.2 Predefined Styles (Default Font)

Define some common styles. These will set the first, firstplural, text and plural keys, even if the regular attribute isn't set to "true". If this attribute is set, commands like `\gls` will use them as per a regular entry, otherwise those keys will be ignored unless explicitly invoked by the user with commands like `\glsfirst`. In order for the first letter uppercase versions to work correctly, `\glsxtrfullformat` needs to be expanded when those keys are set. The final optional argument of `\glsfirst` will behave differently to the final optional argument of `\gls` with some styles.

xtrinsertinside Switch to determine if the insert text should be inside or outside the font changing command. The default is outside.

```
4098 \newif\ifglsxtrinsertinside
4099 \glsxtrinsertinsidefalse
```

long-short

```
4100 \newabbreviationstyle{long-short}%
4101 {%
4102   \renewcommand*{\CustomAbbreviationFields}{%
4103     name={\protect\glsabbrvfont{\the\glsshorttok}},
4104     sort={\the\glsshorttok},
4105     first={\protect\glsfirstlongfont{\the\glslongtok}}%
4106     \protect\glsxtrfullsep{\the\glslabeltok}}%
4107     (\protect\glsfirstabbrvfont{\the\glsshorttok}}),%
```

```

4108   firstplural={\protect\glsfirstlongfont{\the\glslongpltok}}%
4109   \protect\glsxtrfullsep{\the\glslabeltok}%
4110   (\protect\glsfirstabbrvfont{\the\glsshortpltok}}),%
4111   plural={\protect\glsabbvfont{\the\glsshortpltok}},%
4112   description={\the\glslongtok}}%

```

Unset the regular attribute if it has been set.

```

4113   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
4114     \glshasattribute{\the\glslabeltok}{regular}%
4115     {%
4116       \glssetAttribute{\the\glslabeltok}{regular}{false}%
4117     }%
4118   }%
4119 }%
4120}%
4121{%

```

In case the user wants to mix and match font styles, these are redefined here.

```

4122   \renewcommand*{\abbrvpluralsuffix}{\glspluralsuffix}%
4123   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
4124   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
4125   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
4126   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

4127   \renewcommand*{\glsxtrfullformat}[2]{%
4128     \glsfirstlongfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
4129     \ifglxtrinsertinside\else##2\fi
4130     \glsxtrfullsep{##1}%
4131     (\glsfirstabbrvfont{\glsaccessshort{##1}})%
4132   }%
4133   \renewcommand*{\glsxtrfullplformat}[2]{%
4134     \glsfirstlongfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
4135     \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
4136     (\glsfirstabbrvfont{\glsaccessshortpl{##1}})%
4137   }%
4138   \renewcommand*{\Glsxtrfullformat}[2]{%
4139     \glsfirstlongfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
4140     \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
4141     (\glsfirstabbrvfont{\glsaccessshort{##1}})%
4142   }%
4143   \renewcommand*{\Glsxtrfullplformat}[2]{%
4144     \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
4145     \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
4146     (\glsfirstabbrvfont{\glsaccessshortpl{##1}})%
4147   }%
4148 }

```

Set this as the default style for general abbreviations:

```

4149 \setabbreviationstyle{long-short}

```

ngshortdescsort

```
4150 \newcommand*{\glxtrlongshortdescsort}{\the\glslongtok\space(\the\glsshorttok)}
```

long-short-desc User supplies description. The long form is included in the name.

```
4151 \newabbreviationstyle{long-short-desc}%
4152 {%
4153   \renewcommand*{\CustomAbbreviationFields}{%
4154     name={\protect\glxtrfullformat{\the\glslabeltok}{}},
4155     sort={\glxtrlongshortdescsort},%
4156     first={\protect\glsfirstlongfont{\the\glslongtok}%
4157       \protect\glxtrfullsep{\the\glslabeltok}%
4158       (\protect\glsfirstabbrvfont{\the\glsshorttok})},%
4159     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}%
4160       \protect\glxtrfullsep{\the\glslabeltok}%
4161       (\protect\glsfirstabbrvfont{\the\glsshortpltok})},%
4162     plural={\protect\glsabbvfont{\the\glsshortpltok}}}%
4163   }%
4164   \glshasattribute{\the\glslabeltok}{regular}%
4165   {%
4166     \glissetattribute{\the\glslabeltok}{regular}{false}%
4167   }%
4168 }%
4169 }%
4170 }%
4171 {%
4172   \GlsXtrUseAbbrStyleFmts{long-short}%
4173 }
```

Unset the regular attribute if it has been set.

```
4163 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
4164   \glshasattribute{\the\glslabeltok}{regular}%
4165   {%
4166     \glissetattribute{\the\glslabeltok}{regular}{false}%
4167   }%
4168 }%
4169 }%
4170 }%
4171 {%
4172   \GlsXtrUseAbbrStyleFmts{long-short}%
4173 }
```

short-long Short form followed by long form in parenthesis on first use.

```
4174 \newabbreviationstyle{short-long}%
4175 {%
4176   \renewcommand*{\CustomAbbreviationFields}{%
4177     name={\protect\glsabbvfont{\the\glsshorttok}},
4178     sort={\the\glsshorttok},
4179     description={\the\glslongtok},%
4180     first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
4181       \protect\glxtrfullsep{\the\glslabeltok}%
4182       (\protect\glsfirstlongfont{\the\glslongtok})},%
4183     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
4184       \protect\glxtrfullsep{\the\glslabeltok}%
4185       (\protect\glsfirstlongfont{\the\glslongpltok})},%
4186     plural={\protect\glsabbvfont{\the\glsshortpltok}}}%
4187   }%
4188   \glshasattribute{\the\glslabeltok}{regular}%
4189   {%
4190     \glissetattribute{\the\glslabeltok}{regular}{false}%
4191   }%
4192 }
```

Unset the regular attribute if it has been set.

```
4187 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
4188   \glshasattribute{\the\glslabeltok}{regular}%
4189   {%
4190     \glissetattribute{\the\glslabeltok}{regular}{false}%
4191   }%
4192 }
```

```

4191 }%
4192 {}%
4193 }%
4194 }%
4195 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

4196 \renewcommand*{\abbrvpluralsuffix}{\glspluralsuffix}%
4197 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
4198 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
4199 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
4200 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

4201 \renewcommand*{\glsxtrfullformat}[2]{%
4202   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
4203   \ifglsxtrininsertinside\else##2\fi
4204   \glsxtrfullsep{##1}%
4205   (\glsfirstlongfont{\glsaccesslong{##1}})%
4206 }%
4207 \renewcommand*{\glsxtrfullplformat}[2]{%
4208   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
4209   \ifglsxtrininsertinside\else##2\fi
4210   \glsxtrfullsep{##1}%
4211   (\glsfirstlongfont{\glsaccesslongpl{##1}})%
4212 }%
4213 \renewcommand*{\Glsxtrfullformat}[2]{%
4214   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
4215   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4216   (\glsfirstlongfont{\glsaccesslong{##1}})%
4217 }%
4218 \renewcommand*{\Glsxtrfullplformat}[2]{%
4219   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
4220   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4221   (\glsfirstlongfont{\glsaccesslongpl{##1}})%
4222 }%
4223 }

```

`short-long-desc` User supplies description. The long form is included in the name.

```

4224 \newabbreviationstyle{short-long-desc}%
4225 {%
4226   \renewcommand*{\CustomAbbreviationFields}{%
4227     name={\protect\glsxtrfullformat{\the\glslabeltok}{}},
4228     sort={\the\glsshorttok},%
4229     first={\protect\glsfirstabbrvfont{\the\glsshorttok}}%
4230     \protect\glsxtrfullsep{\the\glslabeltok}%
4231     (\protect\glsfirstlongfont{\the\glslongtok}}),%
4232     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}}%
4233     \protect\glsxtrfullsep{\the\glslabeltok}%
4234     (\protect\glsfirstlongfont{\the\glslongpltok}}),%

```

4235 plural={\protect\glsabbvfont{\the\glsshortpltok}}}%

Unset the regular attribute if it has been set.

```
4236 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
4237   \glshasattribute{\the\glslabeltok}{regular}%
4238   {%
4239     \glssetattribute{\the\glslabeltok}{regular}{false}%
4240   }%
4241   {}}%
4242 }%
4243 }%
4244 {%
4245   \GlsXtrUseAbbrStyleFmts{short-long}%
4246 }
```

ongfootnotefont Only used by the “footnote” styles.

```
4247 \newcommand*{\glsfirstlongfootnotefont}[1]{\glslongfootnotefont{#1}}%
```

ongfootnotefont Only used by the “footnote” styles.

```
4248 \newcommand*{\glslongfootnotefont}[1]{\glslongdefaultfont{#1}}%
```

footnote Short form followed by long form in footnote on first use. Take care about using `\glsfirst` as this won’t suppress the hyperlink. (Perhaps modify `\glsfirst` to reflect `nohyperfirst` attribute?)

```
4249 \newabbreviationstyle{footnote}%
4250 {%
4251   \renewcommand*{\CustomAbbreviationFields}{%
4252     name={\protect\glsabbvfont{\the\glsshortttok}},
4253     sort={\the\glsshortttok},
4254     description={\the\glslongtok},%
4255     first={\protect\glsfirstabbvfont{\the\glsshortttok}}%
4256     \protect\footnote{\protect\glsfirstlongfont{\the\glslongtok}}},%
4257     firstplural={\protect\glsfirstabbvfont{\the\glsshortpltok}}%
4258     \protect\footnote{\protect\glsfirstlongfont{\the\glslongpltok}}},%
4259     plural={\protect\glsabbvfont{\the\glsshortpltok}}}%
4260 }
```

Switch off hyperlinks on first use to prevent nested hyperlinks, and unset the regular attribute if it has been set.

```
4260 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
4261   \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
4262   \glshasattribute{\the\glslabeltok}{regular}%
4263   {%
4264     \glssetattribute{\the\glslabeltok}{regular}{false}%
4265   }%
4266   {}}%
4267 }%
4268 }%
4269 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```

4270 \renewcommand*\abbrvpluralsuffix{\glspluralsuffix}%
4271 \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
4272 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvdefaultfont{##1}}%
4273 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongfootnotefont{##1}}%
4274 \renewcommand*\glslongfont[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

4275 \renewcommand*\glsxtrfullformat[2]{%
4276   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
4277   \ifglsxtrininsertinside\else##2\fi
4278   \protect\footnote{\glsfirstlongfont{\glsaccesslong{##1}}}%
4279 }%
4280 \renewcommand*\glsxtrfullplformat[2]{%
4281   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
4282   \ifglsxtrininsertinside\else##2\fi
4283   \protect\footnote{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
4284 }%
4285 \renewcommand*\Glsxtrfullformat[2]{%
4286   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
4287   \ifglsxtrininsertinside\else##2\fi
4288   \protect\footnote{\glsfirstlongfont{\glsaccesslong{##1}}}%
4289 }%
4290 \renewcommand*\Glsxtrfullplformat[2]{%
4291   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
4292   \ifglsxtrininsertinside\else##2\fi
4293   \protect\footnote{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
4294 }%

```

The first use full form and the inline full form use the short (long) style.

```

4295 \renewcommand*\glsxtrinlinefullformat[2]{%
4296   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
4297   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4298   (\glsfirstlongfont{\glsaccesslong{##1}})%
4299 }%
4300 \renewcommand*\glsxtrinlinefullplformat[2]{%
4301   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
4302   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4303   (\glsfirstlongfont{\glsaccesslongpl{##1}})%
4304 }%
4305 \renewcommand*\Glsxtrinlinefullformat[2]{%
4306   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
4307   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4308   (\glsfirstlongfont{\glsaccesslong{##1}})%
4309 }%
4310 \renewcommand*\Glsxtrinlinefullplformat[2]{%
4311   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
4312   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4313   (\glsfirstlongfont{\glsaccesslongpl{##1}})%
4314 }%

```


4315 }

short-footnote

4316 \letabbreviationstyle{short-footnote}{footnote}

postfootnote Similar to the above but the footnote is placed afterwards, outside the link. This avoids nested links and can also move the footnote marker after any following punctuation mark. This deferment won't occur with \glsfirst.

4317 \newabbreviationstyle{postfootnote}%

4318 {%

4319 \renewcommand*{\CustomAbbreviationFields}{%

4320 name={\protect\glsabbrvfont{\the\glsshorttok}},

4321 sort={\the\glsshorttok},

4322 description={\the\glslongtok},%

4323 first={\protect\glsfirstabbrvfont{\the\glsshorttok}}%

4324 \protect\footnote{\protect\glsfirstlongfont{\the\glslongtok}}},%

4325 firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}}%

4326 \protect\footnote{\protect\glsfirstlongfont{\the\glslongpltok}}},%

4327 plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%

Make this category insert a footnote after the link if this was the first use, and unset the regular attribute if it has been set.

4328 \renewcommand*{\GlsXtrPostNewAbbreviation}{%

4329 \csdef{glsxtrpostlink\glscategorylabel}{%

4330 \glsxtrifwasfirstuse

4331 {%

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

4332 \glsxtrdopostpunc{\protect\footnote

4333 {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}}}%

4334 }%

4335 {}%

4336 }%

4337 \glshasattribute{\the\glslabeltok}{regular}%

4338 {%

4339 \glssetattribute{\the\glslabeltok}{regular}{false}%

4340 }%

4341 {}%

4342 }%

The footnote needs to be suppressed in the inline form, so \glsxtrfull must set the first use switch off.

4343 \renewcommand*{\glsxtrsetupfulldefs}{%

4344 \let\glsxtrifwasfirstuse\@secondoftwo

4345 }%

4346 }%

4347 {%

In case the user wants to mix and match font styles, these are redefined here.

```

4348 \renewcommand*{\abbrvpluralsuffix}{\glspluralsuffix}%
4349 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
4350 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
4351 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
4352 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form. The long form is deferred.

```

4353 \renewcommand*{\glsxtrfullformat}[2]{%
4354   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
4355   \ifglsxtrininsertinside\else##2\fi
4356 }%
4357 \renewcommand*{\glsxtrfullplformat}[2]{%
4358   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
4359   \ifglsxtrininsertinside\else##2\fi
4360 }%
4361 \renewcommand*{\Glsxtrfullformat}[2]{%
4362   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
4363   \ifglsxtrininsertinside\else##2\fi
4364 }%
4365 \renewcommand*{\Glsxtrfullplformat}[2]{%
4366   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
4367   \ifglsxtrininsertinside\else##2\fi
4368 }%

```

The first use full form and the inline full form use the short (long) style.

```

4369 \renewcommand*{\glsxtrinlinefullformat}[2]{%
4370   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
4371   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4372   (\glsfirstlongfont{\glsaccesslong{##1}})%
4373 }%
4374 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
4375   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
4376   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4377   (\glsfirstlongfont{\glsaccesslongpl{##1}})%
4378 }%
4379 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
4380   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
4381   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4382   (\glsfirstlongfont{\glsaccesslong{##1}})%
4383 }%
4384 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
4385   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
4386   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4387   (\glsfirstlongfont{\glsaccesslongpl{##1}})%
4388 }%
4389 }

```

rt-postfootnote

```

4390 \letabbreviationstyle{short-postfootnote}{postfootnote}

```

short Provide a style that only displays the short form on first use, but the short and long form can be displayed with the “full” commands that use the inline format. If the user supplies a description, the long form won’t be displayed in the predefined glossary styles, but the post description hook can be employed to automatically insert it.

```

4391 \newabbreviationstyle{short}%
4392 {%
4393   \renewcommand*{\CustomAbbreviationFields}{%
4394     name={\protect\glsabbrvfont{\the\glsshorttok}},
4395     sort={\the\glsshorttok},
4396     first={\protect\glsfirstabbrvfont{\the\glsshorttok}},
4397     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},
4398     text={\protect\glsabbrvfont{\the\glsshorttok}},
4399     plural={\protect\glsabbrvfont{\the\glsshortpltok}},
4400     description={\the\glslongtok}}%
4401   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
4402     \glssetattribute{\the\glslabeltok}{regular}{true}}%
4403 }%
4404 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

4405   \renewcommand*{\abbrvpluralsuffix}{\glspluralsuffix}%
4406   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
4407   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
4408   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
4409   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

4410   \renewcommand*{\glsxtrinlinefullformat}[2]{%
4411     \protect\glsfirstabbrvfont{\glsaccessshort{##1}}%
4412     \ifglsxtrininsertinside##2\fi}%
4413     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4414     (\glsfirstlongfont{\glsaccesslong{##1}})%
4415   }%
4416   \renewcommand*{\glsxtrinlinefullplformat}[2]{%
4417     \protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}%
4418     \ifglsxtrininsertinside##2\fi}%
4419     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4420     (\glsfirstlongfont{\glsaccesslongpl{##1}})%
4421   }%
4422   \renewcommand*{\Glsxtrinlinefullformat}[2]{%
4423     \protect\glsfirstabbrvfont{\glsaccessshort{##1}}%
4424     \ifglsxtrininsertinside##2\fi}%
4425     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4426     (\glsfirstlongfont{\Glsaccesslong{##1}})%
4427   }%
4428   \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
4429     \protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}%
4430     \ifglsxtrininsertinside##2\fi}%
4431     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4432     (\glsfirstlongfont{\Glsaccesslongpl{##1}})%

```

4433 }%

The first use full form only displays the short form, but it typically won't be used as the regular attribute is set by this style.

```
4434 \renewcommand*{\glxtrfullformat}[2]{%
4435   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
4436   \ifglxtrinsertinside\else##2\fi
4437 }%
4438 \renewcommand*{\glxtrfullplformat}[2]{%
4439   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
4440   \ifglxtrinsertinside\else##2\fi
4441 }%
4442 \renewcommand*{\Glsxtrfullformat}[2]{%
4443   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
4444   \ifglxtrinsertinside\else##2\fi
4445 }%
4446 \renewcommand*{\Glsxtrfullplformat}[2]{%
4447   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
4448   \ifglxtrinsertinside\else##2\fi
4449 }%
4450 }
```

Set this as the default style for acronyms:

```
4451 \setabbreviationstyle[acronym]{short}
```

short-nolong

```
4452 \letabbreviationstyle{short-nolong}{short}
```

short-desc The user must supply the description in this style. The long form is added to the name. The short style (possibly with the post-description hooks set) might be a better option.

```
4453 \newabbreviationstyle{short-desc}%
4454 {%
4455   \renewcommand*{\CustomAbbreviationFields}{%
4456     name={\protect\glxtrinlinefullformat{\the\glslabeltok}{}},
4457     sort={\the\glsshorttok},
4458     first={\protect\glsfirstabbrvfont{\the\glsshorttok}},
4459     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},
4460     text={\protect\glsabbrvfont{\the\glsshorttok}},
4461     plural={\protect\glsabbrvfont{\the\glsshortpltok}},
4462     description={\the\glslongtok}}%
4463   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
4464     \glssetattribute{\the\glslabeltok}{regular}{true}}%
4465 }%
4466 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
4467 \renewcommand*{\abbrvpluralsuffix}{\glspluralsuffix}%
4468 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
4469 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
```

```

4470 \renewcommand*\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
4471 \renewcommand*\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

4472 \renewcommand*\glsxtrinlinefullformat}[2]{%
4473   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
4474   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4475   (\glsfirstlongfont{\glsaccesslong{##1}})%
4476 }%
4477 \renewcommand*\glsxtrinlinefullplformat}[2]{%
4478   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
4479   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4480   (\glsfirstlongfont{\glsaccesslongpl{##1}})%
4481 }%
4482 \renewcommand*\Glsxtrinlinefullformat}[2]{%
4483   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
4484   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4485   (\glsfirstlongfont{\Glsaccesslong{##1}})%
4486 }%
4487 \renewcommand*\Glsxtrinlinefullplformat}[2]{%
4488   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
4489   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4490   (\glsfirstlongfont{\Glsaccesslongpl{##1}})%
4491 }%

```

The first use full form only displays the short form, but it typically won't be used as the regular attribute is set by this style.

```

4492 \renewcommand*\glsxtrfullformat}[2]{%
4493   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
4494   \ifglsxtrininsertinside\else##2\fi
4495 }%
4496 \renewcommand*\glsxtrfullplformat}[2]{%
4497   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
4498   \ifglsxtrininsertinside\else##2\fi
4499 }%
4500 \renewcommand*\Glsxtrfullformat}[2]{%
4501   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
4502   \ifglsxtrininsertinside\else##2\fi
4503 }%
4504 \renewcommand*\Glsxtrfullplformat}[2]{%
4505   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
4506   \ifglsxtrininsertinside\else##2\fi
4507 }%
4508 }

```

ort-nolong-desc

```

4509 \letabbreviationstyle{short-nolong-desc}{short-desc}

```

long-desc Provide a style that only displays the long form, but the long and short form can be displayed with the “full” commands that use the inline format. The predefined glossary styles won't

show the short form. The user must supply a description for this style.

```

4510 \newabbreviationstyle{long-desc}%
4511 {%
4512   \renewcommand*{\CustomAbbreviationFields}{%
4513     name={\protect\protect\glsfirstlongfont{\the\glslongtok}},
4514     sort={\the\glslongtok},
4515     first={\protect\glsfirstlongfont{\the\glslongtok}},
4516     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},
4517     text={\the\glslongtok},
4518     plural={\the\glslongpltok}%
4519   }%
4520   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
4521     \glssetattribute{\the\glslabeltok}{regular}{true}}%
4522 }%
4523 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

4524 \renewcommand*{\abbrvpluralsuffix}{\glspluralsuffix}%
4525 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
4526 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
4527 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
4528 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the long format followed by the short form in parentheses.

```

4529 \renewcommand*{\glsxtrinlinefullformat}[2]{%
4530   \glsfirstlongfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
4531   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4532   (\protect\glsfirstabbrvfont{\glsaccessshort{##1}})%
4533 }%
4534 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
4535   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
4536   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4537   (\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}})%
4538 }%
4539 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
4540   \glsfirstlongfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
4541   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4542   (\protect\glsfirstabbrvfont{\glsaccessshort{##1}})%
4543 }%
4544 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
4545   \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
4546   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4547   (\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}})%
4548 }%

```

The first use full form only displays the long form, but it typically won't be used as the regular attribute is set by this style.

```

4549 \renewcommand*{\glsxtrfullformat}[2]{%
4550   \glsfirstlongfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
4551   \ifglsxtrininsertinside\else##2\fi

```

```

4552 }%
4553 \renewcommand*{\glxtrfullplformat}[2]{%
4554   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
4555   \ifglxtrinsertinside\else##2\fi
4556 }%
4557 \renewcommand*{\Glsxtrfullformat}[2]{%
4558   \glsfirstlongfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
4559   \ifglxtrinsertinside\else##2\fi
4560 }%
4561 \renewcommand*{\Glsxtrfullplformat}[2]{%
4562   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
4563   \ifglxtrinsertinside\else##2\fi
4564 }%
4565 }

```

ng-noshort-desc Provide a synonym that matches similar styles.

```

4566 \letabbreviationstyle{long-noshort-desc}{long-desc}

```

long It doesn't really make a great deal of sense to have a long-only style that doesn't have a description, but the best course of action here is to use the short form as the name and the long form as the description.

```

4567 \newabbreviationstyle{long}%
4568 {%
4569   \renewcommand*{\CustomAbbreviationFields}{%
4570     name={\protect\glsabbrvfont{\the\glsshorttok}},
4571     sort={\the\glsshorttok},
4572     first={\protect\glsfirstlongfont{\the\glslongtok}},
4573     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},
4574     text={\the\glslongtok},
4575     plural={\the\glslongpltok},%
4576     description={\the\glslongtok}%
4577   }%
4578   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
4579     \glssetattribute{\the\glslabeltok}{regular}{true}}%
4580 }%
4581 {%
4582   \GlsXtrUseAbbrStyleFmts{long-desc}%
4583 }

```

long-noshort Provide a synonym that matches similar styles.

```

4584 \letabbreviationstyle{long-noshort}{long}

```

1.6.3 Predefined Styles (Small Capitals)

These styles use:

\glxtrscfont

```

4585 \newcommand*{\glxtrscfont}[1]{\textsc{#1}}

```

sxtrfirstscfont

```
4586 \newcommand*{\glxtrfirstscfont}[1]{\glxtrscfont{#1}}
```

and for the default short form suffix:

\glxtrscsuffix

```
4587 \newcommand*{\glxtrscsuffix}{\glstextup{\glspluralsuffix}}
```

long-short-sc

```
4588 \newabbreviationstyle{long-short-sc}%
4589 {%
4590   \GlsXtrUseAbbrStyleSetup{long-short}%
4591 }%
4592 {%
```

Mostly as long-short style:

```
4593   \GlsXtrUseAbbrStyleFmts{long-short}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
4594   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
4595   \renewcommand*{\glsabbrvfont}[1]{\glxtrscfont{##1}}%
4596   \renewcommand*{\glsfirstabbrvfont}[1]{\glxtrfirstscfont{##1}}%
4597 }
```

g-short-sc-desc

```
4598 \newabbreviationstyle{long-short-sc-desc}%
4599 {%
4600   \GlsXtrUseAbbrStyleSetup{long-short-desc}%
4601 }%
4602 {%
```

Mostly as long-short-desc style:

```
4603   \GlsXtrUseAbbrStyleFmts{long-short-desc}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
4604   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
4605   \renewcommand*{\glsabbrvfont}[1]{\glxtrscfont{##1}}%
4606   \renewcommand*{\glsfirstabbrvfont}[1]{\glxtrfirstscfont{##1}}%
4607 }
```

Now the short (long) version

```
4608 \newabbreviationstyle{short-sc-long}%
4609 {%
4610   \GlsXtrUseAbbrStyleSetup{short-long}%
4611 }%
4612 {%
```

Mostly as short-long style:

```
4613   \GlsXtrUseAbbrStyleFmts{short-long}%
```


Use smallcaps and adjust the plural suffix to revert to upright.

```
4614 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
4615 \renewcommand*{\glssabrvfont[1]{\glxtrscfont{##1}}}%
4616 \renewcommand*{\glssfirstabrvfont[1]{\glxtrfirstscfont{##1}}}%
4617 }
```

As before but user provides description

```
4618 \newabbreviationstyle{short-sc-long-desc}%
4619 {%
4620 \GlsXtrUseAbbrStyleSetup{short-long-desc}%
4621 }%
4622 {%
```

Mostly as short-long-desc style:

```
4623 \GlsXtrUseAbbrStyleFmts{short-long-desc}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
4624 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
4625 \renewcommand*{\glssabrvfont[1]{\glxtrscfont{##1}}}%
4626 \renewcommand*{\glssfirstabrvfont[1]{\glxtrfirstscfont{##1}}}%
4627 }
```

short-sc

```
4628 \newabbreviationstyle{short-sc}%
4629 {%
4630 \GlsXtrUseAbbrStyleSetup{short-nolong}%
4631 }%
4632 {%
```

Mostly as short style:

```
4633 \GlsXtrUseAbbrStyleFmts{short-nolong}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
4634 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
4635 \renewcommand*{\glssabrvfont[1]{\glxtrscfont{##1}}}%
4636 \renewcommand*{\glssfirstabrvfont[1]{\glxtrfirstscfont{##1}}}%
4637 }
```

short-sc-nolong

```
4638 \letabbreviationstyle{short-sc-nolong}{short-sc}
```

short-sc-desc

```
4639 \newabbreviationstyle{short-sc-desc}%
4640 {%
4641 \GlsXtrUseAbbrStyleSetup{short-desc}%
4642 }%
4643 {%
```

Mostly as short style:

```
4644 \GlsXtrUseAbbrStyleFmts{short-desc}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
4645 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
4646 \renewcommand*{\glssabrvfont[1]{\glxtrscfont{##1}}}%
4647 \renewcommand*{\glssfirstabrvfont[1]{\glxtrfirstscfont{##1}}}%
4648 }
```

-sc-nolong-desc

```
4649 \letabbreviationstyle{short-sc-nolong-desc}{short-sc-desc}
```

long-noshort-sc The smallcaps font will only be used if the short form is explicitly invoked through commands like \glssshort.

```
4650 \newabbreviationstyle{long-noshort-sc}%
4651 {%
4652 \GlsXtrUseAbbrStyleSetup{long-noshort}%
4653 }%
4654 {%
```

Mostly as long style:

```
4655 \GlsXtrUseAbbrStyleFmts{long-noshort}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
4656 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
4657 \renewcommand*{\glssabrvfont[1]{\glxtrscfont{##1}}}%
4658 \renewcommand*{\glssfirstabrvfont[1]{\glxtrfirstscfont{##1}}}%
4659 }
```

long-sc Backward compatibility:

```
4660 \@glxtr@deprecated@abbrstyle{long-sc}{long-noshort-sc}
```

noshort-sc-desc The smallcaps font will only be used if the short form is explicitly invoked through commands like \glssshort.

```
4661 \newabbreviationstyle{long-noshort-sc-desc}%
4662 {%
4663 \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
4664 }%
4665 {%
```

Mostly as long style:

```
4666 \GlsXtrUseAbbrStyleFmts{long-noshort-desc}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
4667 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
4668 \renewcommand*{\glssabrvfont[1]{\glxtrscfont{##1}}}%
4669 \renewcommand*{\glssfirstabrvfont[1]{\glxtrfirstscfont{##1}}}%
4670 }
```

long-desc-sc Backward compatibility:

```
4671 \@glxtr@deprecated@abbrstyle{long-desc-sc}{long-noshort-sc-desc}
```

ort-sc-footnote

```
4672 \newabbreviationstyle{short-sc-footnote}%  
4673 {%  
4674   \GlsXtrUseAbbrStyleSetup{short-footnote}%  
4675 }%  
4676 {%
```

Mostly as long style:

```
4677   \GlsXtrUseAbbrStyleFmts{short-footnote}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
4678   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%  
4679   \renewcommand*\glsabbrvfont[1]{\glxtrscfont{##1}}%  
4680   \renewcommand*\glsfirstabbrvfont[1]{\glxtrfirstscfont{##1}}%  
4681 }
```

footnote-sc Backward compatibility:

```
4682 \@glxtr@deprecated@abbrstyle{footnote-sc}{short-sc-footnote}
```

sc-postfootnote

```
4683 \newabbreviationstyle{short-sc-postfootnote}%  
4684 {%  
4685   \GlsXtrUseAbbrStyleSetup{short-postfootnote}%  
4686 }%  
4687 {%
```

Mostly as long style:

```
4688   \GlsXtrUseAbbrStyleFmts{short-postfootnote}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
4689   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%  
4690   \renewcommand*\glsabbrvfont[1]{\glxtrscfont{##1}}%  
4691   \renewcommand*\glsfirstabbrvfont[1]{\glxtrfirstscfont{##1}}%  
4692 }
```

postfootnote-sc Backward compatibility:

```
4693 \@glxtr@deprecated@abbrstyle{postfootnote-sc}{short-sc-postfootnote}
```

1.6.4 Predefined Styles (Fake Small Capitals)

These styles require the relsize package, which must be loaded by the user. These styles all use:

\glxtrsmfont

```
4694 \newcommand*{\glxtrsmfont}[1]{\textsmaller{#1}}
```

sxtrfirstsmfont

```
4695 \newcommand*{\glxtrfirstsmfont}[1]{\glxtrsmfont{#1}}
```

and for the default short form suffix:

```

\glxtrmsuffix
4696 \newcommand*{\glxtrmsuffix}{\glspluralsuffix}

long-short-sm
4697 \newabbreviationstyle{long-short-sm}%
4698 {%
4699   \GlsXtrUseAbbrStyleSetup{long-short}%
4700 }%
4701 {%
    Mostly as long-short style:
4702   \GlsXtrUseAbbrStyleFmts{long-short}%
4703   \renewcommand*{\glsabbrvfont[1]{\glxtrsmfont{##1}}}%
4704   \renewcommand*{\glsfirstabbrvfont[1]{\glxtrfirstsmfont{##1}}}%
4705   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrmsuffix}%
4706 }

g-short-sm-desc
4707 \newabbreviationstyle{long-short-sm-desc}%
4708 {%
4709   \GlsXtrUseAbbrStyleSetup{long-short-desc}%
4710 }%
4711 {%
    Mostly as long-short-desc style:
4712   \GlsXtrUseAbbrStyleFmts{long-short-desc}%
4713   \renewcommand*{\glsabbrvfont[1]{\glxtrsmfont{##1}}}%
4714   \renewcommand*{\glsfirstabbrvfont[1]{\glxtrfirstsmfont{##1}}}%
4715   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrmsuffix}%
4716 }

short-sm-long   Now the short (long) version
4717 \newabbreviationstyle{short-sm-long}%
4718 {%
4719   \GlsXtrUseAbbrStyleSetup{short-long}%
4720 }%
4721 {%
    Mostly as short-long style:
4722   \GlsXtrUseAbbrStyleFmts{short-long}%
4723   \renewcommand*{\glsabbrvfont[1]{\glxtrsmfont{##1}}}%
4724   \renewcommand*{\glsfirstabbrvfont[1]{\glxtrfirstsmfont{##1}}}%
4725   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrmsuffix}%
4726 }

rt-sm-long-desc  As before but user provides description
4727 \newabbreviationstyle{short-sm-long-desc}%
4728 {%
4729   \GlsXtrUseAbbrStyleSetup{short-long-desc}%
4730 }%
4731 {%

```

Mostly as short-long-desc style:

```
4732 \GlsXtrUseAbbrStyleFmts{short-long-desc}%
4733 \renewcommand*\glsabbrvfont[1]{\glxtrsmfont{##1}}%
4734 \renewcommand*\glsfirstabbrvfont[1]{\glxtrfirstsmfont{##1}}%
4735 \renewcommand*\abbrvpluralsuffix{\protect\glxtrsmsuffix}%
4736 }
```

short-sm

```
4737 \newabbreviationstyle{short-sm}%
4738 {%
4739 \GlsXtrUseAbbrStyleSetup{short-nolong}%
4740 }%
4741 {%
```

Mostly as short style:

```
4742 \GlsXtrUseAbbrStyleFmts{short-nolong}%
4743 \renewcommand*\glsabbrvfont[1]{\glxtrsmfont{##1}}%
4744 \renewcommand*\glsfirstabbrvfont[1]{\glxtrfirstsmfont{##1}}%
4745 \renewcommand*\abbrvpluralsuffix{\protect\glxtrsmsuffix}%
4746 }
```

short-sm-nolong

```
4747 \letabbreviationstyle{short-sm-nolong}{short-sm}
```

short-sm-desc

```
4748 \newabbreviationstyle{short-sm-desc}%
4749 {%
4750 \GlsXtrUseAbbrStyleSetup{short-nolong-desc}%
4751 }%
4752 {%
```

Mostly as short style:

```
4753 \GlsXtrUseAbbrStyleFmts{short-nolong-desc}%
4754 \renewcommand*\glsabbrvfont[1]{\glxtrsmfont{##1}}%
4755 \renewcommand*\glsfirstabbrvfont[1]{\glxtrfirstsmfont{##1}}%
4756 \renewcommand*\abbrvpluralsuffix{\protect\glxtrsmsuffix}%
4757 }
```

-sm-nolong-desc

```
4758 \letabbreviationstyle{short-sm-nolong-desc}{short-sm-desc}
```

long-noshort-sm The smallcaps font will only be used if the short form is explicitly invoked through commands like \glsshort.

```
4759 \newabbreviationstyle{long-noshort-sm}%
4760 {%
4761 \GlsXtrUseAbbrStyleSetup{long-noshort}%
4762 }%
4763 {%
```

Mostly as long style:

```
4764 \GlsXtrUseAbbrStyleFmts{long-noshort}%
4765 \renewcommand*{\glsabbrvfont[1]{\glxtrsmfont{##1}}}%
4766 \renewcommand*{\glsfirstabbrvfont[1]{\glxtrfirstsmfont{##1}}}%
4767 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrsmsuffix}%
4768 }
```

long-sm Backward compatibility:

```
4769 \@glxtr@deprecated@abbrstyle{long-sm}{long-noshort-sm}
```

noshort-sm-desc The smaller font will only be used if the short form is explicitly invoked through commands like \glsshort.

```
4770 \newabbreviationstyle{long-noshort-sm-desc}%
4771 {%
4772 \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
4773 }%
4774 {%
```

Mostly as long style:

```
4775 \GlsXtrUseAbbrStyleFmts{long-noshort-desc}%
4776 \renewcommand*{\glsabbrvfont[1]{\glxtrsmfont{##1}}}%
4777 \renewcommand*{\glsfirstabbrvfont[1]{\glxtrfirstsmfont{##1}}}%
4778 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrsmsuffix}%
4779 }
```

long-desc-sm Backward compatibility:

```
4780 \@glxtr@deprecated@abbrstyle{long-desc-sm}{long-noshort-sm-desc}
```

short-sm-footnote

```
4781 \newabbreviationstyle{short-sm-footnote}%
4782 {%
4783 \GlsXtrUseAbbrStyleSetup{short-footnote}%
4784 }%
4785 {%
```

Mostly as long style:

```
4786 \GlsXtrUseAbbrStyleFmts{short-footnote}%
4787 \renewcommand*{\glsabbrvfont[1]{\glxtrsmfont{##1}}}%
4788 \renewcommand*{\glsfirstabbrvfont[1]{\glxtrfirstsmfont{##1}}}%
4789 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrsmsuffix}%
4790 }
```

footnote-sm Backward compatibility:

```
4791 \@glxtr@deprecated@abbrstyle{footnote-sm}{short-sm-footnote}
```

sm-postfootnote

```
4792 \newabbreviationstyle{short-sm-postfootnote}%
4793 {%
4794 \GlsXtrUseAbbrStyleSetup{short-postfootnote}%
```

4795 }%
 4796 {%

Mostly as long style:

4797 \GlsXtrUseAbbrStyleFmts{short-postfootnote}%
 4798 \renewcommand*\glsabbrvfont[1]{\glsxtrsmfont{##1}}%
 4799 \renewcommand*\glsfirstabbrvfont[1]{\glsxtrfirstsmfont{##1}}%
 4800 \renewcommand*\abbrvpluralsuffix{\protect\glsxtrsmsuffix}%
 4801 }

postfootnote-sm Backward compatibility:

4802 \@glsxtr@deprecated@abbrstyle{postfootnote-sm}{short-sm-postfootnote}

1.6.5 Predefined Styles (Emphasized)

These styles use \emph for the short form.

\glsabbrvemfont

4803 \newcommand*\glsabbrvemfont[1]{\emph{#1}}%

irstabbrvemfont

4804 \newcommand*\glsfirstabbrvemfont[1]{\glsabbrvemfont{#1}}%

firstlongemfont Only used by the “long-em” styles.

4805 \newcommand*\glsfirstlongemfont[1]{\glslongemfont{#1}}%

\glslongemfont Only used by the “long-em” styles.

4806 \newcommand*\glslongemfont[1]{\emph{#1}}%

long-short-em

4807 \newabbreviationstyle{long-short-em}%
 4808 {%
 4809 \GlsXtrUseAbbrStyleSetup{long-short}%
 4810 }%
 4811 {%

Mostly as long-short style:

4812 \GlsXtrUseAbbrStyleFmts{long-short}%
 4813 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
 4814 }

g-short-em-desc

4815 \newabbreviationstyle{long-short-em-desc}%
 4816 {%
 4817 \GlsXtrUseAbbrStyleSetup{long-short-desc}%
 4818 }%
 4819 {%

Mostly as long-short-desc style:

```
4820 \GlsXtrUseAbbrStyleFmts{long-short-desc}%
4821 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
4822 }
```

ong-em-short-em

```
4823 \newabbreviationstyle{long-em-short-em}%
4824 {%
```

\glslongemfont is used in the description since \glsdesc doesn't set the style.

```
4825 \renewcommand*{\CustomAbbreviationFields}{%
4826   name={\protect\glsabbrvfont{\the\glsshorttok}},
4827   sort={\the\glsshorttok},
4828   first={\protect\glsfirstlongfont{\the\glslongtok}%
4829     \protect\glsxtrfullsep{\the\glslabeltok}%
4830     (\protect\glsfirstabbrvfont{\the\glsshorttok})},%
4831   firstplural={\protect\glsfirstlongfont{\the\glslongpltok}%
4832     \protect\glsxtrfullsep{\the\glslabeltok}%
4833     (\protect\glsfirstabbrvfont{\the\glsshortpltok})},%
4834   plural={\protect\glsabbrvfont{\the\glsshortpltok}},%
4835   description={\protect\glslongemfont{\the\glslongtok}}}%

```

Unset the regular attribute if it has been set.

```
4836 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
4837   \glshasattribute{\the\glslabeltok}{regular}%
4838   {%
4839     \glssetattribute{\the\glslabeltok}{regular}{false}%
4840   }%
4841   {}%
4842 }%
4843 }%
4844 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
4845 \GlsXtrUseAbbrStyleFmts{long-short}%
4846 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
4847 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
4848 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
4849 \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%
4850 }
```

m-short-em-desc

```
4851 \newabbreviationstyle{long-em-short-em-desc}%
4852 {%
4853   \GlsXtrUseAbbrStyleSetup{long-short-desc}%
4854 }%
4855 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
4856 \GlsXtrUseAbbrStyleFmts{long-short-desc}%

```



```

4857 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
4858 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
4859 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
4860 \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%
4861 }

```

short-em-long Now the short (long) version

```

4862 \newabbreviationstyle{short-em-long}%
4863 {%
4864   \GlsXtrUseAbbrStyleSetup{short-long}%
4865 }%
4866 {%

```

Mostly as short-long style:

```

4867   \GlsXtrUseAbbrStyleFmts{short-long}%
4868   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
4869   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
4870 }

```

short-em-long-desc As before but user provides description

```

4871 \newabbreviationstyle{short-em-long-desc}%
4872 {%
4873   \GlsXtrUseAbbrStyleSetup{short-long-desc}%
4874 }%
4875 {%

```

Mostly as short-long-desc style:

```

4876   \GlsXtrUseAbbrStyleFmts{short-long-desc}%
4877   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
4878   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
4879   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
4880   \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%
4881 }

```

short-em-long-em

```

4882 \newabbreviationstyle{short-em-long-em}%
4883 {%

```

\glslongemfont is used in the description since \glsdesc doesn't set the style.

```

4884   \renewcommand*{\CustomAbbreviationFields}{%
4885     name={\protect\glsabbrvfont{\the\glsshorttok}},
4886     sort={\the\glsshorttok},
4887     description={\protect\glslongemfont{\the\glslongtok}},%
4888     first={\protect\glsfirstabbrvfont{\the\glsshorttok}}%
4889     \protect\glsxtrfullsep{\the\glslabeltok}%
4890     (\protect\glsfirstlongfont{\the\glslongtok}}),%
4891     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}}%
4892     \protect\glsxtrfullsep{\the\glslabeltok}%
4893     (\protect\glsfirstlongfont{\the\glslongpltok}}),%
4894     plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```
4895 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
4896   \glshasattribute{\the\glslabeltok}{regular}%
4897   {%
4898     \glsselattribute{\the\glslabeltok}{regular}{false}%
4899   }%
4900   {}%
4901 }%
4902 }%
4903 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
4904 \GlsXtrUseAbbrStyleFmts{short-long}%
4905 \renewcommand*{\glssabrvfont}[1]{\glssabrvemfont{##1}}%
4906 \renewcommand*{\glsfirstabrvfont}[1]{\glsfirstabrvemfont{##1}}%
4907 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
4908 \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%
4909 }
```

em-long-em-desc

```
4910 \newabbreviationstyle{short-em-long-em-desc}%
4911 {%
4912   \GlsXtrUseAbbrStyleSetup{short-long-desc}%
4913 }%
4914 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
4915 \GlsXtrUseAbbrStyleFmts{short-long-desc}%
4916 \renewcommand*{\glssabrvfont}[1]{\glssabrvemfont{##1}}%
4917 \renewcommand*{\glsfirstabrvfont}[1]{\glsfirstabrvemfont{##1}}%
4918 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
4919 \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%
4920 }
```

short-em

```
4921 \newabbreviationstyle{short-em}%
4922 {%
4923   \GlsXtrUseAbbrStyleSetup{short-nolong}%
4924 }%
4925 {%
```

Mostly as short style:

```
4926 \GlsXtrUseAbbrStyleFmts{short-nolong}%
4927 \renewcommand*{\glssabrvfont}[1]{\glssabrvemfont{##1}}%
4928 \renewcommand*{\glsfirstabrvfont}[1]{\glsfirstabrvemfont{##1}}%
4929 }
```

short-em-nolong

```
4930 \letabbreviationstyle{short-em-nolong}{short-em}
```

```

short-em-desc
4931 \newabbreviationstyle{short-em-desc}%
4932 {%
4933   \GlsXtrUseAbbrStyleSetup{short-nolong-desc}%
4934 }%
4935 {%

    Mostly as short style:
4936   \GlsXtrUseAbbrStyleFmts{short-nolong-desc}%
4937   \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
4938   \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
4939 }

-em-nolong-desc
4940 \letabbreviationstyle{short-em-nolong-desc}{short-em-desc}

long-noshort-em  The short form is explicitly invoked through commands like \glsshort.
4941 \newabbreviationstyle{long-noshort-em}%
4942 {%
4943   \GlsXtrUseAbbrStyleSetup{long-noshort}%
4944 }%
4945 {%

    Mostly as long-noshort style:
4946   \GlsXtrUseAbbrStyleFmts{long-noshort}%
4947   \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
4948   \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
4949 }

long-em  Backward compatibility:
4950 \@glsxtr@deprecated@abbrstyle{long-em}{long-noshort-em}

g-em-noshort-em  The short form is explicitly invoked through commands like \glsshort.
4951 \newabbreviationstyle{long-em-noshort-em}%
4952 {%
4953   \renewcommand*\CustomAbbreviationFields{%
4954     name={\protect\glsabbrvfont{\the\glsshorttok}},
4955     sort={\the\glsshorttok},
4956     first={\protect\glsfirstlongfont{\the\glslongtok}},
4957     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},
4958     text={\the\glslongtok},
4959     plural={\the\glslongpltok},%
4960     description={\protect\glslongemfont{\the\glslongtok}}%
4961   }%
4962   \renewcommand*\GlsXtrPostNewAbbreviation{%
4963     \glssetAttribute{\the\glslabeltok}{regular}{true}}%
4964 }%
4965 {%

```

Mostly as long-noshort style:

```
4966 \GlsXtrUseAbbrStyleFmts{long-noshort}%
4967 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
4968 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
4969 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongemfont{##1}}%
4970 \renewcommand*\glslongfont[1]{\glslongemfont{##1}}%
4971 }
```

noshort-em-desc The emphasized font will only be used if the short form is explicitly invoked through commands like \glsshort.

```
4972 \newabbreviationstyle{long-noshort-em-desc}%
4973 {%
4974 \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
4975 }%
4976 {%
```

Mostly as long style:

```
4977 \GlsXtrUseAbbrStyleFmts{long-noshort-desc}%
4978 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
4979 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
4980 }
```

long-desc-em Backward compatibility:

```
4981 \@glsxtr@deprecated@abbrstyle{long-desc-em}{long-noshort-em-desc}
```

noshort-em-desc The short form is explicitly invoked through commands like \glsshort. The long form is emphasized.

```
4982 \newabbreviationstyle{long-em-noshort-em-desc}%
4983 {%
4984 \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
4985 }%
4986 {%
```

Mostly as long style:

```
4987 \GlsXtrUseAbbrStyleFmts{long-noshort-desc}%
4988 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
4989 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
4990 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongemfont{##1}}%
4991 \renewcommand*\glslongfont[1]{\glslongemfont{##1}}%
4992 }
```

ort-em-footnote

```
4993 \newabbreviationstyle{short-em-footnote}%
4994 {%
4995 \GlsXtrUseAbbrStyleSetup{short-footnote}%
4996 }%
4997 {%
```

Mostly as long style:

```
4998 \GlsXtrUseAbbrStyleFmts{short-footnote}%
4999 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
5000 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
5001 }
```

footnote-em Backward compatibility:

```
5002 \@glsxtr@deprecated@abbrstyle{footnote-em}{short-em-footnote}
```

em-postfootnote

```
5003 \newabbreviationstyle{short-em-postfootnote}%
5004 {%
5005 \GlsXtrUseAbbrStyleSetup{short-postfootnote}%
5006 }%
5007 {%
```

Mostly as long style:

```
5008 \GlsXtrUseAbbrStyleFmts{short-postfootnote}%
5009 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
5010 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
5011 }
```

postfootnote-em Backward compatibility:

```
5012 \@glsxtr@deprecated@abbrstyle{postfootnote-em}{short-em-postfootnote}
```

1.6.6 Predefined Styles (User Parentheses Hook)

These styles allow the user to adjust the parenthetical forms. These styles all test for the existence of the field given by:

glsxtruserfield Default is the useri field.

```
5013 \newcommand*\glsxtruserfield{useri}
```

glsxtruserparen The format of the parenthetical information. The first argument is the long/short form. The second argument is the entry's label. If \glscurrentfieldvalue has been defined, then we have at least glossaries v4.23, which makes it easier for the user to adjust this.

```
5014 \ifdef\glscurrentfieldvalue
5015 {
5016 \newcommand*\glsxtruserparen[2]{%
5017 \glsxtrfullsep{#2}%
5018 (#1\ifglshasfield{\glsxtruserfield}{#2}{, \glscurrentfieldvalue}{})%
5019 }
5020 }
5021 {
5022 \newcommand*\glsxtruserparen[2]{%
5023 \glsxtrfullsep{#2}%
5024 (#1\ifglshasfield{\glsxtruserfield}{#2}{, \@glo@thisvalue}{})%
5025 }
5026 }
```

Font used for short form:

lsabbrvuserfont

```
5027 \newcommand*{\glsabbrvuserfont}[1]{#1}
```

Font used for short form on first use:

stabbrvuserfont

```
5028 \newcommand*{\glsfirstabbrvuserfont}[1]{\glsabbrvuserfont{#1}}
```

Font used for long form:

glslonguserfont

```
5029 \newcommand*{\glslonguserfont}[1]{#1}
```

Font used for long form on first use:

rstlonguserfont

```
5030 \newcommand*{\glsfirstlonguserfont}[1]{\glslonguserfont{#1}}
```

The default short form suffix:

lsxtrusersuffix

```
5031 \newcommand*{\glsxtrusersuffix}{\glspluralsuffix}
```

long-short-user

```
5032 \newabbreviationstyle{long-short-user}%  
5033 {%
```

\glslonguserfont is used in the description since \glsdesc doesn't set the style.

```
5034 \renewcommand*{\CustomAbbreviationFields}{%  
5035   name={\protect\glsabbrvfont{\the\glsshorttok}},  
5036   sort={\the\glsshorttok},  
5037   first={\protect\glsfirstlongfont{\the\glslongtok}}%  
5038   \protect\glsxtruserparen{\protect\glsfirstabbrvfont{\the\glsshorttok}}{\the\glslabeltok}},  
5039   firstplural={\protect\glsfirstlongfont{\the\glslongpltok}}%  
5040   \protect\glsxtruserparen{\protect\glsfirstabbrvfont{\the\glsshortpltok}}{\the\glslabeltok}}%  
5041   plural={\protect\glsabbbfont{\the\glsshortpltok}},%  
5042   description={\protect\glslonguserfont{\the\glslongtok}}}%
```

Unset the regular attribute if it has been set.

```
5043 \renewcommand*{\GlsXtrPostNewAbbreviation}{%  
5044   \glshasattribute{\the\glslabeltok}{regular}%  
5045   {%  
5046     \glssetattribute{\the\glslabeltok}{regular}{false}%  
5047   }%  
5048   {}%  
5049 }%  
5050 }%  
5051 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```

5052 \renewcommand*{\abbrvpluralsuffix}{\glxtrusersuffix}%
5053 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
5054 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
5055 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
5056 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

5057 \renewcommand*{\glxtrfullformat}[2]{%
5058   \glsfirstlongfont{\glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
5059   \ifglxtrininsertinside\else##2\fi
5060   \glxtruserparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}{##1}%
5061 }%
5062 \renewcommand*{\glxtrfullplformat}[2]{%
5063   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
5064   \ifglxtrininsertinside\else##2\fi
5065   \glxtruserparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}{##1}%
5066 }%
5067 \renewcommand*{\Glsxtrfullformat}[2]{%
5068   \glsfirstlongfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
5069   \ifglxtrininsertinside\else##2\fi
5070   \glxtruserparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}{##1}%
5071 }%
5072 \renewcommand*{\Glsxtrfullplformat}[2]{%
5073   \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
5074   \ifglxtrininsertinside\else##2\fi
5075   \glxtruserparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}{##1}%
5076 }%
5077 }

```

short-user-desc

```

5078 \newabbreviationstyle{long-short-user-desc}%
5079 {%
5080   \GlsXtrUseAbbrStyleSetup{long-short-desc}%
5081 }%
5082 {%
5083   \GlsXtrUseAbbrStyleFmts{long-short-user}%
5084 }

```

short-long-user

```

5085 \newabbreviationstyle{short-long-user}%
5086 {%

```

\glslonguserfont is used in the description since \glsdesc doesn't set the style.

```

5087 \renewcommand*{\CustomAbbreviationFields}{%
5088   name={\protect\glsabbrvfont{\the\glsshorttok}},
5089   sort={\the\glsshorttok},
5090   description={\protect\glslonguserfont{\the\glslongtok}},%
5091   first={\protect\glsfirstabbrvfont{\the\glsshorttok}}%
5092   \protect\glxtruserparen{\protect\glslonguserfont{\the\glslongtok}}{\the\glslabeltok}},%

```

```

5093     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
5094     \protect\glsxtruserparen{\protect\glslonguserfont{\the\glslongpltok}}{\the\glslabeltok}},%
5095     plural={\protect\glsabbvfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

5096 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
5097   \glsattribute{\the\glslabeltok}{regular}%
5098   {%
5099     \glssetattribute{\the\glslabeltok}{regular}{false}%
5100   }%
5101   {}%
5102 }%
5103 }%
5104 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

5105 \renewcommand*{\abbrevpluralsuffix}{\glsxtrusersuffix}%
5106 \renewcommand*{\glsabbrvfont[1]}{\glsabbrvuserfont{##1}}%
5107 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
5108 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
5109 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

5110 \renewcommand*{\glsxtrfullformat}[2]{%
5111   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
5112   \ifglxtrinsertinside\else##2\fi
5113   \glsxtruserparen{\glsfirstlongfont{\glsaccesslong{##1}}}{##1}%
5114 }%
5115 \renewcommand*{\glsxtrfullplformat}[2]{%
5116   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
5117   \ifglxtrinsertinside\else##2\fi
5118   \glsxtruserparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}{##1}%
5119 }%
5120 \renewcommand*{\Glsxtrfullformat}[2]{%
5121   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
5122   \ifglxtrinsertinside\else##2\fi
5123   \glsxtruserparen{\glsfirstlongfont{\Glsaccesslong{##1}}}{##1}%
5124 }%
5125 \renewcommand*{\Glsxtrfullplformat}[2]{%
5126   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
5127   \ifglxtrinsertinside\else##2\fi
5128   \glsxtruserparen{\glsfirstlongfont{\Glsaccesslongpl{##1}}}{##1}%
5129 }%
5130 }

```

-long-user-desc

```

5131 \newabbreviationstyle{short-long-user-desc}%
5132 {%
5133   \GlsXtrUseAbbrStyleSetup{short-long-desc}%
5134 }%

```



```

5135 {%
5136   \GlsXtrUseAbbrStyleFmts{short-long-user}%
5137 }

```

1.7 Using Entries in Headings

There are four main problems with using entries in sectioning commands: they can mess with the first use flag if they end up in the table of contents, they can add unwanted numbers to the entry's location list, the label is corrupted if used inside `\MakeUppercase` (which is used by the default headings style) and they need to be expandable for PDF bookmarks. The glossaries package therefore recommends the use of the expandable commands, such as `\glsentryshort`, instead but this doesn't reflect the formatting since it doesn't include `\glsabbrfont`. The commands below are an attempt to get around these problems.

The PDF bookmark issue can easily be fixed with `hyperref's \texorpdfstring` which can simply use the expandable command in the PDF string case. The \TeX string case can now use `\glsxtrshort` with the `noindex` key set, which prevents the unwanted additions to the location list, and the `hyper` key set to `false`, which prevents the problem of nested links. This just leaves one thing left that needs to be dealt with, and that's what to do if the heading style uses `\MakeUppercase`.

Note that glossaries automatically loads `textcase`, so the label can be protected from case change with `textcase's \NoCaseChange`. This means that we don't have a problem provided the page style uses `\MakeTextUppercase`, but the default heading page style uses `\MakeUppercase`.

To get around this, save the original definition of `\markboth` and `\markright` and adjust it so that `\MakeUppercase` is temporarily redefined to `\MakeTextUppercase`. Some packages or classes redefine these commands, so we can't just assume they still have the original kernel definition.

`\markright` Save original definition:

```

5138 \let\@glsxtr@org@markright\markright

```

Redefine (grouping not added in case it interferes with the original code):

```

5139 \renewcommand*{\markright}[1]{%
5140   \glsxtrmarkhook
5141   \@glsxtr@org@markright{#1}%
5142   \glsxtrrestoremarkhook
5143 }

```

`\markboth` Save original definition:

```

5144 \let\@glsxtr@org@markboth\markboth

```

Redefine (grouping not added in case it interferes with the original code):

```

5145 \renewcommand*{\markboth}[2]{%
5146   \glsxtrmarkhook
5147   \@glsxtr@org@markboth{#1}{#2}%
5148   \glsxtrrestoremarkhook
5149 }

```

If this causes a problem provide a simple way of switching back to the original definitions:

sxtrRevertMarks

```
5150 \newcommand*{\glxstrRevertMarks}{%
5151   \let\markright\@glxstr@org@markright
5152   \let\markboth\@glxstr@org@markboth
5153 }
```

\glxstrmarkhook Hook used in new definition of \markboth and \markright to make some changes to apply to the marks:

```
5154 \newcommand*{\glxstrmarkhook}{%
```

Save current definitions:

```
5155   \let\@glxstr@org@MakeUppercase\MakeUppercase
5156   \let\@glxstr@org@glxstrtitleshort\glxstrtitleshort
5157   \let\@glxstr@org@glxstrtitleshortpl\glxstrtitleshortpl
5158   \let\@glxstr@org@Glsxtrtitleshort\Glsxtrtitleshort
5159   \let\@glxstr@org@Glsxtrtitleshortpl\Glsxtrtitleshortpl
5160   \let\@glxstr@org@glxstrtitletext\glxstrtitletext
5161   \let\@glxstr@org@Glsxtrtitletext\Glsxtrtitletext
5162   \let\@glxstr@org@glxstrtitleplural\glxstrtitleplural
5163   \let\@glxstr@org@Glsxtrtitleplural\Glsxtrtitleplural
5164   \let\@glxstr@org@glxstrtitlefirst\glxstrtitlefirst
5165   \let\@glxstr@org@Glsxtrtitlefirst\Glsxtrtitlefirst
5166   \let\@glxstr@org@glxstrtitlefirstplural\glxstrtitlefirstplural
5167   \let\@glxstr@org@Glsxtrtitlefirstplural\Glsxtrtitlefirstplural
5168   \let\@glxstr@org@glxstrtitlelong\glxstrtitlelong
5169   \let\@glxstr@org@glxstrtitlelongpl\glxstrtitlelongpl
5170   \let\@glxstr@org@Glsxtrtitlelong\Glsxtrtitlelong
5171   \let\@glxstr@org@Glsxtrtitlelongpl\Glsxtrtitlelongpl
5172   \let\@glxstr@org@glxstrtitlefull\glxstrtitlefull
5173   \let\@glxstr@org@glxstrtitlefullpl\glxstrtitlefullpl
5174   \let\@glxstr@org@Glsxtrtitlefull\Glsxtrtitlefull
5175   \let\@glxstr@org@Glsxtrtitlefullpl\Glsxtrtitlefullpl
```

New definitions

```
5176   \let\MakeUppercase\MakeTextUppercase
5177   \let\glxstrtitleshort\glxstrheadshort
5178   \let\glxstrtitleshortpl\glxstrheadshortpl
5179   \let\Glsxtrtitleshort\Glsxtrheadshort
5180   \let\Glsxtrtitleshortpl\Glsxtrheadshortpl
5181   \let\glxstrtitletext\glxstrheadtext
5182   \let\Glsxtrtitletext\Glsxtrheadtext
5183   \let\glxstrtitleplural\glxstrheadplural
5184   \let\Glsxtrtitleplural\Glsxtrheadplural
5185   \let\glxstrtitlefirst\glxstrheadfirst
5186   \let\Glsxtrtitlefirst\Glsxtrheadfirst
5187   \let\glxstrtitlefirstplural\glxstrheadfirstplural
5188   \let\Glsxtrtitlefirstplural\Glsxtrheadfirstplural
5189   \let\glxstrtitlelong\glxstrheadlong
```

```

5190 \let\glsxtrtitlelongpl\glsxtrheadlongpl
5191 \let\Glsxtrtitlelong\Glsxtrheadlong
5192 \let\Glsxtrtitlelongpl\Glsxtrheadlongpl
5193 \let\glsxtrtitlefull\glsxtrheadfull
5194 \let\glsxtrtitlefullpl\glsxtrheadfullpl
5195 \let\Glsxtrtitlefull\Glsxtrheadfull
5196 \let\Glsxtrtitlefullpl\Glsxtrheadfullpl
5197 }

```

`restoremarkhook` Hook used in new definition of `\markboth` and `\markright` to restore the modified definitions. (This is in case the original `\markboth` and `\markright` shouldn't be grouped for some reason. There already is some grouping within those original definitions, but some of the code lies outside that grouping, and possibly there's a reason for it.)

```

5198 \newcommand*{\glsxtrrestoremarkhook}{%
5199   \let\MakeUppercase\@glsxtr@org@MakeUppercase
5200   \let\glsxtrtitleshort\@glsxtr@org@glsxtrtitleshort
5201   \let\glsxtrtitleshortpl\@glsxtr@org@glsxtrtitleshortpl
5202   \let\Glsxtrtitleshort\@glsxtr@org@Glsxtrtitleshort
5203   \let\Glsxtrtitleshortpl\@glsxtr@org@Glsxtrtitleshortpl
5204   \let\glsxtrtitletext\@glsxtr@org@glsxtrtitletext
5205   \let\Glsxtrtitletext\@glsxtr@org@Glsxtrtitletext
5206   \let\glsxtrtitleplural\@glsxtr@org@glsxtrtitleplural
5207   \let\Glsxtrtitleplural\@glsxtr@org@Glsxtrtitleplural
5208   \let\glsxtrtitlefirst\@glsxtr@org@glsxtrtitlefirst
5209   \let\Glsxtrtitlefirst\@glsxtr@org@Glsxtrtitlefirst
5210   \let\glsxtrtitlefirstplural\@glsxtr@org@glsxtrtitlefirstplural
5211   \let\Glsxtrtitlefirstplural\@glsxtr@org@Glsxtrtitlefirstplural
5212   \let\glsxtrtitlelong\@glsxtr@org@glsxtrtitlelong
5213   \let\glsxtrtitlelongpl\@glsxtr@org@glsxtrtitlelongpl
5214   \let\Glsxtrtitlelong\@glsxtr@org@Glsxtrtitlelong
5215   \let\Glsxtrtitlelongpl\@glsxtr@org@Glsxtrtitlelongpl
5216   \let\glsxtrtitlefull\@glsxtr@org@glsxtrtitlefull
5217   \let\glsxtrtitlefullpl\@glsxtr@org@glsxtrtitlefullpl
5218   \let\Glsxtrtitlefull\@glsxtr@org@Glsxtrtitlefull
5219   \let\Glsxtrtitlefullpl\@glsxtr@org@Glsxtrtitlefullpl
5220 }

```

Instead of using one document-wide conditional, use `headuc` attribute to determine whether or not to use the all upper case form.

`glsxtrheadshort` Command used to display short form in the page header.

```

5221 \newcommand*{\glsxtrheadshort}[1]{%
5222   \protect\NoCaseChange
5223   {%
5224     \glsifattribute{#1}{headuc}{true}%
5225     {%
5226       \GLSxtrshort[noindex,hyper=false]{#1}[]%
5227     }%
5228   }%

```

```

5229     \glxstrshort[noindex,hyper=false]{#1}[]%
5230 }%
5231 }%
5232 }

```

lsxtrtitleshort Command to display short form of abbreviation in section title and table of contents.

```

5233 \newrobustcmd*{\glxtrtitleshort}[1]{%
5234   \glxstrshort[noindex,hyper=false]{#1}[]%
5235 }

```

sxtrheadshortpl Command used to display plural short form in the page header. If you want the text converted to upper case, this needs to be redefined to use `\GLSxtrshortpl` instead. If you are using a smallcaps style, the default fonts don't provide italic smallcaps.

```

5236 \newcommand*{\glxtrheadshortpl}[1]{%
5237   \protect\NoCaseChange
5238   {%
5239     \gl@ifattribute{#1}{headuc}{true}%
5240     {%
5241       \GLSxtrshortpl[noindex,hyper=false]{#1}[]%
5242     }%
5243     {%
5244       \glxtrshortpl[noindex,hyper=false]{#1}[]%
5245     }%
5246   }%
5247 }

```

xtrtitleshortpl Command to display plural short form of abbreviation in section title and table of contents.

```

5248 \newrobustcmd*{\glxtrtitleshortpl}[1]{%
5249   \glxtrshortpl[noindex,hyper=false]{#1}[]%
5250 }

```

GLsxtrheadshort Command used to display short form in the page header with the first letter converted to upper case.

```

5251 \newcommand*{\GLsxtrheadshort}[1]{%
5252   \protect\NoCaseChange
5253   {%
5254     \gl@ifattribute{#1}{headuc}{true}%
5255     {%
5256       \GLSxtrshort[noindex,hyper=false]{#1}[]%
5257     }%
5258     {%
5259       \GLsxtrshort[noindex,hyper=false]{#1}[]%
5260     }%
5261   }%
5262 }

```

lsxtrtitleshort Command to display short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

5263 \newrobustcmd*{\Glsxtrtitleshort}[1]{%
5264   \Glsxtrshort[noindex,hyper=false]{#1}[]%
5265 }

```

sxtrheadshortpl Command used to display plural short form in the page header with the first letter converted to upper case.

```

5266 \newcommand*{\Glsxtrheadshortpl}[1]{%
5267   \protect\NoCaseChange
5268   {%
5269     \glsifattribute{#1}{headuc}{true}%
5270     {%
5271       \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
5272     }%
5273     {%
5274       \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
5275     }%
5276   }%
5277 }

```

xtrtitleshortpl Command to display plural short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

5278 \newrobustcmd*{\Glsxtrtitleshortpl}[1]{%
5279   \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
5280 }

```

\glsxtrheadtext As above but for the text value.

```

5281 \newcommand*{\glsxtrheadtext}[1]{%
5282   \protect\NoCaseChange
5283   {%
5284     \glsifattribute{#1}{headuc}{true}%
5285     {%
5286       \GLStext[noindex,hyper=false]{#1}[]%
5287     }%
5288     {%
5289       \glstext[noindex,hyper=false]{#1}[]%
5290     }%
5291   }%
5292 }

```

glsxtrtitletext Command to display text value in section title and table of contents.

```

5293 \newrobustcmd*{\glsxtrtitletext}[1]{%
5294   \glstext[noindex,hyper=false]{#1}[]%
5295 }

```

\Glsxtrheadtext First letter converted to upper case

```

5296 \newcommand*{\Glsxtrheadtext}[1]{%
5297   \protect\NoCaseChange
5298   {%

```

```

5299 \glsifattribute{#1}{headuc}{true}%
5300 {%
5301 \GLStext[noindex,hyper=false]{#1}[]%
5302 }%
5303 {%
5304 \GLStext[noindex,hyper=false]{#1}[]%
5305 }%
5306 }%
5307 }

```

Glsxtrtitletext Command to display text value in section title and table of contents with the first letter changed to upper case.

```

5308 \newrobustcmd*{\Glsxtrtitletext}[1]{%
5309 \GLStext[noindex,hyper=false]{#1}[]%
5310 }

```

lsxtrheadplural As above but for the plural value.

```

5311 \newcommand*{\lsxtrheadplural}[1]{%
5312 \protect\NoCaseChange
5313 {%
5314 \glsifattribute{#1}{headuc}{true}%
5315 {%
5316 \GLSplural[noindex,hyper=false]{#1}[]%
5317 }%
5318 {%
5319 \GLSplural[noindex,hyper=false]{#1}[]%
5320 }%
5321 }%
5322 }

```

sxtrtitleplural Command to display plural value in section title and table of contents.

```

5323 \newrobustcmd*{\gsxtrtitleplural}[1]{%
5324 \GLSplural[noindex,hyper=false]{#1}[]%
5325 }

```

lsxtrheadplural Convert first letter to upper case.

```

5326 \newcommand*{\Glsxtrheadplural}[1]{%
5327 \protect\NoCaseChange
5328 {%
5329 \glsifattribute{#1}{headuc}{true}%
5330 {%
5331 \GLSplural[noindex,hyper=false]{#1}[]%
5332 }%
5333 {%
5334 \GLSplural[noindex,hyper=false]{#1}[]%
5335 }%
5336 }%
5337 }

```

`sxtrtitleplural` Command to display plural value in section title and table of contents with the first letter changed to upper case.

```
5338 \newrobustcmd*{\Glsxtrtitleplural}[1]{%
5339   \Glsplural[noindex,hyper=false]{#1}[]%
5340 }
```

`glxtrheadfirst` As above but for the first value.

```
5341 \newcommand*{\glxtrheadfirst}[1]{%
5342   \protect\NoCaseChange
5343   {%
5344     \glsifattribute{#1}{headuc}{true}%
5345     {%
5346       \GLSfirst[noindex,hyper=false]{#1}[]%
5347     }%
5348     {%
5349       \glsfirst[noindex,hyper=false]{#1}[]%
5350     }%
5351   }%
5352 }
```

`lsxtrtitlefirst` Command to display first value in section title and table of contents.

```
5353 \newrobustcmd*{\lsxtrtitlefirst}[1]{%
5354   \glsfirst[noindex,hyper=false]{#1}[]%
5355 }
```

`Glsxtrheadfirst` First letter converted to upper case

```
5356 \newcommand*{\Glsxtrheadfirst}[1]{%
5357   \protect\NoCaseChange
5358   {%
5359     \glsifattribute{#1}{headuc}{true}%
5360     {%
5361       \GLSfirst[noindex,hyper=false]{#1}[]%
5362     }%
5363     {%
5364       \Glsfirst[noindex,hyper=false]{#1}[]%
5365     }%
5366   }%
5367 }
```

`lsxtrtitlefirst` Command to display first value in section title and table of contents with the first letter changed to upper case.

```
5368 \newrobustcmd*{\Glsxtrtitlefirst}[1]{%
5369   \Glsfirst[noindex,hyper=false]{#1}[]%
5370 }
```

`headfirstplural` As above but for the firstplural value.

```
5371 \newcommand*{\glxtrheadfirstplural}[1]{%
5372   \protect\NoCaseChange
```

```

5373 {%
5374   \glsifattribute{#1}{headuc}{true}%
5375   {%
5376     \GLSfirstplural[noindex,hyper=false]{#1}[]%
5377   }%
5378   {%
5379     \glsfirstplural[noindex,hyper=false]{#1}[]%
5380   }%
5381 }%
5382 }

```

`\titlefirstplural` Command to display firstplural value in section title and table of contents.

```

5383 \newrobustcmd*{\glsxtrtitlefirstplural}[1]{%
5384   \glsfirstplural[noindex,hyper=false]{#1}[]%
5385 }

```

`\headfirstplural` First letter converted to upper case

```

5386 \newcommand*{\Glsxtrheadfirstplural}[1]{%
5387   \protect\NoCaseChange
5388   {%
5389     \glsifattribute{#1}{headuc}{true}%
5390     {%
5391       \GLSfirstplural[noindex,hyper=false]{#1}[]%
5392     }%
5393     {%
5394       \Glsfirstplural[noindex,hyper=false]{#1}[]%
5395     }%
5396   }%
5397 }

```

`\titlefirstplural` Command to display first value in section title and table of contents with the first letter changed to upper case.

```

5398 \newrobustcmd*{\Glsxtrtitlefirstplural}[1]{%
5399   \Glsfirstplural[noindex,hyper=false]{#1}[]%
5400 }

```

`\glsxtrheadlong` Command used to display long form in the page header.

```

5401 \newcommand*{\glsxtrheadlong}[1]{%
5402   \protect\NoCaseChange
5403   {%
5404     \glsifattribute{#1}{headuc}{true}%
5405     {%
5406       \GLSxtrlong[noindex,hyper=false]{#1}[]%
5407     }%
5408     {%
5409       \glsxtrlong[noindex,hyper=false]{#1}[]%
5410     }%
5411   }%
5412 }

```


`glsxtrtitlelong` Command to display long form of abbreviation in section title and table of contents.

```
5413 \newrobustcmd*{\glsxtrtitlelong}[1]{%  
5414   \glsxtrlong[noindex,hyper=false]{#1}[]%  
5415 }
```

`lgsxtrheadlongpl` Command used to display plural long form in the page header. If you want the text converted to upper case, this needs to be redefined to use `\GLSxtrlongpl` instead. If you are using a smallcaps style, the default fonts don't provide italic smallcaps.

```
5416 \newcommand*{\glsxtrheadlongpl}[1]{%  
5417   \protect\NoCaseChange  
5418   {%  
5419     \glsifattribute{#1}{headuc}{true}%  
5420     {%  
5421       \GLSxtrlongpl[noindex,hyper=false]{#1}[]%  
5422     }%  
5423     {%  
5424       \glsxtrlongpl[noindex,hyper=false]{#1}[]%  
5425     }%  
5426   }%  
5427 }
```

`sxtrtitlelongpl` Command to display plural long form of abbreviation in section title and table of contents.

```
5428 \newrobustcmd*{\glsxtrtitlelongpl}[1]{%  
5429   \glsxtrlongpl[noindex,hyper=false]{#1}[]%  
5430 }
```

`\Glsxtrheadlong` Command used to display long form in the page header with the first letter converted to upper case.

```
5431 \newcommand*{\Glsxtrheadlong}[1]{%  
5432   \protect\NoCaseChange  
5433   {%  
5434     \glsifattribute{#1}{headuc}{true}%  
5435     {%  
5436       \GLSxtrlong[noindex,hyper=false]{#1}[]%  
5437     }%  
5438     {%  
5439       \Glsxtrlong[noindex,hyper=false]{#1}[]%  
5440     }%  
5441   }%  
5442 }
```

`Glsxtrtitlelong` Command to display long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
5443 \newrobustcmd*{\Glsxtrtitlelong}[1]{%  
5444   \Glsxtrlong[noindex,hyper=false]{#1}[]%  
5445 }
```

`\lsxtrheadlongpl` Command used to display plural long form in the page header with the first letter converted to upper case.

```
5446 \newcommand*{\Glsxtrheadlongpl}[1]{%
5447   \protect\NoCaseChange
5448   {%
5449     \glsifattribute{#1}{headuc}{true}%
5450     {%
5451       \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
5452     }%
5453     {%
5454       \Glsxtrlongpl[noindex,hyper=false]{#1}[]%
5455     }%
5456   }%
5457 }
```

`\sxttrtitlelongpl` Command to display plural long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
5458 \newrobustcmd*{\Glsxtrtitlelongpl}[1]{%
5459   \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
5460 }
```

`\glsxtrheadfull` Command used to display full form in the page header.

```
5461 \newcommand*{\glsxtrheadfull}[1]{%
5462   \protect\NoCaseChange
5463   {%
5464     \glsifattribute{#1}{headuc}{true}%
5465     {%
5466       \GLSxtrfull[noindex,hyper=false]{#1}[]%
5467     }%
5468     {%
5469       \glsxtrfull[noindex,hyper=false]{#1}[]%
5470     }%
5471   }%
5472 }
```

`\glsxtrtitlefull` Command to display full form of abbreviation in section title and table of contents.

```
5473 \newrobustcmd*{\glsxtrtitlefull}[1]{%
5474   \glsxtrfull[noindex,hyper=false]{#1}[]%
5475 }
```

`\lsxtrheadfullpl` Command used to display plural full form in the page header. If you want the text converted to upper case, this needs to be redefined to use `\GLSxtrfullpl` instead. If you are using a smallcaps style, the default fonts don't provide italic smallcaps.

```
5476 \newcommand*{\glsxtrheadfullpl}[1]{%
5477   \protect\NoCaseChange
5478   {%
5479     \glsifattribute{#1}{headuc}{true}%
5480     {%
```

```

5481      \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
5482    }%
5483    {%
5484      \glxtrfullpl[noindex,hyper=false]{#1}[]%
5485    }%
5486  }%
5487 }

```

sxtrtitlefullpl Command to display plural full form of abbreviation in section title and table of contents.

```

5488 \newrobustcmd*{\glxtrtitlefullpl}[1]{%
5489   \glxtrfullpl[noindex,hyper=false]{#1}[]%
5490 }

```

\Glsxtrheadfull Command used to display full form in the page header with the first letter converted to upper case.

```

5491 \newcommand*{\Glsxtrheadfull}[1]{%
5492   \protect\NoCaseChange
5493   {%
5494     \gl@ifattribute{#1}{headuc}{true}%
5495     {%
5496       \GLSxtrfull[noindex,hyper=false]{#1}[]%
5497     }%
5498     {%
5499       \Glsxtrfull[noindex,hyper=false]{#1}[]%
5500     }%
5501   }%
5502 }

```

Glsxtrtitlefull Command to display full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

5503 \newrobustcmd*{\Glsxtrtitlefull}[1]{%
5504   \Glsxtrfull[noindex,hyper=false]{#1}[]%
5505 }

```

lxsxtrheadfullpl Command used to display plural full form in the page header with the first letter converted to upper case.

```

5506 \newcommand*{\Glsxtrheadfullpl}[1]{%
5507   \protect\NoCaseChange
5508   {%
5509     \gl@ifattribute{#1}{headuc}{true}%
5510     {%
5511       \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
5512     }%
5513     {%
5514       \Glsxtrfullpl[noindex,hyper=false]{#1}[]%
5515     }%
5516   }%
5517 }

```

`sxtrtitlefullpl` Command to display plural full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
5518 \newrobustcmd*{\Glsxtrtitlefullpl}[1]{%
5519   \Glsxtrfullpl[noindex,hyper=false]{#1}[]%
5520 }
```

`\glsfmtshort` Provide a way of using the formatted short form in section headings. If `hyperref` has been loaded, use `\texorpdfstring` for convenience in PDF bookmarks.

```
5521 \ifdef\texorpdfstring
5522 {
5523   \newcommand*{\glsfmtshort}[1]{%
5524     \texorpdfstring
5525       {\Glsxtrtitleshort{#1}}%
5526       {\Glsentryshort{#1}}%
5527   }
5528 }
5529 {
5530   \newcommand*{\glsfmtshort}[1]{%
5531     \Glsxtrtitleshort{#1}}
5532 }
```

Similarly for the plural version.

`\glsfmtshortpl`

```
5533 \ifdef\texorpdfstring
5534 {
5535   \newcommand*{\glsfmtshortpl}[1]{%
5536     \texorpdfstring
5537       {\Glsxtrtitleshortpl{#1}}%
5538       {\Glsentryshortpl{#1}}%
5539   }
5540 }
5541 {
5542   \newcommand*{\glsfmtshortpl}[1]{%
5543     \Glsxtrtitleshortpl{#1}}
5544 }
```

The case-changing version isn't suitable for PDF bookmarks, so the PDF alternative uses the non-case-changing version.

`\Glsfmtshort` Singular form (first letter uppercase).

```
5545 \ifdef\texorpdfstring
5546 {
5547   \newcommand*{\Glsfmtshort}[1]{%
5548     \texorpdfstring
5549       {\Glsxtrtitleshort{#1}}%
5550       {\Glsentryshort{#1}}%
5551   }
5552 }
```

```

5553 {
5554   \newcommand*{\Glsfmtshort}[1]{%
5555     \Glsxtrtitleshort{#1}}
5556 }

```

`\Glsfmtshortpl` Plural form (first letter uppercase).

```

5557 \ifdef\teorpdfstring
5558 {
5559   \newcommand*{\Glsfmtshortpl}[1]{%
5560     \teorpdfstring
5561       {\Glsxtrtitleshortpl{#1}}%
5562       {\glstryshortpl{#1}}%
5563   }
5564 }
5565 {
5566   \newcommand*{\Glsfmtshortpl}[1]{%
5567     \Glsxtrtitleshortpl{#1}}
5568 }

```

`\glsfmttext` As above but for the text value.

```

5569 \ifdef\teorpdfstring
5570 {
5571   \newcommand*{\glsfmttext}[1]{%
5572     \teorpdfstring
5573       {\glxtrtitletext{#1}}%
5574       {\glstrytext{#1}}%
5575   }
5576 }
5577 {
5578   \newcommand*{\glsfmttext}[1]{%
5579     \glxtrtitletext{#1}}
5580 }

```

`\Glsfmttext` First letter converted to upper case.

```

5581 \ifdef\teorpdfstring
5582 {
5583   \newcommand*{\Glsfmttext}[1]{%
5584     \teorpdfstring
5585       {\Glsxtrtitletext{#1}}%
5586       {\glstrytext{#1}}%
5587   }
5588 }
5589 {
5590   \newcommand*{\Glsfmttext}[1]{%
5591     \Glsxtrtitletext{#1}}
5592 }

```

`\glsfmtplural` As above but for the plural value.

```

5593 \ifdef\teorpdfstring

```

```

5594 {
5595   \newcommand*{\glsfmtplural}[1]{%
5596     \texorpdfstring
5597     {\glsxtrtitleplural{#1}}%
5598     {\glsentryplural{#1}}%
5599   }
5600 }
5601 {
5602   \newcommand*{\glsfmtplural}[1]{%
5603     \glsxtrtitleplural{#1}}
5604 }

```

`\Glsfmtplural` First letter converted to upper case.

```

5605 \ifdef\texorpdfstring
5606 {
5607   \newcommand*{\Glsfmtplural}[1]{%
5608     \texorpdfstring
5609     {\Glsxtrtitleplural{#1}}%
5610     {\glsentryplural{#1}}%
5611   }
5612 }
5613 {
5614   \newcommand*{\Glsfmtplural}[1]{%
5615     \Glsxtrtitleplural{#1}}
5616 }

```

`\glsfmtfirst` As above but for the first value.

```

5617 \ifdef\texorpdfstring
5618 {
5619   \newcommand*{\glsfmtfirst}[1]{%
5620     \texorpdfstring
5621     {\glsxtrtitlefirst{#1}}%
5622     {\glsentryfirst{#1}}%
5623   }
5624 }
5625 {
5626   \newcommand*{\glsfmtfirst}[1]{%
5627     \glsxtrtitlefirst{#1}}
5628 }

```

`\Glsfmtfirst` First letter converted to upper case.

```

5629 \ifdef\texorpdfstring
5630 {
5631   \newcommand*{\Glsfmtfirst}[1]{%
5632     \texorpdfstring
5633     {\Glsxtrtitlefirst{#1}}%
5634     {\glsentryfirst{#1}}%
5635   }
5636 }

```

```

5637 {
5638   \newcommand*{\Glsfmtfirst}[1]{%
5639     \Glsxtrtitlefirst{#1}}
5640 }

```

`\glsfmtfirstpl` As above but for the firstplural value.

```

5641 \ifdef\teorpdfstring
5642 {
5643   \newcommand*{\glsfmtfirstpl}[1]{%
5644     \teorpdfstring
5645       {\Glsxtrtitlefirstplural{#1}}%
5646       {\Glsentryfirstplural{#1}}%
5647   }
5648 }
5649 {
5650   \newcommand*{\glsfmtfirstpl}[1]{%
5651     \Glsxtrtitlefirstplural{#1}}
5652 }

```

`\Glsfmtfirstpl` First letter converted to upper case.

```

5653 \ifdef\teorpdfstring
5654 {
5655   \newcommand*{\Glsfmtfirstpl}[1]{%
5656     \teorpdfstring
5657       {\Glsxtrtitlefirstplural{#1}}%
5658       {\Glsentryfirstplural{#1}}%
5659   }
5660 }
5661 {
5662   \newcommand*{\Glsfmtfirstpl}[1]{%
5663     \Glsxtrtitlefirstplural{#1}}
5664 }

```

`\glsfmtlong` As above but for the long value.

```

5665 \ifdef\teorpdfstring
5666 {
5667   \newcommand*{\glsfmtlong}[1]{%
5668     \teorpdfstring
5669       {\Glsxtrtitlelong{#1}}%
5670       {\Glsentrylong{#1}}%
5671   }
5672 }
5673 {
5674   \newcommand*{\glsfmtlong}[1]{%
5675     \Glsxtrtitlelong{#1}}
5676 }

```

`\Glsfmtlong` First letter converted to upper case.

```

5677 \ifdef\teorpdfstring

```

```

5678 {
5679   \newcommand*{\Glsfmtlong}[1]{%
5680     \texorpdfstring
5681     {\Glsxtrtitlelong{#1}}%
5682     {\glsentrylong{#1}}%
5683   }
5684 }
5685 {
5686   \newcommand*{\Glsfmtlong}[1]{%
5687     \Glsxtrtitlelong{#1}}
5688 }

```

`\glsfmtlongpl` As above but for the longplural value.

```

5689 \ifdef\texorpdfstring
5690 {
5691   \newcommand*{\glsfmtlongpl}[1]{%
5692     \texorpdfstring
5693     {\glsxtrtitlelongpl{#1}}%
5694     {\glsentrylongpl{#1}}%
5695   }
5696 }
5697 {
5698   \newcommand*{\glsfmtlongpl}[1]{%
5699     \glsxtrtitlelongpl{#1}}
5700 }

```

`\Glsfmtlongpl` First letter converted to upper case.

```

5701 \ifdef\texorpdfstring
5702 {
5703   \newcommand*{\Glsfmtlongpl}[1]{%
5704     \texorpdfstring
5705     {\Glsxtrtitlelongpl{#1}}%
5706     {\glsentrylongpl{#1}}%
5707   }
5708 }
5709 {
5710   \newcommand*{\Glsfmtlongpl}[1]{%
5711     \Glsxtrtitlelongpl{#1}}
5712 }

```

`\glsfmtfull` In-line full format.

```

5713 \ifdef\texorpdfstring
5714 {
5715   \newcommand*{\glsfmtfull}[1]{%
5716     \texorpdfstring
5717     {\glsxtrtitlefull{#1}}%
5718     {\glsxtrinlinefullformat{#1}{}}%
5719   }
5720 }

```



```

5721 {
5722   \newcommand*{\glsfmtfull}[1]{%
5723     \glsxtrtitlefull{#1}}
5724 }

```

`\Glsfmtfull` First letter converted to upper case.

```

5725 \ifdef\txorpdfstring
5726 {
5727   \newcommand*{\Glsfmtfull}[1]{%
5728     \txorpdfstring
5729     {\Glsxtrtitlefull{#1}}%
5730     {\Glsxtrinlinefullformat{#1}-{}}%
5731   }
5732 }
5733 {
5734   \newcommand*{\Glsfmtfull}[1]{%
5735     \Glsxtrtitlefull{#1}}
5736 }

```

`\glsfmtfullpl` In-line full plural format.

```

5737 \ifdef\txorpdfstring
5738 {
5739   \newcommand*{\glsfmtfullpl}[1]{%
5740     \txorpdfstring
5741     {\glsxtrtitlefullpl{#1}}%
5742     {\glsxtrinlinefullplformat{#1}-{}}%
5743   }
5744 }
5745 {
5746   \newcommand*{\glsfmtfullpl}[1]{%
5747     \glsxtrtitlefullpl{#1}}
5748 }

```

`\Glsfmtfullpl` First letter converted to upper case.

```

5749 \ifdef\txorpdfstring
5750 {
5751   \newcommand*{\Glsfmtfullpl}[1]{%
5752     \txorpdfstring
5753     {\Glsxtrtitlefullpl{#1}}%
5754     {\Glsxtrinlinefullplformat{#1}-{}}%
5755   }
5756 }
5757 {
5758   \newcommand*{\Glsfmtfullpl}[1]{%
5759     \Glsxtrtitlefullpl{#1}}
5760 }

```

1.8 Multi-Lingual Support

Add the facility to load language modules, if they are installed, but none are provided with this package.

sariesExtraLang

```
5761 \newcommand*{\RequireGlossariesExtraLang}[1]{%
5762   \@ifundefined{ver@glossariesxtr-#1.ldf}{\input{glossariesxtr-#1.ldf}}{}%
5763 }
```

sariesExtraLang

```
5764 \newcommand*{\ProvidesGlossariesExtraLang}[1]{%
5765   \ProvidesFile{glossariesxtr-#1.ldf}%
5766 }
```

Load any required language modules that are available. This doesn't generate any warning if none are found, since they're not essential. (The only command that really needs defining for the document is `\abbreviationsname`, which can simply be redefined.)

```
5767 \@ifpackageloaded{tracklang}
5768 {%
5769   \AnyTrackedLanguages
5770   {%
5771     \ForEachTrackedDialect{\this@dialect}{%
5772       \IfTrackedLanguageFileExists{\this@dialect}%
5773       {glossariesxtr-}% prefix
5774       {.ldf}%
5775       {%
5776         \RequireGlossariesExtraLang{\CurrentTrackedTag}%
5777       }%
5778     }%
5779   }%
5780 }%
5781 }%
5782 {}%
5783 }
5784 {}
```

Load `glossaries-extra-stylemods` if required.

```
5785 \@glsxtr@redefstyles
```

and set the style:

```
5786 \@glsxtr@do@style
```

2 Style Adjustments (glossaries-extra-stylemods.sty)

This package adjusts the predefined styles so that they include the post description hook. Also, some other minor adjustments may be made to make existing styles more flexible.

2.1 Package Initialisation

First identify package:

```
5787 \NeedsTeXFormat{LaTeX2e}
5788 \ProvidesPackage{glossaries-extra-stylemods}[2016/06/18 v1.06 (NLCT)]
```

Provide package options to automatically load required predefined styles. The simplest method is to just test for the existence of the file `glossary-⟨option⟩.sty`. Packages can't be loaded whilst the options are being processed, so save the list in `\@glxtr@loadstyles`.

`sxtr@loadstyles`

```
5789 \newcommand*{\@glxtr@loadstyles}{%
5790 \DeclareOption*{%
5791   \IfFileExists{glossary-\CurrentOption.sty}%
5792   {\eappto\@glxtr@loadstyles{%
5793     \noexpand\RequirePackage{glossary-\CurrentOption}}}%
5794   {\PackageError{glossaries-extra-styles}%
5795     {Unknown option '\CurrentOption'}{}}
5796 }
```

Process the package options:

```
5797 \ProcessOptions
```

Load the required packages:

```
5798 \@glxtr@loadstyles
```

Adjust the styles that the post description hook added, but only for styles that have already been defined. All the tree styles in `glossary-tree` include the post description hook, so they don't require adjustment. Similarly for `glossary-mcols` which builds on the tree styles.

In case we have an old version of `glossaries`:

`ewglossarystyle`

```
5799 \providecommand{\renewglossarystyle}[2]{%
5800   \ifcsundef{@glstyle@#1}%
5801   {%
5802     \PackageError{glossaries}{Glossary style '#1' isn't already defined}{}%
```

```

5803 }%
5804 {%
5805   \csdef{@glsstyle@#1}{#2}%
5806 }%
5807 }

```

2.2 List-Like Styles

The list-like styles mostly already use the post description hook. Only the listdotted style need modifying.

```

5808 \ifcsdef{@glsstyle@listdotted}
5809 {%
5810   \renewglossarystyle{listdotted}{%
5811     \setglossarystyle{list}%
5812     \renewcommand*{\glossentry}[2]{%
5813       \item[]\makebox[\glslistdottedwidth][l]{%
5814         \glsentryitem{##1}%
5815         \glstarget{##1}{\glossentryname{##1}}%
5816         \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}%
5817         \glossentrydesc{##1}\glspostdescription}%
5818     \renewcommand*{\subglossentry}[3]{%
5819       \item[]\makebox[\glslistdottedwidth][l]{%
5820         \glssubentryitem{##2}%
5821         \glstarget{##2}{\glossentryname{##2}}%
5822         \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}%
5823         \glossentrydesc{##2}\glspostdescription}%
5824   }
5825 }
5826 {}

```

The sublistdotted style doesn't display the description for top-level entries. Sub-level entries use the listdottedstyle.

2.3 Longtable Styles

The three and four column styles require adjustment, but not the two column styles.

```

5827 \ifcsdef{@glsstyle@long3col}
5828 {%
5829   \renewglossarystyle{long3col}{%
5830     \renewenvironment{theglossary}%
5831       {\begin{longtable}{lp{\glsdescwidth}p{\glspagelistwidth}}}%
5832       {\end{longtable}}%
5833     \renewcommand*{\glossaryheader}{}%
5834     \renewcommand*{\glsgroupheading}[1]{}%
5835     \renewcommand{\glossentry}[2]{%
5836       \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
5837       \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline

```

```

5838 }%
5839 \renewcommand{\subglossentry}[3]{%
5840     &
5841     \glssubentryitem{##2}%
5842     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
5843     ##3\tabularnewline
5844 }%
5845 \renewcommand*\{ \glsgroupskip }{%
5846     \ifglsgnoglobskip\else & & \tabularnewline\fi}%
5847 }
5848 }
5849 {}

```

Four column style:

```

5850 \ifcsdef{@glsstyle@long4col}
5851 {%
5852     \renewglossarystyle{long4col}{%
5853         \renewenvironment{theglossary}%
5854             {\begin{longtable}{llll}}%
5855             {\end{longtable}}%
5856         \renewcommand*\{ \glossaryheader }{%
5857             \renewcommand*\{ \glsgroupheading }[1]{%
5858                 \renewcommand{\glossentry}[2]{%
5859                     \glssubentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
5860                     \glossentrydesc{##1}\glspostdescription &
5861                     \glossentrysymbol{##1} &
5862                     ##2\tabularnewline
5863                 }%
5864                 \renewcommand{\subglossentry}[3]{%
5865                     &
5866                     \glssubentryitem{##2}%
5867                     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
5868                     \glossentrysymbol{##2} & ##3\tabularnewline
5869                 }%
5870                 \renewcommand*\{ \glsgroupskip }{%
5871                     \ifglsgnoglobskip\else & & \tabularnewline\fi}%
5872             }
5873         }
5874     }

```

The styles in glossary-longbooktabs are all based on the styles in glossary-long, so no adjustments are needed for that package.

2.4 Long Ragged Styles

The three and four column styles require adjustment, but not the two column styles.

```

5875 \ifcsdef{@glsstyle@longragged3col}
5876 {%
5877     \renewglossarystyle{longragged3col}{%

```

```

5878 \renewenvironment{theglossary}%
5879     {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}%
5880      >{\raggedright}p{\glspagelistwidth}}}%
5881     {\end{longtable}}}%
5882 \renewcommand*{\glossaryheader}{}%
5883 \renewcommand*{\glsgroupheading}[1]{}%
5884 \renewcommand{\glossentry}[2]{%
5885     \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
5886     \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline
5887 }%
5888 \renewcommand{\subglossentry}[3]{%
5889     &
5890     \glssubentryitem{##2}%
5891     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
5892     ##3\tabularnewline
5893 }%
5894 \renewcommand*{\glsgroupskip}{%
5895     \ifglsgnোগroupskip\else & \tabularnewline\fi}%
5896 }
5897 }
5898 {}

```

Four column style:

```

5899 \ifcsdef{@glstyle@altlongragged4col}
5900 {%
5901     \renewglossarystyle{altlongragged4col}{%
5902         \renewenvironment{theglossary}%
5903             {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}l%
5904              >{\raggedright}p{\glspagelistwidth}}}%
5905             {\end{longtable}}}%
5906         \renewcommand*{\glossaryheader}{}%
5907         \renewcommand*{\glsgroupheading}[1]{}%
5908         \renewcommand{\glossentry}[2]{%
5909             \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
5910             \glossentrydesc{##1}\glspostdescription & \glossentrysymbol{##1} &
5911             ##2\tabularnewline
5912         }%
5913         \renewcommand{\subglossentry}[3]{%
5914             &
5915             \glssubentryitem{##2}%
5916             \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
5917             \glossentrysymbol{##2} & ##3\tabularnewline
5918         }%
5919         \renewcommand*{\glsgroupskip}{%
5920             \ifglsgnোগroupskip\else & \tabularnewline\fi}%
5921     }
5922 }
5923 {}

```

2.5 Supertabular Styles

The three and four column styles require adjustment, but not the two column styles.

```
5924 \ifcsdef{@glsstyle@super3col}
5925 {%
5926   \renewglossarystyle{super3col}{%
5927     \renewenvironment{theglossary}%
5928       {\tablehead{}\tabletail{}}%
5929       \begin{supertabular}{lp{\glsdescwidth}p{\glspagelistwidth}}}%
5930     {\end{supertabular}}}%
5931   \renewcommand*{\glossaryheader}{}%
5932   \renewcommand*{\glsgroupheading}[1]{}%
5933   \renewcommand{\glossentry}[2]{%
5934     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
5935     \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline
5936   }%
5937   \renewcommand{\subglossentry}[3]{%
5938     &
5939     \glssubentryitem{##2}%
5940     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
5941     ##3\tabularnewline
5942   }%
5943   \renewcommand*{\glsgroupskip}{%
5944     \ifglsgroupskip\else & \tabularnewline\fi}%
5945 }
5946 }
5947 {}
```

Four column styles:

```
5948 \ifcsdef{@glsstyle@super4col}
5949 {%
5950   \renewglossarystyle{super4col}{%
5951     \renewenvironment{theglossary}%
5952       {\tablehead{}\tabletail{}}%
5953       \begin{supertabular}{llll}}}%
5954     {\end{supertabular}}}%
5955   \renewcommand*{\glossaryheader}{}%
5956   \renewcommand*{\glsgroupheading}[1]{}%
5957   \renewcommand{\glossentry}[2]{%
5958     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
5959     \glossentrydesc{##1}\glspostdescription &
5960     \glossentrysymbol{##1} & ##2\tabularnewline
5961   }%
5962   \renewcommand{\subglossentry}[3]{%
5963     &
5964     \glssubentryitem{##2}%
5965     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
5966     \glossentrysymbol{##2} & ##3\tabularnewline
5967   }%
5968   \renewcommand*{\glsgroupskip}{%
```

```

5969      \ifglsnogroupskip\else & & \tabularnewline\fi}%
5970  }
5971 }
5972 {}

```

2.6 Super Ragged Styles

The three and four column styles require adjustment, but not the two column styles.

```

5973 \ifcsdef{@glsstyle@superragged3col}
5974 {%
5975   \renewglossarystyle{superragged3col}{%
5976     \renewenvironment{theglossary}%
5977       {\tablehead{}\tabletail}%
5978       \begin{supertabular}{1>{\raggedright}p{\glsdescwidth}%
5979         >{\raggedright}p{\glspagelistwidth}}}%
5980       {\end{supertabular}}}%
5981   \renewcommand*{\glossaryheader}{}%
5982   \renewcommand*{\glsgroupheading}[1]{}%
5983   \renewcommand{\glossentry}[2]{%
5984     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
5985     \glossentrydesc{##1}\glspostdescription &
5986     ##2\tabularnewline
5987   }%
5988   \renewcommand{\subglossentry}[3]{%
5989     &
5990     \glssubentryitem{##2}%
5991     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
5992     ##3\tabularnewline
5993   }%
5994   \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else &
5995     &\tabularnewline\fi}%
5996 }
5997 }
5998 {}

```

Four columns:

```

5999 \ifcsdef{@glsstyle@altsuperragged4col}
6000 {%
6001   \renewglossarystyle{altsuperragged4col}{%
6002     \renewenvironment{theglossary}%
6003       {\tablehead{}\tabletail}%
6004       \begin{supertabular}{1>{\raggedright}p{\glsdescwidth}1%
6005         >{\raggedright}p{\glspagelistwidth}}}%
6006       {\end{supertabular}}}%
6007   \renewcommand*{\glossaryheader}{}%
6008   \renewcommand{\glossentry}[2]{%
6009     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
6010     \glossentrydesc{##1}\glspostdescription &
6011     \glossentrysymbol{##1} & ##2\tabularnewline

```



```

6012 }%
6013 \renewcommand{\subglossentry}[3]{%
6014     &
6015     \glssubentryitem{##2}%
6016     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
6017     \glossentrysymbol{##2} & ##3\tabularnewline
6018 }%
6019 \renewcommand*{\glsgroupskip}{\ifglsgnোগroupskip\else & &
6020     &\tabularnewline\fi}%
6021 }
6022 }
6023 {}

```

2.7 Inline Style

The inline style is dealt with slightly differently. The `\glspostdescription` hook is actually in `\glspostinline`, which is called at the end of the glossary. The original definition of `\glspostinline` also includes a space, which is unnecessary. Here, instead of redefining the inline style, just redefine `\glspostinline` and `\glsinlinedescformat`.

```

6024 \ifdef{\@glsstyle@inline}
6025 {%
6026     \renewcommand*{\glspostinline}{.\spacefactor\sfcode'\.}
        Just use \glsxtrpostdescription instead of \glspostdescription.
6027     \renewcommand*{\glsinlinedescformat}[3]{%
6028         \space#1\glsxtrpostdescription}
6029     \renewcommand*{\glsinlinesubdescformat}[3]{%
6030         #1\glsxtrpostdescription}
6031 }
6032 {}

```

2.8 Tree Styles

The `alttree` style is redefined to make it easier to made minor adjustments.

```

6033 \ifdef{\@glsstyle@alttree}
6034 {%

```

Only redefine this style if it's already been defined.

mbolDescLocation

```
\glxtraltrtreeSymbolDescLocation{\<label>}{\<location list>}
```

Layout the symbol, description and location for top-level entries.

```

6035 \newcommand{\glxtraltrtreeSymbolDescLocation}[2]{%
6036     {%
6037         \let\par\glsxtrAltTreePar

```

```

6038      \ifglshassymbol{#1}{(\glossentrysymbol{#1})\space}{}%
6039      \glossentrydesc{#1}\glspostdescription \space #2\par
6040    }%
6041  }

```

`\trAltTreeIndent` Paragraph indent for subsequent paragraphs in multi-paragraph descriptions.

```

6042  \newlength\glxstrAltTreeIndent

```

`\lsxtrAltTreePar` Multi-paragraph descriptions need to keep the hanging indent.

```

6043  \newcommand{\glxstrAltTreePar}{%
6044    @@par
6045    \glxstrAltTreeSetHangIndent
6046    \setlength{\parindent}{\dimexpr\hangindent+\glxstrAltTreeIndent}%
6047  }

```

`\symbolDescLocation` `\glxstralttreeSubSymbolDescLocation{<level>}{<label>}{<location list>}`

Layout the symbol, description and location for sub-entries. Defaults to the same as the top-level.

```

6048  \newcommand{\glxstralttreeSubSymbolDescLocation}[3]{%
6049    \glxstralttreeSymbolDescLocation{#2}{#3}%
6050  }

```

`\trtreeindent` The original style has to keep computing the width of the name at each entry. This register allows the style to compute it once for the top-level at the start of the glossary.

```

6051  \newlength\glxstrtreeindent

```

`\glxstralttreeInit` User-level initialisation for the alttree style.

```

6052  \newcommand*{\glxstralttreeInit}{%
6053    \settowidth{\glxstrtreeindent}{\glstreenamfmt{\glsgetwidestname\space}}%
6054    \glxstrAltTreeIndent=\parindent
6055  }

```

`\eglissetwidest` The original `\glissetwidest` only uses `\def`. This uses `\protected@csedef`.

```

6056  \newcommand*{\eglissetwidest}[2][0]{%
6057    \protected@csedef{@glswidestname\romannumeral#1}{#2}%
6058  }

```

`\xglissetwidest` Like the above but uses `\protected@csxdef`.

```

6059  \newcommand*{\xglissetwidest}[2][0]{%
6060    \protected@csxdef{@glswidestname\romannumeral#1}{#2}%
6061  }

```

`\lsgetwidestname` Provide a user-level macro to obtain the widest top-level name.

```

6062  \newcommand*{\glsggetwidestname}{\@glswidestname}

```

`\getwidestsubname` Provide a user-level macro to obtain the widest sub-entry name.

```
6063 \newcommand*\glsgetwidestsubname}[1]{%
6064   \ifcsundef{@glswidestname\romannumeral#1}%
6065     {\@glswidestname}%
6066     {\csuse{@glswidestname\romannumeral#1}}%
6067 }
```

`\findTopLevelName` CamelCase is easier for long command names. Provide a CamelCase synonym of `\glsfindwidesttoplevelname`

```
6068 \let\glsFindWidestTopLevelName\glsfindwidesttoplevelname
```

`\usedTopLevelName` Like `\glsfindwidesttoplevelname` but has an additional check that the entry has been used. Only useful if the glossaries occur at the end of the document, in which case this command should go at the start of the glossary. Alternatively, place at the end of the document and save for the next run.

```
6069 \newrobustcmd*\glsFindWidestUsedTopLevelName}[1][\@glo@types]{%
6070   \dimen@=0pt\relax
6071   \gls@tmplen=0pt\relax
6072   \foralllglossaries[#1]{\@gls@type}%
6073   {%
6074     \forglsentries[\@gls@type]{\@glo@label}%
6075     {%
6076       \ifglsused{\@glo@label}%
6077       {%
6078         \ifglshasparent{\@glo@label}%
6079         {}%
6080         {%
6081           \settowidth{\dimen@}%
6082             {\glstreenamfmt{\glsentryname{\@glo@label}}}%
6083           \ifdim\dimen@>\gls@tmplen
6084             \gls@tmplen=\dimen@
6085             \eglssetwidest{\glsentryname{\@glo@label}}%
6086           \fi
6087         }%
6088       }%
6089     }%
6090   }%
6091 }%
6092 }
```

`\destUsedAnyName` Like the above but doesn't check the parent key. Useful if all levels should have the same width for the name.

```
6093 \newrobustcmd*\glsFindWidestUsedAnyName}[1][\@glo@types]{%
6094   \dimen@=0pt\relax
6095   \gls@tmplen=0pt\relax
6096   \foralllglossaries[#1]{\@gls@type}%
6097   {%
6098     \forglsentries[\@gls@type]{\@glo@label}%
6099     {%
```

```

6100     \ifglused{\@glo@label}%
6101     {%
6102         \settowidth{\dimen@}%
6103         {\glstreenamfmt{\glentryname{\@glo@label}}}%
6104         \ifdim\dimen@>\gls@tmplen
6105             \gls@tmplen=\dimen@
6106             \eglssetwidest{\glentryname{\@glo@label}}%
6107         \fi
6108     }%
6109     {%
6110 }%
6111 }%
6112 }

```

ndWidestAnyName Like the above but doesn't check if the entry has been used.

```

6113 \newrobustcmd*{\glsFindWidestAnyName}[1][\@glo@types]{%
6114     \dimen@=0pt\relax
6115     \gls@tmplen=0pt\relax
6116     \forallglossaries[#1]{\@gls@type}%
6117     {%
6118         \forglsentries[\@gls@type]{\@glo@label}%
6119         {%
6120             \settowidth{\dimen@}%
6121             {\glstreenamfmt{\glentryname{\@glo@label}}}%
6122             \ifdim\dimen@>\gls@tmplen
6123                 \gls@tmplen=\dimen@
6124                 \eglssetwidest{\glentryname{\@glo@label}}%
6125             \fi
6126         }%
6127     }%
6128 }

```

estUsedLevelTwo This is like `\glsFindWidestUsedTopLevelName` but also sets the first two sub-levels as well. Any entry that has a great-grandparent is ignored.

```

6129 \newrobustcmd*{\glsFindWidestUsedLevelTwo}[1][\@glo@types]{%
6130     \dimen@=0pt\relax
6131     \dimen@i=0pt\relax
6132     \dimen@ii=0pt\relax
6133     \forallglossaries[#1]{\@gls@type}%
6134     {%
6135         \forglsentries[\@gls@type]{\@glo@label}%
6136         {%
6137             \ifglused{\@glo@label}%
6138             {%
6139                 \ifglshasparent{\@glo@label}%
6140                 {%
6141                     \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@label}@parent}}%
6142                     \ifglshasparent{\@glo@parent}%
6143                 }%

```

```

6144         \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@parent}@parent}}}%
6145         \ifglshasparent{\@glo@parent}%
6146         {%
6147         {%
6148             \settowidth{\gls@tmplen}%
6149             {\glstreenamefmt{\glsentryname{\@glo@label}}}%
6150             \ifdim\gls@tmplen>\dimen@ii
6151             \dimen@ii=\gls@tmplen
6152             \eglssetwidest[2]{\glsentryname{\@glo@label}}%
6153         \fi
6154         }%
6155     }%
6156     {%
6157         \settowidth{\gls@tmplen}%
6158         {\glstreenamefmt{\glsentryname{\@glo@label}}}%
6159         \ifdim\gls@tmplen>\dimen@i
6160         \dimen@i=\gls@tmplen
6161         \eglssetwidest[1]{\glsentryname{\@glo@label}}%
6162     \fi
6163     }%
6164     }%
6165     {%
6166         \settowidth{\gls@tmplen}%
6167         {\glstreenamefmt{\glsentryname{\@glo@label}}}%
6168         \ifdim\gls@tmplen>\dimen@
6169         \dimen@=\gls@tmplen
6170         \eglssetwidest{\glsentryname{\@glo@label}}%
6171     \fi
6172     }%
6173     }%
6174     {}%
6175     }%
6176     }%
6177 }

```

dWidestLevelTwo This is like `\glsFindWidestUsedLevelTwo` but doesn't check if the entry has been used.

```

6178 \newrobustcmd*{\glsFindWidestLevelTwo}[1][\@glo@types]{%
6179     \dimen@=0pt\relax
6180     \dimen@i=0pt\relax
6181     \dimen@ii=0pt\relax
6182     \foralllglossaries[#1]{\@gls@type}%
6183     {%
6184         \forglsentries[\@gls@type]{\@glo@label}%
6185         {%
6186             \ifglshasparent{\@glo@label}%
6187             {%
6188                 \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@label}@parent}}%
6189                 \ifglshasparent{\@glo@parent}%
6190                 {%

```

```

6191      \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@parent}@parent}}%
6192      \ifglshasparent{\@glo@parent}%
6193      {}%
6194      {%
6195          \settowidth{\gls@tmplen}%
6196              {\glstreenamefmt{\glsentryname{\@glo@label}}}%
6197          \ifdim\gls@tmplen>\dimen@ii
6198              \dimen@ii=\gls@tmplen
6199              \eglssetwidest[2]{\glsentryname{\@glo@label}}%
6200          \fi
6201      }%
6202  }%
6203  {%
6204      \settowidth{\gls@tmplen}%
6205          {\glstreenamefmt{\glsentryname{\@glo@label}}}%
6206      \ifdim\gls@tmplen>\dimen@i
6207          \dimen@i=\gls@tmplen
6208          \eglssetwidest[1]{\glsentryname{\@glo@label}}%
6209      \fi
6210  }%
6211  }%
6212  {%
6213      \settowidth{\gls@tmplen}%
6214          {\glstreenamefmt{\glsentryname{\@glo@label}}}%
6215      \ifdim\gls@tmplen>\dimen@
6216          \dimen@=\gls@tmplen
6217          \eglssetwidest{\glsentryname{\@glo@label}}%
6218      \fi
6219  }%
6220  }%
6221  }%
6222  }

```

edAnyNameSymbol Like the `\glsFindWidestUsedAnyName` but also measures the symbol. The length of the widest symbol is stored in the second argument should be a length register.

```

6223 \newrobustcmd*{\glsFindWidestUsedAnyNameSymbol}[2][\@glo@types]{%
6224     \dimen@=0pt\relax
6225     \gls@tmplen=0pt\relax
6226     #2=0pt\relax
6227     \forallglossaries[#1]{\@gls@type}%
6228     {%
6229         \forglsentries[\@gls@type]{\@glo@label}%
6230         {%
6231             \ifglused{\@glo@label}%
6232             {%
6233                 \settowidth{\dimen@}%
6234                     {\glstreenamefmt{\glsentryname{\@glo@label}}}%
6235                 \ifdim\dimen@>\gls@tmplen
6236                     \gls@tmplen=\dimen@

```

```

6237         \eglssetwidest{\glsentryname{\@glo@label}}%
6238         \fi
6239         \settowidth{\dimen@}%
6240         {\glsentrysymbol{\@glo@label}}%
6241         \ifdim\dimen@>#2\relax
6242             #2=\dimen@
6243         \fi
6244     }%
6245     {}%
6246 }%
6247 }%
6248 }

```

stAnyNameSymbol Like the above but doesn't check if the entry has been used.

```

6249 \newrobustcmd*{\glsFindWidestAnyNameSymbol}[2][\@glo@types]{%
6250     \dimen@=0pt\relax
6251     \gls@tmplen=0pt\relax
6252     #2=0pt\relax
6253     \forallglossaries[#1]{\@gls@type}%
6254     {%
6255         \forglsentries[\@gls@type]{\@glo@label}%
6256         {%
6257             \settowidth{\dimen@}%
6258             {\glstreenamfmt{\glsentryname{\@glo@label}}}%
6259             \ifdim\dimen@>\gls@tmplen
6260                 \gls@tmplen=\dimen@
6261                 \eglssetwidest{\glsentryname{\@glo@label}}%
6262             \fi
6263             \settowidth{\dimen@}%
6264             {\glsentrysymbol{\@glo@label}}%
6265             \ifdim\dimen@>#2\relax
6266                 #2=\dimen@
6267             \fi
6268         }%
6269     }%
6270 }

```

eSymbolLocation Like the `\glsFindWidestUsedAnyNameSymbol` but also measures the location list. This requires `\glsentrynumberlist`. The length of the widest symbol is stored in the second argument should be a length register. The length of the widest location list is stored in the third argument, which should also be a length register.

```

6271 \newrobustcmd*{\glsFindWidestUsedAnyNameSymbolLocation}[3][\@glo@types]{%
6272     \dimen@=0pt\relax
6273     \gls@tmplen=0pt\relax
6274     #2=0pt\relax
6275     #3=0pt\relax
6276     \forallglossaries[#1]{\@gls@type}%
6277     {%
6278         \forglsentries[\@gls@type]{\@glo@label}%

```

```

6279     {%
6280         \ifglsused{\@glo@label}%
6281     {%
6282         \settowidth{\dimen@}%
6283             {\glstreenamefmt{\glentryname{\@glo@label}}}%
6284         \ifdim\dimen@>\gls@tmplen
6285             \gls@tmplen=\dimen@
6286             \eglssetwidest{\glentryname{\@glo@label}}%
6287         \fi
6288         \settowidth{\dimen@}%
6289             {\glentrysymbol{\@glo@label}}%
6290         \ifdim\dimen@>#2\relax
6291             #2=\dimen@
6292         \fi
6293         \settowidth{\dimen@}%
6294             {\GlsXtrFormatLocationList{\glentrynumberlist{\@glo@label}}}%
6295         \ifdim\dimen@>#3\relax
6296             #3=\dimen@
6297         \fi
6298     }%
6299 }%
6300 }%
6301 }%
6302 }

```

eSymbolLocation Like the `\glsFindWidestUsedAnyNameSymbol` but doesn't check if the entry has been used.

```

6303 \newrobustcmd*{\glsFindWidestAnyNameSymbolLocation}[3][\@glo@types]{%
6304     \dimen@=0pt\relax
6305     \gls@tmplen=0pt\relax
6306     #2=0pt\relax
6307     #3=0pt\relax
6308     \foralllglossaries[#1]{\@gls@type}%
6309     {%
6310         \forglsentries[\@gls@type]{\@glo@label}%
6311         {%
6312             \settowidth{\dimen@}%
6313                 {\glstreenamefmt{\glentryname{\@glo@label}}}%
6314             \ifdim\dimen@>\gls@tmplen
6315                 \gls@tmplen=\dimen@
6316                 \eglssetwidest{\glentryname{\@glo@label}}%
6317             \fi
6318             \settowidth{\dimen@}%
6319                 {\glentrysymbol{\@glo@label}}%
6320             \ifdim\dimen@>#2\relax
6321                 #2=\dimen@
6322             \fi
6323             \settowidth{\dimen@}%
6324                 {\GlsXtrFormatLocationList{\glentrynumberlist{\@glo@label}}}%
6325             \ifdim\dimen@>#3\relax

```



```

6326         #3=\dimen@
6327     \fi
6328 }%
6329 }%
6330 }

```

AnyNameLocation Like the `\glsFindWidestUsedAnyNameSymbolLocation` but doesn't measure the symbol. The length of the widest location list is stored in the second argument, which should be a length register.

```

6331 \newrobustcmd*{\glsFindWidestUsedAnyNameLocation}[2][\@glo@types]{%
6332     \dimen@=0pt\relax
6333     \gls@tmplen=0pt\relax
6334     #2=0pt\relax
6335     \forallglossaries[#1]{\@gls@type}%
6336     {%
6337         \forglstentries[\@gls@type]{\@glo@label}%
6338         {%
6339             \ifglstused{\@glo@label}%
6340             {%
6341                 \settowidth{\dimen@}%
6342                 {\glstreenamfmt{\glstentryname{\@glo@label}}}%
6343                 \ifdim\dimen@>\gls@tmplen
6344                     \gls@tmplen=\dimen@
6345                     \eglissetwidest{\glstentryname{\@glo@label}}%
6346             \fi
6347             \settowidth{\dimen@}%
6348             {\GlsXtrFormatLocationList{\glstentrynumberlist{\@glo@label}}}%
6349             \ifdim\dimen@>#2\relax
6350                 #2=\dimen@
6351             \fi
6352         }%
6353     }%
6354 }%
6355 }%
6356 }

```

AnyNameLocation Like the `\glsFindWidestAnyNameLocation` but doesn't check the **first use** flag.

```

6357 \newrobustcmd*{\glsFindWidestAnyNameLocation}[2][\@glo@types]{%
6358     \dimen@=0pt\relax
6359     \gls@tmplen=0pt\relax
6360     #2=0pt\relax
6361     \forallglossaries[#1]{\@gls@type}%
6362     {%
6363         \forglstentries[\@gls@type]{\@glo@label}%
6364         {%
6365             \settowidth{\dimen@}%
6366             {\glstreenamfmt{\glstentryname{\@glo@label}}}%
6367             \ifdim\dimen@>\gls@tmplen
6368                 \gls@tmplen=\dimen@

```

```

6369         \eglssetwidest{\glstryname{\@glo@label}}}%
6370     \fi
6371     \settowidth{\dimen@}%
6372         {\GlsXtrFormatLocationList{\glstrynumberlist{\@glo@label}}}%
6373     \ifdim\dimen@>\#2\relax
6374         #2=\dimen@
6375     \fi
6376 }%
6377 }%
6378 }

```

computeTreeIndent Compute the value of `\glstreeindent`. Argument is the entry label. (Ignored in default definition, but this command may be redefined to take the particular entry into account.) Note that the sub-levels modify `\glstreeindent`.

```

6379 \newcommand*{\glsxtrComputeTreeIndent}[1]{%
6380     \glstreeindent=\glsxtrtreetopindent\relax
6381 }

```

teTreeSubIndent

```

6382 %\cs{\glsxtrComputeTreeSubIndent}\marg{level}\marg{label}\marg{register}
6383 %\end{macrocode}
6384 % Compute the indent for the sub-entries. The first argument is the
6385 % level, the second argument is the entry label and the third
6386 % argument is the length register used to store the computed indent.
6387 % \begin{macrocode}
6388 \newcommand*{\glsxtrComputeTreeSubIndent}[3]{%
6389     \ifcsundef{@glswidestname\romannumeral#1}%
6390     {%
6391         \settowidth{#3}{\glstreenamefmt{\@glswidestname\space}}%
6392     }%
6393     {%
6394         \settowidth{#3}{\glstreenamefmt{%
6395             \csname @glswidestname\romannumeral#1\endcsname\space}}%
6396     }%
6397 }

```

eeSetHangIndent Set `\hangindent` for top-level entries:

```

6398 \newcommand*{\glsxtrAltTreeSetHangIndent}{\hangindent\glstreeindent}

```

etSubHangIndent Set `\hangindent` for sub-entries:

```

6399 \newcommand*{\glsxtrAltTreeSetSubHangIndent}[1]{\hangindent\glstreeindent}

```

Redefine `alttree`:

```

6400 \renewglossarystyle{alttree}{%
6401     \renewenvironment{theglossary}%
6402     {%
6403         \glsxtralttreeInit
6404         \def\@gls@prevlevel{-1}%

```

```

6405     \mbox{}\par}%
6406     {\par}%
6407 \renewcommand*\glossaryheader{}%
6408 \renewcommand*\glsgroupheading[1]{}%
6409 \renewcommand\glossentry[2]{%
6410     \ifnum\@gls@prevlevel=0\relax
6411     \else
6412         \glxtrComputeTreeIndent{##1}%
6413     \fi
6414     \parindent\glstreeindent
6415     \glxtrAltTreeSetHangIndent
6416     \makebox[Opt][r]%
6417     {%
6418         \glstreenamebox{\glstreeindent}%
6419         {%
6420             \glssentryitem{##1}%
6421             \glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
6422         }%
6423     }%
6424     \glxtralttreeSymbolDescLocation{##1}{##2}%
6425     \def\@gls@prevlevel{0}%
6426 }
6427 \renewcommand\subglossentry[3]{%
6428     \ifnum##1=1\relax
6429         \glssubentryitem{##2}%
6430     \fi
6431     \ifnum\@gls@prevlevel=##1\relax
6432     \else
6433         \glxtrComputeTreeSubIndent{##1}{##2}{\@gls@tmplen}%
6434         \ifnum\@gls@prevlevel<##1\relax
6435             \setlength\glstreeindent\@gls@tmplen
6436             \addtolength\glstreeindent\parindent
6437             \parindent\glstreeindent
6438         \else
6439             \ifnum\@gls@prevlevel=0\relax
6440                 \glxtrComputeTreeIndent{##2}%
6441             \else
6442                 \glxtrComputeTreeSubIndent{\@gls@prevlevel}{##2}{\glstreeindent}%
6443             \fi
6444             \addtolength\parindent{-\glstreeindent}%
6445             \setlength\glstreeindent\parindent
6446         \fi
6447     \fi
6448     \glxtrAltTreeSetSubHangIndent{##1}%
6449     \makebox[Opt][r]{\glstreenamebox{\@gls@tmplen}{%
6450         \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}}}%
6451     \glxtralttreeSubSymbolDescLocation{##1}{##2}{##3}%
6452     \def\@gls@prevlevel{##1}%
6453 }%

```

```

6454 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
6455 }
6456 }%
6457 {%

```

Assume the style isn't required if it hasn't already been defined.

```

6458 }

```

Reset the default style

```

6459 \ifx\@glossary@default@style\relax
6460 \else
6461 \setglossarystyle{\@glxtr@current@style}
6462 \fi

```

Glossary

First use The first time a glossary entry is used (from the start of the document or after a reset) with one of the following commands: `\gls`, `\Gls`, `\GLS`, `\glspl`, `\Glspl`, `\GLSpl` or `\glsdisp`. *see* **first use flag** & **first use text**

First use flag A conditional that determines whether or not the entry has been used according to the rules of **first use**.

First use text The text that is displayed on **first use**, which is governed by the first and first-plural keys of `\newglossaryentry`. (May be overridden by `\glsdisp`.)

`makeindex` An indexing application.

`xindy` An flexible indexing application with multilingual support written in Perl.

Change History

0.1 (2015-11-22)		
General: Initial experimental release	4	
0.2 (2015-11-30)		
\Glsfmtshort: new	164	
\glsfmtshort: new	164	
\Glsfmtshortpl: new	165	
\glsfmtshortpl: new	164	
short: switched inline full form to short (long)	131	
0.3 (2015-12-02)		
\@ACRlong: added redefinition	35	
\@ACRlongpl: added redefinition	36	
\@ACRshort: added redefinition	33	
\@ACRshortpl: added redefinition	34	
\@Acrlong: added redefinition	35	
\@Acrlongpl: added redefinition	36	
\@Acrshort: added redefinition	33	
\@Acrshortpl: added redefinition	34	
\@GLSdesc: added redefinition	28	
\@GLSdescplural@: added redefinition	29	
\@GLSfirst@: added redefinition	26	
\@GLSfirstplural@: added redefinition	27	
\@GLSname@: added redefinition	28	
\@GLSplural@: added redefinition	27	
\@GLSsymbol@: added redefinition	30	
\@GLSsymbolplural@: added redefini- tion	30	
\@GLStext@: added redefinition	26	
\@GLSuseri@: added redefinition	30	
\@GLSuserii@: added redefinition	31	
\@GLSuseriii@: added redefinition	31	
\@GLSuseriv@: added redefinition	31	
\@GLSuseriv@: added redefinition	32	
\@GLSuseriv@: added redefinition	32	
\@acrlong: added redefinition	34	
\@acrlongpl: added redefinition	35	
\@acrshort: added redefinition	32	
\@acrshortpl: added redefinition	33	
\@gls@field@link: added optional ar- gument	25	
\@glsdescplural@: added redefinition	29	
\@glsfirst@: added redefinition	26	
\@glsfirstplural@: added redefinition	27	
\@glsplural@: added redefinition	26	
\@glsymbolplural@: added redefini- tion	30	
\@glsxtr@defaultnoglossarywarning: new	69	
\@glsxtr@field@linkdefs: new	25	
\@glsxtr@insertdots: new	104	
\@print@glossary: added redefinition	66	
\glsabbrvdefaultfont: renamed from \abbrvdefaultfont	108	
\glsaccessdesc: new	74	
\glsaccessdescplural: new	75	
\glsaccessfirst: new	72	
\glsaccessfirstplural: new	72	
\glsaccesslong: new	77	
\glsaccesslong: new	77	
\glsaccessname: new	70	
\glsaccessplural: new	71	
\glsaccessshort: new	76	
\glsaccessshort: new	75	
\glsaccessshortpl: new	76	
\@Glsymbol@: added redefinition	29	
\@Glsymbolplural@: added redefini- tion	30	
\@Glstext@: added redefinition	26	
\@Glsuseri@: added redefinition	30	
\@Glsuserii@: added redefinition	31	
\@Glsuseriii@: added redefinition	31	
\@Glsuseriv@: added redefinition	31	
\@Glsuseriv@: added redefinition	32	
\@Glsuseriv@: added redefinition	32	
\@Glsuseriv@: added redefinition	32	
\@Glsdesc: added redefinition	28	
\@Glsdescplural@: added redefinition	29	
\@Glsfirst@: added redefinition	26	
\@Glsfirstplural@: added redefinition	27	
\@Glsname@: added redefinition	28	
\@Glsplural@: added redefinition	27	

\glsaccesssshortpl: new	76	\@cGLSpl: new	52
\glsaccessssymbol: new	73	\@cGLSpl@: new	52
\glsaccessssymbolplural: new	74	\glsxtr@setentrycountunsetattr:	
\glsaccesstext: new	71	new	47
\glstentryfmt: added check for short ..	24	\cGLS: new	51
\glslongpltok: new	104	\cGLSformat: new	52
\glsshortpltok: new	104	\cGLSpl: new	52
\glsxtrdiscardperiod: added check		\cGLSplformat: new	52
for plural	101	\GlossariesExtraWarningNoLine:	
\GLSxtrlongpl: new	117	new	6
\Glsxtrlongpl: new	117	\glsenableentrycount: new	47
\glsxtrlongpl: new	116	\glsfirstabbrvdefaultfont: new ..	108
\glsxtrNoGlossaryWarning: new	9	\glsfirstlongdefaultfont: new ...	108
\glsxtrpostlinkAddDescOnFirstUse:		\Glsfmtfirst: new	166
new	101	\glsfmtfirst: new	166
\glsxtrpostlinkAddSymbolOnFirstUse:		\Glsfmtfirstpl: new	167
new	101	\glsfmtfirstpl: new	167
\glsxtrpostlinkendsentence: new ..	100	\Glsfmtplural: new	166
\GLSxtrshortpl: new	116	\glsfmtplural: new	165
\Glsxtrshortpl: new	115	\Glsfmtshort: changed to use	
\glsxtrshortpl: new	115	\Glsxtrtitleshort	164
short-long-desc: fixed name to use		renamed from \Glstentryfmtshort ..	164
\glslabeltok	126	\glsfmtshort: changed to use	
\newabbreviation: fixed family name in		\glsxtrtitleshort	164
\setkeys	105	renamed from \glstentryfmtshort ..	164
long-short-desc: fixed name to use		\Glsfmtshortpl: changed to use	
\glslabeltok	125	\Glsxtrtitleshortpl	165
0.4 (2015-12-03)		renamed from \Glstentryfmtshortpl ..	165
\@glsxtr@doabbreviationsdef: added		\glsfmtshortpl: changed to use	
redefinition of \acronymtype	7	\glsxtrtitleshortpl	164
\Glsfmtshort: changed to use		renamed from \glstentryfmtshortpl ..	164
\Glsxtrshort	164	\Glsfmttext: new	165
\glsfmtshort: changed to use		\glsfmttext: new	165
\glsxtrshort	164	\glshasattribute: new	82
\Glsfmtshortpl: changed to use		\glshascategoryattribute: new	81
\glsxtrshortpl	165	\GlsXtrEnableEntryCounting: new ..	46
\glsfmtshortpl: changed to use		\glsxtrifcounttrigger: new	49
\glsxtrshortpl	164	\glsxtrscfont: new	135
\glsxtrifemptyglossary: new	12	\glsxtrscsuffix: new	136
\glsxtrnewnumber: added extra argu-		\glsxtrsmfont: new	139
ment	86	\glsxtrsmsuffix: new	140
\glsxtrnewsymbol: added extra argu-		short-em: new	146
ment	85	short-em-desc: new	147
\MakeAcronymsAbbreviations: set the		short-em-footnote: new	148
default type to \acronymtype	60	short-em-long: new	145
\newterm: fixed name argument	85	short-em-long-desc: new	145
0.5 (2015-12-07)		short-em-postfootnote: new	149
\@cGLS: new	51	short-sc-footnote: new	139
\@cGLS@: new	52	short-sc-postfootnote: new	139

short-sm: new	141	\glxtrheadtext: now uses headuc at-tribute	157
short-sm-desc: new	141	short-long: switch off regular attribute if set	125
short-sm-footnote: new	142	short-long-desc: switch off regular attribute if set	127
short-sm-long: new	140	long-short: switch off regular attribute if set	124
short-sm-long-desc: new	140	long-short-desc: switch off regular attribute if set	125
short-sm-postfootnote: new	142	footnote: switch off regular attribute if set	127
long-noshort-em: new	147	postfootnote: switch off regular attribute if set	129
long-noshort-em-desc: new	148		
long-noshort-sm: new	141	0.5.2 (2015-12-08)	
long-noshort-sm-desc: new	142	\@GLSdesc@: added accessibility support	28
long-short-em: new	143	\@GLSdescplural@: added accessibility support	29
long-short-em-desc: new	143	\@GLSfirst@: added accessibility support	26
long-short-sm: new	140	\@GLSfirstplural@: added accessibility support	27
long-short-sm-desc: new	140	\@GLSname@: added accessibility support	28
0.5.1 (2015-12-02)		\@GLSplural@: added accessibility support	27
\Glsaccesstext: new	71	\@GLSsymbol@: added accessibility support	30
0.5.1 (2015-12-07)		\@GLSsymbolplural@: added accessibility support	30
\@glxtr@doaccsupp: new	9	\@GLStext@: added accessibility support	26
General: removed \ifglxtruseuchead	155	\@GLSdesc@: added accessibility support	28
\Glsaccessdesc: new	74	\@GLSdescplural@: added accessibility support	29
\Glsaccessdescplural: new	75	\@Glsfirst@: added accessibility support	26
\Glsaccessfirst: new	72	\@Glsfirstplural@: added accessibility support	27
\Glsaccessfirstplural: new	73	\@GLSname@: add accessibility support ..	28
\Glsaccessname: new	70	\@Glsplural@: added accessibility support	27
\Glsaccessplural: new	72	\@Glsymbol@: added accessibility support	29
\Glsaccesssymbol: new	73	\@Glsymbolplural@: added accessibility support	30
\Glsaccesssymbolplural: new	74	\@Glstext@: added accessibility support	26
\Glsxtrheadfirst: now uses headuc attribute	159	\@GLSdesc@: added accessibility support	28
\glxtrheadfirst: now uses headuc attribute	159	\@GLSdescplural@: added accessibility support	29
\Glsxtrheadfirstplural: now uses headuc attribute	160		
\glxtrheadfirstplural: now uses headuc attribute	159		
\Glsxtrheadplural: now uses headuc attribute	158		
\glxtrheadplural: now uses headuc attribute	158		
\Glsxtrheadshort: now uses headuc attribute	156		
\glxtrheadshort: now uses headuc attribute	155		
\Glsxtrheadshortpl: now uses headuc attribute	157		
\glxtrheadshortpl: now uses headuc attribute	156		
\Glsxtrheadtext: now uses headuc attribute	157		

\@glsfirst@: added accessibility support	26	\mfu@checkword@do: added	98
\@glsfirstplural@: added accessibility support	27	\setabbreviationstyle: added check for post-definition style switch	120
\@glsname@: added accessibility support	28	0.5.3 (2015-12-09)	
\@glsplural@: added accessibility support	26	\@glxtr@autoindex@at: new	95
\@glssymbol@: added accessibility support	29	\@glxtr@autoindex@encap: new	95
\@glssymbolplural@: added accessibility support	30	\@glxtr@autoindex@esc: new	95
\@glstext@: added accessibility support	26	\@glxtr@autoindex@level: new	95
\@glxtr@activate@initialtagging: new	99	\@glxtr@autoindex@setname: new ..	93
\@glxtr@do@titlecaps@warn: new ..	99	\@glxtr@doabbreviationsdef: new ..	7
\@glxtr@tag: new	99	General: removed \GlsXtrNoGlsWarningNoAutoMakeMain	68
General: fixed typo in glossaries-accsupp and tidied up code to use just one \@ifpackageloaded	70	\glsdescwidth: added	21
removed \glxtrabbrvfmt	118	\glspagelistwidth: added	21
\glossaryentrynumbers: added	22	\glxtrdoautoindexname: new	93
\Glossentrydesc: added	97	\glxtrpostnamehook: new	92
\Glossentryname: added	91	\if@glxtr@format@override: new ..	92
\Glossentrysymbol: added	97	\ProvidesGlossariesExtraLang: new	170
\glossentrysymbol: added	97	\RequireGlossariesExtraLang: new	170
\GLSaccessdesc: new	75, 79	0.5.4 (2015-12-15)	
\GLSaccessdescplural: new	75, 80	\@newglossaryentry@defunitcounters: new	53
\GLSaccessfirst: new	72, 78	\@GLSxtr@p@acrlong@: new	45
\GLSaccessfirstplural: new	73, 79	\@GLSxtr@p@acrlongpl@: new	45
\GLSaccesslong: new	77, 80	\@GLSxtr@p@acrshort@: new	45
\GLSaccesslongpl: new	77, 81	\@GLSxtr@p@acrshortpl@: new	45
\Glsaccesslongpl: new	77	\@GLSxtr@p@long@: new	44
\glaccesslongpl: new	77	\@GLSxtr@p@longpl@: new	45
\GLSaccessname: new	71, 78	\@GLSxtr@p@plural@: new	43
\GLSaccessplural: new	72, 78	\@GLSxtr@p@short@: new	44
\GLSaccesssshort: new	76, 80	\@GLSxtr@p@shortpl@: new	44
\GLSaccesssshortpl: new	76, 80	\@GLSxtr@p@text@: new	43
\GLSaccessssymbol: new	73, 79	\@GlsXtrEnableOnTheFly: new	17
\GLSaccessssymbolplural: new ...	74, 79	\@Glsxtr: new	18
\GLSaccessstext: new	71, 78	\@Glsxtr@p@acrlong@: new	45
\glsentryfmt: moved \glsetabbrvfmt from \glxtrabbrvfmt to here	24	\@Glsxtr@p@acrlongpl@: new	45
\GlsXtrEnableInitialTagging: new ..	97	\@Glsxtr@p@acrshort@: new	45
\glxtrfieldtitlecase: new	86	\@Glsxtr@p@acrshortpl@: new	45
\GlsXtrFormatLocationList: new ...	22	\@Glsxtr@p@long@: new	44
\glxtrnewabbrevpresetkeyhook: new	106	\@Glsxtr@p@longpl@: new	44
\glxtrtagfont: new	99	\@Glsxtr@p@plural@: new	43
\KV@printgloss@nonumberlist: added	24	\@Glsxtr@p@short@: new	43
		\@Glsxtr@p@shortpl@: new	44
		\@Glsxtr@p@text@: new	43
		\@Glsxtrpl: new	19
		\@alt@glshyp@opt: new	41
		\@glshyp@opt: new	41
		\@glshyp@opt@char: new	41
		\@glshyp@opt@keys: new	41

\@gls@increment@currunitcount: new	54	\GlsXtrEnableOnTheFly: new	17
\@gls@local@increment@currunitcount: new	54	\Glsxtrpl: new	19
\@gls@setdefault@glslink@opts: new	39	\glxtrpl: new	18
\@glxtr: new	18	\glxtrpostlocalreset: new	46
\@glxtr@addunitcounter: new	53	\glxtrpostlocalunset: new	46
\@glxtr@currunitcount: new	55	\glxtrpostreset: new	46
\@glxtr@ifunitcounter: new	53	\glxtrpostunset: new	46
\@glxtr@p@acrlong@: new	45	\glxtrprotectlinks: new	42
\@glxtr@p@acrlongpl@: new	45	\GlsXtrSetAltModifier: new	41
\@glxtr@p@acrshort@: new	45	\GlsXtrSetDefaultGlsOpts: new	39
\@glxtr@p@acrshortpl@: new	45	\glxtrstarflywarn: new	17
\@glxtr@p@long@: new	44	\GlsXtrWarning: new	19
\@glxtr@p@longpl@: new	44	\MakeAcronymsAbbreviations: now	
\@glxtr@p@plural@: new	43	disables \setacronymstyle	60
\@glxtr@p@short@: new	43	1.0 (2016-01-24)	
\@glxtr@p@shortpl@: new	44	\@glxtr@autoindexcrossrefs: new ..	6
\@glxtr@p@text@: new	43	\@glxtr@idx@displaynumberlist: new	64
\@glxtr@prevunitcount: new	55	\@glxtr@idx@entrynumberlist: new	65
\@glxtr@setentryunitcountunsetattr: new	58	\@glxtr@noidx@displaynumberlist: new	64
\@glxtr@unitcountlist: new	53	\@glxtr@noidx@entrynumberlist: new	65
\@glxtrpl: new	18	\@glxtr@noidx@numberlistloop: new	65
\@newglossaryentryposthook: added empty see value if not set and added 'see' to field key map	14	\@glxtr@reg@glosslist: new	61
\@sGlsXtrEnableOnTheFly: new	17	\makeglossaries: new	62
\cGlsformat: added	52	1.01 (2016-02-02)	
\cglformat: added	52	\glxtrdiscardperiod: added check for first use	101
\cGlsplformat: added	53	short-desc: fixed typo in \glxtrinlinefullformat and added missing second argument	132
\cglplformat: added	52	1.02 (2016-04-25)	
\glstdisablehyper: added	41	\@glxtr@current@style: new	20
\glstdohyperlink: added	41	\Glsfmtfull: new	169
\glstdonohyperlink: added	42	\Glsfmtfull: new	168
\glsenableentryunitcount: new	55	\Glsfmtfullpl: new	169
\glshasattribute: added check for en- try's existence	82	\Glsfmtfullpl: new	169
\glusifattribute: added check for en- try's existence	83	\Glsfmtlong: new	167
\glspostlinkhook: added existence check	100	\Glsfmtlong: new	167
\Glsxtr: new	18	\Glsfmtlongpl: new	168
\glxtr: new	18	\Glsfmtlongpl: new	168
\glxtrcat: new	18	\Glsxtrheadfull: new	163
\glxtrdowrglossaryhook: new	40	\glxtrheadfull: new	162
\GlsXtrEnableEntryUnitCounting: new	58	\Glsxtrheadfullpl: new	163
		\glxtrheadfullpl: new	162
		\Glsxtrheadlong: new	161
		\glxtrheadlong: new	160
		\Glsxtrheadlongpl: new	162

\glxtrheadlongpl: new	161	\@GLSuserii@: set regular format	31
\Glsxtrtitlefull: new	163	\@GLSuseriii@: set regular format	31
\glxtrtitlefull: new	162	\@GLSuseriv@: set regular format	31
\Glsxtrtitlefullpl: new	164	\@GLSuserv@: set regular format	32
\glxtrtitlefullpl: new	163	\@GLSuservi@: set regular format	32
\Glsxtrtitlelong: new	161	\@Glsdesc@: set abbreviation and regular format	28
\glxtrtitlelong: new	161	\@Glsdescplural@: set abbreviation and regular format	29
\Glsxtrtitlelongpl: new	162	\@Glsfirst@: set abbreviation and regu- lar format	26
\glxtrtitlelongpl: new	161	\@Glsfirstplural@: set abbreviation and regular format	27
\ifglxtrininsertinside: new	123	\@Glsname@: set abbreviation and regular format	28
postfootnote: added redef of \glxtrsetupfulldefs	129	\@Glsplural@: set abbreviation and reg- ular format	27
stylemods: new	10	\@Glsymbol@: set regular format	29
1.03 (2016-04-27)		\@Glsymbolplural@: set regular format	30
\@GLSfirstplural@: bug fix: misspelt cs name	27	\@Glstext@: set abbreviation and regular format	26
\@GLSplural@: fixed bug \@GLSplural@ should be redefined not \@GLSplural	27	\@Glsuseri@: set regular format	30
\@Glsfirstplural@: bug fix: misspelt cs name	27	\@Glsuserii@: set regular format	31
\@Glsplural@: fixed bug \@Glsplural@ should be redefined not \@Glsplural	27	\@Glsuseriii@: set regular format	31
\@glspplural@: fixed bug \@glspplural@ should be redefined not \@glspplural	26	\@Glsuseriv@: set regular format	31
\glxtrtitlelongpl: bug fix: changed \glxtrlong to \glxtrlongpl ..	161	\@Glsuserv@: set regular format	32
\glxtrtitleshortpl: bug fix: changed \glxtrshort to \glxtrshortpl	156	\@Glsuservi@: set regular format	32
1.04 (2015-04-30)		\@gls@preglossaryhook: added check for entry's existence	99
short-em-footnote: renamed from "footnote-em"	148	\@glsdesc@: set abbreviation and regular format	28
1.04 (2016-05-02)		\@glsdescplural@: set abbreviation and regular format	29
\@glxtrpostloctag: new	23	\@glsfirst@: set abbreviation and regu- lar format	26
\@GLSdesc@: set abbreviation and regular format	28	\@glsfirstplural@: set abbreviation and regular format	27
\@GLSdescplural@: set abbreviation and regular format	29	\@glsname@: set abbreviation and regular format	28
\@GLSfirst@: set abbreviation format ..	26	\@glsplural@: set abbreviation and reg- ular format	26
\@GLSfirstplural@: set abbreviation and regular format	27	\@glssymbol@: set regular format	29
\@GLSname@: set abbreviation and regular format	28	\@glssymbolplural@: set regular format	30
\@GLSplural@: set abbreviation and reg- ular format	27	\@glstext@: set abbreviation and regular format	26
\@GLSsymbol@: set regular format	30	\@glxtr@deprecated@abbrstyle: new	122
\@GLSsymbolplural@: set regular format	30	\@glxtr@do@style: new	11
\@GLStext@: set abbreviation and regular format	26	\@glxtr@doloctag: new	24
\@GLSuseri@: set regular format	30		

\@glstr@idx@entrynumberlist:		
switched from \let to \newcommand	65	
\@glstr@pagetag: new	23	
\@glstr@pagetag: new	23	
\@glstr@preloctag: new	24	
\@glstr@postloctag: new	23	
\@glstr@preloctag: new	23	
\glossentrydesc: added glossdescfont		
attribute check	87	
\Glossentryname: added glossnamefont		
attribute check	91	
\glossentryname: added glossnamefont		
attribute check	88	
moved post name hook inside condi-		
tion	91	
\glsabbrvemfont: new	143	
\glsabbrvuserfont: new	150	
\glsfirstabbrvemfont: new	143	
\glsfirstabbrvuserfont: new	150	
\glsfirstlongemfont: new	143	
\glsfirstlonguserfont: new	150	
\glsifnotregularcategory: new	83	
\glslongdefaultfont: new	108	
\glslongemfont: new	143	
\glslongfont: new	108	
\glslonguserfont: new	150	
\glstrassignfieldfont: new	25	
\GlsXtrEnablePreLocationTag: new	22	
\glstrfirstscfont: new	136	
\glstrfirstsmfont: new	139	
\glstrlongshortdescsort: new	125	
\glstrpostnamehook: added category		
check	92	
\glstrregularfont: new	24	
\glstruserfield: new	149	
\glstruserparen: new	149	
\glstrusersuffix: new	150	
\GlsXtrWarnDeprecatedAbbrStyle:		
new	122	
short-em-long-em: new	145	
short-em-long-em-desc: new	146	
short-em-nolong: new	146	
short-em-nolong-desc: new	147	
short-em-postfootnote: renamed from		
“postfootnote-em”	149	
short-footnote: new	129	
short-long-user: new	151	
short-long-user-desc: new	152	
short-nolong: new	132	
short-nolong-desc: new	133	
short-postfootnote: new	130	
short-sc-footnote: renamed from		
“footnote-sc”	139	
short-sc-nolong: new	137	
short-sc-nolong-desc: new	138	
short-sc-postfootnote: renamed from		
“postfootnote-sc”	139	
short-sm-footnote: renamed from		
“footnote-sm”	142	
short-sm-nolong: new	141	
short-sm-nolong-desc: new	141	
short-sm-postfootnote: renamed from		
“postfootnote-sm”	142	
\letabbreviationstyle: new	122	
\newabbreviationstyle: bug fix: cor-		
rected test for existence	121	
long-em-noshort-em: new	147	
long-em-noshort-em-desc: new	148	
long-em-short-em: new	144	
long-em-short-em-desc: new	144	
long-noshort: new	135	
long-noshort-desc: new	135	
long-noshort-em: renamed from “long-		
em”	147	
long-noshort-em-desc: renamed from		
“long-desc-em”	148	
long-noshort-sc: renamed from “long-		
sc”	138	
long-noshort-sc-desc: renamed from		
“long-desc-sc”	138	
long-noshort-sm: renamed from “long-		
sm”	141	
long-noshort-sm-desc: renamed from		
\long-desc-sm	142	
long-short-user: new	150	
long-short-user-desc: new	151	
\renewabbreviationstyle: new	122	
style: new	11	
1.05 (2016-06-10)		
\eglssetwidest: new	178	
\glsFindWidestAnyName: new	180	
\glsFindWidestAnyNameLocation:		
new	185	
\glsFindWidestAnyNameSymbol: new	183	
\glsFindWidestAnyNameSymbolLocation:		
new	184	
\glsFindWidestLevelTwo: new	181	
\glsFindWidestUsedAnyName: new	179	

\glsFindWidestUsedAnyNameLocation:	new	178
new	185	
\glsFindWidestUsedAnyNameSymbol:	new	177
new	182	
\glsFindWidestUsedAnyNameSymbolLocation:	\glstrComputeTreeIndent: new ...	186
new	183	
\glsFindWidestUsedLevelTwo: new .	\glstrComputeTreeSubIndent: new	186
180	\glstrtreetopindent: new	178
\glsFindWidestUsedTopLevelName:	short-em-long: fixed incorrect font used	
new	by long form	145
179	\xglsssetwidest: new	178
\glsfirstlongfootnotefont: new ..	1.06 (2016-06-18)	
127	\@glsdoifexistsorwarn: new	6
\glsgetwidestname: new	\@glstr@docdefval: new	5
178	\@glstr@usesee: new	14
\glsgetwidestsubname: new	General: disabled docdef key at the start of	
179	the document	12
\glslongfootnotefont: new	docdef option changed to choice	5
127	\glstr@usesee: new	14
\glstrAltTreeIndent: new	\glstrusesee: new	14
178	\glstruseseeformat: new	14
\glstrAltTreeInit: new	\if@glstrdocdefrestricted: new ...	6
178		
\glstrAltTreePar: new		
178		
\glstrAltTreeSetHangIndent: new		
186		
\glstrAltTreeSetSubHangIndent:		
new		
186		
\glstralttreeSubSymbolDescLocation:		

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	
\.	100, 101, 177
\@@cGLS@	48, 56, 57
\@@cGLSpl@	48, 57
\@@cGLspl@	48, 56
\@@cglS@	48, 56
\@@cglSpl@	48, 50, 56
\@@glo@assign@sortkey	64
\@@glo@no@assign@sortkey	64
\@@glslocalreset	46
\@@glslocalunset	46
\@@glsreset	46
\@@glsunset	46
\@@glsxtr@autoindex@escspch	95, 96
\@@glsxtr@checkspch	94, 96, 97
\@@glsxtr@disabledflycommand	20
\@@glsxtr@postloctag	22
\@@glsxtr@preloctag	22, 23
\@@newglossaryentry@defcounters	47
\@@newglossaryentry@defunitcounters	55
\@@par	178
\@ACRlong	43
\@ACRlongpl	43
\@ACRshort	42
\@ACRshortpl	43
\@Acrlong	43
\@Acrlongpl	43
\@Acrshort	42
\@Acrshortpl	43
\@GLS@	42, 51, 52
\@GLSdesc@	29
\@GLSpl@	42, 51, 52
\@GLSplural@	43
\@GLSsymbol@	30
\@GLStext@	43
\@GLSxtr@full	110
\@GLSxtr@fullpl	111
\@GLSxtr@p@acrlong@	43
\@GLSxtr@p@acrlongpl@	43
\@GLSxtr@p@acrshort@	42
\@GLSxtr@p@acrshortpl@	43
\@GLSxtr@p@long@	42
\@GLSxtr@p@longpl@	42
\@GLSxtr@p@plural@	42
\@GLSxtr@p@short@	42
\@GLSxtr@p@shortpl@	42
\@GLSxtr@p@text@	42
\@GLSxtrlong	42, 114
\@GLSxtrlongpl	42, 117

<code>\@Glsxtrpl</code>	19	<code>\@gls@automake</code>	64
<code>\@Glsxtrshort</code>	42, 112	<code>\@gls@checkedmkidx</code>	94, 96
<code>\@Glsxtrshortpl</code>	42, 115	<code>\@gls@checkmkidxchars</code>	93
<code>\@acrlong</code>	43	<code>\@gls@codepage</code>	66
<code>\@acrlongpl</code>	43	<code>\@gls@declareoption</code>	4
<code>\@acrshort</code>	42	<code>\@gls@doautomake</code>	64
<code>\@acrshortpl</code>	42	<code>\@gls@encapchar</code>	94
<code>\@alt@gls@hyp@opt</code>	41	<code>\@gls@entry@count</code>	48, 49
<code>\@auxout</code>	23, 24, 49, 57, 62, 66	<code>\@gls@entry@field</code>	37, 48
<code>\@cGLS</code>	51	<code>\@gls@entry@unitcount</code>	57
<code>\@cGLS@</code>	48, 51, 56, 57	<code>\@gls@field@font</code>	25–32
<code>\@cGLSpl</code>	52	<code>\@gls@field@link</code>	26–32, 37, 38
<code>\@cGLSpl@</code>	48, 52, 57	<code>\@gls@hyp@opt</code>	37, 38, 41, 51, 52, 108–117
<code>\@cGLspl@</code>	48, 56	<code>\@gls@hyp@opt@cs</code>	41
<code>\@cgls@</code>	48, 51, 56	<code>\@gls@increment@currcount</code>	48
<code>\@cglspl@</code>	48, 56	<code>\@gls@increment@currunitcount</code>	56
<code>\@disable@onlypremakeg</code>	62	<code>\@gls@keymap</code>	14, 36
<code>\@do@auxoutstuff</code>	66	<code>\@gls@label</code>	40, 120
<code>\@do@newglossaryentry</code>	59, 60, 106	<code>\@gls@levelchar</code>	94
<code>\@empty</code>	25, 32–36, 94, 109–118	<code>\@gls@link</code>	25, 32–36, 109–118
<code>\@end@glxstr@addunused</code>	15	<code>\@gls@link@checkfirsthyper</code>	61
<code>\@end@glxstr@usesee</code>	14	<code>\@gls@link@nocheckfirsthyper</code>	25, 32–36, 109–118
<code>\@endfortrue</code>	121	<code>\@gls@local@increment@currcount</code>	48
<code>\@firstofone</code>	25, 87, 88, 93, 98	<code>\@gls@local@increment@currunitcount</code>	56
<code>\@firstofthree</code>	25, 32–35, 41, 109, 110, 112, 113, 115, 117	<code>\@gls@loclist</code>	64, 65
<code>\@firstoftwo</code>	26–30, 33–36, 38, 41, 102, 103, 109–111, 115–118	<code>\@gls@longpl</code>	104–106
<code>\@for</code>	10, 15, 47, 58, 62, 64, 86, 98	<code>\@gls@noidx@nosanitizesort</code>	63
<code>\@glo@assign@sortkey</code>	63	<code>\@gls@noidx@sanitizesort</code>	63
<code>\@glo@category</code>	53	<code>\@gls@noidx@loclist@finalsep</code>	64
<code>\@glo@countunit</code>	53	<code>\@gls@noidx@loclist@prev</code>	64
<code>\@glo@default@sorttype</code>	64	<code>\@gls@noidx@loclist@sep</code>	64
<code>\@glo@label</code>	14, 15, 37, 179–186	<code>\@gls@org@glsnoidx@displayloc</code>	65
<code>\@glo@name</code>	93	<code>\@gls@org@glsseeformat</code>	65
<code>\@glo@parent</code>	180–182	<code>\@gls@preglossaryhook</code>	98
<code>\@glo@see</code>	14, 15	<code>\@gls@prevlevel</code>	186, 187
<code>\@glo@sort</code>	93	<code>\@gls@quotechar</code>	94
<code>\@glo@sorttype</code>	64	<code>\@gls@reference</code>	16, 62
<code>\@glo@thisvalue</code>	149	<code>\@gls@setdefault@glslink@opts</code>	39
<code>\@glo@tmp</code>	37	<code>\@gls@short</code>	105
<code>\@glo@type</code>	15, 59, 62, 63, 66, 69, 70	<code>\@gls@shortpl</code>	104–106
<code>\@glo@types</code>	84, 179–185	<code>\@gls@tmpb</code>	96
<code>\@glossary@default@style</code>	20, 21, 188	<code>\@gls@type</code>	64, 120, 179–185
<code>\@gls@</code>	42, 50, 51	<code>\@gls@write@entrycounts</code>	48
<code>\@gls@actualchar</code>	94	<code>\@gls@write@entryunitcounts</code>	57
<code>\@gls@alt@hyp@opt</code>	41	<code>\@gls@write@entryunitcounts@do</code>	58
<code>\@gls@alt@hyp@opt@char</code>	41	<code>\@gls@abbrv@current@abbreviation</code>	105, 118
<code>\@gls@alt@hyp@opt@keys</code>	41	<code>\@gls@acronymlists</code>	59
		<code>\@gls@doifexistsorwarn</code>	6, 88, 90, 91

<code>\@glsentry</code>	49, 57, 58	<code>\@glsxtr@enabletagging</code>	97
<code>\@glslink</code>	42	<code>\@glsxtr@end@</code>	17
<code>\@glsnumberformat</code>	92, 93	<code>\@glsxtr@enddescspch</code>	94–97
<code>\@glsorder</code>	62	<code>\@glsxtr@entrycount@org@localreset</code> ..	48
<code>\@glspl@</code>	42, 50, 51	<code>\@glsxtr@entrycount@org@localunset</code> ..	48
<code>\@glsplural@</code>	43	<code>\@glsxtr@entrycount@org@reset</code>	48
<code>\@gls punc@token</code>	102	<code>\@glsxtr@entrycount@org@unset</code>	48
<code>\@glsstyle@alttree</code>	177	<code>\@glsxtr@entryunitcount@org@localreset</code>	56
<code>\@glsstyle@inline</code>	177	<code>\@glsxtr@entryunitcount@org@localunset</code>	56
<code>\@glsstyle@listdotted</code>	172	<code>\@glsxtr@entryunitcount@org@reset</code> ..	56
<code>\@glstarget</code>	42	<code>\@glsxtr@entryunitcount@org@unset</code> ..	56
<code>\@glstext@</code>	43	<code>\@glsxtr@field@linkdefs</code>	25
<code>\@glswidestname</code>	178, 179, 186	<code>\@glsxtr@format@overridefalse</code>	92
<code>\@glsxtr</code>	18, 19	<code>\@glsxtr@format@overridetrue</code>	93
<code>\@glsxtr@abbreviationsdef</code>	7, 11	<code>\@glsxtr@foundinlist</code>	103
<code>\@glsxtr@activate@initialtagging</code> ..	98, 99	<code>\@glsxtr@full</code>	108
<code>\@glsxtr@addunitcounter</code>	53	<code>\@glsxtr@fullpl</code>	110
<code>\@glsxtr@addunusedxrefs</code>	15	<code>\@glsxtr@glossdescfont</code>	87, 88
<code>\@glsxtr@attrval</code>	87–93	<code>\@glsxtr@glossnamefont</code>	89–92
<code>\@glsxtr@autoindex@at</code>	93–95	<code>\@glsxtr@gobbleto@endescspch</code>	96
<code>\@glsxtr@autoindex@doextra@esc</code>	93	<code>\@glsxtr@idx@displaynumberlist</code>	62
<code>\@glsxtr@autoindex@encap</code>	93–95	<code>\@glsxtr@idx@entrynumberlist</code>	63
<code>\@glsxtr@autoindex@esc</code>	94, 96, 97	<code>\@glsxtr@ifcsstart</code>	17
<code>\@glsxtr@autoindex@escat</code>	94, 95	<code>\@glsxtr@ifpunctoken</code>	103
<code>\@glsxtr@autoindex@escencap</code>	94, 95	<code>\@glsxtr@ifunitcounter</code>	53
<code>\@glsxtr@autoindex@esclevel</code>	94, 95	<code>\@glsxtr@insert@dots</code>	104
<code>\@glsxtr@autoindex@escquote</code>	94, 96	<code>\@glsxtr@insert@dots@next</code>	104, 105
<code>\@glsxtr@autoindex@level</code>	94, 95	<code>\@glsxtr@insertdots</code>	105
<code>\@glsxtr@autoindex@setname</code>	93	<code>\@glsxtr@label</code>	15, 86
<code>\@glsxtr@autoindex@crossrefs</code>	6, 14	<code>\@glsxtr@loadstyles</code>	171
<code>\@glsxtr@cat</code>	47, 58, 59, 98	<code>\@glsxtr@noidx@displaynumberlist</code> ...	63
<code>\@glsxtr@csname</code>	54, 56	<code>\@glsxtr@noidx@entrynumberlist</code>	63
<code>\@glsxtr@current@style</code>	20, 188	<code>\@glsxtr@noidx@numberlistloop</code>	63
<code>\@glsxtr@currentunitcount</code>	54, 56	<code>\@glsxtr@notfoundinlist</code>	103
<code>\@glsxtr@currunitcount</code>	55, 57	<code>\@glsxtr@optlist</code>	19
<code>\@glsxtr@declareoption</code>	4, 7, 9	<code>\@glsxtr@org@Glsxtrtitlefirst</code> ..	154, 155
<code>\@glsxtr@defaultnoglossarywarning</code> ..	10	<code>\@glsxtr@org@Glsxtrtitlefirstplural</code>	154, 155
<code>\@glsxtr@deprecated@abbrstyle</code>	138, 139, 142, 143, 147–149	<code>\@glsxtr@org@Glsxtrtitlefull</code> ..	154, 155
<code>\@glsxtr@disabledflycommand</code>	19	<code>\@glsxtr@org@Glsxtrtitlefullpl</code> ..	154, 155
<code>\@glsxtr@do@wrindex</code>	40	<code>\@glsxtr@org@Glsxtrtitlelong</code> ..	154, 155
<code>\@glsxtr@do@gl:disablehyperinlist</code> ..	39	<code>\@glsxtr@org@Glsxtrtitlelongpl</code> ..	154, 155
<code>\@glsxtr@do@style</code>	11, 170	<code>\@glsxtr@org@Glsxtrtitleplural</code> ..	154, 155
<code>\@glsxtr@do@titlecaps@warn</code> ...	87–90, 98	<code>\@glsxtr@org@Glsxtrtitleshort</code> ..	154, 155
<code>\@glsxtr@do@abbreviationsdef</code>	7	<code>\@glsxtr@org@Glsxtrtitleshortpl</code> ..	154, 155
<code>\@glsxtr@do@accsupp</code>	9, 11	<code>\@glsxtr@org@Glsxtrtitletext</code> ..	154, 155
<code>\@glsxtr@docdefval</code>	5, 6, 16	<code>\@glsxtr@org@MakeUppercase</code>	154, 155
<code>\@glsxtr@doloctag</code>	22, 23		
<code>\@glsxtr@dostylewarn</code>	120, 121		

<code>\@glxtr@org@checkfirsthyper</code>	38, 61	<code>\@glxtr@usesee</code>	14
<code>\@glxtr@org@delimN</code>	23	<code>\@glxtrdocdeffalse</code>	16
<code>\@glxtr@org@delimR</code>	23	<code>\@glxtrindexcrossrefsfalse</code>	6
<code>\@glxtr@org@glignore</code>	23	<code>\@glxtrindexcrossrefstrue</code>	6
<code>\@glxtr@org@glxtrtitlefirst</code>	154, 155	<code>\@glxtrlong</code>	42, 113
<code>\@glxtr@org@glxtrtitlefirstplural</code>	154, 155	<code>\@glxtrlongpl</code>	42, 116, 117
<code>\@glxtr@org@glxtrtitlefull</code>	154, 155	<code>\@glxtrpl</code>	18, 19
<code>\@glxtr@org@glxtrtitlefullpl</code>	154, 155	<code>\@glxtrpostloctag</code>	22, 24
<code>\@glxtr@org@glxtrtitlelong</code>	154, 155	<code>\@glxtrpreloctag</code>	22, 24
<code>\@glxtr@org@glxtrtitlelongpl</code>	154, 155	<code>\@glxtrshort</code>	42, 112
<code>\@glxtr@org@glxtrtitleplural</code>	154, 155	<code>\@glxtrshortpl</code>	42, 115
<code>\@glxtr@org@glxtrtitleshort</code>	154, 155	<code>\@glxtrundeftag</code>	5, 12
<code>\@glxtr@org@glxtrtitleshortpl</code>	154, 155	<code>\@gobble</code>	7, 104
<code>\@glxtr@org@glxtrtitletext</code>	154, 155	<code>\@gobbletwo</code>	104
<code>\@glxtr@org@makeglossaries</code>	61, 62	<code>\@ifnextchar</code>	41
<code>\@glxtr@org@markboth</code>	153, 154	<code>\@ifpackageloaded</code>	4, 7, 70, 87, 88, 91, 92, 170
<code>\@glxtr@org@markright</code>	153, 154	<code>\@ifstar</code>	17, 41, 97
<code>\@glxtr@org@newacronymstyle</code>	60, 61	<code>\@ifundefined</code>	170
<code>\@glxtr@org@postdescription</code>	99	<code>\@input@</code>	66
<code>\@glxtr@org@setacronymstyle</code>	60, 61	<code>\@istfilename</code>	62
<code>\@glxtr@org@printglossary</code>	19	<code>\@makeglossary</code>	62
<code>\@glxtr@org@warndep</code>	104	<code>\@mfu@domakefirstuc</code>	98, 99
<code>\@glxtr@p@acrlong@</code>	43	<code>\@mfu@nocaplist</code>	99
<code>\@glxtr@p@acrlongpl@</code>	43	<code>\@one</code>	49, 58
<code>\@glxtr@p@acrshort@</code>	42	<code>\@newglossaryentry@defcounters</code>	47, 55
<code>\@glxtr@p@acrshortpl@</code>	42	<code>\@newglossaryentryposthook</code>	37
<code>\@glxtr@p@long@</code>	42	<code>\@newglossaryentryprehook</code>	37
<code>\@glxtr@p@longpl@</code>	42	<code>\@nnil</code>	94, 96, 97, 103, 104
<code>\@glxtr@p@plural@</code>	42	<code>\@no@makeglossaries</code>	70
<code>\@glxtr@p@short@</code>	42	<code>\@onelevel@sanitize</code>	19
<code>\@glxtr@p@shortpl@</code>	42	<code>\@onlypreamble</code>	20, 23, 57, 93, 95–97
<code>\@glxtr@p@text@</code>	42	<code>\@printgloss@setsort</code>	63
<code>\@glxtr@pagetag</code>	22, 23	<code>\@printglossary</code>	19
<code>\@glxtr@pagetag</code>	22, 23	<code>\@sGlsXtrEnableOnTheFly</code>	17
<code>\@glxtr@prevunitcount</code>	55	<code>\@secondofthree</code>	26, 27, 33–37, 109, 111, 112, 114, 116, 117
<code>\@glxtr@redefstyles</code>	10, 170	<code>\@secondoftwo</code>	25, 28–36, 38, 42, 103, 109, 110, 112–118, 129
<code>\@glxtr@reg@glosslist</code>	62–64	<code>\@thirdofthree</code>	26–28, 33–36, 38, 110, 111, 113, 114, 116, 118
<code>\@glxtr@savepreloctag</code>	22–24	<code>\@thirdoftwo</code>	28–32
<code>\@glxtr@setentrycountunsetattr</code>	47	<code>\@warn@nomakeglossaries</code>	66
<code>\@glxtr@setentryunitcountunsetattr</code>	58	<code>\@xdy@main@language</code>	66
<code>\@glxtr@setupshortcuts</code>	8, 9, 11	<code>\@xdy@language</code>	66
<code>\@glxtr@swaptwo</code>	103		
<code>\@glxtr@tag</code>	98		
<code>\@glxtr@taggingcs</code>	98		
<code>\@glxtr@thisloctag</code>	23		
<code>\@glxtr@tmp</code>	10		
<code>\@glxtr@type</code>	86		
<code>\@glxtr@unitcountlist</code>	53		

A

<code>\AB</code>	8
------------------	---

<code>\Ab</code>	7		
<code>\ab</code>	7		
abbreviation styles:			
long-noshort	147, 148		
short	132		
<code>\abbreviationsname</code>	7		
<code>\abbrvpluralsuffix</code>	105, 106, 124,		
	126, 128, 130–132, 134, 136–143, 151, 152		
<code>\ABP</code>	8		
<code>\Abp</code>	7		
<code>\abp</code>	7		
<code>\ACRfullfmt</code>	60		
<code>\Acrfullfmt</code>	60		
<code>\acrfullfmt</code>	60		
<code>\ACRfullplfmt</code>	60		
<code>\Acrfullplfmt</code>	60		
<code>\acrfullplfmt</code>	60		
<code>\acronymentry</code>	59		
<code>\acronymfont</code>	32–36, 45, 60, 61		
<code>\acronymname</code>	7		
<code>\acronymsort</code>	59		
<code>\acronymtype</code>	7, 59, 60		
<code>\acrpluralsuffix</code>	59		
<code>\actualchar</code>	96		
<code>\addtolength</code>	187		
<code>\advance</code>	49, 58		
<code>\AF</code>	8		
<code>\Af</code>	8		
<code>\af</code>	7		
<code>\AFP</code>	8		
<code>\Afp</code>	8		
<code>\afp</code>	7		
<code>\AL</code>	8		
<code>\Al</code>	8		
<code>\al</code>	7		
<code>\ALP</code>	8		
<code>\Alp</code>	8		
<code>\alp</code>	7		
<code>\AnyTrackedLanguages</code>	170		
<code>\appto</code> ...	10, 14, 36, 37, 40, 47, 55, 93, 102, 105		
<code>\AS</code>	8		
<code>\As</code>	8		
<code>\as</code>	7		
<code>\ASP</code>	8		
<code>\Asp</code>	8		
<code>\asp</code>	7		
<code>\AtBeginDocument</code>	12, 21		
<code>\AtEndDocument</code>	15, 48, 57, 66		
		B	
		<code>babel package</code>	93, 95, 102
		<code>\begin</code>	51, 68, 172–176, 186
		C	
		category attributes:	
		discardperiod	101
		entrycount	46, 47, 49, 58
		firstuc	90
		glossdesc	87
		glossdescfont	87
		glossname	88
		glossnamefont	88, 91
		headuc	155
		indexname	93
		indexonlyfirst	40
		insertdots	105
		nohyper	39
		nohyperfirst	127
		regular	24, 52, 123–
			125, 127, 129, 132–134, 144, 146, 150, 152
		<code>\cGLS</code>	8, 47, 58
		<code>\cGLs</code>	7, 47, 58
		<code>\cglS</code>	7, 47, 58
		<code>\cGLSformat</code>	51
		<code>\cGLsformat</code>	50
		<code>\cglSformat</code>	50, 52
		<code>\cGLSpl</code>	8, 47, 58
		<code>\cGlSpl</code>	7, 47, 58
		<code>\cglSpl</code>	7, 47, 58
		<code>\cGLSplformat</code>	51
		<code>\cGlSplformat</code>	50
		<code>\cglSplformat</code>	50, 52
		<code>\columnwidth</code>	21
		<code>\count@</code>	49, 57, 58
		<code>\cs</code>	51, 186
		<code>\csdef</code> 36–38, 48, 53, 54, 56, 81, 121, 122, 129, 172	
		<code>\csedef</code>	54
		<code>\csgdef</code>	16, 22, 48, 54, 56, 57
		<code>\csletcs</code>	122
		<code>\csname</code>	20, 24, 32–38,
			54, 66, 69, 70, 86, 104, 109–118, 123, 186
		<code>\csuse</code>	23, 37,
			38, 53–57, 81, 92, 100, 121, 122, 179–182
		<code>\csxdef</code>	14, 54, 57
		<code>\CurrentOption</code>	11, 171
		<code>\CurrentTrackedTag</code>	170
		<code>\CustomAbbreviationFields</code>	
			106, 123, 125–127, 129,
			131, 132, 134, 135, 144, 145, 147, 150, 151

D	
<code>\DeclareAcronymList</code>	59
<code>\DeclareOption</code>	4, 171
<code>\DeclareOptionX</code>	4, 11
<code>\def</code>	12, 14, 15, 17–19, 22, 24–36, 41–45, 50–52, 59, 62, 64, 92–99, 102–106, 109–118, 120, 186, 187
<code>\define@boolkey</code>	6, 39
<code>\define@choicekey</code>	5, 6, 8, 10
<code>\define@key</code>	10, 11, 36, 104
<code>\DefineAcronymSynonyms</code>	9
<code>\delimN</code>	23
<code>\delimR</code>	23
<code>\detokenize</code>	17
<code>\dimen@</code>	61, 179–186
<code>\dimen@i</code>	180–182
<code>\dimen@ii</code>	180–182
<code>\dimexpr</code>	21, 178
<code>\disable@keys</code>	7, 12, 16
<code>\do</code>	10, 15, 47, 58, 62, 64, 86, 98
<code>\do@glsl@link@checkfirsthyper</code>	25, 32–36, 109–118
<code>\do@glsl:disablehyperinlist</code>	39
<code>doc package</code>	96
<code>\DTLifinlist</code>	62, 63
E	
<code>\eappto</code>	10, 93, 171
<code>\edef</code>	38, 53, 54, 56, 62, 63, 66, 87, 88, 90–92, 94, 96, 104, 180–182
<code>\eglssetwidest</code>	179–186
<code>\else</code>	6, 7, 9, 10, 16, 17, 22, 24, 40, 49, 61, 63, 67–69, 93, 94, 96, 103–105, 112–118, 124, 126, 128, 130–135, 151, 152, 173–177, 187, 188
<code>\emph</code>	143
<code>\encapchar</code>	96
<code>\end</code>	68, 172–176, 186
<code>\endcsname</code>	20, 24, 32–38, 54, 66, 69, 70, 86, 104, 109–118, 123, 186
entry categories:	
abbreviation	118
general	81, 83
index	85
<code>\epreto</code>	93
<code>\equal</code>	69
<code>etoolbox package</code>	4
<code>\expandafter</code>	11, 14, 15, 17–19, 37, 38, 41, 52, 53, 62, 63, 86, 89, 90, 93, 96, 103, 105, 106
<code>\expandonce</code>	59, 94
F	
<code>\fi</code>	5–7, 9, 10, 15–17, 20– 22, 24, 40, 49, 57, 58, 61, 63, 64, 66–70, 93, 94, 97, 103, 105, 112–118, 124, 126, 128, 130–135, 151, 152, 173–177, 179–188
first use	189
flag	189
text	189
<code>\firstacronymfont</code>	60, 61
<code>\footnote</code>	127–129
<code>\forall glossaries</code>	15, 84, 86, 179–185
<code>\forall glsentries</code>	49, 57
<code>\ForEachTrackedDialect</code>	170
<code>\forall glsentries</code>	15, 84, 86, 179–185
<code>\forall listcsloop</code>	58
<code>\forall listloop</code>	64, 65, 99
<code>\futurelet</code>	102
G	
<code>\gdef</code>	23, 95, 96
<code>\Genacrfullformat</code>	60
<code>\genacrfullformat</code>	60
<code>\GenericAcronymFields</code>	59
<code>\Genplacrfullformat</code>	60
<code>\genplacrfullformat</code>	60
<code>\glo@name</code>	89, 90
glossaries package	171
glossaries-accsupp package	9, 11, 70
glossaries-extra package	2
glossaries-extra-stylemods package ..	10, 99, 170
<code>\GlossariesExtraWarning</code>	5, 7, 17, 19, 60, 68, 87–92, 98, 122
<code>\GlossariesExtraWarningNoLine</code> ..	7, 49, 58
<code>\GlossariesWarning</code>	22, 64, 65, 120
<code>\GlossariesWarningNoLine</code>	62, 66
glossary styles:	
alttree	177, 178, 186
inline	177
listdotted	172
listdottedstyle	172
sublistdotted	172
glossary-long package	173
glossary-longbooktabs package	173
glossary-mcols package	171
glossary-tree package	171
<code>\glossaryentrynumbers</code>	24
<code>\glossaryheader</code>	172–176, 187
<code>\glossarysection</code>	69

<code>\glossarytitle</code>	69	<code>\Glsaccessshort</code> 33, 112, 119, 126, 128, 130, 133, 152
<code>\glossarytoctitle</code>	69	<code>\glsaccessshort</code> 32, 33, 107, 112, 113, 119, 124, 126, 128, 130–134, 151, 152
<code>\glossentry</code>	172–176, 187	<code>\Glsaccessshortpl</code> 34, 116, 118, 126, 128, 130, 133, 152
<code>\glossentrydesc</code>	172–178	<code>\glsaccessshortpl</code>	. 33, 34, 107, 115, 116, 118, 119, 124, 126, 128, 130–134, 151, 152
<code>\glossentryname</code>	172–176, 187	<code>\GlsAccesssymbol</code> 30
<code>\glossentrysymbol</code>	173–178	<code>\Glsaccesssymbol</code> 29, 97
<code>\GLS</code>	47, 58	<code>\glsaccesssymbol</code> 29, 97, 101
<code>\Gls</code>	18, 47, 58	<code>\GlsAccesssymbolplural</code> 30
<code>\gls</code>	18, 20, 47, 58, 67	<code>\Glsaccesssymbolplural</code> 30
<code>\gls@assign@field</code>	37	<code>\glsaccesssymbolplural</code> 30
<code>\gls@checkseallowed</code>	16, 17, 62	<code>\GLSaccessstext</code> 26
<code>\gls@codepage</code>	66	<code>\Glsaccessstext</code> 26
<code>\gls@defdocnewglossaryentry</code>	47, 55	<code>\glsaccessstext</code> 26
<code>\gls@defglossaryentry</code>	18, 19	<code>\glsacrshortcutstrue</code> 9
<code>\gls@save@numberlist</code>	22, 24	<code>\glsacspacemax</code> 61
<code>\gls@tmplen</code>	179–185, 187	<code>\glsadd</code> 15, 67
<code>\glsabbrvdefaultfont</code> 108, 124, 126, 128, 130–132, 134	<code>\glsaddstoragekey</code> 81
<code>\glsabbrvfont</code> 143–149	<code>\glsbackslash</code> 17
<code>\glsabbrvfont</code> 43, 44, 60, 108, 112, 113, 115, 116, 118, 119, 123–132, 134–152	<code>\glsapscase</code> 25–38, 109–120
<code>\glsabbrvuserfont</code> 150–152	<code>\glsacategory</code> 24, 25, 38, 43, 44, 82–84, 87, 88, 90–92, 97, 100, 109–113, 115, 116
<code>\glsabbrvfont</code> 124, 125, 127, 129, 144, 145, 150, 152	<code>\glsacategorylabel</code> 38, 39, 104–106, 129
<code>\GLSAccessdesc</code> 29	<code>\glsaclosebrace</code> 68, 69
<code>\Glsaccessdesc</code> 28, 87, 97	<code>\glsacurrententrylabel</code>	.. 22, 23, 92, 99, 100
<code>\glsaccessdesc</code> 28, 87, 101	<code>\glsacurrentfieldvalue</code> 149
<code>\GLSAccessdescplural</code> 29	<code>\glsacustomtext</code> 25, 32–36, 109–118, 120
<code>\Glsaccessdescplural</code> 29	<code>\glsadefaulttype</code> 7, 67
<code>\glsaccessdescplural</code> 29	<code>\glsadescriptionaccessdisplay</code>	.. 74, 75, 87
<code>\GLSAccessfirst</code> 26	<code>\glsadescriptionpluralaccessdisplay</code>	. 75
<code>\Glsaccessfirst</code> 26	<code>\glsadescwidth</code> 172, 174–176
<code>\glsaccessfirst</code> 26	<code>\glsadetoklabel</code> 12–17, 48, 53–58, 64, 65, 86, 89, 90, 180–182
<code>\GLSAccessfirstplural</code> 28	<code>\glsdisplaynumberlist</code> 62, 64
<code>\Glsaccessfirstplural</code> 27	<code>\glsdohyperlink</code> 42
<code>\glsaccessfirstplural</code> 27	<code>\glsdoifexists</code>	6, 14, 25, 32–36, 64, 65, 109–118
<code>\Glsaccesslong</code>	35, 107, 114, 124, 131, 134, 151	<code>\glsdoifexistsorwarn</code> 6, 87, 97
<code>\glsaccesslong</code> 34, 35, 107, 113, 114, 124, 126, 128, 130, 131, 133–135, 151, 152	<code>\glsdonohyperlink</code> 41, 42
<code>\Glsaccesslongpl</code> 36, 107, 117, 124, 131, 134, 151	<code>\glsdosanitizesort</code> 63
<code>\glsaccesslongpl</code>	.. 35, 36, 107, 117, 118, 124, 126, 128, 130, 131, 133–135, 151, 152	<code>\glsenableentrycount</code> 47, 49, 57
<code>\GLSAccessname</code> 28	<code>\glsenableentryunitcount</code> 48, 58
<code>\Glsaccessname</code> 28	<code>\glsentrycurrcount</code> 48, 49, 55
<code>\glsaccessname</code> 28	<code>\Glsentrydesc</code> 75, 79, 88
<code>\GLSAccessplural</code> 27	<code>\glsentrydesc</code> 74, 75, 79, 88
<code>\Glsaccessplural</code> 27	<code>\Glsentrydescplural</code> 75, 80
<code>\glsaccessplural</code> 27	<code>\glsentrydescplural</code> 75, 79, 80

<code>\Glsentryfirst</code>	52, 72, 78	<code>\glsfirstabbrvdefaultfont</code>	
<code>\glsentryfirst</code>	52, 72, 78, 166	108, 124, 126, 128, 130–132, 134
<code>\Glsentryfirstplural</code>	53, 73, 79	<code>\glsfirstabbrvemfont</code>	144–149
<code>\glsentryfirstplural</code> ...	52, 73, 78, 79, 167	<code>\glsfirstabbrvfont</code>	60, 107, 123–134, 136–152
<code>\Glsentryfull</code>	60	<code>\glsfirstabbrvuserfont</code>	151, 152
<code>\glsentryfull</code>	60	<code>\glsfirstaccessdisplay</code>	72
<code>\Glsentryfullpl</code>	60	<code>\glsfirstlongdefaultfont</code>	
<code>\glsentryfullpl</code>	60	124, 126, 131, 133, 134
<code>\glsentryitem</code>	172–176, 187	<code>\glsfirstlongemfont</code>	144–146, 148
<code>\Glsentrylong</code>	44, 45, 52, 77, 80	<code>\glsfirstlongfont</code>	
<code>\glsentrylong</code>	44, 45, 52, 77, 80, 129, 167, 168	107, 123–131, 133–135, 144–148, 150–152
<code>\Glsentrylongpl</code>	44, 45, 53, 77, 80	<code>\glsfirstlongfootnotefont</code>	128–130
<code>\glsentrylongpl</code> ...	44, 45, 52, 77, 80, 81, 168	<code>\glsfirstlonguserfont</code>	151, 152
<code>\Glsentryname</code>	70, 78, 89–92	<code>\GLSfirstplural</code>	160
<code>\glsentryname</code>	70, 71, 78, 93, 179–186	<code>\Glsfirstplural</code>	160
<code>\glsentrynumberlist</code>	63, 65, 184–186	<code>\glsfirstplural</code>	160
<code>\Glsentryplural</code>	72, 78	<code>\glsfirstpluralaccessdisplay</code>	73
<code>\glsentryplural</code>	71, 72, 78, 166	<code>\glsforeachincategory</code>	120
<code>\glsentryprevcount</code>	48, 49, 55	<code>\glsenentryfmt</code>	24
<code>\glsentryprevmaxcount</code>	56	<code>\glsgetattribute</code> ...	49, 53–55, 87, 88, 90–93
<code>\glsentryprevtotalcount</code>	55	<code>\glsgetcategoryattribute</code>	82
<code>\Glsentryshort</code>	43, 45, 76, 80	<code>\glsgetwidestname</code>	178
<code>\glsentryshort</code> ...	43–45, 61, 75, 76, 80, 164	<code>\glsgroupheading</code>	172–176, 187
<code>\Glsentryshortpl</code>	44, 45, 76, 80	<code>\glsgroupskip</code>	173–177, 188
<code>\glsentryshortpl</code> ...	44, 45, 76, 80, 164, 165	<code>\glshasattribute</code>	49, 54, 56, 58, 87, 88, 90,
<code>\Glsentrysymbol</code>	73, 79	91, 93, 124, 125, 127, 129, 144, 146, 150, 152
<code>\glsentrysymbol</code>	73, 74, 79, 183, 184	<code>\glshascategoryattribute</code>	82
<code>\Glsentrysymbolplural</code>	74, 79	<code>\glshypernumber</code>	93
<code>\glsentrysymbolplural</code>	74, 79	<code>\glsifattribute</code>	
<code>\Glsentrytext</code>	71, 78	..	39, 40, 84, 87–90, 99, 101, 102, 155–163
<code>\glsentrytext</code>	71, 78, 165	<code>\glsifcategory</code>	84
<code>\Glsentryuseri</code>	30	<code>\glsifcategoryattribute</code> ..	39, 83, 105, 106
<code>\glsentryuseri</code>	31	<code>\glsifnotregular</code>	25
<code>\Glsentryuserii</code>	31	<code>\glsifnotregularcategory</code>	84
<code>\glsentryuserii</code>	31	<code>\glsifplural</code>	
<code>\Glsentryuseriii</code>	31	25, 27–30, 32–36, 101, 102, 109–119
<code>\glsentryuseriii</code>	31	<code>\glsifregular</code>	24, 25, 52, 53
<code>\Glsentryuseriv</code>	31	<code>\glsifregularcategory</code>	83
<code>\glsentryuseriv</code>	31	<code>\glsignore</code>	23
<code>\Glsentryuserv</code>	32	<code>\glsinlinedescformat</code>	177
<code>\glsentryuserv</code>	32	<code>\glsinlinesubdescformat</code>	177
<code>\Glsentryuservi</code>	32	<code>\glsinsert</code>	25, 32–36, 109–120
<code>\glsentryuservi</code>	32	<code>\glskeylisttok</code>	59, 105, 106
<code>\glsfieldxdef</code>	86	<code>\glslabel</code> ..	24, 38, 39, 100, 101, 118–120, 129
<code>\glsfindwidesttoplevelname</code>	179	<code>\glslabeltok</code>	59, 105, 106, 123–127,
<code>\GLSfirst</code>	159	129, 131, 132, 134, 135, 144–147, 150–152
<code>\Glsfirst</code>	159	<code>\glsletentryfield</code>	93
<code>\glsfirst</code>	159	<code>\glslink</code>	60

<code>\glslink options</code>		<code>\glssetabbrvfmt</code>	24, 25,
<code>format</code>	92		43, 44, 87, 88, 90, 91, 97, 109–113, 115, 116
<code>noindex</code>	39	<code>\glssetattribute</code>	124, 125, 127, 129,
<code>\glslinkcheckfirsthyperhook</code>	39		131, 132, 134, 135, 144, 146, 147, 150, 152
<code>\glslinkvar</code>	41	<code>\glssetcategoryattribute</code>	
<code>\glslistdottedwidth</code>	172		47, 59, 61, 82, 83, 85, 86, 98
<code>\glslongaccessdisplay</code>	77	<code>\glsshortaccessdisplay</code>	75, 76
<code>\glslongdefaultfont</code>		<code>\glsshortpltok</code>	106,
	108, 124, 126, 127, 131, 133, 134		124–127, 129, 131, 132, 144, 145, 150, 152
<code>\glslongemfont</code>	143–148	<code>\glsshortpluralaccessdisplay</code>	76
<code>\glslongfont</code>	44, 45,	<code>\glsshorttok</code>	59, 105, 106, 123, 125–127,
	108, 113, 114, 117, 118, 124, 126, 128,		129, 131, 132, 135, 144, 145, 147, 150, 151
	130, 131, 133, 134, 144–146, 148, 151, 152	<code>\glssubentryitem</code>	172–177, 187
<code>\glslongfootnotefont</code>	127, 128, 130	<code>\glssymbolaccessdisplay</code>	73, 74
<code>\glslongpltok</code>	106, 124–	<code>\glssymbolpluralaccessdisplay</code>	74
	127, 129, 134, 135, 144, 145, 147, 150, 152	<code>\glstarget</code>	172–177, 187
<code>\glslongpluralaccessdisplay</code>	77	<code>\GLStext</code>	157, 158
<code>\glslongtok</code>	59, 105, 106, 123–127, 129,	<code>\Glstext</code>	158
	131, 132, 134, 135, 144, 145, 147, 150, 151	<code>\glstext</code>	157
<code>\glslonguserfont</code>	150–152	<code>\glstextaccessdisplay</code>	71
<code>\glsnameaccessdisplay</code>	70, 71, 89, 91	<code>\glstextup</code>	136
<code>\glsnamefont</code>	89–92	<code>\glstreeindent</code>	186, 187
<code>\glsnoidxdisplayloc</code>	65	<code>\glstreenamebox</code>	187
<code>\glsnoidxdisplayloclisthandler</code>	64	<code>\glstreenamefmt</code>	178–187
<code>\glsnoidxloclist</code>	65	<code>\glstype</code>	32–36, 109–118
<code>\glsnoidxnumberlistloophandler</code>	65	<code>\glunset</code>	15, 50, 51
<code>\glsnonumberlistfalse</code>	22	<code>\glswrite</code>	62
<code>\glsnonumberlisttrue</code>	22	<code>\Glsxtr</code>	19
<code>\glsnumberlistloop</code>	63	<code>\glxtr</code>	19
<code>\glsnumlistlastsep</code>	64	<code>\glxtr@addunused</code>	15
<code>\glsnumlistsep</code>	64	<code>\glxtr@applyabbrvfmt</code>	118
<code>\glsopenbrace</code>	68, 69	<code>\glxtr@applyabbrvstyle</code>	104, 105, 121
<code>\glsorder</code>	62	<code>\glxtr@doption</code>	4, 7, 11
<code>\glspagelistwidth</code>	172, 174–176	<code>\glxtr@ifnextpunc</code>	102
<code>\GLSpl</code>	47, 58	<code>\glxtr@ifpunctoken</code>	102
<code>\Glspl</code>	19, 47, 58	<code>\glxtr@keylist</code>	18, 19
<code>\Glspl</code>	19, 47, 58	<code>\glxtr@next</code>	103
<code>\Glsplural</code>	158	<code>\glxtr@orgmakenoidxglossaries</code>	16
<code>\Glsplural</code>	158, 159	<code>\glxtr@punclist</code>	102, 103
<code>\glsplural</code>	158	<code>\glxtr@usesee</code>	14
<code>\glspluralaccessdisplay</code>	71, 72	<code>\glxtr@warnonexistsordo</code>	5, 13
<code>\glspluralsuffix</code>	105, 108,	<code>\glxtrabbrvtype</code>	7, 106
	124, 126, 128, 130–132, 134, 136, 140, 150	<code>\glxtraddallcrossrefs</code>	15
<code>\glspostdescription</code>	99, 172–178	<code>\glxtrAltTreeIndent</code>	178
<code>\glspostinline</code>	177	<code>\glxtralttreeInit</code>	186
<code>\glspostlinkhook</code>	25, 32–36, 109–118	<code>\glxtrAltTreePar</code>	177
<code>\glsprestandardsort</code>	63	<code>\glxtrAltTreeSetHangIndent</code>	178, 187
<code>\glsseeformat</code>	14, 65	<code>\glxtrAltTreeSetSubHangIndent</code>	187
		<code>\glxtralttreeSubSymbolDescLocation</code>	187

<code>\glxtralttreeSymbolDescLocation</code> ..	<code>\glxtrheadplural</code>	154
..... 178, 187	<code>\Glsxtrheadshort</code>	154
<code>\glxtrassignfieldfont</code>	<code>\glxtrheadshort</code>	154
26–32	<code>\Glsxtrheadshortpl</code>	154
<code>\glxtrcat</code>	<code>\glxtrheadshortpl</code>	154
18, 19	<code>\Glsxtrheadtext</code>	154
<code>\glxtrComputeTreeIndent</code>	<code>\glxtrheadtext</code>	154
187	<code>\glxtrtrifcounttrigger</code>	50, 51
<code>\glxtrComputeTreeSubIndent</code>	<code>\glxtrtrifemptyglossary</code>	69
187	<code>\glxtrtrifindexing</code>	40
<code>\GlsXtrDefineAbbreviationShortcuts</code> ..	<code>\glxtrtrifnextpunc</code>	102, 103
9	<code>\glxtrtrifperiod</code>	101, 102
<code>\GlsXtrDefineOtherShortcuts</code>	<code>\glxtrtrifwasfirstuse</code>	
9	. 25–28, 32–36, 38, 101, 109, 112–118, 129	
<code>\glxtrdiscardperiod</code>	<code>\Glsxtrinlinefullformat</code>	108,
100	109, 121, 122, 128, 130, 131, 133, 134, 169	
<code>\glxtrdoautoindexname</code>	<code>\glxtrinlinefullformat</code>	
40, 92 108–110, 121, 122, 128, 130–134, 168	
<code>\glxtrdopostpunc</code>	<code>\Glsxtrinlinefullplformat</code>	108,
129	111, 121, 122, 128, 130, 131, 133, 134, 169	
<code>\glxtrdowrglossaryhook</code>	<code>\glxtrinlinefullplformat</code> 107, 108, 110,	
40	111, 121, 122, 128, 130, 131, 133, 134, 169	
<code>\GlsXtrEnableEntryCounting</code>	<code>\glxtrininsertinsidefalse</code>	123
58	<code>\GLSxtrlong</code>	8, 160, 161
<code>\GlsXtrEnableEntryUnitCounting</code>	<code>\Glsxtrlong</code>	8, 161
47	<code>\glxtrlong</code>	7, 160, 161
<code>\GlsXtrEnableOnTheFly</code>	<code>\GLSxtrlongpl</code>	8, 161, 162
17, 20	<code>\Glsxtrlongpl</code>	8, 162
<code>\glxtrfieldtitlecase</code>	<code>\glxtrlongpl</code>	7, 161
87–90	<code>\glxtrlongshortdescsort</code>	125
<code>\glxtrfirstscfont</code>	<code>\glxtrmarkhook</code>	153
136–139	<code>\glxtrnewabbrevpresetkeyhook</code>	106
<code>\glxtrfirstsmfont</code>	<code>\glxtrnewnumber</code>	8
140–143	<code>\glxtrnewsymbol</code>	8
<code>\GlsXtrFormatLocationList</code> 22, 24, 184–186	<code>\glxtrNoGlossaryWarning</code>	10, 66
<code>\GLSxtrfull</code>	<code>\GlsXtrNoGlsWarningAutoMake</code>	69
8, 162, 163	<code>\GlsXtrNoGlsWarningBuildInfo</code>	70
<code>\Glsxtrfull</code>	<code>\GlsXtrNoGlsWarningCheckFile</code>	69
8, 163	<code>\GlsXtrNoGlsWarningEmptyMain</code>	69
<code>\glxtrfull</code>	<code>\GlsXtrNoGlsWarningEmptyNotMain</code>	69
7, 162	<code>\GlsXtrNoGlsWarningEmptyStart</code>	69
<code>\Glsxtrfullformat</code>	<code>\GlsXtrNoGlsWarningHead</code>	69
107, 120–122,	<code>\GlsXtrNoGlsWarningMisMatch</code>	70
124, 126, 128, 130, 132, 133, 135, 151, 152	<code>\GlsXtrNoGlsWarningNoOut</code>	70
<code>\glxtrfullformat</code>	<code>\GlsXtrNoGlsWarningTail</code>	70
107, 120–	<code>\Glsxtrpl</code>	19
122, 124–126, 128, 130, 132–134, 151, 152	<code>\glxtrpl</code>	19
<code>\GLSxtrfullpl</code>	<code>\glxtrpostdescription</code>	85, 99, 177
8, 163	<code>\glxtrpostlink</code>	100
<code>\Glsxtrfullpl</code>		
8, 163, 164		
<code>\glxtrfullpl</code>		
7, 163		
<code>\Glsxtrfullplformat</code> . 107, 119, 121, 122,		
124, 126, 128, 130, 132, 133, 135, 151, 152		
<code>\glxtrfullplformat</code> 119, 121, 122,		
124, 126, 128, 130, 132, 133, 135, 151, 152		
<code>\glxtrfullsep</code>		
107, 123–		
126, 128, 130, 131, 133, 134, 144, 145, 149		
<code>\glxtrgenabbrvfmt</code>		
24		
<code>\Glsxtrheadfirst</code>		
154		
<code>\glxtrheadfirst</code>		
154		
<code>\Glsxtrheadfirstplural</code>		
154		
<code>\glxtrheadfirstplural</code>		
154		
<code>\Glsxtrheadfull</code>		
155		
<code>\glxtrheadfull</code>		
155		
<code>\Glsxtrheadfullpl</code>		
155		
<code>\glxtrheadfullpl</code>		
155		
<code>\Glsxtrheadlong</code>		
155		
<code>\glxtrheadlong</code>		
154		
<code>\Glsxtrheadlongpl</code>		
155		
<code>\glxtrheadlongpl</code>		
155		
<code>\Glsxtrheadplural</code>		
154		

<code>\ifglshasfield</code>	149	<code>\listcseadd</code>	54
<code>\ifglshaslong</code>	52, 53	<code>\listcsgadd</code>	16
<code>\ifglshasparent</code>	179–182	<code>\listcsxadd</code>	54
<code>\ifglshasshort</code>	24, 25	<code>\loadglsentries</code>	16, 67
<code>\ifglshassymbol</code>	101, 178		
<code>\ifglsindeonlyfirst</code>	40		
<code>\ifglsnogroupskip</code>	173–177, 188		
<code>\ifglsnonumberlist</code>	24		
<code>\ifglssanitizesort</code>	63		
<code>\ifglssused</code>	15,		
	38, 40, 49, 58, 118, 179, 180, 182, 184, 185		
<code>\ifglsxindy</code>	66–68		
<code>\ifglsxtrinsertinside</code>			
	112–118, 124, 126, 128, 130–135, 151, 152		
<code>\ifHy@hyperindex</code>	92		
<code>\ifinlistcs</code>	16		
<code>\ifKV@glslink@noindex</code>	40		
<code>\ifnum</code>	5, 6, 49, 57, 58, 187		
<code>\ifthenelse</code>	69		
<code>\IfTrackedLanguageFileExists</code>	170		
<code>\ifundef</code>	41, 62, 98		
<code>\ifx</code>	20, 22, 94, 96, 103, 104, 188		
<code>\immediate</code>	49, 57, 66		
<code>\index</code>	93		
<code>\indexspace</code>	188		
<code>\input</code>	170		
<code>\istfilename</code>	62		
<code>\item</code>	68, 172		
	J		
<code>\jobname</code>	66–70		
	K		
<code>\key@ifundefined</code>	36		
<code>\KV@glslink@hyperfalse</code>	39, 42		
<code>\KV@glslink@noindexfalse</code>	39		
<code>\KV@glslink@noindextrue</code>	42		
	L		
<code>\LaTeX</code>	67, 68		
<code>\leaders</code>	172		
<code>\let</code> ...	4, 7, 8, 11, 16, 17, 19–23, 25–39, 41–		
	43, 47, 48, 56–62, 64, 65, 87–94, 98, 99,		
	102–105, 109–118, 129, 153–155, 177, 179		
<code>\letabbreviationstyle</code>	129,		
	130, 132, 133, 135, 137, 138, 141, 146, 147		
<code>\letcs</code>	14, 15, 37, 64, 65, 87–92		
<code>\levelchar</code>	96		
<code>\listadd</code>	53		
<code>\listbreak</code>	98		
		M	
		<code>\MakeAcronymsAbbreviations</code>	61
		<code>\makeatletter</code>	66, 95
		<code>\makeatother</code>	95
		<code>\makebox</code>	172, 187
		<code>\makefirsttuc</code>	99
		<code>\makeglossaries</code>	61, 66, 68–70
		<code>\makeglossary</code>	62
		<code>makeindex</code>	189
		<code>makeindex</code>	61
		<code>\makenoidxglossaries</code>	68
		<code>\MakeTextUppercase</code>	154
		<code>\MakeUppercase</code>	154, 155
		<code>\marg</code>	186
		<code>\markboth</code>	154
		<code>\markright</code>	154
		<code>\maxdimen</code>	21
		<code>\mbox</code>	187
		<code>\medskip</code>	69
		<code>\MessageBreak</code>	16, 20, 49, 58, 63, 120, 121
		<code>mfistuc package</code>	98
		<code>\mfistucMakeUppercase</code>	
			26–36, 38, 44, 45, 52, 60, 71–
			81, 89, 90, 110, 111, 113, 114, 116, 118–120
		<code>\mfu@checkword@arg</code>	98, 99
		<code>\mfu@checkword@do</code>	99
		N	
		<code>\NeedsTeXFormat</code>	4, 171
		<code>\new@glossaryentry</code>	16, 63
		<code>\new@ifnextchar</code> .	37, 38, 51, 52, 102, 108–117
		<code>\newabbr</code>	8
		<code>\newabbreviation</code>	8, 60
		<code>\newabbreviationhook</code>	106
		<code>\newabbreviationstyle</code>	
			123, 125–127, 129, 131, 132, 134–152
		<code>\newacronym</code>	59, 60
		<code>\newacronymhook</code>	59
		<code>\newacronymstyle</code>	60, 61
		<code>\newcommand</code> ...	4–15, 17–20, 22–25, 37–42,
			46–49, 51–58, 60, 61, 64, 65, 67–86, 92–
			118, 120–123, 125, 127, 135, 136, 139,
			140, 143, 149, 150, 154–171, 177–179, 186
		<code>\newcount</code>	5
		<code>\newentry</code>	8

<code>\newglossary</code>	7, 62	<code>\ns@GLSxtrshortpl</code>	116
<code>\newglossaryentry</code>	8, 16, 47, 55, 59, 85, 86, 106	<code>\ns@Glsxtrshortpl</code>	115
<code>\newglossaryentry options</code>		<code>\ns@glxtrshortpl</code>	115
<code>desc</code>	74, 75, 79	<code>\null</code>	10
<code>descplural</code>	75, 79, 80	<code>\number</code>	54–57
<code>first</code>	41, 72, 78, 123, 159, 160, 166, 189	<code>\numexpr</code>	54, 57
<code>firstplural</code>	72, 73, 78, 79, 123, 159, 160, 167, 189		
<code>hyper</code>	153		
<code>long</code>	77, 80, 167		
<code>longplural</code>	77, 81, 168		
<code>name</code>	70, 71, 78, 93		
<code>noindex</code>	153		
<code>plural</code>	71, 72, 78, 123, 158, 159, 165		
<code>see</code>	6, 14, 16, 17, 62		
<code>short</code>	76, 80, 104		
<code>shortplural</code>	76, 80, 104		
<code>symbol</code>	73, 79		
<code>symbolplural</code>	74, 79		
<code>text</code>	41, 71, 78, 123, 157, 158, 165		
<code>\newif</code>	92, 123		
<code>\newlength</code>	178		
<code>\newnum</code>	8		
<code>\newrobustcmd</code>	37, 38,		
	51, 52, 98, 99, 108–117, 156–164, 179–185		
<code>\newsym</code>	8		
<code>\newterm</code>	85		
<code>\newtoks</code>	104		
<code>\newwrite</code>	62		
<code>\NoCaseChange</code>	155–163		
<code>\noexpand</code>	10, 59, 66, 94, 106, 171		
<code>\nofiles</code>	69		
<code>\noindent</code>	69		
<code>\nopostdesc</code>	18, 19, 85		
<code>\nr</code>	5, 6, 8, 10		
<code>\ns@GLSxtrfull</code>	110		
<code>\ns@Glsxtrfull</code>	109		
<code>\ns@glxtrfull</code>	108		
<code>\ns@GLSxtrfullpl</code>	111		
<code>\ns@Glsxtrfullpl</code>	110		
<code>\ns@glxtrfullpl</code>	110		
<code>\ns@GLSxtrlong</code>	114		
<code>\ns@Glsxtrlong</code>	114		
<code>\ns@glxtrlong</code>	113		
<code>\ns@GLSxtrlongpl</code>	117		
<code>\ns@Glsxtrlongpl</code>	117		
<code>\ns@glxtrlongpl</code>	116		
<code>\ns@GLSxtrshort</code>	113		
<code>\ns@Glsxtrshort</code>	112		
<code>\ns@glxtrshort</code>	111, 112		
		O	
		<code>\or</code>	5, 9, 16
		<code>\org@glossaryentrynumbers</code>	22
		P	
		<code>\p@glS@hyp@opt</code>	41
		package options:	
		abbreviations	7
		accsupp	9, 70
		acronym	7
		automake	64, 67
		docdef	5, 16, 47, 55
		false	16
		restricted	6
		true	16
		nonnumberlist	22
		numbers	8
		shortcuts	8
		all	8
		false	8
		none	8
		true	8
		style	11
		stylemods	11
		symbols	8, 85
		undefaction	12, 13
		warn	5
		<code>\PackageError</code>	5, 10, 16, 20, 37, 38,
			47, 48, 55, 57, 58, 60, 62, 63, 120–123, 171
		<code>\PackageWarning</code>	6
		<code>\PackageWarningNoLine</code>	6
		<code>\par</code>	69, 70, 177, 178, 187
		<code>\parindent</code>	178, 187
		<code>\PassOptionsToPackage</code>	4
		<code>\preto</code>	39
		<code>\printabbreviations</code>	7
		<code>\printglossaries</code>	62, 68
		<code>\printglossary</code>	7, 62, 68
		<code>\printglossary options</code>	
		nonnumberlist	24
		<code>\printnoidxglossaries</code>	68
		<code>\printnoidxglossary</code>	68
		<code>\printnumbers</code>	8, 85, 86

